

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Євсеєв С.П.
Мілов О.В.
Остапов С.Е.
Сєверінов О.В.

КІБЕРБЕЗПЕКА:
ОСНОВИ КОДУВАННЯ ТА КРИПТОГРАФІЇ

Навчальний посібник

Харків. Вид. “Новий Світ-2000”, 2023

Затверджено на засіданні вченої ради НТУ “ХПІ”, протокол № 9 від 24.11.2022 р.

Рецензенти:

докт. техн. наук, професор, завідувач кафедри “Захисту інформації” Національного університету “Львівська політехніка”, керівник Західного регіонального навчально-наукового центру захисту інформації, Заслужений винахідник України Дудикевич В. Б.;

докт. техн. наук, професор, Поліський національний університет, керівник навчально-наукового центру інформаційних технологій Молодецька К. В.

Кібербезпека: основи кодування та криптографії/ С.П. Євсєєв, О.В. Мілов, С.Е. Остапов, О.В. Северінов. – Харків: Вид. “Новий Світ-2000”, 2023. – 657 с. (Укр. мов.)

В навчальному посібнику розглянуто питання основ теорії інформації, формування дискретного каналу, основні критерії та показники теорії інформації та кодування, приведені основні підходи щодо побудови кодових систем на основі класичних завадостіких кодів, а також побудова кодових систем на основі алгеброгеометричних кодів, приведені алгоритми кодування та розкодування. Розглянути питання теорії захисту інформації, основи криптографії, розглянуті основні підходи побудови симетричних та несиметричних криптосистем, їх властивості. Окремо розглянути питання квантової механіки, та використання квантових алгоритмів, щодо їх використання в сучасних системах безпеки. Навчальний посібник призначений для аспірантів та студентів галузі знань 12 – Інформаційні технології, а також може бути корисним для студентів усіх інших спеціальностей та освітньо-професійних програм, в межах яких викладаються дисципліни циклу інформаційних технологій.

© Національний технічний університет
“Харківський політехнічний інститут”
© Євсєєв С.П., Мілов О.В., Остапов С.Е.,
Северінов О.В., 2023

Зміст

Вступ	8
Тема 1. Теорія інформації та кодування	10
1.1. Загальні поняття про системи передачі інформації	10
1.2. Кількість інформації та ентропія	16
1.2.1. Ентропія випадкового експерименту з рівноймовірними результатами	19
1.2.2. Ентропія випадкового експерименту з нерівноймовірними результатами	23
1.2.3. Умови максимізації ентропії джерела повідомлень	25
1.2.4. Ентропія складного експерименту при взаємозалежності результатів	28
1.2.5. Ентропія джерел дискретних повідомлень	35
1.2.6. Надмірність джерел повідомлень	38
1.2.7. Ентропія джерел неперервних повідомлень. Диференційна ентропія	39
1.2.8. Ентропія джерела з нормальним та рівномірним законом розподілу	41
1.3 Канали передачі інформації	45
1.3.1 Класифікація каналів передачі інформації	45
1.3.2. Модель дискретного каналу передачі інформації	46
1.3.3. Кількість інформації, що передається по дискретному каналу передачі інформації	51
1.3.4. Пропускна здібність дискретного каналу передачі інформації	57
1.4 Методи ефективного кодування інформації	61
1.4.1. Ефективне кодування Шеннона – Фано	63
1.4.2 Ефективне кодування Хаффмана	66
1.5. Основні визначення та поняття теорії скінчених полів	68
Галуа	
1.5.1. Групи	69
1.5.2. Підгрупи	70
1.5.3. Кільця	71
1.5.4. Підполе і суміжні класи	72
1.5.5. Скінченні поля, побудовані за кільцем цілих чисел	72
1.5.6. Скінченні поля, побудовані за кільцем многочленів	75

1.6. Алгебраїчна теорія блокових кодів	88
1.6.1. Призначення та класифікація завадостійких кодів	88
1.6.2. Загальні принципи завадостійкого кодування	91
1.6.3. Кодові межі блокових і безперервних кодів	100
1.6.4. Лінійні групові коди	112
1.6.5. Кодуючі і декокуючі пристрої лінійних групових кодів	121
1.6.6. Лінійні циклічні коди	123
1.6.7. Вимоги до породжувального многочлену циклічних кодів	132
1.6.8. Пристрої кодування та декодування циклічних кодів	136
1.6.9. Коди Боуза–Чоудхурі–Хоквінгема, коди Ріда–Соломона	149
1.6.10. Альтернантні коди, коди Гоппи	159
1.6.11. Алгеброгеометричні коди	166
1.7. Методи та алгоритми кодування алгеброгеометричних кодів	175
1.8. Методи та алгоритми декодування алгеброгеометричних кодів	181
1.8.1. Методи декодування недвійкових блокових кодів	181
1.8.2. Декодування алгебраїчних кодів методом Берлекемпа	186
1.8.3. Декодування двійкових кодів БЧХ	195
1.8.4. Декодування за допомогою алгоритму Евкліда	197
1.8.5. Алгебраїчне декодування алгеброгеометричних кодів	202
1.8.6. Переставне декодування алгеброгеометричних кодів	217
1.8.7. Мажоритарне декодування алгеброгеометричних кодів	221
1.8.8. Згортувальні коди	226
1.8.9. Декодування згортувальних кодів. Алгоритм Витербі	232
1.9. Багаторазова передача кодових комбінацій	236
1.10. Системи передачі інформації зі зворотним зв'язком	238
Контрольні запитання до теми 1	248
Використана література до теми 1	250

Тема 2. Криптографічні механізми захисту інформації в інформаційних системах	251
2.1. Математичні основи сучасної теорії захисту інформації	251
2.1.1. Математичні положення теорії скінченних полів та систем класів лишків	251
2.1.2. Математичні положення теорії чисел	263
2.1.3. Методи булевої алгебри, елементи кореляційного та спектрального аналізу	294
2.2. Компоненти криптосистеми та їх функціональні характеристики	300
2.3. Прості шифри	307
2.4. Криптографічні примітиви й типи структур симетричної криптосистеми	319
2.4.1. Основні криптографічні примітиви	319
2.4.2. Структура Файстеля алгоритму блокового симетричного шифрування	321
2.4.3. SPN-структура алгоритму блокового симетричного шифрування	323
2.4.4. Типи структур поточкових алгоритмів шифрування	324
2.4.5. Математичні моделі нелінійних вузлів замін у термінах булевої алгебри	326
2.5. Алгоритми блокового симетричного шифрування DES, ГОСТ-28147	331
2.5.1. Алгоритм блокового симетричного шифрування DES	331
2.5.2. Алгоритм блокового симетричного шифрування ГОСТ-28147	338
2.6. Типові режими роботи криптосистеми	340
2.6.1. Шифрування в режимі Electronic Code Book, ECB	341
2.6.2. Шифрування в режимі Cipher Block Changing, CBC	343
2.6.3. Шифрування в режимі Electronic Feedback, CFB	345
2.6.4. Шифрування в режимі Output Feedback, OFB	351
2.6.5. Шифрування в режимах удосконаленого OFB і PCBC	355
2.6.6. Інші режими шифрування	355
2.7. Атаки на блокові шифри	359
2.7.1. Диференціальний криптоаналіз	360

2.7.2. Лінійний криптоаналіз	366
2.7.3. Силова атака на основі розподілених розв'язань	369
2.8. Інші відомі блокові шифри	375
2.8.1. RC2	375
2.8.2. RC5	376
2.8.3. IDEA	377
2.8.4. Алгоритм SAFER	385
2.8.5. FEAL	385
2.8.6. Skipjack	386
2.8.7. Blowfish	386
2.8.8. REDOC	388
2.8.9. LOKI	388
2.8.10. Khufu	388
2.8.11. Khafre	389
2.8.12. Алгоритм Rijndael	389
2.9. Поточкові шифри	397
2.9.1. Регістри зсуву зі зворотнім зв'язком	398
2.9.2. Алгоритм A5	401
2.9.3. RC4	401
2.9.4. SEAL	402
2.10. Принципи побудови криптосистем з відкритим ключем	403
2.10.1. Криптосистема шифрування даних RSA	410
2.10.2. Криптосистема Ель Гамала	418
2.10.3. Комбінований метод шифрування	423
2.10.4. Керування криптографічними ключами	425
2.10.5. Механізми керування ключами на основі використання несиметричних методів (ISO/IEC 11770-3)	454
2.10.6. Безпека розподілу ключів	457
2.10.7. Криптографія на еліптичних кривих	459
2.11. Методи та алгоритми забезпечення автентичності та цілісності даних	483
2.11.1. Електронний підпис	484
2.11.2. MDC-коди	519
2.11.3. MAC-коди	533
Контрольні запитання до теми 2	551
Використана література до теми 2	552

Тема 3. Основи квантової криптографії	553
3.1. Небезпека криптографічних технологій	553
3.2. Фізичні основи квантових обчислень	555
3.3. Квантові біти	577
3.4. Квантові операції	581
3.5. Парадокс Ейнштейна-Подольські-Розена	591
3.6. Квантові алгоритми	598
3.6.1. Протокол квантового щільного кодування	602
3.6.2. Алгоритм Дойча	606
3.7. Квантова криптографія	609
3.7.1. Алгоритм факторизації цілих чисел (алгоритм Шора)	609
3.7.2. Алгоритм пошуку у невідсортованому масиві (алгоритм Гровера)	615
3.7.3. Квантові протоколи узгодження ключів	619
3.7.4. Квантовий протокол узгодження криптографічного ключа BB84	620
3.7.5. Протокол узгодження ключа B92	624
3.7.6. Атаки на протоколи квантового узгодження ключа	626
3.7.7. Реалізації протоколу квантового узгодження ключа	628
3.7.8. Протокол узгодження ключа E91	629
3.7.9. Квантові гроші Стівена Візнера	630
3.8. Архітектура та основні вимоги до квантових комп'ютерів	632
3.9. Огляд фізичних реалізацій квантових комп'ютерів	636
3.9.1. Гармонічний осцилятор як модель квантового комп'ютера	638
3.9.2. Квантовий комп'ютер на фотонах	640
3.9.3. Квантовий комп'ютер на оптичних резонаторах	643
3.9.4. Йони у пастках	644
3.9.5. Ядерний магнітний резонанс	647
Контрольні запитання до теми 2	653
Використана література до теми 3	655

Вступ

Інформаційні технології сьогодні є чи не найважливішим чинником розвитку сучасних держав унаслідок надзвичайно широкого впровадження телекомунікаційних технологій в усі галузі людської діяльності. Неможливо виділити хоча би один актуальний напрямок розвитку суспільства без використання інформаційних технологій. Дедалі більша кількість державних та приватних установ переходять на безпечний електронний документообіг, який вимагає ґрунтовної підтримки з боку обчислювальних систем різного ґатунку.

Не менш важливою на цьому шляху є кібербезпека. Особливого захисту потребують конфіденційні документи, які охоплюють різні ділянки як державного, так і приватного сектору. Основним способом захисту конфіденційної інформації здавна служить криптографія. Ну, а сьогодні, коли стрімко розвивається квантова інформатика і невдовзі очікується поява програмованого квантового комп'ютера, вимоги до криптографічного захисту інформації значно зростають.

Усі ці питання ґрунтуються на теоретичній базі, яку нагромадило людство за час розвитку інформаційних технологій. Для розуміння вказаних питань студенти мають набути знання та навички, які складають основу сучасних застосувань, теоретичних, алгоритмічних та програмних рішень, що лежать в основі більшості обчислювальних систем сучасності.

Видання, яке пропонують студентам фахівці Національного технічного університету "Харківський політехнічний інститут", Чернівецького національного університету імені Юрія Федьковича та Харківський національний університет радіоелектроніки, й присвячено висвітленню таких питань. Посібник охоплює три важливих розділи: теорію інформації та кодування, криптографічні механізми захисту інформації та основи квантової криптографії.

Перший розділ, який присвячено висвітленню питань теорії інформації та кодування, містить опис систем передавання інформації, ентропії різних джерел інформації, подано класифікацію та моделі каналів передачі інформації, розглянуто класичні алгоритми кодування інформації. Багато уваги приділено теорії скінченних полів Галуа, висвітлюються питання груп та кілець над цими полями і багато інших важливих для подальшого застосування питань.

У другому розділі обговорюються питання криптографічного захисту інформації в інформаційних системах. Зокрема, розглянуто математичні положення теорії чисел, еліптичні криві та проблеми факторизації та

дискретного логарифма, методи булевої алгебри та елементи кореляційного і спектрального аналізу, а також інші питання, що становлять основу сучасних криптографічних рішень.

Третій розділ присвячено основам квантової криптографії. Тут розглянуто фізичні та алгоритмічні основи, поняття квантового прискорення обчислень, парадокс Ейнштейна-Подольські-Розена, квантову телепортацію, алгоритми Д. Дойча, П. Шора та Л.Гровера, які складають базу квантової криптографії, і спонукали науковців до пошуку стійких до квантових обчислень криптографічних алгоритмів. Основну увагу приділено квантовим протоколам розподілу криптографічних ключів та атакам на них. Розглянуто сучасний стан квантового комп'ютингу та перспективи його розвитку.

Оволодіння матеріалом, викладеним у навчальному посібнику, дозволить студентам набути ґрунтовних знань та навичок, які сприятимуть розширенню загальнонаукового світогляду, вдосконаленню фахових компетенцій та дозволить бути обізнаним із загальними тенденціями розвитку сучасних інформаційних технологій.

Видання призначене для студентів, які навчаються у закладах вищої освіти за спеціальностями: 121 – Інженерія програмного забезпечення; 122 – Комп'ютерні науки; 123 – Комп'ютерна інженерія; 124 – Системний аналіз; 125 – Кібербезпека та захист інформації; 126 – Інформаційні системи і технології, а також може бути корисним для студентів усіх інших спеціальностей та освітньо-професійних програм, в межах яких викладаються дисципліни циклу інформаційних технологій. Звичайно, ця книга буде корисною й тим особам, що займаються освітою самостійно.

Тема 1. Теорія інформації та кодування

1.1. Загальні поняття про системи передачі інформації

Будь-яке відображення матеріального миру, яке може бути зафіксоване живою істотою або приладом, несе в собі інформацію.

Інформація (від латинського *information* – роз'яснення, викладення) в загальноприйнятому значенні пов'язана зі змістовою стороною інформаційного процесу та означає деякі відомості, сукупність яких-небудь даних, знань. Існує багато різних визначень інформації. Звичайно під інформацією розуміють нові відомості про мир, що нас оточує, відомості, що ми отримуємо в результаті взаємодії з ним, пристосування до нього та зміни його у процесі діяльності [1, 2].

Віддзеркалення результатів людської діяльності або осмислення навколишнього світу може бути надано у формалізованому вигляді, наприклад, у вигляді наборів букв або цифр. Такі формалізовані набори звичайно називають *даними*. Дані, отримані від джерела інформації, називають *повідомленнями*. Інформацією стають ті повідомлення, що знімають невизначеність, існуючу до їхнього надходження [1, 2].

Потрібно розрізняти поняття “інформація” і “повідомлення”. *Повідомлення* – це форма подання інформації, це результат віддзеркалення джерелом на кінцевому інтервалі часу кінцевої кількості інформації, поданої у формі, що може бути сприйнята одержувачем.

Передавати інформацію, тобто які-небудь відомості або повідомлення з одного пункту в інший або від одних людей до інших, можна різними засобами і способами. Інформацію (повідомлення, відомості) передають за допомогою листів, газет, книг тощо. Можна передавати інформацію по лініях передачі. В них від відправника до одержувача передається не який-небудь вид енергії (механічної, електричної), а повідомлення, інформація.

Передача інформації здійснюється за допомогою спеціальних систем, які надалі називатимемо *системами передачі інформації* (СПІ). У таких системах повідомлення передаються за допомогою сигналів, що мають певні фізичні властивості. *Сигнал* – це матеріальний носій повідомлення. Сучасне визначення сигналу зводиться до його уявлення у вигляді фізичного явища або процесу, як функції часу, несучого

повідомлення (інформацію) про яку-небудь подію, стан об'єкта спостереження, команди управління тощо.

Сукупність засобів, призначених для передачі сигналу, називають *каналом передачі інформації*. Канал передачі інформації разом із відправником і одержувачем повідомлень утворює систему передачі інформації. На рис. 1.1 наведена узагальнена схема системи передачі інформації. Розглянемо всі елементи, що входять у цю схему.

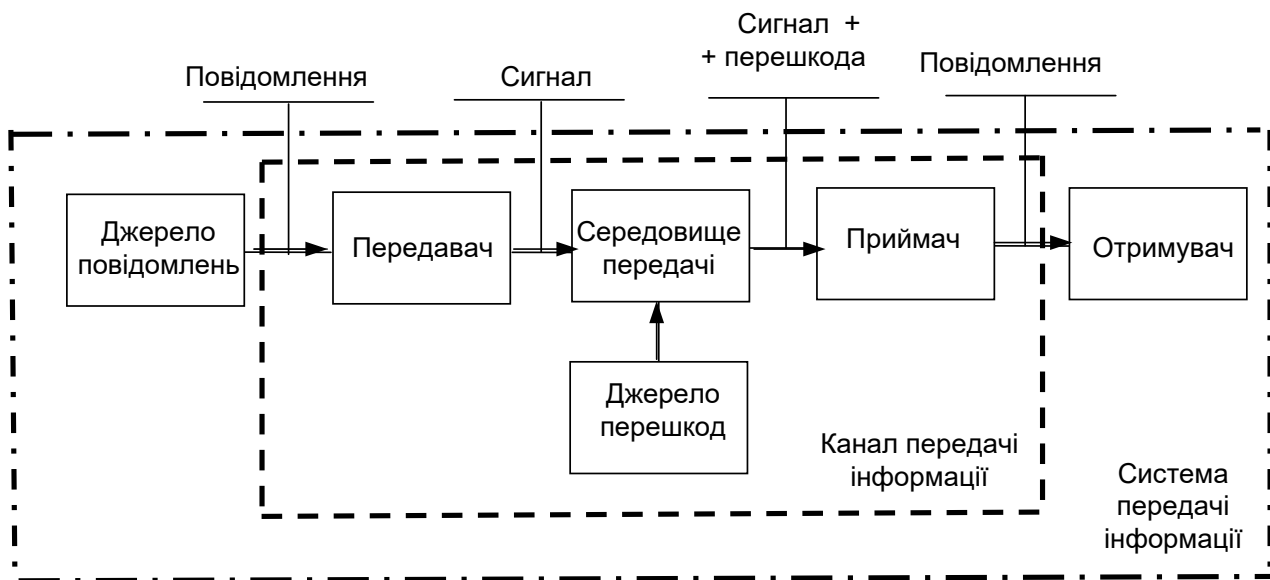


Рис. 1.1. Узагальнена схема системи передачі інформації

Джерело повідомлень. Його завдання – сформувати повідомлення. За своєю внутрішньою структурою джерело повідомлення може бути дискретним або безперервним.

Передавач. У загальному випадку він виконує наступні три функції:

- однозначне перетворення повідомлень у низькочастотний електричний сигнал;
- перетворення структури отриманих електричних сигналів з метою додання їм якостей, що забезпечують високу *завадостійкість* (здатність системи протистояти шкідливому впливу перешкод, обумовлена її власними властивостями, називається завадостійкістю) і ефективність використання *пропускної здатності* (кількість відомостей, що може бути передано з системи за одиницю часу, називається пропускною здатністю системи передачі інформації);
- перетворення низькочастотних електричних сигналів у високочастотні для передачі їх на задані відстані (*модуляція*).

Однак передавач не завжди виконує всі три функції. Наприклад, при телефонному зв'язку він (передавачем у цьому випадку є мікрофон) виконує тільки першу функцію – перетворення неелектричних величин у низькочастотні електричні сигнали. Навпаки, у радіозв'язку передавач виконує або всі три функції, або першу та третю.

Лінія передачі інформації. Передавач і приймач зв'язані між собою лінією передачі інформації, по якій передаються сигнали. Лінія може являти собою, наприклад, пару проводів, кабель або деяку обмежену зону простору.

Приймач. Це елемент, що перетворює сигнал, який надходить з лінії передачі інформації на його вхід, у повідомлення: при цьому він виконує функції, протилежні до функцій передавача. Однак у загальному випадку на вхід приймача надходить не чистий сигнал, а сукупність сигналу та завади.

Завадою називається стороннє проникнення, що діє в системі передачі інформації та перешкоджає правильному прийому сигналів. Завади можуть бути як зовнішні, так і внутрішні. В узагальненій схемі передачі інформації всі джерела завад звичайно перераховуються в один – зовнішній, який діє на лінії.

Отже, якщо на виході передавача маємо сигнал , то на вході приймача буде прийнятий сигнал $y(t)$, дещо відмінний від вихідного. Це відбувається, по-перше, через те, що сигнал, проходячи по фізичному середовищу, буде витратити потужність, значить амплітуда його на вході приймача зменшується. Це зменшення характеризується коефіцієнтом загасання в каналі – " α " (у загальному випадку " α " є функція часу, тобто $\alpha = \alpha(t)$).

Крім того, сигнал на вході приймача буде прийнятий з деяким запізнюванням τ щодо відліку часу t_0 . Величина τ також може бути не постійною, а залежати від часу (наприклад, при зв'язку на коротких хвилях час проходження сигналу залежить від висоти іонізованих шарів, а ця величина, у свою чергу, є функцією часу доби).

Нарешті, сигнал спотворюється завадою $n(t)$, що діє в лінії. Тепер, виходячи із усього вищевикладеного, можна записати вид сигналу на вході приймача:

$$y(t) = \alpha(t) \times x[t + \tau(t)] + n(t)x(t) \quad (1.1)$$

Основне призначення СПІ – передати повідомлення від джерела повідомлень до одержувача. При цьому йдеться не просто про передачу

деякої кількості інформації, а оскільки інформація має очевидну властивість старіння, то необхідно говорити про передачу деякого заданого об'єму інформації по каналу з завадами за заданий (допустимий) час. Тому при оцінці СПІ перш за все виходять з того, яку точність передачі повідомлень вона забезпечує при заданій швидкості передачі інформації.

Точність передачі повідомлень визначається ступенем відповідності прийнятого повідомлення переданому. Цей показник називають *достовірністю*. При передачі дискретних повідомлень цю відповідність можна оцінити відношенням числа помилково прийнятих елементарних повідомлень до загального числа переданих. При достатньо великій довжині повідомлення і незмінних умовах передачі це відношення є ймовірністю помилкового прийняття елементарного повідомлення (або відповідного йому елементарного сигналу) – $P_{\text{пом}}$.

При передачі безперервних повідомлень ступінь відповідності прийнятого повідомлення переданому характеризують середньоквадратичним відхиленням одного від іншого.

Характеристикою тісно пов'язаною з достовірністю є *завадостійкість*, під якою розуміється здатність системи передачі інформації протистояти дії різних завад і спотворень, що заважають.

Якщо хочуть підкреслити здатність СПІ протистояти навмисним завадам, то використовують термін *завадозахищеність*.

Для кількісної оцінки завадостійкості використовують різні показники. Найбільше поширення набув показник, відповідно до якого завадостійкість оцінюється відношенням середньої потужності сигналу до середньої потужності завади на вході приймача, при якому забезпечується задана достовірність. Чим менше величина цього відношення, тим вище завадостійкість системи.

Визначення залежностей, що пов'язують достовірність передачі з величиною відношення потужності сигналу до потужності завади на вході приймача, є однією з основних задач теорії завадостійкості.

Різні системи передачі інформації дозволяють передавати одні й ті ж повідомлення з різною швидкістю. Чисельно швидкість передачі інформації визначається кількістю інформації, що поступила від відправника до одержувача за одну секунду. Кількість інформації вимірюють в спеціальних одиницях – так званих *двійкових одиницях* або *бітах*. Швидкість передачі інформації випадковим чином залежить від ряду чинників: статистичних властивостей повідомлень, що передаються,

перешкод, характеристик каналу передачі інформації. Гранична швидкість передачі інформації по каналу передачі інформації визначається його *пропускною здібністю*. Не вдаючись в подробиці обговорення цього поняття, відзначимо, що пропускна здібність характеризує максимальну кількість інформації, яка може бути передана по каналу в одиницю часу і є одним з фундаментальних понять, що характеризують потенційні можливості системи передачі інформації.

При передачі дискретних повідомлень для характеристики швидкодії апаратури користуються поняттям *технічної швидкості передачі*. Вона визначається кількістю посилок дискретного повідомлення (числом змін сигналу), сформованих за секунду і вимірюється в бодах.

Достовірність і швидкість передачі, будучи основними характеристиками, не можуть в той же час бути безумовною оцінкою системи передачі інформації, оскільки завжди виникає таке питання: а якою ціною ці показники досягнуті? Необхідно мати таку систему передачі інформації, яка забезпечувала б задану достовірність і швидкість передачі інформації найбільш економічно, тобто з якнайменшими витратами, під якими можна розуміти потужність сигналу, смугу частот, вартість апаратури тощо. Сукупність властивостей (показників), що характеризують економічність системи з погляду вказаних витрат, називають ефективністю системи.

Важливою характеристикою деяких систем передачі інформації є *скритність*, під якою розуміється здатність системи працювати в режимі, що утрудняє виявлення повідомлень, які передаються, і оцінку їх параметрів спеціальною розвідувальною апаратурою зломисника. Розрізняють енергетичну, просторову, часову і частотну скритність. Енергетична скритність досягається зниженням середньої потужності сигналу; просторова – застосуванням гостронаправлених антен; часова – зменшенням тривалості сеансу передачі інформації за рахунок підвищення швидкості передачі; частотна – зміною робочих частот. Зрідка для підвищення скритності використовуються криптографічні методи, що полягають в спеціальному перетворенні повідомлень при передачі, яке утруднює їх розшифровку зломисником.

Існують багато інших параметрів, що характеризують системи передачі інформації з різних точок зору. До них, зокрема, належать надійність, габарити, маса, вартість, потужність тощо.

У найзагальнішому вигляді процес передачі інформації виглядає так. Повідомлення, що видається джерелом інформації, перетворюється на

сигнал, який за допомогою певних технічних засобів передається одержувачеві. Приймавши сигнал і знаючи закон відповідності сигналів і повідомлень, одержувач може вилучити інформацію, що міститься в повідомленні. У реальних умовах, в результаті дії завад порушується взаємно-однозначна відповідність між сигналами, що передаються та приймаються, і, як наслідок, зменшується вірність передачі, що еквівалентно втраті деякої кількості інформації.

Розглянемо передачу і приймання одного елементарного повідомлення λ . В процесі передачі – приймання повідомлень розрізняють два стани: апіорний (до приймання повідомлення) і апостеріорний (після приймання повідомлення). Апіорним називається стан, в якому знаходиться одержувач до того, як буде послане повідомлення λ . Апостеріорний стан – це стан, в якому знаходиться одержувач після того, як буде послане повідомлення λ , а на приймальній стороні сформульована відповідна оцінка λ^* . Відомості, які має в своєму розпорядженні одержувач в апіорному і апостеріорному станах, називають відповідно апіорними і апостеріорними [1, 2].

У апіорному стані одержувач знає, що буде створене повідомлення λ , яке може прийняти одне з m наперед відомих значень $\lambda_1, \lambda_2, \dots, \lambda_m$. Яке саме значення з множини $\{\lambda_1, \lambda_2, \dots, \lambda_m\}$ прийме повідомлення λ , спостерігач (одержувач) не знає. Останнє положення є принциповим. Дійсно, якби одержувач наперед знав, яке повідомлення буде передане, не мало б сенсу передавати це повідомлення по каналу передачі інформації: нічого нового, одержавши повідомлення, він би не дізнався. Інакше кажучи, передача інформації була б відсутня. Створення повідомлень джерелом повідомлень є типовим випадковим експериментом, множиною результатів якого є множина елементарних повідомлень $\{\lambda_1, \lambda_2, \dots, \lambda_m\}$. Імовірність різних результатів $P(\lambda_i)$ звичайно можна вважати відомими одержувачеві в апіорному стані. Так, наприклад, якщо елементарні повідомлення – це букви тексту, то $P(\lambda_i)$ – ймовірність того, що довільно вибрана з тексту (або створена джерелом) буква буде λ_i . Таку ймовірність неважко підрахувати: потрібно взяти достатньо довгий текст і підрахувати кількість λ_i – N_i у цьому тексті. Якщо в тексті загальне число букв N , то $P(\lambda_i) = N_i / N$.

Таким чином, робота джерела повідомлень є випадковим експериментом, а одержувачеві в апіорному стані відомий закон

розподілу ймовірності різних результатів в цьому експерименті. Якщо розглядати повідомлення, сформоване на тривалішому інтервалі часу, то це буде послідовність результатів випадкових експериментів або послідовність випадкових величин, яка відповідно до визначень теорії ймовірності є випадковим процесом.

Нехай було послане повідомлення λ і одержана оцінка λ^* . У апостеріорному стані одержувач не може бути впевнений, що одержана їм оцінка λ^* співпадає з посланим повідомленням λ через випадковий (імовірнісний) характер завад. Коливання, що приймається, а, отже, і сформована по ньому оцінка λ^* також носять випадковий характер. У цілому передача – приймання повідомлення в системі передачі інформації є випадковим експериментом з двома статистично пов'язаними результатами λ і λ^* . Останні вказують на статистичний характер проблеми організації ефективної передачі інформації і роблять доцільним вивчення цієї проблеми статистичними методами. Відповідно основним апаратом теорії інформації є теорія ймовірності, математична статистика та теорія випадкових процесів.

Практично основна трудність, що виникає при передачі повідомлень, полягає у тому, що завади, які завжди присутні в каналі, істотно утрудняють, а іноді роблять нездійсненним відтворення повідомлень, що передаються, на приймальній стороні.

1.2. Кількість інформації та ентропія

Введемо кількісну міру інформації, що передається по каналу. Це деяка формалізована кількість інформації, що міститься в повідомленні, яка в першу чергу характеризує завантаження каналу передачі інформації і трудність передачі цього повідомлення.

Почнемо розгляд з простих повідомлень – дискретних. Виділимо один інтервал дискретності $t \in (0, T_0)$. Джерело повідомлень на цьому інтервалі створює елементарне повідомлення X , яке може прийняти одне з M значень символів: $x_1, x_2, \dots, x_M$. Загальне число символів алфавіту M називають основою (потужністю) алфавіту джерела повідомлень. Одержувач після обробки коливання, що приймається, ухвалює рішення про те, яке саме повідомлення з множини $\{x_i\}$ було передано. Відповідно до позначень, прийнятих в теорії інформації, це рішення (оцінку

повідомлення, що приймається) позначимо через Y . Рішень взагалі може бути M , тобто $Y \in \{y_1, y_2, \dots, y_M\}$, і y_1 означає рішення про те, що передано повідомлення x_1 і т.д.

Нагадаємо характеристику двох основних станів: апіорного і апостеріорного, в яких знаходиться одержувач в процесі приймання кожного елементарного повідомлення.

Відповідно до реальних умов роботи більшості СПІ в теорії інформації приймається, що в апіорному стані одержувачеві відомий набір можливих елементарних повідомлень $x_1, x_2, \dots, x_M$ і ймовірність посилки кожного повідомлення $P(x_1), P(x_2), \dots, P(x_M)$. Характерним є стан невизначеності у одержувача: він знає, що йому послано одне з M наперед відомих йому повідомлень, але яке саме послане, він не знає. Цю невизначеність одержувача в апіорному стані називають апіорною невизначеністю. Кількісно ця невизначеність може характеризуватися апіорним законом розподілу ймовірності посилки різних повідомлень $P(x_i)$, $i = 1, 2, \dots, M$.

Апостеріорний стан одержувача характеризується тим, що на вхід приймальної системи (приймача) надійшло коливання (що приймається) і по ньому сформовано рішення (оцінка) Y .

Припустимо, що одержувач, діючи за наказаним йому алгоритмом обробки, ухвалив деяке конкретне рішення, скажімо, y_j . Якщо завад у каналі немає, то одержувач в цьому випадку точно знає, що було послане повідомлення x_j . Ніякої невизначеності у нього не залишається. Але основним інтересом є урахування явищ, пов'язаних з обов'язковою наявністю різних завад в каналі. Якщо завади є, то після отримання оцінки y_j в адресата ще залишається деяка невизначеність відносно того, яке повідомлення з множини $x_1, x_2, \dots, x_M$ насправді було послане.

Ця невизначеність, називається апостеріорною невизначеністю адресата (одержувача) і характеризується апостеріорним розподілом $P(x_i / x_j)$, тобто розподілом імовірності посилки різних повідомлень x_i за умови, що прийняте повідомлення (оцінка) x_j . В окремому випадку, коли дію завад можна не враховувати, апостеріорний розподіл вироджується в розподіл вигляду $P(x_i / x_j) = P(x_i)$, що характеризує детермінованість результату.

Отримання інформації тотожне усуненню незнання чогось або усуненню невизначеності чогось. Тому, розумно визначити кількість інформації, яку одержує адресат або яку передають йому по каналу передачі інформації, як кількість невизначеності, що знімається в адресата при отриманні ним повідомлення. Відповідно кількість інформації, що передана по каналу передачі інформації, ми можемо визначити за допомогою наступної, поки що словесної, формули, що показана на рис. 1.2.

$$\boxed{\begin{array}{c} \text{Кількість інформації,} \\ \text{що передана по} \\ \text{каналу зв'язку} \\ \text{(одержана} \\ \text{адресатом)} \end{array}} = \boxed{\begin{array}{c} \text{Апріорна} \\ \text{невизначеність} \\ \text{адресата} \end{array}} - \boxed{\begin{array}{c} \text{Апостеріорна} \\ \text{невизначеність} \\ \text{адресата} \end{array}}$$

Рис. 1.2. Формула визначення кількості інформації, що передана по каналу зв'язку з завадами

В ідеалізованому каналі без завад, апостеріорна невизначеність відсутня, тобто дорівнює нулю. При цьому кількість інформації визначається згідно формули, що показана на рис. 1.3.

$$\boxed{\begin{array}{c} \text{Кількість інформації,} \\ \text{що передана по} \\ \text{каналу зв'язку} \\ \text{(одержана} \\ \text{адресатом)} \end{array}} = \boxed{\begin{array}{c} \text{Апріорна} \\ \text{невизначеність} \\ \text{адресата} \end{array}}$$

Рис. 1.3. Формула визначення кількості інформації, що передана по каналу зв'язку без завад

Таким чином, кількість інформації виражається через кількість невизначеності в адресата в апріорному і апостеріорному стані. Інакше кажучи, необхідність визначення кількісної міри інформації зводиться до необхідності визначення кількісної міри невизначеності.

Апріорна і апостеріорна невизначеності адресата є окремим випадком загальної невизначеності – невизначеності довільного апріорного експерименту, або невизначеності закону розподілу ймовірності $P(x_i)$, що описує довільний випадковий експеримент X з результатами $x_1, x_2, \dots, x_M$. Кількісно міру невизначеності випадкового

експерименту в теорії інформації умовилися називати ентропією і позначати буквою H .

Поняття “ентропія” запозичений з термодинаміки, де аналогічний термін характеризує середню невизначеність стану системи молекул речовини. Таким чином, термін “ентропія”, використовується надалі як синонім поняття невизначеності (кількості невизначеності), пов’язаний з результатом деякого випадкового експерименту.

1.2.1. Ентропія випадкового експерименту з рівноймовірними результатами

Будь-який випадковий експеримент, зокрема робота джерел повідомлень, описується законом розподілу імовірності різних результатів і має деяку невизначеність в тому значенні, що результат експерименту точно передбачити неможливо. Невизначеність тим більша, чим менше шансів (менша ймовірність) передбачити результат експерименту. І також, чим менша апіорна ймовірність окремої події, тим більшу кількість інформації вона несе. Ці твердження будуть лежати в основі визначення кількісної міри невизначеності (ентропії) випадкового експерименту і кількості інформації, що міститься в одному результаті.

Припустимо, що кількісною мірою невизначеності може бути величина, зворотна його апіорній імовірності $1/P(x_i)$. Однак при визначенні кількісної міри невизначеності необхідно, щоб ентропія випадкового експерименту задовольняла таким умовам.

1. Ентропія H випадкового експерименту з рівноймовірними результатами повинна монотонно зростати зі збільшенням кількості результатів (потужності алфавіту) M .

Дійсно, при киданні монети $M = 2$, ймовірність передбачити результат експерименту дорівнює $1/2$ (шанси передбачити результат 50 %). При киданні гральної кості $M = 6$, ймовірність передбачити результат експерименту дорівнює $1/6$.

До першої умови приєднаємо ще дві очевидні умови, яким повинна задовольняти кількісна міра невизначеності будь-яких випадкових експериментів (не тільки з рівноймовірними результатами).

2. Якщо число результатів експерименту дорівнює одиниці ($M = 1$), то результат експерименту відомий наперед і невизначеності немає ($H = 0$).

3. Якщо відбуваються два незалежних експерименти A і B , то повна невизначеність експерименту AB дорівнює сумі невизначеностей експерименту A й експерименту B (властивість адитивності)

$$H(AB) = H(A) + H(B) \quad (1.2)$$

Остання умова еквівалентна наступній очевидній вимозі, яку необхідно пред'явити до кількісної міри невизначеності.

Невизначеність $H(AA)$ двічі незалежно повтореного експерименту A в два рази більша невизначеності $H(A)$ одного експерименту, $H(AA) = 2H(A)$. Якщо експеримент A незалежним чином повторюється n разів, то повна невизначеність n -кратного експерименту повинна дорівнювати $nH(A)$.

Виразимо тепер сформульовані умови або вимоги до кількісної міри невизначеності H в аналітичному вигляді. На підставі першої умови можна записати наступне.

Якщо експеримент має M рівноймовірних результатів, то

$$H = f(M),$$

де $f(\bullet)$ – деяка монотонно зростаюча функція, яку належить визначити.

Друга умова дає

$$f(1) = 0. \quad (1.3)$$

Для того, щоб скористатися третьою умовою, припустимо, що експеримент A має M_A рівноймовірних результатів і відповідно ентропію $H(A) = f(M_A)$, а експеримент B має M_B рівноймовірних результатів і ентропію $H(B) = f(M_B)$.

Тоді експеримент AB , що є сукупністю експерименту A і експерименту B (які в цьому випадку вважаються незалежними), має $M_A M_B$ рівноймовірних результатів і його ентропія повинна мати вираз $H(AB) = f(M_A M_B)$. На підставі третьої умови (1.2) для функції $f(\bullet)$ повинна виконуватися рівність

$$f(M_A M_B) = f(M_A) + f(M_B) \quad (1.4)$$

Співвідношенням (1.3) і (1.4) при довільних значеннях M_A і M_B задовольняє логарифмічна функція. Отже,

$$f(\bullet) = \log_a(\bullet). \quad (1.5)$$

Вибір основи логарифму принципового значення не має. Основа логарифму визначає величину одиниці вимірювання невизначеності. Вибір тієї або іншої одиниці є питанням зручності. Найчастіше всього приймають основу логарифма в (1.5) рівною 2.

Тоді кількість інформації $I(x_i)$, що міститься в символі x_i , вибраному з множини $\{x_i\}$ ($i = 1, 2, \dots, M$; M – об'єм алфавіту джерела повідомлень), з імовірністю $P(x_i)$, причому $\sum_{i=1}^M P(x_i) = 1$, визначається як

$$I(x_i) = \log \frac{1}{P(x_i)} = -\log P(x_i). \quad (1.6)$$

У випадку рівної імовірності повідомлень $P(x_1) = P(x_2) = \dots = P(x_M) = 1/M$ вираз (1.6) для кількості інформації зводиться до вигляду

$$I(x_i) = -\log P(x_i) = \log M. \quad (1.7)$$

Така міра кількості інформації була запропонована в 1928 році Р. Хартлі.

У виразі (1.7) і в подальшому, відповідно до символіки, прийнятої в теорії інформації, основа логарифма a не ставиться, якщо $a = 2$.

Визначимо ентропію випадкового експерименту з рівноймовірними результатами, як середню кількість інформації, що припадає на один символ

$$\begin{aligned} H &= \frac{1}{M} \sum_{i=1}^M I(x_i) = \frac{1}{M} \left[-\sum_{i=1}^M \log P(x_i) \right] = \\ &= \frac{1}{M} \sum_{i=1}^M \log M = \frac{1}{M} M \log M = \log M \end{aligned} \quad (1.8)$$

У виразі (1.8) за одиницю невизначеності прийнята невизначеність, яку має випадковий процес з двома рівноймовірними результатами ($M = 2$). Прикладами такого експерименту є кидання монети, посилка двійкового повідомлення “0” або “1” (при рівності ймовірностей “0” і “1”).

Дійсно, підставлення в (1.8) $M = 2$ дає $H = 1$. Цю величину називають “двійкова одиниця” (дв. од.), або “біт”, що є скороченням англійського терміну “binary digit”. Отже, в експерименті з двома рівноймовірними результатами

$$H = \log_2 2 = 1 \text{ дв. од.}$$

Одиниці вимірювання кількості інформації та ентропії залежать від вибору основи логарифма у формулах. При використанні десяткових логарифмів кількість інформації та ентропія визначаються в десяткових одиницях – *дітах*. У разі використання двійкових логарифмів кількість інформації та ентропія визначаються в двійкових одиницях – *бітах*. Нарешті, при використовуванні натуральних логарифмів одиницею вимірювання є натуральна одиниця – *ніт*.

При аналізі інформаційних процесів в пристроях, що функціонують на основі двійкової системи числення, зручно користуватися двійковими одиницями. У математичних викладеннях зручно користуватися натуральними одиницями.

Розглянемо приклад. Нехай джерело повідомлень створює одне з 32-х рівноймовірних повідомлень, тоді ентропія джерела повідомлень або апіорна ентропія адресата дорівнює

$$H = \log_2 32 = 5 \text{ дв. од.}$$

Кожне повідомлення при роботі такого джерела можна надати п'ятизначним двійковим числом і звести до послідовного проведення п'яти простих незалежних експериментів з двома рівноймовірними результатами $\{0, 1\}$.

Не дивлячись на збіг залежностей (1.7) та (1.8), ентропія H і кількість інформації $I(x_i)$ принципово різні.

Ентропія H виражає середню невизначеність стану джерела повідомлень і є об'єктивною характеристикою джерела повідомлень. Якщо відома статистика повідомлень, то H може бути обчислена апіорно, тобто до отримання повідомлення. Ентропія є міра недоліку інформації про стан джерела повідомлень. З надходженням інформації про стан джерела повідомлень ентропія останньої зменшується.

Кількість інформації $I(x_i)$ є апостеріорною характеристикою і визначає кількість інформації, що одержана з надходженням повідомлення x_i .

Збіг виражень для ентропії та кількості інформації свідчить лише про те, що кількість одержаної інформації чисельно дорівнює ентропії джерела у випадку рівної ймовірності його повідомлень.

1.2.2. Ентропія випадкового експерименту з нерівноймовірними результатами

Узагальнимо отриманий аналітичний вираз для ентропії випадкового експерименту з рівноймовірними результатами на експеримент з нерівноймовірними результатами.

Нехай повідомлення можуть приймати одне з M значень: $x_1, x_2, \dots, x_M$. Але тепер імовірність $P(x_i) = P_i$, $i = 1, 2, \dots, M$ різних результатів x_i (наприклад, імовірність посилки різних повідомлень) будемо вважати різними. Визначимо ентропію експерименту X .

Замість одноразово проведеного експерименту X будемо розглядати дуже довгу послідовність $X^{(1)}, X^{(2)}, \dots, X^{(N)}$, що складається з N незалежно проведених експериментів X . Ентропія N разів повтореного експерименту H_N (N -кратного експерименту) при обумовленій нами статистичній незалежності окремих експериментів дорівнює ентропії одноразового експерименту, помноженій на N

$$H_N = NH.$$

Тому можна записати

$$H = \frac{1}{N} H_N.$$

Розглянемо послідовність (ланцюжок) результатів N -кратного експерименту

$$X^{(1)}, X^{(2)}, \dots, X^{(N)}. \quad (1.9)$$

Якщо ланцюжок результатів (1.9) достатньо довгий (теоретично, якщо $N \rightarrow \infty$), то на підставі цього ланцюжка можна підрахувати ймовірність реалізації різних результатів

$$P(x_1) = P_1 = \frac{N_1}{N}, \dots, P(x_M) = P_M = \frac{N_M}{N}, \quad (1.10)$$

де $N_1, \dots, N_M$ – кількість випадань в ланцюжку результатів (1.9) результату $(x_1, \dots, x_M)$ відповідно.

Навпаки, якщо ймовірності різних наслідків $P_1, \dots, P_M$ задані, як це має місце у випадку, що розглядається, на підставі (1.10), то можна

передбачити наперед, яка кількість різних результатів наступить в ланцюжку результатів N-кратного експерименту.

Якщо N достатньо велике, то можна записати, що $N_1 = P_1 N$. Але на яких саме місцях ланцюжка з'являться N_1 результатів x_1 не відомо. Кількість результатів x_2 буде дорівнювати $N_2 = P_2 N$ і т.д. Нарешті, результат x_M наступить $N_M = P_M N$ разів. Отже, всі ланцюжки, що є різними реалізаціями N-кратного експерименту, матимуть один і той же набір елементів. Цей набір елементів або результатів графічно зображений на рис. 1.4.

Різні реалізації відрізняються одна від одної тільки розташуванням елементів в ланцюжку результатів. Невизначеність результату N-кратного експерименту полягає в невизначеності розташування наперед відомого набору результатів.

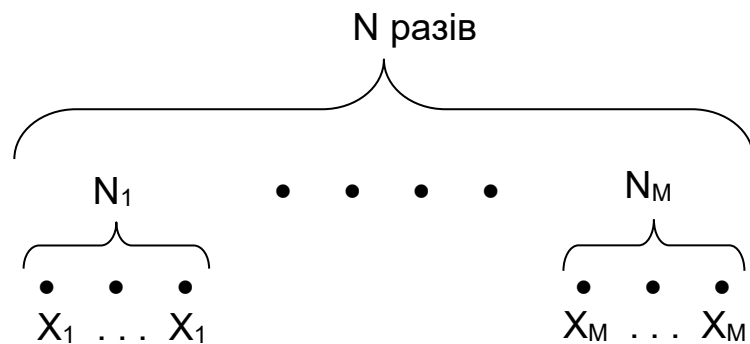


Рис. 1.4. Набір результатів N-кратного експерименту

Випадання кожного елемента ланцюжка (1.9) є незалежною подією. Тому ймовірність реалізації якого-небудь конкретного ланцюжка P_Σ дорівнює добутку ймовірностей реалізації окремих елементів і є однією і тією ж для всіх ланцюжків, що представляють різні реалізації N-кратного експерименту

$$P_\Sigma = \underbrace{P_1 \cdots P_1}_{P_1 N \text{ разів}} \cdots \underbrace{P_M \cdots P_M}_{P_M N \text{ разів}},$$

або

$$P_\Sigma = P_1^{P_1 N} \cdots P_M^{P_M N}.$$

Таким чином, всі ланцюжки результатів N-кратного експерименту при $N \rightarrow \infty$ рівноймовірні і ми приходимо до схеми експерименту з рівноймовірними результатами.

Ентропію N -кратного експерименту H_N можна визначити за допомогою виразу (1.8):

$$H_N = -\log P_\Sigma = -N[P_1 \log P_1 + \dots + P_M \log P_M].$$

Тепер неважко визначити ентропію $H = H(X)$ експерименту X з нерівноймовірними результатами $x_1, \dots, x_M$, скориставшись виразом (1.10),

$$H = -\sum_{i=1}^M P_i \log P_i. \quad (1.11)$$

Вираз (1.11) можна записати так:

$$H(X) = -\sum_{i=1}^M P(x_i) \log P(x_i). \quad (1.12)$$

Одержана загальна формула для ентропії випадкового експерименту X (або випадкової величини X) є основною, на якій базуються всі подальші висновки теорії інформації. В окремому випадку, коли всі результати рівноймовірні, тобто коли $P_1 = P_2 = \dots = P_M = 1/M$, вираз (1.12) переходить у (1.8). В цьому випадку ентропія визначає середню кількість інформації, що міститься в одному з символів (букв, цифр) алфавіту. Тоді кількість інформації повідомлення з k символів дорівнюватиме $I = kH$.

Приклад. Джерело інформації видає символи з ансамблю $X = \{x_i\}$; $i = \{1, 2, 3, 4\}$ з імовірностями 0,2; 0,3; 0,4; 0,1. Знайти ентропію джерела інформації.

Ентропію даного джерела знаходимо, використовуючи вираз (1.12)

$$\begin{aligned} H_1(X) &= -\sum_{i=1}^M P(x_i) \log P(x_i) = -\sum_{i=1}^4 P(x_i) \log P(x_i) = \\ &= -0,2 \log 0,2 - 0,3 \log 0,3 - 0,4 \log 0,4 - 0,1 \log 0,1 = \\ &= 0,4644 + 0,5211 + 0,5288 + 0,3322 = 1,85 \text{ біт/симв.} \end{aligned}$$

1.2.3. Умови максимізації ентропії джерела повідомлень

Припустимо, що експеримент X має M результатів $x_1, \dots, x_M$. Визначимо, який повинен бути розподіл імовірностей результатів $P(x_1), \dots, P(x_M)$, щоб ентропія експерименту X була максимальною, тобто

$$H_{\max}(X) = \max \left\{ -\sum_{i=1}^M P(x_i) \log P(x_i) \right\}. \quad (1.13)$$

При максимізації виразу (1.13) слід врахувати, що ймовірність $P(x_i)$ не може вибиратися довільно і має місце обмеження:

$$\sum_{i=1}^M P(x_i) = 1.$$

Сформульована задача є задачею на умовний екстремум. Для її вирішення запишемо функцію Лагранжа:

$$F = -\sum_{i=1}^M P_i \log P_i + \lambda \sum_{i=1}^M P_i.$$

Далі, користуючись системою рівнянь:

$$\left. \begin{aligned} \frac{\partial F}{\partial P_1} &= -\log P_1 - \frac{1}{\ln 2} + \lambda = 0 \\ &\dots \\ &\dots \\ \frac{\partial F}{\partial P_M} &= -\log P_M - \frac{1}{\ln 2} + \lambda = 0 \end{aligned} \right\},$$

знаходимо, що

$$P_1 = P_2 = \dots = P_M.$$

Таким чином, при заданій кількості результатів M , найбільшу ентропію має експеримент з рівноймовірними результатами. Цей результат відповідає початковому положенню про те, що невизначеність тим менша, чим більше шансів передбачити результат експерименту.

Так, наприклад, ентропія експерименту з двома рівноймовірними результатами ($P_1 = P_2 = 0,5$) дорівнює

$$H = -2 \frac{1}{2} \log \frac{1}{2} = 1 \text{ дв. од.}$$

Якщо результати не рівноймовірні, наприклад $P_1 = 0,99$, $P_2 = 0,01$, то невизначеності майже немає, тому що можна передбачити з великим

ступенем вірності, що результатом експерименту буде x_1 . Ентропія в цьому випадку буде дорівнювати:

$$H = -0,99 \log 0,99 - 0,01 \log 0,01 \approx 0,08 \text{ біт/симв.}$$

На рис. 2.4 зображений графік залежності ентропії H експерименту з двома результатами від імовірності P одного з результатів. З рис. 1.5 видно, що з наближенням закону розподілу до рівномірного ентропія експерименту зростає. Ентропія досягає максимуму, якщо символи незалежні та рівноймовірні:

$$H(X) = H_{\max} = \log M.$$

Величину $H_{\max}$ називають інформаційною місткістю алфавіту.

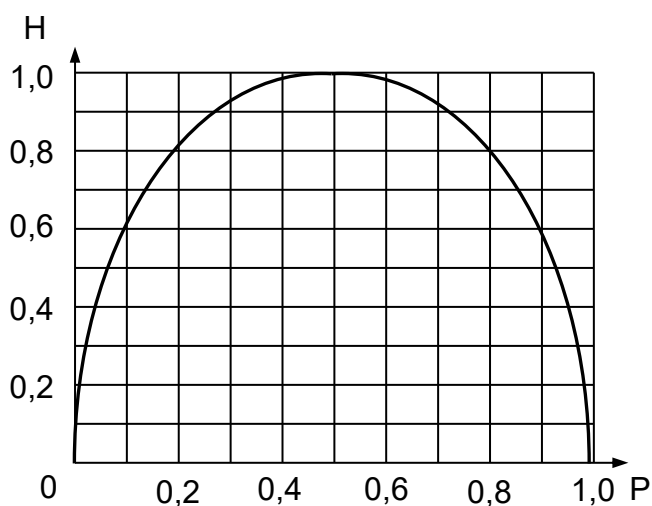


Рис. 1.5. Залежність ентропії експерименту з двома результатами від імовірності одного з результатів

Приведемо *приклад* розрахунку ентропії джерела повідомлень, що створює текст українською мовою. У українському алфавіті є всього 33 букви. Якби поява всіх букв була рівноймовірною, то ентропія джерела повідомлень була б $H = \log 33 \approx 5$ біт/симв.

Але ймовірність появи різних букв не однакова. Для визначення безумовної ймовірності появи різних букв потрібно розрахувати їх відносну частоту появи в типовому тексті. Фахівці з математичної лінгвістики зробили такі розрахунки. Виявилося, що $P(Е) = 0,054$, $P(О) = 0,091$, $P(Ф) = 0,002$ і т.д.

Якщо тепер підставити ймовірність появи різних букв у вираз (1.12), то одержимо ентропію джерела повідомлень, що створює текст української мови: $H = -P(A)\log P(A) - \dots - P(Я)\log P(Я) \approx 4,35$ біт/симв.

Грамматика мови накладає певні зв'язки на сусідні букви. Щоб точніше оцінити ентропію джерела типового буквенного тексту, потрібно визначити ентропію складного експерименту, результатом якого є двомірний (у загальному випадку n -мірний; $n \geq 2$) випадковий величини, не накладаючи умову незалежності окремих складових багатомірної випадкової величини.

1.2.4. Ентропія складного експерименту при взаємозалежності результатів

При рішенні задач передачі інформації часто мають справу з декількома джерелами, що дають залежні повідомлення. Сукупність повідомлень декількох джерел називають складним повідомленням.

Розглянемо складний експеримент XY , що складається в спільному проведенні експерименту X з результатами $x_1, \dots, x_M$ та експерименту Y з результатами $y_1, \dots, y_N$. Результатами експерименту XY є двомірні дискретні випадкові величини $\{x_i, y_j\}$, $i = 1, 2, \dots, M$, $j = 1, 2, \dots, N$ (звичайно $M = N$).

Будемо вважати, що задано безумовний закон розподілу ймовірності $P(y_j)$ компоненти y_j двомірної величини $\{x_i, y_j\}$ й умовний закон розподілу ймовірності $P(x_i / y_j)$ компоненти x_i .

Помітимо, що задані закони розподілу дають вичерпний опис експерименту XY (чи випадкової величини $\{x_i, y_j\}$). Використовуючи задані закони розподілу можна записати

– спільний закон розподілу ймовірності x_i і y_j :

$$P(x_i, y_j) = P(y_j)P(x_i / y_j);$$

– безумовний закон розподілу ймовірності складової x_i

$$P(x_i) = \sum_{j=1}^N P(x_i, y_j);$$

– умовний закон розподілу ймовірності складової y_j

$$P(y_j / x_i) = \frac{P(x_i, y_j)}{P(x_i)}.$$

Необхідно визначити таке:

- умовну ентропію $H(X / Y)$ експерименту X за умови, що експеримент Y проведений і його результат відомий;
- ентропію складного експерименту XY (тобто сукупності випадкових величин XY) з урахуванням взаємозалежності складових x_i і y_j .

Сформульована задача носить загальний характер. Однак нас буде цікавити, по-перше, апостеріорна ентропія адресата, тобто невизначеність адресата відносно того, яке повідомлення було передано після того, як став відомий результат експерименту Y (проведена оцінка прийнятого повідомлення); по-друге, ентропія, пов'язана з появою чергової букви тексту чи чергового елементарного повідомлення (експерименту X), за умови, що відома попередня буква тексту чи попереднє елементарне повідомлення (експерименту Y).

При визначенні ентропії і кількості інформації передбачалося, що елементи повідомлень статистично незалежні. Однак, у реальних умовах незалежність елементів повідомлення – явище досить рідке. Наприклад, при передачі тексту українською мовою ймовірності появи окремих букв залежать від того, які букви передували. Наприклад, якщо передана буква “п”, ймовірність того, що наступною може бути “а”, набагато більша, ніж ймовірність того, що наступною буквою буде “р”. Така залежність між елементами утворилася історично в процесі тривалого формування сучасної мови. Очевидно, що при визначенні ентропії й інформації повідомлень, елементи яких взаємозалежні (корельовані), не можна обмежуватися тільки безумовними ймовірностями елементів повідомлень. Необхідно враховувати й умовні ймовірності появи елементів повідомлень.

Кореляційні зв'язки між елементами повідомлень можуть поширюватися на різні групи елементів. Якщо є кореляційний зв'язок тільки між двома сусідніми елементами, то ймовірність передачі будь-якого елемента повідомлення буде залежати лише від того, який був попередній символ, тобто умовна ймовірність передачі елемента x_j буде

дорівнювати $P(x_j / y_i)$. Сукупність умовних імовірностей $P(x_j / y_i)$ зручно записати у вигляді матриці:

$$P(X / Y) = \begin{vmatrix} p(x_1 / y_1), & p(x_1 / y_2), & \dots, & p(x_1 / y_N) \\ p(x_2 / y_1), & p(x_2 / y_2), & \dots, & p(x_2 / y_N) \\ \vdots & \vdots & \ddots & \vdots \\ p(x_M / y_1), & p(x_M / y_2), & \dots, & p(x_M / y_N) \end{vmatrix}.$$

Якщо корелятивні зв'язки охоплюють три елементи повідомлень, то умовна ймовірність передачі елемента x_j буде дорівнювати $P(x_j / y_i, z_k)$.

Більшість повідомлень у реальних умовах являють собою послідовність, у якій корелятивні зв'язки поширюються на кінцеве число елементів. При достатній довжині такої послідовності, з достатньою точністю можуть бути визначені ймовірності й умовні ймовірності появи окремих повідомлень. Мова є типовим прикладом такої послідовності. У будь-якій книзі (крім вузькоспеціальних) частота повторення окремих букв і їхніх різних сполучень буде постійною, незалежно від змісту книги.

Нехай маємо корелятивний зв'язок тільки між сусідніми елементами. У цьому випадку ентропія елемента x_j буде визначатися умовною ймовірністю $P(x_j / y_i)$.

Умовна ентропія $H(X / y_1)$ експерименту X за умови, що експеримент Y проведений і наступила подія y_1 , дорівнює

$$H(X / y_1) = - \sum_{i=1}^M P(x_i / y_1) \log P(x_i / y_1).$$

Якщо результатом експерименту Y з'явилася подія y_2 , то ентропія експерименту X визначається розподілом $P(x_i / y_2)$ і дорівнює

$$H(X / y_2) = - \sum_{i=1}^M P(x_i / y_2) \log P(x_i / y_2).$$

Ентропія $H(X / y_i)$ називається частковою умовною ентропією.

Часткова умовна ентропія $H(X / y_1)$ може характеризувати, наприклад, невизначеність, пов'язану з появою чергової букви тексту, за умови, що попередня буква була "а" ($y_1 = a$). Аналогічно $H(X / y_2)$ –

ентропія, пов'язана з появою чергової букви за умови, що попередня буква була "б" ($y_2 = б$) і т.д.

Визначимо середню умовну ентропію, пов'язану, наприклад, з появою чергової букви за умови, що попередня буква відома.

У загальному випадку для різних результатів y_j експерименту Y , яким відповідають імовірності $P(y_j)$, величина умовної ентропії $H(X / y_j)$ експерименту X буде різною. Середнє значення величини $H(X / y_j)$, що отримана шляхом усереднення її за всіма можливими результатами y_j , називається середньою умовною ентропією експерименту X , за умови, що відомий результат експерименту Y , і дорівнює

$$H_2(X) = H(X/Y) = \sum_{j=1}^N P(y_j) H(X/y_j).$$

Таким чином, ми одержали умовну ентропію експерименту X

$$H_2(X) = H(X/Y) = - \sum_{i=1}^M \sum_{j=1}^N P(y_j) P(x_i / y_j) \log P(x_i / y_j) \quad (1.14)$$

або

$$H_2(X) = H(X/Y) = - \sum_{i=1}^M \sum_{j=1}^N P(x_i, y_j) \log P(x_i / y_j).$$

Ентропія $H_3(X) = H(X / YZ)$ визначається умовним розподілом $P(x_k / y_j, z_i)$ випадкової величини X при заданих значеннях y_j і z_i

$$H_3(X) = H(X / YZ) = - \sum_i \sum_j \sum_k P(x_k / y_j, z_i) P(y_j, z_i) \log P(x_k / y_j, z_i).$$

Якщо корелятивними зв'язками охоплена велика кількість елементів, то ентропія визначиться аналогічно. При наявності корелятивних зв'язків між елементами ентропія повідомлень, а отже, і кількість переданої інформації зменшуються. Це зменшення буде тим інтенсивніше, чим сильніше корелятивні зв'язки і чим більша кількість елементів буде охоплена цими зв'язками.

Розглянемо приклад. Джерело виробляє двійкові символи x_1 і x_2 з імовірностями $P(x_1) = 0,2$, $P(x_2) = 0,8$, $P(x_1 / x_1) = 0,4$, $P(x_2 / x_1) = 0,6$, $P(x_1 / x_2) = 0,15$, $P(x_2 / x_2) = 0,85$. Визначити умовну ентропію джерела повідомлень.

Ентропію цього джерела знаходимо, використовуючи вираз (1.14):

$$\begin{aligned} H_2(X) &= -\sum_{i=1}^2 P(x_i) \sum_{j=1}^2 P(x_j / x_i) \log P(x_j / x_i) = \\ &= -P(x_1) [P(x_1 / x_1) \log P(x_1 / x_1) + P(x_2 / x_1) \log P(x_2 / x_1)] - \\ &\quad -P(x_2) [P(x_1 / x_2) \log P(x_1 / x_2) + P(x_2 / x_2) \log P(x_2 / x_2)] = \\ &= -0,2(0,4 \log 0,4 + 0,6 \log 0,6) - 0,8(0,15 \log 0,15 + 0,85 \log 0,85) = \\ &= 0,682 \text{ біт/симв.} \end{aligned}$$

Якби символи були незалежними, то

$$H_1(X) = -\sum_{i=1}^2 P(x_i) \log P(x_i) = -0,2 \log 0,2 - 0,8 \log 0,8 = 0,722 \text{ біт/симв. .}$$

Таким чином, якщо символи джерела повідомлень залежні, то кожен з них несе в середньому меншу кількість інформації порівняно з незалежними символами. Це відбувається тому, що відомий попередній символ через наявність статистичних зв'язків зменшує невизначеність в появі подальшого символу, що і призводить до зниження його інформативності.

Знайдемо значення ентропії $H(X/Y)$ для двох граничних випадків.

1. Експерименти X і Y статистично незалежні:

$$P(x_i / y_j) = P(x_i).$$

При цьому

$$H(X/Y) = \sum_j P(y_j) \left[-\sum_i P(x_i) \log P(x_i) \right] = H(X).$$

Таким чином, середня умовна ентропія дорівнює безумовній. Із-за незалежності X і Y значення Y не може зменшити невизначеність відносно X .

2. Результат експерименту X цілком (функціонально) пов'язаний з результатом експерименту Y :

$$P(x_i / y_j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \quad i, j = 1, \dots, M.$$

При цьому, приймається до уваги, що $\lim_{x \rightarrow 0} x \log x = 0$ і $\log 1 = 0$. Для цього випадку одержимо

$$H(X/Y) = 0.$$

Таким чином, результат експерименту Y однозначно визначає результат експерименту X і невизначеність відсутня.

Тому в загальному випадку можна записати, що

$$0 \leq H(X/Y) \leq H(X).$$

Таким чином, величина умовної ентропії знаходиться в діапазоні між нулем і значенням безумовної ентропії. Перше має місце при жорсткому зв'язку між повідомленнями. Друге має місце при незалежності повідомлень.

Тепер перейдемо до визначення ентропії складного експерименту $H(X, Y)$.

Нехай маємо два джерела повідомлень. Повідомлення першого джерела приймають значення $x_1, x_2, \dots, x_M$ з імовірностями $P(x_1), P(x_2), \dots, P(x_M)$ і повідомлення другого джерела приймають значення $y_1, y_2, \dots, y_N$ з імовірностями $P(y_1), P(y_2), \dots, P(y_N)$.

Спільну ентропію сукупності повідомлень X і Y (ентропію об'єднання) можна записати у вигляді

$$H(X, Y) = - \sum_{i=1}^M \sum_{j=1}^N P(x_i, y_j) \log P(x_i, y_j),$$

де $P(x_i, y_j)$ – ймовірність спільної появи повідомлень x_i і y_j .

Вираз для ентропії об'єднання шляхом математичних перетворень можна представити у вигляді

$$H(X, Y) = H(X) + \sum_{i=1}^M P(x_i) H(Y / x_i),$$

де $H(Y / x_i) = - \sum_{j=1}^N P(y_j / x_i) \log P(y_j / x_i)$, – часткова умовна ентропія, що виражає ентропію повідомлення Y за умови, що мало місце повідомлення X .

Вираз $\sum_{i=1}^M P(x_i) H(Y / x_i)$ являє собою усереднення $H(Y / x_i)$ по всіх повідомленнях $x_1, x_2, \dots, x_M$ і є середньою умовною ентропією $H(Y / X)$ повідомлення Y за умови надходження повідомлення X . Тоді остаточно маємо

$$H(X, Y) = H(X) + H(Y / X).$$

Основний зміст умовної ентропії $H(Y / X)$ полягає в тому, що вона показує, яку ентропію дають повідомлення Y , коли вже відома ентропія повідомлень X .

З очевидної рівності $H(X, Y) = H(Y, X)$ одержимо

$$H(X, Y) = H(Y) + H(X / Y).$$

Таким чином, спільна ентропія двох джерел повідомлень дорівнює сумі безумовної ентропії одного з джерел повідомлень і умовної ентропії другого джерела повідомлень.

При безлічі джерел повідомлень $X, Y, Z, \dots$ їх спільна ентропія визначається наступним чином:

$$\begin{aligned} H(X, Y, Z, \dots) &= H(X) + H(Y, Z, \dots / X) = \\ &= H(Y) + H(X, Z, \dots / Y) = \\ &= H(Z) + H(X, Y, \dots / Z) = \dots \end{aligned}$$

Розглянемо основні властивості ентропії об'єднання.

1. При статистично незалежних повідомленнях X і Y спільна ентропія дорівнює сумі ентропій повідомлень

$$H(X, Y) = H(X) + H(Y),$$

тому що в цьому випадку $H(Y / X) = H(Y)$.

2. При повній статистичній залежності повідомлень X і Y спільна ентропія дорівнює безумовній ентропії одного з повідомлень:

$$H(X, Y) = H(X) = H(Y),$$

тому що в цьому випадку $H(Y / X) = 0$ і $H(X / Y) = 0$.

Розглянемо приклад. Ансамблі повідомлень $X(x_1, x_2, x_3)$ и $Y(y_1, y_2)$ поєднані. Ймовірності сумісних повідомлень $P(X, Y)$ такі: $p(x_1, y_1) = 0,05$; $p(x_1, y_2) = 0,1$; $p(x_2, y_1) = 0,6$; $p(x_2, y_2) = 0,05$; $p(x_3, y_1) = 0,1$; $p(x_3, y_2) = 0,1$;

Визначити ентропію об'єднаного повідомлення $H(X, Y)$.

$$\begin{aligned} H(X, Y) &= -\sum_{i=1}^M \sum_{j=1}^N p(x_i, y_j) \log p(x_i, y_j) = -\sum_{i=1}^3 \sum_{j=1}^2 p(x_i, y_j) \log p(x_i, y_j) = \\ &= -0,05 \log 0,05 - 0,1 \log 0,1 - 0,6 \log 0,6 - 0,05 \log 0,05 - 0,1 \log 0,1 - 0,1 \log 0,1 = \\ &= -(-0,216 - 0,332 - 0,442 - 0,216 - 0,332 - 0,332) = 1,87 \text{ біт/симв.} \end{aligned}$$

1.2.5. Ентропія джерел дискретних повідомлень

Поняття джерела інформації є найважливішим поняттям теорії інформації. Як вже говорилося, джерело інформації може бути представлено ансамблем повідомлень. Дискретним джерелом інформації вважається джерело, безліч значень якого є дискретним. Неперервним джерелом інформації вважається джерело, безліч значень якого є неперервним.

З погляду теорії інформації джерело вважається заданим повністю, якщо є деяка ймовірнісна модель, що описує процес появи сигналів на виході джерела. Для опису послідовності станів джерела використовуються математичні моделі у вигляді дискретних або неперервних випадкових процесів. Основу цього опису складає ймовірнісна міра ансамблю джерела. При цьому, в загальному випадку, вважається, що ймовірнісна міра визначається станом джерела і може змінюватися під час переходу його з одного стану в інший. Для побудови дискретного джерела необхідно знати алфавіт джерела $X = \{x_1, \dots, x_M\}$, з якого формуються букви u_i , відповідні певним станам U_i .

Хай $\bar{u} = (\dots, u_{-1}, u_0, u_1, \dots)$ позначає послідовність букв, що створюється джерелом, в якому кожна буква u_i вибирається з дискретного алфавіту X . Тоді повний опис джерела задається ймовірністю $P(u_{j+1}, \dots, u_{j+L})$, визначеною для всіх L послідовностей всіх початкових моментів j . При такому підході джерело описується як довільний дискретний випадковий процес.

Дискретне джерело називається стаціонарним, якщо ймовірність $P(u_{j+1}, \dots, u_{j+L})$ не залежить від початку відліку часу (умови формування повідомлень дискретним джерелом не змінюються за часом), тобто якщо виконується умова

$$P(u_1, u_2, \dots, u_L) = P(u_{j+1}, u_{j+2}, \dots, u_{j+L}) \quad (1.15)$$

для всіх j і послідовностей $\bar{u}_L = u_{j+1}, u_{j+2}, \dots, u_{j+L}$.

Вираз (1.15) означає, що ймовірність того, що джерело виробляє послідовність $\bar{u}_L = (u_1, u_2, \dots, u_L)$ на інтервалі від 1 до L , дорівнює ймовірності того, що виробляється точно така ж послідовність на інтервалі від $j+1$ до $j+L$. Тобто зсув послідовності на j не змінює її ймовірності.

Дискретне джерело називається періодичним, якщо вираз (1.15) справедливий для всіх j , які кратні деякому цілому числу $m > 1$. Найменше значення m , що задовольняє цій умові, називається періодом і позначається як m_T . Якщо розглядати блоки букв періодичного джерела з періодом m_T як деякі “супербукви” більшого алфавіту, то послідовність “супербукв” так само виявиться стаціонарною.

Стаціонарне джерело, що має властивість ергодичності, називається ергодичним. Ергодичність означає, що статистичні закономірності, одержані при дослідженні одного досить довгого повідомлення ($\bar{u}_L = (u_1, u_2, \dots, u_L)$ при $L \rightarrow \infty$) з імовірністю близькою до одиниці, справедливі для всіх повідомлень, що створюються джерелом.

Основною інформаційною характеристикою джерел інформації є ентропія. Ця величина кількісно характеризує невизначеність адресата відносно того, яке чергове повідомлення буде послане.

За відсутності завад в каналі передачі інформації кількість інформації, що передається по каналу передачі інформації, дорівнює апіорній ентропії адресата (або ентропії джерела). Тому ентропія джерела є мірою інформативності повідомлень, тобто характеризує кількість інформації, що міститься в повідомленнях, створюваних джерелом. Нарешті, виявляється, що ентропія джерела визначає кількість двійкових знаків, необхідних для кодування повідомлень, що створюються джерелом. Розглянемо джерела дискретних повідомлень і їх характеристики.

Нехай будь-який символ алфавіту в повідомленні може бути використаний незалежно від інших елементів і є рівноправним зі всіма іншими символами алфавіту.

Статистичний аналіз символів в таких послідовностях здійснювати не потрібно, оскільки у рівноймовірних статистично незалежних символів умовна ймовірність дорівнює безумовній і визначається як $P(x_i / x_j) = P(x_i) = 1/M$. Таким чином, в цьому випадку, $P(x_i)$ залежить лише від величини потужності алфавіту M . Такі джерела називаються джерелами нульового типу або джерелами Хартлі, а величини їх інформаційних мір відзначаються підрядковим індексом “0”. Ентропія джерела Хартлі H_0 є максимальною та визначається виразом (1.8).

Нескладно уявити собі інші, ніж у джерел Хартлі, правила формування повідомлень з елементів алфавіту.

Припустимо, наприклад, що про джерело відоме таке: алфавіт джерела має потужність M , але його здатність генерувати різні символи алфавіту різна: елемент x_1 він може генерувати необмежено; елементів x_2 джерело здатне створити за час своєї роботи декілька менше, ніж елементів x_1 , елементів x_3 – ще менше і т.д. Очевидно, що в повідомленнях дуже великої довжини у такого джерела ймовірність появи різних букв різна, навіть якщо вибір елементів алфавіту для формування повідомлення здійснюється статистично незалежно.

Математичною моделлю таких повідомлень буде дискретна випадкова послідовність, елементи в якій нерівноймовірні, але вибір елементу здійснюється в незалежних від попередніх виборів умовах. Такі послідовності в математиці відомі як послідовності Бернуллі, а джерело, що формує такі послідовності називається дискретним джерелом без пам'яті, а величини їх інформаційних мір будемо відзначати підрядковим індексом "1" (I_1, H_1). Повний математичний опис дискретного джерела без пам'яті створюється на основі статистичного аналізу безумовної ймовірності появи символів алфавіту в повідомленнях $P(x_i)$.

Для математичного опису джерела Бернуллі необхідно задати його алфавіт у вигляді ансамблю ймовірностей

$$\begin{aligned} X(x_1, x_2, x_3, \dots, x_M) &= \{a_i\}, i = 1, \dots, M; \\ \{P(x_1), P(x_2), \dots, P(x_M)\} &= \{P(a_i)\}, i = 1, \dots, M \end{aligned}$$

і вказати довжину повідомлення. Зрозуміло, що кількість різних повідомлень N_1 , яка може бути створена джерелом Бернуллі при відомих його параметрах M і n , вже не буде дорівнювати, як у джерел Хартлі M^n , а буде декілька меншою.

Ентропія джерела Бернуллі обчислюється за формулою (1.12).

До джерел з нерівноймовірними і статистично залежними символами належать такі джерела, повідомлення яких формуються при нерівноймовірній і залежній вибірці елементів алфавіту. Дискретні випадкові послідовності з такими властивостями вивчав російський математик А.А. Марков. Його ім'ям і названі джерела з вказаними правилами формування повідомлень. Дискретним джерелом Маркова називається дискретне джерело з пам'яттю. Це джерело, що створює послідовності u_L , у яких ймовірність формування чергового знака залежить від того, які знаки були вибрані до цього.

Причини, що призводять до появи нерівноймовірності елементів алфавіту в повідомленні зрозумілі з розгляду джерел Бернуллі. В українській мові, письмовий текст якої є характерним прикладом джерела Маркова, залежність елементів в словах обумовлена обмеженнями, що накладаються граматиною на правила формування повідомлень. Обмеження такого вигляду можуть інтерпретуватися як двомірна пов'язаність в послідовності Маркова (або ланцюги, які іноді називають послідовностями). Обмеження можуть розповсюджуватися на три і більше символів повідомлень дискретного джерела. У цих випадках математичними моделями повідомлень будуть двозв'язні, тризв'язні і більшої зв'язності ланцюги Маркова. Зрозуміло, що при цьому ускладнюються методи визначення інформаційних характеристик повідомлень в порівнянні з джерелами Хартлі і Бернуллі.

Для математичного опису джерел Маркова необхідно окрім вказівки параметрів M і n повністю задати ансамбль імовірності, як при описі джерел Бернуллі, і матрицю переходів кожного попереднього стану джерела в подальше.

Ентропія двомірного (двозв'язкового) джерела Маркова обчислюється за формулою (2.13).

Дискретні джерела Маркова є найзагальнішими моделями джерел дискретних повідомлень. Одномірні (джерела Бернуллі) і нульмірні (джерела Хартлі) є їх окремими випадками.

1.2.6. Надмірність джерел повідомлень

Більшість реальних джерел повідомлень мають надмірність, яка визначається відмінністю закону розподілу ймовірності появи символів від рівномірного і наявністю зв'язків між ними. Ентропія таких джерел менша за максимальну – кожен символ в середньому несе меншу кількість інформації, ніж він міг би нести.

Надмірність дискретного M -ічного джерела обчислюється за формулою

$$\rho = \frac{H_{\max}(X) - H(X)}{H_{\max}(X)} = 1 - \frac{H(X)}{H_{\max}(X)} = 1 - \frac{H(X)}{\log M}.$$

Вона характеризує ступінь використання інформаційної ємності алфавіту джерела. Якщо $\rho = 0$, то ємність алфавіту використовується

повністю, якщо $\rho > 0$, то, в принципі, існує інший, більш стислий спосіб подання повідомлень джерела.

Надмірність повідомлень джерела призводить до надмірності кодових комбінацій, що призводить до зменшення швидкості передачі інформації. Припустимо, що необхідно закодувати 32 букви українського алфавіту. При двійковому коді їх можна кодувати послідовностями з п'яти двійкових символів, оскільки таких послідовностей теж 32. Якщо вважати алфавіт рівноймовірним, то кожна буква і кожна кодова комбінація переносять 5 біт інформації, а кожен кодовий символ 1 біт.

Але насправді через нерівноймовірність і залежність кожна буква несе в середньому меншу кількість інформації. Так, ентропія джерела при урахуванні статистичних зв'язків всього лише між трьома буквами дорівнює 3 біт і на частину кожного кодового символу доводиться в середньому 0,6 біт. При цьому швидкість передачі інформації буде менша, ніж пропускна здібність каналу, тобто символ коду при одній і тій же тривалості τ міг би доставити за секунду $1/\tau$ біт інформації, а доставляє лише $0,6/\tau$ біт. Отже, канал використовується неефективно. Швидкість передачі інформації в ньому менша, ніж пропускна здібність.

Велика надмірність повідомлень, з одного боку, перешкоджає інформаційному обміну, вимагаючи зайвих витрат енергії і часу на передачу повідомлень. З іншого боку, повідомлення з малою інформаційною надмірністю виявляються дуже чутливими до дії завад, а це, в свою чергу, перешкоджає забезпеченню достовірності їх передачі по реальних каналах передачі інформації.

1.2.7. Ентропія джерел неперервних повідомлень. Диференційна ентропія

На відміну від дискретних повідомлень, утворених рахунковою множиною елементів (символів) x_i , що задані ймовірностями появи $P(x_i)$, символи алфавіту неперервних повідомлень пронумерувати не можна. Ймовірність кожного окремого значення неперервного повідомлення дорівнює нулю, а розподіл ймовірностей характеризується щільністю ймовірності $W(x)$.

Узагальнимо вираз (1.12)

$$H = - \sum_{i=1}^M P(x_i) \log P(x_i)$$

на випадок неперервних повідомлень. Для цього здійснимо квантування за рівнем, тобто замінимо неперервні значення повідомлень дискретними відліками $x_1, x_2, \dots, x_M$, як зображено на рис. 1.6. Імовірності появи дискретних відліків можна записати так: $P(x_k) = W(x_k)\Delta x$.

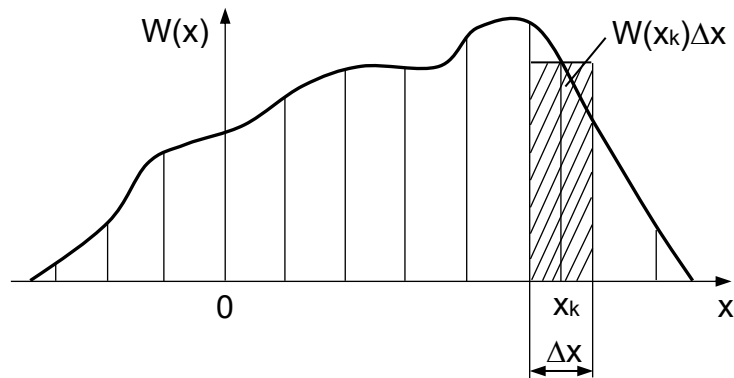


Рис. 1.6. Щільність імовірності неперервних повідомлень

Така заміна буде більш точною, чим менший інтервал Δx . Одержимо ентропію квантованого неперервного повідомлення, підставивши значення $P(x_k)$ в (1.12):

$$H^*(X) \approx - \sum_{k=1}^M W(x_k) \Delta x \log[W(x_k) \Delta x] = - \sum_{k=1}^M [W(x_k) \log W(x_k)] \Delta x - \sum_{k=1}^M \log \Delta x [W(x_k)] \Delta x. \quad (1.16)$$

При граничному переході ($\Delta x \rightarrow 0$, $m \rightarrow \infty$) перша сума виразу (1.16) переходить до інтеграла $\int_{-\infty}^{\infty} W(x) \log W(x) dx$, а друга сума – до величини

$$\lim_{\Delta x \rightarrow 0} \log \Delta x, \text{ тому що } \int_{-\infty}^{\infty} W(x) dx = 1.$$

Тоді вираз (1.16) приймає такий вигляд:

$$H(X) = - \int_{-\infty}^{\infty} W(x) \log W(x) dx - \lim_{\Delta x \rightarrow 0} \log \Delta x. \quad (1.17)$$

Вираз (1.17) називається абсолютною ентропією неперервного повідомлення і дорівнює нескінченності, тому що дорівнює нескінченності $\lim_{\Delta x \rightarrow 0} \log \Delta x$.

У цьому немає нічого несподіваного, тому що ентропія джерела повідомлення з нескінченною кількістю можливих станів також нескінченна. Однак ця обставина не є нездоланною завадою для визначення ентропії неперервних джерел. Дійсно, у реальних системах крок квантування Δx не може дорівнювати нулю, тому що неперервне повідомлення не можна відтворити з абсолютною точністю, – цьому заважають погрішності виміру і шуми в каналі передачі інформації. Тому інтервал Δx завжди кінцева величина і абсолютна ентропія неперервного повідомлення має вигляд:

$$H(X) = - \int_{-\infty}^{\infty} W(x) \log W(x) dx - \log \Delta x .$$

При порівнянні ентропії різних неперервних повідомлень, відтворених з однаковою точністю Δx , доданок $\log \Delta x$ нехай являє собою дуже велику, але кінцеву постійну величину, однаковою для порівнюваних функцій. Тому його звичайно опускають, а розглядають вираз для ентропії неперервних повідомлень у такому вигляді:

$$h(X) = - \int_{-\infty}^{\infty} W(x) \log W(x) dx . \quad (1.18)$$

Визначену в такий спосіб ентропію неперервного повідомлення називають диференціальною (оскільки вона пов'язана з диференціальним законом розподілу ймовірностей). Диференціальна ентропія не має абсолютного характеру, властивого ентропії дискретних повідомлень, і не може бути мірою невизначеності джерела повідомлень, оскільки залежить від масштабу і навіть здатна приймати негативні значення. Однак вона відіграє важливу роль при розгляді інформаційних характеристик систем передачі, оскільки ці характеристики в основному залежать від різниці ентропії різних джерел повідомлень.

1.2.8. Ентропія джерела з нормальним та рівномірним законом розподілу

Визначимо диференціальну ентропію повідомлення, значення якого розподілені за нормальним законом:

$$W(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left\{ -\frac{(x - \bar{x})^2}{2\sigma^2} \right\},$$

де $\bar{x}$ – математичне очікування;

σ^2 – дисперсія.

З виразу (1.18) отримаємо

$$\begin{aligned} h(X) &= - \int_{-\infty}^{\infty} W(x) \log W(x) dx = - \int_{-\infty}^{\infty} W(x) \left[\log \sqrt{2\pi\sigma^2} + \frac{\log e}{2\sigma^2} (x - \bar{x})^2 \right] dx = \\ &= \log \sqrt{2\pi\sigma^2} \int_{-\infty}^{\infty} W(x) dx + \frac{\log e}{2\sigma^2} \int_{-\infty}^{\infty} (x - \bar{x})^2 W(x) dx. \end{aligned}$$

Перший інтеграл за загальною властивістю щільності ймовірності дорівнює одиниці, а другий інтеграл дорівнює дисперсії σ^2 . Тоді

$$h(X) = \log \sqrt{2\pi\sigma^2} + \frac{1}{2} \log e = \log \sqrt{2\pi\sigma^2} + \log \sqrt{e} = \log \sqrt{2\pi e \sigma^2}.$$

Таким чином, диференціальна ентропія гаусовської випадкової величини не залежить від її математичного очікування і монотонно зростає зі збільшенням дисперсії.

Визначимо диференціальну ентропію процесу, рівномірно розподіленого на інтервалі $(-\alpha, \alpha)$.

З виразу (1.18) отримаємо

$$h(x_2) = - \int_{-a}^a \frac{1}{2a} \log \frac{1}{2a} dx = \log 2a.$$

Дисперсія процесу з рівномірним розподілом дорівнює σ^2 при $a = \sigma\sqrt{3}$. Таким чином, диференціальна ентропія процесу з рівномірним розподілом дорівнює

$$h(x_2) = \log \sqrt{12\sigma^2}.$$

При заданій дисперсії σ^2 диференціальна ентропія нормального процесу більша диференціальної ентропії процесу з рівномірним розподілом на

$$\Delta h = \log \sqrt{2\pi e \sigma^2} - \log \sqrt{12\sigma^2} = \log \sqrt{\frac{\pi e}{6}} \approx 0,3 \text{ біт/відлік}.$$

Раніше було доведено, що ентропія дискретного джерела максимальна при рівній імовірності появи його символів. При цьому виникає питання, а при якому розподілі ймовірностей ентропія неперервних повідомлень максимальна. Дослідження показали, що повідомлення з нормальним розподілом мають максимальну ентропію в порівнянні з будь-якими іншими випадковими повідомленнями з тією же дисперсією. З висновку формули диференціальної ентропії нормального закону розподілу виходить, що

$$-\int_{-\infty}^{\infty} W(x) \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left(-\frac{(x-\bar{x})^2}{2\sigma^2} \right) \right] dx = \log \sqrt{2\pi e \sigma^2}$$

для будь-якого закону розподілу.

Для знаходження максимальної ентропії визначимо різницю диференціальної ентропії довільного закону розподілу $W'(x)$ і нормального закону

$$\begin{aligned} h(X) - h_n(X) &= -\int_{-\infty}^{\infty} W'(x) \log W'(x) dx + \int_{-\infty}^{\infty} W'(x) \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left(-\frac{(x-\bar{x})^2}{2\sigma^2} \right) \right] dx = \\ &= \int_{-\infty}^{\infty} W'(x) \log \frac{1}{W'(x)} dx + \int_{-\infty}^{\infty} W'(x) \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left(-\frac{(x-\bar{x})^2}{2\sigma^2} \right) \right] dx = \\ &= \int_{-\infty}^{\infty} W'(x) \log \left[\frac{1}{W'(x)\sqrt{2\pi\sigma^2}} \exp \left(-\frac{(x-\bar{x})^2}{2\sigma^2} \right) \right] dx. \end{aligned}$$

Враховуючи, що $\log(x) \leq x-1$, одержимо

$$\begin{aligned} h(X) - h_n(X) &= \int_{-\infty}^{\infty} W'(x) \log \left[\frac{1}{W'(x)\sqrt{2\pi\sigma^2}} \exp \left(-\frac{(x-\bar{x})^2}{2\sigma^2} \right) \right] dx \leq \\ &\leq \int_{-\infty}^{\infty} W'(x) \left[\frac{1}{W'(x)\sqrt{2\pi\sigma^2}} \exp \left(-\frac{(x-\bar{x})^2}{2\sigma^2} \right) - 1 \right] dx = \\ &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left(-\frac{(x-\bar{x})^2}{2\sigma^2} \right) dx - \int_{-\infty}^{\infty} W'(x) dx = 1 - 1 = 0 \end{aligned}$$

Отже, максимальна диференціальна ентропія досягається при нормальному законі розподілу. За аналогією з формулами для дискретного джерела кількість інформації, що міститься в одному неперервному відліку процесу $Y(t)$ відносно відліку процесу $X(t)$, визначається за формулою

$$I(X, Y) = - \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W(x, y) \log \frac{W(x, y)}{W(x)W(y)} dx dy ,$$

де $W(x, y)$ – сумісна щільність імовірності процесів $X(t)$ і $Y(t)$.

Величину $I(X, Y)$ можна записати так:

$$I(X, Y) = h(X) - h(X / Y) = h(Y) - h(Y / X),$$

де $h(X)$, $h(Y)$ – диференціальна ентропія на відлік процесів $X(t)$ і $Y(t)$;

$$h(X / Y) = - \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W(x, y) \log W(x / y) dx dy \quad \text{– умовна диференціальна}$$

ентропія відліку $X(t)$ при відомому відліку $Y(t)$;

$$h(Y / X) = - \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W(x, y) \log W(y / x) dx dy \quad \text{– умовна диференціальна}$$

ентропія відліку $Y(t)$ при відомому відліку $X(t)$.

Розглянемо приклад. По каналу передачі інформації передається сигнал $X(t)$, який являє собою нормальний випадковий процес з нульовим середнім значенням і дисперсією $\sigma_x^2 = 4$ мВт. У каналі діє незалежний до сигналу нормальний шум $n(t)$ з нульовим середнім значенням і дисперсією $\sigma_n^2 = 1$ мВт. Знайти диференціальні ентропії вхідного та вихідного сигналів, а також умовні диференціальні ентропії $h(X / Y)$ та $h(Y / X)$.

Сигнал на виході $Y(t) = X(t) + n(t)$. Так як $X(t)$ і $n(t)$ незалежні та мають нормальний розподіл, $Y(t)$ також буде мати нормальний розподіл з дисперсією $\sigma_y^2 = \sigma_x^2 + \sigma_n^2$. Диференціальні ентропії вхідного та вихідного сигналів будуть дорівнювати

$$h(X) = \log \sqrt{2\pi e \sigma_x^2} = 3,05 \text{ біт/відлік} ;$$

$$h(Y) = \log \sqrt{2\pi e (\sigma_x^2 + \sigma_n^2)} = 3,21 \text{ біт/відлік} .$$

Умовна диференціальна ентропія відліку $Y(t)$ при відомому $X(t)$ визначається ентропією шуму у каналі. Отже,

$$h(Y / X) = \log \sqrt{2\pi e \sigma_n^2} = 2,05 \text{ біт/відлік}.$$

Найдемо умовну диференціальну ентропію відліку $X(t)$ при відомому $Y(t)$:

$$\begin{aligned} h(X / Y) &= - \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W(x, y) \log W(x / y) dx dy = \\ &= - \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W(x) W(y / x) \log \frac{W(x) W(y / x)}{W(y)} dx dy. \end{aligned}$$

Так як $\int_{-\infty}^{\infty} W(x) W(y / x) dx = W(y)$, отримуємо:

$$h(X/Y) = h(X) + h(Y/X) - h(Y) = \log \sqrt{2\pi e \frac{\sigma_x^2 \sigma_n^2}{\sigma_x^2 + \sigma_n^2}} = 1,89 \text{ біт/відлік}.$$

1.3 Канали передачі інформації

1.3.1 Класифікація каналів передачі інформації

При теоретичному аналізі процесів передачі інформації під каналом передачі інформації розуміють сукупність технічних засобів (передавача і приймача) і середовища розповсюдження (лінії передачі інформації), що забезпечують передачу повідомлень і відповідних їм сигналів між деякими точками (перетинами) системи передачі інформації, вибраними відповідно до тих задач, що ставить перед собою дослідник.

Можна виділити два основні напрями класифікації каналів і відповідних їм моделей, що виходять із самої природи та характеру переданих системою передачі інформації повідомлень (первинних сигналів), які ми раніше розділили на два типи: дискретні та неперервні.

Відповідно розрізняють дискретні і неперервні канали передачі інформації (моделі каналів передачі інформації).

У дискретних каналах передачі інформації повідомлення на вході і виході (у перетинах) каналу передачі інформації є дискретними, в неперервних каналах – неперервними.

Якщо перетини беруться в проміжних точках каналу передачі інформації, то, прив'язуючись вже до вигляду сигналу в цих перетинах, які можуть бути неперервними незалежно від характеру повідомлень, переданих в системі передачі інформації, розглядаються також дискретно – неперервні і неперервно – дискретні канали. У першому випадку на вхід каналу поступають дискретні сигнали, а з виходу знімаються неперервні, в другому – навпаки. У результаті в одній і тій же системі передачі інформації можна виділити як дискретні, так і неперервні канали (і відповідно канали змішаного типу). Все залежить від того, які вибрані точки входу і виходу (перетини) каналу.

Вибір таких точок найчастіше диктується тим, що визначені раніше функції, здійснювані над сигналами в передавачі і приймачі, у багатьох випадках виконуються окремими відносно самостійними функціонально закінченими блоками (пристроями), які можуть оптимізуватися автономно за своїми показниками. Наприклад, в сучасних системах передачі дискретних повідомлень прийнято розрізняти такі дві групи пристроїв: модеми і декоди.

Модемами називають пристрої, що перетворюють код в сигнал і сигнал в код (модулятори–демодулятори). Оптимізація модему дозволяє знайти для заданих умов передачі інформації найкращий вигляд сигналу і відповідно оптимальний спосіб його приймання.

Кодеками називають пристрої, що перетворюють повідомлення в код і код в повідомлення (кодери – декодери). Оптимізація кодека при заданих статистичних характеристиках решти частини системи передачі інформації дозволяє визначити оптимальні методи кодування і декодування.

При класифікації враховують і інші особливості каналних пристроїв: тип лінії передачі інформації (фізичний ланцюг, радіолінія, оптична лінія); смугу частот, що відводяться для сигналу (вузькосмуговий, широкосмуговий); технічну швидкість передачі дискретних сигналів (низькошвидкісні, високошвидкісні).

1.3.2. Модель дискретного каналу передачі інформації

Пояснимо ідеї викладеного вище підходу до визначення типів каналів на прикладі дискретної системи передачі даних, структурна схема якої зображена на рис. 1.7.

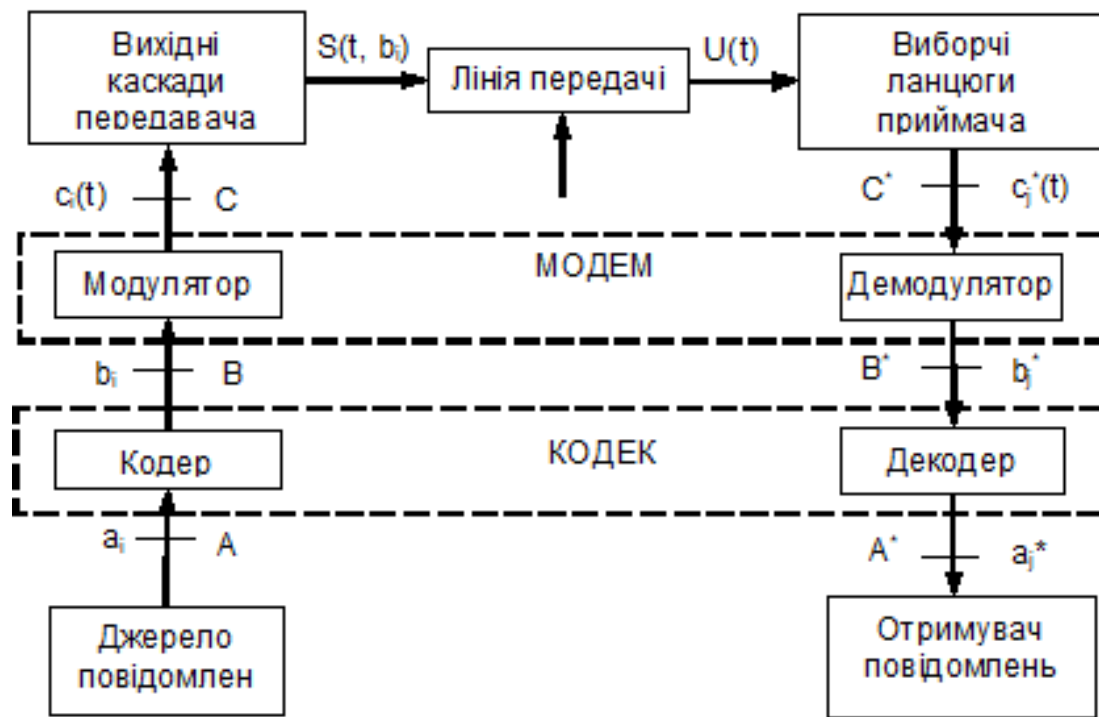


Рис. 1.7. Структурна схема системи передачі дискретної інформації

Об'єктом передачі в дискретному каналі є кодові символи. Розглянемо канал передачі інформації, зображений на рис. 1.7 перетинами B і B^* , по якому передається кодовий символ b_i , що може приймати одне з M значень $b_1, b_2, \dots, b_M$.

Дискретний канал вважається математично описаним, якщо заданий алфавіт кодових символів на вході b_i ($i = 1, \dots, M$) разом з їх імовірністю $P(b_i)$ (якщо джерело має пам'ять, то повинні бути задані апіорна й умовна ймовірності передачі символів), алфавіт кодових символів на виході b_j^* ($j = 1, \dots, M$) і значення ймовірності переходів $P(b_j^* / b_i)$, тобто ймовірність того, що на виході каналу з'явиться символ b_j^* за умови, що на вхід поданий символ b_i .

В моделі каналу можливі переходи вхідних символів у вихідні описуються у вигляді матриці $\|P(b_j^* / b_i)\|$ розмірності $M \times M$, що називається матрицею перехідних ймовірностей або каналною матрицею.

При розгляді складного експерименту XU знання цих характеристик є вичерпним для статистичного аналізу процесу передачі – приймання окремого кодового символу. Зокрема, з їх допомогою визначаються

– сумісна ймовірність подачі символу b_i на вхід і появи символу b_j^* на виході

$$P(b_i, b_j^*) = P(b_i)P(b_j^* / b_i) = P(b_j^*)P(b_i / b_j^*);$$

– безумовна ймовірність появи символу b_j^* на виході каналу

$$P(b_j^*) = \sum_{i=1}^m P(b_i, b_j^*) ;$$

– умовна ймовірність того, що на вхід поданий символ b_i за умови, що на виході з'явився символ b_j^* (апостеріорна ймовірність)

$$P(b_i / b_j^*) = \frac{P(b_i)P(b_j^* / b_i)}{\sum_{i=1}^m P(b_i)P(b_j^* / b_i)}.$$

Дискретний канал називається однорідним (стаціонарним) і без пам'яті, якщо ймовірність переходів $P(b_j^* / b_i)$ для кожної пари (i, j) не змінюється за часом і не залежить від символів, що були передані раніше.

Якщо ймовірність $P(b_j^* / b_i)$ залежить від часу, то канал називається неоднорідним (нестационарним); якщо ж вона залежить від символів переданих раніше, то канал називається каналом з пам'яттю.

Якщо в однорідному дискретному каналі алфавіти на вході і виході однакові ($M = M^*$) і для будь-якої пари $i \neq j$ ймовірності $P(b_j^* / b_i) = p_0 / (M - 1)$, а для пари $i = j$ ймовірності $P(b_j^* / b_i) = q = 1 - p_0$, то такий канал називають симетричним каналом без стирання. При $M = 2$ приходимо до моделі двійкового симетричного каналу. Якщо ж об'єм алфавіту символів на виході каналу M^* перевищує об'єм алфавіту вхідних символів M , то канал називають каналом із стиранням.

Найчастіше на практиці зустрічаються дискретні канали із стиранням, в яких $M^* = M + 1$. У цих каналах алфавіт вихідних символів містить один додатковий символ, який використовується для відмови від ухвалення рішення про символ, що приймається, в тих випадках, коли він не може бути визначений з необхідною надійністю. При появі такого

символу можна послати запит на передавальну сторону системи передачі інформації на повторення вхідного символу.

В дискретних каналах передачі інформації повідомлення передаються за допомогою послідовностей символів дискретного джерела найчастіше однаковою для всіх повідомлень довжини. Такий спосіб передачі повідомлень називається блоковим, а повідомлення ідентифікуються по блоку кодових символів або дискретних сигналів, що відображають ці символи. Кодер у цьому випадку виконує задачу перетворення блоку інформаційних символів в кодовий блок, що має властивість (за рахунок кодування) протистояти помилкам, що виникають в каналі передачі інформації.

В цьому випадку, користуючись додаванням в дискретному векторному просторі (порозрядним, за модулем основи коду M), для будь-якої моделі дискретного каналу можна записати

$$b_j^* = b_i + l_i,$$

де b_i і b_j^* – випадкові послідовності (кодові комбінації або блоки) з n символів на вході і виході каналу;

l_i – випадковий вектор помилок, який в загальному випадку залежить від b_i .

Часто недостатньо знати ймовірності поодиноких помилок (при посимвольному прийманні). Необхідно визначити статистичні особливості груп помилок, особливості їх групування в межах одного блоку переданого повідомлення.

Відомі моделі помилок в дискретних каналах передачі інформації відрізняються розподілом імовірності вектора l . Звернемо увагу на моделі помилок при двійковому кодуванні ($M = 2$) і симетричному каналі. У цьому випадку компоненти (розряди) вектора помилок приймають значення 0 і 1. Всяка 1 у векторі помилки означає, що у відповідному місці переданої послідовності символ прийнятий помилково, а 0 означає безпомилкове прийняття.

Двома найпоширенішими типами моделей джерел помилок є модель потоку незалежних помилок і модель статистично залежних потоків помилок.

Доцільність використання тієї чи іншої моделі джерела помилок визначається дослідженням статистичних властивостей дискретних каналів на основі тестових випробувань. У цьому випадку, на вхід каналу передачі інформації подають наперед підготовлену послідовність кодових блоків і, шляхом поелементного порівняння переданої послідовності з прийнятою, відтворюється послідовність помилок, яка називається потоком помилок.

Якщо в результаті статистичних випробувань каналу встановлено, що в потоці помилок безумовні частоти помилок p_0 дорівнюють умовним їх частотам, то такі канали вважаються каналами з незалежними помилками або каналами без пам'яті і їх моделюють однорідними двійковими послідовностями.

Символи в таких послідовностях характеризуються постійною за часом імовірністю $P(I_i = 1) = p_0$ і $P(I_i = 0) = 1 - p_0 = q_0$, а їх умовна ймовірність дорівнює безумовній

$$P(I_i = 1 / I_{i-1} = 0) = P(I_i = 1 / I_{i-1} = 1) = P(I_i = 1) = p_0;$$

$$P(I_i = 0 / I_{i-1} = 0) = P(I_i = 0 / I_{i-1} = 1) = P(I_i = 0) = p_0.$$

Закон розподілу кратностей (ваг) помилок для послідовностей символів довжиною n описується відомою в теорії ймовірності формулою Бернуллі, яка відома під назвою біноміального закону розподілу ймовірності

$$P_n(w_1) = C_n^{w_1} p_0^{w_1} (1 - p_0)^{n - w_1},$$

де $p_0^{w_1} (1 - p_0)^{n - w_1}$ – імовірність поєднання у векторі помилок w_1 одиничних і $n - w_1$ нульових символів;

$C_n^{w_1}$ – число різних поєднань помилок кратності w_1 .

Модель каналу з незалежними помилками є зручною для обчислення всіх характеристик каналу, проте в реальних умовах помилки є залежними і мають тенденцію до групування.

1.3.3. Кількість інформації, що передається по дискретному каналу передачі інформації

За відсутності в каналі передачі інформації завад, символи, що передаються і приймаються, пов'язані однозначно і кількість прийнятої інформації на виході каналу завжди дорівнює кількості інформації, переданої від джерела. Дія завад на сигнали, що передаються, призводить до руйнування і необоротної втрати частини інформації. Задача визначення інформаційних характеристик каналів в цих умовах є дуже важливою, оскільки практично не існує систем передачі без завад.

Повідомлення дискретного джерела перетворюються в дискретні сигнали, що поступають в канал. Сигнальні символи на вході каналу позначимо $b_1, b_2, \dots, b_i, \dots, b_M$, а на виході – $b_1^*, b_2^*, \dots, b_j^*, \dots, b_N^*$. У загальному випадку розмірності алфавітів на вході і виході каналу можуть не співпадати ($M \neq N$). Наприклад, на практиці зустрічаються канали, в яких $N = M + 1$, оскільки вихідний алфавіт містить додатковий символ стирання, який з'являється тоді, коли немає можливості надійно визначити переданий символ.

Якщо в каналі є завади, то символи, що передаються і приймаються, будуть пов'язані неоднозначно і в процесі передачі символ b_i з різним ступенем імовірності може перейти в будь-який з символів b_j^* і, навпаки, прийнятий символ b_j^* може з'явитися в результаті відправлення будь-якого з символів b_i .

Розглянемо способи інформаційного опису каналів за наявності в них завад. Спочатку розглянемо канал передачі інформації з боку приймача. При цьому відомі прийнятий символ b_j^* і відповідна ймовірність $P(b_j^*)$. З отриманням символу b_j^* можна припустити, що був посланий якийсь з символів $b_1, b_2, \dots, b_i, \dots, b_M$. Отже, ймовірність посилки символу b_i в канал характеризуватиметься вже не апіорною ймовірністю $P(b_i)$, а деякою умовною апостеріорною ймовірністю $P(b_i / b_j^*)$ – ймовірністю передачі символу b_i , якщо прийнятий символ b_j^* . Розглянута ситуація схематично зображена на рис 1.8.

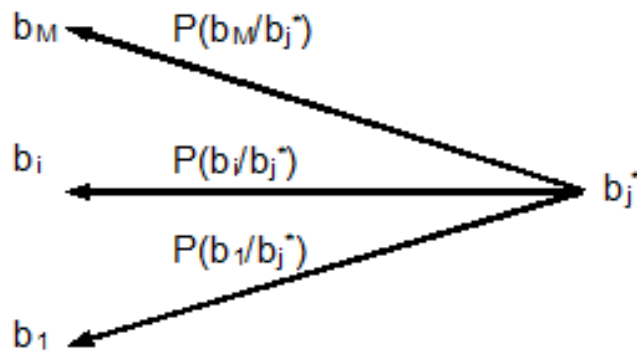


Рис. 1.8. Опис каналу передачі інформації з боку приймача

Оскільки в каналі з завадами, прийнятому символу b_j^* може відповідати передача одного з декількох символів b_i , то після приймання b_j^* залишається деяка невизначеність відносно переданого символу. Наприклад, прийнятий символ b_1 . Якби не було завад, то з упевненістю можна сказати, що був посланий символ b_1 . Але наявність завад позбавляє повної упевненості в цьому.

Ми все ще припускаємо, що був посланий символ b_1 , але й припускаємо можливість, що міг бути посланий і інший символ. Тому одержана інформація менша за ту, яку ми одержали б за відсутності завад на величину цієї невизначеності. Знаючи апостеріорну ймовірність $P(b_i / b_j^*)$ і ймовірність $P(b_j^*)$, можна знайти умовну ентропію $H(B / B^*)$

$$H(B / B^*) = - \sum_{j=1}^N p(b_j^*) \sum_{i=1}^M p(b_i / b_j^*) \log p(b_i / b_j^*).$$

Ця ентропія характеризує середню кількість інформації, що міститься у вхідному символі за умови, що вже стали відомі символи, що приймаються. Середня ж кількість інформації, що посиляється з кожним символом в канал, визначається апіорною ймовірністю і дорівнює

$$H(B) = - \sum_{i=1}^M P(b_i) \log P(b_i),$$

де ймовірність $P(b_i)$ може бути обчислена за формулою

$$P(b_i) = \sum_{j=1}^N P(b_i, b_j^*) = \sum_{j=1}^N P(b_j^*) P(b_i / b_j^*).$$

У загальному випадку знання послідовності вихідних символів знижує в середньому первинну невизначеність послідовності символів, що поступають у канал передачі інформації. Отже, середня кількість інформації, яку ще може принести одержувачеві один символ в каналі з завадами, дорівнює різниці між $H(B)$ і $H(B / B^*)$

$$I(B, B^*) = H(B) - H(B / B^*). \quad (1.19)$$

Величина $I(B, B^*)$ називається взаємною інформацією між множинами B і B^* . Вона характеризує середню кількість інформації, що міститься в символі множини B^* відносно символу множини B і є однією з основних інформаційних характеристик каналу з завадами.

Формула (1.19) показує, що середня взаємна інформація $I(B, B^*)$ дорівнює різниці між середньою кількістю інформації, що міститься в символі до і після приймання. При цьому величина $H(B)$ характеризує середню на символ кількість посланої в канал інформації, а $I(B, B^*)$ – середню на символ кількість прийнятої інформації. Умовну ентропію $H(B / B^*)$, рівну різниці між $H(B)$ і $I(B, B^*)$, можна вважати середньою кількістю втраченої кожним символом інформації. Її також називають ненадійністю каналу. Таким чином, формула (1.19) має таке просте фізичне значення: середня на символ кількість переданої по каналу передачі інформації з завадами інформації дорівнює різниці між кількістю інформації на вході каналу і кількістю інформації, втраченої в каналі через завади.

Розглянемо межі, в яких змінюється величина $I(B, B^*)$. Оскільки для умовної ентропії справедливе співвідношення $0 \leq H(B / B^*) \leq H(B)$, то $I(B, B^*)$ може приймати значення в діапазоні від $H(B)$ до нуля. При цьому рівність $H(B / B^*) = 0$, що відповідає $I(B, B^*) = H(B)$, має місце в тому випадку, коли при кожному b_j^* умовна ймовірність $P(b_i / b_j^*)$ передачі символу b_i дорівнює одиниці, а всі інші $P(b_i / b_j^*) = 0$. Це означає, що, знаючи послідовність вихідних символів $b^*(t)$, можна точно встановлювати послідовність переданих символів $b(t)$. Іншими словами, $b^*(t)$ містить повну інформацію про $b(t)$. Ця ситуація характерна для каналу, в якому завади відсутні. Інший крайній випадок, коли

$H(B / B^*) = H(B)$, має місце, якщо послідовності вхідних і вихідних символів незалежні. В цьому випадку знання $b^*(t)$ не зменшує невизначеності $b(t)$, тобто $b^*(t)$ не містить ніякої інформації про $b(t)$. Ця ситуація характерна для каналу з дуже сильними завадами або при обриві передачі інформації.

Розглянемо приклад. Двійковий канал передачі інформації з завадами описаний з боку приймача. Імовірності реєстрації символів b_1^* , b_2^* такі: $P(b_1^*) = 0,65$, $P(b_2^*) = 0,35$. Апостеріорна ймовірність посилки символів $P(b_1 / b_1^*) = 0,98$, $P(b_2 / b_1^*) = 0,02$, $P(b_1 / b_2^*) = 0,06$, $P(b_2 / b_2^*) = 0,94$. Обчислити середню на символ кількість інформації, що передається по цьому каналу передачі інформації.

Знаходимо умовну ентропію $H(B / B^*)$

$$H(B / B^*) = -\sum_i P(b_i^*) \sum_j P(b_j / b_i^*) \log P(b_j / b_i^*) = -0,65 \cdot (0,98 \cdot \log 0,98 + 0,02 \cdot \log 0,02) - 0,35 \cdot (0,94 \cdot \log 0,94 + 0,06 \cdot \log 0,06) = 0,210 \text{ біт/симв}$$

Обчислимо ймовірності $P(b_i)$ і ентропію $H(B)$

$$P(b_1) = \sum_j P(b_j^*) P(b_1 / b_j^*) = P(b_1^*) P(b_1 / b_1^*) + P(b_2^*) P(b_1 / b_2^*) = 0,65 \cdot 0,98 + 0,35 \cdot 0,06 = 0,66;$$

$$P(b_2) = 1 - P(b_1) = 1 - 0,66 = 0,34$$

$$H(B) = -\sum_{i=1}^2 P(b_i) \log P(b_i) = -0,66 \cdot \log 0,66 - 0,34 \cdot \log 0,34 = 0,925 \text{ біт/симв.}$$

Обчислимо $I(B, B^*)$:

$$I(B, B^*) = H(B) - H(B / B^*) = 0,925 - 0,210 = 0,715 \text{ біт.}$$

Розглянемо тепер опис каналу передачі інформації з боку передавача. В цьому випадку відомі переданий символ b_i , безумовна ймовірність $P(b_i)$ і умовна ймовірність $P(b_j^* / b_i)$, що характеризують ймовірність того, що при передачі символу b_i по каналу з завадами буде одержаний символ b_j^* . Ці ймовірності називаються перехідними. Така ситуація схематично зображена на рис. 1.9.

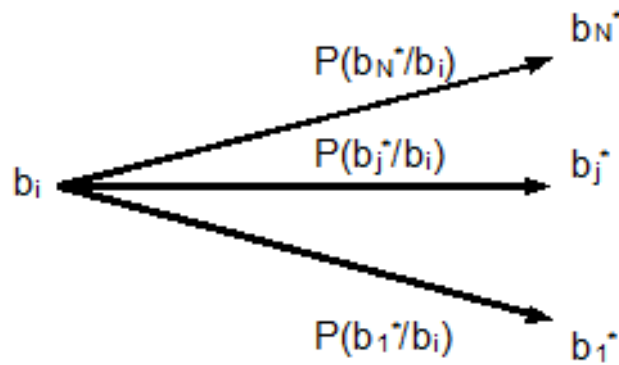


Рис. 1.9. Опис каналу передачі інформації з боку передавача

Обчисливши безумовну ймовірність $P(b_j^*)$ за формулою

$$P(b_j^*) = \sum_{i=1}^M P(b_i, b_j^*) = \sum_{i=1}^M P(b_i)P(b_j^* / b_i),$$

можна знайти ентропію вихідних символів каналу

$$H(B^*) = -\sum_{j=1}^N P(b_j^*) \log P(b_j^*).$$

Частина цієї інформації є корисною (інформація про вхідні символи b_i), а частина – помилковою (створеною завадами в каналі).

Знаючи ймовірність $P(b_i)$ і перехідну ймовірність $P(b_j^* / b_i)$ можна розрахувати умовну ентропію $H(B^* / B)$

$$H(B^* / B) = -\sum_{i=1}^M P(b_i) \sum_{j=1}^N P(b_j^* / b_i) \log P(b_j^* / b_i).$$

Величина $H(B^* / B)$ визначає інформацію, що міститься в послідовності вихідних символів $b^*(t)$ при відомій послідовності вхідних символів $b(t)$ з розрахунку на один символ. Оскільки вихідна послідовність відрізняється від вхідної виключно через завади в каналі, то $H(B^* / B)$ характеризує інформацію саме про завади в каналі і називається ентропією шуму. Отже, в середньому кожен символ доставляє одержувачеві кількість інформації, що визначається різницею між $H(B^*)$ і $H(B^* / B)$.

$$I(B, B^*) = H(B^*) - H(B^* / B). \quad (1.20)$$

Розглянемо приклад. Двійковий канал передачі інформації з завадами описаний з боку передавача. Імовірність символів $P(b_1) = 0,6$, $P(b_2) = 0,4$. Умовна імовірність переходів $P(b_1^* / b_1) = 0,95$, $P(b_2^* / b_1) = 0,05$, $P(b_1^* / b_2) = 0,1$, $P(b_2^* / b_2) = 0,9$.

Обчислити середню на символ кількість інформації, що передається по цьому каналу.

Знаходимо умовну ентропію $H(B^* / B)$.

$$\begin{aligned} H(B^* / B) &= -\sum_{i=1}^2 P(b_i) \sum_{j=1}^2 P(b_j^* / b_i) \log P(b_j^* / b_i) = \\ &= -0,6 \cdot (0,95 \cdot \log 0,95 - 0,05 \cdot \log 0,05) - \\ &\quad - 0,4 \cdot (0,10 \cdot \log 0,10 - 0,90 \cdot \log 0,90) = 0,360 \text{ біт}. \end{aligned}$$

Обчислюємо ймовірності $P(b_i^*)$ і ентропію $H(B^*)$.

$$\begin{aligned} P(b_1^*) &= \sum_{i=1}^2 P(b_i) P(b_1^* / b_i) = P(b_1) P(b_1^* / b_1) + P(b_2) P(b_1^* / b_2) = \\ &= 0,6 \cdot 0,95 + 0,4 \cdot 0,1 = 0,61; \end{aligned}$$

$$P(b_2^*) = 1 - P(b_1^*) = 1 - 0,61 = 0,39.$$

$$H(B^*) = -\sum_{j=1}^2 P(b_j^*) \log P(b_j^*) = -0,61 \cdot \log 0,61 - 0,39 \cdot \log 0,39 = 0,965 \text{ біт}.$$

$$\text{Отримаємо } I(B, B^*) = H(B^*) - H(B^* / B) = 0,965 - 0,360 = 0,605 \text{ біт}.$$

Формули (1.19) і (1.20) наочно зображені на рис. 1.10.

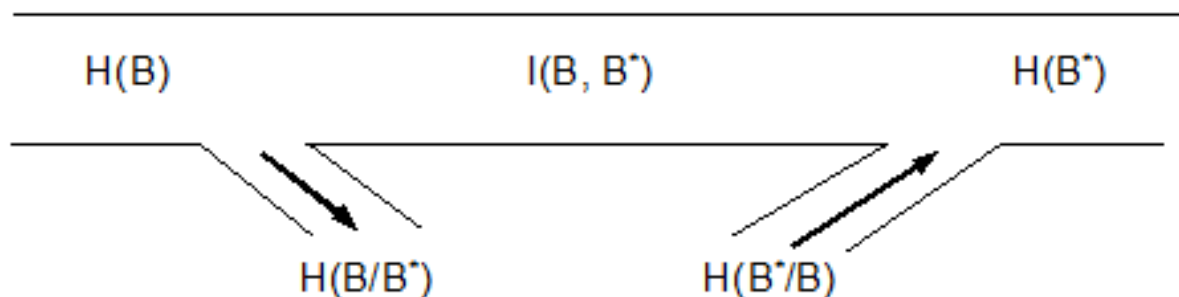


Рис. 1.10. Передача інформації по каналу передачі інформації з завадами

Тут $H(B)$ – середня кількість інформації, що поступає з кожним символом в канал, а $H(B^*)$ – середня кількість інформації, що доставляється кожним прийнятим символом.

Величина $H(B/B^*)$ є середньою на символ кількістю інформації, що втрачається з каналу; $H(B^*/B)$ – середня кількість інформації, що поступає в канал.

Кожний з наборів $P(b_i)$, $P(b_j^*/b_i)$ і $P(b_j^*)$, $P(b_i/b_j^*)$ є інформаційно повним, оскільки за допомогою будь-якого з них шляхом перетворень можна одержати всі інші інформаційні характеристики каналу.

Хай всі символи b_i , як і відповідні їм символи b_j^* , мають середню тривалість τ_k . Тоді в середньому за одиницю часу по каналу з завадами передається кількість інформації $I'(B, B^*)$, що дорівнює:

$$\begin{aligned} I'(B, B^*) &= \frac{I(B, B^*)}{\tau_k} = \\ &= v_k [H(B) - H(B/B^*)] = v_k [H(B^*) - H(B^*/B)], \end{aligned} \quad (1.21)$$

де $v_k = 1/\tau_k$ – швидкість посилки символів в канал.

Величина $I'(B, B^*)$ називається швидкістю передачі інформації по каналу.

1.3.4. Пропускна здібність дискретного каналу передачі інформації

У співвідношенні (1.21) з чотирьох ентропій одна – $H(B)$ визначається джерелом дискретних повідомлень, а інші три залежать як від властивостей джерела, так і від властивостей каналу. Отже, швидкість передачі інформації не може повною мірою охарактеризувати інформаційні можливості каналу. Необхідно знайти оцінку здібності самого каналу передавати інформацію.

Уявимо, що на вхід каналу можна подавати від джерела символи, розподіл імовірностей яких можна змінювати. При цьому, очевидно, що швидкість передачі інформації при одній і тій же швидкості посилки символів в каналі і одній і тій же завади буде різною. Максимально можлива швидкість передачі інформації при заданих властивостях каналу

передачі інформації називається пропускну здібністю каналу передачі інформації.

Пропускна здібність каналу C визначається шляхом максимізації виразу (1.21)

$$C = \max_{P(b_i)} I'(B, B^*) = v_k \max_{P(b_i)} I(B, B^*). \quad (1.22)$$

Максимізація здійснюється шляхом вибору оптимального розподілу ймовірностей $P(b_i)$ при заданих v_k і N .

Таким чином, на відміну від швидкості передачі інформації, яка залежить і від статистики джерела, і від властивостей каналу зв'язку, пропускна здібність каналу є характеристикою його самого. Вона не залежить від статистики джерела повідомлень і характеризує граничні можливості каналу відносно передачі середньої кількості інформації за одиницю часу.

Розглянемо приклад. Обчислити пропускну здібність дискретного каналу передачі інформації, якщо тривалість символів $\tau_k = 10^{-3}$ с, а умовна ймовірність переходів $P(b_1^* / b_1) = 0,999$, $P(b_2^* / b_1) = 0,01$, $P(b_1^* / b_2) = 0,3$, $P(b_2^* / b_2) = 0,7$.

Пропускна здібність цього каналу обчислюється шляхом максимізації величини $I(B, B^*)$ по всіх можливих статистиках $P(b_i)$. Отже, необхідно розрахувати швидкість передачі інформації при різних розподілах імовірності символів $P(b_1)$ і $P(b_2)$ та знайти максимальну. Методика розрахунку швидкості передачі така ж, як і в попередньому прикладі. Результати рішення надані в табл. 1.1.

Таблиця 1.1

Результати вирішення задачі максимізації величини $I(B, B^*)$

$P(b_1)$	$P(b_2)$	$P(b_1^*)$	$P(b_2^*)$	$H(B^*),$ біт	$H(B^*/B),$ біт	$I(B, B^*),$ біт	$I'(B, B^*),$ біт/с
0,4	0,6	0,58	0,42	0,9814	0,5294	0,452	452
0,5	0,5	0,65	0,35	0,9341	0,4411	0,493	493
0,6	0,4	0,72	0,28	0,8554	0,3531	0,502	502
0,7	0,3	0,79	0,21	0,7415	0,2651	0,476	476

Аналіз результатів показує, що пропускна здібність каналу передачі інформації, що досліджується, буде при $P(b_1) \approx 0,6$, $P(b_2) \approx 0,4$.

Якщо в каналі передачі інформації завади роблять однаковий вплив на всі символи, що передаються, то канал називається симетричним. У такому каналі ймовірність всіх помилкових переходів однакова, так само як однакова і ймовірність правильного прийняття всіх символів.

Визначимо пропускну здібність двійкового симетричного каналу передачі інформації з завадами, в який символи поступають із швидкістю v_k . Символи, що передаються, b_1 і b_2 незалежні. Імовірність перетворення в результаті дії завад кожного з них $P(b_j^* / b_i) = P_{\text{пом}}$. На рис. 1.11 зображені можливі варіанти передачі і прийняття символів по двійковому каналу передачі інформації.

Згідно виразу (1.22) маємо

$$C = v_k \max_{P(b_i)} [H(B^*) - H(B^* / B)].$$

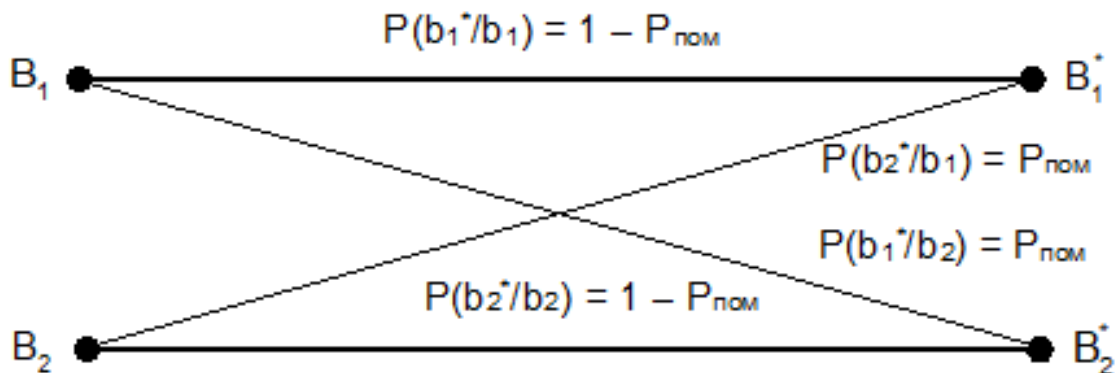


Рис. 1.11. Двійковий симетричний канал передачі інформації

Визначимо ентропію шуму

$$\begin{aligned} H(B^* / B) &= -\sum_{i=1}^2 P(b_i) \sum_{j=1}^2 P(b_j^* / b_i) \log P(b_j^* / b_i) = \\ &= -P(b_1) [(1 - P_{\text{пом}}) \log(1 - P_{\text{пом}}) + P_{\text{пом}} \log P_{\text{пом}}] - \\ &\quad -P(b_2) [P_{\text{пом}} \log P_{\text{пом}} + (1 - P_{\text{пом}}) \log(1 - P_{\text{пом}})] = \\ &= -[P(b_1) + P(b_2)] [P_{\text{пом}} \log P_{\text{пом}} + (1 - P_{\text{пом}}) \log(1 - P_{\text{пом}})] = \\ &= -P_{\text{пом}} \log P_{\text{пом}} - (1 - P_{\text{пом}}) \log(1 - P_{\text{пом}}). \end{aligned}$$

Виявляється, що з причини симетрії каналу передачі інформації умовна ентропія $H(B^* / B)$ не залежить від статистики вхідних символів, а

визначається тільки перехідною ймовірністю. Тому максимальна швидкість передачі інформації досягається тоді, коли прийняті символи b_1^* і b_2^* незалежні і рівноймовірні, що в свою чергу має місце при рівноймовірних символах, що передаються.

Отже, $H(B^*)_{\max} = 1$. Тоді пропускна здібність дорівнює:

$$C = v_k [1 + P_{\text{пом}} \log P_{\text{пом}} + (1 - P_{\text{пом}}) \log (1 - P_{\text{пом}})].$$

На рис. 1.12 зображена залежність C від імовірності помилки для двійкового каналу.

При $P_{\text{пом}} = 0$ пропускна здібність максимальна і дорівнює v_k . Збільшення $P_{\text{пом}}$ приводить до зниження пропускної здібності, яка стає рівною нулю при $P_{\text{пом}} = 0,5$. В цьому випадку повністю зникає яка-небудь залежність між символами, що передаються і приймаються. Значення $P_{\text{пом}} = 0,5$ є граничним для двійкового каналу. Випадок $P_{\text{пом}} = 0,5$ називається обривом каналу. Те, що пропускна здібність при $P_{\text{пом}} = 1$ така ж, як при $P_{\text{пом}} = 0$ (канал без перешкод), пояснюється тим, що при $P_{\text{пом}} = 1$ для правильного відновлення символів, що передаються, достатньо інвертувати всі вихідні символи.

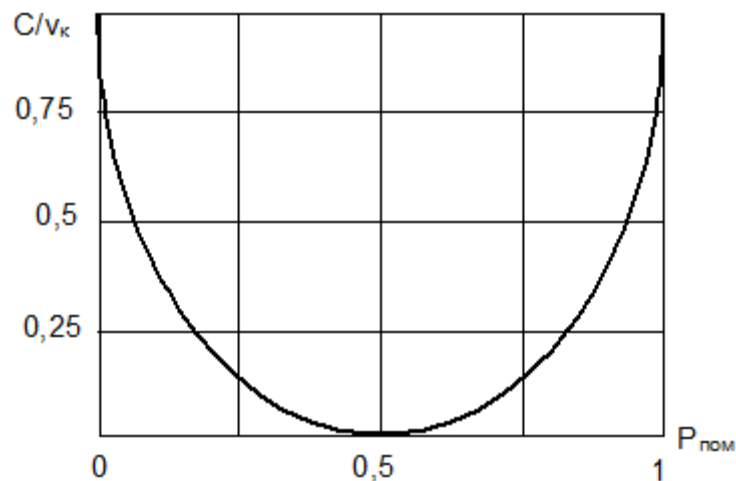


Рис. 1.12. Залежність пропускної здібності двійкового симетричного каналу від імовірності помилкового прийняття символу

З поняттям пропускної здібності каналу пов'язана одна з найважливіших теорем теорії інформації – теорема Шеннона про оптимальне кодування. Стосовно дискретного джерела теорема свідчить: якщо продуктивність джерела повідомлень менша пропускної здібності

каналу $H'(A) < C$, то існує спосіб оптимального кодування і декодування (перетворення повідомлень в сигнал на передачі і зворотного його перетворення в повідомлення на прийманні), при якому ймовірність помилки може бути зроблена як завгодно малою. Якщо $H'(A) > C$, то такого способу не існує.

Середня ймовірність помилки при оптимальному кодуванні обчислюється співвідношенням

$$P_{\text{пом}} = 2^{-T[C-H'(A)]},$$

де T – тривалість сигналу, відповідного послідовності символів джерела досить великої довжини n ; $C - H'(A)$ визначає запас пропускної здібності каналу.

Для дискретного каналу без шумів теорема оптимального кодування формулюється таким чином: якщо продуктивність джерела менша пропускної здібності каналу, то існують способи кодування і декодування, що забезпечують скільки завгодно високу надійність відповідності прийнятих комбінацій переданим.

1.4 Методи ефективного кодування інформації

Розв'язати задачу узгодження джерела повідомлень з каналом передачі інформації можна шляхом усунення надмірності повідомлень методом перетворення їх статистичних властивостей.

Процес перетворення дискретних повідомлень з однієї форми в іншу називається кодуванням. Якщо в результаті кодування надмірність повідомлень зменшується – таке кодування називається ефективним або статистичним (інакше – кодуванням для джерела); якщо надмірність зростає, кодування називається завадостійким (інакше – кодуванням для каналу).

У загальному випадку кодуванням називається відображення елементів (символів) дискретного повідомлення комбінаціями кодових символів. Спочатку символи x_i алфавіту X джерела повідомлень нумеруються, $X = \{x_1, x_2, \dots, x_i, \dots, x_M\}$. Так, наприклад, для букв українського алфавіту $M = 32$, тобто кожна з букв позначається одним з чисел: 0, 1, ... 31. Потім кожному з чисел приводиться в однозначну відповідність комбінація кодових символів. Якщо кодові символи можуть приймати m

різних значень (основа коду), то будь-яке число N можна представити n розрядами кодової комбінації, записавши його у вигляді багаточлена

$$N = \sum_{i=0}^{n-1} \gamma_i m^i,$$

де γ_i – цілі числа, що задовольняють умові $0 < \gamma_i < m - 1$; для двійкової системи ($m = 2$) коефіцієнти γ_i можуть приймати значення 0 або 1.

Для представлення будь-якого з чисел M потрібно $n = \log_2 M$ символів в кодовій комбінації. Умовимося нумерувати позиції символів в кодовій комбінації справа наліво

$$(\gamma_{n-1}, \dots, \gamma_0) \leftrightarrow \gamma_{n-1} 2^{n-1} + \dots + \gamma_0 2^0.$$

Тоді кодову комбінацію будь-якої з букв українського алфавіту можна представити сукупністю $n = \log_2 32 = 5$ кодових символів. Наприклад, для числа $v = 13$ маємо

$$v = 13 \leftrightarrow (01101) \leftrightarrow 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0.$$

Через взаємну однозначність перетворення символів повідомлення в кодові комбінації кількість останніх дорівнює кількості можливих символів алфавіту повідомлень. Якщо всі кодові комбінації мають однакову довжину n , то код називають рівномірним, а n – довжиною коду.

Надмірність повідомлень джерела призведе до надмірності кодових комбінацій, що врешті-решт призведе до зниження швидкості передачі інформації в каналі. Проте, теорема Шеннона доводить, що відповідним вибором способу кодування при будь-якій надмірності джерела повідомлень можна забезпечити швидкість передачі інформації по каналу без завад скільки завгодно близьку до його пропускної здібності.

Для оптимального узгодження джерела з каналом передачі інформації без завад необхідно, щоб середня кількість двійкових символів на букву джерела була не менша ентропії $H(X)$. Це очевидно, якщо символи джерела незалежні і рівноімовірні. Такі джерела не мають надмірності, а їх ентропія максимальна і дорівнює $\log M$.

Якщо $M = 2^n$, то застосувавши рівномірний код, можна кожний символ джерела закодувати n -значною кодовою комбінацією. Оскільки використовуються всі 2^n комбінацій, які так само, як і символи джерела, є

рівноймовірними і незалежними, то $n = \log M = H(X)$, тобто кількість символів коду на символ джерела дорівнює ентропії джерела.

Наприклад, якщо джерело видає чотири рівноймовірні букви ($H_0 = 2$ біт/симв), то ефективний код має вигляд: $x_1 = 00$, $x_2 = 01$, $x_3 = 10$, $x_4 = 11$. Кількість символів коду, що приходяться на одну букву джерела, дорівнює ентропії і ні при якому іншому способі кодування не може бути меншим. В цьому випадку можна говорити про повне узгодження джерела і каналу.

Нерівноймовірні і взаємозалежні символи також можна кодувати рівномірним кодом, витрачаючи на кожний з них $n = \log_2 M$ двійкових символів. Але таке кодування не буде ефективним.

Розглянемо джерела повідомлень з нерівноймовірними символами $P(x_1) = 0,5$, $P(x_2) = 0,25$, $P(x_3) = P(x_4) = 0,125$. Ентропія такого джерела $H_1(X) = 1,75$ біт/симв. Застосувавши рівномірний код ($x_1 = 00$, $x_2 = 01$, $x_3 = 10$, $x_4 = 11$), на кожну букву повідомлення витратимо два кодових символи. Але з теореми Шеннона виходить, що для даного джерела можливе економніше кодування з витратою на кожну букву в середньому 1,75 двійкових кодових символів.

Розроблено багато методів ефективного кодування для різних джерел. Близькі до граничних результати можна одержати, застосувавши, наприклад так зване, статистичне кодування. При такому методі використовуються нерівномірні коди. Символи джерела, що часто зустрічаються, відображаються короткими кодовими комбінаціями, а рідкісніші – довгими. При цьому вдається значно зменшити середню довжину кодової комбінації. До таких кодів належать ефективні коди Шеннона – Фано та Хаффмена.

1.4.1. Ефективне кодування Шеннона – Фано

Статистичне кодування незалежних символів джерела вперше було запропоноване Шенноном і Фано. Методика побудови коду наступна: символи алфавіту джерела виписуються в порядку спадання ймовірності. Потім вони розділяються на дві групи так, щоб суми ймовірностей в кожній з груп були б, якщо можна, однаковими. Першій групі присвоюємо кодовий символ 0 (або 1), а другий – 1 (або 0). Кожна з одержаних груп, в свою чергу, розбивається на дві підгрупи з однаковою сумарною ймовірністю і

знов першій підгрупі присвоюємо символ 0 (або 1), а другий – 1(або 0) і т.д.

Застосуємо цю методику для розглянутого раніше джерела з нерівноймовірними символами.

Після першого розбиття першій групі, до якої ввійшло повідомлення x_1 , як перший кодовий символ присвоюємо 0, а другий, куди ввійшли x_2 , x_3 і x_4 присвоюємо 1. Далі другу групу розбиваємо на дві рівноймовірні підгрупи і першій присвоюємо символ 0, а другий 1. А підгрупу, що залишилася, яка складається з двох повідомлень x_3 і x_4 , знову розбиваємо на дві частини і першої з них присвоюємо 0, а другий 1. Підсумкові кодові комбінації мають вигляд: $x_1 = 0$, $x_2 = 10$, $x_3 = 110$, $x_4 = 111$. Приклад кодування Шеннона – Фано зображений на рис. 1.13. При такій процедурі, дійсно, найбільш імовірні повідомлення одержують найкоротші кодові позначення, а найменш імовірні – довгі.

Середня кількість кодових символів $\bar{n}$, що приходяться на букву повідомлення при такому кодуванні

$$\bar{n} = \sum_{i=1}^M n_i P(x_i) = \sum_{i=1}^4 n_i P(x_i) = 1 \cdot 0,5 + 2 \cdot 0,25 + 3 \cdot 0,125 + 3 \cdot 0,125 = 1,75$$

дорівнює ентропії джерела.

x_i	$P(x_i)$	Кодування			Код
x_1	0,5	} 1	–	–	1
x_2	0,25	} 0	} 1	–	01
x_3	0,125		} 0	} 1	001
x_4	0,125			} 0	000

Рис. 1.13. **Метод кодування Шеннона – Фано**

Таким чином, побудований код реалізує граничні результати, вказані теоремою, і в цьому значенні може бути названий оптимальним статистичним кодом.

Якщо ймовірність появи символів джерела не є степенем двійки, то неможливо розбити їх на групи з рівною сумарною ймовірністю. Тоді проводять наближене розбиття, але при цьому одержані результати

будуть гіршими. Для поліпшення ефективності коду в цьому випадку необхідно переходити від кодування поодиноких символів до кодування повідомлень, складених з великої кількості символів (блокове або укрупнене кодування).

Приклад. Розглянемо цей принцип на прикладі кодування повідомлень джерела, алфавіт якого складається всього з двох букв x_1 і x_2 з імовірністю появи $P(x_1)=0,89$, $P(x_2)=0,11$. При по буквеному кодуванні на передачу кожної букви потрібен один кодовий символ – або 0, або 1, тоді як ентропія джерела дорівнює приблизно 0,5 біт. При поблочному кодуванні відбувається як би збільшення первинного алфавіту. При кодуванні по дві букви в блоці алфавіт укрупненого джерела матиме вигляд: x_1x_1 , x_1x_2 , x_2x_1 , x_2x_2 . Оскільки букви статистично не зв'язані, ймовірність блоків визначається добутком імовірності складових букв: $P(x_1x_1)=0,792$, $P(x_1x_2)=P(x_2x_1)=0,098$, $P(x_2x_2)=0,012$. Кодові комбінації, одержані за методикою Шеннона – Фано, мають вигляд: $x_1x_1=0$, $x_1x_2=10$, $x_2x_1=110$, $x_2x_2=111$.

Середня кількість кодових символів на блок $\bar{n}=1 \cdot 0,792 + 2 \cdot 0,098 + 3 \cdot 0,098 + 3 \cdot 0,012 = 1,318$, а на одну букву дорівнює 0,659.

Кодування блоків, що містять по три букви, дає ще більший ефект. Кількість кодових символів на блок дорівнює 1,658, а на одну букву 0,552, що всього на 11 % більше ентропії джерела. Збільшуючи і далі число символів в блоці, можна скільки завгодно наблизитися до теоретичного мінімуму – ентропії джерела.

Збільшення ефективності при статистичному кодуванні шляхом укрупнення блоків не пов'язане з урахуванням все більш далеких міжсимвольних залежностей, оскільки розглянуте джерело має алфавіт з некорельованими символами. Підвищення ефективності визначається лише тим, що розширений набір імовірностей, що виходить при укрупненні, дозволяє ділити блоки на підгрупи, сумарна ймовірність яких все менше і менше відрізняється одна від одної. Це і дає можливість скільки завгодно близько підійти до межі. Проте, укрупнення – це універсальний спосіб зменшення надмірності не тільки при нерівномірності символів, але і при їх взаємній залежності. Дійсно, якщо символи джерела об'єднати в довгі повідомлення (блоки), то від символів попереднього блоку залежать тільки декілька символів подальшого блоку і ця залежність

зменшуватиметься з віддаленням символу від початку блоку. Чим більше символів в блоці, тим більше буде символів, які не залежать від попередніх блоків. Відбувається декореляція символів, тобто усувається їх взаємозалежність, і при дуже великій довжині блоку укрупнений алфавіт складатиметься з незалежних один від одного повідомлень.

Наочно значення укрупнення може бути пояснено на прикладі буквеного тексту. Якщо ймовірнісні зв'язки між буквами в будь-якій мові відносно сильні, то між словами вони менші, ще менші між фразами, абзацами. Тому, застосовуючи кодування слів, фраз, абзаців, можна достатньо повно усунути надмірність, обумовлену ймовірнісними зв'язками.

Блокове кодування дозволяє побудувати оптимальний код для будь-якого джерела. Це забезпечує збільшення швидкості передачі, але його використання призводить до запізнення отримання інформації. На передавальній стороні необхідно спочатку накопичити весь блок, а потім здійснити його кодування, аналогічно на приймальній стороні декодування також може бути здійснене тільки після накопичення всього блока.

1.4.2 Ефективне кодування Хаффмана

Іншим найпоширенішим алгоритмом, на основі якого будуються нерівномірні коди, є алгоритм Хаффмана, що дозволяє будувати нерівномірний код, який може давати найкоротшу середню довжину коду для алфавіту, що використовують.

Процес кодування починається з переліку вхідних символів разом з їх імовірністю (або частотами появи) в порядку спадання. Кожній гілці дерева присвоюється її ваговий коефіцієнт, рівний імовірності цієї гілки. Два входи з найнижчою ймовірністю об'єднуються, щоб утворити нову гілку з їх загальною ймовірністю. Після кожного об'єднання гілки переупорядковуються, при цьому нова гілка ставиться зверху гілок з рівною ймовірністю. Приклад кодування Хаффмана зображений на рис. 1.14.

Після формування дерева, щоб розрізняти дві гілки кожній гілці присвоюється символ "1" або "0". Для кожного символу вхідного алфавіту записується відповідний йому код за напрямом гілки від її кінця. Для цього прикладу середня довжина коду дорівнює 2,4 біт на символ. Для порівняння рівномірний код повинен був мати 3 символи.

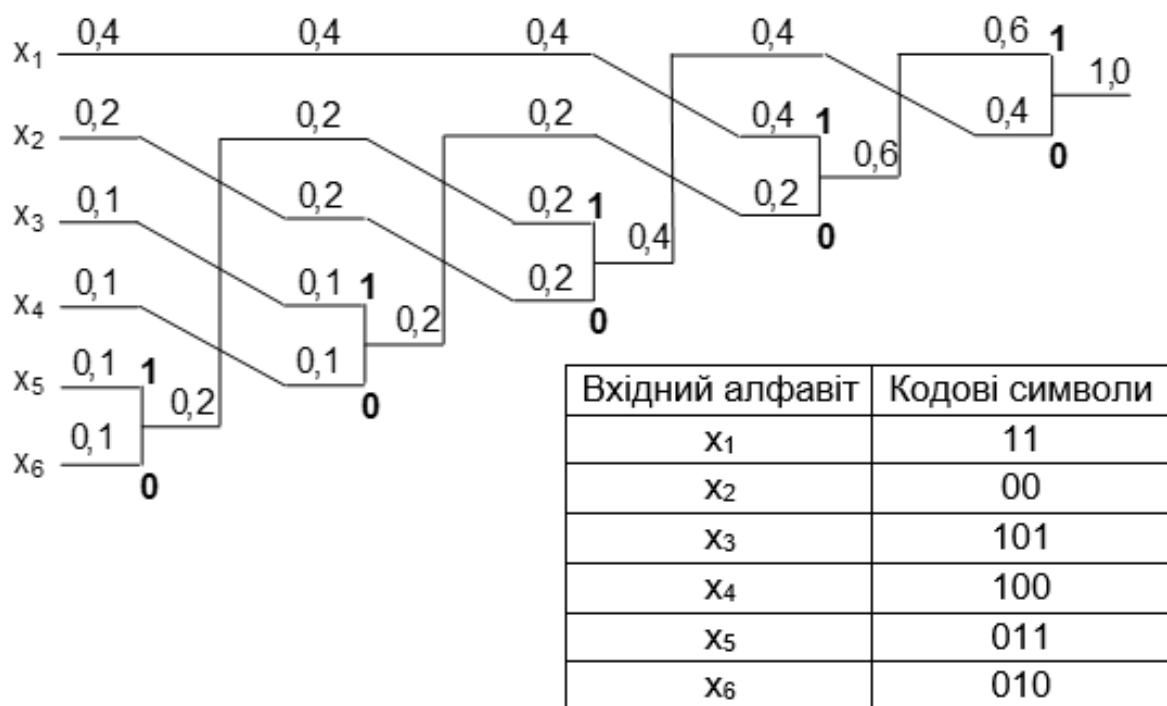


Рис. 4.10. **Дерево кодування Хаффмана**

Ефективність кодування Хаффмана зростає зі збільшенням кількості символів вхідного алфавіту. Розглянемо приклад, зображений на рис. 1.14.

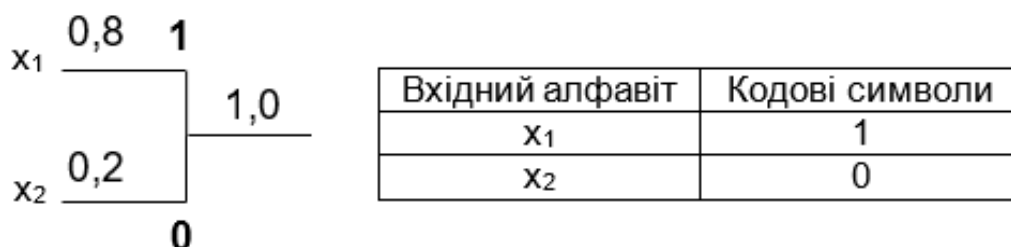


Рис. 1.14. **Кодування Хаффмана без укрупнення символів**

Для цього прикладу середня довжина коду становить 1 біт на символ і дорівнює довжині рівномірного коду. Щоб поліпшити ефективність кодування слід укрупнити алфавіт джерела шляхом вибору двох символів, імовірність яких буде дорівнювати добутку імовірностей символів, що зображено на рис. 1.15.

Для цього прикладу середня довжина коду дорівнює 1,56 біт на символ. Для порівняння рівномірний код повинен був мати 2 символи.

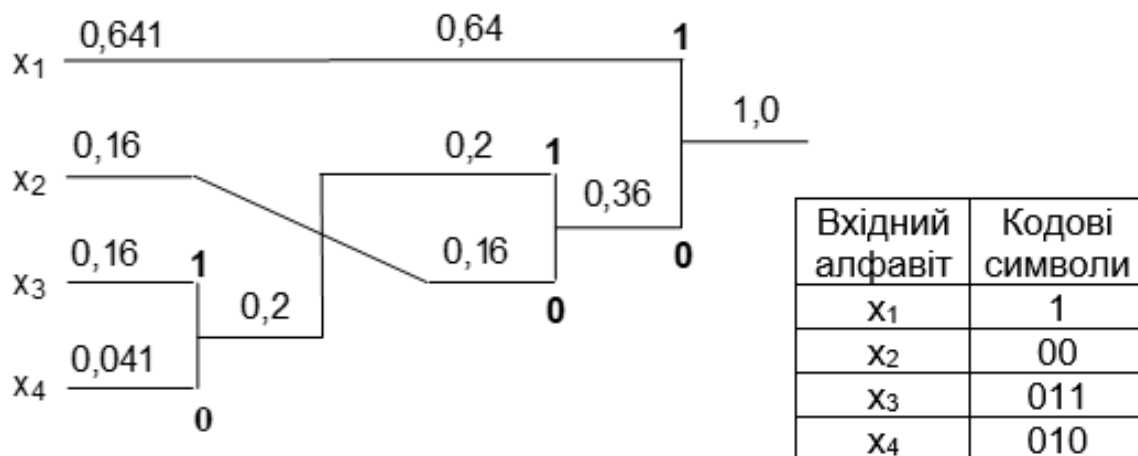


Рис. 1.15. Кодування Хаффмана з укрупненням символів

Існують такі методи кодування, що забезпечують усунення або значне зменшення надмірності, пов'язаної з взаємозалежністю близько розташованих елементарних повідомлень.

Методи ефективного кодування, що дозволяють записувати повідомлення таким числом двійкових символів, яке трохи відрізняється від ентропії джерел повідомлень, є одним з основних розділів сучасної теорії кодування. Останнє обумовлене великим прикладним значенням ефективного кодування, особливо в космічних лініях передачі інформації. Ефективне кодування дозволяє записати і передати повідомлення за допомогою меншого числа двійкових символів. Разом з тим не завжди потрібно прагнути до повного усунення надмірності. Деяка надмірність, особливо якщо вона відповідним чином введена в повідомлення, корисна для підвищення завадостійкості передачі інформації.

1.5. Основні визначення та поняття теорії скінченних полів Галуа

При розв'язанні задачі синтезу й аналізу дискретних сигналів і кодів широко використовується апарат алгебри, і, зокрема, теорії скінченних полів. Розв'язання цих задач ґрунтується на виконанні алгебраїчних операцій над сукупністю довільних елементів множин, що утворюють кінцеві поля.

Багато кодів, що виправляють помилки, засновані на структурах кілець многочленів і полів Галуа. Крім того, ці алгебраїчні поняття й методи є необхідним робочим інструментом для конструювання кодерів і декодерів.

Досліджувані в сучасній алгебрі арифметичні системи класифікуються відповідно до ускладнення їх математичної структури.

Демо наступні формальні поняття:

абелева група – множина математичних об'єктів, які можна додавати й віднімати;

кільце – множина математичних об'єктів, які можна додавати, віднімати, множити;

поле – множина математичних об'єктів, які можна додавати, віднімати, множити й ділити.

У полі, що складається з 2 елементів, виконаємо найпростіші операції:

$$\begin{aligned}0 + 0 = 0; 0 + 1 = 1; 1 + 0 = 1; 1 + 1 = 0; 0 \times 0 = 0; \\ 0 \times 1 = 0; 1 \times 0 = 0; 1 \times 1 = 1.\end{aligned}$$

Операції додавання й множення в полі називаються додаванням і множенням за модулем 2.

При цьому з $1 + 1 = 0$ випливає, що $-1 = 1$, а з рівняння $1 -$ що $1^{-1} = 1$.

Поле із двох елементів має алфавіт 0 і 1.

1.5.1. Групи

Множина елементів або об'єктів в G , що утворюють алгебраїчну систему, на якій визначена деяка операція (x) і виконуються аксіоми 1,2,4 і 5 (табл. 1.2), називається *групою* [4; 5].

Надалі групу G з визначеною на ній операцією (x) будемо позначати $\langle G, x \rangle$. З визначення поняття групи випливає, що для будь-яких двох елементів $a, b \in G$ завжди однозначно визначається елемент $(a \times b) \in G$, тобто виконується аксіома замкненості, а також аксіома асоціативності операції (аксіома 2); аксіоми 4 і 5 вказують на існування відповідно одиничного e та зворотного $\bar{a}$ елементів.

Група $\langle G, x \rangle$ називається комутативною (абелевою), якщо крім аксіом 1, 2, 4 і 5 на ній також виконується аксіома комутативності (3). Відповідно до введених визначень групу $\langle G, x \rangle$, тобто групу з уведеною на ній операцією додавання, будемо називати комутативною адитивною групою, а групу $\langle G, x \rangle$ з операцією множення – комутативною мультиплікативною групою [5].

Група, що містить кінцеву кількість елементів q , називається кінцевою групою. Кінцева група позначається $\langle \{a, b, c, \dots\}, x \rangle$, де $a, b, c, \dots$

$\in G$ – кінцеві кількості елементів. Порядком кінцевої групи називається кількість елементів q групи. Із сказаного випливає, що всі елементи будь-якого кінцевого поля утворюють адитивну групу, тому порядок адитивної групи поля q завжди збігається з порядком поля q , мультиплікативна ж група поля не містить нульового елемента, тому мультиплікативна група поля порядку q має порядок $q-1$.

Приклад Покажемо, що множина елементів $S_5 = \{0, 1, 2, 3, 4\}$ з уведеною на ній операцією додавання (+) утворює групу $\langle S_1, + \rangle$.

Розв'язання. Оскільки група має порядок 5, що необхідно для виконання аксіоми G1 операцію додавання слід виконувати за $\text{mod } (S = 5)$. У табл. 1.2 наводиться правило додавання елементів.

Таблиця 1.2

Додавання елементів

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

З табл. 1.2 випливає, що операція "+" не є звичайним додаванням. Якщо б операція "+" була звичайним десятковим додаванням, то аксіома замкненості не виконувалась б. Безпосередньою перевіркою можна показати, що для $S_5 = \{0, 1, 2, 3, 4\}$ виконуються аксіоми G 1-G5, тому $\langle \{0, 1, 2, 3, 4\}, + \rangle$ є комутативною групою.

1.5.2. Підгрупи

Нехай $\langle G, x \rangle$ – група, а H – множина елементів $\{h_1, h_2, \dots, h_n\}$, причому $h_i \in G$, що є групою щодо групової операції (x), тоді множина H називається підгрупою групи G [4]. Відповідно до введеного визначення група G і група H , що складається з одиничного елемента e групи G , є підгрупами групи G . Всі підгрупи G , за винятком зазначених (якщо вони існують), називаються власними. Кількість елементів підгрупи, як і кількість елементів групи, будемо називати *порядком підгрупи*.

Приклад 2.2. Нехай $G = \langle \{0,1,2,3,4,5\}, + \rangle$ – група порядку 6, що задається табл. 2.2, а $H = \langle \{0,3\}, + \rangle$ – група, що задається табл. 2.3. Переконаємося, що H – підгрупа групи G порядку 2.

Розв’язання. Безпосередньою перевіркою переконуємося, що для множини $\{0,3\}$ виконуються аксіоми $G1 - G5$. Дійсно, сума елементів $3 + 0, 0 + 3, 0 + 0$ і $3 + 3 = 0$ належить множині $\{0,3\}$. Для $\{0,3\}$ виконуються аксіоми комутативності та асоціативності, елемент 0 – суть одиничний, оберненим до елемента 3 є сам елемент 3, тому що $3 + 3 = 0$ (e). Підгрупа $H = \{0, 2\}$ має порядок 2, тому що містить два елементи.

1.5.3. Кільця

Кільцем $\langle R, +, \cdot \rangle$ називається множина елементів R , на якому визначені операції додавання та множення й виконуються наступні аксіоми [4; 5].

$R1. \langle R, + \rangle$ – комутативна група; $R2. \langle R, \cdot \rangle$ – підгрупа, тобто алгебраїчна система, для якої виконуються аксіоми 1, 2 (табл. 2.2).

Таблиця 1.3

Комутативна група над $GF(5)$

+	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

Для будь-яких трьох елементів $a, b \in R$ справедливий дистрибутивний закон:

$$a(b + c) = ab + ac;$$

$$(a + b)c = ac + bc.$$

Слід зазначити, що в кільці для операції $(\times)$ можуть не виконуватися аксіоми $M4, M5$ і $M3$. Якщо ж операція $(\times)$ комутативна в кільці, то таке кільце називається комутативним. Якщо в кільці існує одиничний елемент щодо операції $(\times)$, тобто виконується аксіома $M4$, то кільце називається кільцем з одиницею.

Підмножина і кільце R називається ідеалом кільця R , якщо воно задовольняє наступні дві умови:

для будь-яких двох елементів a і b з підмножини і елемент $a + b$ належить і;

для будь-якого елемента x кільця R та будь-якого a з підмножини і обидва добутки a_x і x_a належать і.

1.5.4. Підполе і суміжні класи

Наведемо визначення таких понять, як підполе та суміжні класи, які будуть необхідні надалі [5; 6].

Підмножина S_0 елементів поля Галуа $S = GF(p^n)$ називається *підполем*, якщо для нього виконуються аксіоми поля 1 – 6.

З визначення випливає, що підполе S_0 саме є простим $GF(p)$ або розширенням $GF(p^k)$ полем, однак оскільки S_0 включає тільки частину елементів поля S , то завжди виконується умова $k < n$. Крім того, порядок поля завжди кратний порядку підполя, інакше k завжди ділить n , а будь-який ненульовий елемент поля a належить своєму підполю тоді і тільки тоді, якщо p^{k-1} кратне ε , де ε – період елемента в мультиплікативній групі поля $GF(p^n)$.

Знайдемо всі підполя поля Галуа $GF(5^2)$. Для цього поля дільниками показника степеня поля $n = 2$ є числа 1 і 2, тому поле $GF(5^2)$ містить одне підполе $GF(5) = \{1, 2, 3, 4\}$.

Множина елементів вигляду $b_j \times h_j$, де h_j є елементи підгрупи $H = \{h_1, h_2, \dots, h_k\}$ групи $G = \{b_1, b_2, \dots, b_n\}$, а b_i – фіксований елемент групи G , називається *суміжним класом* по H [8] і позначається у вигляді bH [8].

Елемент b_i , що належить суміжному класу $\{b_i H\}$, називається *твірним елементом* цього суміжного класу. Як твірний елемент може бути обраний будь-який елемент $b \in \{b_i H\}$. З визначення випливає, що суміжний клас за H і підгрупа G містять однакову кількість елементів.

1.5.5. Скінченні поля, побудовані за кільцем цілих чисел

Множина, що складається з p елементів, кожний з яких становить собою позитивне ціле число за модулем простого числа p , яке не перевищує p , називається простим полем Галуа й позначається $GF(p)$.

Для побудови і дослідження властивостей простого поля скористаємося поняттям порівняння чисел x і y за модулем m . Порівняння цілих чисел x і y за модулем m [6]

$$x \equiv y(\text{mod } m), \quad (1.23)$$

еквівалентне рівності

$$x - y = km. \quad (1.24)$$

для деякого цілого числа k . Відомо [6], що всі цілі числа x такі, що:

$$x \equiv r(\text{mod } m) \quad (1.25)$$

при фіксованому r утворюють клас відрахувань за модулем m , позначаються $\{r\}$. З наведеного визначення випливає, що всім числам класу $\{r\}$ відповідає той самий остаток r , а всі числа класу можна одержати, якщо у виразі $km+r$, k набуває значень цілих чисел, як додатних, так і від'ємних. Крім того, з виразів (1.23) і (1.25) випливає, що всього існує m різних значень r , тому всього буде m класів чисел за модулем m . Кожне із чисел класу є відрахуванням за модулем m . Якщо ж взяти від кожного класу по одному найменшому від'ємному відрахуванню $0, 1, 2, \dots, m-1$, то одержимо повну систему відрахувань.

З відомих властивостей порівняння (1.25), визначеного у вигляді (1.23) і (1.24), видно, що якщо:

$$x \equiv b(\text{mod } m) \text{ і } y \equiv c(\text{mod } m),$$

а b і c є елементами повної системи відрахувань за модулем m , то

$$x + y \equiv (b + c)(\text{mod } m); \quad (1.26)$$

$$x \times y \equiv (b \times c)(\text{mod } m) \quad (1.27)$$

над множиною елементів, що становлять повну систему відрахувань, визначені операції додавання і множення за модулем m :

$$\{b\} + \{c\} = \{b + c\}; \quad (1.28)$$

$$\{b\} \times \{c\} = \{b \times c\}. \quad (1.29)$$

Безпосередньо перевіркою можна показати, що всі аксіоми, наведені в табл. 1.2, крім M_5 , виконуються при довільних m , однак аксіома M_5 виконується тільки за умови $m = p$, де p – просте число.

Дійсно, якщо m є не просте число, то звичайно існують дільники нуля, *наприклад*, при $m = 10$:

$$4 \times 5 \equiv 0 \pmod{m},$$

при $m = 6$

$$4 \times 3 \equiv 0 \pmod{6}$$

і т. д., тому аксіома M5 не виконується.

З викладеного випливає, що повна система відрахувань за модулем простого числа p утворить кінцеве поле $GF(p)$ порядку p .

Приклад. Знайти елементи поля $GF(13)$.

Розв'язання. Елементами простого поля Галуа $GF(13)$ є цілі числа 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12.

Приклад. Побудувати таблиці додавання і множення в полі $GF(7)$.

Розв'язання. Елементами поля $GF(7)$ є цілі числа 0, 1, 2, 3, 4, 5, 6. Правила додавання і множення елементів у поле $GF(7)$ наведені в табл. 1.4 і 1.5.

Приклад. Знайти елемент поля $GF(7)$ $\bar{x}$, зворотний за додаванням елемента $x = 4$.

Розв'язання. Використовуючи аксіому G5, одержимо:

$$x + \bar{x} = 0, \text{ а } \bar{x} = -x = -4 + 7 = 3.$$

Приклад. Знайти елементи поля $GF(11)$, зворотні по множенню елементам 3 і 9.

Розв'язання. Використовуючи аксіому M5, одержимо:

$$\bar{x} \times 9 = 1 \pmod{11}.$$

Таблиця 1.4

Додавання в полі $GF(7)$

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

Множення в полі GF (7)

x	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	1	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Підставляючи послідовно в рівняння елементи поля GF(13), тобто 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 і 12, знайдемо єдиний розв'язок $\bar{x}=3$.

1.5.6. Скінченні поля, побудовані за кільцем многочленів

Елементами множини, що становить *розширене поле Галуа* GF (p^n), є поліноми $A(x)$ степеня n , коефіцієнти яких набувають значень над полем GF (p), тобто 0, 1, 2, ..., $p - 2$, $p - 1$, причому p – просте число.

Для побудови і дослідження елементів розширеного поля GF (p^n) уведемо аналогічно до виразів (1.23), (1.24) визначення порівняння для поліномів $A(x)$ над полем GF (p). Насамперед слід зауважити, що поліном

$A(x) = \sum_{j=0}^n \alpha_j x^j$ називається поліномом над полем GF (p), якщо його коефіцієнти α_j належать полю GF (p).

Порівняння для поліномів $A(x)$ і $B(x)$ за модулем полінома $F(x)$ над полем GF (p) [6] має вигляд:

$$A(x) \equiv B(x) \pmod{F(x)}, \quad (1.30)$$

що еквівалентно рівнянню:

$$A(x) - B(x) = K(x)F(x), \quad (1.31)$$

для деякого полінома $A(x)$. Оскільки порівняння у виразі (1.30) виконується за модулем p , то будемо його записувати як:

$$A(x) \equiv B(x) \pmod{F(x), p} \quad (1.32)$$

і говорити, що поліном $A(x)$ порівнюємо з поліномом $B(x)$ за подвійним модулем $(F(x), p)$.

Як і в простому полі $GF(p)$, поліноми $A(x)$ над полем $GF(p)$ задовольняють умову:

$$A(x) \equiv R(x) \pmod{F(x), p} \quad (1.33)$$

при заданому фіксованому $R(x)$ утворюють клас поліномів за подвійним модулем $(F(x), p)$, що позначимо $\{R(x)\}$. З виразу (1.33) випливає, що якщо $F(x)$ – поліном степеня над полем $GF(p)$, то будь-які остатки $R(x)$ є поліномами степеня не вище $n-1$, тобто:

$$R(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \dots + a_1x + a_0, \quad (1.34)$$

причому, якщо коефіцієнти a_i є елементами поля $GF(p)$, то існує всього p^n різних поліномів вигляду (1.34).

Для поліномів аналогічно (1.26) і (1.27) визначені операції додавання і множення у вигляді :

$$A(x) + B(x) \equiv (R(x) + P(x)) \pmod{F(x), p}; \quad (1.35)$$

$$A(x) \cdot B(x) \equiv (R(x) \cdot P(x)) \pmod{F(x), p}. \quad (1.36)$$

Можна показати [8], що дані аксіоми $G1-G5$, $M1-M5$ і $D1, D2$ виконуються для додавання й множення при довільному $F(x)$ над полем $GF(p)$. Однак, як і при порівнянні цілих чисел, $M5$ виконується тільки за умови, якщо $F(x) = f(x)$ – незвідний поліном [8] над полем $GF(p)$. Надалі поліном $f(x)$ степеня $n \geq 1$ з коефіцієнтами з поля $GF(p)$ будемо називати незвідним [8] над полем $GF(p)$, якщо не існує таких двох поліномів $g(x)$ і $h(x)$ над полем $GF(p)$, кожний степеня менше n , для яких виконується умова:

$$f(x) = g(x)h(x). \quad (1.37)$$

Відзначимо основну властивість незвідних поліномів, яка полягає в тому, що якщо поліном незвідний, то $f(x)$ ділить добуток $A(x) \times B(x)$, або ділить $A(x)$ або ділить $B(x)$.

Таким чином, якщо $f(x)$ – незвідний над $GF(p)$, то, взявши від кожного класу $\{R(x)\}$ по одному відрахуванню, одержимо повну систему

відрахувань. Найбільш зручно використовувати повну систему відрахувань p^n поліномів вигляду (1.34).

Із сказаного випливає, що повна система відрахувань за подвійним модулем $(f(x), p)$ утворює кінцеве поле, що містить p^n елементів, які виходячи з роботи [4], будемо позначати і називати розширеним полем або розширенням степеня n простого поля $GF(p)$.

Насамкінець ще раз підкреслимо, що елементи поля $GF(p^n)$ – поліноми степеня не вище $n-1$ з коефіцієнтами з поля $GF(p)$.

Приклад. Знайти елементи розширеного поля $GF(3^2)$.

Розв'язання. Відповідно до виразу (2.12) $n = 2$, тоді значення елементів поля $GF(3)$ одержимо як $GF\{3^2\} = \{0, 1, 2, x, 2x, x+1, 2x+1, x+2, 2x+2\}$.

Приклад. Побудуємо таблицю додавання і множення для елементів поля $GF(3^2)$.

Розв'язання. Використовуючи роботу [4], виберемо незвідний над полем $GF(3)$ поліном степеня $n = 2$ вигляду $f(x) = x^2 - 2$. Скориставшись елементами поля $GF(3^2)$, знайденими в прикладі, складемо таблиці додавання і множення в поля $GF(3^2)$.

Таблиця 1.6

Додавання в полі $GF(3^2)$

+	0	1	2	x	2x	x+1	2x+1	x+2	2x+2
0	0	1	2	x	2x	x+1	2x+1	x+2	2x+2
1	1	2	0	x+1	2x+1	x+2	2x+2	x	2x
2	2	0	1	x+2	2x+2	x	2x	x+1	2x+1
x	x	x+1	x+2	2x	0	2x+1	1	2x+2	2
2x	2x	2x+1	2x+2	0	x	1	x+1	2	x+2
x+1	x+1	x+2	x	2x+1	1	2x+2	2	2x	0
2x+1	2x+1	2x+1	2x	1	x+1	2	x+2	0	x
x+2	x+2	x	x+1	2x+2	2	2x	0	2x+1	1
2x+2	2x+2	2x	2x+1	2	x+2	0	x	1	x+1

Перш ніж звести таблицю множення елементів у полі $GF(3^2)$, розглянемо правило знаходження добутку елементів на ряді прикладів. Наприклад, знайдемо добуток елементів $2x + 2$ і $2x + 1$. Виконуючи множення, одержимо $(2x + 2) \times (2x + 1) = 4x^2 + 6x + 2$.

Приводячи отриманий вираз за mod 3, маємо:

$$4x^2 + 6x + 2 = x^2 + 2.$$

Елементами розширеного поля $GF(3^2)$ є поліноми степеня $n < 2$, поліном $x^2 + 2$ у модулі незведеного полінома $f(x) = x^2 - 2$. Для цього, замінивши операцію віднімання еквівалентною операцією додавання, визначимо з (2.9) $B(x)$ як $B(x) = A(x) + K(x) \times F(x)$.

Для $A(x) = x^2 + 2$; $F(x) = x^2 - 2$. Отже, невідомими є поліноми $B(x)$ і $K(x)$. Для знаходження $K(x)$ врахуємо той факт, що найбільший степінь $B(x)$ буде меншим від n у тому випадку, якщо сума коефіцієнтів при $x^2 A(x)$ і $K(x) \times F(x)$ буде дорівнювати p . Розділивши $x^2 + 2$ на поліном $x^2 - 2$ з урахуванням того, що операція додавання виконується за модулем 3, одержимо:

$$\begin{array}{r|l} x^2 + 2 = A(x) & x^2 - 2 = f(x) \\ 2x^2 - 4 & 2 = K(x) \\ \hline -2 = 1 = B(x) & \end{array}.$$

Аналогічно для знаходження добутку елементів $x + 1$ і $2x$ маємо

$$(x + 1)2x = 2x^2 + 2x:$$

$$\begin{array}{r|l} 2x^2 + 2x = A(x) & x^2 - 2 = f(x) \\ x^2 - 2 & 1 = K(x) \\ \hline 2x - 2 = 2x + 1 = B(x) & \end{array}.$$

Аналогічно можна знайти добуток усіх можливих пар елементів поля $GF(3^2)$. Результати розрахунку наведені в табл. 1.7.

Таблиця 1.7

Множення в полі $GF(3^2)$

*	1	2	x	2x	x + 1	2x + 1	X + 2	2x + 2
1	1	2	x	2x	x + 1	2x + 1	X + 2	2x + 2
2	2	1	2x	x	2x + 2	x + 2	2x + 1	x + 1
x	x	2x	2	1	x + 2	x + 1	2x + 2	2x + 1
2x	2x	x	1	2	2x + 1	2x + 2	x + 1	X + 2
x + 1	X + 1	2x + 2	x + 2	2x + 1	2x	2	1	x
2x + 1	2x + 1	x + 2	x + 1	2x + 2	2	X	2x	1
x + 2	X + 2	2x + 1	2x + 2	X + 1	1	2x	X	2
2x + 2	2x + 2	x + 1	2x + 1	X + 2	x	1	2	2x

Приклад. Знайти елемент поля $GF(3^2)$ $\bar{y}$, зворотний за додаванням і множенням елементу $y = 2x + 1$.

Розв'язання. Скориставшись таблицями 2.4 і 2.5, визначимо:

$$\bar{y} + y = \bar{y} + 2x + 1 = 1 \text{ і } \bar{y} = x;$$

$$\bar{y} + y = y(2x + 1) = 1 \text{ і } \bar{y} = 2x + 2.$$

Правильність визначення $\bar{y}$ можна встановити за допомогою множення елементів $(2x + 1)$ і $2x + 2$, а для додавання шляхом додавання елементів $2x + 1$ і x .

Скінченні поля існують не за будь-якої кількості елементів, а тільки в тому випадку, коли кількість елементів є простим числом. Для кожного допустимого значення q існує лише одне поле. Якщо q – проста кількість, то елементами поля є числа $0, 1, 2, \dots, q - 1$, а додавання і множення є звичайними додаванням і множенням за модулем q .

Теорема 1.1. (алгоритм ділення) [4; 5]. Для кожної пари цілих чисел c і d при відмінному від нуля d знайдеться єдина пара цілих чисел Q (ціле) і s (остаток) таких, що:

$$c = dQ + s, \quad (1.38)$$

де $0 \leq s < |d|$.

Доведення. Безпосередня перевірка.

Звичайно більш цікаве не ціле, а остаток. Часто він записується у вигляді:

$$s = R_d[c]. \quad (1.39)$$

Іншим позначенням є

$$s \equiv c \pmod{d}. \quad (1.40)$$

Теорема 1.2 (алгоритм Евкліда) [4; 5]. Найбільший спільний дільник двох різних ненульових цілих чисел r і s може бути обчислений ітеративним застосуванням алгоритму розподілу. Припустимо, що $r < s$ й обидва ці числа додатні; тоді алгоритм полягає в такому:

$$\begin{aligned} s &= Q_1 r + r_1; \\ r &= Q_2 r + r_2; \\ r &= Q_3 r + r_3; \\ &\vdots \\ r_{n-1} &= Q_{n+1} r_n \end{aligned} \quad (1.41)$$

і процес закінчується, коли одержаний остаток дорівнює нулю. Останній ненульовий остаток r_n дорівнює найбільшому загальному дільнику.

Висновок 1.1 [4; 5; 7]. Для будь-яких цілих чисел r і s існують цілі числа a і b , такі, що найменший загальний дільник (НЗД):

$$\text{НЗД}(r,s) = ar + bs. \quad (1.42)$$

Доведення. Останній остаток в теоремі дорівнює найменшому загальному дільнику (r,s) . Скористаємося множиною виписаних у цій теоремі рівнянь, щоб виключити всю решту остатків. Це дасть вираз для r_n у вигляді лінійної комбінації r і s із цілочисловими коефіцієнтами.

Приклад. Розглянемо скінченне просте поле $GF(5)$ при $q = 5$ з елементами поля $\{0, 1, 2, 3, 4\}$.

Визначимо операції додавання і множення, віднімання і ділення, результати операцій зобразимо у вигляді табл. 1.8 – 1.11.

Таблиця 1.8

Операція додавання за mod q

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Таблиця 1.9

Операція множення за mod q

×	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Таблиця 1.10

Операція віднімання за mod q

+	-0	-1	-2	-3	-4
0	$0 + 0 = 0$	$0 + 4 = 4$	$0 + 3 = 3$	$0 + 2 = 2$	$0 + 1 = 1$
1	$1 + 0 = 1$	$1 + 4 = 0$	$1 + 3 = 4$	$1 + 2 = 3$	$1 + 1 = 2$
2	$2 + 0 = 2$	$2 + 4 = 1$	$2 + 3 = 0$	$2 + 2 = 4$	$2 + 1 = 3$
3	$3 + 0 = 3$	$3 + 4 = 2$	$3 + 3 = 1$	$3 + 2 = 0$	$3 + 1 = 4$
4	$4 + 0 = 4$	$4 + 4 = 3$	$4 + 3 = 2$	$4 + 2 = 1$	$4 + 1 = 0$

Таблиця 1.11

Операція ділення за mod q

×	1^{-1}	2^{-1}	3^{-1}	4^{-1}
1	$1 \times 1 = 1$	$1 \times 3 = 3$	$1 \times 2 = 2$	$1 \times 4 = 4$
2	$2 \times 1 = 2$	$2 \times 3 = 1$	$2 \times 2 = 4$	$2 \times 4 = 3$
3	$3 \times 1 = 3$	$3 \times 3 = 4$	$3 \times 2 = 1$	$3 \times 4 = 2$
4	$4 \times 1 = 4$	$4 \times 3 = 2$	$4 \times 2 = 3$	$4 \times 4 = 1$

Для виконання операції віднімання обчислимо адитивно обернені елементи поля GF(5) за виразом:

$$a + (-a) = 0. \quad (1.43)$$

Для виконання операції ділення обчислимо мультиплікативно обернені елементи поля GF(5) за виразом:

$$a \times a^{-1} = 1. \quad (1.44)$$

Розглянемо скінченні поля, які базуються на кільцях многочленів [4; 5; 7], введемо основні поняття і визначення.

Многочленом над полем GF(q) називається математичний вираз:

$$f(x) = f_{n-1}x^{n-1} + f_{n-2}x^{n-2} + \dots + f_1x + f_0, \quad (1.45)$$

де символ x називається невизначеною змінною, коефіцієнти $f_{n-1} \dots f_0$ належать полю GF(q) і є цілими числами.

Нульовим многочленом називається многочлен, для якого

$$f(x) = 0.$$

Звідним многочленом називається многочлен, старший коефіцієнт якого f_{n-1} дорівнює 1.

Степенем ненульового многочлена $f(x)$ називається індекс коефіцієнта f_{n-1} який позначається $\deg f(x)$.

Незвідним многочленом називається многочлен $p(x)$, що ділиться тільки на многочлени $\alpha p(x)$ або α , де α – довільний ненульовий елемент поля $GF(q)$. Простим многочленом називається зведений незвідний многочлен. Кільце многочленів за модулем зведеного многочлена $p(x)$ є полем тоді і тільки тоді, коли многочлен $p(x)$ простий.

Для побудови полів Галуа використовують примітивні елементи поля $GF(q)$ – елемент поля α , такий, за якого вся решта елементів поля, за винятком нуля, можуть бути зображені у вигляді степеня α .

Примітивним многочленом $p(x)$ над полем $GF(q)$ називається простий многочлен, такий, за якого в розширеному полі, побудованому за модулем $p(x)$, відповідний многочлену x елемент поля є примітивним.

Скінченне поле, побудоване над кільцем многочленів, називається розширеним полем Галуа і позначається $GF(q)$, де $q = p^m$: p (просте ціле число) називається характеристикою розширеного поля; m (ненульове додатне ціле число за $\text{mod } q(x)$) називається степенем розширення примітивного поля.

У загальному вигляді породжувальний многочлен дорівнює:

$$f(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1}. \quad (1.46)$$

Елементами поля є всі многочлени, одержані шляхом підстановки характеристики поля в породжувальний многочлен.

Правила множення і додавання таких многочленів отримуються зі звичайного множення і додавання многочленів та подальшого зведення результату за модулем деякого спеціального многочлена $q(x)$ до степеня m . Цей многочлен має ту властивість, що його не можна розкласти на множники, використовуючи тільки многочлени з коефіцієнтами з $GF(p)$. Такі многочлени називаються незвідними, вони аналогічні до простих чисел. Як і прості числа, вони звичайно знаходяться методом перебору; таблиці незвідних многочленів є в декількох книгах (наприклад, книга Пітерсона і Уелдона "Коди, що виправляють помилки").

Для визначення елементів розширеного скінченного поля $GF(p^m)$, яке будується за кільцем многочленів з операціями за модулем породжувального многочлена:

$$g(x) = g_0 + g_1x + g_2x^2 + \dots + g_{n-1}x^m,$$

використовують регістри зсуву зі зворотними зв'язками. Структурна схема такого пристрою зображена на рис. 1.15.

Початкове положення елементів регістра зсуву дорівнює нулю. Робота починається з подачі на вхід пристрою одиничного символу. На першому такті з елементів регістра зсуву зчитуються елементи нульового елемента поля Галуа, тобто послідовність із m нулів.

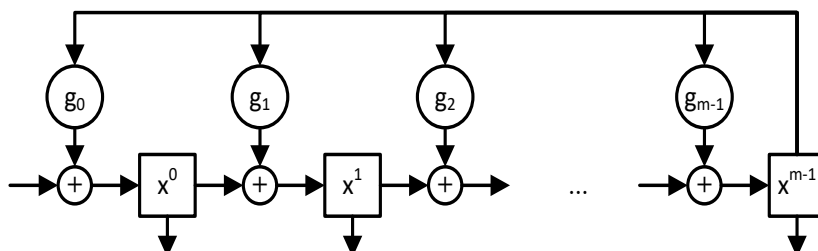


Рис. 1.15. Структурна схема регістра зсуву зі зворотними зв'язками для побудови елементів розширених скінченних полів Галуа

Елементи зчитаної послідовності задають коефіцієнти многочлена, тобто елемента розширеного поля. На другому та наступних тактах регістр зсуву заповнюється відповідно до роботи пристрою, яка задається логікою зворотних зв'язків. Обернені зв'язки відповідають коефіцієнтам породжувального многочлена $g(x)$.

На кожному такті з елементів зчитується наступний елемент поля у вигляді коефіцієнтів відповідних многочленів.

Розглянемо множину всіх многочленів від формальної змінної x в степені $m - 1$ з коефіцієнтами з $GF(p)$, де p – проста кількість.

Приклад. Розглянемо розширене поле $GF(4)$, побудоване через породжувальний многочлен $q(x) = x^2 + x + 1$. Многочлен $q(x)$ не зводиться над $GF(2)$ і може бути використаний для побудови $GF(2^2)$. Елементами поля є $\{0, 1, x, x + 1\}$.

Поле завжди містить мультиплікативну групу $\{1, 2, \dots, q\}$ з мультиплікативною одиницею 1, таку, за якої $a \times a^{-1} = 1$, й адитивну групу $\{0, 1, \dots, q - 1\}$, таку, за якої $a + (-a) = 0$.

Для побудови скінченного поля $GF(2^2)$ за кільцем многочленів за модулем:

$$g(x) = x^2 + x + 1$$

скористаємося регістром зсуву зі зворотними зв'язками, структурна схема якого зображена на рис. 1.16. Коефіцієнти породжувального многочлена $g(x)$ задають логіку пристрою, тобто обернені зв'язки регістра зсуву. Початковий стан елементів регістра дорівнює нулю. Робота пристрою

починається з подачі на його вхід одиничного символу. На кожному такті з комірок регістра зчитуються елементи поля $GF(2^2)$ у вигляді коефіцієнтів відповідних многочленів. У табл. 1.12 поданий зміст елементів регістра зсуву та відповідні дії, що виконуються в пристрої.

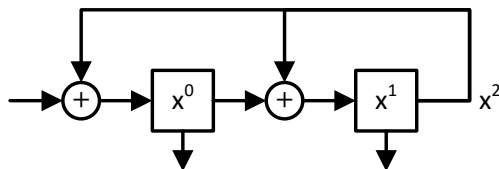


Рис. 1.16. Структурна схема регістра зсуву зі зворотними зв'язками для побудови елементів поля $GF(2^2)$

Стовпчик таблиці " x^2 " відповідає значенням, що отримані на виході регістру та подаються по ланцюгу зворотного зв'язку.

Побудуємо таблиці множення, додавання, віднімання і ділення для розширеного поля $GF(p^m)$ при $p = 2$ (поле задається тільки 0 і 1), $m = 2$ та примітивного многочлена $q(x) = x^2 + x + 1$. Елементами поля будуть всі можливі варіанти многочленів, одержаних шляхом підстановки характеристики поля p в породжувальний многочлен $\{0, 1, x, x + 1\}$. Для визначення елементів поля використовують регістри зсуву зі зворотними зв'язками. Регістр зсуву із зворотними зв'язками складається з m клітинок (табл. 1.12).

Таблица 1.12

Регістр зсуву зі зворотними зв'язками

x^0	x^1	x^2	Виконувані операції
0	0	0	На початковому етапі регістр обнулений
1	0	0	Введемо в регістр 1 у праву комірку " x^0 "
0	1	0	На наступному етапі вміст зсувається на одну комірку вправо
0	0	1	Якщо в комірці " x^2 " з'являється 1, то до всього регістра додається породжувальний (примітивний) многочлен $x^2 + x + 1$
$\oplus$			
1	1	1	1 (111)
x^0	x^1	x^2	Виконувані операції
1	1	0	На наступному етапі вміст зсувається на одну комірку вправо
0	1	1	Якщо в комірці " x^2 " з'являється 1, то до всього регістра додається породжувальний (примітивний) многочлен $x^2 + x + 1$
$\oplus$			
1	1	1	$x^2 + x + 1$ (111)
1	0	0	Такий набір вже є, відбувся цикл

Таким чином, елементами поля є $\{0, 1, x, x + 1\}$.

Результати операцій додавання, множення, віднімання та ділення для розширеного поля $GF(4)$ наведені в табл. 1.13 – 1.16.

При цьому необхідно враховувати, що всі операції обчислюються в скінченному розширеному полі за $\text{mod } q(x)$.

Таблиця 1.13

Операція додавання за $\text{mod } q(x)$

+	0	1	x	x + 1
0	0	1	x	x + 1
1	1	0	x + 1	x
x	x	x + 1	0	1
x + 1	x + 1	x	1	0

Таблиця 1.14

Операція множення за $\text{mod } q(x)$

x	0	1	x	x + 1
0	0	0	0	0
1	0	1	x	x + 1
x	0	x	x + 1	1
x + 1	0	x + 1	1	x

Для виконання операції віднімання обчислимо адитивно обернені елементи поля $GF(4)$. У всіх полях $GF(2^m)$ адитивно обернені елементи дорівнюють самим елементам.

Таблиця 1.15

Операція віднімання за $\text{mod } q(x)$

+	- 0	- 1	- x	- x + 1
0	$0 + 0 = 0$	$0 + 1 = 1$	$0 + x = x$	$0 + x + 1 = x + 1$
1	$1 + 0 = 1$	$1 + 1 = 0$	$1 + x = 1 + x$	$1 + x + 1 = x$
x	$x + 0 = x$	$x + 1 = 1$	$x + x = 0$	$x + x + 1 = 1$
x+1	$x+1+0=x+1$	$x + 1 + 1 = x$	$x+1+x=1$	$x + 1 + x + 1 = 0$

Для виконання операції ділення обчислимо мультиплікативно обернені елементи поля $GF(4)$.

Операція ділення за mod $q(x)$

$\times$	1^{-1}	x^{-1}	$(x+1)^{-1}$
1	$1 \times 1 = 1$	$1 \times (x+1) = x+1$	$1 \times x = x$
x	$x \times 1 = x$	$x \times (x+1) = 1$	$x \times x = x+1$
$x+1$	$(x+1) \times 1 = x+1$	$(x+1) \times (x+1) = x$	$(x+1) \times x = 1$

Розглянемо основні властивості полів Галуа [4]:

- кількість елементів будь-якого поля Галуа дорівнює степеню простого числа;
- для будь-якого простого p і цілого додатного m найменшим підполем поля $GF(p^m)$ є поле $GF(p)$. Елементи поля $GF(p)$ називаються цілими числами поля $GF(p^m)$, а кількість p – його характеристикою;
- у $GF(2)$ для кожного β поля справедливе рівняння $-\beta = \beta$;
- для будь-якого простого p і цілого додатного m існує $GF(p^m)$ з p^m елементами;
- кожне поле Галуа містить хоча б один примітивний елемент;
- над кожним полем Галуа існує хоча б один примітивний многочлен будь-якого додатного степеня;
- кожен примітивний елемент має над будь-яким підполем простий мінімальний многочлен;
- два поля Галуа з однією й тією ж кількістю елементів ізоморфні.

Між многочленами й елементами скінченних полів є додатковий зв'язок – зв'язок між співмножниками і коренями многочленів з дійсними коефіцієнтами.

Наприклад, заданий многочлен може бути зображений у вигляді

$$f(x) = (x - \beta_1)(x - \beta_2) \dots (x - \beta_j),$$

де j значень $\beta_1, \dots, \beta_j$ є коренями многочлена $f(x)$.

Многочлени й елементи поля Галуа мають ряд властивостей, які виявляються корисними в процесі опису кодів і побудови декодерів [4; 5].

Властивість 1.1. Незвідні многочлени. Многочлен $f(x)$ з елементами деякого скінченного поля називається незвідним, якщо його не можна розкласти на множники, використовуючи лише елементи поля. Проте цей многочлен завжди можна розкласти на множники, використовуючи

елементи з певного розширення. Тому многочлен завжди має корені в певному розширенні.

Якщо $f(x)$ – незвідний многочлен з коефіцієнтами з $GF(p)$ і α – його корінь, то $\alpha^p, \alpha^{p^2}, \alpha^{p^3}, \dots$ також будуть коренями. Крім того, всі корені $f(x)$ можуть бути знайдені таким чином. Многочлен $f(x)$ називається мінімальною функцією від α . Якщо α – примітивний елемент (генератор – елемент, який має ту властивість, що будь-який інший елемент поля є деяким степенем цього елемента), то $f(x)$ називається примітивним многочленом.

Приклад. Многочлен $f(x) = 1 + x + x^3$ незвідний над полем $GF(2)$. Елемент α є коренем, оскільки $f(\alpha) = 1 + \alpha + \alpha^3 = (100) + (010) + (110) = (000) = 0$.

Використовуючи цю властивість, одержуємо, що α^2, α^4 теж є коренями:

$$f(\alpha^2) = 1 + \alpha^2 + \alpha^6 = (100) + (001) + (101) = (000) = 0;$$

$$f(\alpha^4) = 1 + \alpha^4 + \alpha^{12} = 1 + \alpha^4 + \alpha^5 = (100) + (011) + (111) = (000) = 0.$$

І навпаки, якщо $\alpha, \alpha^2, \alpha^4$ є коренями $f(x)$, то $f(x)$ можна записати у вигляді:

$$\begin{aligned} f(x) &= (x - \alpha)(x - \alpha^2)(x - \alpha^4) = x^3 - x^2(\alpha + \alpha^2 + \alpha^4) + x(\alpha^3 + \alpha^6 + \alpha^5) - \alpha^7 = \\ &= x^3 + x + 1. \end{aligned}$$

Властивість 1.2. $f(x^p) = f^p(x)$. Якщо $f(x)$ – довільний многочлен, коефіцієнти якого лежать в $GF(p)$, то $f(x^p) = f^p(x)$.

Справедливість цієї властивості випливає з того, що всі попарні або багатократні добутки в $f^p(x)$ з'являються з коефіцієнтом, який ділиться на p а отже, дорівнює 0 в $GF(p)$.

Приклад. $(1 + x + x^2)^2 = 1 + x^2 + x^4 + 2(x + x^2 + x^3) = 1 + x^2 + x^4$.

Властивість 1.3. Співмножники $x^m - 1$. Порядком m елемента β скінченного поля називається найменше значення m , для якого $\beta^m = 1$. За визначенням β є коренем многочлена $x^m - 1$. Якщо β є коренем деякого незвідного многочлена $f(x)$, то $f(x)$ повинен бути дільником $x^m - 1$.

Властивість 1.4. Корені многочлена $x^{p^m-1} - 1$ поля $GF(p^m)$ збігаються з ненульовими елементами $GF(p^m)$.

Приклад властивостей 3 і 4. Многочлен $f(x) = x^7 - 1$ розкладається на множники таким чином: $x^7 - 1 = (x^3 + x^2 + 1)(x^3 + x + 1)(x + 1)$. Використовуючи табл. 2.3, легко показати, що коренями $x^3 + x^2 + 1$ є α, α^2 ,

α^4 , а коренями $x^3 + x + 1$ є $\alpha^3, \alpha^5, \alpha^6, \alpha x + 1$. Ці сім коренів є сімома ненульовими елементами $GF(8)$.

Властивість 1.5. Подільність $x^m - 1$ на $f(x)$. Найменше значення m , для якого довільний многочлен $f(x)$ без кратних коренів ділить $x^m - 1$, збігається з найменшим загальним кратним (НЗД) порядків коренів $f(x)$. Тому m є довжиною найкоротшого циклу, породженого регістром зі зворотними зв'язками, які визначаються многочленом $f(x)$.

Приклад. Порядок коренів $1 + x + x^3$ дорівнює 7, а порядок коренів $1 + x^3 + x^4$ дорівнює 15. НЗД, кратне 7 і 15, дорівнює 105.

Тому $g(x) = (1 + x + x^3)(1 + x^3 + x^4) = 1 + x + x^5 + x^6 + x^7$ ділить $x^{105} - 1$ і породжує цикл довжиною 105.

Властивість 1.6. Подільність $x^n - 1$ на $x^m - 1$. Многочлен $x^n - 1$ ділиться на $x^m - 1$ в тому і лише в тому випадку, якщо n ділиться на m .

Це впливає з того, що якщо корені $x^m - 1$ є також коренями $x^n - 1$, то n повинне ділитися на m .

Приклад. Використовуючи прості обчислення, одержимо, що на $x^3 - 1$ ділиться $x^6 - 1$, але не ділиться $x^5 - 1$ і $x^7 - 1$.

1.6. Алгебраїчна теорія блокових кодів

1.6.1. Призначення та класифікація завадостійких кодів

Одним з основних показників якості передачі інформації по каналах передачі інформації (зараз ми говоримо про передачу дискретної інформації) є, як ми вже відзначали раніше, достовірність. Вона може бути охарактеризована імовірністю помилкового декодування (приймання) кодової комбінації відповідної окремому повідомленню первинного коду.

Вимоги до достовірності передачі дуже різноманітні і залежать від призначення системи передачі інформації. Вважається, що ймовірність помилки на кодову комбінацію не повинна перевищувати $10^{-6} \dots 10^{-9}$. В особливо відповідальних системах існують ще жорсткіші вимоги.

Імовірність помилкового прийняття символу в каналах, використовуваних для передачі дискретної інформації, знаходиться в межах $10^{-3} \dots 10^{-5}$.

Всі існуючі способи підвищення достовірності передачі можна розділити на три групи:

- заходи експлуатаційного і профілактичного характеру (зменшення числа джерел завад, поліпшення стабільності роботи основних вузлів, резервування тощо);

- зменшення ймовірності помилкового прийняття за рахунок підвищення відношення сигнал/завада (вибір методів модуляції, приймання, тривалості, спектрів, потужності);

- використання завадостійких кодів.

Завадостійке кодування є одним з найбільш ефективних засобів збільшення достовірності передачі інформації.

До теперішнього часу розроблена велика кількість завадостійких кодів. Принципи їх побудови надзвичайно різноманітні, проте, за своєю структурою всі завадостійкі коди діляться на лінійні і нелінійні.

За способом виконання процедури кодування лінійні коди можна підрозділити на два великі класи: блокові і неперервні.

При блоковому кодуванні кожна буква повідомлення (або послідовність букв) перетворюється в певну послідовність (блок) кодових символів, що називається кодовою комбінацією. При неперервному кодуванні кодові символи визначаються всією послідовністю (або значною частиною послідовності) елементів повідомлення (інформаційних символів) і не можуть бути розділені на окремі відособлені кодові комбінації.

На практиці найчастіше використовуються блокові коди, що в свою чергу діляться на рівномірні і нерівномірні. У рівномірних кодах всі кодові комбінації містять однакову кількість символів і передаються по каналу елементами сигналу незмінної тривалості. Це значно спрощує техніку передачі і приймання повідомлень. У нерівномірних кодах кількість символів в комбінаціях, відповідних різним повідомленням, різне. Вони не знайшли широкого практичного застосування з причини складності техніки їх формування і обробки.

Блокові коди діляться також на роздільні і нероздільні. У роздільних кодах функції окремих символів кодової комбінації чітко розмежовуються на інформаційні і контрольні (перевірочні). Повідомлення, що передаються, визначаються k інформаційними символами. Надмірні (контрольні, перевірочні) r символи доповнюють кодову комбінацію до $n = k + r$ символів, через що такі блокові коди позначають як (n, k) коди.

Найбільший клас роздільних кодів складають систематичні коди, у яких значення перевірочних символів визначаються в результаті

проведення лінійних операцій над інформаційними символами. Виявлення і виправлення помилок здійснюється для таких кодів формалізовано, за чіткими правилами, що дозволяє автоматизувати процес передачі повідомлень та зменшити складність апаратури.

Класифікація розглянутих кодів наведена на рис. 1.17.

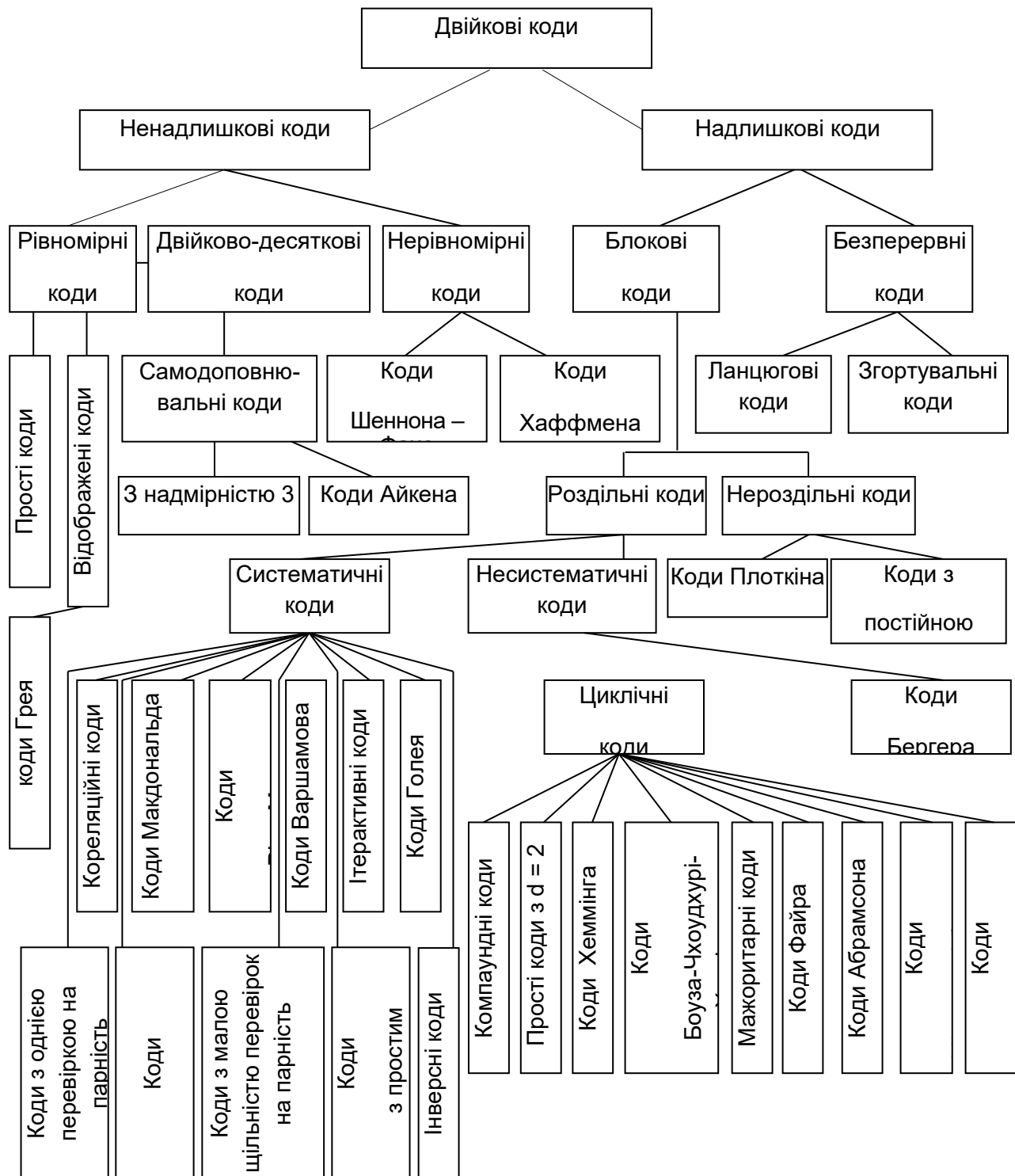


Рис. 1.17. Класифікація двійкових кодів

Найбільше розповсюдження серед блокових кодів знайшли коди з перевіркою на парність, з повторенням символів, рівноважні коди, коди Хеммінга, коди Ріда – Малера та циклічні коди. До останнього класу належать коди Боуза – Чоудхурі – Хоквінгема (БЧХ) і коди Ріда – Соломона (РС), що набули найбільшого поширення в техніці передачі інформації.

1.6.2. Загальні принципи завадостійкого кодування

Розглянемо загальну постановку задачі завадостійкого кодування повідомлень.

Від джерела інформації поступає послідовність елементарних повідомлень b_i . Кодування полягає у тому, що послідовність символів джерела замінюється послідовністю m -ічних кодових символів (надалі розглядатимемо тільки двійкові коди, для яких $m = 2$). Вказане перетворення є взаємооднозначним, що і дозволяє здійснити в приймачі декодування, тобто відновити повідомлення за прийнятою кодовою комбінацією.

В результаті дії завади деякі кодові символи в приймачі виділяються спотвореними – одиничні елементи комбінації сприймаються як нульові і навпаки. Може бути передбачений і режим стирання символів, коли рішення про приймання деяких з них не може бути виконано з необхідною надійністю (впевненістю).

Мета завадостійкого кодування повідомлень – перетворення повідомлень b_i , $i = 1, 2, \dots, m$ в послідовність кодових символів b_j , $j = 1, 2, \dots, n$, яка має властивість виявлення присутності в ній помилок або виявлення і виправлення цих помилок.

Відразу відзначимо, що не існує кодів кінцевої довжини, які дозволяють знайти і, тим більше, виправити всі можливі помилки. Можна побудувати лише коди, що знаходять або виправляють деяку кількість помилок певного вигляду (звично найімовірніших або найбільш небезпечних).

Ця мета може бути досягнута тільки введенням надмірності, тобто збільшенням числа кодових символів в послідовності, що передається, відносно символів, що забезпечують однозначне подання повідомлень, що передаються (інформаційних символів). Наявність додаткових (надмірних) символів дозволяє накласти на послідовності символів, що

передаються (кодові комбінації), додаткові зв'язки (умови) між ними, перевірка яких на приймальній стороні дає можливість знайти і виправити помилки. Вся сукупність можливих кодових комбінацій в цьому випадку називається завадостійким (коректуючим або надмірним) кодом.

Отже, здатність коду знаходити і виправляти помилки обумовлена наявністю надмірних символів. Розглянемо, яким же чином їх введення в кодові комбінації рівномірного блокового коду дозволяє підвищити достовірність передачі повідомлень.

Розглянемо множину n -розрядних комбінацій блокового рівномірного коду. Для двійкових кодів всього можна побудувати 2^n різних комбінацій. У звичайному, некоректуючому, коді розрядність кодової комбінації n вибирається так, щоб кількість комбінацій була мінімально необхідна для подання всіх переданих повідомлень

$$N = 2^n \geq K \quad (1.47)$$

(практично в цьому випадку кодові комбінації – це двійкове подання номерів переданих повідомлень). При цьому всі комбінації коду використовуються при передачі (при записі рівняння (1.47)).

Завадостійкі коди будуються так, що кількість можливих кодових комбінацій N істотно перевищує кількість кодованих повідомлень. При цьому частина кодових комбінацій, які називаються дозволеними, використовується для кодування. Їх кількість дорівнює $N_d = K = 2^k$. Інша частина комбінацій для кодування не використовується, і вони називаються забороненими. Кількість заборонених комбінацій дорівнює 2^r або 2^{n-k} .

На приймальній стороні відомо, які з комбінацій є дозволеними, а які забороненими. Тому, якщо передана комбінація в результаті помилки перетвориться в деяку заборонену комбінацію, то така помилка буде знайдена або виправлена. Ясно, що помилки, які призводять до утворення іншої дозволеної комбінації не можуть бути виявленими або виправленими. Проте, при розумному підході до побудови коду, може досягатися дуже мала імовірність не знайденої помилки. При цьому очевидно, що ймовірність трансформації однієї дозволеної комбінації в іншу дозволена тим менша, чим більше комбінації відрізняються одна від одної. Відмінність між комбінаціями b_i і b_j рівномірного коду прийнято характеризувати відстанню Хеммінга d_{ij} , яка дорівнює кількості позицій з

неспівпадаючими символами (вазі вектора $b = b_i \oplus b_j - w_{ij}$). Кількість відмінних від 0 символів в кодовому слові b_i називається його вагою w_i . Наприклад, кодові комбінації 1100101 і 1011100 відрізняються символами в чотирьох розрядах, тому $b = 0111001$ і відстань Хеммінга дорівнює чотирьом.

Для будь-якого коду $1 \leq d_{ij} \leq n$. Мінімальна відстань між дозволеними комбінаціями в цьому коді називається мінімальною кодовою відстанню і позначається $d_{\min}$.

Таким чином, якщо код має кодову відстань $d_{\min}$, то це означає, що дозвалені комбінації цього коду відрізняються одна від одної не менше ніж в $d_{\min}$ символах і тільки поява $d_{\min}$ і більше помилок в прийнятій комбінації можуть перевести її в іншу дозволену комбінацію.

З другого боку, якщо кількість помилок в комбінації буде менша ніж $d_{\min}$, то стає зрозумілим, що така комбінація помилок призведе до забороненої комбінації.

У цілому в процесі передачі інформації можна розрізнати три ситуації: спотворення (помилки) відсутні, спотворення виявляються (прийнята заборонена комбінація) і спотворення не виявляються – одна дозволена комбінація переходить в іншу дозволену. Говорять, що в каналі відбулася помилка кратності t , якщо в кодовій комбінації прийнято помилково t символів. В результаті можна дійти висновку, що максимальна кількість помилок, яка достовірно виявляється в комбінації завадостійкого коду, пов'язана з кодовою відстанню наступним співвідношенням:

$$t_{\text{зн}} = d_{\min} - 1.$$

При $d_{\min} \leq t_{\text{зн}} + 1$ або при $t \geq t_{\text{зн}}$ прийнята кодова комбінація може виявитися дозвальною і помилка не буде знайдена. Але і при $t \geq t_{\text{зн}}$ можливі ситуації, коли комбінація виявляється забороненою, тобто частково виявлятимуться і помилки кратності більшої, ніж $t_{\text{зн}}$.

Як приклад, розглянемо властивості трирозрядного коду, що до знаходження помилок. При $d_{\min} = 1$ дозвалені комбінації – це все комбінації коду 000, 001, 010, 011, 100, 101, 110, 111. Будь-яка поодинокі помилка трансформує кожну з цих комбінацій в іншу дозволену. Таким чином достовірно виявлення помилок неможливе.

При $d_{\min} = 2$ підмножина дозволених комбінацій може бути створена за принципом парності в них числа одиниць, тобто 000, 011, 101, 110. До

заборонених комбінацій віднесемо решту комбінацій: 001, 010, 100, 111. Жодна із заборонених кодових комбінацій при поодинокій помилці не переходить в іншу дозволена комбінацію, тобто при $d_{\min} = 2$ код достовірно знаходить всі поодинокі помилки. Що стосується помилок вищої кратності, то частина з них виявляється, а частина ні. Так виявляються всі триразові помилки, проте двократні помилки переводять одну дозволена кодову комбінацію в іншу і не можуть бути знайдені. Збільшення кодової відстані до $d_{\min} = 3$ дозволяє використати для передачі тільки дві кодові комбінації 000 і 111 або 001 і 110 і тощо. Але в цьому випадку і двократна помилка не може перевести одну дозволена кодову комбінацію в іншу.

Модель трирозрядного коду можна використати і для пояснення принципів використання надмірності для виправлення помилок. Основою цього є припущення про те, що чим більша кратність помилки, тим вона менше ймовірна. Це припущення добре узгоджується з біноміальною моделлю помилок в дискретному каналі передачі інформації. В цьому випадку при помилках найбільшу ймовірність матимуть помилкові комбінації, віддалені від переданої дозволена на відстань $d = 1$, менш ймовірними будуть переходи в комбінації, віддалені від переданої на відстань $d = 2$.

Як результат, в процесі приймання – передачі кодових комбінацій по каналу передачі інформації з помилками, кожна з дозволених (переданих) комбінацій трансформується в деяку множину прийнятих кодових комбінацій, локалізованих навколо переданої дозволена кодової комбінації (якщо її відображати точкою відповідного n -мірного простору, як зображено на рис. 1.18).

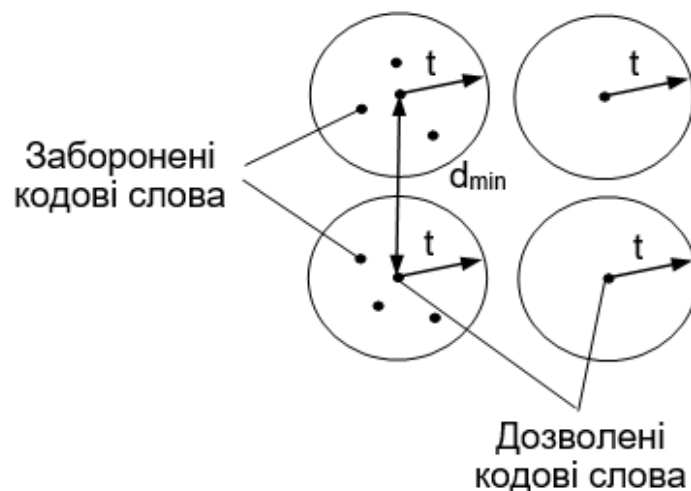


Рис. 1.18 Сфери декодування

Збільшуючи відстань між центрами цих областей в n -мірному просторі (крапками, що відповідні дозволеним кодовим комбінаціям), можна реалізувати процедуру виправлення помилок заданої кратності, якщо ухвалювати рішення про декодування тієї дозволеної комбінації, відстань якої від прийнятої (забороненої) кодової комбінації виявляється мінімальною (декодування за найменшою відстанню). Цей метод декодування є оптимальним для симетричного каналу (для інших може бути і не оптимальним).

Тепер легко зрозуміти, що максимальна кількість помилок, які достовірно виправляються, пов'язана з кодовою відстанню такими нерівностями:

– якщо $d_{\min}$ – непарне, то

$$t_{\text{випр}} \leq \frac{d_{\min} - 1}{2};$$

– якщо $d_{\min}$ – парне, то

$$t_{\text{випр}} \leq \frac{d_{\min}}{2} - 1.$$

Дійсно, для виправлення помилок кратності t необхідно включити в кожен з вказаних вище підмножин всі заборонені кодові комбінації, віддалені від дозволеної на відстань $d = t$. Для того, щоб сусідні підмножини не перетиналися, відстань між найближчими дозволеними комбінаціями повинна бути не менше $2t + 1$.

Таким чином, код має властивість знаходити і виправляти t -кратні помилки, якщо виконується умова:

$$d_{\min} \geq 2t + 1.$$

Звернемося знов до трирозрядного коду. За дозвалені комбінації приймемо 000 і 111. Тоді дозволеним комбінації 000 відповідає підмножина заборонених комбінацій 001, 010, 100, що утворюються за рахунок виникнення одноразової помилки. Подібним же чином дозволеним комбінації 111 відповідає підмножина заборонених комбінацій 110, 101, 011.

При одноразовому спотворенні будь-якої з двох дозволених комбінацій, комбінація, що приймається, не вийде за межі "своїх" підмножин і помилка буде виправлена. Що стосується подвійних

помилки, то вони виявляються, але при їх виправленні ухвалюються невірні рішення. Потрійні помилки не виявляються взагалі.

Завадостійке кодування можна застосовувати також в дискретних каналах із стиранням. Якщо в прийнятій комбінації спотворених символів немає, а є тільки $t_{\text{ст}}$ стертих (непізнаних) символів, то при виконанні умови $t_{\text{ст}} \leq d_{\text{min}} - 1$ ці символи можуть бути відновлені при декодуванні за мінімумом відстані. Один і той же код може декодуватися з виправленням помилок і стирань. У загальному випадку код з відстанню d_{min} виправляє будь-які $t \leq t_{\text{ст}}$ стирань і будь-які $t \leq t_{\text{вип}}$ помилок, якщо $d_{\text{min}} \geq 2t_{\text{вип}} + t_{\text{ст}} + 1$.

Крім поєднань помилок і стирань, що повністю виправляються і знаходяться, кратність яких визначається кодовою відстанню, коди з корекцією часто виправляють або знаходять і багато інших поєднань помилок великої кратності.

Кодова відстань коду d_{min} є основним показником його здатності, щодо знаходження і виправлення помилок. Збільшуючи довжину коду n і зберігаючи кількість дозволених кодових комбінацій, можна набути скільки завгодно велике значення d_{min} . Але зі збільшенням n зменшується швидкість передачі повідомлень. Якщо довжина коду n задана, то будь-яке значення $d_{\text{min}} \leq n$ можна одержати, зменшуючи кількість дозволених комбінацій, але при цьому зменшується і кількість повідомлень, які можна закодувати. Отже, задачу пошуку найкращого коду необхідно формулювати так: при заданих n і N_d знайти код, який має більше значення d_{min} .

Характеристикою завадостійкого коду, яка вказує ступінь подовження кодової комбінації, є надмірність. Відносна надмірність може бути записана співвідношенням

$$\eta_k = \frac{n-k}{n} = 1 - \frac{\log K}{n \log m}.$$

Другою характеристикою є відносна швидкість коду

$$R_n = \frac{\log K}{n \log m} = \frac{k}{n} = 1 - \eta_k.$$

Коди, що забезпечують задану коректуючу здатність при мінімальній надмірності, називаються оптимальними. Але загального методу пошуку оптимальних кодів в наш час не існує.

Кодова відстань $d_{\min}$ повністю характеризує стійкість коду щодо заданого рівня завад. Проте, в реальних умовах кількість спотворених символів в кодовій комбінації може з певною ймовірністю перевищити t . В цьому випадку завадостійкість коду зручно описувати статистичними показниками. До їх числа належать ймовірність правильного прийняття $P_{\text{пр}}$, ймовірність помилки $P_{\text{пом}}$. Цю ймовірність можна визначити, знаючи структуру коду і статистику завад. Знайдемо $P_{\text{пр}}$ та $P_{\text{пом}}$ для каналу з незалежними помилками.

Ймовірність помилкового декодування кодових комбінацій. При передачі інформації безнадмірним кодом в системах без зворотного зв'язку використовуються всі кодові комбінації. При цьому помилкове прийняття хоча б одного кодового символу призводить до переходу однієї дозволеної кодової комбінації в іншу. Отже, умовою правильного прийняття кодової комбінації є правильне прийняття кожного символу. Знаючи ймовірність помилкового прийняття символу p_0 , можна знайти ймовірність його правильного прийняття $p_{\text{пр}} = 1 - p_0$. Якщо спотворення окремих символів незалежні, то ймовірність правильного прийняття всієї кодової комбінації дорівнює

$$P_{\text{пр}} = (1 - p_0)^n \approx np_0.$$

Таким чином, у системах з безнадмірним кодуванням ймовірність помилкового прийняття кодової комбінації приблизно в n разів більша ймовірності помилкового прийняття символу.

Оцінимо ймовірність спотворення кодової комбінації завадостійкого коду з $n = k + r$ символів. Оскільки такий код дозволяє виявляти або виправляти частину помилок, то, мабуть, не всі помилки при прийманні окремих символів призводять до неправильної реєстрації кодової комбінації. Нехай кратність помилки, що виявляється або виправляється завадостійким кодом, дорівнює t . Тоді ймовірність правильної реєстрації комбінацій завадостійкого коду дорівнює ймовірності того, що комбінація з n елементів містить не більше t помилок. Позначимо ймовірність помилкового приймання символу завадостійкого коду як p_0 . Ймовірність того, що в кодовій комбінації на i позиціях відбулася помилка, а на $n - i$

позиціях помилок немає, дорівнює $p_0^i (1-p_0)^{n-i}$. У кодовій комбінації помилка на i позиціях може відбутися у всіляких сполученнях цих позицій. Кількість таких сполучень дорівнює

$$C_n^i = \frac{n!}{i!(n-i)!}.$$

Отже, ймовірність того, що в кодовій комбінації з n символів з'явиться помилка кратністю i (у будь-якому сполученні позицій), визначається як

$$P_i = C_n^i p_0^i (1-p_0)^{n-i}.$$

Ймовірність того, що комбінація з n елементів містить не більше t помилок, дорівнює

$$P_t(i \leq t) = \sum_{i=0}^t C_n^i p_0^i (1-p_0)^{n-i}.$$

Цей вираз можна розглядати як ймовірність виявлення або виправлення помилок, тому що за умовою було прийнято, що всі помилки кратності t та менше можуть бути виявлені або виправлені.

Якщо при передачі виникнуть помилки кратності $t+1$ і вище, то такі помилки вже не будуть виявлені або виправлені. Отже, повну ймовірність не виявлення або не виправлення помилок, що i визначає ймовірність спотворення кодової комбінації завадостійкого коду, можна знайти за формулою

$$P_{\text{пом}} = 1 - \sum_{i=0}^t C_n^i p_0^i (1-p_0)^{n-i} = \sum_{j=t+1}^n C_n^j p_0^j (1-p_0)^{n-j}.$$

Конкретизуємо останній вираз з урахуванням способу декодування. Ввівши в нього параметр $d_{\min}$ і взявши до уваги зв'язок між кодовою відстанню та кількістю помилок, що виявляються або виправляються, отримаємо наступні оцінки ймовірності помилкового декодування при виправленні помилок $P_{\text{пом}}^{(\text{випр})}$ і ймовірності невиявленої помилки при виявленні помилок $P_{\text{пом}}^{(\text{виявл})}$

$$P_{\text{пом}}^{(\text{випр})} \leq 1 - \sum_{i=0}^{(d_{\text{min}}-1)/2} C_n^i p_0^i (1-p_0)^{n-i};$$

$$P_{\text{пом}}^{(\text{виявл})} \leq 1 - \sum_{i=0}^{d_{\text{min}}-1} C_n^i p_0^i (1-p_0)^{n-i}.$$

Необхідно відзначити, що отримані формули справедливі для того випадку, коли розглянутий код виправляє (або виявляє) всі помилки – однократні, двократні і так далі до t -кратних включно. Всі інші помилки, починаючи з кратності $t+1$ і вище, цей код не виправляє (не виявляє). Більшість реальних кодів виправляють (або виявляють) і частину помилок більш високої кратності, ніж t . При розрахунку коригуючих характеристик конкретних кодів цю обставину необхідно враховувати.

Порівнюючи ймовірності помилкового приймання кодових комбінацій завадостійкого коду з імовірністю помилкового приймання безнадмірної кодової комбінації, можна встановити, при яких умовах застосування завадостійкого коду можна підвищити завадостійкість. Ясно, що таке порівняння повинне відбуватися при деяких однакових умовах. Найчастіше порівняння за завадостійкості систем передачі інформації, що використовують різні коди, здійснюється при однаковій швидкості передачі інформації. При цьому час, що відводиться на передачу комбінацій первинного й завадостійкого кодів, повинен бути однаковим.

Отже, тривалість елементарних сигналів при завадостійкому кодуванні буде меншою тривалості елементарних сигналів при безнадмірному кодуванні. Скорочення тривалості сигналів при заданій середній потужності зменшує їхню енергію, що призводить до збільшення ймовірності помилкового приймання символу завадостійкого коду. В цих умовах застосування завадостійкого коду доцільно лише в тому випадку, якщо підвищення завадостійкості за рахунок виявлення або виправлення помилок перекиє її зниження за рахунок збільшення p_0 .

Аналіз отриманих виразів показує, що ефективність завадостійких кодів зростає зі збільшенням достовірності прийняття окремих символів. При низькій же достовірності прийняття символів використання завадостійких кодів може призвести навіть до негативних наслідків, тому що переважним може виявитися збільшення ймовірності помилкового прийняття елементарних сигналів із-за зменшення їхньої енергії при збільшенні надмірності.

Остання обставина – зменшення енергії елементарного сигналу при збільшенні коригуючих можливостей кодів, безумовно знижує ефективність завадостійкого кодування. Однак, завадостійкі коди вигідно відрізняються від безнадмірних здатністю усувати будь-які помилки, якщо їхня кількість в інтервалі тривалості кодової комбінації не перевищує виправляючої здатності завадостійкого коду.

Розглянуті вище загальні принципи завадостійкого кодування показують, що процес виявлення і виправлення помилок може бути просто реалізований шляхом порівняння прийнятої комбінації зі всіма дозволеними. Незбіг прийнятої комбінації ні з однією з дозволених свідчить про наявність помилок. Виправлення помилок зводиться до відшукування комбінації, найближчої до прийнятої дозвільної комбінації.

Проте така процедура навіть при помірних значеннях n вимагає пам'яті з об'ємом, який не завжди може бути реалізований (складність пропорційна 2^n).

Основний напрям розробки теорії завадостійкого кодування полягає у пошуку таких кодів, для яких відбір різних кодових комбінацій, виявлення і виправлення помилок здійснювалися б не шляхом перебору, а за допомогою достатньо простих регулярних правил, які можуть бути реалізовані технічно з невеликими (мінімальними) апаратними витратами.

1.6.3. Кодові межі блокових і безперервних кодів

Граничні можливості кодів, що виправляють помилки, необхідно знати, по-перше, для оцінки того, наскільки реально використовувані коди гірші від "ідеальних" кодів, по-друге, для визначення характеристик систем у цілому і, по-третє, для порівняння систем різних типів. Одержимо найпростіші верхні межі для мінімальної відстані [4; 5; 7, 8].

Спочатку одержимо межу Чернова, що буде необхідна надалі.

Лема 1.1. Нехай $t < (q - 1)n / q$. Тоді:

$$\frac{q^{n\phi(t/n)}}{n+1} \leq C_t^n (q-1)^t \leq \sum_{i=0}^t C_i^n (q-1)^i \leq q^{n\phi(t/n)},$$

де $\phi(x) = x \log_q (q-1) - x \log_q x - (1-x) \log_q (1-x)$.

Доведення. Нехай $A(z) = (1 + (q-1)z)^n = \sum_{i=0}^n A_i z^i$; зрозуміло, що

$A_i = C_i^n (q-1)^i$. Якщо $0 < z \leq 1$, то:

$$\sum_{i=0}^t C_i^n (q-1)^i = \sum_{i=0}^t A_i z^{i-t} = z^{-t} \sum_{i=0}^t A_i z^i \leq z^{-t} A(z). \quad (1.48)$$

З іншого боку, оскільки відношення $A_i / A_{i+1} = (i+1) / [(n-i)(q-1)]$ є монотонно зростаючою функцією i при $0 \leq i < n$, крім того, $t < (q-1)n / q$, то знайдеться таке число $\bar{z}$, $0 < \bar{z} \leq 1$,

$$\frac{A_{t-1}}{A_t} \leq \bar{z} \leq \frac{A_t}{A_{t+1}}.$$

Оскільки $A_{i-1} / A_i \leq A_{t-1} / A_t \leq \bar{z}$ при $0 \leq i < t$ і $\bar{z} \leq A_t / A_{t+1} \leq A_i / A_{i+1}$ при $t < i$, то для всіх :

$$A_t \bar{z}^t \geq A_i \bar{z}^i$$

і, отже, $A(\bar{z}) = \sum_{i=0}^n A_i \bar{z}^i \leq (n+1) A_t \bar{z}^t$. Звідси та з формули (1.48) одержуємо:

$$\min_{0 < z \leq 1} \frac{z^{-t} A(z)}{n+1} \leq A_t < \sum_{i=0}^t A_i \leq \min_{0 < z \leq 1} z^{-t} A(z). \quad (1.49)$$

Припустимо, що $t/n = \tau$. Значення z , при якому досягається екстремум функції $z^{-n\tau} A(z) = z^{-n\tau} (1 + (q-1)z)^n$, знаходиться з рівняння:

$$-n\tau z^{-n\tau-1} (1 + (q-1)z)^n + n z^{-n\tau} (q-1) (1 + (q-1)z)^{n-1} = 0.$$

Звідси випливає, що екстремум досягається в точці $z_e = \tau / [(1-\tau)(q-1)]$. Оскільки досліджувана функція є монотонно спадною при $0 < z < z_e$ й монотонно зростаючою при $z_e < z$, то в точці z_e вона досягає мінімуму. Оскільки $t < (q-1)n / q$, те $z_e < 1$, в точці z_e розглянута функція дорівнює

$$\tau^{-n\tau} [(1-\tau)(q-1)]^{n\tau} (1-\tau)^{-n} = q^{n\phi(\tau)}.$$

Звідси та з формули (1.48) одержуємо доказ леми 1.1.

Уведена вище функція $\phi(x)$ при $q = 2$ збігається з ентропією $H(x)$.

Далі знайдемо верхню межу Хеммінга.

Теорема 1.3. Якщо існує q -й блоковий код довжини n зі швидкістю R та мінімальною відстанню Хеммінга $2t + 1$ або більше, то:

$$\sum_{i=0}^t C_i^n (q-1)^i \leq q^{n(1-R)}.$$

Доведення. Для кожного кодового слова ϖ розглянемо наступну множину $D(\varpi)$ послідовностей довжини n :

$$D(\varpi) = \{\varpi' \mid d(\varpi, \varpi') \leq t, \varpi' \in V_n\},$$

таким чином, $D(\varpi)$ включає всі послідовності, де ϖ' відрізняються від ϖ не більш ніж у t компонентах. Отже, кількість послідовностей довжини n , що входять у $D(\varpi)$, дорівнює:

$$\sum_{i=0}^t C_i^n (q-1)^i.$$

Оскільки множини $D(\varpi)$, що відповідають різним кодовим словам, не мають загальних елементів, то:

$$K \sum_{i=0}^t C_i^n (q-1)^i \leq q^n,$$

де K – загальна кількість кодових слів. Звідси та з рівняння $K = q^{nR}$ випливає справедливості теореми.

Якщо для деякого коду верхня межа Хеммінга виконується зі знаком рівності, то код називається *досконалим*. Як видно з висновку межі Хеммінга, для досконалих кодів уведено вище множини $D(\varpi)$, або, інакше кажучи, "сфери" радіуса t , центрами яких є кодові слова, попарно не перетинаються й заповнюють весь простір V_n . З огляду на це досконалі коди іноді називають також щільноупакованими кодами.

Крім тривіальних двійкових кодів, кодові слова яких виходять повторенням того самого символу деяку фіксовану кількість разів, доуконалими кодами є також наступні:

1) q -чні досконалі коди, що виправляють одиничні помилки та мають довжину $n = (q^l - 1) / (q - 1)$, $q = p^m$,

де p – прості;

2) двійковий (23,12)-код Голея, що виправляє потрійні помилки;

3) трійковий (11, 6)-код Голея, що виправляє подвійні помилки.

Твердження 1.1. Якщо існує q -чний систематичний досконалий код довжини n , то

$$n = (q^r - 1) / (q - 1),$$

де r – кількість перевірових символів.

Ідея доведення. Оскільки код систематичний, то $R = k / n$. Далі варто скористатися верхньою межею Хеммінга в окремому випадку $t = 1$.

Якщо q є ступенем простого числа, q -чні досконалі коди довжини $n = (q^r - 1) / (q - 1)$ завжди існують.

Твердження 1.2. Якщо q -чний блоковий код довжини $n = q^m - 1$ зі швидкістю $R \geq 1 - mt / n$ такий, що:

$$\frac{(t+1)!}{(q-1)^{t+1}} + \frac{(t+1)(t+2)}{2} \leq q^m,$$

тоді

$$d_{\min} \leq 2t + 2.$$

Короткий доказ.

$$\begin{aligned} C_{t+1}^{q^m-1}(q-1)^{t+1} &= \frac{[(q-1)q^m]^{t+1}}{(t+1)!} \prod_{i=1}^{t+1} (1 - iq^{-m}) \geq \frac{[(q-1)q^m]^{t+1}}{(t+1)!} \left(1 - \sum_{i=1}^{t+1} iq^{-m} \right) = \\ &= \frac{[(q-1)q^m]^{t+1}}{(t+1)!} \left(1 - \frac{q^{-m}(t+2)(t+1)}{2} \right) \geq q^{mt} \geq q^{n(1-R)}. \end{aligned}$$

Звідси та з теореми 1.3 безпосередньо випливає, що нерівність $d_{\min} \leq 2t + 3$ не може мати місця. Це твердження використовується для одержання верхньої оцінки мінімальної відстані деяких двійкових БЧХ-кодів.

Верхню межу Хеммінга можна записати в такий спосіб:

$$R \leq 1 - \frac{1}{n} \log_q \left[\sum_{i=0}^t C_i^n (q-1) \right].$$

Використовуючи нерівність Чернова, можна показати, що при фіксованому відношенні t/n та досить великих n межа Хеммінга має вигляд:

$$R \leq 1 - \phi\left(\frac{t}{n}\right). \quad (1.50)$$

Оскільки $\phi'(x) = \log_q(q-1) - \log_q x + \log_q(1-x)$, то в інтервалі $0 \leq x \leq (q-1)/q$ функція $\phi(x)$ монотонно зростає від 0 до 1. Тому для кожного $R, 0 < R < 1$ існує єдине значення x , при якому $\phi(x) = 1-R$. Позначимо це значення x через $\delta(R)$.

Наслідок 1.1. Для будь-якого блокового коду зі швидкістю R й мінімальною відстанню $d_{\min}$ при досить великих n :

$$\frac{d_{\min}}{2n} \leq \delta(R). \quad (1.51)$$

В області малих значень R верхня межа Хеммінга є досить приблизною. При малих R більш точної, ніж межа Хеммінга, є верхня межа Плоткіна, обумовлена наступною теоремою.

Теорема 1.4. Якщо існує q -чний блоковий код довжини n із загальною кількістю кодових слів K і мінімальною відстанню $d_{\min}$, то:

$$d_{\min} \leq \frac{(q-1)nK}{q(K-1)}.$$

Доведення. Ця межа виводиться шляхом оцінки зверху середньої відстані між кодовими словами. Нехай $\varpi^{(1)}, \varpi^{(2)}, \dots, \varpi^{(K)}$ – кодові слова коду, існування якого передбачається в теоремі, і нехай $\varpi_m^{(i)}$ – m -я ($1 \leq m \leq n$)– компонента кодового слова $\varpi^{(i)}$. Сума попарних відстаней Хеммінга між кодовими словами визначається наступними рівняннями:

$$D_t = \sum_{i=1}^K \sum_{j=1}^K d(\varpi^{(i)}, \varpi^{(j)}) = \sum_{i=1}^K \sum_{j=1}^K \sum_{m=1}^n d(\varpi_m^{(i)}, \varpi_m^{(j)}) = \sum_{m=1}^n \sum_{i=1}^K \sum_{j=1}^K d(\varpi_m^{(i)}, \varpi_m^{(j)}),$$

де функція $d(\varpi_m^{(i)}, \varpi_m^{(j)})$ дорівнює 1, якщо $\varpi_m^{(i)} \neq \varpi_m^{(j)}$, і дорівнює 0, якщо $\varpi_m^{(i)} = \varpi_m^{(j)}$.

Нехай $J_i^{(m)}$ – кількість символів каналу серед компонент $\varpi^{(1)}, \varpi^{(2)}, \dots, \varpi^{(K)}$. Тоді мають місце наступні рівності:

$$\sum_{i=1}^q J_i^{(m)} = K; \quad (1.52)$$

$$\sum_{i=1}^K \sum_{j=1}^K d(\varpi_m^{(i)}, \varpi_m^{(j)}) = \sum_{i=1}^q \sum_{j=1}^q \delta_{ij} J_i^{(m)} J_j^{(m)},$$

де $\delta_{ij} = 1$, якщо $i \neq j$, і $\delta_{ij} = 0$, якщо $i = j$.

Позначимо через Δ_q максимум квадратичної форми $\sum_{i=1}^q \sum_{j=1}^q \delta_{ij} x_i x_j$ за наступних умов: $\sum_{i=1}^q x_i = K$ і $x_i \geq 0$ ($1 \leq i \leq m$) (тут будемо вважати, що змінні x_i набувають дійсних значення). Методом індукції по q покажемо, що $\Delta_q = K^2(q-1)/q$. У випадку $q=2$ справедливість останнього рівняння перевіряється легко. Отримані з використанням методу невизначених множників Лагранжа значення $x_1, \dots, x_q$, при яких досягається екстремум розглянутої квадратичної форми, повинні задовольняти наступні умови:

$$\sum_{j=1}^q \delta_{ij} x_j = \lambda; \quad 1 \leq i \leq q.$$

Звідси випливає, що $x_i = K/q$ ($1 \leq i \leq q$). У цій точці значення квадратичної форми дорівнює $K^2(q-1)/q$. Оскільки розглянута область значень змінних x_i є обмеженою і замкненою, то максимум квадратичної форми досягається або в цій точці, або на межі. На межі, принаймні, одна зі змінних виявляється рівною нулю, так що ми переходимо до випадку, коли кількість змінних дорівнює $q-1$ або менше. За припущенням індукції значення функції в таких точках не перевищують $K^2(q-2)/(q-1)$. Отже, $\Delta_q = K^2(q-1)/q$. З іншого боку, мінімальна відстань $d_{\min}$ повинна задовольняти нерівність:

$$(K^2 - K)d_{\min} \leq D_t \leq nK^2(q-1)/q,$$

тому що кількість пар різних кодових слів дорівнює $K^2 - K$.

З висновку межі Плоткіна можна побачити, що кодами, мінімальна відстань яких досягає межі Плоткіна, можуть бути лише коди, у яких відстань між будь-якими двома різними кодовими словами та сама; такі коди називаються *еквідистантними*. Прикладами двійкових еквідистантних кодів є коди, що складаються з послідовностей максимальної довжини, і коди, що виходять із матриць Адамара.

Використовуючи ідеї доказу верхніх меж Хеммінга і Плоткіна, Елайс одержав нову межу для мінімальної відстані, що виявилася кращою від обох вихідних меж, принаймні, при середніх швидкостях. Ця межа надалі одержала назву межі Елайса. Розглянемо q -чний блоковий код C довжини n з K кодовими словами, швидкістю R та мінімальною відстанню Хеммінга $d_{\min}$.

Вважатимемо, що всі послідовності довжини n з V_n перенумеровані яким-небудь чином числами від 1 до q^n . Нехай s – довільне ціле додатне число і K_i – кількість кодових слів коду C , що перебувають на відстані Хеммінга s або меншій від i -тої послідовності довжини n . Оскільки кожне кодове слово перебуває на відстані s або меншій від $\sum_{i=0}^s C_i^n (q-1)^i$ послідовностей довжини n , то:

$$\sum_{i=1}^{q^n} K_i = KC_{n,q}(s), \text{ де } C_{n,q}(s) = \sum_{i=0}^s C_i^n (q-1)^i.$$

З огляду на те, що $K = q^{nR}$, одержуємо з останнього рівняння наступну оцінку знизу для $K' = \max_i K_i$:

$$K' \geq KC_{n,q}(s) / q^n = C_{n,q}(s) / q^{n(1-R)}. \quad (1.53)$$

Припустимо, що $K_j = K'$. Без обмеження спільності можна вважати, що $V_1 = \{0, 1, \dots, q-1\}$. Нехай $(a_1, a_2, \dots, a_n)$ – j -та послідовність довжини n . Розглянемо далі замість блокового коду C код:

$$C' = \{(b_1 - a_1), (b_2 - a_2), \dots, (b_n - a_n) \mid (b_1, b_2, \dots, b_n) \in C\},$$

де знак "–" означає віднімання за модулем q .

Якщо кожне кодове слово $(b_1, b_2, \dots, b_n)$ коду C зіставити з кодовим словом $(b_1 - a_1), (b_2 - a_2), \dots, (b_n - a_n)$

$(b_1 - a_1), (b_2 - a_2), \dots, (b_n - a_n)$ коду C' , то відстань Хеммінга між кодовими словами коду C буде збігатися з відстанню Хеммінга між відповідними словами коду C' . Отже, код C' має ті ж параметри $n, K, d_{\min}$, що і код C , і містить K' кодових слів, що перебувають на відстані s або меншій від послідовності $(0, 0, \dots, 0)$ довжини n . Тому замість коду C із самого початку можна було розглядати код C' , так що ми просто будемо вважати, що зазначені K' кодових слів існують у коді C . Позначимо через $J_i^{(m)}$ кількість таких кодових слів серед K' кодових слів коду C , що перебувають на відстані від послідовності $(0, 0, \dots, 0)$ довжини n , у яких i -тій компонент дорівнює i . Але тоді, як і при визначенні межі Плоткіна, одержуємо, що сума D'_t попарних відстаней Хеммінга між зазначеними вище кодовими словами визначається рівнянням:

$$D'_t = \sum_{m=1}^n \sum_{i=1}^{q-1} \sum_{j=0}^{q-1} \delta_{ij} J_i^{(m)} J_j^{(m)},$$

$$\text{де } K' = \sum_{i=0}^{q-1} J_i^{(m)}; \quad 1 \leq m \leq n.$$

Оскільки відстань Хеммінга між кожним з K' кодових слів і кодовим словом $(0, 0, \dots, 0)$ не перевищує s , то:

$$\sum_{m=1}^n \sum_{i=1}^{q-1} J_i^{(m)} \leq K' s.$$

Оскільки D'_t та $d_{\min}$, як і раніше, пов'язані між собою нерівністю:

$$(K'^2 - K') d_{\min} \leq D'_t, \quad (1.54)$$

то слід оцінити зверху D'_t . Візьмемо довільну дійсну кількість $y^{(m)}$, $0 \leq y^{(m)} \leq K'$ і знайдемо максимум $I = \sum_{i=0}^{q-1} \sum_{j=0}^{q-1} \delta_{ij} x_i x_j$ при наступних обмеженнях:

$$\begin{aligned} \sum_{i=0}^{q-1} x_i &= K'; \\ \sum_{i=1}^{q-1} x_i &= y^{(m)}; \\ x_i &\leq 0, \quad 0 \leq i < q. \end{aligned} \quad (1.55)$$

Оскільки, як випливає з формул (3.56), $x_0 = K' - y^{(m)}$, то:

$$I = \sum_{i=1}^{q-1} \sum_{j=1}^{q-1} \delta_{ij} x_i x_j + 2(K' - y^{(m)}) \sum_{i=1}^{q-1} x_i = \sum_{i=1}^{q-1} \sum_{j=1}^{q-1} \delta_{ij} x_i' x_j' - \frac{q-1}{q-2} (K' - y^{(m)})^2,$$

де $x_i' = x_i + (K' - y^{(m)}) / (q - 2)$. Обмеження на x_i' мають такий вигляд:

$$\sum_{i=1}^{q-1} x_i' = y^{(m)} + \frac{q-1}{q-2} (K' - y^{(m)}).$$

Таким чином, задача звелася до тієї ж самої задачі знаходження максимуму квадратичної форми, що розв'язувалася й при визначенні межі Плоткіна. Шуканий максимум $I(y^{(m)})$ квадратичної форми I визначається рівнянням:

$$I(y^{(m)}) = \frac{[(q-1)K' - y^{(m)}]^2}{(q-2)(q-1)} - \frac{q-1}{q-2} (K' - y^{(m)})^2 = \left(2K' - \frac{q}{q-1} y^{(m)} \right) y^{(m)}.$$

Оскільки $\sum_{m=1}^n I(y^{(m)})$ є опуклою вгору функцією $y^{(1)}, \dots, y^{(n)}$, то за допомогою методу множників Лагранжа можна показати, що максимум $\sum_{m=1}^n I(y^{(m)})$ за умови $\sum_{m=1}^n y^{(m)} \leq K's$ досягається в точці:

$$y^{(m)} = K's / n; \quad 1 \leq m \leq n$$

і дорівнює

$$K'^2 s \left(2 - \frac{qs}{(q-1)n} \right).$$

Звідси одержуємо:

$$d_{\min} \leq \frac{K's}{K'-1} \left(2 - \frac{qs}{(q-1)n} \right) \leq \frac{s C_{n,q}(s) q^{-n(1-R)}}{C_{n,q}(s) q^{-n(1-R)} - 1} \left(2 - \frac{qs}{(q-1)n} \right).$$

Далі параметр s варто вибрати так, щоб він мінімізував праву частину останньої нерівності. Однак, оскільки в загальному випадку це зробити складно, розглянемо тут випадок великих n . З леми 1 маємо:

$$q^{n\phi(s/n)} / (n+1) \leq C_{n,q}(s).$$

Якщо вибрати відношення s/n трохи більшим, ніж $\delta(R)$, і фіксувати, то при $n \rightarrow \infty$ добуток

$$C_{n,q}(s)q^{-n(1-R)}$$

буде прагнути до нескінченності, так що при великих n відношення $K'/(K'-1)$ можна вважати рівним одиниці.

Теорема 1.5. Для будь-якої швидкості передачі R і будь-якого додатного числа ε при досить великій довжині коду n :

$$\frac{d_{\min}}{n} < \delta(R) \left(2 - \frac{q\delta(R)}{q-1} \right) + \varepsilon,$$

де $\delta(R)$ – функція, уведена вище в наслідку.

Твердження 1.3. При фіксованому відношенні t/n та досить великих n отримана вище верхня межа краща від верхньої межі.

Твердження 1.4. При малих швидкостях R і досить великих n отримана вище верхня межа і межа Плоткіна майже збігаються.

Доведення. Як вказувалося вище (при введенні функції $\delta(R)$), $\delta(R) \rightarrow (q-1)/q$ при $R \rightarrow 0$. З іншого боку, $K = q^{nR} \rightarrow \infty$ при $n \rightarrow \infty$.

Таким чином, одержують верхні межі для довільних блокових кодів.

Теорема 1.6. Мінімальна вага лінійного (n, k) -коду C дорівнює d тоді й тільки тоді, коли будь-які $(d-1)$ стовпців перевірної матриці цього коду лінійно незалежні, але деякі d стовпців перевірної матриці лінійно залежні.

Прямим наслідком цієї теореми є наступне.

Наслідок 1.1 $d \leq n - k + 1$.

Далі одержимо нижню межу Варшамова – Гілберта, що гарантує існування лінійного коду із заданою кількістю перевіркових символів $r = n - k$ і заданою мінімальною відстанню d , кількість інформаційних символів якого більша від визначеної величини (спочатку Гілберт одержав цю межу для блокових кодів; Варшамов поліпшив результат Гілберта, одержавши аналогічну межу для лінійних кодів).

Теорема 1.7. Нехай q – степінь простого числа, а r та d – деякі цілі позитивні числа. Тоді існує q -чний лінійний код довжини n з r

перевірковими символами і кодовою відстанню, не меншою від d , параметри якого задовольняють нерівність:

$$q^r \leq \sum_{i=0}^{d-2} C_i^n (q-1)^i. \quad (1.56)$$

Використовуючи функцію ϕ з нерівності Чернова, можна визначити:

$$\frac{r}{n} \leq \phi\left(\frac{d-2}{n}\right). \quad (1.57)$$

Зокрема, при $q = 2$ остання нерівність має вигляд:

$$\frac{r}{n} \leq H\left(\frac{d-2}{n}\right). \quad (1.58)$$

Доведення. Як випливає з теореми, побудова матриці H , будь-які $d-1$ стовпців якої лінійно незалежні, еквівалентна побудові коду з мінімальною відстанню, не меншою від d . Побудуємо матрицю H , що задовольняє зазначені вище властивості, у такий спосіб. Спочатку довільним чином виберемо в множини $V_r' = V_r$ – (нульовий вектор) 1-й стовпчик h_1 матриці H .

У якості 2-го стовпчика h_2 матриці H візьмемо довільний вектор з V_r' , що не є добутком ah_1 , $a \in GF(q)$. Якщо в V_r' існує хоча б один вектор, що не є лінійною комбінацією h_1 й h_2 , то виберемо один із цих векторів (довільний) у якості h_3 . Припустимо, що в такий спосіб ми вибрали j векторів h_i , $0 \leq i \leq j$; за побудовою h_i не є лінійною комбінацією ніяких $d-2$ або менше стовпчиків з $h_1, \dots, h_{i-1}$. Оскільки і $(0 \leq i \leq d-2)$ векторів з $h_1, \dots, h_j$ можна вибрати C_i^j способами, а кількість способів вибору і ненульових коефіцієнтів лінійної комбінації дорівнює $(q-1)^i$, то кількість векторів, що є лінійними комбінаціями $d-2$ або менше стовпчиків з $h_1, \dots, h_j$, не більша

$$\sum_{i=1}^{d-2} C_i^j (q-1)^i.$$

Якщо ця кількість менша від загальної кількості векторів в V_r' , то в V_r' існує вектор, що не збігається з жодною із зазначених вище лінійних комбінацій. Цей вектор можна вибрати в якості h_{j+1} . До того моменту, коли новий вектор h_{n+1} вибрати вже не можна, загальна кількість n обраних раніше векторів h_i задовольняє нерівність (n є довжиною коду):

$$\sum_{i=1}^{d-2} C_i^n (q-1)^i \geq q^r - 1.$$

Із цієї нерівності та межі Чернова випливають нерівності (1.57) і (1.58).

На рис. 1.19 показане поведження верхніх меж Плоткіна, Хеммінга і Елайса, а також нижньої межі Варшамова – Гілберта при досить великих n (тут по вертикальній осі відкладається швидкість коду $R = k/n$, а по горизонтальній – відношення мінімальної відстані $d_{\min}$ к) $2n$).

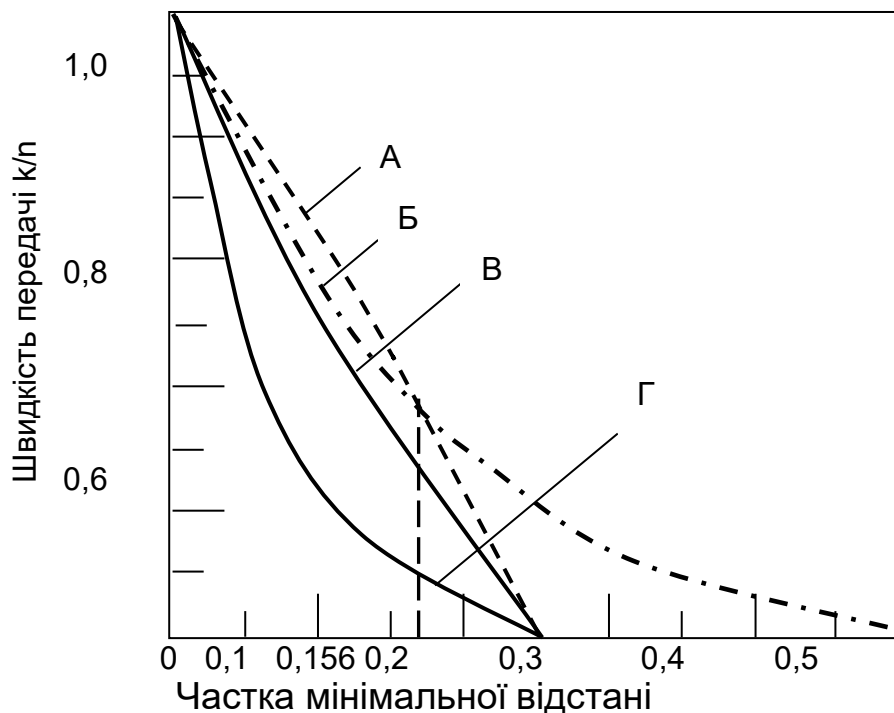


Рис. 1.19. **А** – верхня межа Плоткіна;
Б – верхня межа Хеммінга; **В** – верхня межа Елайса;
Г – нижня межа Варшамова – Гілберта

Висновок нерівності (1.57), допускає будь-який спосіб вибору векторів $h_1, \dots, h_i, \dots$, що задовольняють умовам теореми 1.4. Тому якщо задано лінійний (n, k) -код з мінімальною вагою d , параметри n, k і d якого

задовольняють нерівність (1.57), то до перевіркової матриці цього коду завжди можна додати ще один стовпчик, не порушуючи при цьому умов теореми 1.5. Інакше кажучи, зберігаючи постійними і кількість перевіркових символів, і мінімальну вагу, і збільшуючи кількість інформаційних символів, можна розширити вихідний код. Як показують нескладні розрахунки, мінімальна вага майже всіх лінійних кодів близька до величини, гарантованої нерівністю (1.57).

З нерівності (1.58) випливає, що при фіксованому відношенні d/n ($< 1/2$) й $n \rightarrow \infty$ існують лінійні коди, для яких відношення k/n більше $1 - H(d/n)$.

Тут слід зауважити, що для БЧХ-кодів, що мають гарні властивості з погляду кодування та декодування, при фіксованому d/n й $n \rightarrow \infty$ відношення k/n наближається до нуля. За винятком кодів Юстесена, всі відомі до сьогодні алгебраїчні коди мають цей недолік. Однак і для кодів Юстесена відношення k/n значно менше від величини, гарантовані межою Варшамова – Гілберта. Побудова кодів які лежать на межі Варшамова – Гілберта (або близько до неї), спосіб побудови яких можна було б задати явно (конструктивно) і які мали би гарні процедури кодування та декодування, є однією з невирішених проблем теорії кодування.

Таким чином, з розглянутого вище можна зробити висновок, що межі Хеммінга і Плоткіна є необхідними умовами існування коду, а межа Варшамова – Гілберта – достатньою.

1.6.4. Лінійні групові коди

Систематичні лінійні коди будують таким чином. Спочатку розглядається безнадмірний код довжиною k , що є множиною всіх k -ічних послідовностей двійкових символів, які називаються інформаційними. Потім до кожної з цих послідовностей, за визначеним для коду правилом приписуються $r = n - k$ перевірочних символів. Для лінійних кодів ці перевірочні символи отримують в результаті лінійних операцій над інформаційними символами.

Умовимося, що інформаційні символи розташовуються на перших k , а перевірочні на останніх $r = n - k$ позиціях кодової комбінації. Інформаційні символи будемо позначати як b_i , а перевірочні як β_i , де

$i = 1, 2, \dots, k$, $j = 1, 2, \dots, r$. Тоді i -та комбінація завадостійкого коду може бути записана у вигляді

$$B_i = b_{i1} b_{i2} \dots b_{ik} \beta_{i1} \beta_{i2} \dots \beta_{ir}.$$

Для лінійних групових кодів може виконуватися умова, що лінійна операція над будь-якими двома і більше дозволеними кодовими комбінаціями приводить до нової комбінації, що також належить до числа дозволених. Як лінійна групова операція використовується складання за модулем 2. У цьому випадку говорять, що дозвалені кодові комбінації систематичного лінійного коду утворюють групу. Звідси йде і назва цих кодів.

Очевидно, що в будь-який груповий код входить і нульова комбінація, а тому кодова відстань групового коду дорівнює мінімальній вазі ненульових комбінацій.

Побудова конкретного завадостійкого коду здійснюється, виходячи з необхідного об'єму коду і статистичних даних про кратність найімовірніших помилок у каналі передачі інформації, що використовується. Об'єм коду визначається кількістю переданих повідомлень (кількістю інформаційних розрядів k звичайного безнадмірного коду), а кількість належних виправленню помилок визначає кількість надмірних (контрольних) розрядів (символів).

Методику визначення кількості контрольних символів розглянемо на прикладі побудови кодів, що виправляють поодинокі помилки.

Якщо завадостійкий код має довжину $n = k + r$, то кількість поодиноких всіляких помилок дорівнює n . Для кодів, що виправляють помилки, знання тільки факту виникнення помилки є недостатнім. Для того, щоб виправити помилку необхідно знати ще і місце, де вона відбулася. Для цього контрольні символи повинні дозволяти виконати перевірки, за допомогою яких можна було б визначити місце помилки. Ці контрольні числа називаються пізнавачами помилок або синдромами. Тому кількість контрольних символів повинна бути такою, щоб кількість сформованих за їх допомогою синдромів була не меншою кількості можливих помилок. Отже, для виправлення поодиноких помилок необхідна кількість перевірочних символів повинна задовольняти умові

$$2^r - 1 \geq C_n^1 = n$$

і дорівнювати

$$r = \lfloor \log_2(n + 1) \rfloor. \quad (1.59)$$

Знак $\lfloor \cdot \rfloor$ тут означає, що береться округлене значення до найближчого більшого цілого числа. Розрахунки, виконані за цими співвідношеннями для кодів з $d_{\min} = 3$, зведені в табл. 1.17.

Таблиця 1.17

Результати розрахунків для лінійних групових кодів

n	k	r
3	1	2
4	1	3
5	2	3
6	3	3
7	4	3
8	4	4
9	5	4
10	6	4
11	7	4
12	8	4
13	9	4
14	10	4
15	11	4

Формулу (1.59) зручно використовувати, якщо відома повна довжина коду n . Якщо ж відома кількість інформаційних символів k , то розрахунки ведуться за емпіричною формулою

$$r = \left\lfloor \log_2 \left((k + 1) + \left\lfloor \log_2(k + 1) \right\rfloor \right) \right\rfloor.$$

Найбільш економічними з погляду мінімізації надмірних символів при максимумі інформаційних символів є коди, для яких $\log_2(n + 1)$ виражається цілим числом, наприклад коди (3,1); (7,4); (15,11); (31,26); (63,57) і т.д. Надмірність цих кодів повністю витрачається на виправлення всіх однократних помилок. Ніяка помилка вищої кратності не може бути виправлена.

Коди з іншим розподілом інформаційних і контрольних символів, наприклад коди (8,4), (9,4) можуть виправляти всі поодинокі помилки і частину подвійних.

Код, що виправляє всі помилки кратності $t \leq t_{\text{вип}}$ і не виправляє жодного поєднання помилок вищої кратності $t = t_{\text{вип}} + 1$, називається досконалим. Код, що виправляє всі поєднання помилок $t \leq t_{\text{вип}} + 1$, але не виправляє жодного поєднання помилок вищої кратності $t \geq t_{\text{вип}} + 1$, називається квазідосконалим.

Якщо необхідно виправляти не тільки поодинокі помилки, а і всі подвійні незалежні помилки, то кількість перевірочних символів визначається за співвідношенням

$$2^r - 1 \geq C_n^1 + C_n^2.$$

У загальному випадку для виправлення всіх незалежних помилок, до кратності t включно, кількість перевірочних символів визначається як

$$2^r - 1 \geq C_n^1 + C_n^2 + \dots + C_n^t.$$

В наведених співвідношеннях указується теоретична межа мінімально можливої кількості перевірочних символів, яку не завжди вдається досягнути на практиці. Щоб уникнути неоднозначності при декодуванні помилок різної кратності, кількість перевірочних символів часто доводиться брати більшою, ніж це впливає з відповідних співвідношень.

Правило побудови лінійного коду повністю визначається заданням функціональної залежності набору перевірочних символів від інформаційних. Для лінійних кодів ця залежність буде лінійною

$$\beta_{ij} = \sum_{i=1}^k c_{ij} b_{li}, \quad j=1, 2, \dots, r. \quad (1.60)$$

де c_{ij} – коефіцієнти, що можуть приймати значення 0 або 1, додавання яких виконується за модулем 2.

Записавши (1.60) в розгорнутому вигляді, одержимо r співвідношень, при допомозі яких за інформаційними символами кодової комбінації знаходяться перевірочні

$$\begin{aligned} \beta_{11} &= c_{11}b_{l1} + c_{21}b_{l2} + \dots + c_{k1}b_{lk}; \\ \beta_{12} &= c_{12}b_{l1} + c_{22}b_{l2} + \dots + c_{k2}b_{lk}; \\ &\dots\dots\dots \\ \beta_{1r} &= c_{1r}b_{l1} + c_{2r}b_{l2} + \dots + c_{kr}b_{lk}. \end{aligned} \quad (1.61)$$

Задання g співвідношень вигляду (1.61) повністю визначає правило побудови лінійного коду. Ці співвідношення в свою чергу визначаються заданням kr двійкових коефіцієнтів c_{ij} . Задавши kr двійкових коефіцієнтів c_{ij} і підставляючи у співвідношення (1.61) по черзі всі інформаційні комбінації, можна одержати значення перевірочних символів для кожної з них і, таким чином, сформулювати всі комбінації завадостійкого коду.

Лінійні групові коди можуть бути описані (задані) компактніше за допомогою матриць. Раніше наголошувалося, що всі 2^k дозволених комбінацій лінійного групового коду утворюють групу. З цієї групи можна виділити k незалежних комбінацій і об'єднати їх у матрицю розмірністю $k \times n$ (k рядків та n стовпців):

$$G_{n \times k} = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1k} & \beta_{11} & \beta_{12} & \dots & \beta_{1r} \\ b_{21} & b_{22} & \dots & b_{2k} & \beta_{21} & \beta_{22} & \dots & \beta_{2r} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ b_{k1} & b_{k2} & \dots & b_{kk} & \beta_{k1} & \beta_{k2} & \dots & \beta_{kr} \end{bmatrix}.$$

Лінійна незалежність комбінацій (рядків) $G_{n \times k}$ означає, що жодна з них не може бути створена додаванням інших рядків.

Матриця $G_{n \times k}$ називається породжувальною, оскільки дозволяє одержати всі дозвалені комбінації коду шляхом додавання за модулем 2 усіх можливих поєднань рядків (по два, по три і т.д.).

Породжувальну матрицю можна представити двома матрицями – інформаційною, яка містить тільки інформаційні символи, і додатковою (перевірочною), що містить перевірочні розряди. Інформаційна матриця квадратна, розмірністю $k \times k$, розмірність додаткової матриці $k \times r$.

Як інформаційну матрицю дуже зручно взяти одиничну матрицю

$$\begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{bmatrix}.$$

В цьому випадку породжувальна матриця лінійного коду називається канонічною. Будь-яку породжувальну матрицю можна звести до канонічного вигляду

$$G_{n \times k} = \begin{vmatrix} 1 & 0 & 0 & \dots & 0 & \beta_{11} & \beta_{12} & \dots & \beta_{1r} \\ 0 & 1 & 0 & \dots & 0 & \beta_{21} & \beta_{22} & \dots & \beta_{2r} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 & \beta_{k1} & \beta_{k2} & \dots & \beta_{kr} \end{vmatrix}.$$

Встановимо зв'язок між коефіцієнтами c_{ij} співвідношення (1.61) з символами β_{ij} додаткової матриці. Для цього у співвідношення (1.61) по черзі підставимо рядки інформаційної (одиначної) матриці

$$\begin{aligned} \beta_{11} &= c_{11} \cdot 1 + c_{21} \cdot 0 + \dots + c_{k1} \cdot 0 = c_{11}; \\ \beta_{12} &= c_{12} \cdot 1 + c_{22} \cdot 0 + \dots + c_{k2} \cdot 0 = c_{12}; \\ \beta_{13} &= c_{13} \cdot 1 + c_{23} \cdot 0 + \dots + c_{k3} \cdot 0 = c_{13}; \\ &\dots \\ \beta_{1r} &= c_{1r} \cdot 1 + c_{2r} \cdot 0 + \dots + c_{kr} \cdot 0 = c_{1r}. \end{aligned}$$

Виконавши аналогічні операції для інших рядків одиначної матриці, встановимо тотожність коефіцієнтів c_{ij} і перевірочних символів β_{ij} , де $i = 1$.

Отже, перевірочні символи всіх дозволених комбінацій виражаються через перевірочні символи породжувальної матриці і інформаційні символи кодової комбінації, що дозволяє співвідношення (6.4) переписати в еквівалентному вигляді

$$\begin{aligned} \beta_{1_1} &= \beta_{11}b_{1_1} + \beta_{21}b_{1_2} + \dots + \beta_{k1}b_{1_k}; \\ \beta_{1_2} &= \beta_{12}b_{1_1} + \beta_{22}b_{1_2} + \dots + \beta_{k2}b_{1_k}; \\ &\dots \\ \beta_{1_r} &= \beta_{1r}b_{1_1} + \beta_{2r}b_{1_2} + \dots + \beta_{kr}b_{1_k}. \end{aligned} \tag{1.62}$$

Таким чином, лінійний (n, k) код визначається kr двійковими числами, які у принципі можуть виражатися довільно. Тому при заданих n і k можуть бути побудовані 2^k різних лінійних (n, k) коди. Задача полягає в тому, щоб вибрати оптимальний варіант коду. Зробити це простим перебором трудно з причини великої кількості варіантів. Так, наприклад, при $n = 30$, $k = 20$ кількість різних лінійних кодів дуже велика ($2^{20} > 10^6$). Для того, щоб одержати код із заданим $d_{\min}$, не вдаючись до повного перебору, на коефіцієнти β_{ij} накладаються деякі обмеження, які враховуються при виборі додаткової матриці.

Резонно вважати, що елементи додаткової матриці необхідно вибирати так, щоб забезпечити максимально можливу відстань між

дозволеними комбінаціями. Для цього ваги рядків додаткової матриці повинні бути великими. З другого боку, не можна всі рядки додаткової матриці зробити одиничними, оскільки в цьому випадку відстань між дозволеними комбінаціями визначатиметься тільки відстанню між рядками інформаційної матриці, яка дорівнює двом. Тому бажано, щоб рядки додаткової матриці маючи велику вагу, в той же час відрізнялися як можливо більшою кількістю розрядів. Ці очевидні міркування дозволили сформулювати вимоги до додаткової матриці.

Для того, щоб матриця розміром $k \times r$ була додатковою для (n, k) коду із заданою мінімальною відстанню $d_{\min}$, необхідно і достатньо, щоб комбінація, що виходить в результаті додавання за модулем 2 l її рядків ($1 \leq l \leq k$), містила не менше, ніж $d_{\min} - 1$ відмінних від нуля символів. Дійсно, серед k інформаційних символів кожного рядка породжувальної матриці є тільки один, відмінний від нуля, при цьому для різних комбінацій він розташовується на різних позиціях. Тому серед перших k символів комбінації B_l , одержаних в результаті додавання за модулем 2 l її рядків, завжди знайдеться точно l відмінних від нуля символів. А кожна комбінація B_l групового коду повинна мати вагу не меншу, ніж $d_{\min}$, отже, серед її останніх r символів повинно бути не менше, ніж $d_{\min} - 1$ відмінних від нуля.

Через вказані обставини вага кожного рядка додаткової матриці повинна бути не меншою $d_{\min} - 1$, а рядки повинні відрізнятися між собою не менше, ніж в $d_{\min} - 2$ позиціях. Вказані положення справедливі для кодів з будь-якою кодовою відстанню.

Розглянемо приклад побудови породжувальної матриці лінійного групового коду, що виправляє всі поодинокі помилки при передачі 16 повідомлень.

Для передачі 16 повідомлень необхідно мати чотири інформаційні розряди ($16 = 2^4$). Для кодів, що виправляють одну помилку ($d_{\min} = 3$), кількість перевірочних символів наведена в табл. 6.1. При $k = 4$ $r = 3$. Тоді $n = k + r = 4 + 3 = 7$. Враховуючи, що вага кожного рядка додаткової матриці повинна бути не менше $d_{\min} - 1 = 3 - 1 = 2$, вибираємо, як рядки додаткової матриці, тризначні комбінації з числом одиниць, більшим або рівним 2: 111, 110, 101, 011.

Побудована матриця має вигляд

$$G_{7 \times 4} = \begin{bmatrix} 1000111 \\ 0100110 \\ 0010101 \\ 0001011 \end{bmatrix}, \text{ або } G_{7 \times 4} = \begin{bmatrix} 1000011 \\ 0100101 \\ 0010110 \\ 0001111 \end{bmatrix}.$$

Виявлення і виправлення помилок лінійним кодом засноване на перевірці виконання співвідношень (1.62) для символів прийнятої комбінації $B^* = b_1^* b_2^* \dots b_k^* \beta_1^* \beta_2^* \dots \beta_r^*$ та на аналізі результатів цієї перевірки.

Для цього з прийнятих інформаційних символів відповідно до співвідношення (1.62), тобто за такими ж, що й на передавальній стороні, правилами формуються перевірочні розряди β_{ij}^* , які порівнюють з прийнятими контрольними $\beta_j^* + \beta_j^{**} = S_j$. Кількість перевірок дорівнює кількості перевірочних символів.

У результаті перевірок утворюється контрольне число $S_1 S_2 \dots S_r$, яке називається синдромом або пізнавачем, де S_j – двійковий символ, що дорівнює нулю, якщо в системі перевірок (1.62) j -а рівність виконана, і одиниці, якщо вона не виконана. Нульовий синдром указує на те, що символи прийняті правильно або відбулася помилка, що не знаходиться. Якщо хоча б один розряд контрольного числа містить одиницю, то в прийнятій комбінації є помилка. Оскільки спотворення одного з елементів b_i (або β_j) одержує цілком певну рівність, то кожному з $2^r - 1$ ненульових синдромів відповідає своя помилка, яка і виправляється.

Розглянемо на прикладі, яким чином кожному синдрому ставиться у відповідність своя помилка. Хай груповий код побудований за такою матрицею:

$$G_{7 \times 4} = \begin{bmatrix} 1000111 \\ 0100110 \\ 0010101 \\ 0001011 \end{bmatrix} = \begin{bmatrix} 1000 \\ 0100 \\ 0010 \\ 0001 \end{bmatrix} \cdot \begin{bmatrix} 111 \\ 110 \\ 101 \\ 011 \end{bmatrix}.$$

Система перевірок для цього коду має вигляд

$$\begin{aligned} \beta_1^* + \beta_{11} b_1^* + \beta_{21} b_2^* + \beta_{31} b_3^* + \beta_{41} b_4^* &= b_1^* + b_2^* + b_3^* = S_1; \\ \beta_2^* + \beta_{12} b_1^* + \beta_{22} b_2^* + \beta_{32} b_3^* + \beta_{42} b_4^* &= b_1^* + b_2^* + b_4^* = S_2; \\ \beta_3^* + \beta_{13} b_1^* + \beta_{23} b_2^* + \beta_{33} b_3^* + \beta_{43} b_4^* &= b_1^* + b_3^* + b_4^* = S_3. \end{aligned}$$

Взявши до уваги, що код призначений тільки для виправлення однієї помилки, проаналізуємо одержану систему перевірок.

Символ β_1^* входить тільки в перше рівняння. Отже, якщо він спотворений, то синдром дорівнює 100. Якщо спотворені символи β_2^* і β_3^* , кожний з яких також входить тільки в одне рівняння, синдроми мають вигляд 010 і 001. Якщо спотворений символ b_1^* , що входить у всі три рівняння, то виникає синдром 111. Спотвореним символам b_2^* , b_3^* і b_4^* , кожний з яких входить тільки до двох певних рівнянь, відповідають синдроми 110, 101 і 011. Визначивши за допомогою синдрому, який символ спотворений, змінюють його значення і, тим самим, відновлюють правильну інформацію.

Хай передавалися кодові комбінації 0110011, 1110100, 1101010, що належать цьому коду. Через завади відбулися спотворення в першому розряді першої комбінації, в першому розряді другої комбінації і шостому розряді третьої комбінації, тобто вони прийняті у вигляді 1110011, 0110100, 1101000.

Знаходимо контрольні числа відповідно до системи перевірок. Для першої комбінації

$$\beta_1^* \oplus b_1^* \oplus b_2^* \oplus b_3^* = 0 \oplus 1 \oplus 1 \oplus 1 = 1;$$

$$\beta_2^* \oplus b_1^* \oplus b_2^* \oplus b_4^* = 1 \oplus 1 \oplus 1 \oplus 0 = 1;$$

$$\beta_3^* \oplus b_1^* \oplus b_3^* \oplus b_4^* = 1 \oplus 1 \oplus 1 \oplus 0 = 1.$$

Синдром 111 показує, що помилка відбулася в першому розряді і символ 1 необхідно замінити 0.

Для другої комбінації

$$1 \oplus 0 \oplus 1 \oplus 1 = 1;$$

$$0 \oplus 0 \oplus 1 \oplus 0 = 1;$$

$$0 \oplus 0 \oplus 1 \oplus 0 = 1.$$

Синдром 111 показує, що помилка в першому розряді.

Для третьої комбінації

$$0 \oplus 1 \oplus 1 \oplus 0 = 0;$$

$$0 \oplus 1 \oplus 1 \oplus 1 = 1;$$

$$0 \oplus 1 \oplus 0 \oplus 1 = 0.$$

Синдром 010 показує, що помилка в шостому розряді.

Синдром можна побудувати так, щоб одержаний при перевірках результат $S_1 S_2 \dots S_r$ прямо указував номер спотвореного розряду.

Таким чином, перевагою лінійних групових кодів є те, що в декодері не потрібно зберігати всі 2^k дозволених кодових комбінацій. Достатньо зберігати тільки k рядків додаткової матриці і таблицю синдромів. Проте, при $n \geq 100$ і декількох помилках, що виправляються, реалізація декодуючих пристроїв стає проблематичною.

1.6.5. Кодуючі і декодуючі пристрої лінійних групових кодів

Найбільш широке поширення одержали кодуючі і декодуючі пристрої, виконані на основі регістрів зсуву. Реалізація цього методу проста і дозволяє створити високонадійні схеми на базі інтегральної технології і забезпечує можливість зміни параметрів кодових комбінацій.

Кодуючий пристрій, що формує лінійний (n, k) код, складається з n -розрядного регістра зсуву і r суматорів за модулем 2, що зображено на рис. 1.20.

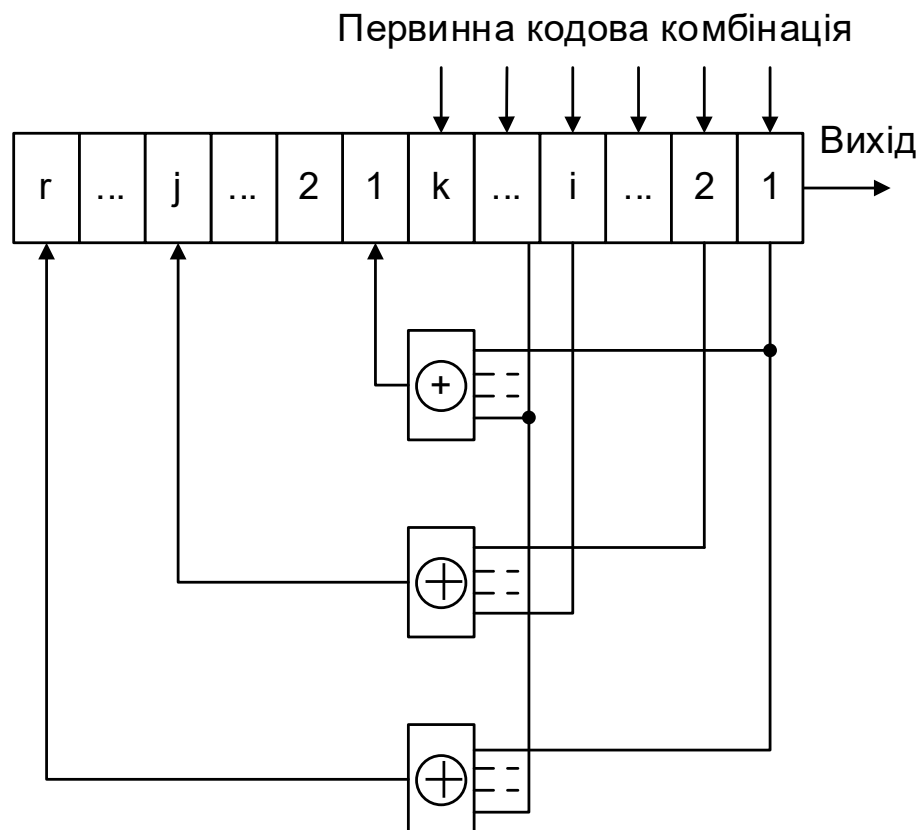


Рис. 1.20. Структурна схема кодуєчого пристрою лінійного (n, k) коду

У регістрі k чарунок призначені для інформаційних розрядів, а інші g чарунок – для перевірочних розрядів. Кожний з g суматорів за модулем 2 служить для формування перевірочного символу, що знаходиться на визначеній позиції. Інформаційні чарунки регістра підключаються до відповідних суматорів за правилами побудови конкретного завадостійкого коду.

Пристрій працює таким чином. Кодова комбінація первинного (безнадмірного) коду записується паралельно в k інформаційних чарунках регістра. Далі у всіх g двійкових суматорах формуються перевірочні символи, що записуються в g перевірочних чарунках регістра. На цьому формування кодової комбінації завадостійкого коду закінчується і починається її вивід з регістра шляхом подачі на нього імпульсів зсуву (тактових імпульсів) від генератора тактових імпульсів (на рис. 1.20 і далі на всіх схемах кодуєчих і декодуєчих пристроїв, в основі яких лежать регістри зсуву, з метою спрощення рисунків генератори тактових імпульсів не показані).

Після n тактових імпульсів кодова комбінація виводиться з регістра, а сам регістр готовий до запису і формування наступної кодової комбінації.

Розглянемо принцип роботи декодуєчого пристрою, структурна схема якого зображена на рис. 1.21. Пристрій складається з n -розрядного приймального регістра, g суматорів за модулем 2 і аналізатора помилок. Прийнята кодова комбінація, частина символів якої може бути спотворена завадою, записується в n чарунок регістра. Суматори за модулем 2 підключаються до чарунок регістра зсуву відповідно до перевірочних рівностей для даного коду. Якщо в прийнятій кодовій комбінації помилок немає, то на виходах усіх суматорів формуються нулі.

При наявності в прийнятій кодовій комбінації спотворень, на виходах деяких суматорів з'являться одиниці. У цьому випадку аналізатор помилок сформує сигнал "помилка".

Для виправлення помилок схема декодуєчого пристрою повинна бути ускладнена в порівнянні з розглянутою, тому що в неї необхідно ввести ряд додаткових вузлів: пристрій для запам'ятовування кодової комбінації, що виправляється, логічні пристрої для визначення спотворених розрядів і їхнього виправлення тощо.

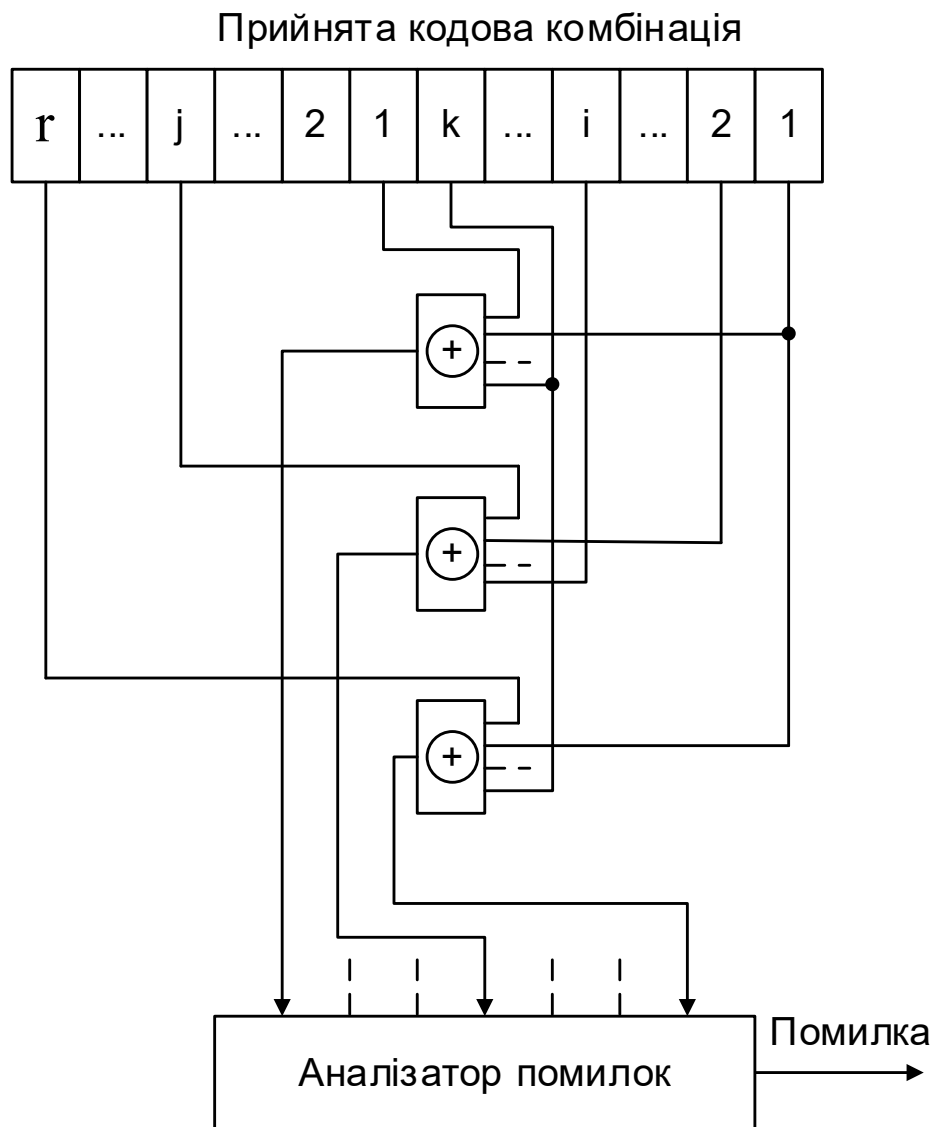


Рис. 1.21. Структурна схема декодуючого пристрою лінійного (n, k) коду

При великому числі надмірних символів і досить довгих кодах декодуючий пристрій виходить дуже складним.

1.6.6. Лінійні циклічні коди

Перевагою використання лінійних групових кодів є те, що в кодері і декодері не потрібно зберігати всі 2^k різних комбінацій, а при декодуванні не потрібно обчислювати 2^k кодових відстаней. Достатньо зберігати k рядків додаткової матриці, а в процесі кодування перевіряти r рівностей. В порівнянні з табличним способом це спрощує кодування і декодування з виявленням помилок. Проте, для виправлення помилок необхідно в

декодері перебрати таблицю синдромів, об'єм якої швидко зростає зі збільшенням довжини коду і кількості помилок, що виправляються. Це призводить до того, що при значеннях n рівних декільком десяткам реалізація декодувального пристрою може стати утрудненою.

Спростити техніку кодування і особливо декодування вдалося, коли серед лінійних групових кодів було знайдено один з класів кодів, який отримав назву циклічного. Особливістю циклічного коду, через яку він отримав свою назву, є те, що k лінійно незалежних рядків його матриці додатково пов'язані умовою циклічності. Це означає, що всі лінійно незалежні комбінації коду можуть бути отримані послідовним зсувом на один розряд єдиної "базової" комбінації. Кількість можливих циклічних (n, k) кодів значно менше кількості різних групових (n, k) кодів. Проте, властивість циклічності дозволяє значно зменшити об'єм пам'яті кодера і декодера при виявленні помилок і реалізувати виправлення поодиноких помилок так же просто, як і виявлення. Виправлення багатократних помилок при використанні циклічних кодів може бути реалізоване також простіше, ніж при довільних лінійних кодах. Все це і визначило широке застосування циклічних кодів в сучасних системах передачі інформації.

Формування циклічних кодів засновано на відображенні кодових комбінацій елементами алгебри многочленів. Основою такого опису є подання кодової комбінації довжини n як многочлена степеня $n-1$, що містить фіктивну перемінну x . Показники степеня перемінної x відповідають номерам розрядів (позиціям), а коефіцієнтами при x є символи 1 або 0 (для двійкових комбінацій). При цьому найменшому розряду числа відповідає фіктивна перемінна $x^0 = 1$.

Запишемо, наприклад, у вигляді многочлена п'ятирозрядну двійкову кодову комбінацію 10101

$$B(x) = 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x + 1 \cdot x^0.$$

Виконуючи операцію множення перемінних x на відповідні коефіцієнти, отримаємо

$$B(x) = x^4 + x^2 + 1.$$

Подання кодових комбінацій як многочленів дозволяє крім операцій додавання і множення на символи β_{ij} , які використовувалися при формуванні довільних лінійних кодів, ввести ще й операції множення і

ділення многочленів і отримати за рахунок цього додаткові ознаки для виявлення дозволених комбінацій.

Суть циклічного кодування зводиться до наступного. Дозволені кодові комбінації (дозволені многочлени) циклічного коду створюються шляхом множення деякого многочлена (утворюючого) $P(x)$ на многочлени $B_i(x)$, які являють собою комбінації безнадмірного коду. На приймальній стороні ухвалена комбінація піддається зворотному перетворенню її многочлена на многочлен $P(x)$. Якщо при діленні не утворюється остачі (остача від ділення дорівнює нулю), комбінація вважається дозволеною і повідомлення, яке міститься в ній декодується.

Такий випадок можливий за відсутності завади або при такій заваді, яка трансформує одну дозволена комбінацію в іншу. Якщо при діленні утворюється не нульова остача, комбінація вважається забороненою. Наявність остачі дозволяє виявити помилку, а вид остачі – визначити номер (або номери) помилкового розряду, тобто виправити помилку.

Нагадаємо основні правила, за якими виконуються операції алгебри над многочленами з двійковими коефіцієнтами.

Додавання многочленів зводиться до складання за модулем 2 коефіцієнтів при членах з однаковими степенями.

Як приклад, розглянемо суму многочленів $B_1(x) = x^6 + x^5 + x^3 + x^2 + 1$ і $B_2(x) = x^5 + x^4 + x^2$.

$$B_1(x) \rightarrow 1101101$$

$$B_2(x) \rightarrow \underline{0110100}$$

$$B_1(x) + B_2(x) \rightarrow 1011001 \rightarrow x^6 + x^4 + x^3 + 1$$

У полі двійкових чисел операції додавання і віднімання (за модулем 2) еквівалентні. Це дозволяє при всіх операціях над многочленами (включаючи і ділення) користуватися тільки додаванням за модулем 2, а при перенесенні членів рівності з однієї частини в іншу не змінювати знаки.

Ділення многочленів виконується звичайним алгебраїчним способом, тільки замість віднімання проміжних результатів їх додають за модулем 2.

Розглянемо приклад.

Розділимо многочлен $x^6 + x^4 + x^3 + 1 \rightarrow 1011001$ на многочлен $x^4 + 1 \rightarrow 10001$.

$$\begin{array}{r}
 x^6 + x^4 + x^3 \qquad \qquad \qquad | \ x^4 + 1 \\
 \underline{x^6} \qquad \qquad \qquad + x^2 \qquad \qquad \qquad | \underline{x^2 + 1} \\
 \qquad x^4 + x^3 + x^2 \qquad \qquad \qquad \\
 \qquad \underline{x^4} \qquad \qquad \qquad + 1 \\
 \qquad \qquad x^3 + x^2 \qquad \qquad + 1 \leftarrow \text{Залишок}
 \end{array}$$

Ділення многочленів виконується до тих пір, поки степінь останньої остачі не стане меншим степеня дільника.

Визначимо тепер операцію множення. Незавжди встановити, що операція множення многочленів (з перенесенням), може дати многочлени більш високого степеня, ніж $n-1$. Відповідні їм кодові комбінації будуть мати кількість розрядів більшу, ніж n , і вони не будуть належати до цієї множини. Тому для циклічного коду вводиться операція, яка отримала назву символічного множення. При виконанні цієї операції многочлени перемножуються за звичайними правилами, але з приведенням подібних за модулем 2. Якщо старший степінь добутку не перевищує $n-1$, то це перетворення і є результатом символічного множення. Якщо старший степінь перетворення більший або рівний n , то многочлен перетворення ділиться на многочлен $x^n + 1$ і результатом символічного множення вважається остача від ділення (тобто символічне множення є множенням за модулем $x^n + 1$). При такому множенні перетворення будь-яких двох многочленів з групи многочленів ступеня $n-1$ має степінь, яка не перевищує $n-1$, тобто степінь помилки завжди менший степеня дільника. Крім того, множення за модулем $x^n + 1$ дозволяє математично записати й операцію циклічного зсуву кодового многочлена $V(x)$. Цей зсув еквівалентний простому множенню (за модулем $x^n + 1$) $V(x)$ на x і не призводить до появи многочленів степеня більшого, ніж $n-1$.

Розглянемо приклад. Нехай кодовий многочлен має вигляд $V(x) = x + 1 \rightarrow 00 \dots 011$. Множення його на x призводить до появи нового многочлена і нової комбінації, яка є циклічним зсувом попередньої:

$$V_1(x) = xV(x) = x(x + 1) = x^2 + x \rightarrow 00 \dots 0110.$$

Наступний зсув переміщує одиниці кодової комбінації ще на один розряд вліво

$$B_2(x) = x^2B(x) = xB_1(x) = x(x^2 + x) = x^3 + x^2 \rightarrow 00 \dots 01100.$$

Степінь отриманих многочленів менший n і відповідний многочлен не виходить за межі групи. Але після перемноження $x(x^{n-1} + x^{n-2})$ одержуємо многочлен $x^n + x^{n-1}$, що виходить за межі групи. Виконуючи символічне множення, розділимо цей многочлен на $x^n + 1$.

Остача від ділення $x^{n-1} + 1 \rightarrow 10 \dots 01$, по-перше, не виходить за межі групи; а по-друге, являє собою результат чергового циклічного зсуву.

Існують декілька різних способів кодування циклічних кодів. Принципово просто комбінації циклічного коду можна отримати, помножаючи многочлени $B_i(x)$, відповідні комбінаціям безнадмірного коду (інформаційним символам), на утворюючий многочлен $P(x)$. Такий спосіб легко реалізується, але він має недолік, який полягає в тому, що комбінації коду, які виходять, в результаті множення, не містять інформаційних символів в явному вигляді, тобто такий код є нероздільним. Внаслідок цього після виправлення помилок ухвалені комбінації необхідно додатково ділити на утворюючий многочлен.

Звичайно прагнуть отримати код роздільного типу, тобто у такому вигляді, коли під інформаційні символи відводиться k членів із старшими степенями многочлена відповідного коду, а під r контрольних символів – члени, що залишилися, з низькими степенями. Щоб отримати такий роздільний (систематичний) код, застосовується наступна процедура кодування.

Многочлен $B_i(x)$, відповідний k -розрядній комбінації безнадмірного коду, помножується на x^m , де $m = n - k$, що еквівалентно дописуванню справа до комбінації безнадмірного коду m нулів. Потім перетворення $x^m B_i(x)$ ділиться на породжувальний многочлен. В загальному випадку результат ділення складається з цілого числа $Q(x)$ і остачі $R(x)$

$$x^m B_i(x) = Q(x)P(x) + R(x).$$

Обчислена остача $R(x)$ використовується для утворення циклічного коду у вигляді суми

$$B(x) = x^m B_i(x) + R(x).$$

Оскільки додавання і віднімання за модулем 2 дають один і той же результат, то неважко помітити, що $B(x) = Q(x)P(x)$, тобто отримана комбінація задовольняє вимогам подільності на породжувальний многочлен.

Оскільки степінь многочлена $P(x)$ вибирається рівним m , степінь остачі $R(x)$ не перевищує $m-1$ і, отже, вказана операція додавання рівносильна приписуванню остачі $R(x)$, виконуючої роль перевірочних символів, до $B_i(x)$ збоку молодших розрядів.

Розглянемо приклад. Закодувати циклічним кодом з $d_{\min} = 3$ чотирирозрядну інформаційну комбінацію $1001 \rightarrow B_i(x) = x^3 + 1$.

Визначивши необхідну кількість контрольних символів $r = 3$, з таблиці примітивних многочленів вибираємо породжувальний многочлен третього степеня $P(x) = x^3 + x^2 + 1 \rightarrow 1101$. Помножуючи x^m на $B_i(x)$, одержуємо $x^3(x^3 + 1) = x^6 + x^3$ (для двійкової комбінації це означає просте приписування справа трьох нулів: 1001000). Розділивши $x^m B_i(x)$ на породжувальний многочлен, знаходимо остачу, яка дорівнює $x + 1$.

Оскільки остачі є контрольними розрядами, то їхні довжини повинні дорівнювати r . У разі браку розряду до остачі приписується зліва необхідна кількість нулів. Тому остача $x + 1 \rightarrow 11$ повинна бути записана як 011.

Тоді $B(x) = x^m B_i(x) + R(x) = x^6 + x^3 + x + 1$ є кодовою комбінацією циклічного коду і має вигляд 1001011 (для двійкової комбінації приписані три нулі необхідно замінити остачею). В отриманій комбінації 1001 – інформаційні символи, 011 – контрольні.

Якби комбінація циклічного коду формувалася множенням інформаційної комбінації (многочлена) на породжувальний многочлен, то вона мала б вигляд 1100101. Інформаційні і контрольні символи тут нероздільні.

Розглянутим способом можна закодувати всі повідомлення, які треба передати. Проте для повного опису (визначення) всіх кодових комбінацій циклічного коду достатньо, як і для будь-якого групового коду, записати тільки лінійно незалежні комбінації і об'єднати їх в породжувальну матрицю. Породжувальна матриця містить k рядків, які доцільно використовувати як рядки одиничної матриці зі своїми остачами. Побудова матриці циклічного коду, зводиться, таким чином, до складання одиничної матриці, множення кожного її рядка на x^r (для двійкової

комбінації – до добавки справа r нулів) і отримання додаткової матриці остач шляхом ділення $x^r B_i(x)$ на породжувальний многочлен $P(x)$.

Розглянемо приклад. Скласти породжувальну матрицю циклічного коду $(7, 4)$. Помноживши рядки одиничної матриці

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

на $x^r = x^3$, отримаємо матрицю проміжних комбінацій з $n = k + r = 7$ стовпців і $k = 4$ рядків

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \cdot$$

Після ділення кожного рядка цієї матриці на породжувальний многочлен $P(x) = x^3 + x^2 + 1$ одержуємо остачі 110, 011, 111, 101, що використовуються як додаткова матриця. Породжувальна матриця має вигляд

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \cdot$$

Рядки цієї матриці являють собою перші комбінації шуканого коду. Інші комбінації виходять в результаті підсумовування за модулем 2 всіляких поєднань її рядків.

Відзначимо, що породжувальна матриця може бути побудована і в результаті безпосереднього множення одиничної матриці на породжувальний многочлен:

$$\begin{aligned} 1000 \times 1101 &= 1101000; \\ 0100 \times 1101 &= 0110100; \\ 0010 \times 1101 &= 0010100; \\ 0001 \times 1101 &= 0001101. \end{aligned}$$

Повний набір кодових комбінацій буде мати такий же вигляд, як і в раніше розглянутому випадку.

Весь набір дозволених кодових комбінацій, отриманий за допомогою породжувальної матриці дозволяє закодувати 2^k повідомлень, де для розглянутих кодів k приймає такі дискретні значення: 4 – для коду (7, 4), 11 – для коду (15, 11) тощо. При необхідності можна зменшити кількість комбінацій коду, побудувавши укорочений циклічний код. Для цього в породжувальній матриці необхідно викреслити j перших рядків і стільки ж стовпців зліва. Отриманий укорочений циклічний $(n - k, k - j)$ код має такі ж характеристики, що і початковий.

Дія завад на кодові комбінації може бути описана за допомогою многочленів помилок $N_j(x)$. Члени многочлена $N_j(x)$ розташовані на позиціях, відповідних спотвореним розрядам. Тому завада, що спотворює один $v + 1$ розряд кодової комбінації, може бути описана одночленом $N_v(x) = x^v$. Завада, що спотворює $v + 1$ та $\mu + 1$ ($v > \mu$) розряди має вигляд $N_{v,\mu}(x) = x^v + x^\mu$. Взаємодія кодової комбінації з завадою призводить до того, що на приймальній стороні спотворена комбінація описується многочленом $B^*(x) = B(x) + N(x)$. Помилки в циклічних кодах виявляються і виправляються за допомогою остачі від ділення отриманої комбінації на породжувальний многочлен. Якщо при передачі по каналу передачі інформації помилок не було, то $B^*(x) = B(x)$ і ухвалений многочлен ділиться на $P(x)$ без остачі. Якщо ухвалена комбінація, не ділиться на $P(x)$ без остачі, то це свідчить про те, що виникла помилка.

Остача, що виникає при діленні многочлена $B^*(x)$ на породжувальний многочлен $P(x)$, дорівнює остачі від ділення $N_j(x)$ на $P(x)$. Дійсно

$$\frac{B^*(x)}{P(x)} = \frac{B_i(x)}{P(x)} = \frac{N_j(x)}{P(x)}. \quad (1.63)$$

Отже, помилка може бути виявлена в тому і лише в тому випадку, якщо многочлен помилок не ділиться без остачі на $P(x)$.

Для виправлення виявлених помилок код будується так, щоб многочлени помилок, які підлягають виправленню, давали при діленні на $P(x)$ різні остачі.

Розглянемо процес виправлення помилок на прикладі коду (7, 4).

Як впливає з виразу (1.63) остачі $R(x)$, які з'являються після ділення тих многочленів, у яких спотворені однакові розряди, є однаковими.

Прийнято вважати, що розрядність двійкових еквівалентів остач завжди на одиницю менша розрядності дільника. Визначимо види остач. Остача від ділення на $P(x) = x^3 + x^2 + 1 \rightarrow 1101$ будь-якого дозволеного многочлена степеня b при спотвореному першому розряді

$$R_0(x) = \frac{N_0(x)}{P(x)} = \frac{1}{x^3 + x^2 + 1} = 1 \rightarrow 001$$

(многочлен $N_0(x)$ на $P(x)$ не ділиться, тому значущі символи остачі $R_0(x)$ і $N_0(x)$ співпадають, але степінь $R_0(x)$ зменшується до $r - 1$).

Остача при спотворенні другого розряду

$$R_1(x) = \frac{N_1(x)}{P(x)} = \frac{x}{x^3 + x^2 + 1} = x \rightarrow 010.$$

І далі

$$R_2(x) = \frac{N_2(x)}{P(x)} = \frac{x^2}{x^3 + x^2 + 1} = x^2 \rightarrow 100;$$

$$R_3(x) = \frac{N_3(x)}{P(x)} = \frac{x^3}{x^3 + x^2 + 1} = x^2 + 1 \rightarrow 101;$$

$$R_4(x) = \frac{N_4(x)}{P(x)} = \frac{x^4}{x^3 + x^2 + 1} = x^2 + x + 1 \rightarrow 111;$$

$$R_5(x) = \frac{N_5(x)}{P(x)} = \frac{x^5}{x^3 + x^2 + 1} = x + 1 \rightarrow 011;$$

$$R_6(x) = \frac{N_6(x)}{P(x)} = \frac{x^6}{x^3 + x^2 + 1} = x^2 + x \rightarrow 110.$$

Таким чином, кожному спотвореному розряду відповідає своя остача. Тому на приймальній стороні порівнюють отриману остачу з наявною таблицею, щоб визначити номер спотвореного розряду.

Розглянута процедура декодування циклічних кодів показує, що виправлення поодиноких помилок реалізується дуже просто. Виправлення многократних помилок при циклічному кодуванні хоча й здійснюється дещо простіше, ніж при довільному лінійному коді, але в

загальному випадку залишається достатньо громіздким. Виправлення помилок великої кратності ефективнішими і простішими методами вдається реалізувати при використанні найважливішого класу циклічних кодів – кодів Боуза – Чоудхурі – Хоквінгема (БЧХ).

1.6.7. Вимоги до породжувального многочлену циклічних кодів

На дозволу множини кодових комбінацій при циклічному кодуванні накладаються додаткове обмеження в порівнянні з рештою лінійних кодів: подільність на породжувальний многочлен $P(x)$. Іншими словами, породжувальний многочлен виділяє з усієї групи многочленів $n-1$ -го степеня особливу підгрупу, елементи якої діляться без остачі на $P(x)$. Саме ця ознака спрощує апаратну реалізацію кодуючих і декодуючих пристроїв. Оскільки склад групи визначається видом породжувального многочлена $P(x)$, то його задання – один з головних етапів при циклічному кодуванні. Розглянемо основні вимоги до породжувального многочлена та методику його вибору, орієнтуючись для простоти викладу на коди, що виправляють поодинокі помилки ($d_{\min} = 3$).

Для того, щоб усі многочлени, відповідні кодовим комбінаціям циклічного коду, ділилися без остачі на $P(x)$, достатньо, щоб на нього ділилися без остачі всі многочлени, що відображають кодові комбінації породжувальної матриці. Останні можуть бути одержані циклічним зсувом. Вони будуть остачею від ділення многочленів $B(x)P(x)$ (символічне множення) на біном $x^n + 1$.

Тому можна записати

$$B(x)P(x) = Q(x)(x^n + 1) + R(x),$$

де $Q(x)$ – частка від ділення. Використовуючи правило перенесення доданків, маємо

$$R(x) = Q(x)(x^n + 1) + B(x)P(x).$$

Звідси ясно, що всі многочлени матриці, а тому і всі многочлени коду будуть ділитися без остачі тільки в тому випадку, якщо на $P(x)$ без остачі ділитиметься поліном $x^n + 1$.

Так, щоб $P(x)$ міг сформувати циклічний код, він повинен бути дільником многочлена $x^n + 1$. Це перша і основна вимога до $P(x)$.

Вся решта вимог до породжувального многочлена розроблена з метою поліпшення кодуєчих властивостей циклічного коду. Без виконання першої вимоги циклічний код побудувати взагалі неможливо.

Визначниками помилок на приймальній стороні є остачі від ділення прийнятих комбінацій на породжувальний многочлен. Кожному варіанту спотворення повинна відповідати своя остача. Чим більше різних остач може виникнути від дільника комбінацій, що спотворені завадами, тим більшу кількість різних варіантів спотворень здатний знайти і розпізнати код, тим вищі його коректуючі властивості.

Найбільша можлива кількість остач, що дорівнює $2^m - 1$ (виключаючи нульову), може забезпечити незвідний (простий) многочлен степеня m . Це такий многочлен, який ділиться тільки сам на себе і на одиницю і не ділиться ні на який інший многочлен. Але тут виникає запитання, чи задовольняють незвідні многочлени першій вимозі, тобто чи є вони дільниками бінома $x^n + 1$.

Доведено, що біном $x^n + 1$ ділиться без остачі на простий многочлен степеня m , якщо $n = 2^m - 1$. Більш того біном ділиться в цьому випадку і на інші прості многочлени, степені яких кратні m .

Розглянемо приклад. Розглянемо біном $x^{15} + 1$. Його степінь $n = 15$ може бути записаний як $2^m - 1$ ($m = 4$). Отже, біном $x^{15} + 1$ ділитиметься без остачі на прості многочлени ступенів 4, 2, 1. Існують таблиці розкладання двочленів $x^n + 1$ на незвідні многочлени. Цей многочлен (двочлен) $x^{15} + 1$ може бути записаний у вигляді множення п'яти незвідних многочленів

$$x^{15} + 1 = (x + 1)(x^2 + x + 1)(x^4 + x + 1)(x^4 + x^3 + 1)(x^4 + x^3 + x^2 + x + 1).$$

При використанні, як породжувального, будь-якого з многочленів, що входять в розкладання, буде одержано циклічний код довжиною $n = 15$. Але коректуючі здібності цього коду, які визначаються кількістю різних остач, що виникають при діленні спотворених комбінацій на $P(x)$, будуть неоднакові.

З табл. 1.18 видно, що як породжувальний многочлен для циклічного коду з $n = 15$ потрібно брати многочлени $x^4 + x^3 + 1$ і $x^4 + x + 1$, кількість різних остач яких дорівнює 15. Коди, одержані за допомогою цих многочленів, здатні виправити будь-яку поодинокую помилку.

Проте використовувати для цієї ж мети поліном $x^4 + x^3 + x^2 + x + 1$ не можна, оскільки він створює тільки 5 різних остач. Це пояснюється тим, що цей многочлен входить в розкладання не тільки двочлена $x^{15} + 1$, але і двочлена $x^5 + 1$. Породжувальним многочленом, доцільно вибрати незвідний многочлен $P(x)$, який є дільником двочлена $x^n + 1$ і не входить в розкладання жодного двочлена $x^\lambda + 1$, де $\lambda < n$.

Таблиця 1.18

Кількість різних остач для різних многочленів

Многочлен	Степінь многочлена	Максимально можлива кількість різних остач $2^m - 1$	Реальна кількість різних остач
$x + 1$	1	1	1
$x^2 + x + 1$	2	3	3
$x^4 + x^3 + 1$	4	15	15
$x^4 + x + 1$	4	15	15
$x^4 + x^3 + x^2 + x + 1$	4	15	5

Такі незвідні многочлени називаються примітивними. Таким чином, друга вимога до породжувального многочлена полягає в тому, щоб він був незвідним і примітивним.

При заданому значенні n рівність $n = 2^m - 1$ однозначно визначає значення степенів незвідних (простих) примітивних многочленів, які можуть бути взяті як породжувальні.

Відмітимо, що можна побудувати циклічні коди і для довжин, відмінних від $2^m - 1$. Проте, ці коди будуть малоефективними. Тому, коли говорять про циклічні коди, то завжди мають на увазі, що їх довжина задовольняє рівності $n = 2^m - 1$.

Формування циклічних кодів, як і будь-яких лінійних, починається з складання безнадмірних інформаційних комбінацій – многочленів $B_i(x)$, степінь яких $(k - 1)$ визначається кількістю повідомлень, що належать передачі (кількістю дозволених кодових комбінацій $K = 2^k$). Далі

многочлен $V_i(x)$ множиться на породжувальний многочлен $P(x)$ степеню m . При цьому кількість дозволених многочленів не змінюється, але їх степінь зростає до величини

$$k - 1 + m = n - 1,$$

оскільки при перемножуванні степені поліномів складаються.

Множення безнадмірної кодової комбінації – інформаційного многочлена $V_i(x)$ на $P(x)$ вводить в код таку ж надмірність, як і звичне додавання до інформаційних розрядів $r = m$ контрольних. Таким чином, степінь m породжувального многочлена повинний бути не менше кількості контрольних символів r , а кількість його ненульових членів повинна бути не меншою мінімальної кодової відстані $d_{\min}$. Це третя вимога до $P(x)$.

Для кодів, що виправляють поодинокі помилки, кількість різних ненульових залишків повинна дорівнювати кількості розрядів кодової комбінації n , оскільки в кожному з них можливі спотворення, які потрібно відрізнити одне від одного. Оскільки степінь породжувального многочлена дорівнює кількості контрольних символів, то вибір степеня цього многочлена зводиться до вибору кількості контрольних символів, а це робиться так само як і для будь-якого групового коду, що виправляє поодинокі помилки:

$$r = \log_2(n + 1)$$

або

$$r = \left\lfloor \log_2 \left((k + 1) + \left\lfloor \log_2(k + 1) \right\rfloor \right) \right\rfloor.$$

У табл. 1.19 наведені всі примітивні незвідні многочлени і їх двійкові еквіваленти до сьомого степеня включно.

Таким чином, розглянуті загальні принципи завадостійкого кодування а також найбільш поширені класи циклічних блокових кодів показують, що процес виявлення і виправлення помилок може бути просто реалізований шляхом порівняння ухваленої комбінації зі всіма дозволеними. Незбіг ухваленої комбінації ані з однією з дозволених свідчить про наявність помилок. Виправлення помилок зводиться до знаходження найближчої до ухваленої дозвільної комбінації.

Примітивні незвідні многочлени і їх двійкові еквіваленти

Примітивні незвідні многочлени	Двійкові еквіваленти	Примітивні незвідні многочлени	Двійкові еквіваленти
$x + 1$	11	$x^6 + x^5 + x^2 + x + 1$	1100111
$x^2 + x + 1$	111	$x^6 + x^5 + x^4 + x + 1$	1110011
$x^3 + x + 1$	1011	$x^6 + x^5 + x^3 + x^2 + 1$	1101101
$x^3 + x^2 + 1$	1101	$x^6 + x^4 + x^3 + x + 1$	1011011
$x^4 + x + 1$	10011	$x^7 + x^3 + 1$	10001001
$x^4 + x^3 + 1$	11001	$x^7 + x + 1$	10000011
$x^5 + x^2 + 1$	100101	$x^7 + x^3 + x^2 + x + 1$	10001111
$x^5 + x^3 + 1$	101001	$x^7 + x^6 + x^4 + x^2 + 1$	11010101
$x^5 + x^4 + x^2 + x + 1$	110111	$x^7 + x^4 + x^3 + x^2 + 1$	10011101
$x^5 + x^4 + x^3 + x + 1$	111011	$x^7 + x^6 + x^3 + x + 1$	11001011
$x^5 + x^4 + x^3 + x^2 + 1$	111101	$x^7 + x^6 + x^5 + x^2 + 1$	11100101
$x^5 + x^3 + x^2 + x + 1$	101111	$x^7 + x^6 + x^5 + x + 1$	11100011
$x^6 + x + 1$	1000011	$x^7 + x^5 + x^4 + x^3 + x^2 + x + 1$	10111111
$x^6 + x^5 + 1$	1100001	$x^7 + x^6 + x^3 + x + 1$	11001011

Проте, така процедура навіть при помірних значеннях n вимагає пам'яті з об'ємом, який не завжди може бути реалізований (складність пропорційна 2^n).

Основний напрям опрацювання теорії завадостійкого кодування полягає у пошуку таких кодів, для яких відбір різних кодових комбінацій, виявлення і виправлення помилок здійснювалися б не шляхом перебору, а за допомогою достатньо простих регулярних правил, які можуть бути реалізовані технічно з невеликими (мінімальними) апаратними витратами.

1.6.8. Пристрої кодування та декодування циклічних кодів

Циклічність перестановок при формуванні дозволених кодових комбінацій циклічного коду лежить в основі техніки побудови кодуєчих і декодуєчих пристроїв циклічних кодів. Ця техніка застосовує регістри зсуву у вигляді триггерних ланцюжків з тими або іншими зворотними

зв'язками. Такі регістри зсуву називають також багатотактними лінійними перемикальними схемами або фільтрами Хафмена.

Для побудови лінійних послідовних схем використовуються такі види елементарних пристроїв.

1. Суматор, що має, як правило, два входи й один вихід, при цьому для двійкових кодів підсумовування здійснюється за модулем 2 (рис. 1.22, а).

2. Запам'ятовуючий пристрій, що має один вхід і один вихід і являє собою одну триггерну чарунку (один розряд) регістра зсуву (рис. 1.22, б).

3. Пристрій множення на постійну величину, що має один вхід і один вихід (рис. 1.22, в).

Лінійними перемикальними схемами з кінцевою кількістю станів називаються будь-які схеми, що містять кінцеву кількість суматорів, пристроїв пам'яті та пристроїв множення на константу, з'єднаних будь-яким припустимим способом.

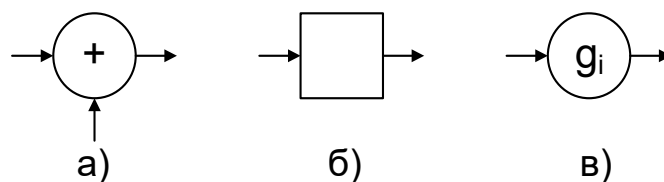


Рис. 1.22. Елементарні пристрої, що використовуються для побудови лінійних перемикальних схем

Пристрої затримки, що включені послідовно, складають регістр зсуву, в чарунках якого вихідний символ збігається з вхідним символом у попередній момент часу. На регістр зсуву також надходять тактові імпульси, за допомогою яких здійснюється просування по розрядах записаної кодової інформації. Як правило, ланцюги подачі тактових імпульсів на схемах не показують.

При формуванні й обробці двійкових циклічних кодів введення в схему помножувача на константу, що дорівнює 1, еквівалентно введенню додаткового з'єднання, а помножувач на константу, що дорівнює 0, відповідає відсутності такого з'єднання.

Передбачається, що на вхід регістра зсуву, що входить до складу лінійної послідовної схеми, кодова комбінація подається послідовно з періодичністю, що дорівнює періоду проходження тактових імпульсів. Аналогічно, послідовно за часом, з'являються кодові символи на виході регістра зсуву. Коли входом або виходом є многочлен, що представляє

при двійковій обробці набір 1 і 0, то на вхідному або вихідному кінці регістра зсуву з'являються тільки коефіцієнти 1 або 0, починаючи з коефіцієнтів вищих порядків. Це обумовлюється тим, що при діленні у дільника спочатку повинні бути оброблені коефіцієнти вищих порядків.

Розглянемо схеми множення та ділення будь-яких многочленів на деякий фіксований многочлен.

Перший варіант лінійної послідовної схеми для множення многочленів зображений на рис. 1.23. Схема використовується для множення будь-якого многочлена на вході $A(x) = a_0 + a_1x + a_2x^2 + \dots + a_kx^k$ на фіксований многочлен $G(x) = g_0 + g_1x + g_2x^2 + \dots + g_rx^r$.

Передбачається, що спочатку всі розряди регістра зсуву містять нулі, а на вхід надходять коефіцієнти многочлена $A(x)$, починаючи з коефіцієнтів вищих порядків (зі старших розрядів), після чого надходить r нулів.

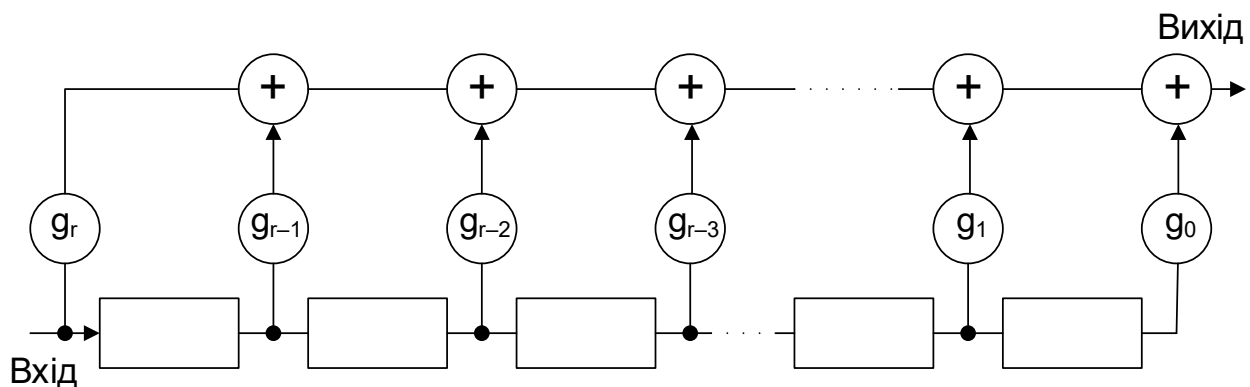


Рис. 1.23. **Схема множення многочленів (перший варіант)**

Добуток многочленів можна записати у такому вигляді:

$$A(x)G(x) = a_0g_0 + (a_0g_1 + a_1g_0)x + \dots + a_kg_rx^{k+r}. \quad (1.64)$$

Коли на вході схеми з'являється перший (старший) коефіцієнт многочлена $A(x)$, то він помножить в першому пристрої множення на g_r і з'явиться на виході вже як результат множення a_kg_r , пройшовши через всі схеми додавання за модулем 2. Крім того, a_k запишеться в перший розряд регістра зсуву, а всі інші розряди будуть містити нулі.

Через одиницю часу, з появою наступного тактового імпульсу, на вході з'явиться a_{k-1} , що перемножить з g_r і додається в першій схемі

додавання за модулем 2 з $a_k g_{r-1}$, сформувавши на виході суму $a_{k-1} g_r + a_k g_{r-1}$, тобто другий коефіцієнт добутку $A(x)G(x)$. Подальші операції здійснюються аналогічно. Після $r + k$ зсувів регістр зсуву повністю обнуляється і на виході з'являється значення $a_0 g_0$, яке дорівнює першому коефіцієнту добутку (1.64).

Розглянемо тепер другий варіант лінійної послідовної схеми для множення многочленів (рис. 1.24).

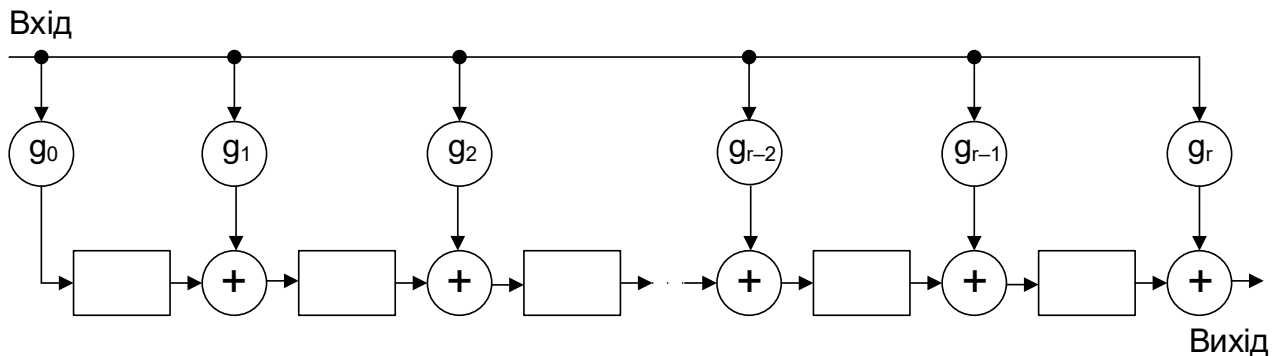


Рис. 1.24. Схема множення многочленів (другий варіант)

У цьому варіанті коефіцієнти добутку формуються безпосередньо в регістрі зсуву. Після того, як перший символ буде поданий на вхід, на виході з'явиться останній коефіцієнт (1.64) $a_k g_r$, а розряди регістра зсуву будуть містити тільки нулі.

Після одного зсуву чарунки регістра містять елементи $a_k g_0, a_k g_1, \dots, a_k g_{r-1}$, а вхід дорівнює a_{k-1} . При цьому вихід регістра дорівнює $a_k g_{r-1} + a_{k-1} g_r$, тобто дорівнює другому коефіцієнту (1.64). Після появи чергового тактового імпульсу на виході схеми з'являється третій коефіцієнт (1.64). Подальші операції здійснюються аналогічно.

Схеми множення можуть мати більше, ніж один вхід, якщо додати до схеми, зображеної на рис. 1.24, другий ланцюжок пристроїв множення, пов'язаних з відповідними схемами додавання за модулем 2. Тоді схема буде реалізовувати процедуру додавання добутків двох пар многочленів:

$$C(x) = A_1(x)G_1(x) + A_2(x)G_2(x),$$

при цьому запам'ятовувальний пристрій у вигляді регістра зсуву буде тільки один.

Розглянемо приклад. Побудувати схеми множення многочленів (за двома розглянутими варіантами) для кодуєчих пристроїв циклічного (7, 4)

коду. Як породжувальний многочлен будемо використовувати многочлен $G(x) = x^3 + x + 1$ (рис. 1.25).

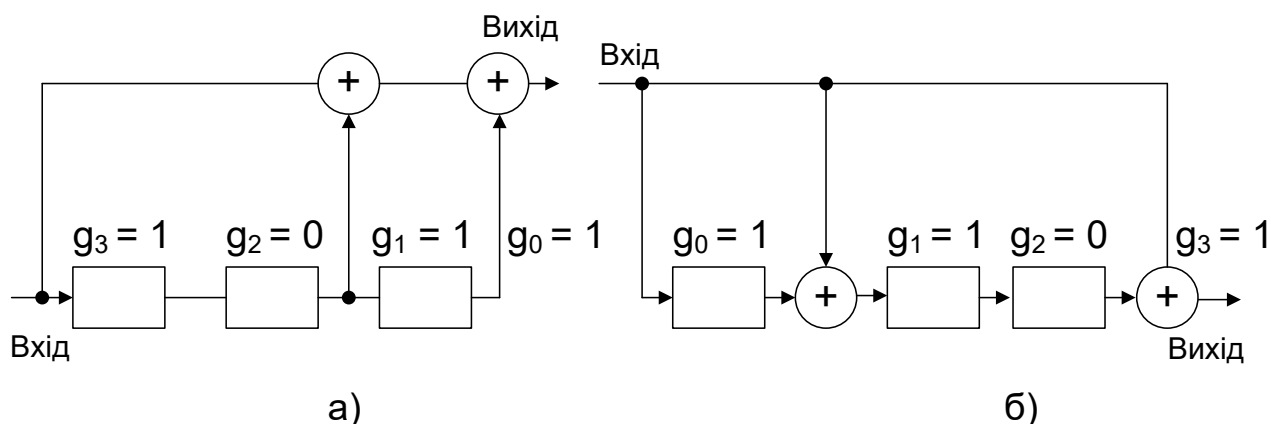


Рис. 1.25. Схеми множення многочленів для кодуючих пристроїв циклічного (7, 4) коду: перший варіант (а), другий варіант (б)

Застосування в кодерах методу множення призводить до формування нероздільних циклічних кодів. Тому для побудови кодеру використовують ділення многочленів на базі лінійних послідовних схем.

Перший варіант схеми ділення многочлена

$$A(x) = a_0 + a_1x + a_2x^2 + \dots + a_kx^k$$

на многочлен

$$G(x) = g_0 + g_1x + g_2x^2 + \dots + g_kx^k$$

зображений на рис. 1.26.

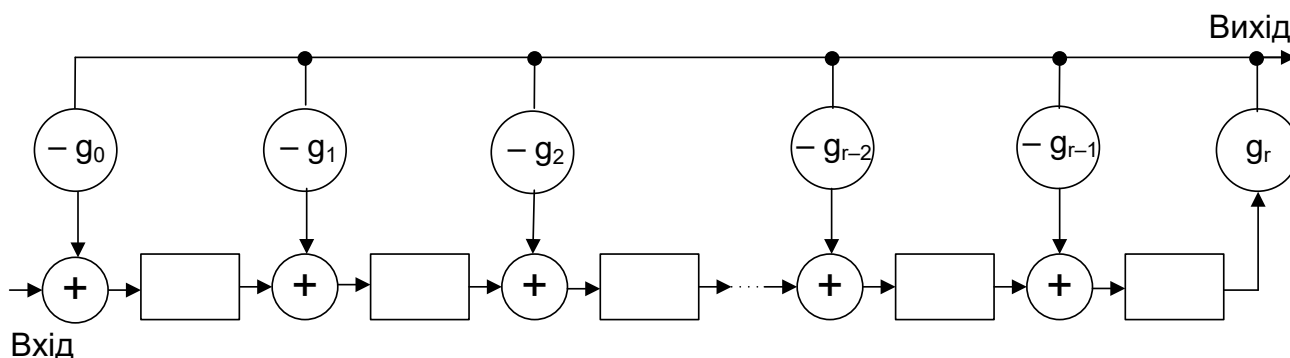


Рис. 1.26. Схема ділення многочленів (перший варіант)

У вихідному стані всі чарунки регістра зсуву містять нулі. Для ділення поліномів регістр охоплений зворотним зв'язком, тобто вихід регістра з'єднується з входом.

Для перших r зсувів, тобто доти, поки перший вхідний символ не досягне кінця регістра зсуву, вихід приймає значення 0. Після цього на виході з'являється перший ненульовий вихідний сигнал, що дорівнює $a_k g_r$ – першому коефіцієнту частки. Для кожного коефіцієнта частки g_j необхідно відняти від діленого многочлен $G(x)$. Це віднімання здійснюється за допомогою зворотного зв'язку. Після k зсувів на виході регістра з'явиться частка від ділення, а остача від ділення буде знаходитися у чарунках регістра зсуву.

При побудові кодуючих пристроїв циклічних кодів, а також генераторів різних кодових послідовностей, зокрема, послідовностей максимальної довжини (М-послідовностей), застосовується в ряді випадків так званий генераторний многочлен $H(x)$. Цей многочлен називають також перевірочним, якщо він утворюється при діленні бінома $1 + x^n$ на породжувальний многочлен $G(x)$

$$H(x) = \frac{1 + x^n}{G(x)}. \quad (1.65)$$

Другий варіант схеми ділення зображений на рис. 1.27.

При використанні цієї схеми, як кодуючого пристрою циклічного коду вихідну кодову комбінацію $A(x)$ записують паралельно в k розрядів регістра зсуву.

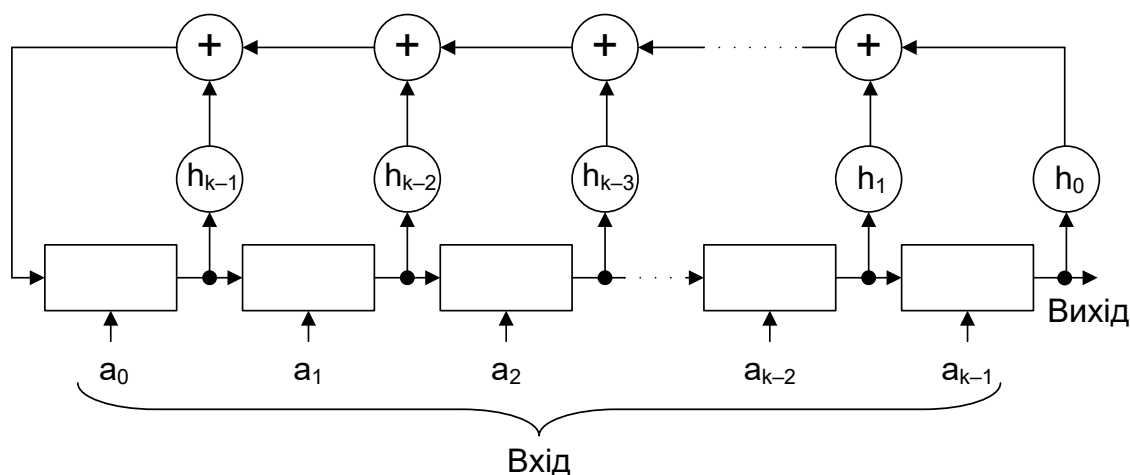


Рис. 1.27. Схема ділення многочленів (другий варіант)

З першим тактом на вихід буде виданий коефіцієнт $b_{n-1} = a_{k-1}$, відбудеться зсув вправо в регістрі і в чарунку пам'яті, що звільниться, буде записане обчислене значення перевірного біта

$$r_{n-k-1} = h_0 a_{k-1} + h_1 a_{k-2} + \dots + h_{k-1} a_0.$$

З другим тактом на виході буде зчитаний коефіцієнт $b_{n-2} = a_{k-2}$, відбудеться зсув і в першу чарунку регістра, що звільнилася, запишеться другий перевіірочний біт $r_{n-k-2} = h_0 a_{k-2} + h_1 a_{k-3} + \dots + h_{k-1} r_{n-k-1}$. Через $n - k$ тактів будуть обчислені всі $n - k$ перевіірочних символів $r_0, r_1, \dots, r_{n-k-1}$ і записані в регістр зсуву. Після k тактів, тобто після виводу на вихід всіх інформаційних символів, стануть виводитися перевіірочні символи в тім же порядку, у якому вони обчислювалися. На виході утворюється блоковий код. Після k тактів процес кодування однієї комбінації $A(x)$ закінчується і регістр зсуву приймає вихідний стан. Для кодування наступної комбінації необхідно стерти $A(x)$, ввести в регістр нову кодову комбінацію $A_j(x)$ і повторити цикл із n тактів.

Розглянемо конкретніше роботу цієї схеми на прикладі використання її як кодуючого пристрою циклічного коду. Для цього:

1) побудуємо схему кодуючого пристрою циклічного $(7, 4)$ коду з породжувальним многочленом $G(x) = 1 + x + x^3$, шляхом обчислення блока перевіірочних символів, використовуючи перевіірочний многочлен $H(x)$;

2) простежимо потактно процес кодування і стан елементів схеми при кодуванні вхідної комбінації $1001 - A(x) = 1 + x^3$.

Побудова схеми кодуючого пристрою визначається перевіірочним многочленом (1.65)

$$H(x) = \frac{1 + x^7}{1 + x + x^3} = 1 + x + x^2 + x^4.$$

Так як $k = 4$, то кількість розрядів регістра зсуву дорівнює чотирьом. З вигляду перевіірочного многочлена визначаємо, що $h_0 = h_1 = h_2 = h_4 = 1$, $h_3 = 0$. Схема кодуючого пристрою для умов прикладу, що розглядається, зображена на рис. 1.28. Стан чарунок регістра зсуву й виходу схеми потактно показано у табл. 1.20.

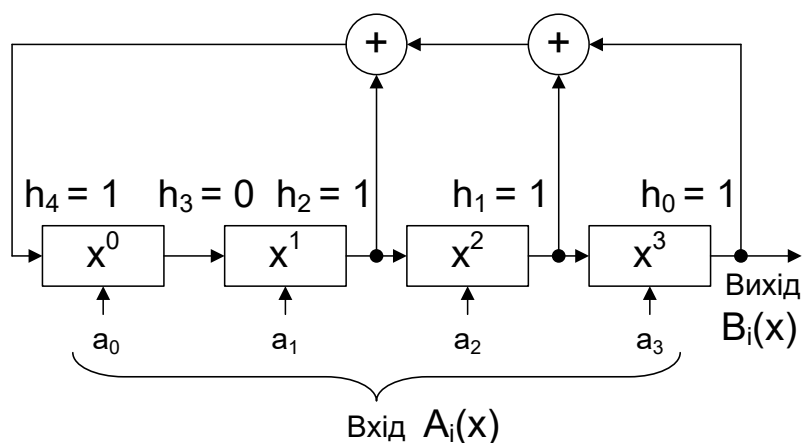


Рис. 1.28. Схема кодера циклічного (7, 4) коду

У вихідному стані в чарунки регістра зсуву записуються інформаційні символи $A(x) = 1 + x^3 \rightarrow 1001$. З огляду на наявність зворотного зв'язку в регістрі, додавання за модулем 2 виходів чарунок x^1 , x^2 і x^3 дасть символ запису в чарунку x^0 .

Після першого зсуву в чарунку x^0 буде записаний символ перевіркої групи r_1 , що при наступних зсувах надійде на вихід регістра. З табл. 1.20 видно, що після $n = 7$ тактів, на виході утвориться комбінація 0111001 (старший розряд попереду).

Таблица 1.20

Стани чарунок регістра зсуву й виходу для схеми кодера циклічного (7, 4) коду

Номер такту	Стан чарунок				Вихід
	x^0	x^1	x^2	x^3	
$A(x)$	1	0	0	1	—
1	1	1	0	0	1
2	1	1	1	0	0
3	0	1	1	1	0
4	1	0	1	1	1
5	0	1	0	1	1
6	0	0	1	0	1
7	1	0	0	1	0

При цьому чарунки регістра зсуву приймають вихідне значення 1001 і при необхідності можливе повторення процедури кодування цієї ж кодової комбінації $A(x)$ шляхом подачі чергових $n = 7$ тактів.

Таким чином, цей спосіб кодування так само, як і перший варіант схеми для ділення многочленів, забезпечує одержання кодових комбінацій роздільного циклічного коду. Крім того, подібна лінійна послідовна схема може бути використана для генерації певної кодової комбінації, зокрема, M -послідовностей.

Розгляд варіантів побудови схем, що виконують операції множення та ділення многочленів, з метою використання в кодах циклічних кодів, дозволяє зробити такі висновки.

1. У кодуючих пристроях циклічних кодів процедура множення многочленів призводить до одержання нероздільних кодів, що ускладнює їх наступне декодування. Тому операція множення рідко використовується в пристроях формування й обробки циклічних кодів.

2. При діленні на породжувальний многочлен $G(x)$, код на виході кодуемого пристрою виходить роздільним і регістр зсуву містить r розрядів. Якщо в більшості випадків використовуються коди, у яких кількість перевірочних символів r істотно менша кількості інформаційних ($r < k$), то регістр зсуву в цьому випадку буде мати меншу кількість розрядів, ніж при діленні на генераторний многочлен.

3. При діленні в кодуєму пристрої вихідної кодової комбінації на генераторний многочлен циклічний код також утворюється роздільним, але в регістрі зсуву потрібно використовувати не r , а k розрядів, яких, як правило, більше. Застосування цього способу більш доцільно в тих випадках, коли та сама кодова комбінація передається по каналу передачі інформації багаторазово, наприклад, у системах зі зворотним зв'язком.

Розглянемо принципи побудови кодуєчих і декодуєчих пристроїв циклічного $(7, 4)$ коду. Приклад побудови кодуемого пристрою циклічного $(7, 4)$ коду за класичною схемою ділення зображений на рис. 1.29. Для кодування необхідні такі операції: множення інформаційного многочлена $A(x)$ на оператор зсуву x^r ; знаходження многочлена – остачі від ділення $x^r A(x)$ на $G(x)$; додавання многочлена – остачі до многочлена $x^r A(x)$. Для виконання зазначених операцій схема на рис. 1.26 повинна бути змінена, а саме: для виконання операції множення варто розмістити суматор, на який підключений вхід, наприкінці регістра зсуву, перед

зворотним зв'язком g_r . Таке підключення входу еквівалентне множенню на x^r , тому що виключається затримка на r тактових імпульсів (ТІ). Для виконання операції додавання залишку $R(x)$ до многочлена $x^r A(x)$ необхідно вихід кодуючого пристрою підключити до одного з входів схеми логічного додавання АБО, а другий вхід схеми АБО з'єднати з входом кодуючого пристрою.

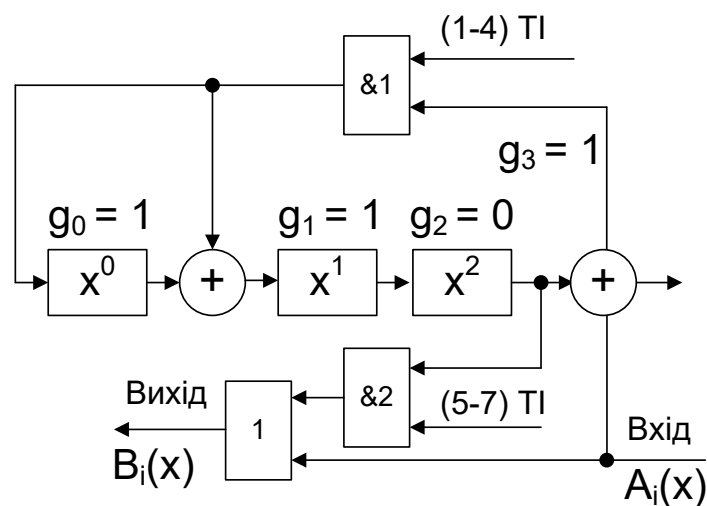


Рис. 1.29. Схема кодера циклічного (7, 4) коду

Розглянемо приклад. Побудувати схему кодуючого пристрою, що забезпечує кодування циклічного (7, 4) коду з породжувальним многочленом $G(x) = 1 + x + x^3$, шляхом визначення перевіркої групи методом ділення многочленів і визначення остачі $R(x)$. Простежити за тактами процес кодування та стани елементів схеми при кодуванні вихідного многочлена $A(x) = 1 + x^3 \rightarrow 1001$. Схема кодера для умов прикладу зображена на рис. 5.11, а стани чарунок регістра зсуву та виходу для схеми кодера потактно наведені у табл. 1.21.

Поряд з відзначеними особливостями побудови кодуючої схеми пристрій доповнений двома ключовими схемами, роль яких виконують схеми логічного множення (схеми І) – &1 і &2, відповідно. Протягом перших $k = 4$ тактів на другий вхід схеми &1 надходять тактові імпульси (ТІ), забезпечуючи проходження символів від вихідного суматора в ланцюг зворотного зв'язку регістра. З 5-го по 7-й такти тактові імпульси на другий вхід схеми &1 не надходять.

Таблиця 1.21

**Стани чарунок регістра зсуву та виходу для схеми кодера
циклічного (7, 4) коду**

Номер такту	Вхід	Стан					
		чарунок			ключів		
		x^0	x^1	x^2	Вихід	&1	&2
0	—	0	0	0	—	Замкнутий	Розімкнутий
1	1	1	1	0	1		
2	0	0	1	1	0		
3	0	1	1	1	0		
4	1	0	1	1	1	Розімкнутий	Замкнутий
5	—	0	0	1	1		
6	—	0	0	0	1		
7	—	0	0	0	0		

При цьому зворотний зв'язок розривається. У цей час тактові імпульси надходять на другий вхід схеми &2, завдяки чому вихід регістра зсуву підключається до виходу всього кодуєчого пристрою, забезпечуючи видачу остачі від ділення кодової комбінації $A(x)$ на породжувальний многочлен $G(x)$ на вихід для підставлення перевіірочних символів до $A(x)$.

З табл. 1.21 видно, що після 4-го такту в регістрі зсуву утвориться залишок 011, тобто $R(x) = x + x^2$, а протягом n тактів на вихід надходить кодова комбінація $0111001 \rightarrow x + x^2 + x^3 + x^6$ (старшим розрядом уперед).

При апаратній реалізації декодерів циклічного коду для визначення синдрому використовують схему, що здійснює процедуру ділення многочлена на многочлен. Схема декодера циклічного (7, 4) коду зображена на рис. 1.30. При побудові декодуючих пристроїв треба додатково включити запам'ятовуючий пристрій (буферний регістр) на k елементів і схему опитування залишку при діленні.

Ця схема складається зі схеми логічного додавання (АБО) на r входів і схеми логічного множення на два входи. Регістр зсуву та зворотні зв'язки повинні відповідати структурі породжувального многочлена $G(x)$, тобто кількість чарунок регістра повинна дорівнювати r , а замкнутий зворотний зв'язок повинен відповідати не нульовим коефіцієнтам многочлена $G(x)$.

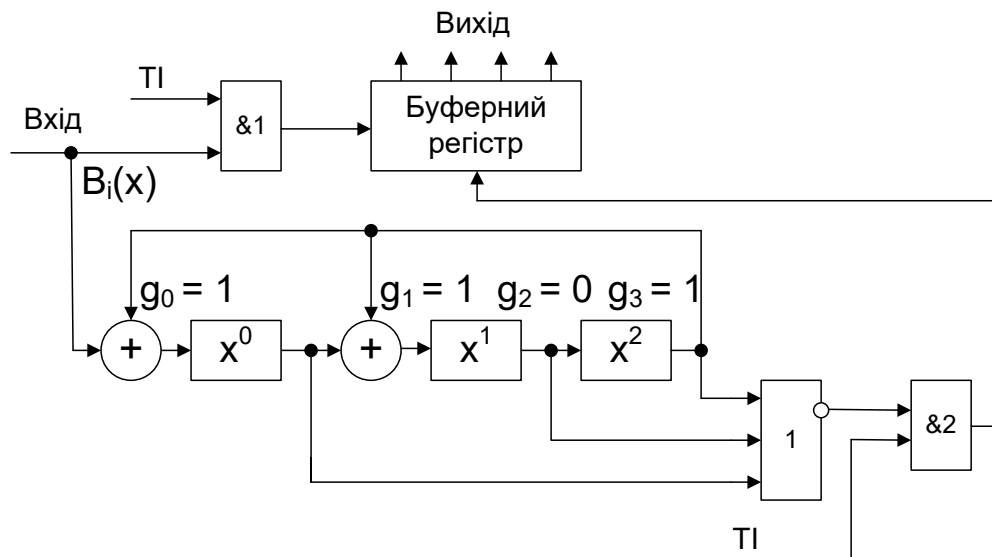


Рис. 1.30. Схема декодера циклічного (7, 4) коду

Розглянемо приклад. Побудувати схему декодуючого пристрою циклічного (7, 4) коду з породжувальним многочленом $G(x) = 1 + x + x^3$ і за тактами імпульсів, що зсувають, проаналізувати його роботу. Схема декодуючого пристрою повинна вирішувати завдання виявлення помилок.

У табл. 1.22 надані стани чарунок регістра при декодуванні вхідної кодової комбінації $B_i(x) = x + x^2 + x^3 + x^6 \rightarrow 0111001$, прийнятої без помилок.

Таблица 1.22

Стани чарунок регістра зсуву при декодуванні вхідної кодової комбінації 0111001

Вхід $B(x)$	Номер такту	Стан чарунок			Вихід регістра
		x^0	x^1	x^2	
—	Початковий стан	0	0	0	—
1	1	1	0	0	0
0	2	0	1	0	0
0	3	0	0	1	0
1	4	0	1	0	1
1	5	1	0	1	0
1	6	0	0	0	1
0	7	0	0	0	0

Декодуючий пристрій працює так. Кодова комбінація $V_i(x)$ старшим розрядом уперед надходить на регістр зсуву для визначення остачі від ділення й у запам'ятовувальний пристрій на k елементів через відкриту схему &1, що через k тактів закривається, тому що припиняється подача тактових імпульсів на один із входів схеми &1. При цьому в запам'ятовуючому пристрої запам'ятовуються k інформаційних символів прийнятої кодової комбінації $V_i(x)$.

У регістр зсуву надходять усі n елементів $V_i(x)$, і після n тактів відбувається опитування стану чарунок регістра шляхом подачі циклового імпульсу на схему &2. Якщо $R(x) \neq 0$, то на виході схеми &2 імпульс не з'явиться і зчитування з запам'ятовуючого пристрою прийнятих інформаційних символів не відбудеться. Якщо $R(x) = 0$, то імпульс, що з'явився на виході &2, дозволяє передати $A(x)$ на вихід і видати чотири інформаційні біти одержувачеві повідомлень.

Для виправлення помилок схема декодуючого пристрою ускладнюється. Інформацію про розряди, у яких відбулася помилка містить синдром, що одержаний у результаті ділення многочленів. Схема декодера з виправленням помилок зображена на рис. 1.31.

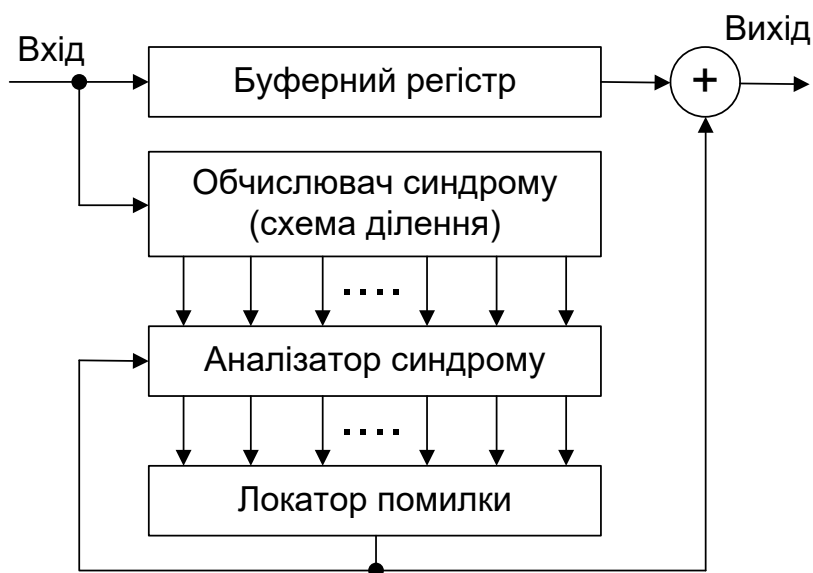


Рис. 1.31. **Схема декодера з виправленням помилок**

Символи кодової комбінації, що містять помилку, послідовно, починаючи зі старшого розряду, вводяться в n -розрядний буферний регістр зсуву й одночасно в схему обчислювача синдрому, де за n тактів

ділення обчислюється остача, що у випадку синхронної неперервної передачі кодових комбінацій відразу ж переписується в аналогічний регістр схеми аналізатора синдрому.

До складу схеми аналізатора синдрому може входити постійний запам'ятовуючий пристрій, у якому записані всі можливі конфігурації синдромів з відповідними їм векторами помилок. Кодові комбінації векторів помилок містять “одиничні” символи на тих позиціях, які в процесі передачі повідомлення по каналу передачі інформації виявилися спотвореними завадами.

Локаатор помилок (визначник місця помилок) являє собою комбінаторно–логічну схему, що видає на вихід одиничні символи в ті моменти часу, коли кожний з помилкових символів прийнятої кодової комбінації займає в буферному регістрі крайню праву чарунку. При наступному тактовому зсуві локаатор помилки формує символ 1, що надходить на суматор по модулю 2, де виправляється спотворений символ.

Одночасно по ланцюгу зворотного зв'язка з виходу локаатора помилки подається одиничний символ на аналізатор синдрому, що в ряді конкретних схемних рішень спрощує його побудову. Складність аналізатора синдрому й локаатора помилки залежить від числа помилок, що виправляються. Ясно, що найпростіші схемні рішення утворюються при обробці кодів, розрахованих на виправлення одиночних помилок.

Як видно з розгляду логіки роботи схеми декодера, найбільш складною його частиною є необхідність запам'ятовування заздалегідь обчислених синдромних многочленів і відповідних їм векторів помилок. Достойнством циклічних кодів є те, що аналізатор синдрому можна значно спростити, скориставшись алгебраїчною структурою коду для відшукування зв'язків між синдромами при великій кількості помилок, що виправляються.

1.6.9. Коду Боуза–Чоудхурі–Хоквінгема, коди Ріда–Соломона

Найбільшого поширення серед циклічних кодів одержали коди БЧХ, у яких як корінь $g(x)$ використовуються $2t$ послідовних степенів довільного елемента $\beta \in GF(q^m)$ (як показано вище – не обов'язково примітивного).

Іншими словами, породжувальний многочлен визначається як:

$$g(x) = \text{Н. З. К. } [f_j(x), f_{j+1}(x), \dots, f_{j+2t-1}(x)], \quad (1.66)$$

де $f_i(x)$ – мінімальний многочлен елемента

$$\beta^i \in GF(q^m), i = j, \dots, j + 2t - 1 \dots$$

Коренями $f_i(x)$ є також всі елементи класу сполучених елементів:

$$\left\{ \beta^i, (\beta^i)^q, (\beta^i)^{q^2}, \dots, (\beta^i)^{q^{s-1}} \right\},$$

де s – найменше ціле число, яке визначається $\beta^{q^s} = \beta$, $s < m$.

Довжина коду дорівнює порядку елемента β , тобто найменшому n , для якого $\beta^n = 1$. Отже, код БЧХ можна задати перевіркою матрицею вигляду:

$$H = \begin{pmatrix} \beta^0 & \beta^j & \beta^{2j} & \dots & \beta^{(n-1)j} \\ \beta^0 & \beta^{j+1} & \beta^{j+2} & \dots & \beta^{(n-1)(j+1)} \\ \dots & \dots & \dots & \dots & \dots \\ \beta^0 & \beta^{j+2t-1} & \beta^{j+2t} & \dots & \beta^{(n-1)(j+2t-1)} \end{pmatrix},$$

де кожен елемент повинен бути замінений на відповідний стовпчик із m елементів над $GF(q)$. Справедлива така теорема.

Теорема 1.8 [5]. Параметри циклічного блокового (n, k, d) -коду БЧХ над $GF(q)$, заданого породжувальним многочленом вигляду (1.66) з коренями $\beta \in GF(q^m)$, задовольняють умову:

$$d \geq 2t + 1, k \geq n - m \times 2t.$$

Теорема 1.8 дає нижню межу розмірності коду БЧХ і його мінімальної кодової відстані. Кодові параметри більшості кодів БЧХ, як правило, лежать вище за цю межу. Так, наприклад, над $GF(2^m)$ виконується рівняння:

$$f_j(x) = f_{2j}(x),$$

отже, для примітивного двійкового коду БЧХ кодові межі з теореми 1.8 можна переписати у вигляді:

$$d \geq 2t + 1, k \geq n - m \times t,$$

породжувальна матриця запишеться у вигляді:

$$H = \begin{pmatrix} \beta^0 & \beta^1 & \beta^2 & \dots & \beta^{n-1} \\ \beta^0 & \beta^3 & \beta^6 & \dots & \beta^{3(n-1)} \\ \dots & \dots & \dots & \dots & \dots \\ \beta^0 & \beta^{2t-1} & \beta^{2t} & \dots & \beta^{(n-1)(2t-1)} \end{pmatrix}, \quad (1.67)$$

де кожен елемент замінюється відповідним m -розрядним двійковим стовпчиком.

Короткі коди БЧХ мають добрі кодові параметри і лежать вище за межу Варшамова–Гілберта. На жаль, зі зростанням довжини коду їх характеристики погіршуються.

Розглянемо приклад побудови коду БЧХ.

Приклад. Розглянемо скінченне поле, побудоване за кільцем многочленів з коефіцієнтами з $GF(2)$, примітивний многочлен $g(x) = x^4 + x + 1$, тоді елементи поля $\{0, 1, x, x^2, x^3, x + 1, x^2 + x, x^3 + x^2, x^2 + 1, x^3 + x, x^2 + x + 1, x^3 + x^2 + x, x^3 + x^2 + x + 1, x^3 + x^2 + 1, x^3 + 1\}$. Знайдемо класи спряжених елементів, визначимо мінімальні многочлени для кожного класу та найменше загальне кратне (НЗК) – породжувальний многочлен коду БЧХ. Знайдемо породжувальну та перевірну матриці.

1. Визначимо класи спряжених елементів через вираз:

$$\beta^{q^r} = \beta,$$

де $q = 2, r = 0, 1, 2, \dots, n - 1$,

$$n = q^m - 1; n = 15:$$

1 клас – по α^0 ,

$$f_{\alpha^0} = \{ \alpha^0 \}$$

2 клас – по

$$\alpha^1 = (\alpha^1)^{2^0} = \alpha^1, (\alpha^1)^{2^1} = \alpha^2, (\alpha^1)^{2^2} = \alpha^4, (\alpha^1)^{2^3} = \alpha^8, (\alpha^1)^{2^4} = \alpha^1;$$

$$f_{\alpha^1} = \{ \alpha^1, \alpha^2, \alpha^4, \alpha^8 \};$$

3 клас – по

$$\alpha^3 = (\alpha^3)^{2^0} = \alpha^3, (\alpha^3)^{2^1} = \alpha^6, (\alpha^3)^{2^2} = \alpha^{12}, (\alpha^3)^{2^3} = \alpha^9; (\alpha^3)^{2^4} = \alpha^3;$$

$$f_{\alpha^3} = \{ \alpha^3, \alpha^6, \alpha^9, \alpha^{12} \};$$

$$4 \text{ клас – по } \alpha^5 = (\alpha^5)^{2^0} = \alpha^5, (\alpha^5)^{2^1} = \alpha^{10}, (\alpha^5)^{2^2} = \alpha^5;$$

$$f_{\alpha^5} = \{ \alpha^5, \alpha^{10} \};$$

5 клас – по

$$\alpha^7 = (\alpha^7)^{2^0} = \alpha^7, (\alpha^7)^{2^1} = \alpha^{14}, (\alpha^7)^{2^2} = \alpha^{13}, (\alpha^7)^{2^3} = \alpha^{11}; (\alpha^7)^{2^4} = \alpha^7;$$

$$f_{\alpha^7} = \{ \alpha^7, \alpha^{11}, \alpha^{13}, \alpha^{14} \}.$$

2. Визначимо мінімальні многочлени для кожного класу:

$$f(\alpha^0) = (x + \alpha^0) = x + 1;$$

$$f(\alpha^1) = (x + \alpha^1) (x + \alpha^2) (x + \alpha^4) (x + \alpha^8) = x^4 + x + 1;$$

$$f(\alpha^3) = (x + \alpha^3) (x + \alpha^6) (x + \alpha^9) (x + \alpha^{12}) = x^4 + x^3 + x^2 + x + 1;$$

$$f(\alpha^5) = (x + \alpha^5) (x + \alpha^{10}) = x^2 + x + 1;$$

$$f(\alpha^7) = (x + \alpha^7) (x + \alpha^{11}) (x + \alpha^{13}) (x + \alpha^{14}) = x^4 + x^3 + 1.$$

3. Припустимо, що $g(x) = \text{НЗК}[f_1(x), f_2(x), f_3(x), f_4(x)]$, тоді породжувальний многочлен коду

$$\begin{aligned} g(x) &= \text{НЗК}[x^4 + x + 1, x^4 + x + 1, x^4 + x^3 + x^2 + x + 1, x^4 + x + 1] = \\ &= (x^4 + x + 1) \times (x^4 + x^3 + x^2 + x + 1) = x^8 + x^7 + x^6 + x^4 + 1 \end{aligned}$$

За породжувальним многочленом можна побудувати необхідні матриці, навіть без розширення поля. Оскільки кодові слова записуються $s(x) = i(x)g(x)$, то в матричному вигляді:

$$G = \begin{bmatrix} 0 & \cdots & 0 & g_{n-k} & g_{n-k-1} & \cdots & g_2 & g_1 & g_0 \\ 0 & \cdots & g_{n-k} & g_{n-k-1} & g_{n-k-2} & \cdots & g_1 & g_0 & 0 \\ 0 & \cdots & g_{n-k-1} & g_{n-k-2} & g_{n-k-3} & \cdots & g_0 & 0 & 0 \\ \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ g_{n-k} & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix}.$$

Перевірні матриця знаходиться через вираз $h(x) \times g(x) = x^n - 1$ і дорівнює:

$$H = \begin{bmatrix} 0 & 0 & 0 & \cdots & & & h_{k-1} & h_k \\ \vdots & 0 & h_0 & h_1 & & & \vdots & \vdots \\ \vdots & h_0 & h_1 & \cdots & h_k & 0 & \cdots & 0 & 0 \\ h_0 & h_1 & h_2 & \cdots & h_k & \cdots & 0 & 0 & 0 \end{bmatrix},$$

де $h(x)$ – перевірючий многочлен циклічного коду.

Приклад. Нехай дано поле $GF(2^4)$, примітивний многочлен $g(x) = x^4 + x + 1$, породжувальний многочлен $g(x) = x^8 + x^7 + x^6 + x^4 + 1$
Тоді породжувальна матриця дорівнює:

$$G = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Для знаходження перевірного многочлена необхідно $h(x) \times g(x) = x^n - 1$ поділити многочлен x^n на породжувальний многочлен

$$\begin{array}{r} x^{15} - 1 \qquad \qquad \qquad x^8 + x^7 + x^6 + x^4 + 1 \\ \hline x^{15} + x^{14} + x^{13} + x^{11} + x^7 \quad x^7 + x^6 + x^4 + 1 \\ \hline x^{14} + x^{13} + x^{11} + x^7 + 1 \\ x^{14} + x^{13} + x^{12} + x^{10} + x^6 \\ \hline x^{12} + x^{11} + x^{10} + x^7 + x^6 + 1 \\ x^{12} + x^{11} + x^{10} + x^8 + x^4 + 1 \end{array}.$$

Отримавши перевірний многочлен $h(x) = x^7 + x^6 + x^4 + 1$, побудуємо матрицю:

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Лінійний код, обумовлений матрицею H , у якій $2^r - 1$ стовпці збігаються з $2^r - 1$ ненульовими двійковими r -вимірними векторами, упорядкованими довільно, називається *кодом Хеммінга*. Для будь-якого r існує двійковий код Хеммінга, який містить r перевірочних позицій.

Блокова довжина такого коду дорівнює $n = 2^r - 1$, а кількість інформаційних позицій визначається рівнянням:

$$k = n - r = 2^r - 1 - r.$$

Код Хеммінга, який виправляє одиничні помилки, являє собою окремий випадок циклічного коду. Хоча коди Хеммінга не є занадто потужними, вони належать до дуже обмеженого класу блокових кодів, які називаються *щільно впакованими* [5; 6; 7].

Уявимо сфери із центрами у всіх кодових словах, які мають той самий (цілочисельний) радіус. Нехай тепер радіус сфер збільшується (залишаючись цілим числом) доти, доки подальше збільшення радіуса буде неможливе без перетинання сфер. Значення цього радіуса дорівнює кількості помилок, які виправляють кодом. Цей радіус називається радіусом сферичного впакування коду. Тепер дозволимо радіусу збільшуватися далі (залишаючись при цьому цілим числом) доти, доки кожна точка простору не виявиться усередині хоча б однієї сфери. Такий радіус називається радіусом покриття коду. *Щільноупакований код* – це код, для якого сфери деякого однакового радіуса навколо кодових слів, не перетинаючись, покривають весь простір.

Важливою характеристикою лінійного блокового коду, відомого в теорії завадостійкого кодування [5; 6], є ваговий спектр коду – розподіл кількості кодових слів за значенням ваги Хеммінга $w(C^i)$. Ваговий спектр коду може бути записаний у вигляді табл. 1.23.

Таблица 1.23

Ваговий спектр коду

$w(C^i)$	0	1	...	$d - 1$	d	$d + 1$	...	n
Кількість кодових слів	1	0	...	0	# d	# $d + 1$	...	# n

Кожен лінійний код містить нульове кодове слово (одне) ваги $w(C^i) = 0$ і не містить інших слів ваги $w(C^i) < d$ (табл. 1.23). Розподіл кількості кодових слів за вагою Хеммінга для $d \leq w(C^i) < n$ для кожного коду різний. Обчислення вагового спектра для $d \leq w(C^i) < n$ є нетривіальним і на сьогодні невирішеним у загальному вигляді завданням [5; 7]. У той же час існують деякі класи лінійних кодів, для яких ваговий спектр обчислений

повністю [5; 7]. Найпростішими щодо цього класом кодів є еквідистантні коди.

Еквідистантним називається код, відстань між двома різними кодовими словами якого та ж сама. Інакше кажучи, ваговий спектр такого коду можна записати у вигляді табл. 1.24.

Найбільш важливий клас еквідистантних кодів описується визначенням 1.1 і теоремою 1.8.

Таблиця 1.24

Ваговий спектр еквідистантного коду

$w(C^i)$	0	1	...	$d-1$	d	$d+1$	...	n
Кількість кодових слів	1	0	...	0	q^k-1	0	...	0

Визначення 1.1 [5; 7]. Реєстровим кодом максимальної довжини називається циклічний код, перевірний многочлен $h(x)$ якого є примітивним многочленом степеня k над полем $GF(q)$. Якщо n – блокова довжина коду, то $h(x)$ ділить $x^n - \xi$, де $\xi \in GF(q)$ і порядок ξ дорівнює $(q^r - 1) / n$.

Теорема 1.9 [5; 7]. Кожне ненульове кодове слово циклічного реєстрового коду максимальної довжини є циклічним зсувом будь-якого іншого ненульового кодового слова. Таким чином, у метриці Хеммінга всі реєстрові коди максимальної довжини є еквідистантними.

Важливою і широко використовуваною підмножиною кодів БЧХ є коди Ріда–Соломона. Це такі коди БЧХ, у яких мультиплікативний порядок алфавіту символів кодового слова ділиться на довжину коду. Таким чином, $t = 1$ і поле символів $GF(q)$ збігається з полем локаторів помилок $GF(q^m)$. Звичайно ми будемо вибирати α примітивним; тоді

$$n = q^m - 1 = q - 1.$$

Мінімальний многочлен над $GF(q)$ елемента β , узятого з того ж поля, дорівнює:

$$f_{\beta}(x) = x - \beta.$$

Оскільки поле символів і поле локаторів помилок збігаються, всі мінімальні многочлени лінійні. У кодах Ріда – Соломона, що виправляє t помилок, звичайно вважається, що $j_n = 1$, і тоді породжувальний многочлен записується у вигляді:

$$g(x) = (x - a)(x - a^2) \dots (x - a^{2^t}).$$

Степінь цього многочлена завжди дорівнює $2t$, звідки випливає, що параметри коду Ріда–Соломона пов'язані співвідношенням:

$$n - k = 2t.$$

У коді Ріда–Соломона можна вибрати також будь-яке інше значення j_0 , причому за допомогою розумного вибору j_0 іноді вдається спростити кодер. Таким чином,

$$g(x) = (x - a^{j_0})(x - a^{j_{0+1}}) \dots (x - a^{l+2^{t-1}}).$$

Як приклад знайдемо $g(x)$ для (15,11)-коду Ріда–Соломона з $t = 2$ над $GF(16)$. Може бути обране будь-яке j_0 ; виберемо $j_0 = 1$. Тоді:

$$\begin{aligned} g(x) &= (x - a)(x - a^2)(x - a^3)(x - a^4) = \\ &= x^4 + (z^3 + z^2 + 1)x^3 + (z^3 + z^2)x^2 + z^3x + (z^2 + z + 1) = x^4 + a^{13}x^3 + \\ &\quad + a^{11}x^2 + a^3x + a^{10}. \end{aligned}$$

Оскільки степінь $g(x)$ дорівнює 4, $n - k = 4$ і $k = 11$. Інформаційний многочлен являє собою послідовність одинадцяти символів з $GF(16)$, що еквівалентно 44 бітам.

Як другий приклад знайдемо $g(x)$ для (7, 3)-коду Ріда–Соломона з $t = 2$ над $GF(8)$. Може бути обране будь-яке j_0 ; виберемо $j_0 = 0$. Тоді:

$$\begin{aligned} g(x) &= (x - a^4)(x - a^5)(x - a^6)(x - a^0) = \\ &= x^4 + (z^2 + 1)x^3 + (z^2 + 1)x^2 + (z + 1)x + z. \end{aligned}$$

(тут використане подання елементів поля $GF(8)$ у вигляді многочленів від z). Інформаційний многочлен являє собою послідовність трьох восьмеричних символів (що еквівалентно дев'яти бітам). Припустимо, що:

$$i(x) = (z^2 + z)x^2 + x + (z + 1).$$

Кодове слово несистематичного коду запишеться у вигляді:

$$\begin{aligned} c(x) &= i(x)g(x) = (a^4x^2 + x + a^3)(x^4 + a^0x^3 + a^6x^2 + a^3x + a) = \\ &= a^4x^6 - ax^5 + a^5x^4 + 0x^3 + 0x^2 + a^5x + a^4. \end{aligned}$$

що являє собою послідовність семи восьмеричних символів.

Коди Ріда–Соломона є оптимальними в сенсі межі Сінглтона.

Теорема 1.10. Код Ріда–Соломона має мінімальну відстань $n - k + 1$ і є кодом з максимальною відстанню.

Доведення. Нехай $d = 2t + 1$ – конструктивна відстань коду. Мінімальна відстань d^* задовольняє нерівність:

$$d^* \geq d - 2t + 1 = n - k + 1,$$

оскільки для кодів Ріда–Соломона $2t = n - k$. Але для будь-якого лінійного коду має місце межа Сінглтона:

$$d^* \leq n - k + 1.$$

Отже,

$$d^* = n - k + 1 \quad \text{та} \quad d^* = d.$$

Доведена теорема стверджує, що при фіксованих n і k не існує коду, в якого мінімальна відстань більша, ніж у коду Ріда–Соломона. Цей факт часто є вагомою підставою для використання коду Ріда–Соломона. Не слід, однак, розуміти це твердження буквально. Часта перевага віддається кодам з такими параметрами (n', k') , за яких не існує коду Ріда–Соломона. У той же час коди Ріда–Соломона завжди виявляються коротшими від усіх інших циклічних кодів над тим же алфавітом.

Коди Ріда–Соломона надзвичайно ефективні для виправлення пакетів помилок, тобто вони виявляються ефективними в каналах з пам'яттю. Також вони добре зарекомендували себе в каналах з більшим набором вхідних символів. Особливістю коду Ріда–Соломона є те, що до коду довжини n можна додати два інформаційних символи, не зменшуючи при цьому мінімальної відстані. Такий розширений код має довжину $n + 2$ і ту ж кількість символів контролю парності, що й вихідний код. З рівняння

$$P_B \approx \frac{1}{n} \sum_{j=t+1}^n \binom{n}{j} p^j (1-p)^{n-j}$$

імовірність появи помилки в декодованому символі, P_B , можна записати через імовірність появи помилки в каналному символі, P_E :

$$P_E \approx \frac{1}{2^m - 1} \sum_{j=t+1}^{2^m-1} j \binom{2^m-1}{j} p^j (1-p)^{2^m-1-j}.$$

Тут t – кількість помилкових бітів у символі, які може виправити код, а символи містять m бітів кожного.

Для побудови коду та перевірної матриці кодів Ріда–Соломона необхідно:

- вибрати примітивний многочлен степеня m і побудувати поле $GF(q^m)$;
- знайти породжувальний многочлен

$$g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^3) \dots (x - \alpha^{2t});$$

Знайти параметри коду РС при $d = n - k + 1$, $n = q^m - 1$, $k = n - r$, $r = 2t = n - k$.

Приклад: $GF(2^3) = \{0, \alpha^0, \alpha^1, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}$;

$$g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^3) \dots (x - \alpha^{2t}) = x^4 + \alpha^2 x^3 + \alpha^5 x^2 + \alpha^5 + \alpha^6;$$

$$n = 7, t = 2, r = 4, k = 3.$$

Одержимо код (7,3,4).

Перевірочна матриця дорівнює:

$$H = \begin{bmatrix} \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 \\ \alpha^1 & \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 \\ \alpha^2 & \alpha^0 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^1 & \alpha^3 \\ \alpha^3 & \alpha^0 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha^1 \\ \alpha^4 & \alpha^0 & \alpha^4 & \alpha^2 & \alpha^5 & \alpha^1 & \alpha^4 \end{bmatrix}.$$

За перевіркою матрицею обчислюється породжувальна матриця:

$$H \times G^T = 0.$$

$$\begin{bmatrix} \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 \\ \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \alpha^0 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^1 & \alpha^3 & \alpha^5 \\ \alpha^0 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha^1 & \alpha^4 \end{bmatrix} \times \begin{bmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \\ g_{41} & g_{42} & g_{43} \\ g_{51} & g_{52} & g_{53} \\ g_{61} & g_{62} & g_{63} \\ g_{71} & g_{72} & g_{73} \end{bmatrix} = \|0\|.$$

Нехай $g_{11} = 1$, $g_{22} = 1$, $g_{33} = 1$, а $g_{12} = g_{13} = g_{21} = g_{23} = g_{31} = g_{32} = 0$, тим самим зведемо породжувальну матрицю G до систематичного вигляду.

Далі перемноживши матриці H , G , одержимо систему рівнянь, розв'язавши її, одержимо значення g_{ij} і побудуємо породжувальну матрицю.

1.6.10. Альтернативні коди, коди Гоппи

Розглянемо циклічні коди для випадку $m = 1$. Тоді поле символів (n, k, d) -коду над $GF(q)$ збігається з полем $GF(q^m)$. Якщо код примітивний, то одержимо код Ріда–Соломона (РС), довжина якого дорівнює

$$n = q^m - 1 = q - 1.$$

Справедлива така теорема.

Теорема 1.11 [5; 6; 7]. Код РС має мінімальну відстань $d = n - k + 1$ і є кодом з максимально досяжною кодовою відстанню (МДВ кодом). Породжувальний многочлен записується у вигляді

$$g(x) = \prod_{i=1}^{j+2t-1} (x - \beta^i),$$

де t – конструктивна здатність коду до виправлення: $t = \lfloor (d-1)/2 \rfloor$, причому $n - k = 2t$.

Вираз $d \leq n - k + 1$ (межа Сінглтона) установлює верхню межу для параметрів лінійного блокового (n, k, d) -коду [5; 7; 8]. Теорема 1.8 стверджує, що при фіксованих n і k не існує коду, в якого мінімальна відстань більша, ніж у коду РС. У той же час коди РС є коротшими за всі інші циклічні коди над тим же алфавітом.

Визначення 1.2. [5; 6; 7]. Нехай

$$X = (X_1, X_2, \dots, X_n)$$

– вектор над $GF(q^m)$, причому всі X_i – різні елементи $GF(q^m)$. Нехай також

$$h = (h_1, h_2, \dots, h_n)$$

– вектор над $GF(q^m)$ з необов'язково різними h_i елементами $GF(q^m)$. Тоді (n, k, d) узагальнений код Ріда–Соломона $URC_k(X, h)$ складається з усіх векторів вигляду:

$$(h_1 \times F(X_1), h_2 \times F(X_2), \dots, h_n \times F(X_n)),$$

де $F(x)$ – будь-який многочлен з коефіцієнтами з $GF(q^m)$, степінь якого не перевищує k .

Код URC є кодом з максимально досяжною кодовою відстанню, тобто $d = r + 1$, $r = n - k$.

Перевірні матриця $URC_k(X, h)$ дорівнює:

$$\begin{aligned}
H &= \begin{pmatrix} Y_1 & Y_2 & \dots & Y_n \\ X_1 \times Y_1 & X_2 \times Y_2 & \dots & X_n \times Y_n \\ X_1^2 \times Y_1 & X_2^2 \times Y_2 & \dots & X_n^2 \times Y_n \\ \dots & \dots & \dots & \dots \\ X_1^{r-1} \times Y_1 & X_2^{r-1} \times Y_2 & \dots & X_n^{r-1} \times Y_n \end{pmatrix} = \\
&= \begin{pmatrix} 1 & 1 & \dots & 1 \\ X_1 & X_2 & \dots & X_n \\ X_1^2 & X_2^2 & \dots & X_n^2 \\ \dots & \dots & \dots & \dots \\ X_1^{r-1} & X_2^{r-1} & \dots & X_n^{r-1} \end{pmatrix} \cdot \begin{pmatrix} Y_1 & 0 & \dots & 0 \\ 0 & Y_2 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & Y_n \end{pmatrix}, \tag{1.68}
\end{aligned}$$

де вектор $Y = (Y_1, Y_2, \dots, Y_n)$ такий, за якого $\forall Y_i \in GF(q^m)$, $Y_i \neq 0$ і $UPC_k^\perp(X, h) = UPC_r(X, Y)$.

Коди УРС дають механізм побудови великого класу альтернантних кодів [6].

Визначення 1.3. [6]. *Альтернантний (n, k, d) -код $A(X, h)$* складається з усіх слів коду $UPC_k^\perp(X, h)$ таких, елементи яких лежать у полі $GF(q)$. Інакше кажучи, $A(X, h)$ дорівнює обмеженню коду $UPC_k(X, h)$ на підполе $GF(q)$. Таким чином, $A(X, h)$ складається з усіх векторів X над $GF(q)$, що задовольняють рівність:

$$H \times X^T = 0,$$

де H – перевірна матриця $UPC_k(X, h)$.

Параметри альтернантного (n, k, d) -коду $A(X, h)$ пов'язані співвідношенням:

$$n - mr \leq k \leq n - r; d \geq r + 1.$$

Породжувальна матриця $A(X, h)$ може бути отримана заміною кожного елемента матриці H коду УРС відповідним вектором-стовпчиком довжини m над $GF(q)$.

Альтернантні коди являють собою великий клас лінійних блокових кодів і узагальнюють (містять як підклас) усі розглянуті вище випадки. Справедливі такі леми.

Лема 1.2. Альтернантні коди узагальнюють (містять) всі циклічні коди.

Доведення. Дійсно, довільний альтернантний (n, k, d) -код $A(X, h)$ над $GF(q)$ складається з усіх слів коду $UPC_k^\perp(X, h)$ над $GF(q^m)$ таких, елементи яких лежать у полі $GF(q)$. За визначенням (n, k, d) -код $UPC_k(X, h)$ складається з усіх векторів вигляду:

$$(h_1 \times F(X_1), h_2 \times F(X_2), \dots, h_n \times F(X_n)),$$

де $F(x)$ – будь-який многочлен з коефіцієнтами з $GF(q^m)$, степінь якого не перевищує k .

Якщо

$$h = (h_1, h_2, \dots, h_n)$$

– одиничний вектор, а вектор

$$X = (X_1, X_2, \dots, X_n)$$

містить всі елементи $GF(q^m)$, то відповідний їм альтернантний код $A(X, h)$ над $GF(q)$ задається многочленом $F(x)$, степінь якого не перевищує k . Якщо $F(x)$ – наведений ненульовий многочлен, то, вірогідно,

$$F(x) = g(x)$$

і маємо циклічний (n, k, d) -код над $GF(q)$.

Лема 1.3. Альтернантні коди узагальнюють (містять) всі коди БЧХ.

Доведення очевидне. Коди БЧХ – підклас циклічних кодів. Якщо многочлен $F(x) = g(x)$ (див. лему 1.1), то маємо циклічний (n, k, d) -код БЧХ над $GF(q)$.

Лема 1.4. Альтернантні коди узагальнюють (містять) всі коди РС.

Доведення. За визначенням (n, k, d) альтернантний код $A(X, h)$ над $GF(q)$ складається з усіх слів коду $UPC_k^\perp(X, h)$. Якщо

$$h = (h_1, h_2, \dots, h_n)$$

– одиничний вектор, вектор

$$X = (X_1, X_2, \dots, X_n)$$

містить всі елементи $GF(q^m)$, а многочлен

$$F(x) = g(x) = \prod_{i=j}^{j+2t-1} (x - \beta^i),$$

тоді за теоремою 1.10 маємо (n, k, d) -код РС над $GF(q^m)$.

Альтернантні коди являють собою потужний клас алгебраїчних лінійних блокових кодів. Найбільш значними результатами їх побудови є коди Гоппи, коди Срівестави та узагальнення Ченя – Чоя [4].

Визначення 1.4. [6]. Альтернантний (n, k, d) -код Гоппи $\Gamma(L, G)$ над $GF(q)$ складається з усіх векторів

$$c = (c_1, c_2, \dots, c_n),$$

таких, за яких

$$R_c(x) \equiv 0 \pmod{G(x)}, \quad (1.69)$$

де

$$R_c(x) = \sum_{i=1}^n \frac{c_i}{x - \alpha_i},$$

$G(x)$ – многочлен з коефіцієнтами з $GF(q^m)$ (многочлен Гоппи),

$$L = (\alpha_1, \alpha_2, \dots, \alpha_n)$$

– підмножина елементів з $GF(q^m)$ таких, за яких

$$G(\alpha_i) \neq 0 \quad \forall \alpha_i \in L.$$

Використовуючи введене визначення й вираз (1.69), можна задати перевірочну матрицю коду Гоппи. Дійсно, многочлен $x - \alpha_i$ у кільці многочленів за модулем $G(x)$ має протилежний многочлен:

$$(x - \alpha_i)^{-1} = -\frac{G(x) - G(\alpha_i)}{x - \alpha_i} G^{-1}(\alpha_i).$$

Отже, вектор $c = (c_1, c_2, \dots, c_n)$ належить коду Гоппи $\Gamma(L, G)$ тоді й тільки тоді, коли

$$\sum_{i=1}^n c_i \frac{G(x) - G(\alpha_i)}{x - \alpha_i} G^{-1}(\alpha_i) = 0. \quad (1.70)$$

Якщо:

$$G(x) = \sum_{i=0}^r g_i x^i,$$

де $g_i \in GF(q^m)$ і $g_r \neq 0$, тоді:

$$\begin{aligned} \frac{G(x) - G(\alpha_i)}{x - \alpha_i} &= g_r(x^{r-1} + x^{r-2}\alpha_i + \dots + \alpha_i^{r-1}) + g_{r-1}(x^{r-2}\alpha_i + \dots + \alpha_i^{r-1}) + \dots + \\ &+ g_2(x + \alpha_i) + g_1. \end{aligned}$$

Прирівнюючи, згідно з (1.70), всі коефіцієнти до нуля, при $x^{r-1}, x^{r-2}, \dots, 1$, одержимо за умови $Hc = 0$:

тільки якщо матриця $\begin{pmatrix} g_r & 0 & \dots & 0 \\ g_{r-1} & g_r & \dots & 0 \\ \dots & \dots & \dots & \dots \\ g_1 & g_2 & \dots & g_r \end{pmatrix}$ – протилежна.

Отже, перевірна матриця

$$H = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \dots & \dots & \dots & \dots \\ \alpha_1^{r-1} & \alpha_2^{r-1} & \dots & \alpha_n^{r-1} \end{pmatrix} \begin{pmatrix} G^{-1}(\alpha_1) & 0 & \dots & 0 \\ 0 & G^{-1}(\alpha_2) & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & G^{-1}(\alpha_n) \end{pmatrix} =$$

$$= \begin{pmatrix} G^{-1}(\alpha_1) & G^{-1}(\alpha_2) & \dots & G^{-1}(\alpha_n) \\ \alpha_1 G^{-1}(\alpha_1) & \alpha_2 G^{-1}(\alpha_2) & \dots & \alpha_n G^{-1}(\alpha_n) \\ \dots & \dots & \dots & \dots \\ \alpha_1^{r-1} G^{-1}(\alpha_1) & \alpha_2^{r-1} G^{-1}(\alpha_2) & \dots & \alpha_n^{r-1} G^{-1}(\alpha_n) \end{pmatrix},$$

$$H = \begin{pmatrix} g_r G^{-1}(\alpha_1) & g_r G^{-1}(\alpha_2) & \dots & g_r G^{-1}(\alpha_n) \\ (g_{r-1} + \alpha_1 g_r) G^{-1}(\alpha_1) & (g_{r-1} + \alpha_2 g_r) G^{-1}(\alpha_2) & \dots & (g_{r-1} + \alpha_n g_r) G^{-1}(\alpha_n) \\ \dots & \dots & \dots & \dots \\ (g_1 + \alpha_1 g_2 + \dots + \alpha_1^{r-1} g_r) G^{-1}(\alpha_1) & (g_1 + \alpha_2 g_2 + \dots + \alpha_2^{r-1} g_r) G^{-1}(\alpha_2) & \dots & (g_1 + \alpha_n g_2 + \dots + \alpha_n^{r-1} g_r) G^{-1}(\alpha_n) \end{pmatrix} =$$

$$= \begin{pmatrix} g_r & 0 & \dots & 0 \\ g_{r-1} & g_r & \dots & 0 \\ \dots & \dots & \dots & \dots \\ g_1 & g_2 & \dots & g_r \end{pmatrix} \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \dots & \dots & \dots & \dots \\ \alpha_1^{r-1} & \alpha_2^{r-1} & \dots & \alpha_n^{r-1} \end{pmatrix} \begin{pmatrix} G^{-1}(\alpha_1) & 0 & \dots & 0 \\ 0 & G^{-1}(\alpha_2) & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & G^{-1}(\alpha_n) \end{pmatrix}$$

також задає визначений вище (n, k, d) -код Гоппи $\Gamma(L, G)$ над $GF(q)$.

Останній вираз при

$$Y = (Y_1, Y_2, \dots, Y_n),$$

$$Y_1 = G^{-1}(\alpha_1), Y_2 = G^{-1}(\alpha_2), \dots, Y_n = G^{-1}(\alpha_n)$$

еквівалентний виразу (1.68). Перевірну матрицю $\Gamma(L, G)$ над $GF(q)$ з елементами з $GF(q)$ можна одержати шляхом подання кожного елемента з $GF(q^m)$ вектором-стовпчиком довжини m символів з $GF(q)$. Справедлива така теорема.

Теорема 1.12. [6]. Параметри (n, k, d) -коду Гоппи $\Gamma(L, G)$ пов'язані співвідношеннями:

$$n = |L|, k \geq n - mr, r = \deg G(x), d \geq r + 1.$$

Якщо $G(x)$ – незвідний многочлен степеня r над $GF(q^m)$ і $L = GF(q^m)$, то існує код Гоппи над $GF(q^m)$ з параметрами, що лежать на межі Варшамова–Гілберта.

Теорема 1.12 гарантує існування добрих альтернантних кодів, побудованих через многочлен Гоппи. У той же час не існує конструктивного способу побудови добрих кодів Гоппи [6].

Іншим великим класом альтернантних кодів є узагальнені коди Срівестави.

Визначення 1.5. [6]. Нехай

$$(\alpha_1, \alpha_2, \dots, \alpha_n)$$

і

$$(\beta_1, \beta_2, \dots, \beta_s)$$

– дві множини різних елементів з $GF(q^m)$ і $(\gamma_1, \gamma_2, \dots, \gamma_n)$ – множина ненульових елементів з $GF(q^m)$. *Узагальнений (n, k, d) -код Срівестави* – це такий альтернантний код над $GF(q)$, перевіркова матриця якого задається виразом (3.7), причому вектор $Y = (Y_1, Y_2, \dots, Y_n)$ такий, за якого

$$Y_i = \frac{\gamma_i}{\prod_{j=1}^s (\alpha_i - \beta_j)^t}, \quad i = 1, \dots, n \dots$$

Справедлива така теорема.

Теорема 1.13. [6]. Параметри узагальненого (n, k, d) -коду Срівестави пов'язані співвідношеннями:

$$k \geq n - mr, \quad r = st, \quad d \geq r + 1.$$

Блокова довжина узагальнених кодів Срівестави не перевищує

$$q^m - s.$$

Якщо

$$n = q^m - s,$$

то за аналогією з кодами БЧХ коди Срівестави називають примітивними.

Іншим окремим випадком альтернантних кодів є узагальнення Ченя–Чоя кодів БЧХ – далі узагальнені коди БЧХ (УБЧХ).

Визначення 1.6 [6]. Нехай n і q взаємно прості, $GF(q^m)$ – найменше розширення поля $GF(q)$, що містить всі корені з одиниці, $P(x)$ і $G(x)$ – взаємно прості з $x^n - 1$ многочлени з коефіцієнтами з $GF(q^m)$, причому

$$\deg(x) \leq n - 1, r = \deg(x) \leq n - 1.$$

Узагальнений код БЧХ $UBCH(P, G)$ – це такий альтернантний код над $GF(q)$, перевірна матриця якого задається виразом (1.68), причому вектор

$$X = (X_1, X_2, \dots, X_n)$$

складається з усіх коренів з одиниці, тобто

$$X = (\alpha^0, \alpha^1, \alpha^2, \dots, \alpha^{n-1})$$

і вектор

$$Y = (Y_1, Y_2, \dots, Y_n),$$

такий, що

$$Y_i = \frac{\alpha^{i-1} \times p_{i-1}}{g_{i-1}}, i = 1, \dots, n \dots$$

Справедлива така теорема.

Теорема 1.14 [6]. Параметри (n, k, d) -коду $UBCH(P, G)$ пов'язані співвідношеннями:

$$k \geq n - mr, r = \deg(x) \leq n - 1, d \geq r + 1.$$

Лема 1.5. Коди $UBCH(P, G)$ узагальнюють (містять) всі коди БЧХ.

Доведення. Очевидно, що якщо:

$$P(x) = x^{j+2t-2} \text{ і } G(x) = x^{j-1},$$

то код $UBCH(P, G)$ є кодом БЧХ із перевіркою матрицею вигляду (3.6).

Таким чином, альтернантні коди являють собою великий клас лінійних блокових кодів, що містить у собі всі циклічні коди, коди БЧХ та їх узагальнення, коди Ріда–Соломона та їх узагальнення, коди Гоппи, коди Срівестави та ін. [6]. У роботах [6; 8] показано, що при відповідному виборі вектора-шаблону

$$Y = (Y_1, Y_2, \dots, Y_n)$$

вдається побудувати добрі блокові коди, які лежать вище за межу Варшамова–Гілберта. Однак для більших n не відоме правило вибору добрих шаблонів i , відповідно, конструктивних алгоритмів побудови добрих блокових кодів. Як показано нижче, алгебраїчні блокові коди, побудовані за алгебраїчними кривими (алгеброгеометричні коди), мають добрі конструктивні характеристики та регулярні алгоритми їх побудови.

1.6.11. Алгеброгеометричні коди

Одним із перспективних напрямів у розвитку алгебраїчної теорії кодів є методи алгеброгеометричного кодування [6]. Доведено, що за великої довжини ці коди лежать вище за межу Варшамова–Гілберта [5 – 8]. У роботах [6; 8] показано, що застосування алгеброгеометричних кодів для завадостійкої передачі даних дискретним каналом із незалежними помилками дозволяє одержати значний енергетичний виграш.

Зафіксуємо скінченне поле $GF(q)$. Нехай X – гладка проективна алгебраїчна крива в проективному просторі P^n над $GF(q)$, $g = g(X)$ – рід кривої, $X(GF(q))$ – множина її точок над скінченним полем, $N = X(GF(q))$ – їх кількість. Нехай C – клас дивізорів на X степеня $\alpha > g - 1$. Тоді C визначає відображення

$$\varphi: X \rightarrow P^{k-1},$$

де

$$k \geq \alpha - g + 1.$$

Набір

$$y_i = \varphi(x_i)$$

задає код.

Кількість точок у перетині $\varphi(X)$ з гіперплощиною дорівнює α , тобто

$$n - d \leq \alpha.$$

Ця конструкція дозволяє будувати коди з параметрами:

$$k + d \geq n - g + 1,$$

довжина n яких менша або дорівнює кількості точок на кривій X .

При $2g < \alpha \leq n$ алгеброгеометричний код має параметри:

$$(n, \alpha - g + 1, d), d \geq n - \alpha.$$

Дуальний до нього код також є алгеброгеометричним і має параметри [6]:

$$(n, n - \alpha + g - 1, d^\perp),$$

$$d^\perp \geq \alpha - 2g + 2.$$

Для оцінки потенційної коригувальної здатності алгеброгеометричного (n, k, d) -коду проведемо дослідження його кодкових характеристик, порівняємо з кодovими межами.

У роботі [6] наведена асимптотична межа алгеброгеометричних кодів:

$$R \leq 1 - \delta - (\sqrt{q} - 1)^{-1}.$$

На рис. 1.32 зображені асимптотичні межі: 1 – межа Сінглтона; 2 – межа Варшамова–Гілберта; 3 – межа алгеброгеометричних кодів.

Залежності, наведені на рис. 1.32, свідчать про те, що алгеброгеометричні коди мають добрі асимптотичні властивості. При зростанні потужності алфавіту кодovих символів асимптотичні властивості таких кодів поліпшуються.

Очевидно, що за великої довжини ці коди лежать вище за межу Варшамова–Гілберта, що свідчить про високі потенційні характеристики цих кодів. Таким чином, алгеброгеометричне кодування переданої інформації керування для підвищення її вірогідності є одним з перспективних напрямів.

Дамо таке визначення алгеброгеометричного коду.

Визначення 1.7. [6]. Нехай X – гладка проєктивна алгебраїчна крива в проєктивному просторі P^n , тобто сукупність розв'язання однорідного незвідного алгебраїчного рівняння степеня $\deg$ з коефіцієнтами з $GF(q)$.

Розглянемо різновиди, що відповідають проєктивним гіперповерхням, заданим у P^n рівняннями $F = 0$, де F – однорідні многочлени степеня $\deg$.

Нехай $I = (I_0, I_1, \dots, I_{k-1})$ – інформаційна послідовність.

Алгеброгеометричний код над $GF(q)$, побудований через відображення кривої X вигляду $\varphi: EC \rightarrow P^{k-1}$, – це лінійний код довжини $n \leq N$, кодovі слова $C(c_0, c_1, \dots, c_{n-1})$ якого задаються рівнянням:

$$\sum_{i=0}^{k-1} I_i F_i(P_i) = c_i,$$

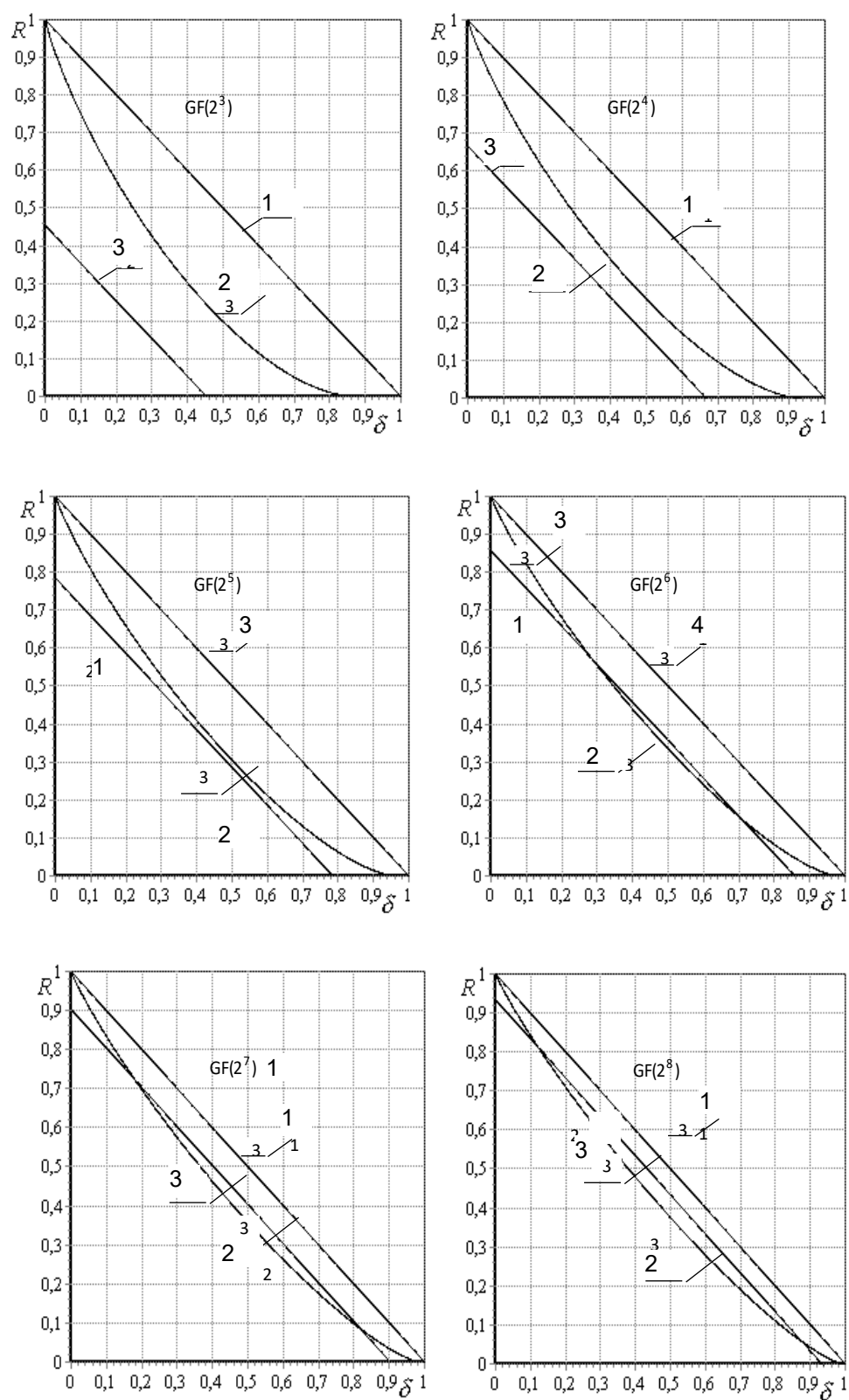


Рис. 1.32. Результати дослідження асимптотичних властивостей алгеброгеометричних кодів

де $P_i(X_i, Y_i, Z_i)$ – проєктивні точки кривої X , тобто (X_i, Y_i, Z_i) – розв'язки однорідного незвідного алгебраїчного рівняння, що задають криву X , $i = \overline{1, n}$; $F_j(P_i)$ – значення генераторних функцій у точках кривої.

Це визначення рівносильне матричному поданню алгеброгеометричного коду:

$$G(i_0, i_1, \dots, i_{k-1})^T = (c_0, c_1, \dots, c_{n-1}),$$

де G – породжувальна матриця розмірності $k \times n$, $k = \alpha - g + 1$, $\alpha = \deg X \times \deg F$:

$$G = \begin{pmatrix} F_0(P_0) & F_0(P_1) & \dots & F_0(P_{n-1}) \\ F_1(P_0) & F_1(P_1) & \dots & F_1(P_{n-1}) \\ \dots & \dots & \dots & \dots \\ F_{k-1}(P_0) & F_{k-1}(P_1) & \dots & F_{k-1}(P_{n-1}) \end{pmatrix} = \|F_j(P_i)\|_{n,k}. \quad (1.71)$$

Визначення 1.8. [6]. Нехай X – гладка проєктивна алгебраїчна крива у P^n , тобто сукупність розв'язків однорідного алгебраїчного незвідного рівняння степеня $\deg$ з коефіцієнтами з $GF(q)$, F – однорідні одночлени степеня $\deg$.

Алгеброгеометричний код над $GF(q)$ побудований через відображення кривої X вигляду $\varphi: EC \rightarrow P^{r-1}$ – це лінійний код, що складається з усіх слів $(c_0, c_1, \dots, c_{n-1})$ довжини $n \leq N$, для яких виконується рівність $d + g - 1$ рівнянь:

$$\sum_{i=0}^{n-1} c_i F_j(P_i) = 0,$$

де $c_i \in GF(q)$, $d \geq \alpha - 2g + 2$, $\alpha = \deg \times \deg$.

Це визначення рівносильне матричному поданню алгеброгеометричного коду:

$$H(c_0, c_1, \dots, c_{n-1})^T = 0,$$

де H – перевірна матриця коду розмірності $r \times n$, $r = n - k = d + g - 2$:

$$H = \begin{pmatrix} F_0(P_0) & F_0(P_1) & \dots & F_0(P_{n-1}) \\ F_1(P_0) & F_1(P_1) & \dots & F_1(P_{n-1}) \\ \dots & \dots & \dots & \dots \\ F_{r-1}(P_0) & F_{r-1}(P_1) & \dots & F_{r-1}(P_{n-1}) \end{pmatrix} = \|F_j(P_i)\|_{n,r}. \quad (1.72)$$

Визначення 1.9. *Еліптичною кривою* (EC) в афінному просторі A^2 над полем $GF(q)$ називається гладка крива, задана рівнянням:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad (1.73)$$

або в P^2 задана однорідним рівнянням:

$$y^2z + a_1xyz + a_3yz^2 = x^3 + a_2x^2z + a_4xz + a_6z^3,$$

де $a_i \in GF(q)$, рід кривої $g = 1$.

Нехай $X(GF(q))$ – множина точок гладкої проєктивної кривої X над скінченним полем $GF(q)$, $N = |X(GF(q))|$ – їх кількість.

Кількість N точок кривої X над $GF(q)$ обмежена зверху виразом Хассе – Вейля [6; 8]:

$$N \leq 2\sqrt{q} \times g + g + 1,$$

де g – рід кривої.

Точні значення верхньої межі кількості точок алгебраїчних кривих над $GF(q)$, $q = 2^m$, $m = 2, 10$ надані в табл. 1.25.

Таблиця 1.25

Оцінка верхньої межі кількості точок алгебраїчних кривих

	GF(4)	GF(8)	GF(16)	GF(32)	GF(64)	GF(128)	GF(256)	GF(512)	GF(1 024)
N_{EC}	9	14	25	44	81	151	289	558	1 089
N_{Hur}	–	21	–	–	105	–	–	197	–
N_{HC}	9	–	65	–	513	–	4 097	–	32 769
N_{FC}	–	27	–	–	945	–	–	32 193	–
N_{SC}	17	65	257	1 025	4 097	16 385	65 537	262 145	1 048 577

Визначення 3.10. [6]. *Кривою Гурвіца* (Hurwitz curve) у проєктивному просторі P^2 над полем $GF(q)$ називається гладка крива, задана однорідним рівнянням:

$$x^m y + y^m z + z^m x = 0, \quad (1.74)$$

де $q = r^3$, r – додатне ціле число, характеристика поля $GF(q)$ ділиться на:

$$m^2 - m + 1.$$

Рід такої кривої визначається виразом вигляду:

$$g = \frac{(q^{\frac{1}{3}} + 1) \times q^{\frac{1}{3}}}{2}.$$

Кількість точок кривої Гурвіца нижча за межу Хассе – Вейля і визначається виразом вигляду:

$$N = q + 2q^{\frac{2}{3}} + 2q^{\frac{1}{3}} + 1.$$

Точні значення верхньої межі кількості точок кривої Гурвіца над $GF(q)$, $q = 2^m$, $m = \overline{2, 10}$ наведені в табл. 1.25.

Визначення 1.11. [6]. *Кривою Ерміта* (Hermit curve) у проєктивному просторі P^2 над полем $GF(q)$ називається гладка крива, задана однорідним рівнянням:

$$x^{\sqrt{q}+1} + y^{\sqrt{q}+1} + z^{\sqrt{q}+1} = 0, \quad (1.75)$$

де $q = r^2$, r – додатне ціле число.

Рід такої кривої визначається виразом вигляду:

$$g = \frac{q - \sqrt{q}}{2}.$$

Кількість точок задовольняє верхню межу Хассе – Вейля і визначається виразом вигляду:

$$N = q\sqrt{q} + 1.$$

Точні значення верхньої межі кількості точок кривої Ерміта над $GF(q)$, $q = 2^m$, $m = \overline{2, 10}$ надані в табл. 1.25.

Визначення 1.12. [6]. *Кривою Ферма* (Fermat curve) у проєктивному просторі P^2 над полем $GF(q)$ називається гладка крива, задана однорідним рівнянням:

$$x^{\frac{2}{3} + \frac{1}{q^3}} + x^{\frac{2}{3} + \frac{1}{q^3} + 1} + x^{\frac{2}{3} + \frac{1}{q^3} + 1} = 0, \quad (1.76)$$

де $q = r^3$, r – додатне ціле число.

Рід такої кривої визначається виразом вигляду:

$$g = \frac{q^{\frac{4}{3}} + 2q - q^{\frac{1}{3}}}{2}.$$

Кількість точок кривої Ферма нижча за межу Хассе–Вейля й визначається виразом вигляду:

$$N = q^{\frac{5}{3}} - q - q^{\frac{2}{3}} + 1.$$

Точні значення верхньої межі кількості точок кривої Ферма над $GF(q)$, $q = 2^m$, $m = \overline{2, 10}$ надані в табл. 1.25.

Визначення 1.13 [6]. *Кривою Сузукі (Susuki curve) у проєктивному просторі P^2 над полем $GF(q)$ називається гладка крива, задана однорідним рівнянням:*

$$x^{q_0}(z^q + zx^{q-1}) + y^{q_0}(y^q + yx^{q-1}) = 0, \quad (1.77)$$

де $q = 2^{2f+1}$, $q_0 = 2^f$, f – додатне ціле число.

Рід такої кривої визначається виразом вигляду

$$g = \left\lfloor \frac{q^2 - q}{2\sqrt{q}} \right\rfloor$$

Кількість точок кривої Сузукі нижча за межу Хассе–Вейля і визначається виразом вигляду $N = q^2 + 1$.

Точні значення верхньої межі кількості точок кривої Сузукі над $GF(q)$, $q = 2^m$, $m = \overline{2, 10}$ надані в табл. 1.25.

Граничні (для випадку $n = N$) кодові співвідношення алгеброгеометричних кодів, побудованих за еліптичними кривими, задаються такими теоремами [6].

Теорема 1.15. *Алгеброгеометричний (n, k, d) -код за еліптичною кривою над $GF(q)$ пов'язаний характеристиками:*

$$k + d \geq n. \quad (1.78)$$

Параметри кодів задовольняють співвідношення:

$$n = 2\sqrt{q} + q + 1, k \geq \alpha, d \geq n - \alpha, d^\perp \geq \alpha, \alpha = 3 \times \deg. \quad (1.79)$$

Теорема 1.16. Алгеброгеометричний (n, k, d) -код за кривою Гурвіца над $GF(q)$ зв'язаний характеристиками:

$$k + d \geq n - \frac{(q^{\frac{1}{3}} + 1) \cdot q^{\frac{1}{3}}}{2} + 1. \quad (1.80)$$

Параметри кодів задовольняють співвідношення:

$$n \leq q + 2q^{\frac{2}{3}} + 2q^{\frac{1}{3}} + 1, k \geq \alpha - \frac{(q^{\frac{1}{3}} + 1) \times q^{\frac{1}{3}}}{2} + 1, d \geq n - \alpha, \quad (1.81)$$

$$d^\perp \geq \alpha - (q^{\frac{1}{3}} + 1) \times q^{\frac{1}{3}} + 2, \alpha = (m + 1) \times \deg F.$$

Теорема 1.17. Алгеброгеометричний (n, k, d) -код за кривою Ерміта над $GF(q)$ зв'язаний характеристиками:

$$k + d \geq n - \frac{q - \sqrt{q}}{2} + 1. \quad (1.82)$$

Параметри кодів задовольняють співвідношення:

$$n \leq q\sqrt{q} + 1, k \geq \alpha - \frac{q - \sqrt{q}}{2} + 1, d \geq n - \alpha, \quad (1.83)$$

$$d^\perp \geq \alpha - q + \sqrt{q} + 2, \alpha = (\sqrt{q} + 1) \deg F.$$

Теорема 1.18. Алгеброгеометричні (n, k, d) -коди за кривою Ферма над $GF(q)$ зв'язані характеристиками:

$$k + d \geq n - \frac{q^{\frac{4}{3}} + 2q - q^{\frac{1}{3}}}{2} + 1. \quad (1.84)$$

Параметри кодів задовольняють співвідношення:

$$n \leq q^{\frac{5}{3}} - q - q^{\frac{2}{3}} + 1, k \geq \alpha - \frac{q^{\frac{4}{3}} + 2q - q^{\frac{1}{3}}}{2} + 1, \quad (1.85)$$

$$d^\perp \geq \alpha - q^{\frac{4}{3}} - 2q + q^{\frac{1}{3}} + 2, d \geq n - \alpha, \alpha = \left(q^{\frac{2}{3}} + q^{\frac{1}{3}} + 1 \right) \cdot \deg F.$$

Теорема 1.19. Алгеброгеометричні (n, k, d) -коди за кривою Сузукі над $GF(q)$ зв'язані характеристиками:

$$k + d \geq n - \left\lfloor \frac{q^2 - q}{2\sqrt{q}} \right\rfloor + 1. \quad (1.86)$$

Параметри кодів задовольняють співвідношення:

$$n \leq q^2 + 1, k \geq \alpha - \left\lfloor \frac{q^2 - q}{2\sqrt{q}} \right\rfloor + 1, \quad (1.87)$$

$$d^\perp \geq \alpha - \left\lfloor \frac{q^2 - q}{\sqrt{q}} \right\rfloor + 2, d \geq n - \alpha, \alpha = (q + 1) \times \deg F.$$

У роботах [6; 8] досліджений зв'язок алгеброгеометричних альтернантних кодів, показано, що узагальнені коди РС та альтернантні коди можуть бути задані через зрізання відповідних алгеброгеометричних кодів. Крім того, використовуючи викладений вище підхід до опису альтернантних кодів, можна сформулювати зрізані алгеброгеометричні коди.

Визначення 1.14 [6]. *Зрізаний алгеброгеометричний (n, k, d) -код* над $GF(q)$ складається зі всіх слів коду алгеброгеометричного (N, K, D) -коду над $GF(q^m)$, таких, що їх компоненти лежать у полі $GF(q)$. Інакше кажучи, зрізаний код дорівнює обмеженню вихідного коду на підполі $GF(q)$, а перевірна матриця зрізаного коду може бути отримана заміною кожного елемента перевірної матриці алгеброгеометричного коду відповідним вектором-стовпчиком довжиною m над $GF(q)$.

Таким чином, кодові слова зрізаного коду складаються зі всіх векторів X над $GF(q)$, що задовольняють рівність:

$$H \times X^T = 0,$$

де H – перевірна матриця алгеброгеометричного (N, K, D) -коду над $GF(q^m)$.

Параметри зрізаного алгеброгеометричного коду задає така теорема.

Теорема 1.20. Параметри зрізаного (n, k, d) -коду зв'язані співвідношенням:

$$n = N, N - m(N - K) \leq k \leq K, d \geq D. \quad (1.88)$$

Породжувальна матриця зрізаного коду може бути отримана заміною кожного елемента перевірної матриці алгеброгеометричного коду відповідним вектором-стовпчиком довжиною m над $GF(q)$.

1.7. Методи та алгоритми кодування алгеброгеометричних кодів

У роботі [6] пропонуються способи систематичного та несистематичного кодування, що дозволяють формувати кодові слова для довільних k символів повідомлення та які допускають практичну реалізацію кодів за довільною кривою в P^2 над $GF(q)$. Складність реалізації знижена на 30% за рахунок побудови генераторної матриці коду відображенням тільки точок кривої вигляду $P(X, Y, 1)$.

Розглянемо криву X в P^2 над $GF(q)$. Розглянемо різновиди, що відповідають проєктивним гіперповерхням, заданим у P^2 рівняннями $F = 0$, де F – однорідні одночлени.

При систематичному кодуванні скористаємося визначенням алгеброгеометричного коду, побудованого через перевірну матрицю.

Систематичний спосіб алгеброгеометричного кодування полягає в такому [6]. Нехай I – множина k інформаційних позицій кодового слова (тобто множина номерів позицій, що входять у заданий інформаційний набір коду) і h – множина $r = n - k$ перевірних позицій.

Об'єднання множин:

$$I \cup h$$

містить всі цілі числа (номери) від 0 до $n - 1$.

На інформаційних позиціях розмістимо k символів повідомлення, а на перевірних – нулі.

Обчислимо суми:

$$S_j = \sum_{i \in I} c_i F_j(P_i), \quad j = \overline{0, r-1}$$

або в матричній формі:

$$\|S_j\|_r = \|F_j(P_i)\|_{k,r} \|c_i\|_k^T. \quad (1.89)$$

Завдання полягає в тому, щоб обчислити й записати на перевірних позиціях такі символи c_i , $i \in h$, які задовольняють рівняння $Hc = 0$. З визначення алгеброгеометричного коду випливає, що значення $r = n - k$ перевірних символів можуть бути знайдені із системи лінійних рівнянь:

$$\sum_{i \in h} c_i F_j(P_i) = -S_j, \quad j = \overline{0, r-1}.$$

У матричному вигляді останній запис еквівалентний виразу:

$$\|F_j(P_i)\|_{r,r} \|c_i\|_r^T = \| -S_j \|_r.$$

Для обчислення значень:

$$r = n - k$$

перевірних символів можна використати методи обернених матриць.

Запишемо в матричній формі:

$$\|c_i\|_r = \|F_j(P_i)\|_{r,r}^{-1} \| -S_j \|_r^T, \quad (1.90)$$

де $\|F_j(P_i)\|_{r,r}^{-1}$ – обернена матриця,

$$\|F_j(P_i)\|_{r,r}^{-1} = \left\| \frac{A[F_j(P_i)]}{\Delta} \right\|_{r,r},$$

$A[F_j(P_i)]$ – алгебраїчне доповнення елемента $F_j(P_i)$,

Δ – визначник матриці $\|F_j(P_i)\|_{r,r}$.

Оскільки розміщення перевірних позицій звичайно відоме і фіксоване, то заздалегідь можна знайти обернену матрицю для системи рівнянь (1.89) і одержати всі перевірні символи множенням вектора

$$(S_0, S_1, \dots, S_{r-1})$$

на матрицю $\|F_j(P_i)\|_{r,r}^{-1}$.

Як інформаційні можуть бути обрані будь-які k позиції кодового слова. Отже, завжди можна вибрати такі множини перевірних та інформаційних позицій, для яких матриця $\|F_j(P_i)\|_{r,r}^{-1}$ найбільш зручна для обчислень.

Таким чином, для реалізації систематичного способу алгеброгеометричного кодування досить зберігати елементи матриць

$$\|F_j(P_i)\|_{k,r}$$

і

$$\|F_j(P_i)\|_{r,r}^{-1}$$

або по черзі обчислювати

$$\|F_j(P_i)\|_{k,r}$$

як значення генераторних функцій у точках кривої.

Алгоритм систематичного кодування задамо у вигляді послідовності таких кроків.

Крок 1. На заздалегідь визначені інформаційні позиції кодового слова помістимо k символів повідомлення.

Крок 2. Обчислимо за виразом (1.89) матрицю-рядок

$$\|S_j\|_r.$$

Крок 3. Обчислимо за виразом (1.90) матрицю-рядок

$$\|c_i\|_r.$$

Крок 4. Помістимо елементи матриці

$$\|c_i\|_r$$

на перевірочні позиції кодового слова.

Несистематичний спосіб кодування полягає в такому [5; 8]. Несистематичне кодування алгоритмічно простіше, але його використання допускає виділення інформаційної частини повідомлення на приймальній стороні з декодованого кодового слова, що потребує додаткових обчислювальних витрат.

Скористаємося визначенням алгеброгеометричного коду через породжувальну матрицю. Кодове слово в цьому випадку може бути сформоване за правилом:

$$c_i = \sum_{j \in I} l_j F_j(P_i), \quad i = \overline{0, n-1}$$

або в матричній формі – як добуток інформаційного вектора-рядка i породжувальної матриці:

$$\|c_j\|_n = G \|l_i\|_k^T = \|F_i(P_j)\|_{n,k} \|l_i\|_k^T. \quad (1.91)$$

Таким чином, для реалізації несистематичного способу кодування необхідно зберігати елементи матриці

$$\|F_i(P_j)\|_{n,k}$$

або по черзі обчислювати їх як значення генераторних функцій у точках кривої.

Алгоритм несистематичного кодування задамо у вигляді послідовного формування символів кодового слова за правилом (1.91).

Розроблені алгоритми алгеброгеометричного кодування дозволяють формувати кодові слова для довільних k символів повідомлення. Алгоритми допускають реалізацію кодів за довільною кривою в P^2 над $GF(q)$. Параметри таких кодів асимптотично лежать вище за межу Варшавова–Гілберта.

Оцінимо алгоритмічну складність розроблених алгоритмів. Якщо заздалегідь сформовані

$$\|F_j(P_i)\|_{k,r} \quad \text{і} \quad \|F_j(P_i)\|_{r,r}^{-1},$$

то при систематичному кодуванні необхідні $k \times r$ операцій додавання й множення для обчислення вектора синдромів й $k \times r$ операцій додавання й множення для обчислення вектора перевірних символів. Усього при відомих і збережених у пам'яті масивах

$$\|F_j(P_i)\|_{k,r} \quad \text{і} \quad \|F_j(P_i)\|_{r,r}^{-1}$$

необхідно виконати

$$r \times (k + r) = r \times n$$

операцій додавання та множення. Формально складність алгоритму систематичного кодування без урахування складності обчислення генераторних функцій має вигляд:

$$O(r \times n).$$

Для несистематичного алгоритму кодування при відомих (заздалегідь сформованих) і збережених у пам'яті елементах:

$$\|F_i(P_j)\|_{n,k}$$

усього необхідно виконати $k \times n$ операцій додавання й множення. Формально складність алгоритму несистематичного кодування без урахування складності обчислення генераторних функцій можна подати як:

$$O(k \times n).$$

Таким чином, для алгеброгеометричних кодів з

$$R = k/n \sim 0,5$$

обидва розроблені алгоритми мають еквівалентну алгоритмічну складність, порівнянну із традиційними процедурами (складність алгоритмів циклічного кодування $O(k \times n)$).

Зберігання елементів масивів

$$\|F_j(P_i)\|_{k,r} \quad \text{і} \quad \|F_i(P_j)\|_{n,k}$$

для довгих кодів може потребувати значних системотехнічних витрат.

У цьому випадку доцільно по черзі обчислювати $F_i(P_j)$. Основною обчислювальною операцією в цьому випадку є знаходження значення генераторної функції F_i у точці алгебраїчної кривої P_j . Для обчислення $F_i(P_j)$ у P^2 потрібні в загальному вигляді три операції піднесення до степеня і дві операції множення. Якщо вважати рівними обчислювальну складність операцій множення і піднесення до степеня, тоді формально складність алгоритму систематичного кодування можна показати:

$$O(5 \times r \times n),$$

складність алгоритму несистематичного кодування – подати у такому вигляді:

$$O(5 \times k \times n).$$

Якщо при побудові генераторної матриці алгеброгеометричного коду використати тільки точки кривої вигляду $P(X, Y, 1)$, то вдається спростити більш ніж на третину необхідні для алгеброгеометричного кодування обчислення. Так, для знаходження $F_i(P_j)$ у P^2 потрібні в цьому випадку дві операції піднесення до степеня і одна операція множення. Формально в цьому випадку складність алгоритму систематичного кодування така:

$$O(3 \times r \times n),$$

а складність алгоритму несистематичного кодування така:

$$O(3 \times k \times n).$$

Розглянемо приклад. Зафіксуємо алгебраїчну криву $x^3 + y^2z + yz^2 = 0$. Над полем $GF(2^2)$ маємо 9 точок кривої:

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
X	0	0	0	1	2	3	1	2	3
Y	1	0	1	2	2	2	3	3	3
Z	0	1	1	1	1	1	1	1	1

На множині точок $\{P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9\}$ побудуємо алгеброгеометричний код.

Зафіксуємо множину генераторних функцій у вигляді одночленів степеня $\deg = 2$: $\{x^2, y^2, z^2, xy, yx, xz\}$. Обчислимо значення генераторних функцій у точках кривої та сформуємо перевірну матрицю коду:

$$H = \|F_j(P_i)\|_{n,r} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 2 & 3 & 1 & 2 & 3 \\ 0 & 1 & 2 & 2 & 2 & 3 & 3 & 3 \\ 0 & 0 & 1 & 3 & 2 & 1 & 3 & 2 \\ 0 & 0 & 2 & 3 & 1 & 3 & 1 & 2 \\ 0 & 1 & 3 & 3 & 3 & 2 & 2 & 2 \end{pmatrix}.$$

Відповідна породжувальна матриця дорівнює:

$$G = \begin{pmatrix} 2 & 2 & 3 & 0 & 1 & 3 & 0 & 1 \\ 3 & 3 & 2 & 1 & 0 & 2 & 1 & 0 \end{pmatrix},$$

і маємо (8, 2, 6)-код. За допомогою перевірної й породжувальної матриць коду легко реалізувати кожний із розглянутих алгоритмів кодування.

Таким чином, у результаті проведених досліджень розроблені методи й практичні алгоритми побудови алгеброгеометричних кодів, які дозволяють будувати кодові конструкції з необхідними конструктивними параметрами в систематичному й несистематичному вигляді, розробляти практичні схеми апаратної й програмної реалізації кодерів, складність яких порівнянна із традиційними (наприклад, кодерами циклічних кодів).

1.8. Методи та алгоритми декодування алгеброгеометричних кодів

1.8.1. Методи декодування недвійкових блокових кодів

Розглянемо лінійний блоковий (n, k, d) -код над полем $GF(q)$. Припустимо, що при передаванні каналом з помилками кодове слово $c = (c_0, c_1, \dots, c_{n-1})$ спотворилося. Вектор помилок позначимо як $e = (e_0, e_1, \dots, e_{n-1})$. Прийняте слово c^* , після передачі каналом з помилками, запишеться у вигляді

$$c^* = c + e = (e_0 + c_0, e_1 + c_1, \dots, e_{n-1} + c_{n-1}).$$

Завдання декодера полягає у відновленні кодового слова c за прийнятим словом c^* . В основі більшості відомих методів декодування недвійкових блокових кодів лежать різні модифікації окремих етапів декодера Пітерсона–Горенштейна–Цирлера [5; 8], який зображений на рис. 1.33. Схема декодування Пітерсона–Горенштейна–Цирлера орієнтована на декодування кодів БЧХ (кодів Ріда–Соломона).

Розглянемо кожний етап зображеної на рис. 1.33 схеми декодування, проаналізуємо можливість їх використання для декодування алгеброгеометричних кодів. На першому етапі обчислюється синдромний вектор $S = (S_0, S_1, \dots, S_{r-1})$. Якщо лінійний блоковий код заданий перевіркою матрицею H :

$$H = \begin{pmatrix} h_{00} & h_{10} & \dots & h_{n-1 0} \\ h_{01} & h_{11} & \dots & h_{n-1 1} \\ \dots & \dots & \dots & \dots \\ h_{0r-1} & h_{1r-1} & \dots & h_{n-1 r-1} \end{pmatrix} = \|h_{ij}\|_{n,r},$$

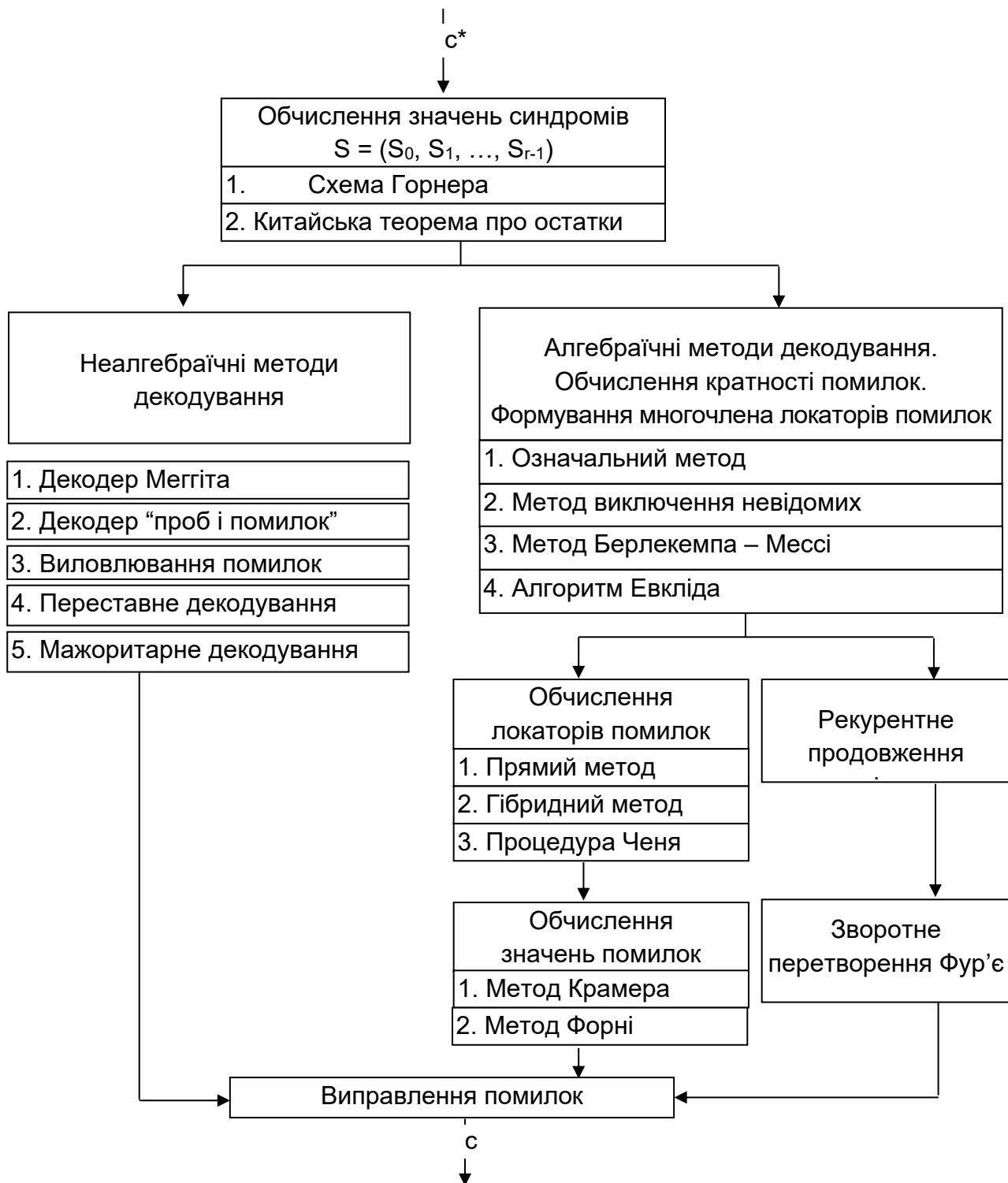


Рис. 1.33. Декодер Пітерсона–Горенштейна–Цирлера

то обчислення синдрому здійснюється за виразом:

$$S_j = \sum_{i=0}^{n-1} c_i^* h_{ij} = \sum_{i=0}^{n-1} e_i h_{ij}, j = 0, \dots, r-1$$

або в матричній формі

$$\|S\|_r = \|h_{ij}\|_{n,r} \|e_i\|_n^T.$$

Стосовно декодування кодів БЧХ елементи синдромної послідовності обчислюються як

$$S_i = \sum_{j=0}^{n-1} e_j X_i^j,$$

де $X \in GF(q^m)$ і однозначно вказує положення помилки, n – дільник $q^m - 1$.

Якщо матричний опис циклічного коду замінити поліноміальним, то обчислення компонентів синдромного вектора еквівалентне обчисленню значення перевірного многочлена в точці X . З цією метою на практиці застосовують швидкі алгоритми обчислення значення многочленів, такі, наприклад, як *схема Горнера*, алгоритм Винограду, заснований на використанні результатів китайської теореми про остачі [5–8].

Обчислення синдрому за схемою Горнера вимагає $r(n - 1)$ операцій додавання і стільки ж операцій множення у полі $GF(q)$.

Якщо довжина коду є складним числом, застосування алгоритму Винограду дозволяє зменшити сумарну кількість операцій.

Теоретичні підходи до алгоритмічного обчислення завдання декодування засновані на алгебраїчних методах розв'язання систем нелінійних рівнянь над кінцевим полем. На другому етапі алгебраїчного декодування за *схемою Пітерсона–Горенштейна–Цирлера* обчислюють кількість допущених помилок. Для вирішення цього завдання використовують штучний прийом, що полягає у введенні в розгляд *многочлена локаторів помилок*, обчислення якого однозначно локалізують (вказують місце розташування виниклих помилок) [5; 6; 8]. У традиційних алгоритмах декодування під *многочленом локаторів помилок* розуміють многочлен степеня $\leq t$ від однієї змінної:

$$a_0 + a_1x + \dots + x^t = 0,$$

де t – кількість помилок, що може виправити код.

Помноживши обидві частини многочлена на e_i і взявши суму за всіма $i = 0 \dots n - 1$ значеннями у точці ($x = X_i$), одержимо рекурентний вираз:

$$a_0 S_i + a_1 S_{i+1} + \dots + S_{i+t} = 0,$$

який задає систему лінійних рівнянь щодо невідомих коефіцієнтів многочлена локаторів помилок. У матричній формі система запишеться у вигляді:

$$\begin{pmatrix} S_0 & S_1 & \dots & S_{t-1} \\ S_1 & S_2 & \dots & S_t \\ \dots & \dots & \dots & \dots \\ S_{t-1} & S_t & \dots & S_{2t-2} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_{t-1} \end{pmatrix} = \begin{pmatrix} -S_t \\ -S_{t+1} \\ \dots \\ -S_{2t-1} \end{pmatrix}.$$

Ранг квадратної матриці в лівій частині задає правдиву кількість допущених помилок. Обчислення системи дасть значення коефіцієнтів многочлена локаторів помилок, що однозначно вказує на місце розташування помилок.

Для обчислення системи використовують алгоритми, які подані на рис. 1.33. Так, означальний метод полягає у знаходженні рангу матриці та обчисленні системи рівнянь за правилом Крамера. З оцінок [6] цей алгоритм вимагає потребує t^3 операцій.

Використання методу виключення невідомих за методом Гаусса і його модифікацій полягає в приведенні матриці до діагонального вигляду й вимагає близько t^3 операцій [6].

Суть *алгоритму Берлекемпа* полягає в синтезі регістра зсуву найменшої довжини з лінійним зворотним зв'язком, що генерує послідовність синдромів (синдромний вектор). Реалізація другого етапу декодування при використанні алгоритму Берлекемпа вимагає близько t^2 операцій [5].

На сьогодні розроблені модифіковані алгоритми Берлекемпа, що дозволяють знизити складність декодування, прискорити швидкодію (алгоритм Берлекемпа – Мессі і його модифікації) [5; 6; 8].

Алгоритм Евкліда полягає в знаходженні найбільшого загального дільника двох многочленів при обчисленні ключового рівняння. Алгоритм Евкліда за кількістю операцій порівнюємо з алгоритмом Берлекемпа.

На наступному етапі алгебраїчного декодування обчислюються локатори, що однозначно вказують на положення помилок. Суть прямого й гібридного методів полягає у приведенні многочлена локаторів до нормального вигляду, що легко реалізувати за невеликої кількості помилок.

У процедурі Ченя не накладається обмеження на кількість помилок. Суть такого алгоритму полягає в підстановці всіх можливих локаторів і виборі тих із них, які перетворюють у нуль многочлен локаторів помилок. Стосовно до декодування кодів БЧХ у многочлен локаторів помилок підставляються всі елементи поля $X \in GF(q^m)$. Кількість операцій при реалізації процедури Ченя становить $2t(q^m - 1)$.

На останньому етапі алгебраїчного декодування обчислюються значення e_i і формується вектор помилок $e = (e_0, e_1, \dots, e_{n-1})$. Підставивши знайдені локатори, одержимо систему лінійних рівнянь щодо невідомих значень e_i . Для обчислення отриманої системи використовують різні алгоритми, а також методи другого етапу декодування (рис. 1.33). При обчисленні ключового рівняння, яке вводиться додатково, використовують також *метод Форні*, що дозволяє знаходити чергове значення e_i у момент відповідного локатора. Складність реалізації алгоритму Форні – близько $2t(t-1)$ [6].

Виправлення помилок здійснюється за правилом:

$$c = c^* - e = (c_0^* - e_0, c_1^* - e_1, \dots, c_{n-1}^* - e_{n-1}).$$

Для деяких низькошвидкісних блокових кодів для формування вектора помилок доцільно використовувати процедури спектрального декодування. Суть спектрального декодування полягає в рекурентному обчисленні компонентів синдрому й формуванні вектора $S = (S_0, S_1, \dots, S_{n-1})$. Це завдання вирішують різними способами: генерацією синдромів через регістри зсуву, методом ланцюгових дробів, безпосередньо через рекурентний вираз. Складність формування повного вектора $S = (S_0, S_1, \dots, S_{n-1})$ через рекурентний вираз становить $(n-2t)(2t-1)$ операцій. Зворотне перетворення Фур'є є окремим випадком інтерполяційної формули Лагранжа [6] і дозволяє сформувати вектор помилок. Складність зворотного перетворення Фур'є становить $2n^2$ операцій. Переходячи до швидких методів перетворення Фур'є, вдається досягти зменшення кількості операцій.

Декодування лінійних блокових кодів істотно спрощується, якщо вдається застосувати ефективні неалгебраїчні методи декодування. Коротко проаналізуємо можливість застосування цих алгоритмів для декодування довгих недвійкових кодів.

Декодер Меггіта є найпростішим алгоритмом декодування циклічних кодів. Цей алгоритм є різновидом табличного декодера, суть якого полягає в зберіганні таблиці синдромів у запам'ятовувальному пристрої. Для декодування недвійкових кодів цей алгоритм не використовується через значний об'єм запам'ятовувального пристрою [1; 2; 3; 8]. Для декодування алгеброгеометричних кодів цей алгоритм не використовується через високі системотехнічні витрати, які необхідні для його реалізації.

Метод проб і помилок заснований на почерговій зміні символів кодового слова й визначенні моменту зменшення вектора помилок. Такий алгоритм є різновидом методу повного перебору і вимагає значних часових витрат [6; 8]. Для декодування алгеброгеометричних кодів цей алгоритм не використовується через високі обчислювальні витрати, які необхідні для його реалізації.

Ідея методу виявлення помилок полягає в перетворенні кодового слова з метою виведення комбінації з t помилок у перевірну частину. Необхідною й достатньою умовою виправлення t помилок методом виявлення є $t < n/k$, що істотно звужує область застосування цього алгоритму [6].

Переставне декодування являє собою послідовне застосування до прийнятого слова методу виявлення помилок і перестановки символів, інваріантного відносно коду. Цей процес триває до моменту виведення всіх помилок у перевіркову частину. Застосування переставного декодера стримується невизначеністю у правилі вибору сукупності перестановок. У той же час для деяких алгеброгеометричних кодів (невеликої довжини) може бути успішно реалізований переставний алгоритм декодування.

Мажоритарний алгоритм декодування має високу швидкість й низьку складність [6; 8]. Суть цього алгоритму полягає в мажоритарному оцінюванні ваги помилки для кожного символу окремо. Алгоритм реалізується тільки для кодів, що допускають повну ортогоналізацію. Якщо алгоритмічний код допускає повну ортогоналізацію, то для його декодування можна використати мажоритарний алгоритм.

Проведений аналіз відомих процедур декодування недвійкових блокових кодів дозволяє зробити висновок про доцільність розробки алгебраїчного декодера алгеброгеометричних кодів. Перспективним напрямом також є розробка переставного і мажоритарного алгоритмів неалгебраїчного декодування алгеброгеометричних кодів.

1.8.2. Декодування алгебраїчних кодів методом Берлекемпа

Розглянемо задачу декодування циклічного (N, K, D) -коду. Позначимо через X_i i -й корінь породжувального многочлена $g(x)$, причому

$$X_i = \alpha^{J_i}$$

для деякого J_i , $X_i \in GF(q^m)$.

Якщо

$$X_0, X_1, \dots, X_{r-1}$$

– всі корені многочлена $g(x)$, тобто

$$g(x) = (x + X_0) \cdot (x + X_1) \cdot \dots \cdot (x + X_{r-1}),$$

справедливе рівняння

$$c(X_i) = c_0 + c_1 X_i + c_2 X_i^2 + \dots + c_{N-1} X_i^{N-1} = 0.$$

Перепишемо останній вираз у вигляді матричного добутку:

$$c(X_i) = (c_0, c_1, c_2, \dots, c_{N-1}) \cdot (1, X_i, X_i^2, \dots, X_i^{N-1})^T = 0.$$

Узагальнивши останнє рівняння для всіх коренів $g(x)$, одержимо:

$$(c_0, c_1, c_2, \dots, c_{N-1}) \times \begin{vmatrix} 1 & X_0 & X_0^2 & \dots & X_0^{N-1} \\ 1 & X_1 & X_1^2 & \dots & X_1^{N-1} \\ 1 & X_2 & X_2^2 & \dots & X_2^{N-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & X_{r-1} & X_{r-1}^2 & \dots & X_{r-1}^{N-1} \end{vmatrix}^T = 0.$$

Отриманий вираз відповідає умові взаємної ортогональності довільного кодового слова:

$$c = (c_0, c_1, c_2, \dots, c_{N-1})$$

і матриці в правій частині добутку. Отже, припустимо, що:

$$\|H\|_{N-K, N} = \begin{vmatrix} 1 & X_0 & X_0^2 & \dots & X_0^{N-1} \\ 1 & X_1 & X_1^2 & \dots & X_1^{N-1} \\ 1 & X_2 & X_2^2 & \dots & X_2^{N-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & X_{r-1} & X_{r-1}^2 & \dots & X_{r-1}^{N-1} \end{vmatrix}.$$

Припустимо тепер, що кодове слово c_i спотворилося при передачі. Нехай кількість помилок на блоці з N символів не перевищує виправної здатності:

$$t = (D - 1)/2$$

циклічного (N, K, D) -коду.

Позначимо многочлен помилок $e(x)$ так, що:

$$e(x) = e_0 + e_1x + e_2x^2 + \dots + e_{N-1}x^{N-1}$$

$e \leq t$ є ненульовими коефіцієнтами.

Нехай

$$c^*(x) = c^*_0 + c^*_1x + c^*_2x^2 + \dots + c^*_{N-1}x^{N-1}$$

– кодове слово з помилками, тобто

$$c^*(x) = c(x) + e(x) = (c_0 + e_0) + (c_1 + e_1)x + (c_2 + e_2)x^2 + \dots + (c_{N-1} + e_{N-1})x^{N-1}.$$

Визначимо синдромний многочлен:

$$s(x) = s_0 + s_1x^1 + \dots + s_{r-1}x^{r-1}$$

як остачу від ділення многочлена $c^*(x)$ на $g(x)$:

$$s(x) = R_{g(x)}[c^*(x)] = R_{g(x)}[c(x) + e(x)] = R_{g(x)}[e(x)].$$

У матричному вигляді коефіцієнти синдромного многочлена, записані у вигляді векторів-рядків

$$s = (s_0, s_1, \dots, s_{r-1}),$$

знайдемо з такого виразу:

$$s = c^* \times H^T = (c + e) \times H^T = c \times H^T + e \times H^T = e \times H^T$$

Таким чином, значення синдрому залежать тільки від конфігурації помилок і не залежать від кодового слова c .

Перепишемо останній вираз у вигляді:

$$(s_0, s_1, \dots, s_{r-1}) = (e_0, e_1, e_2, \dots, e_{N-1}) \times \begin{vmatrix} 1 & X_0 & X_0^2 & \dots & X_0^{N-1} \\ 1 & X_1 & X_1^2 & \dots & X_1^{N-1} \\ 1 & X_2 & X_2^2 & \dots & X_2^{N-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & X_{r-1} & X_{r-1}^2 & \dots & X_{r-1}^{N-1} \end{vmatrix}^T,$$

що еквівалентно такій системі рівнянь:

$$s_0 = e_0 + e_1 X_0 + e_2 X_0^2 + \dots + e_{N-1} X_0^{N-1} = \sum_{i=0}^{N-1} e_i X_0^i,$$

$$s_1 = e_0 + e_1 X_1 + e_2 X_1^2 + \dots + e_{N-1} X_1^{N-1} = \sum_{i=0}^{N-1} e_i X_1^i,$$

...

$$s_{r-1} = e_0 + e_1 X_{r-1} + e_2 X_{r-1}^2 + \dots + e_{N-1} X_{r-1}^{N-1} = \sum_{i=0}^{N-1} e_i X_{r-1}^i.$$

Задача декодування блоку даних з N символів полягає в знаходженні всіх e_i , $i = 0, \dots, N - 1$ за відомими елементами синдромної послідовності

$$(s_0, s_1, \dots, s_{r-1}) \dots$$

Система рівнянь нелінійна, прямі методи її обчислення невідомі [5]. Для знаходження вектора помилок

$$(e_0, e_1, e_2, \dots, e_{N-1})$$

скористаємося штучним прийомом [5; 6]. Уведемо многочлен локаторів помилок $\Lambda(x)$, коренями якого є ненульові елементи вектора помилок), тобто

$$\Lambda(x) = \prod_j (x + X_j),$$

де j – індекс ненульових елементів вектора помилок;

X_j – локатор помилки, що відбулася в j -му символі кодового слова.

Розкриємо дужки у виразі та одержимо:

$$\Lambda(x) = x^w + \lambda_{w-1} x^{w-1} + \dots + \lambda_1 x + \lambda_0,$$

де степінь w многочлена $\Lambda(x)$ задає кількість помилок, що відбулися, на блоці з N символів, тобто кількість ненульових елементів вектора помилок:

$$(e_0, e_1, e_2, \dots, e_{N-1}).$$

Набір

$$(\lambda_0, \lambda_1, \dots, \lambda_{w-1})$$

коефіцієнтів многочлена однозначно задає його корені, які однозначно вказують (локалізують) розташування допущених помилок, що відбулися.

Помножимо многочлен на e_i та обчислимо його значення в X_j , одержимо:

$$e_i X_j^{w+i} + e_i \lambda_{w-1} X_j^{w+i-1} + \dots + e_i \lambda_1 X_j^{i+1} + e_i \lambda_0 X_j^i = 0,$$

де $X_j \in GF(q^m)$, тобто

$$X_j = \alpha^{Jj}$$

для деякого J .

Отже,

$$X_j^{a+b} = \alpha^{a+b+Jj} = X_{j+a}^b,$$

таким чином, справедливий вираз:

$$e_i X_{j+w}^i + e_i \lambda_{w-1} X_{j+w-1}^i + \dots + e_i \lambda_1 X_{j+1}^i + e_i \lambda_0 X_j^i = 0.$$

Останнє рівняння виконується для будь-якого j і при кожному i . Підсумуємо за всіма $i = 0 \dots N-1$, одержимо:

$$\sum_{i=0}^{N-1} (e_i X_{j+w}^i + e_i \lambda_{w-1} X_{j+w-1}^i + \dots + e_i \lambda_1 X_{j+1}^i + e_i \lambda_0 X_j^i) = 0.$$

Змінимо порядок підсумовування, винесемо коефіцієнти многочлена локаторів помилок за знак додавання, одержимо:

$$\sum_{i=0}^{N-1} e_i X_{j+w}^i + \lambda_{w-1} \times \sum_{i=0}^{N-1} e_i X_{j+w-1}^i + \dots + \lambda_1 \times \sum_{i=0}^{N-1} e_i X_{j+1}^i + \lambda_0 \times \sum_{i=0}^{N-1} e_i X_j^i = 0$$

Значення кожного доданка в останньому виразі відповідає добутку коефіцієнтів многочлена локаторів помилок і відповідних синдромів:

$$s_{j+w} + \lambda_{w-1} \times s_{j+w-1} + \dots + \lambda_1 \times s_{j+1} + \lambda_0 \times s_j = 0.$$

Перепишемо вираз для кожного $j = 0 \dots w$, одержимо систему лінійних рівнянь:

$$\begin{aligned} s_w + \lambda_{w-1} \times s_{w-1} + \dots + \lambda_1 \times s_1 + \lambda_0 \times s_0 &= 0, \\ s_{w+1} + \lambda_{w-1} \times s_w + \dots + \lambda_1 \times s_2 + \lambda_0 \times s_1 &= 0, \\ &\dots \\ s_{2 \cdot w} + \lambda_{w-1} \times s_{2 \cdot w-1} + \dots + \lambda_1 \times s_{w+1} + \lambda_0 \times s_w &= 0. \end{aligned}$$

Система з w лінійних рівнянь з w невідомими розв'язна, складність її обчислення зростає поліноміально від кількості невідомих [5; 6]. Так, наприклад, для обчислення системи методом Гаусса необхідно виконати t^2 додавань і множень над елементами $GF(q^m)$, або, формально, складність алгоритму становить $O(t^2)$.

Обчислення системи рівнянь дає значення коефіцієнтів многочлена локаторів помилок. Коренями многочлена є локатори – елементи $GF(q^m)$, які однозначно вказують на розташування помилок. Отже, для локалізації помилок необхідно знайти корені.

Найбільш проста процедура пошуку коренів многочлена локаторів помилок полягає в підстановці всіх елементів поля $GF(q^m)$ і виборі тих елементів, які перетворюють у нуль многочлен. У літературі такий прийом одержав назву процедури Ченя [5; 6].

Після локалізації помилок – знаходження локаторів помилок X_j – необхідно обчислити значення помилок в j -му символі, тобто обчислити вектор помилок:

$$(e_0, e_1, e_2, \dots, e_{N-1})$$

і відновити кодове слово:

$$c = c^* - e.$$

Для знаходження значень помилок підставимо значення знайдених локаторів X_j і невідомі значення e_j у систему рівнянь. Інші e_i при $i \neq j$ дорівнюють нулю. Отже, система рівнянь запишеться у вигляді:

$$s_0 = \sum_{i \in J} e_i X_0^i,$$

$$s_1 = \sum_{i \in J} e_i X_1^i,$$

...

$$s_{r-1} = \sum_{i \in J} e_i X_{r-1}^i,$$

де J – множина індексів ненульових елементів вектора помилок, тобто набір номерів локаторів помилок, причому $|J| = w \leq t$.

Система з r лінійних рівнянь містить:

$$|J| = w \leq t$$

невідомих значень помилок e_i , причому $t < r$. Отже, система розв'язна, її обчислення дає невідомі ненульові значення помилок вектора:

$$(e_0, e_1, e_2, \dots, e_{N-1}).$$

Для відновлення кодового слова довжини N кодових символів досить зняти дію знайденого вектора помилок:

$$c = c^* - e.$$

Приклад. Розглянемо код Ріда–Соломона $(7, 3, 5)$ з перевіркою та породжувальною матрицями:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 3 & 5 & 7 & 2 & 4 & 6 \\ 1 & 4 & 7 & 3 & 6 & 2 & 5 \end{bmatrix} \quad G = \begin{bmatrix} 1 & 0 & 0 & 7 & 6 & 6 & 3 \\ 0 & 1 & 0 & 2 & 3 & 5 & 1 \\ 0 & 0 & 1 & 7 & 7 & 4 & 2 \end{bmatrix}.$$

Припустимо, що необхідно передати інформаційну послідовку $i = 102$. Нехай під час передачі спотворився вектор помилок $e = 1003000$

1. Розрахуємо кодове слово за виразом:

$$c = i \times G.$$

2. Знайдемо отримане з каналу кодове слово за виразом:

$$c = [1 \ 0 \ 2] \times \begin{bmatrix} 1 & 0 & 0 & 7 & 6 & 6 & 3 \\ 0 & 1 & 0 & 2 & 3 & 5 & 1 \\ 0 & 0 & 1 & 7 & 7 & 4 & 2 \end{bmatrix} = 1023510.$$

$$c^* = c + e$$

$$c^* = 1023510 \oplus 1003000 = 0020510.$$

3. Синдромний вектор дорівнює:

$$S = c^* \times H^T;$$

$$S_0 = 2 + 5 + 1 = 7;$$

$$S_1 = 2 \times 3 + 5 \times 5 + 6 = 5;$$

$$S_2 = 2 \times 5 + 5 \times 2 + 4 = 4;$$

$$S_3 = 2 \times 7 + 5 \times 6 + 2 = 6.$$

Таким чином, синдромний вектор $S = (7546)$.

4. Многочлен локаторів помилок у загальному вигляді такий:

$$\Lambda(x) = x^2 + a_1x + a_0 = 0.$$

У матричному вигляді система лінійних рівнянь запишеться так:

$$\begin{bmatrix} S_0 & S_1 \\ S_1 & S_2 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} S_2 \\ S_3 \end{bmatrix} \quad \begin{bmatrix} 7 & 5 \\ 5 & 4 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \end{bmatrix}.$$

Підставивши значення елементів вектора синдромів, одержимо:

$$\begin{bmatrix} 7 & 5 & 4 \\ 5 & 4 & 6 \end{bmatrix} = \begin{bmatrix} 7 & 5 & 4 \\ 0 & 6 & 7 \end{bmatrix} = \begin{bmatrix} 7 & 0 & 3 \\ 0 & 6 & 7 \end{bmatrix}$$

$$7a_0 = 3; \quad 6a_1 = 7.$$

Розв'язання отриманої системи лінійних рівнянь:

$$a_0 = 4; \quad a_1 = 2.$$

Перевіримо отримані значення коефіцієнтів a_i :

$$7a_0 + 5a_1 = 4; \quad 7 \times 4 + 5 \times 2 = 4;$$

$$5a_0 + 4a_1 = 6; \quad 5 \times 4 + 4 \times 2 = 6.$$

Шуканий многочлен локаторів помилок набуває вигляду:

$$\Lambda(x) = x^2 + 2x + 4 = 0.$$

5. Скористаємося процедурою Ченя, підставивши всі x у многочлен локаторів помилок:

$$x = 1, \quad \Lambda(1) = 1^2 + 2 \times 1 + 4 = 1 + 2 + 4 = 0 \text{ — помилка;}$$

$$x = 2, \quad \Lambda(2) = 2^2 + 2 \times 2 + 4 = 3 + 3 + 4 = 4 \neq 0 \text{ — помилки немає;}$$

$$x = 3, \quad \Lambda(3) = 3^2 + 2 \times 3 + 4 = 5 + 4 + 4 = 5 \neq 0 \text{ — помилки немає;}$$

$$x = 4, \quad \Lambda(4) = 4^2 + 2 \times 4 + 4 = 7 + 5 + 4 = 0 \text{ — помилка;}$$

$$x = 5, \quad \Lambda(5) = 5^2 + 2 \times 5 + 4 = 2 + 6 + 4 = 5 \neq 0 \text{ — помилки немає;}$$

$$x = 6, \quad \Lambda(6) = 6^2 + 2 \times 6 + 4 = 4 + 7 + 4 = 7 \neq 0 \text{ — помилки немає;}$$

$$x = 7, \quad \Lambda(7) = 7^2 + 2 \times 7 + 4 = 6 + 1 + 4 = 7 \neq 0 \text{ — помилки немає.}$$

Таким чином локалізуємо помилки, вказуючи на те, де вони розташовані:

$$e = (e_1 0 0 e_4 0 0 0).$$

6. Розрахуємо значення символів вектора помилок e_1, e_4 за виразом:

$$e_i \times H_i^T = S_i$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 3 & 5 & 7 & 2 & 4 & 6 \\ 1 & 4 & 7 & 3 & 6 & 2 & 5 \end{bmatrix} = S(7546)$$

$$\begin{matrix} \times & \times \\ e_1 & e_4 \end{matrix}$$

$$\begin{aligned} e_1 + e_4 &= 7; \\ e_1 &= 7 + e_4; \\ e_1 + 4e_4 &= 5; \\ 7 + e_4 + 4e_4 &= 5; \\ 2e_4 &= 4. \end{aligned}$$

Шукані значення символів вектора помилок $e_4 = 3$; $e_1 = 7 + 3 = 1$.

Таким чином, вектор помилки складає $e = (1003000)$.

Для зменшення ймовірності неправильного декодування кількість помилок, що виправляються декодером, можна обмежити кількістю t , такою, що $2t + 1$ буде строго менше d^* . У цьому випадку декодер завжди буде виявляти не конфігурацію, яка виправляється, за умови, що кількість v дійсно допущених помилок задовольняє нерівність $t + v < d^*$, оскільки для переходу кодового слова в неправильну сферу декодування повинно відбутися не менше $d^* - t$ помилок.

У такому декодері добре застосовувати алгоритм Берлекемпа–Мессі. Для обчислення $\Lambda(x)$ необхідно зробити $2t$ ітерацій за алгоритмом Берлекемпа–Мессі. Інші $\tau - t$ ітерацій, $\tau = d^* - 1 - t$, потрібні для перевірки рівності нулю нев'язань Δ_r . Якщо хоча б при одній із цих додаткових ітерацій Δ_r не дорівнює нулю, то прийняте слово вважається словом, у якому відбулося більше t помилок.

Правомірність описаної процедури доводиться наступною теоремою.

Теорема 1.21. Нехай відбулося не більше τ помилок і задані S_j при $j = 1, \dots, t + \tau$ та $\tau > t$. Нехай далі – регістр зсуву з лінійним зворотним зв'язком, що генерує послідовність – $S_j, j = 1, \dots, 2t$ і

$$\Delta_r = \sum_{j=0}^{n-1} \Lambda_j^{(2t)} S_{r-1},$$

причому Δ_r дорівнюють нулю при $r = 2t + 1, \dots, t + \tau$ та $L_{2t} \leq t$. Тоді відбулося не більше t помилок і $\Lambda^{(2t)}(x)$ буде правильним многочленом локаторів помилок.

Доведення. В умові теореми задані компоненти синдрому S_j для $j = 1, \dots, t + \tau$. Якби були ще $\tau - t$ компонент синдрому $S_j, j = t + \tau + 1, \dots, 2\tau$, то можна було б виправляти t помилок, тобто будь-яку конфігурацію передбачуваних помилок. Використовуючи алгоритм Берлекемпа–Мессі, можна знайти многочлен локаторів помилок, степінь якого дорівнює кількості помилок. Але при $2t$ -й ітерації $L_{2t} \leq t$, і за припущенням Δ_r дорівнює нулю при $r = 2t + 1, \dots, t + \tau$. Тому значення L не буде змінюватися до ітерації з номером $t + \tau + 1$, і, отже, за правилом зміни L

$$L_{2\tau} \geq (t + \tau + 1) - L_{2t} \geq (t + \tau + 1) - t = \tau + 1,$$

а це суперечить припущенню про те, що відбулося не більше τ помилок.

1.8.3. Декодування двійкових кодів БЧХ

У випадку поля $GF(2)$ проведені міркування можна додатково спростити. Очевидне спрощення полягає в тому, що при декодуванні необхідно обчислювати лише позиції помилок, а значення помилок завжди дорівнюють одиниці (хоча декодер може обчислювати значення помилок у якості додаткової перевірки).

Неважко помітити, що значення Δ_r завжди зводиться до нуля при парних ітераціях. Якщо це відбувається завжди при декодуванні двійкових кодів, то ітерації з парними номерами можна пропускати.

У цьому підпункті буде доведено, що справа є саме такою. Доведення ґрунтується на наступному факті: у полі $GF(2)$ синдроми з парними номерами визначаються синдромами з непарними номерами за формулою:

$$S_{2j} = \sum_{j=1}^n v(\alpha^{2j}) = \left[\sum_{j=1}^n v(\alpha^j) \right]^2 = S_j^2,$$

Одержимо для декількох початкових значень r алгебраїчні вирази для коефіцієнтів $\Lambda^{(r)}(x)$. Дотримуючись алгоритму і використовуючи справедливі для всіх двійкових кодів рівності $S_4 = S_2^2 = S_1^4$, одержуємо:

$$\begin{aligned}
\Delta_1 &= S_1, \\
\Delta_2 &= S_2 + S_1^2 = 0, \\
\Delta_3 &= S_3 + S_1 S_2, \\
\Delta_4 &= S_4 + S_1 S_3 + S_1^{-1} S_2 S_3 + S_2^2 = 0, \\
\Lambda^{(1)}(x) &= S_1 x + 1, \\
\Lambda^{(2)}(x) &= S_1 x + 1, \\
\Lambda^{(3)}(x) &= (S_1^{-1} S_3 + S_2) x^2 + S_1 x + 1.
\end{aligned}$$

Звідси випливає, що для будь-яких двійкових кодів БЧХ Δ_2 і Δ_4 завжди дорівнюють нулю.

Нескінченне продовження цього прямого шляху для обчислення інших значень синдромів з парними номерами неможливе. Замість цього сформулюємо загальний результат, який показує, що $\Delta_r = 0$ для всіх парних r .

Теорема 1.22. Нехай у розширенні поля $GF(2)$ задана довільна послідовність $S_1, S_2, \dots, S_{2v-1}$, що задовольняє умову $S_{2j} = S_j^2$ для $2j \leq 2v-1$; нехай також заданий регістр зсуву з лінійним зворотним зв'язком $\Lambda(x)$ і виконуються рівняння:

$$S_j = \sum_{i=1}^{n-1} \Lambda_i S_{j-i}, j = v, \dots, 2v-1$$

Якщо наступний член послідовності задається рівнянням:

$$S_{2v} = \sum_{i=1}^{n-1} \Lambda_i S_{2v-i},$$

то

$$S_{2v} = S_v^2.$$

Доведення. Покажемо, що при зроблених припущеннях вираз для S_v^3 і S_{2v} збігаються. З одного боку,

$$S_{2v} = \left(\sum_{i=1}^{n-1} \Lambda_i S_{v-i} \right)^2 = \sum_{i=1}^{n-1} \Lambda_i^2 S_{v-i}^2 = \sum_{i=1}^{n-1} \Lambda_i^2 S_{2v-2i}.$$

З іншого боку,

$$S_{2v} = \sum_{k=1}^{n-1} \Lambda_k S_{2v-k} = \sum_{k=1}^{n-1} \sum_{i=1}^{n-1} \Lambda_k \Lambda_i S_{2v-k-i}$$

Із симетрії останнього виразу випливає, що кожний доданок із $i \neq k$ входить у суму двічі. Оскільки обчислення виконуються в розширенні поля $GF(2)$, сума цих доданків буде дорівнюватися нулю. Тому внесок в останню суму дадуть тільки діагональні члени, для яких $i = k$:

$$S_{2v} = \sum_{i=1}^{n-1} \Lambda_i^2 S_{2v-2i}.$$

Цей вираз збігається з отриманим на початку доказу виразом для S_v^2 , що доводить теорему.

Таким чином, за індукцією одержуємо, що Δ_r дорівнює нулю для парних r ; тому можна розглядати тільки непарні ітерації, формально поєднуючи дві послідовні ітерації:

$$\Lambda^{(r)}(x) = \Lambda^{(r-2)}(x) - \Lambda_r x^2 B^{(r-2)}(x),$$

$$B^{(r)}(x) = \sigma_r \Delta_r^{-1} \Lambda^{(r-2)}(x) + (1 - \sigma_r) x^2 B^{(r-2)}(x).$$

Використовуючи ці формули, можна пропустити ітерації з парними номерами, і в результаті декодування двійкових кодів прискорюється.

Слід зауважити, що оскільки при доказі теореми були використані лише співвідношення спряженості між компонентами синдрому та нічого не говорилось про двійкові конфігурації помилок, то описане спрощення можливе навіть для конфігурацій більш ніж t помилок. Тому ті ж види перевірки застосовуються при кількості помилок, більшій від t .

1.8.4. Декодування за допомогою алгоритму Евкліда

Декодування за допомогою алгоритму Евкліда являє собою один зі способів декодування, відмінний від розглянутих раніше. Принцип роботи такого декодера трохи простіше зрозуміти, однак вважається, що він менш ефективний у практичному використанні; втім, справедливність такої думки, можливо, значною мірою залежить від конкретного застосування.

Алгоритм Евкліда це рекурентна процедура знаходження найбільшого спільного дільника двох многочленів. Невелике розширення

цього алгоритму дозволяє додатково обчислювати многочлени $a(x)$ і $b(x)$, такі, що:

$$\text{НЗД}[s(x), t(x)] = a(x)s(x) + b(x)t(x).$$

Для двох довільних многочленів $s(x)$ і $t(x)$ повторимо алгоритм Евкліда в більш зручній для нас матричній формі, використовуючи для алгоритму розподілу запис $S(x) = [s(x)/t(x)]t(x) + r(x)$.

Теорема 1.23. (алгоритм Евкліда для многочленів). Нехай задані два многочлени $s(x)$ і $t(x)$, такі, що $\deg s(x) \geq \deg t(x)$, нехай $s^{(0)}(x) = s(x)$, $t^{(0)}(x) = t(x)$ і нехай $A^{(0)}(x) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

Тоді розв'язкою рекурентних рівнянь

$$Q^{(r)}(x) = \left[\frac{s^{(r)}(x)}{t^{(r)}(x)} \right],$$

$$A^{(r+1)}(x) = \begin{bmatrix} 0 & 1 \\ 1 & -Q^{(r)}(x) \end{bmatrix} A^{(r)}(x),$$

$$\begin{bmatrix} s^{(r+1)}(x) \\ t^{(r+1)}(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -Q^{(r)}(x) \end{bmatrix} \begin{bmatrix} s^{(r)}(x) \\ t^{(r)}(x) \end{bmatrix} = A^{(r+1)}(x) \begin{bmatrix} s(x) \\ t(x) \end{bmatrix}$$

буде многочлен

$$S^{(R)}(x) = \gamma \text{НЗД}[s(x), t(x)],$$

причому знайдеться такий скаляр γ , що:

$$\text{НЗД}[s(x), t(x)] = A_{11}^{(R)}(x)s(x) + A_{12}^{(R)}(x)t(x),$$

де R таке, що $t^{(R)}(x) = 0$.

Доведення. Оскільки, то зрештою $t^{(R)}(x) = 0$ при деякому R , і процес обов'язково закінчиться. Таким чином,

$$\begin{bmatrix} S^{(R)}(x) \\ 0 \end{bmatrix} = \left\{ \prod_{r=R-1}^0 \begin{bmatrix} 0 & 1 \\ 1 & -Q^{(r)}(x) \end{bmatrix} \right\} \begin{bmatrix} s(x) \\ t(x) \end{bmatrix},$$

звідки випливає, що будь-який дільник багаточленів $s(x)$ і $t(x)$ ділить також і $S^{(R)}(x)$.

Легко перевірити, що:

$$\begin{bmatrix} 0 & 1 \\ 1 & -Q^{(r)}(x) \end{bmatrix}^{-1} = \begin{bmatrix} Q^{(r)}(x) & 1 \\ 1 & 0 \end{bmatrix}$$

і тому

$$\begin{bmatrix} s(x) \\ t(x) \end{bmatrix} = \left\{ \prod_{r=0}^{R-1} \begin{bmatrix} Q^{(r)}(x) & 1 \\ 1 & 0 \end{bmatrix} \right\} \begin{bmatrix} s^{(R)}(x) \\ 0 \end{bmatrix},$$

так що многочлен $S^{(R)}(x)$ повинен ділити $s(x)$ і $t(x)$, а отже, і НЗД $[s(x), t(x)]$. Звідси одержуємо, що НЗД $[s(x), t(x)]$ ділить $S^{(R)}(x)$ і сам ділиться на $S^{(R)}(x)$; таким чином,

$$S^{(R)}(x) = \gamma \text{НЗД}[s(x), t(x)],$$

Далі

$$\begin{bmatrix} S^{(R)}(x) \\ 0 \end{bmatrix} = A^{(R)}(x) \begin{bmatrix} s(x) \\ t(x) \end{bmatrix},$$

і, отже,

$$S^{(R)} = A_{11}^{(R)}(x)s(x) + A_{12}^{(R)}(x)t(x).$$

Це доводить останнє твердження теореми.

У теоремі 1.23 знайдений зміст матричних елементів $A_{11}^{(R)}(x)$ і $A_{12}^{(R)}(x)$

.

Можна інтерпретувати і два інших елементи, для чого нам потрібно буде звернути матрицю $A^{(r)}(x)$. Нагадаємо, що:

$$A^{(r)}(x) = \prod_{l=r-1}^0 \begin{bmatrix} 0 & 1 \\ 1 & -Q^{(l)}(x) \end{bmatrix}$$

Із цього рівняння видно, що визначник матриці $A^{(r)}(x)$ дорівнює $(-1)^r$. Зворотна матриця запишеться в наступному вигляді:

$$\begin{bmatrix} A_{11}^{(r)}(x) & A_{12}^{(r)}(x) \\ A_{21}^{(r)}(x) & A_{22}^{(r)}(x) \end{bmatrix}^{-1} = (-1)^r \begin{bmatrix} A_{22}^{(r)}(x) & -A_{12}^{(r)}(x) \\ -A_{21}^{(r)}(x) & A_{11}^{(r)}(x) \end{bmatrix}$$

Визначення 1.15. Многочлени $A_{21}^{(r)}(x)$ і $A_{22}^{(r)}(x)$, отримані за допомогою алгоритму Евкліда, задовольняють рівності:

$$s(x) = (-1)^R A_{22}^{(R)}(x) \gamma \text{НЗД}[s(x), t(x)],$$

$$t(x) = -(-1)^R A_{21}^{(R)}(x) \gamma \text{НЗД}[s(x), t(x)].$$

Доведення. Використовуючи вираз для зворотної до $A^{(R)}(x)$ матриці, одержуємо

$$\begin{bmatrix} s(x) \\ t(x) \end{bmatrix} = (-1)^R \begin{bmatrix} A_{22}^{(R)}(x) & -A_{12}^{(R)}(x) \\ -A_{21}^{(R)}(x) & A_{11}^{(R)}(x) \end{bmatrix} \begin{bmatrix} s^{(R)}(x) \\ 0 \end{bmatrix},$$

звідки й випливає твердження визначення.

Опишемо два зовсім різних способи використання алгоритму Евкліда при декодуванні.

Нагадаємо, що синдромний многочлен, многочлен локаторів помилок і многочлен значень помилок пов'язані співвідношенням:

$$\Omega(x) = S(x)\Lambda(x) \pmod{x^{2t}}$$

і умовами $\deg \Lambda(x) \leq t$ й $\deg \Omega(x) \leq t-1$. Спробуємо тепер використати доказ алгоритму Евкліда для рішення цього розв'язання відносно $\Lambda(x)$ й $\Omega(x)$. Із цього доказу легко встановити, що:

$$\begin{bmatrix} s^{(r)}(x) \\ t^{(r)}(x) \end{bmatrix} = (-1)^R \begin{bmatrix} A_{11}^{(r)}(x) & -A_{12}^{(r)}(x) \\ A_{21}^{(r)}(x) & A_{22}^{(r)}(x) \end{bmatrix} \begin{bmatrix} s(x) \\ t(x) \end{bmatrix},$$

і тому

$$t^{(r)}(x) = t(x)A_{22}^{(r)}(x) \pmod{s(x)}.$$

Якщо припустити, що $t(x) = S(x)$ і $s(x) = x^{2t}$, то останнє рівняння можна розглядати як рівняння, яке нам необхідно розв'язати.

Це рівняння виконується для кожного r .

Щоб вирішити поставлене завдання, необхідно знайти таке значення r (якщо воно існує), при якому $\deg A_{22}^{(r)}(x) \leq t$ і $\deg t^{(r)}(x) \leq t-1$. Задовольнити останні умови можна, вибравши r' що дорівнює такому значенню r , при якому:

$$\deg t^{(r-1)}(x) \geq t \text{ і } \deg t^{(r-1)}(x) \leq t - 1$$

Оскільки $\deg t^{(0)}(x) = 2t$ й степінь $t^{(r)}(x)$ строго спадає зі зростанням r , ці нерівності визначають єдине значення r' .

За визначенням r' одержуємо:

$$\deg t^{(r')}(x) \leq t - 1.$$

Степінь $A_{22}^{(r)}(x)$ зростає зі зростанням r . Залишається лише показати, що:

$$\deg A_{22}^{(r')}(x) \leq t.$$

Ця нерівність доводиться за допомогою обігу матриці $A(x)$. Спочатку нагадаємо, що:

$$A^{(r')}(x) = \prod_{r=r'-1}^0 \begin{bmatrix} 0 & 1 \\ 1 & -Q^{(r)}(x) \end{bmatrix}.$$

звідки випливає що $\deg A_{22}^{(r)}(x) \geq \deg A_{12}^{(r)}(x)$.

Нагадаємо також, що $\deg s^{(r)}(x) > \deg t^{(r)}(x)$. Із цих нерівностей і матричного рівняння:

$$\begin{bmatrix} s(x) \\ t(x) \end{bmatrix} = (-1)^r \begin{bmatrix} A_{22}^{(r)}(x) & -A_{12}^{(r)}(x) \\ -A_{21}^{(r)}(x) & A_{11}^{(r)}(x) \end{bmatrix} \begin{bmatrix} s^{(r)}(x) \\ t^{(r)}(x) \end{bmatrix}$$

впливає, що $\deg s(x) = \deg A_{22}^{(r')}(x) + \deg s^{(r')}(x)$, і, оскільки $s^{(r')}(x) = t^{(r'-1)}(x)$, одержуємо

$$\deg A_{22}^{(r')}(x) = \deg s(x) - \deg t^{(r'-1)}(x) \leq 2t - 1 = t,$$

де нерівність випливає з визначення r' .

Тепер ми майже повністю довели наступну теорему.

Теорема 1.24. Нехай задані $S^{(0)}(x) = x^{2t}$, $t^{(0)}(x) = S(x)$, де $S(x)$ – синдромний многочлен коду БЧХ, що виправляє t помилок, і нехай $A^{(0)}(x) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Будемо розв'язувати наступні рекурентні рівняння доти,

поки не буде виконана нерівність $\deg t^{(r')}(x) \leq t - 1$:

$$Q^{(r)}(x) = \left[\frac{s^{(r)}(x)}{t^{(r)}(x)} \right]$$

$$A^{(r+1)}(x) = \begin{bmatrix} 0 & 1 \\ 1 & -Q^{(r)}(x) \end{bmatrix} A^{(r)}(x)$$

$$\begin{bmatrix} s^{(r+1)}(x) \\ t^{(r+1)}(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -Q^{(r)}(x) \end{bmatrix} \begin{bmatrix} s^{(r)}(x) \\ t^{(r)}(x) \end{bmatrix}$$

Тоді многочлени:

$$\Omega(x) = \Delta^{-1} t^{(r)}(x),$$

$$\Lambda(x) = \Delta^{-1} A_{22}^{(r)}(x),$$

де $\Delta(x) = A_{22}^{(r)}(0)$ є єдиним розв'язком, рівняння

$$\Omega(x) = S(x)\Lambda(x) \pmod{x^{2t}},$$

задовольняє умови $\deg \Lambda(x) \leq t, \Lambda_0 = 1$ і $\deg \Omega(x) \leq t - 1$.

Доведення. Розподіл на Δ забезпечує рівність $\Lambda_0 = 1$. З іншого боку, з попередніх міркувань випливає, що останнє рівняння і всі умови задовольняються. Одиначність отриманого розв'язку випливає з того факту, що для синдрому коду БЧХ, що виправляє t помилок, існує тільки один такий розв'язок.

Після того як $\Lambda(x)$ і $\Omega(x)$ знайдені, декодування можна завершити кожним з методів, запропонованих для алгоритму Берлекемпа–Мессі. Блок-схема відповідного декодера наведена на рис. 1.34, де декодер завершується алгоритмом Форні (хоча його можна завершити інакше).

1.8.5. Алгебраїчне декодування алгеброгеометричних кодів

Розглянемо завдання декодування кодового слова алгеброгеометричного коду, а потім узагальнимо результат для обмежень кодів на довільне підполе.

Припустимо, що при передачі каналом з помилками кодове слово алгеброгеометричного коду спотворилося, вектор помилок позначимо як:

$$e = (e_0, e_1, \dots, e_{n-1}).$$

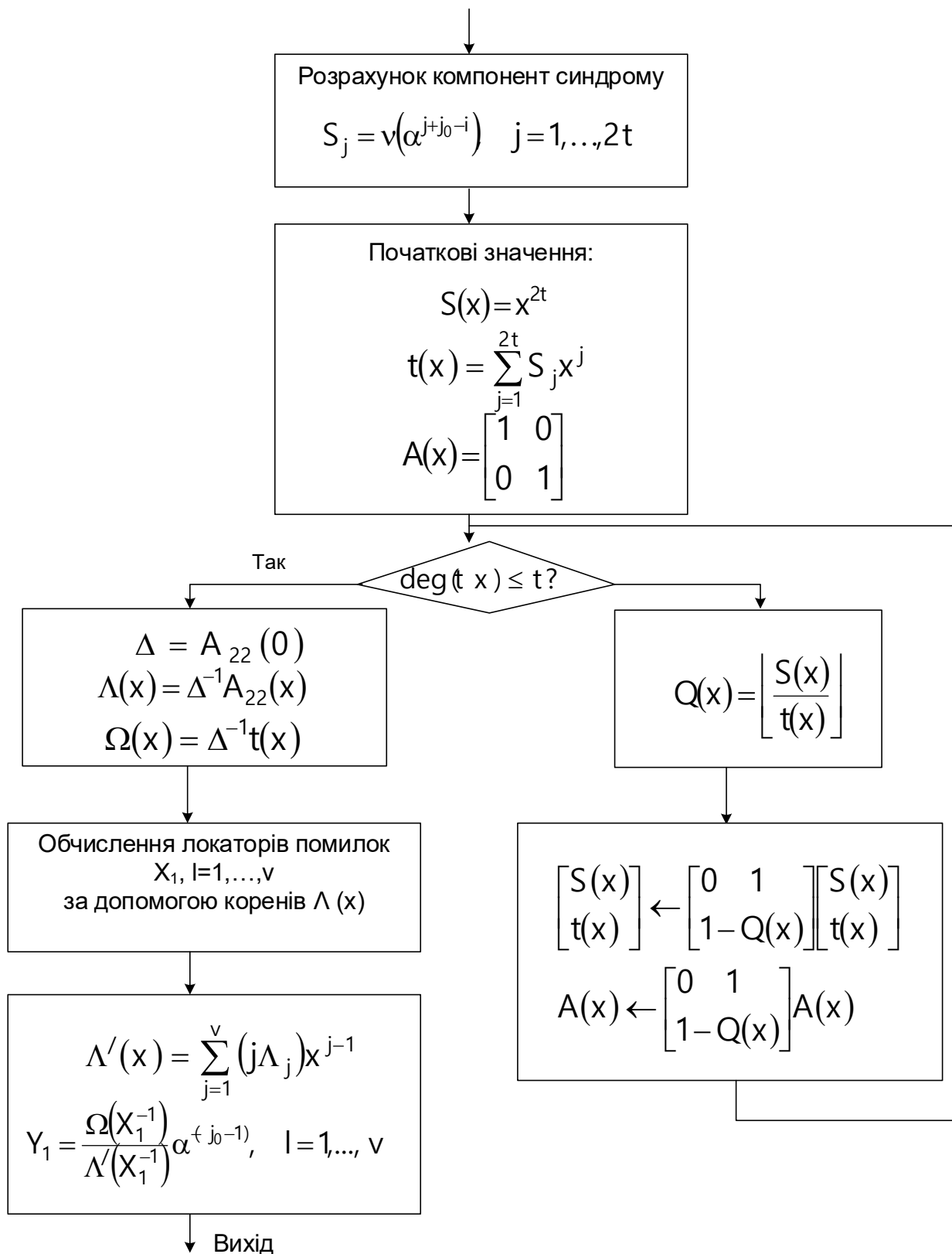


Рис. 1.34. Декодування кодів БЧХ за допомогою алгоритму Евкліда

Прийняте слово s^* після передачі каналом із помилками запишеться у вигляді:

$$c^* = c + e = (e_0 + c_0, e_1 + c_1, \dots, e_{n-1} + c_{n-1}).$$

Визначимо синдромну послідовність як вектор:

$$S = (S_1, S_2, \dots, S_r),$$

обчислений за таким правилом:

$$S_j = \sum_{i=0}^{n-1} c_i^* F_j(P_i), j = 1..r,$$

або в матричній формі:

$$\|S_j\|_r = \|F_j(P_i)\|_{n,r} \|c_i^*\|_n^T.$$

Очевидно, що:

$$S_j = \sum_{i=0}^{n-1} [c_i F_j(P_i) + e_i F_j(P_i)] = \sum_{i=0}^{n-1} e_i F_j(P_i), j = 1, \dots, r,$$

або в матричній формі:

$$\|S_j\|_r = \|F_j(P_i)\|_{n,r} \|e_i\|_n^T = H \|e_i\|_n^T,$$

значення синдрому залежить тільки від вектора помилок і не залежить від кодового слова.

Завдання декодування алгеброгеометричного коду полягають у знаходженні вектора помилок:

$$e = (e_0, e_1, \dots, e_{n-1})$$

за відомою синдромною послідовністю:

$$S = (S_0, S_1, \dots, S_{r-1}).$$

Розглянемо як генераторні функції однорідні одночлени степеня $\deg$. Кожен такий одночлен запишемо у вигляді:

$$f_{lmp} = x^l y^m z^p, l + m + p = \deg.$$

На множині проєктивних точок кривої X , які зображені в однорідних координатах у вигляді $P(x, y, 1)$, значення генераторних функцій набудуть вигляду:

$$f_{lm} = X_i^l Y_i^m, i = 0 \dots n-1, l+m \leq \deg.$$

Перевірна матриця H запишеться у вигляді:

$$H = \begin{pmatrix} 1 & 1 & \dots & 1 \\ X_0 & X_1 & \dots & X_{n-1} \\ \dots & \dots & \dots & \dots \\ Y_0^{\deg F} & Y_1^{\deg F} & \dots & Y_{n-1}^{\deg F} \end{pmatrix}.$$

Елементи синдромної послідовності як елементи вектора $\|S_{lm}\|$ обчислимо за правилом:

$$S_{lm} = \sum_{i=0}^{n-1} c_i^* X_i^l Y_i^m = \sum_{i=0}^{n-1} e_i X_i^l Y_i^m, l+m \leq \deg$$

або в матричній формі:

$$\|S_{lm}\|_r = H \|c_n^*\|_n^T = \|X_i^l Y_i^m\|_{n,r} \|e_i\|_n^T. \quad (1.92)$$

Таким чином, завдання декодування алгеброгеометричного коду, побудованого через відображення проєктивних точок $P(x, y, 1)$ кривої однорідними одночленами степеня $\deg$, еквівалентне завданню розв'язання системи з $r = d + g$ нелінійних рівнянь від $3t$ змінних.

Для вирішення цього завдання скористаємося штучним прийомом, що полягає у введенні в розгляд многочлена локаторів помилок, вирішення якого однозначно вкаже місце розташування помилок (локалізує помилки). Визначимо многочлен локаторів помилок алгеброгеометричного коду як многочлен від двох змінних степеня $\leq (t-1)$:

$$a_{00} + a_{10}x + \dots + y^{t-1} = 0, \quad (1.93)$$

де t – кількість помилок, що може виправити алгеброгеометричний код.

Дамо геометричне трактування введеного многочлена локаторів помилок від двох змінних. Пояснимо його фізичний зміст і особливо пояснимо його відмінність від многочлена локаторів помилок від однієї змінної, який використовується в алгоритмі Берлекемпа [5; 8].

Многочлен локаторів помилок від однієї змінної (в алгоритмі Берлекемпа) задається виразом $a_0 + a_1x + \dots + x^t = 0$ і в двовимірному просторі (на площині) описує деяку лінію (рис. 1.35), що перетинає вісь x

не більше t разів. Точки перетину з віссю x відповідають локаторам помилок. Для локалізації t помилок досить знайти t точок.

Інакше кажучи, якщо многочлен локаторів помилок задає пряму лінію, потрібно знайти тільки одну точку – точку перетину цієї прямої з віссю x .

Якщо многочлен задає лінію другого порядку (параболу), потрібно знайти дві точки перетину цієї параболи з віссю Ox і т. д.

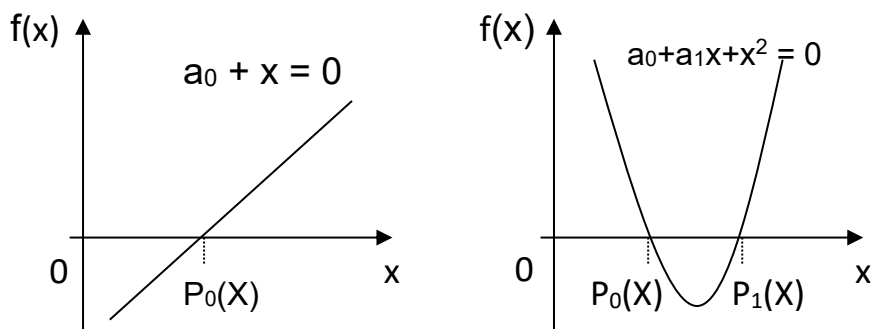


Рис. 1.35. Графічне подання многочлена локаторів помилок в алгоритмі Берлекемпа

У тривимірному просторі многочлен локаторів помилок алгеброгеометричного коду від двох змінних (1.93) описується поверхнею, що перетинає площину xu по деякій лінії порядку t (рис. 1.36).

Для того, щоб однозначно задати лінію порядку t на площині xOy , потрібно визначити як мінімум $t + 1$ точок. Інакше кажучи, для локалізації t помилок необхідно знайти точки перетину поверхні із площиною xOy . Цих точок у загальному випадку може бути більше t , але тільки t точок відповідають реально, що відбулися помилки.

Помноживши обидві частини многочлена (1.93) на e_i і підсумувавши за всіма $i = 0 \dots n - 1$ значеннями в точці

$$(x = X_i, y = Y_i),$$

одержимо рекурентний вираз:

$$a_{ij}S_{ij} + a_{i+1j} S_{i+1j} + \dots + S_{i+j-1} = 0,$$

який задає систему лінійних рівнянь щодо невідомих коефіцієнтів многочлена локаторів помилок.

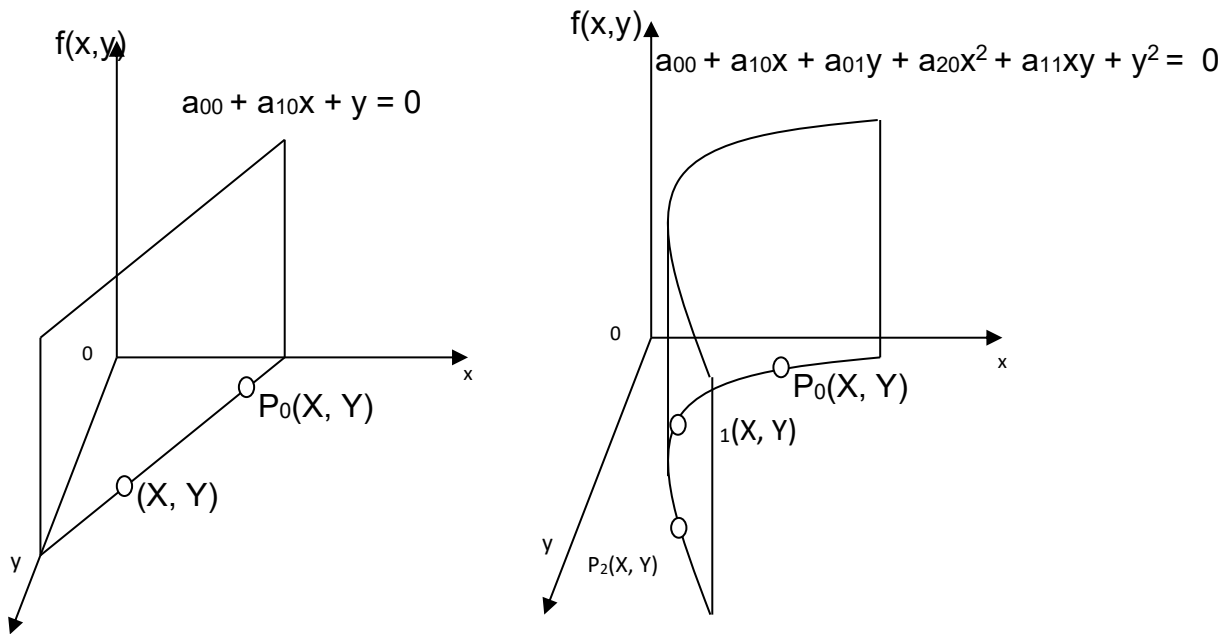


Рис. 1. 36. Графічне подання многочлена локаторів помилок алгеброгеометричних кодів

У матричному вигляді система лінійних рівнянь запишеться так:

$$\begin{pmatrix} S_{00} & S_{10} & \dots & S_{1t-2} \\ S_{10} & S_{20} & \dots & S_{2t-2} \\ \dots & \dots & \dots & \dots \\ S_{1t-2} & S_{0t-2} & \dots & S_{2t-4} \end{pmatrix} \begin{pmatrix} a_{00} \\ a_{10} \\ \dots \\ a_{1t-2} \end{pmatrix} = \begin{pmatrix} -S_{0t-1} \\ -S_{1t-1} \\ \dots \\ -S_{12t-3} \end{pmatrix}. \quad (1.94)$$

Після знаходження коефіцієнтів многочлена локаторів помилок процедура локалізації помилок полягає в підстановці всіх можливих локаторів і виборі тих із них, які перетворюють у нуль многочлен локаторів помилок, тобто підставляються всі пари (x, y) проєктивних точок кривої, які задані в однорідних координатах $P(x, y, 1)$.

Після знаходження локаторів помилок, що вказують на розташування виниклої помилки, процедура знаходження кратності помилки (значення всіх $e_i \neq 0$) полягає в підстановці локаторів у систему (1.94), що вироджується в систему $\leq r$ лінійних рівнянь відносно $\leq t$ невідомих.

Алгоритм декодування алгеброгеометричних кодів задамо у вигляді послідовності таких кроків.

Крок 1. За виразом (1.89) обчислимо елементи синдромної послідовності.

Крок 2. Розв'яжемо систему лінійних рівнянь (1.94). Одержимо коефіцієнти многочлена локаторів помилок.

Крок 3. Підставимо всі пари (x, y) , що відповідають проєктивним точкам кривої, у многочлен локаторів помилок. Ті пари, які при підстановці в цей многочлен перетворюють його в нуль, локалізують помилки, тобто вказують на їхнє шукане розташування.

Крок 4. Підставляємо отримані локатори помилок у систему рівнянь (1.89). Розв'язання системи лінійних рівнянь дасть значення (кратність) помилок, що відбулися. Локалізація помилок і знайдені їхні значення дозволяють сформувати вектор помилок:

$$e = (e_0, e_1, \dots, e_{n-1}).$$

Крок 5. Виправимо помилки:

$$c = c^* - e.$$

Основні етапи розробленого алгебраїчного алгоритму складаються з розв'язання системи лінійних рівнянь (кроки 2 й 4 і крок 3). Ці стандартні процедури, а також процедура обчислення вектора синдромів, можуть бути реалізовані кожним із відомих на сьогодні алгоритмів [5; 6]. Складність розв'язання системи лінійних рівнянь методом Гаусса – це $O(n^2)$, де n – кількість змінних. Оцінимо кількість змінних при розв'язанні системи лінійних рівнянь (1.93).

Многочлен локаторів помилок (1.92) заданий як многочлен від двох невідомих і складається зі всіх одночленів від двох змінних степеня $\leq (t - 1)$. Кількість одночленів від двох змінних степеня b визначимо з виразу:

$$M = C_{b-1}^1 = b - 1.$$

Тоді кількість невідомих у многочлені локаторів помилок запишеться у вигляді:

$$n = \sum_{i=1}^{t-1} C_{i-1}^1 = \sum_{i=1}^{t-1} (i - 1) = \frac{t^2 - t}{2}.$$

Отже, для декодування алгеброгеометричного коду, побудованого через відображення проєктивних точок $P(x, y, 1)$ на кроці 2, необхідно

розв'язати систему лінійних рівнянь від $(t^2 - t)/2$ змінних, складність її розв'язання методом Гаусса – $O((t^2 - t)^{2/4})$.

Загальна складність алгоритму декодування:

$$O(4t^2 + (t^2 - t)^{2/4}).$$

Розглянемо приклад декодування алгеброгеометричного коду з використанням розробленого алгоритму.

Приклад. Розглянемо алгебраїчну криву як сукупність розв'язків однорідного алгебраїчного рівняння:

$$x^{\sqrt{q}+1} + y^{\sqrt{q}}z + yz^{\sqrt{q}} = 0$$

над полем $GF(q)$, $q = r^2$, r – додатне ціле число.

Над $GF(2^2)$ це рівняння набуде вигляду:

$$x^3 + y^2z + yz^2 = 0.$$

При підстановці елементів поля $GF(2^2)$ у рівняння кривої одержимо 9 розв'язків, які подані в табл. 1.26, рід кривої $g = 1$.

Таблица 1.26

Точки кривої Ерміта в P^2 над $GF(4)$

X	Y	Z
0	1	0
0	0	1
0	1	1
1	2	1
2	2	1
3	2	1
1	3	1
2	3	1
3	3	1

Нехай $\deg = 2$, тоді алгеброгеометричний код за кривою, заданою множиною розв'язків рівняння (1.72) вигляду $P(x, y, 1)$ над $GF(2^2)$, – це лінійний код, що складається з усіх слів:

$$c = (c_0, c_2, \dots, c_7),$$

для яких виконується рівність $r = d + g - 1 = 6$ рівнянь:

$$\sum_{i=0}^7 c_i X_i^l Y_i^m = 0, l + m \leq 2,$$

де $c_i \in GF(4)$, $\alpha = 6$, $d \geq \alpha - 2g + 2 = 6$.

Перевірочна матриця H запишеться у вигляді:

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 2 & 3 & 1 & 2 & 3 \\ 0 & 1 & 2 & 2 & 2 & 3 & 3 & 3 \\ 0 & 0 & 1 & 3 & 2 & 1 & 3 & 2 \\ 0 & 0 & 2 & 3 & 1 & 3 & 1 & 2 \\ 0 & 1 & 3 & 3 & 3 & 2 & 2 & 2 \end{pmatrix}.$$

Нехай при передачі каналом із помилками кодове слово

$$c = (c_0, c_2, \dots, c_7)$$

спотворилося, вектор помилок

$$e = (0 \ 0 \ 0 \ 2 \ 0 \ 0 \ 0 \ 1).$$

Синдромний вектор дорівнює:

$$S = \begin{pmatrix} S_{00} \\ S_{10} \\ S_{01} \\ S_{20} \\ S_{11} \\ S_{02} \end{pmatrix} = \begin{pmatrix} 3 \\ 0 \\ 0 \\ 3 \\ 3 \\ 3 \end{pmatrix}.$$

Кількість помилок, які може виправити код, $t = 2$. Многочлен локаторів помилок у загальному вигляді набуде вигляду:

$$a_{00} + a_{10}x + y = 0.$$

У матричному вигляді система лінійних рівнянь запишеться так:

$$\begin{pmatrix} S_{00} & S_{10} \\ S_{10} & S_{20} \end{pmatrix} \begin{pmatrix} a_{00} \\ a_{10} \end{pmatrix} = \begin{pmatrix} -S_{01} \\ -S_{11} \end{pmatrix},$$

або для розглянутого прикладу:

$$\begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} a_{00} \\ a_{10} \end{pmatrix} = \begin{pmatrix} 0 \\ 3 \end{pmatrix}.$$

Розв'язання отриманої системи лінійних рівнянь:

$$a_{00} = 0, a_{10} = 1.$$

Шуканий многочлен локаторів помилок набуде вигляду:

$$x + y = 0.$$

Скористаємося процедурою Ченя, підставивши всі пари (x, y) , що відповідають проєктивним точкам $P(x, y, 1)$, у многочлен локаторів помилок. Пари

$$(0, 0), (2, 2) \text{ і } (3, 3)$$

перетворюють його в нуль, тобто локалізують помилки, вказуючи, що помилки знаходяться в символах c_0^*, c_3^*, c_7^* , і відповідають проєктивним точкам $P(0, 0, 1), P(2, 2, 1), P(3, 3, 1)$.

Використовуючи знайдені локатори помилок одержимо

$$\begin{aligned} e_0 + e_3 + e_7 &= 3; \\ 2e_3 + 3e_7 &= 0; \\ 2e_3 + 3e_7 &= 0; \\ 3e_3 + 2e_7 &= 1; \\ 3e_3 + 2e_7 &= 2; \\ 3e_3 + 2e_7 &= 3. \end{aligned}$$

Розв'язавши останню систему, одержимо:

$$\begin{aligned} e_0 &= 0; \\ e_3 &= 2; \\ e_7 &= 1. \end{aligned}$$

Вектор помилок $e = (0 \ 0 \ 0 \ 2 \ 0 \ 0 \ 0 \ 1)$.

Виправимо помилку:

$$c = c^* - e.$$

Розглянемо ще один приклад декодування алгеброгеометричного коду, що дозволяє виправляти більш складні комбінації помилок.

Приклад. Над $GF(2^6)$ рівняння (1.74) набуде вигляду

$$x^5 + y^4z + yz^4 = 0.$$

При підстановці елементів поля $GF(2^6)$ у рівняння кривої одержимо 513 розв'язків, поданих у табл. 1.27, рід кривої $g = 28$.

Таблиця 1.27

Точки кривої Ерміта в P^2 над $GF(64)$

X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	1	0	20	5	1	55	8	1	21	13	1	54	16	1	16	21	1
0	0	1	27	5	1	62	8	1	28	13	1	61	16	1	23	21	1
0	1	1	34	5	1	4	9	1	35	13	1	7	17	1	30	21	1
4	2	1	41	5	1	11	9	1	42	13	1	14	17	1	37	21	1
11	2	1	48	5	1	18	9	1	49	13	1	21	17	1	44	21	1
25	2	1	62	5	1	32	9	1	63	13	1	35	17	1	58	21	1
32	2	1	3	6	1	39	9	1	7	14	1	42	17	1	1	22	1
39	2	1	10	6	1	46	9	1	14	14	1	49	17	1	8	22	1
46	2	1	17	6	1	53	9	1	21	14	1	56	17	1	15	22	1
53	2	1	24	6	1	60	9	1	28	14	1	63	17	1	22	22	1
60	2	1	31	6	1	0	10	1	35	14	1	5	18	1	29	22	1
7	3	1	38	6	1	5	11	1	42	14	1	12	18	1	36	22	1
14	3	1	45	6	1	12	11	1	49	14	1	19	18	1	43	22	1
21	3	1	52	6	1	19	11	1	56	14	1	26	18	1	50	22	1
28	3	1	59	6	1	26	11	1	63	14	1	33	18	1	57	22	1
35	3	1	4	7	1	33	11	1	4	15	1	40	18	1	1	23	1
42	3	1	11	7	1	40	11	1	11	15	1	47	18	1	8	23	1
49	3	1	18	7	1	47	11	1	18	15	1	54	18	1	15	23	1
56	3	1	25	7	1	54	11	1	25	15	1	61	18	1	22	23	1
63	3	1	32	7	1	61	11	1	32	15	1	0	19	1	29	23	1
6	4	1	39	7	1	1	12	1	39	15	1	6	20	1	36	23	1
13	4	1	46	7	1	8	12	1	46	15	1	13	20	1	43	23	1
20	4	1	53	7	1	15	12	1	53	15	1	20	20	1	50	23	1
27	4	1	60	7	1	22	12	1	60	15	1	27	20	1	57	23	1
34	4	1	6	8	1	29	12	1	5	16	1	34	20	1	5	24	1
41	4	1	13	8	1	36	12	1	12	16	1	41	20	1	12	24	1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
48	4	1	20	8	1	43	12	1	19	16	1	48	20	1	19	24	1
55	4	1	27	8	1	50	12	1	26	16	1	55	20	1	26	24	1
62	4	1	34	8	1	57	12	1	33	16	1	62	20	1	33	24	1
6	5	1	41	8	1	7	13	1	40	16	1	2	21	1	40	24	1
13	5	1	48	8	1	14	13	1	47	16	1	9	21	1	47	24	1
54	24	1	42	29	1	34	33	1	15	38	1	14	42	1	0	46	1

X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z
61	24	1	49	29	1	41	33	1	22	38	1	21	42	1	2	47	1
6	25	1	56	29	1	48	33	1	29	38	1	28	42	1	9	47	1
13	25	1	63	29	1	55	33	1	36	38	1	35	42	1	16	47	1
20	25	1	3	30	1	62	33	1	43	38	1	42	42	1	23	47	1
27	25	1	10	30	1	7	34	1	50	38	1	49	42	1	30	47	1
34	25	1	17	30	1	14	34	1	57	38	1	56	42	1	37	47	1
41	25	1	24	30	1	21	34	1	4	39	1	63	42	1	44	47	1
48	25	1	31	30	1	28	34	1	11	39	1	1	43	1	51	47	1
55	25	1	38	30	1	35	34	1	18	39	1	8	43	1	58	47	1
62	25	1	45	30	1	42	34	1	25	39	1	15	43	1	5	48	1
1	26	1	52	30	1	49	34	1	32	39	1	22	43	1	12	48	1
8	26	1	59	30	1	56	34	1	39	39	1	29	43	1	19	48	1
15	26	1	2	31	1	63	34	1	46	39	1	36	43	1	26	48	1
22	26	1	9	31	1	2	35	1	53	39	1	43	43	1	33	48	1
29	26	1	16	31	1	9	35	1	60	39	1	50	43	1	40	48	1
36	26	1	23	31	1	16	35	1	3	40	1	57	43	1	47	48	1
43	26	1	30	31	1	23	35	1	10	40	1	3	44	1	54	48	1
50	26	1	37	31	1	30	35	1	17	40	1	10	44	1	61	48	1
57	26	1	44	31	1	37	35	1	24	40	1	17	44	1	4	49	1
6	27	1	51	31	1	44	35	1	31	40	1	24	44	1	11	49	1
13	27	1	58	31	1	51	35	1	38	40	1	31	44	1	18	49	1
20	27	1	2	32	1	58	35	1	45	40	1	38	44	1	25	49	1
27	27	1	9	32	1	7	36	1	52	40	1	45	44	1	32	49	1
34	27	1	16	32	1	14	36	1	59	40	1	52	44	1	39	49	1
41	27	1	23	32	1	21	36	1	3	41	1	59	44	1	46	49	1
48	27	1	30	32	1	28	36	1	10	41	1	1	45	1	53	49	1
55	27	1	37	32	1	35	36	1	17	41	1	8	45	1	60	49	1
62	27	1	44	32	1	42	36	1	24	41	1	15	45	1	4	50	1
0	28	1	51	32	1	49	36	1	31	41	1	22	45	1	11	50	1
7	29	1	58	32	1	56	36	1	38	41	1	29	45	1	18	50	1
35	29	1	27	33	1	8	38	1	7	42	1	53	50	1	30	54	1
5	59	1	47	62	1	60	50	1	37	54	1	12	59	1	54	62	1
1	51	1	44	54	1	19	59	1	61	62	1	8	51	1	51	54	1
26	59	1	3	63	1	15	51	1	58	54	1	33	59	1	10	63	1
22	51	1	0	55	1	40	59	1	17	63	1	29	51	1	3	56	1
47	59	1	24	63	1	36	51	1	10	56	1	54	59	1	31	63	1
43	51	1	17	56	1	61	59	1	38	63	1	50	51	1	24	56	1
2	60	1	45	63	1	57	51	1	31	56	1	9	60	1	52	63	1
2	52	1	38	56	1	16	60	1	59	63	1	9	52	1	45	56	1
23	60	1	57	45	1	16	52	1	52	56	1	30	60	1	46	50	1
23	52	1	59	56	1	37	60	1	14	29	1	30	52	1	6	57	1

X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z
44	60	1	21	29	1	37	52	1	13	57	1	51	60	1	28	29	1
44	52	1	20	57	1	58	60	1	6	33	1	51	52	1	27	57	1
3	61	1	13	33	1	58	52	1	34	57	1	10	61	1	20	33	1
4	53	1	41	57	1	17	61	1	63	36	1	11	53	1	48	57	1
24	61	1	0	37	1	18	53	1	55	57	1	31	61	1	1	38	1
25	53	1	62	57	1	38	61	1	45	41	1	32	53	1	5	58	1
45	61	1	52	41	1	39	53	1	12	58	1	52	61	1	59	41	1
46	53	1	19	58	1	59	61	1	36	45	1	53	53	1	26	58	1
5	62	1	43	45	1	60	53	1	33	58	1	12	62	1	50	45	1
2	54	1	40	58	1	19	62	1	25	50	1	9	54	1	47	58	1
26	62	1	32	50	1	16	54	1	54	58	1	33	62	1	39	50	1
23	54	1	61	58	1	40	62	1									

Нехай $\deg = 7$, тоді алгеброгеометричний код за кривою – це лінійний код, що складається з усіх слів $c = (c_0, c_2, \dots, c_{511})$, для яких виконується рівність $r = d + g - 1 = 36$ та $\sum_{i=0}^{511} c_i X_i^l Y_i^m = 0$, $l + m \leq 7$, де

$c_i \in GF(64)$, $\alpha = 63$, $d \geq \alpha - 2g + 2 = 9$.

Нехай при передачі каналом із помилками кодове слово:

$$c = (c_0, c_2, \dots, c_{511})$$

спотворилося, вектор помилок $e = (e_0, e_1, \dots, e_{511})$ дорівнює:

$$e_1 = 32, e_{71} = 2, e_{343} = 1, e_{508} = 57,$$

$$e_i = 0, i \notin (1, 71, 343, 508).$$

Помилки локалізуються відповідними точками:

$$P_1(0, 1, 1), P_{71}(46, 9, 1), P_{343}(29, 43, 1), P_{508}(38, 63, 1).$$

Синдромний вектор дорівнює:

$$\begin{aligned} S_{00} = 61, S_{10} = 42, S_{01} = 24, S_{20} = 17, S_{11} = 24, S_{02} = 41, S_{30} = 37, \\ S_{21} = 53, S_{12} = 48, S_{03} = 38, S_{40} = 29, S_{31} = 63, S_{22} = 60, S_{13} = 15, S_{04} = 10, \\ S_{50} = 45, S_{41} = 33, S_{32} = 36, S_{23} = 4, S_{14} = 24, S_{05} = 16, S_{60} = 12, S_{51} = 45, \\ S_{42} = 0, S_{33} = 3, S_{24} = 31, S_{15} = 25, S_{06} = 14, S_{70} = 13, S_{61} = 55, S_{52} = 53, \\ S_{43} = 9, S_{34} = 55, S_{25} = 53, S_{16} = 16, S_{07} = 22. \end{aligned}$$

Конструктивні характеристики (512, 476, 9) коду гарантують здатність до виправлення, $t = 4$ помилок.

Многочлен локаторів помилок у загальному вигляді такий:

$$a_{00} + a_{10}x + a_{01}y + a_{20}x^2 + a_{11}xy + a_{02}y^2 + a_{30}x^3 + a_{21}x^2y + a_{12}xy^2 + y^3 = 0.$$

У матричному вигляді система лінійних рівнянь запишеться так:

$$\begin{pmatrix} S_{00} & S_{10} & S_{01} & S_{20} & S_{11} & S_{02} & S_{30} & S_{21} & S_{12} \\ S_{10} & S_{20} & S_{11} & S_{30} & S_{21} & S_{12} & S_{40} & S_{31} & S_{22} \\ S_{01} & S_{11} & S_{02} & S_{21} & S_{12} & S_{03} & S_{31} & S_{22} & S_{13} \\ S_{20} & S_{30} & S_{21} & S_{40} & S_{31} & S_{22} & S_{50} & S_{41} & S_{32} \\ S_{11} & S_{21} & S_{12} & S_{31} & S_{22} & S_{13} & S_{41} & S_{32} & S_{23} \\ S_{02} & S_{12} & S_{03} & S_{22} & S_{13} & S_{04} & S_{32} & S_{23} & S_{14} \\ S_{30} & S_{40} & S_{31} & S_{50} & S_{41} & S_{32} & S_{60} & S_{51} & S_{42} \\ S_{21} & S_{31} & S_{22} & S_{41} & S_{32} & S_{23} & S_{51} & S_{42} & S_{33} \\ S_{12} & S_{22} & S_{13} & S_{32} & S_{23} & S_{14} & S_{42} & S_{33} & S_{24} \end{pmatrix} \begin{pmatrix} a_{00} \\ a_{10} \\ a_{01} \\ a_{20} \\ a_{11} \\ a_{02} \\ a_{30} \\ a_{21} \\ a_{12} \end{pmatrix} = \begin{pmatrix} S_{03} \\ S_{13} \\ S_{04} \\ S_{23} \\ S_{14} \\ S_{05} \\ S_{33} \\ S_{24} \\ S_{15} \end{pmatrix},$$

або для розглянутого прикладу:

$$\begin{pmatrix} 61 & 42 & 24 & 17 & 24 & 41 & 37 & 53 & 48 \\ 42 & 17 & 24 & 37 & 53 & 48 & 29 & 63 & 60 \\ 24 & 24 & 41 & 53 & 48 & 38 & 63 & 60 & 15 \\ 17 & 37 & 53 & 29 & 63 & 60 & 45 & 33 & 36 \\ 24 & 53 & 48 & 63 & 60 & 15 & 33 & 36 & 4 \\ 41 & 48 & 38 & 60 & 15 & 10 & 36 & 4 & 24 \\ 37 & 29 & 63 & 45 & 33 & 36 & 12 & 45 & 0 \\ 53 & 63 & 60 & 33 & 36 & 4 & 45 & 0 & 3 \\ 48 & 60 & 15 & 36 & 4 & 24 & 0 & 3 & 31 \end{pmatrix} \begin{pmatrix} a_{00} \\ a_{10} \\ a_{01} \\ a_{20} \\ a_{11} \\ a_{02} \\ a_{30} \\ a_{21} \\ a_{12} \end{pmatrix} = \begin{pmatrix} 38 \\ 15 \\ 10 \\ 4 \\ 24 \\ 16 \\ 3 \\ 31 \\ 25 \end{pmatrix}.$$

Система рівнянь, ранг квадратної матриці в лівій частині дорівнює 4. З урахуванням значення рангу матриці многочлен локаторів помилок запишемо у вигляді $a_{00} + a_{10}x + a_{01}y + a_{20}x^2 + xy = 0$, а систему лінійних рівнянь запишемо як

$$\begin{pmatrix} S_{00} & S_{10} & S_{01} & S_{20} \\ S_{10} & S_{20} & S_{11} & S_{30} \\ S_{01} & S_{11} & S_{02} & S_{21} \\ S_{20} & S_{30} & S_{21} & S_{40} \end{pmatrix} \begin{pmatrix} a_{00} \\ a_{10} \\ a_{01} \\ a_{20} \end{pmatrix} = \begin{pmatrix} S_{11} \\ S_{21} \\ S_{12} \\ S_{31} \end{pmatrix}.$$

Підставивши значення елементів вектора синдромів, одержимо:

$$\begin{pmatrix} 61 & 42 & 24 & 17 \\ 42 & 17 & 24 & 37 \\ 24 & 24 & 41 & 53 \\ 17 & 37 & 53 & 29 \end{pmatrix} \begin{pmatrix} a_{00} \\ a_{10} \\ a_{01} \\ a_{20} \end{pmatrix} = \begin{pmatrix} 24 \\ 53 \\ 48 \\ 63 \end{pmatrix}.$$

Розв'язання отриманої системи лінійних рівнянь:

$$a_{00} = 4, a_{10} = 46, a_{01} = 4, a_{20} = 55.$$

Шуканий многочлен локаторів помилок набуде вигляду:

$$4 + 46x + 4y + 55x^2 + xy = 0.$$

Скористаємося процедурою Ченя, підставивши всі пари (x, y) , що відповідають проєктивним точкам $P(x, y, 1)$, у многочлен локаторів помилок. Пари, що відповідають проєктивним точкам, поданим у табл. 1.28, перетворюють у нуль многочлен локаторів помилок. Тим самим вони локалізують помилки, вказуючи на те, де вони розташовані.

Таблиця 1.28

**Проективні точки, що перетворюють у нуль
многочлен локаторів помилок**

	P1	P71	P114	P129	P160	P232	P289	P336	P337	P343	P408	P416	P417	P491	P508
X	0	46	25	7	23	17	42	49	56	29	36	30	37	45	38
Y	1	9	15	17	21	30	36	42	42	43	51	52	52	61	63
Z	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Підставивши знайдені локатори помилок у систему рівнянь (1.94) і розв'язавши її, одержимо:

$$e_1 = 32; e_{71} = 2; e_{114} = 0; e_{129} = 0; e_{160} = 0; e_{232} = 0; e_{289} = 0; e_{336} = 0; e_{337} = 0; e_{343} = 1; e_{408} = 0; e_{416} = 32; e_{417} = 0; e_{491} = 0; e_{508} = 57.$$

Вектор помилок знайдений правильно. Виправимо помилку:

$$c = c^* - e.$$

Таким чином, запропонований алгебраїчний алгоритм декодування алгеброгеометричних кодів полягає в розв'язанні систем лінійних рівнянь

і дозволяє ефективно декодувати кодові слова з поліноміальною функцією складності.

У роботах [5; 6] досліджені неалгебраїчні методи та практичні алгоритми неалгебраїчного декодування алгеброгеометричних кодів, які дозволяють використати мажоритарні та переставні процедури неалгебраїчного декодування недвійкових нециклічних блокових кодів, що дозволяє в деяких випадках без використання складних алгебраїчних перетворень реалізувати прості схеми декодування.

1.8.6. Переставне декодування алгеброгеометричних кодів

Основна ідея переставного декодування полягає у формуванні набору кандидатів у кодові слова і відбору серед цих кандидатів найближчого до кодового слова. Якщо при цьому в прийнятій послідовності немає помилок в одній з інформаційних компонент, передане кодове слово буде міститися в переліку кандидатів. Таким чином, якщо фактична комбінація помилок може бути виправлена декодером максимальної правдоподібності, то це кодове слово буде мати найменшу відстань до прийнятої послідовності й буде обране переставним декодером. Оскільки символи, які утворюють інформаційну множину, можна задати незалежно, то вони визначають кодове слово. Якщо у відповідних місцях немає помилок, то інші символи кодового слова можна відтворити. Ця властивість лежить в основі всіх переставних алгоритмів декодування. Загальний алгоритм декодування має такий вигляд [6].

Алгоритм переставного декодування у загальному вигляді.

1. Вибрати кілька різних інформаційних множин відповідно до деякого правила.

2. Побудувати кодове слово для кожної з множин, вважаючи, що символи інформаційної множини прийняті без помилок.

3. Порівняти кожне гіпотетичне слово із прийнятою послідовністю й вибрати найближче кодове слово (яке перебуває на найменшій відстані).

При переставному декодуванні комбінація помилок буде знайдена, якщо вдасться знайти таку інформаційну множину, що повністю включає цю комбінацію. Така множина, що є покривною комбінацією помилок, і набір перевірових множин, які покривають усі набори помилок даного типу, називаються покриттям. Завдання декодера полягає в тому, щоб знайти перевірову множину, що покриває невідому комбінацію помилок.

Алгоритм випадкового вибору для кодів був розроблений Омуруо та використовується для декодування двійкових кодів. Розрахунками доведена скінченна кількість операцій [5; 6]. Виконання алгоритму починається з випадково обраної перевіркової множини та відповідної перевіркової матриці.

На першому кроці обчислюється синдром і висувається припущення, за якого фактична комбінація помилок збігається із синдромом на перевіркових елементах і дорівнює 0 – на інформаційних. Наступний крок полягає в перетворенні перевіркової множини, а саме заміни одного з його елементів на елемент інформаційної множини. Заміна проводиться так, щоб вага нового синдрому не збільшувала існуючої ваги. Після цього розглядається новий розв'язок на новому синдромі. Таким чином, вага синдрому використовується як градієнтна функція при пошуку правильного розв'язку.

У роботі [6] запропонований модифікований алгоритм переставного декодування Омури, що дозволяє декодувати недвійкові коди (алгеброгеометричні коди в тому числі). Алгоритм декодування зображений на рис. 1.37 і складається з таких кроків.

Крок 1. Знаходиться перевіркова множина та перевіркова матриця, що відповідає цій множині. Для однозначного визначення початкових умов як перевіркову множину пропонується використати перевіркову матрицю H коду.

Крок 2. Розраховується синдром S_i за формулою

$$S_i = H_i \times c.$$

Синдром S_i має вигляд:

$$S_i^T = \{S_1, S_2, \dots, S_{n-k}\}.$$

Якщо в каналі при передачі виникла помилка:

$$e^T = \{0, 0, \dots, 0\},$$

то синдром буде мати вигляд:

$$S_i^T = \{0, 0, \dots, 0\}.$$

Крок 3. Порівнюється вага розрахованого синдрому S_i з t -здатністю виправного коду. Якщо вага синдрому W_{S_i} менша або дорівнює t , то вектор виниклої помилки буде мати вигляд:

$$e^T = \{0, 0, \dots, 0, S_1, S_2, \dots, S_{n-k}\},$$

і розв'язок знайдений.

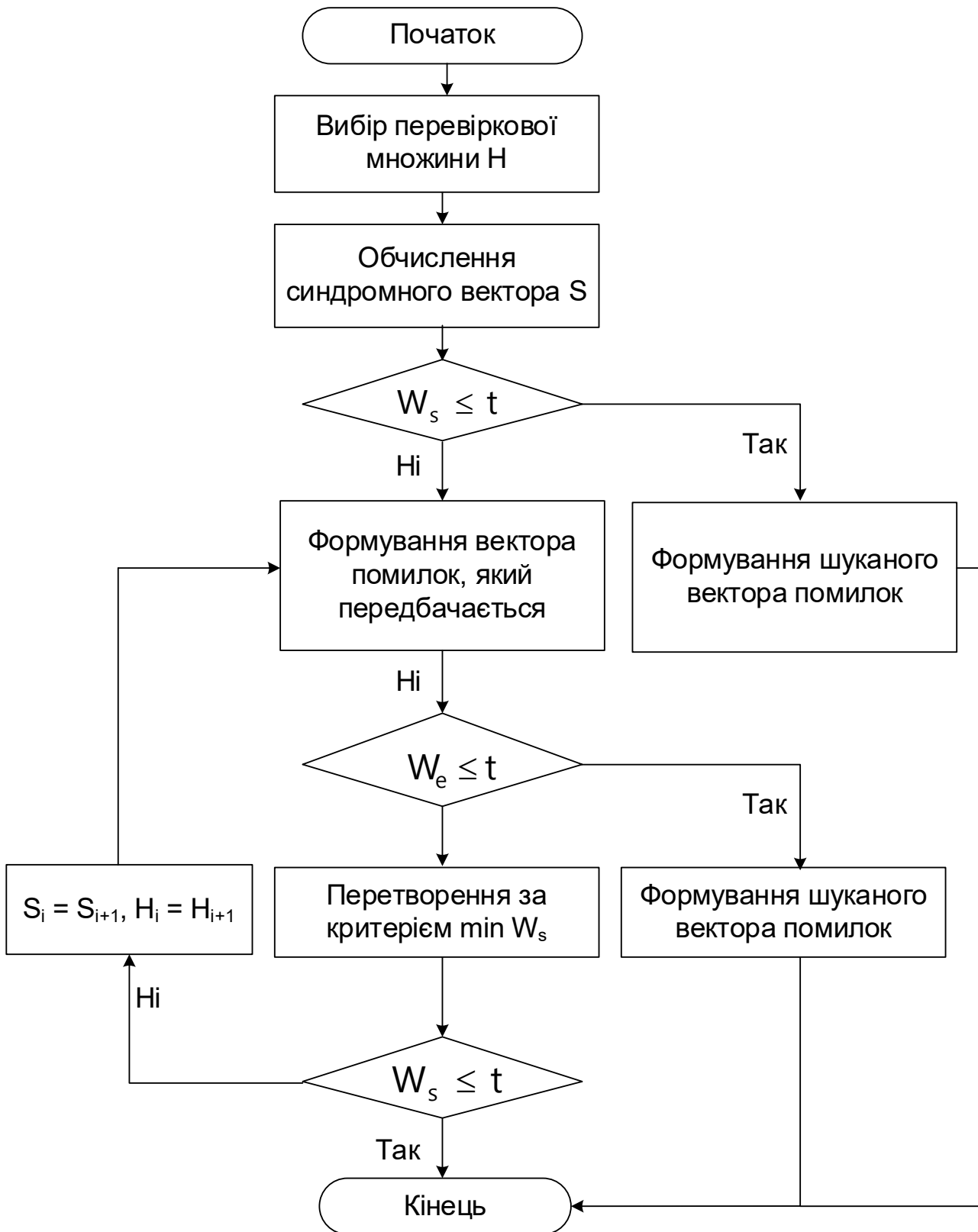


Рис. 1.37. Схема переставного алгоритму декодування

Крок 4. Якщо вага W_{S_i} більша від t , необхідно визначити, який з елементів інформаційної множини є спотвореним. Помилка в інформаційній множині буде одноразовою тоді, коли шляхом додавання стовпця h_i до інформаційної множини із синдромом S_i над полем $GF(q)$ вага помилки W_{e_i} зменшується. У цьому випадку вектор виниклої помилки буде мати вигляд:

$$e^T = \{0, 0, \dots, 1, \dots, 0, S_1, S_2, \dots, S_{n-k}\},$$

де 1 – кратність помилки.

Крок 5. Порівнюємо вагу помилки e_i і t . Якщо $W_{e_i} \leq t$, то розв'язок знайдений: вектор помилки дорівнює:

$$e^T = \{0, 0, \dots, 1, \dots, 0, S_1, S_2, \dots, S_{n-k}\}.$$

Крок 6. Якщо:

$$W_{e_i} \geq t,$$

тоді береться елемент інформаційної множини й переставляється з елементом перевіркової множини шляхом виконання елементарних операцій з рядками матриці. Перестановка можлива у випадку присутності ненульових символів одного рядка.

За критерієм мінімізації $\min W_{S_i}$ ураховується вага синдрому розрахованої множини, тобто якщо:

$$W_{S_i} < W_{S_{i+1}},$$

то така множина приймається до подальшого розгляду. Формується матриця H_{i+1} .

Крок 7. Розраховується синдром:

$$S_{i+1} = H_{i+1} \times c.$$

Синдром S_{i+1} буде мати вигляд:

$$S_{i+1}^T = \{S_1, S_2, \dots, S_{n-k}\}.$$

Крок 8. Якщо

$$W_{S_{i+1}} \leq t,$$

то вектор виниклої помилки буде мати вигляд:

$$e^T = \{0, 0, \dots, 0, S_1, S_2, \dots, S_{n-k}\}.$$

Якщо

$$W_{Si+1} > t,$$

то здійснюється перехід до кроку 4 за умови:

$$S_i = S_{i+1}, H_i = H_{i+1}.$$

1.8.7. Мажоритарне декодування алгеброгеометричних кодів

Мажоритарне декодування є одним із найбільш простих і зручних у реалізації методів декодування алгебраїчних блокових кодів. Практична реалізація таких декодерів приваблює високою швидкістю та низькою складністю [5; 6].

Суть мажоритарного декодування полягає в мажоритарному оцінюванні ваги помилки, що відбулася в i -му символі кодового слова. Подібна оцінка проводиться над ортогональними відносно i -го кодового символу перевірковими рівняннями. Ортогональна щодо i -ї координати підмножина перевіркових рівнянь складається з усіх рівнянь, до кожного з яких входить i -й елемент, а інші елементи входять не більш ніж до одного рівняння. Для циклічних кодів така підмножина збігається для всіх координат, декодування полягає в оцінюванні ваги помилки, що відбулася в i -му символі циклічно зсуненого кодового слова. Якщо структура коду така, що множина перевіркових рівнянь ортогональна щодо декількох координат, то мажоритарне вирішення застосовне для локалізації помилки в підмножині цих компонентів.

Алгеброгеометричні коди у загальному вигляді не мають циклічної структури, ортогональні підмножини щодо кожної координати різні. Застосування стандартного алгоритму мажоритарного декодування для нециклічних кодів вимагає виконання $2t$ ортогональних перевірок для кожної з n компонент (n, k, d) -коду [5; 6]. Розробка таких декодерів ускладнюється також пошуком ортогональних підмножин (систем перевірних рівнянь) для кожної з n координат. У роботі [5; 6] запропонований алгоритм формування таких підмножин з верхньою оцінкою потужності множини ортогональних рівнянь, що дозволяють здійснити ортогоналізацію нециклічного (n, k, d) -коду в один крок (якщо така ортогоналізація можлива).

Алгоритм формування системи перевірних рівнянь [5; 6] полягає в такому. Нехай заданий алгебраїчний блоковий нециклічний (n, k, d) -код,

t – кількість помилок, що виправляють при мажоритарному декодуванні коду в один крок, $t \leq (n - 1)/(2d_{\perp} - 2)$, $(n, n - k, d_{\perp})$ – дуальний код.

Запропонований алгоритм формування системи перевірних рівнянь складається з таких кроків.

1. За допомогою елементарних перетворень над перевірною частиною коду знайдемо $2t$ перевірних рівнянь, ортогональних за першою координатою. Кожне з отриманих рівнянь містить як мінімум $d_{\perp}$ складових і відповідно може бути використане для ортогоналізації за $d_{\perp} - 1$ іншими координатами.

2. Знайдемо $2t$ перевірочних рівнянь, ортогональних за другою координатою. На цьому кроці використаємо вже знайдені рівняння, до складу яких входить доданок із другим компонентом. Інші рівняння знаходимо шляхом елементарних перетворень над перевірною частиною коду.

3. Повторивши крок 2 для кожної координати, одержимо систему перевіркових рівнянь, що дозволяють здійснити ортогоналізацію (n, k, d) -коду в один крок.

Теорема 1.25. [5; 6]. Мінімальний набір рівнянь для однокрокової ортогоналізації нециклічного (n, k, d) -коду містить не більше $2tn/d_{\perp}$ рівнянь, де t – кількість помилок, які виправляє однокроковий мажоритарний декодер, $d_{\perp}$ – мінімальна відстань дуального коду.

Доведення. Для мажоритарного декодування (n, k, d) -коду в один крок за визначенням необхідно виконати $2tn$ перевірок. Нехай M – кількість рівнянь, отриманих у результаті виконання запропонованого алгоритму. Кожне з перевірних рівнянь використовується як мінімум в $d_{\perp}$ перевірках, отже:

$$Md_{\perp} \leq 2tn.$$

Рівність виконується завжди, якщо дуальний код еквідистантний. У загальному випадку отримана в результаті виконання запропонованого алгоритму система рівнянь містить $M \leq 2tn/d_{\perp}$ рівнянь.

Якщо нециклічний (n, k, d) -код допускає однокрокову ортогоналізацію, застосування результату останнього твердження дозволяє спростити процедуру однокрокового мажоритарного декодування. Досить записати систему $M \leq 2tn/d_{\perp}$ перевірних рівнянь у вигляді матриці коефіцієнтів рівнянь розмірності $n \times M$ і здійснювати декодування у вигляді послідовності таких кроків [6].

Крок 1. Систему з $M \leq 2tn/d_{\perp}$ перевірних рівнянь запишемо у вигляді матриці $n \times M$, значення елементів матриці – коефіцієнти перевірних рівнянь. Кожен стовпчик матриці містить $2t$ ненульових елементів. Рядки з ненульовими елементами в i -му стовпці відповідають перевірним рівнянням, ортогональним за i -ю координатою.

Крок 2. Виконуємо M перевірок. Мажоритарно оцінюємо вагу помилки в кожному символі. Локалізуємо помилку.

Крок 3. Обчислюємо кратність помилки в локалізованих помилкових символах. Використовуємо ті перевірні рівняння, до яких входить тільки один з помилкових компонентів.

Приклад. Розглянемо алгеброгеометричний код $(9, 3, 6)$ за кривою Ерміта над $GF(4)$.

Породжувальна й перевірна матриці коду в канонічному вигляді відповідно такі:

$$G = \begin{pmatrix} 1 & 0 & 0 & 2 & 3 & 1 & 2 & 3 & 0 \\ 0 & 1 & 0 & 0 & 3 & 2 & 2 & 1 & 2 \\ 0 & 0 & 1 & 2 & 0 & 3 & 1 & 3 & 3 \end{pmatrix},$$

$$H = \begin{pmatrix} 2 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 0 & 0 & 1 & 0 & 0 & 0 \\ 2 & 2 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 3 & 1 & 3 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Кількість помилок, які код може виправити, $t = 2$. Дуальний до розглянутого коду – код з параметрами $(9, 6, 3)$. Сформуємо систему перевіркових рівнянь відповідно до запропонованого алгоритму. Усього необхідно виконати 36 перевірок.

Для розглянутого прикладу мінімальна кількість перевірних рівнянь відповідно до доведеного твердження $M = 36/3 = 12$.

У табл. 1.29 наведена матриця коефіцієнтів перевіркових рівнянь.

Кожен стовпчик матриці містить $2t = 4$ ненульові елементи. Рядки з ненульовими елементами в i -му стовпчику відповідають перевірковим рівнянням, ортогональним за i -ю координатою.

Нехай c – кодове слово розглянутого $(9, 3, 6)$ коду,

$$c = \{102130021\}.$$

Нехай e – вектор помилок, $e = \{300000100\}$.

Прийнята з помилками кодова послідовність

$$c^* = \{202130121\}.$$

Таблиця 1.29

**Матриця коефіцієнтів перевірних рівнянь, яка дозволяє
мажоритарно декодувати алгеброгеометричний код (9, 3, 6)**

1	1	0	0	2	0	0	0	0
1	0	1	3	0	0	0	0	0
1	0	0	0	0	1	0	0	1
1	0	0	0	0	0	2	3	0
0	1	2	0	0	0	0	0	3
0	1	0	1	0	0	0	1	0
0	1	0	0	0	2	1	0	0
0	0	1	0	3	0	1	0	0
0	0	1	0	0	3	0	1	0
0	0	0	1	1	2	0	0	0
0	0	0	1	0	0	1	0	1
0	0	0	0	1	0	0	1	1

Для першого елемента кодової послідовності коефіцієнти ортогональних рівнянь записані в перших чотирьох рядках табл. 1.29. Для другого компонента коефіцієнти рівнянь записані в першому, п'ятому, шостому і сьомому рядках, і т.д. для всіх компонентів. Арифметичні операції виконуємо над скінченим полем із чотирьох елементів.

Результат ортогональних перевірок для зручності запишемо у вигляді матриці $n \times m$ (табл. 1.30). Значення середини табл. 1.30 заповнюємо символами істинності: "True" – якщо результат перевірки позитивний, "False" – у протилежному випадку. Значення "0" відповідає нульовому коефіцієнту в перевіркового рівнянні (у перевірці не бере участь). Останній рядок табл. 1.30 містить результат мажоритарного оцінювання ваги помилки в кожному елементі кодової послідовності (мажоритарний поріг – " ≥ 2 "). Результат мажоритарного оцінювання дозволив локалізувати помилку – помилка відбулася в першому й сьомому символі. Виправити помилку можна за допомогою будь-якого перевіркового рівняння, до якого входить тільки один із помилкових елементів.

**Результат виконання ортогональних перевірок при декодуванні
алгеброгеометричного коду (9, 3, 6) в один крок**

False	False	0	0	False	0	0	0	0
False	0	False	False	0	0	0	0	0
False	0	0	0	0	False	0	0	False
True	0	0	0	0	0	True	True	0
0	True	True	0	0	0	0	0	True
0	True	0	True	0	0	0	True	0
0	False	0	0	0	False	False	0	0
0	0	False	0	False	0	False	0	0
0	0	True	0	0	True	0	True	0
0	0	0	True	True	True	0	0	0
0	0	0	False	0	0	False	0	False
0	0	0	0	True	0	0	True	True
False	True	True	True	True	True	False	True	True

Для першого символу – це перше, друге або третє рівняння, для сьомого символу – це сьоме, восьме або одинадцяте рівняння.

У результаті одержимо:

$$c_1 = c_2 + 2 \times c_5 = 0 + 2 \times 3 = 1; \quad c_1 = c_3 + 3 \times c_4 = 2 + 3 \times 1 = 1;$$

$$c_1 = c_6 + c_9 = 0 + 1 = 1; \quad c_7 = c_2 + 2 \times c_6 = 0 + 2 \times 0 = 0;$$

$$c_7 = c_3 + 3 \times c_5 = 2 + 3 \times 3 = 0; \quad c_7 = c_4 + c_9 = 1 + 1 = 0.$$

Помилка локалізована й виправлена.

Алгоритм формування системи перевірних рівнянь дозволяє одержати мінімальний набір ортогональних перевірок. Використання мінімального набору перевірок дозволяє істотно скоротити обсяг запам'ятовувального пристрою для зберігання коефіцієнтів ортогональних рівнянь мажоритарного декодера нециклічного коду. Так, у розглянутому прикладі використання набору з 12 перевірних рівнянь (замість 36) дозволяє скоротити в три рази обсяг необхідної пам'яті (середини матриці коефіцієнтів рівнянь). На розглянутому прикладі продемонстрована конструктивність мажоритарного декодування алгеброгеометричних кодів.

1.8.8. Згортувальні коди

При використанні для захисту від помилок блокових кодів інформаційна послідовність розбивається на окремі блоки, що кодуються незалежно один від одного. Таким чином, закодована послідовність стає послідовністю незалежних слів однакової довжини.

При використанні згортувальних кодів потік даних розбивається на набагато менші блоки довжиною k символів (в окремому випадку $k_0 = 1$), які називаються кадрами інформаційних символів.

Кадри інформаційних символів кодуються кадрами кодових символів довжиною n_0 символів. При цьому кодування кадру інформаційних символів у кадр кодового слова здійснюється з урахуванням попередніх m кадрів інформаційних символів. Процедура кодування, таким чином, зв'язує між собою послідовні кадри кодових слів. Передана послідовність стає одним напівнескінченим кодовим словом.

Основними характеристиками згортувальних кодів є такі величини:

- k_0 – розмір кадру інформаційних символів;
- n_0 – розмір кадру кодових символів;
- m – довжина пам'яті коду;
- $k = (m + 1)k_0$ – інформаційна довжина слова;
- $n = (m + 1)n_0$ – кодова довжина блока.

Кодова довжина блока – це довжина кодової послідовності, на якій зберігається вплив одного кадру інформаційних символів.

Нарешті, згортувальний код має ще один важливий параметр – швидкість $R = k / n$, що характеризує ступінь надмірності коду, яка вводиться для забезпечення виправляючих властивостей коду.

Як і блокові, згортувальні коди можуть бути систематичними і несистематичними й визначаються як лінійні згортувальні (n, k) коди.

Систематичним згортувальним кодом є такий код, для якого у вихідній послідовності кодових символів утримується без зміни послідовність інформаційних символів. У протилежному випадку згортувальний код є несистематичним.

Згортувальні коди мають дуже просту схему кодування. Інформаційні символи послідовно надходять у регістр зсуву, при цьому кодові символи являють собою суму за модулем 2 вмісту чарунок пам'яті регістра зсуву й вхідних символів. Приклади схем кодерів для систематичного $(8, 4)$ й несистематичного $(6, 3)$ згортувальних кодів зображені на рис. 1.38.

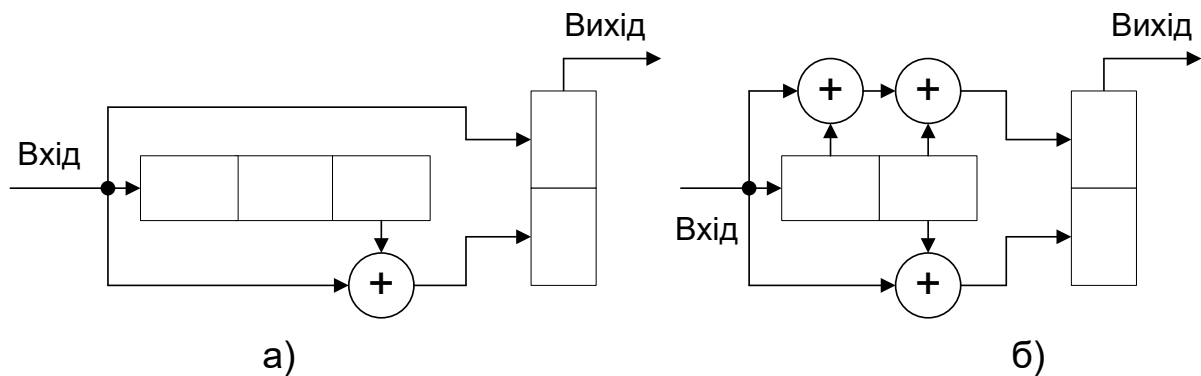


Рис. 1.38. Приклади схем кодерів для (8, 4) систематичного (а) і (6, 3) несистематичного (б) згортувальних кодів

Можливі різні способи опису згортувальних кодів, наприклад, за допомогою породжувальної матриці. Правда, із-за нескінченності послідовності, що кодується, й породжувальна матриця буде мати нескінченні розміри.

Більш зручним способом опису згортувального коду є його завдання за допомогою імпульсної перехідної характеристики еквівалентного фільтра Хафмена (лінійної послідовної схеми або регістра зсуву зі зворотними зв'язками) або відповідного йому породжувального многочлена.

Імпульсна перехідна характеристика фільтра (а кодер згортувального коду, по суті є фільтром) – це реакція на одиничний вплив вигляду $\bar{\delta} = 100000 \dots$. Для кодерів, що зображені на рис. 1.38, імпульсні характеристики будуть мати вигляд

$$h_1 = 1100\ 00\ 0100\ 00;$$

$$h_2 = 1110\ 1100\ 00\ 00.$$

Ще одна форма завдання згортувальних кодів – це використання породжувальних многочленів, однозначно пов'язаних з імпульсною перехідною характеристикою еквівалентного фільтра,

$$h_1(x) = 1 + x + x^7;$$

$$h_2(x) = 1 + x + x^2 + x^4 + x^5.$$

При цьому кодова послідовність U на виході згортувального кодера утворюється у результаті згортки вхідної інформаційної послідовності m з імпульсною перехідною характеристикою h .

Розглянемо приклади кодування послідовностей з використанням імпульсної характеристики еквівалентного фільтра.

Нехай $m = 110 \dots$, тоді для кодера з імпульсною перехідною характеристикою $h = 1100\ 00\ 0100\ 00 \dots$:

$$\begin{array}{r} U = 11\ 00\ 00\ 01\ 00\ 00\dots \\ \quad 11\ 00\ 00\ 01\ 00\dots \\ \hline U = 11\ 11\ 00\ 01\ 01\ 00\ 00\dots \end{array}$$

або $m = 1011000 \dots$:

$$\begin{array}{r} U = 11\ 00\ 00\ 01\ 00\ 00\ 00\dots \\ \quad 11\ 00\ 00\ 01\ 00\dots \\ \quad 11\ 00\ 00\ 01\dots \\ \hline U = 11\ 00\ 11\ 10\ 00\ 01\ 01\ 00\dots \end{array}$$

Іноді зручніше розглядати повний породжувальний многочлен $G(x)$ згортувального коду як сукупність декількох многочленів менших ступенів, що відповідають чарункам вихідного регістра кодера. Так, для кодера на рис. 1.38б відповідні часткові породжувальні многочлени будуть такими:

$$G_1(x) = 1 + x + x^2;$$

$$G_2(x) = 1 + x^2.$$

Нехай, наприклад, кодується послідовність $m = 1010 \dots$. Відповідний інформаційний многочлен запишеться як $m(x) = 1 + x^2$.

Тоді на вході першої чарунки вихідного регістра при кодуванні буде послідовність $U_1 = 11011000 \dots$, якій відповідає многочлен $U_1(x) = 1 + x + x^3 + x^4$, а на вході другої чарунки буде послідовність $U_2 = 10001000 \dots$ і, відповідно, многочлен $U_2(x) = 1 + x^4$.

Легко помітити, що при цьому справедливі рівності

$$U_1(x) = m(x)G_1(x);$$

$$U_2(x) = m(x)G_2(x).$$

Така форма запису зручна, оскільки видно структуру пристрою, що кодує (за набором многочленів можна відразу синтезувати пристрій). На

практиці многочлени задаються набором своїх коефіцієнтів. У табл. 1.31 наведені приклади кодерів згортувальних кодів, заданих своїми коефіцієнтами при ступенях породжувальних многочленів.

Таблиця 1.31.

Коефіцієнти породжувальних многочленів деяких згортувальних кодів

G_1	G_2
111	101
1111	1101
11101	10011
111011	10100111
110001	11111001

Як приклад кодера з $k_0 \neq 1$ можна привести кодер згортувального (12, 9) коду Вайнера–Еша з параметрами: $k_0 = 3$, $n_0 = 4$, $k = 9$, $n = 12$, $R = 3/4$, що зображений на рис. 1.39.

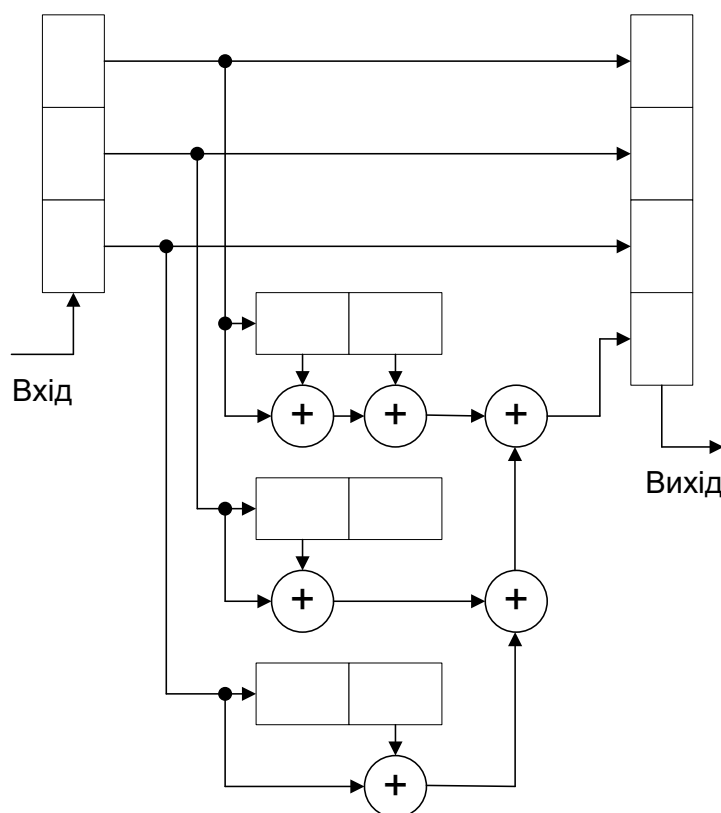


Рис. 1.39. Схема кодера згортувального (12, 9) коду Вайнера–Еша

Нехай, наприклад, $m = 100\,000\,000 \dots$. Тоді кодове слово $U = 1001000100010000$. Видно, що це – систематичний код, у якому до трьох інформаційних символів додається один перевірочний, залежний від значень інформаційних символів не тільки поточного кадру, але й двох попередніх кадрів. При цьому вплив одного кадру інформаційних символів поширюється на 12 символів кодової послідовності, тобто кодова довжина блоку для цього коду $n = 12$.

Ще одним способом опису й ілюстрації роботи згортувальних кодів є використання так званого кодового дерева.

Розглянемо згортувальний $(6, 3)$ код зі схемою, зображеною на рис. 1.38б. Відповідне цьому кодеру кодове дерево – послідовність вихідних станів кодера при подачі на його вхід ланцюжка вхідних символів 0 і 1 – зображене на рис. 1.40.

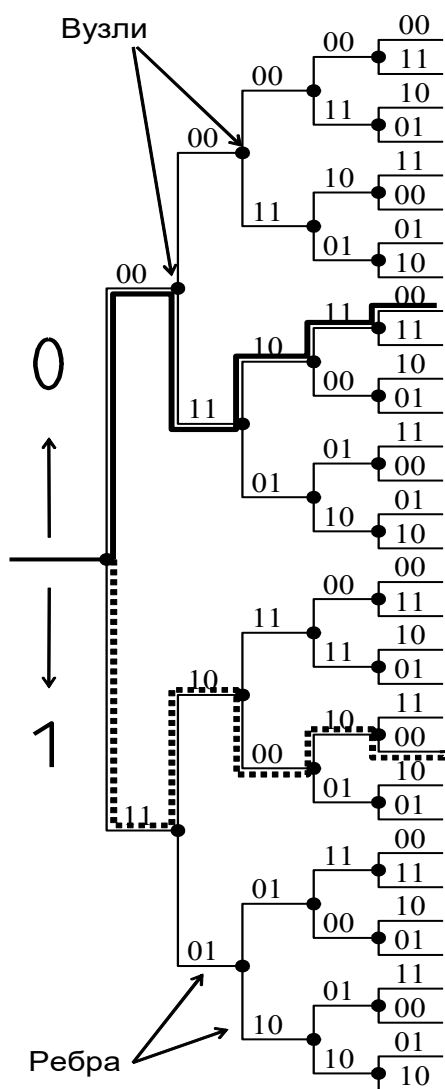


Рис. 1.40. Кодове дерево згортувального $(6, 3)$ коду

На діаграмі рис. 1.40 зображені вхідні та вихідні послідовності кодера: вхідні - напрямком руху по дереву (нагору – 0, униз – 1), вихідні - символами уздовж ребер дерева. При цьому кодування складається в прямуванні вправо по кодовому дереву.

Наприклад, вхідна послідовність $m = 01000 \dots$ кодується як $U = 0011101100000 \dots$, послідовність $m = 1010100000 \dots$ – як $U = 1110001000 \dots$.

Якщо уважно подивитися на будову зображеного на рис. 1.40 кодового дерева, можна помітити, що починаючи з четвертого ребра його структура повторюється, тобто яким би не був перший крок, починаючи з четвертого ребра значення вихідних символів кодера повторюються. Однакові ж вузли можуть бути об'єднані, і тоді починаючи з деякого перетину розмір кодового дерева перестане збільшуватися.

Інакше кажучи, вихідна кодова послідовність у певний момент перестає залежати від значень вхідних символів, введених у кодер раніше.

Дійсно, з рис. 1.40 видно, що, коли третій вхідний символ вводиться у кодер, перший вхідний символ залишає регістр зсуву і не зможе надалі впливати на вихідні символи кодера.

З урахуванням цього необмежене кодове дерево, зображене на рис. 1.40, переходить в обмежену гратчасту діаграму (кодове дерево з вузлами, що зливаються), зображену на рис. 1.41.

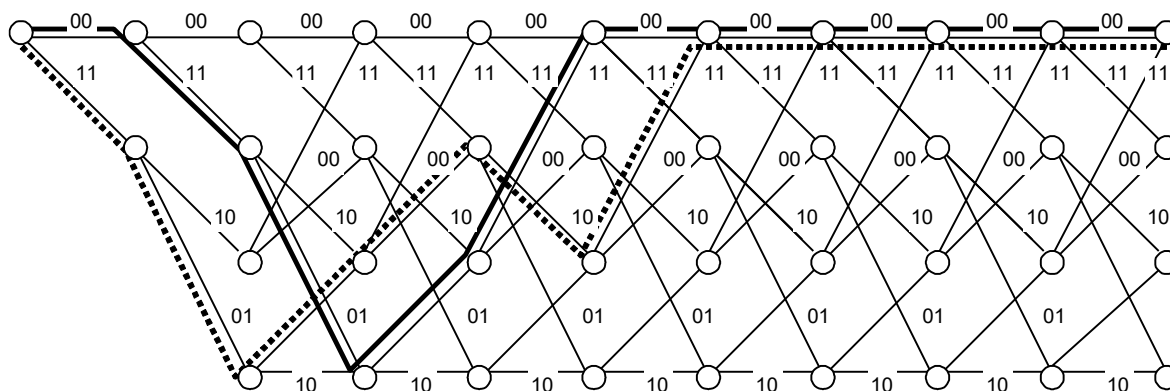


Рис. 1.41. Гратчаста діаграма згортувального (6, 3) коду

Кодування згортувальним кодами з використанням гратчастої діаграми, як і у випадку з кодовим деревом, являє собою надзвичайно просту процедуру: чергові символи вхідної послідовності визначають напрямок руху з вузлів грат: якщо 0, то йдемо по верхньому ребру, якщо 1 – по нижньому ребру. Однак, на відміну від кодового дерева гратчаста

діаграма не розростається по ширині з кожним кроком, а має, починаючи з третього перетину, постійну ширину.

Як приклад, закодуємо за допомогою гратчастої діаграми кілька інформаційних послідовностей.

Нехай $m = 0110000 \dots$. Відповідна їй кодова послідовність буде мати вигляд

$$U = 001101011100 \dots,$$

або $m = 110100 \dots$, тоді

$$U = 1101010010110000 \dots$$

1.8.9. Декодування згортувальних кодів. Алгоритм Витербі

Найкращою схемою декодування коригувальних кодів, як ми вже відзначали, є декодування методом максимальної правдоподібності.

Для двійкового симетричного каналу без пам'яті (каналу, у якому ймовірності передачі 0 і 1, а також ймовірності помилок виду $0 \rightarrow 1$ і $1 \rightarrow 0$ однакові, помилки в j -му і i -му символах коду незалежні) декодер максимальної правдоподібності зводиться до декодера мінімальної відстані Хеммінга. Останній обчислює відстань Хеммінга між прийнятою послідовністю r і всіма можливими кодовими векторами U_i та виносить рішення на користь того вектора, що виявляється ближче до прийнятого.

Очевидно, що в загальному випадку такий декодер виявляється дуже складним і при великих розмірах кодів n і k практично нереалізованим. Характерна структура згортувальних кодів (повторюваність структури за межами вікна довжиною n) дозволяє створити цілком прийнятний за складністю декодер максимальної правдоподібності. Уперше ідея такого декодера була запропонована Витербі. Працює він так.

Припустимо, на вхід декодера надійшов сегмент послідовності r довжиною b , що перевищує кодову довжину блока n . Назвемо b вікном декодування. Порівняємо всі кодові слова цього коду (у межах сегмента довжиною b) з прийнятим словом і виберемо кодове слово, найближче до прийнятого. Перший інформаційний кадр обраного кодового слова приймемо як оцінку інформаційного кадру декодованого слова.

Після цього у декодер вводиться n_0 нових символів, а введені раніше самі старі n_0 символів скидаються, і процес повторюється для визначення наступного інформаційного кадру.

Таким чином, декодер Витербі послідовно обробляє кадр за кадром, рухаючись по ґратах, аналогічних тим, що використовуються кодером. У кожний момент часу декодер не знає, у якому вузлі перебуває кодер, і не намагається його декодувати. Замість цього декодер за прийнятою послідовністю визначає найбільш правдоподібний шлях до кожного вузла та відстань між кожним таким шляхом і прийнятою послідовністю. Ця відстань називається мірою розходження шляху. За оцінку прийнятої послідовності вибирається сегмент, що має найменшу міру розходження. Шлях з найменшою мірою розходження називається шляхом, що вижив.

Розглянемо роботу декодера Витербі на простому *прикладі*. Нехай кодування здійснюється з використанням згортувального (6, 3) коду (схема кодера зображена на рис. 1.38б, ґратчаста діаграма, що відповідає цьому кодеру, – на рис. 1.41).

Користуючись ґратчастою діаграмою кодера, спробуємо, прийнявши деякий сегмент r , простежити найбільш імовірний шлях кодера. При цьому для кожного перетину ґратчастої діаграми будемо відзначати міру розходження шляху до кожного її вузла.

Припустимо, що передано кодову послідовність $U = 0000000 \dots$, а прийнята послідовність має вигляд $r = 10001000 \ 00 \dots$, тобто в першому й у третьому кадрах кодового слова виникли помилки. Як ми вже переконалися, процедура та результат декодування не залежать від переданого кодового слова і визначаються тільки помилкою, що міститься у прийнятій послідовності. Тому найпростіше вважати, що передано нульову послідовність, тобто $U = 0000000 \dots$.

Прийнявши першу пару символів (10), визначимо міру розходження для першого перетину ґрат (див. рис. 1.41), прийнявши наступну пару символів (00), – для другого перетину й т.д. При цьому із вхідних у кожний вузол шляхів залишаємо шлях з меншим розходженням, оскільки шлях з більшим на цей момент розходженням вже не зможе стати надалі коротшим.

Помітимо, що для розглянутого прикладу, починаючи з четвертого рівня метрика (або міра розходження) нульового шляху менша будь-якої іншої метрики. Оскільки помилок у каналі більше не було, видно, що, зрештою як відповідь, буде обраний саме цей шлях. Із цього прикладу також видно, що шляхи, що вижили, можуть досить довго відрізнятися один від одного.

Однак, на шостому – сьомому рівні перші сім ребер всіх шляхів, що вижили, збігаються один з одним. У цей момент відповідно до алгоритму Витербі й приймається рішення про передані символи, тому що всі шляхи, що вижили, виходять із однієї вершини, тобто відповідають одному інформаційному символу.

Процедура декодування послідовності, що має дві помилки, ілюструється послідовністю, зображеною на рис. 1.42.

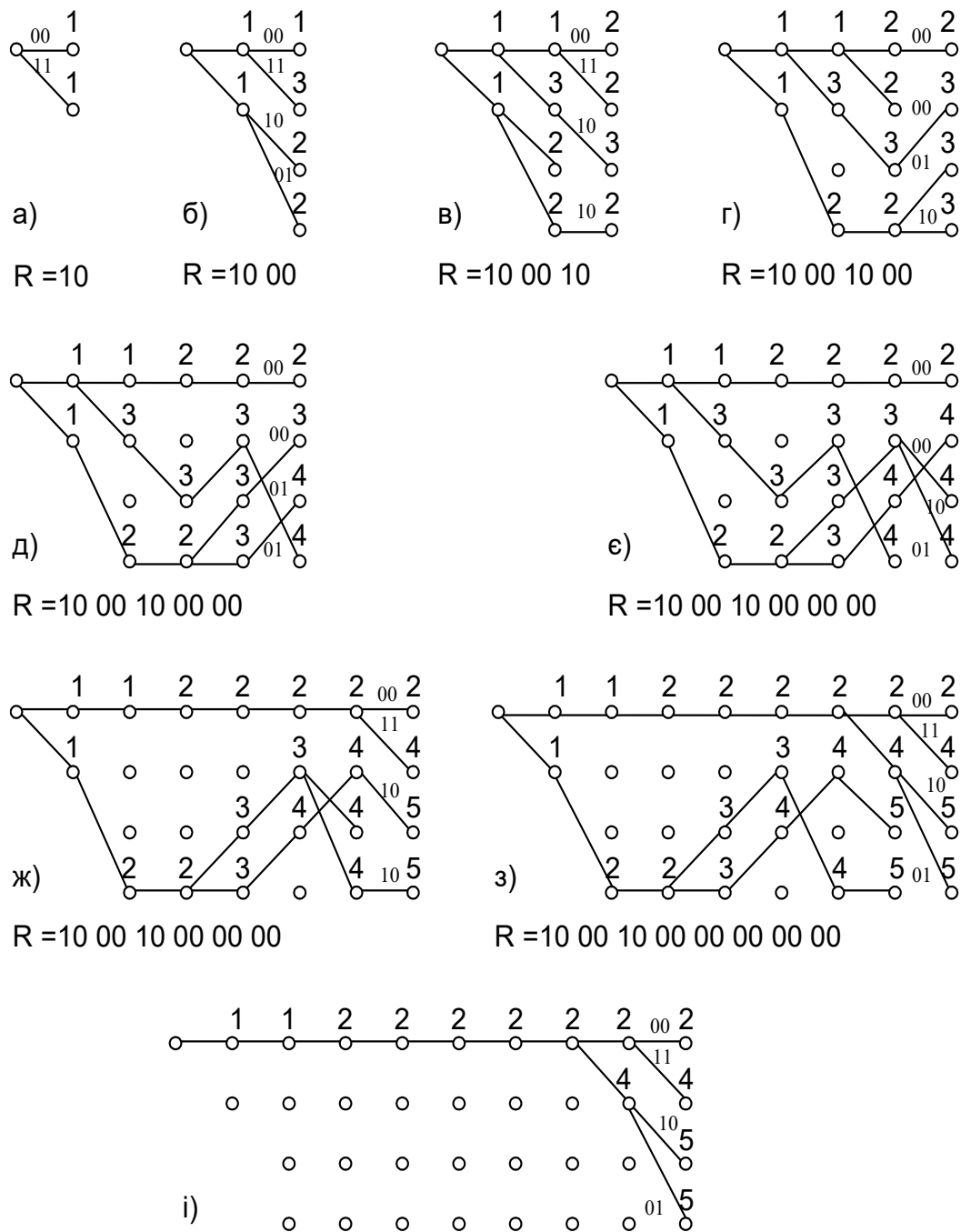


Рис. 1.42. Декодування декодером Витербі послідовності з двома помилками

Глибина, на якій відбувається злиття шляхів, що вижили, не може бути обчислена заздалегідь. Вона є випадковою величиною, що залежить від кратності й імовірності виникаючих у каналі помилок. Тому на практиці звичайно не чекають злиття шляхів, а встановлюють фіксовану глибину декодування.

З рис. 1.42, є видно, що ступінь розходження метрик правильного і неправильного шляхів досить велика ($d_{\text{пр}} = 2$, $d_{\text{пом}} = 4$), тобто в цьому випадку можна було б обмежити глибину декодування величиною $b \leq 6$. На практиці глибину декодування звичайно вибирають у діапазоні $n \leq b \leq n + l$, де l – кількість помилок, що виправляється цим кодом.

З рис. 1.42 також видно, що, незважаючи на наявність у прийнятому фрагменті двох помилок, його декодування відбулося без помилки і як відповідь буде прийнята передана нульова послідовність.

Характеристики згортувальних кодів, як і будь-яких інших кодів, поліпшуються зі збільшенням їхнього розміру, у цьому випадку – кодової довжини блока n . При цьому, однак, декодер Витербі стає нереалізовано складним. Так, при $n = 10$ він повинен пам'ятати вже не менш $2^{10} = 1024$ шляхів, що виживуть.

Для зменшення складності декодера максимальної правдоподібності при великих n була розроблена стратегія, що ігнорує малоймовірні шляхи пошуку по ґратах. Однак рішення про те, щоб остаточно відкинути цей шлях, не приймається. Час від часу декодер повертається назад і продовжує залишений шлях.

Подібні стратегії пошуку найбільш імовірного шляху по ґратах відомі під загальною назвою послідовного декодування. На відміну від оптимальної процедури Витербі послідовний декодер, переглянувши перший кадр, переходить у черговий вузол решітки з найменшим на цей момент розходженням. Із цього вузла він аналізує наступний кадр, вибираючи ребро, найближче до цього кадра, і переходить у наступний вузол, і так далі.

При відсутності помилок ця процедура працює дуже добре, однак при виникненні помилок на якому-небудь кроці декодер може випадково вибрати неправильну гілку. Якщо він продовжить просуватися по неправильному шляху, то незабаром виявить, що відбувається занадто багато помилок, і розходження шляху почне швидко наростати. Але це будуть помилки декодера, а не каналу. Тому декодер повертається на кілька кадрів назад і починає аналізувати інші шляхи, поки не знайде

найбільш правдоподібний. Потім він буде просуватися уздовж цього нового шляху.

Загальним недоліком методу послідовного декодування є велика кількість повернень до попередніх вузлів, що збільшує час пошуку найбільш правдоподібного шляху. Вхідна послідовність при цьому для уникнення втрати даних повинна зберігатися в буферному регістрі. Через обмежений об'єм пам'яті регістра можливо його переповнення й відмова від декодування.

1.9. Багаторазова передача кодових комбінацій

Підвищення достовірності в системах з безнадмірним кодуванням можна забезпечити шляхом багаторазової передачі однієї й тієї ж кодової комбінації. На приймальній стороні здійснюється порівняння або за символами кодових комбінацій, або за самими кодовими комбінаціями (посимвольна перевірка та групова перевірка) і рішення застосовується на користь таких символів або кодових комбінацій, які мають найбільшу кількість збігів (критерій більшості або мажоритарний принцип). Для того, щоб виключити невизначеність при прийманні, повідомлення в таких системах повинне передаватися не менше трьох разів. Дійсно, нехай по каналу передається кодова комбінація, що відповідає букві А. Якщо при першому прийманні ця буква прийнята правильно, а при другому прийомі замість букви А прийнята, наприклад, буква М, то виникає невизначеність і вибрати букву на підставі критерію більшості не вважається можливим. Якщо ж буква А буде передана третій раз і прийнята правильно, то на підставі критерію більшості вона й вибирається.

Безумовно, при сильних завадах не виключена можливість появи невизначеності і при трьох передачах, якщо щоразу кодові комбінації будуть декодуватися як різні букви (наприклад А, М або Б). У цьому випадку невизначеність при триразовій передачі можна виключити шляхом п'ятиразової передачі.

Розглянемо методику визначення ймовірності помилкового декодування кодової комбінації при її багаторазовому повторенні.

Припустимо, що n -символьна комбінація передається три рази. На приймальній стороні здійснюється порівняння за комбінаціями (групова перевірка). Правильне рішення при триразовому повторенні виходить тоді, коли кодова комбінація всі три рази прийнята без помилок або коли вона прийнята без помилок два рази. Ймовірність першої події дорівнює

$(1 - P_{\text{пом}})^{3n}$. Імовірність другої події дорівнює $3(1 - P_{\text{пом}})^{2n}[1 - (P_{\text{пом}})^n]$, де 3 – кількість варіантів, коли із трьох кодових комбінацій одна прийнята з помилкою; $(1 - P_{\text{пом}})^{2n}$ – імовірність правильного прийняття двох кодових комбінацій; $[1 - (P_{\text{пом}})^n]$ – імовірність помилкового прийняття однієї кодової комбінації.

Таким чином, імовірність того, що кодова комбінація при триразовому повторенні й груповій перевірці буде прийнята з помилкою, може бути знайдена за формулою:

$$P_{\text{пом}}(3) = 1 - (1 - P_{\text{пом}})^{3n} - 3(1 - P_{\text{пом}})^{2n}[1 - (P_{\text{пом}})^n].$$

У системах з посимвольною перевіркою здійснюється порівняння прийнятих комбінацій по символах. Значення символу (1 або 0) визначається по більшості. Так само, як і при груповому підсумовуванні, кількість передач повинна бути непарною.

При триразовій передачі кодовий символ буде зафіксований правильно, якщо він прийнятий без помилки всі три рази або двічі прийнятий без помилки, а один раз з помилкою. З урахуванням цього ймовірність правильного прийняття символу дорівнює $(1 - P_{\text{пом}})^3 + 3P_{\text{пом}}(1 - P_{\text{пом}})^2$. Отже, імовірність правильного прийому кодової комбінації, утвореної з n трьохсимвольних груп буде дорівнювати

$$P_{\text{пр}}(3) = \left[(1 - P_{\text{пом}})^3 + 3P_{\text{пом}}(1 - P_{\text{пом}})^2 \right]^3,$$

а ймовірність помилкового декодування дорівнює

$$P_{\text{пом}}(3) = 1 - \left[P_{\text{пом}}^3 + 3P_{\text{пом}}(1 - P_{\text{пом}})^2 \right]^3.$$

Аналіз отриманих формул показує, що система з посимвольною перевіркою більш ефективна, ніж система із груповою перевіркою.

Поряд з багаторазовою передачею комбінацій первинного коду, для підвищення вірності застосовується й багаторазова передача комбінацій завадостійкого коду. Один з можливих варіантів такої системи функціонує таким способом. Кодова комбінація повторюється кілька разів. При кожному прийнятті здійснюється декодування з метою виявлення помилок. Якщо при першому прийнятті помилки не виявлено, що

відповідає правильній передачі або наявності помилок, що не виявляються, то комбінація вважається дозволеною і не замінюється комбінаціями з наступних передач. Якщо ж при першому прийманні виявлена помилка, то здійснюється заміна на кодову комбінацію, прийняту при другому прийманні. Якщо помилка виявлена і при повторній передачі, то відбувається заміна кодової комбінації з третьої передачі. Кожне повторення зменшує ймовірність помилкового декодування кодової комбінації. Залежно від якості каналу, ефективності виявлення помилок завадостійким кодом, а також від вимог до достовірності інформації в системі встановлюється певна кількість повторів.

Повідомлення в системах з повтореннями можуть передаватися послідовно по одному каналу або одночасно по декількох каналах. Обидва способи, в принципі, ідентичні за своїми можливостями відносно підвищення достовірності і вибір того або іншого способу залежить від конкретних умов і вимог до швидкості передачі.

Підвищення достовірності за рахунок багаторазових повторів призводить при незмінній тривалості елементарного символу до зменшення швидкості передачі. Подібна ситуація розглянута при переході від безнадмірного коду до завадостійкого. Цю обставину необхідно враховувати при оцінці ефективності застосування таких систем.

1.10. Системи передачі інформації зі зворотним зв'язком

Надмірність завадостійких кодів у цілому завжди більша, ніж необхідно для корекції помилок. Дійсно, надмірність мають всі кодові комбінації, а помилки містять тільки деякі з них. Було б доцільним вводити надмірність, що залежить від стану каналу, тобто з ростом кількості помилок надмірність збільшувати, а при їх зменшенні – надмірність зменшувати. Надмірність, необхідна для виявлення помилок, завжди менша, ніж надмірність, необхідна для їх виправлення. Технічна реалізація виявлення помилок також істотно простіша. Тому частіше всього завадостійкі коди застосовуються з метою виявлення помилок, а підвищення достовірності передачі дискретних повідомлень досягається за рахунок введення додаткового зворотного каналу. По цьому каналу тим або іншим способом передається інформація, що характеризує умови приймання по прямому каналу, що дозволяє оцінити вплив завади на кожний окремий відрізок повідомлення в ході

його передачі і регулювати на підставі цієї інформації надмірність, що вводитьься.

У загальному випадку система передачі інформації повинна забезпечувати передачу повідомлень в обох напрямках, тому що на обох кінцевих пунктах є і джерела інформації і її одержувачі. За таких умов з'являється можливість використати частину пропускну здатності кожного з каналів для передачі додаткових даних, призначених для підвищення достовірності.

Щоб наочно викласти суть принципів функціонування систем зі зворотним зв'язком, розглянемо окремий випадок, коли передача повідомлень відбувається лише в одному напрямку, а канал другого напрямку використовується тільки для зворотного зв'язку. Структурна схема такої системи передачі інформації зображена на рис. 1.43. У схему входять передавач прямого каналу (ППК), приймач прямого каналу (ПрПК), передавач зворотного каналу (ПЗК) і приймач зворотного каналу (ПрЗК).

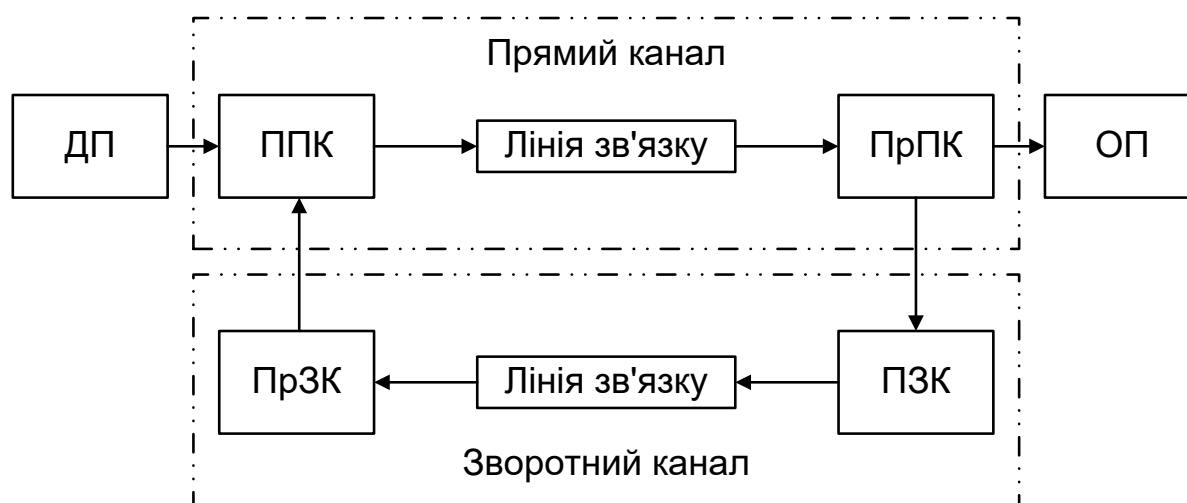


Рис. 1.43. Структурна схема системи передачі інформації зі зворотним каналом

Залежно від того, на якій стороні здійснюється перевірка правильності переданого повідомлення, системи зі зворотним зв'язком можна підрозділити на системи з інформаційним зворотним зв'язком (ІЗЗ) і системи з керуючим зворотним зв'язком (КЗЗ).

У системах з керуючим зворотним зв'язком або з перезапитом, перевірка правильності переданого повідомлення здійснюється на приймальній стороні за встановленим правилом. Потім приймається рішення або про направлення декодованого повідомлення одержувачеві

(у цьому випадку по зворотному каналу може посилатися підтвердження), або про направлення по зворотному каналу запиту на повторення, на зміну способу передачі, і т.д.

У системах з інформаційним зворотним зв'язком або з порівнянням, рішення про те, які зміни вносити в процесі передачі повідомлень, приймається на передавальній стороні на підставі тієї інформації, що надходить по зворотному каналу. Найчастіше приймається рішення про повторення помилково прийнятих повідомлень, але можливі рішення і про зміну коду, і про припинення передачі при поганому стані каналу.

У системах з комбінованим зворотним зв'язком рішення про помилкове прийняття кодової комбінації приймається на приймальній стороні, а рішення про необхідність повторної передачі приймається на передавальній стороні.

Системи зі зворотним зв'язком підрозділяються також на системи з обмеженою і необмеженою кількістю повторень. У системах з необмеженою кількістю повторень кожна кодова комбінація повторюється доти, поки приймач або передавач не ухвалить рішення щодо видачі цієї комбінації споживачеві. У системах з обмеженою кількістю повторень кожна кодова комбінація може повторюватися не більше декількох разів, після чого канал бракується.

Більшість існуючих систем з керуючим зворотним зв'язком належить до систем з перезапитом. Найпоширенішою серед них є система з виявленням помилок на приймальній стороні за допомогою кодів, що виявляють помилки. Прийняті кодові комбінації розбиваються на дозволені і заборонені групи. Якщо комбінація віднесена до дозволеної, то вважається, що така ж комбінація була й передана, і вона направляється одержувачеві. При цьому по зворотному каналу передається сигнал підтвердження. Після одержання сигналу підтвердження керуючий пристрій видає команду на передачу наступної кодової комбінації з одночасним записом її в буферний запам'ятовувальний пристрій передавача. При цьому раніше передана кодова комбінація, що зберігалася тут, стирається.

Очевидно, що в цій системі можливий випадок, коли під дією завад одна дозволена кодова комбінація може бути переведена в іншу. Однак імовірність такої помилки можна зробити досить малою, вибираючи відповідний код.

Якщо прийнята комбінація віднесена до групи заборонених, то на приймальній стороні формується сигнал перезапиту, по якому повторюється передача відповідної кодової комбінації.

Ефективність використання пропускну здібності каналу передачі інформації в подібних системах порівняно невелика, тому що прямий канал простоє в проміжки часу між передачами кодових комбінацій, чекаючи одержання сигналів підтвердження або перезапиту. Прагнення до більш повного використання каналів призвело до розробки систем, у яких передавач посилає в лінію неперервну послідовність комбінацій, не очікуючи одержання сигналів підтвердження. Приймач стирає лише ті комбінації, в яких виявлені помилки, і по них дає сигнал перезапиту. Інші комбінації видаються одержувачеві повідомлень за їх надходженням. При реалізації такої системи виникають труднощі, викликані кінцевим часом передачі і розповсюдження сигналів. Дійсно, нехай у деякий момент часу закінчилося прийняття кодової комбінації, у якій виявлена помилка, і починається передача сигналу перезапиту по зворотному каналу. До цього моменту по прямому каналу вже ведеться передача наступної кодової комбінації, а якщо тривалість розповсюдження сигналу в каналі перевищує тривалість кодової комбінації, то передавач відправить у лінію одну або кілька комбінацій. Ще якась кількість кодових комбінацій буде передана до того моменту, поки буде прийнятий і декодований сигнал перезапиту. Таким чином, при неперервній передачі за час між моментом виявлення помилки і приходом повторної кодової комбінації буде прийнято ще M комбінацій.

Так як передавач повторює лише комбінації, по яких прийнятий сигнал перезапиту, то в результаті повторення з запізнюванням на M комбінацій порядок проходження комбінацій в інформації, що видається одержувачеві повідомлень, буде відрізнятися від порядку надходження кодових комбінацій у систему. Але одержувачеві кодові комбінації повинні бути видані у тому ж порядку, в якому вони передавалися. Тому M комбінацій, прийнятих після виявлення помилки, повинні зберігатися в деякому буферному пристрої пам'яті до моменту, поки не буде прийнята повторена комбінація. Оскільки й у повторених комбінаціях може бути виявлена помилка, ємність буферної пам'яті потрібно збільшити ще в кілька разів, забезпечивши її до того ж складною системою введення і виведення інформації.

Щоб уникнути ускладнення і подорожчання приймачів використовують трохи видозмінену систему з перезапитом, у якій після

виявлення помилки приймач блокується, тобто не приймає M наступних комбінацій. У передавальній частині пристрій пам'яті завжди зберігає $M+1$ останніх переданих комбінацій. Після одержання сигналу перезапиту повторюється не тільки помилково прийнята комбінація, але й весь інший зміст. При цьому зберігається порядок надходження кодових комбінацій одержувачеві, можливість переповнення пристрою пам'яті виключається, керування введенням і виведенням інформації досить прості.

Визначимо основні характеристики системи з керуючим зворотним зв'язком при завадостійкому кодуванні.

При передачі кодової комбінації в системі можливі такі ситуації:

- завада не спотворила комбінацію і з імовірністю $P_{\text{пр}}$ вона прийнята правильно;
- через дію завад комбінація перейшла в іншу дозволену комбінацію, тобто з імовірністю $P_{\text{нвп}}$ виникла помилка, що не може бути виявлена;
- через дію завад комбінація перейшла в заборонену, тобто з імовірністю $P_{\text{вп}}$ виникла помилка, що виявляється.

У першому і у другому випадках декодовані повідомлення направляються одержувачеві, в останньому випадку комбінація бракується й по зворотному каналу направляється запит, після якого все повторюється спочатку.

Помилка, що не виявляється, може виникнути в першому циклі, в другому циклі, в третьому і т.д. При цьому одержувачеві буде передана помилкова інформація. Отже, система може бути охарактеризована залишковою ймовірністю помилкового прийняття кодової комбінації $P_{\text{пом зал}}$. Ця ймовірність складається з імовірності невиявленої помилки при першій передачі, при другій передачі і т.д. Якщо комбінація приймалася m разів із виявленням помилки, а потім була прийнята з невиявленням помилки, то ймовірність цієї події дорівнює $P_{\text{вп}}^m P_{\text{нвп}}$. Тоді для $P_{\text{пом ост}}$ можна записати

$$P_{\text{пом зал}} = P_{\text{нвп}} + P_{\text{вп}} P_{\text{нвп}} + P_{\text{вп}}^2 P_{\text{нвп}} + \dots = P_{\text{нвп}} \sum_{m=0}^{\infty} P_{\text{вп}}^m = \frac{P_{\text{нвп}}}{1 - P_{\text{вп}}} . \quad (1.95)$$

Імовірності $P_{\text{пр}}$, $P_{\text{нвп}}$ і $P_{\text{вп}}$ можна визначити знаючи структуру коду та параметри каналу. Імовірність невиявлення помилки може бути оцінена за формулою

$$P_{\text{нвп}} \leq 1 - \sum_{i=0}^{d_{\text{min}}-1} C_n^i p_0^i (1-p_0)^{n-i}.$$

Імовірність правильного прийняття може бути знайдена за очевидною формулою $P_{\text{пр}} = (1-p_0)^n \approx np_0$. Імовірність виявлення помилки можна одержати виходячи з того, що $P_{\text{пр}} + P_{\text{нвп}} + P_{\text{вп}} = 1$. З огляду на те, що в системі з перезапитом код повинен бути обраний так, щоб $P_{\text{нвп}} \ll P_{\text{вп}}$, для $P_{\text{виявл}}$ маємо $P_{\text{вп}} \approx 1 - (1-p_0)^n$.

Швидкість передачі в системах з КЗЗ визначається не тільки відношенням k/n , але і втратою часу на повторення. У зв'язку з цим системи зі зворотним зв'язком можна охарактеризувати також відносною швидкістю передачі R' , під якою розуміється відношення математичного очікування числа інформаційних символів, що надійшли до одержувача, до загального числа кодових символів, що надійшли в канал. Кодова комбінація надходить до одержувача тільки тоді, коли в ній не виявлені помилки. Отже, при одержанні кожної кодової комбінації одержувачеві з імовірністю $1 - P_{\text{вп}}$ видається k інформаційних символів. Тому

$$R' = \frac{k}{n}(1 - P_{\text{вп}}).$$

Оскільки $P_{\text{вп}}$ залежить від стану каналу, то і надмірність у системі з КЗЗ змінюється також відповідно до стану каналу.

В системах із блокуванням відносна швидкість передачі ще менша, тому що при запиті повторюється не одна, а кілька кодових комбінацій.

При розгляді характеристик систем з керуючим зворотним зв'язком не враховувалася ймовірність виникнення помилок у сигналах підтвердження і перезапиту, що найчастіше допустимо, тому що такими сигналами вибираються завадозахищені кодові послідовності. Проте, в реальних каналах імовірність помилкового прийняття цих сигналів все-таки не дорівнює нулю й необхідно враховувати можливість того, що сигнал підтвердження буде прийнятий як перезапит або навпаки. При цьому виникають специфічні помилки у прийнятій інформації. Якщо при правильному прийнятті по прямому каналу у зворотному каналі відбувається трансформація сигналу підтвердження в сигнал запиту, то система повторює передачу кодової комбінації, в результаті чого відбувається вставка. У тих же випадках, коли при передачі по прямому

каналу приймач виявляє помилки, а у зворотному каналі відбувається трансформація сигналу запиту в сигнал підтвердження, передавач видає нове повідомлення. При цьому попередні повідомлення стираються і відбувається випадання.

Для боротьби з випаданнями та вставками, крім захисту сигналів підтвердження і перезапиту за допомогою кодів, приймаються спеціальні заходи. Так, для зменшення кількості випадань у зворотному каналі встановлюється несиметричне правило декодування, при якому ймовірність помилкового прийняття сигналу перезапиту мізерно мала за рахунок збільшення ймовірності помилкового прийняття сигналу підтвердження. Для того щоб часті помилки при передачі сигналу підтвердження не призводили до появи зайвих вставок, до кожної кодової комбінації додають розпізнавальні символи, що вказують на те, чи передається ця комбінація вперше, чи повторюється. Якщо на декодер надійде повторна комбінація, а перед цим перезапиту не було, то вона стирається і не надходить до одержувача.

Всі ці заходи, як легко бачити, вимагають деякого збільшення надмірності й ускладнення апаратури.

Застосування систем з керуючим зворотним зв'язком ефективно в каналах, де змінюється ймовірність помилки. Коли $P_{\text{пом кор}}$ стає близькою до $1/2$, настає обривання зв'язку, тому що пропускна здатність каналу падає майже до нуля. При цьому система перебуває в режимі постійного перезапиту. Але якщо вона побудована раціонально, то одержувачеві помилкова інформація не видається. При поліпшенні завадової обстановки швидкість передачі збільшується, а вірність продовжує залишатися на заданому рівні. Таким чином, система з КЗЗ як би пристосовується до стану каналу. Це є важливим достоїнством системи з КЗЗ.

У найпоширеніших системах з інформаційним зворотним зв'язком прийняті приймачем кодові комбінації або символи повністю передаються по зворотному каналу на передавач, де відбувається їх порівняння з переданими повідомленнями. У випадку збігу комбінацій прямого та зворотного каналів здійснюється передача наступної комбінації, а у випадку відмінності – повтор раніше переданої. Такий інформаційний зворотний зв'язок називають повним або ретрансляційним.

При ідеальному зворотному каналі повний інформаційний зворотний зв'язок дозволяє без застосування завадостійких кодів забезпечити

абсолютно достовірну передачу інформації в прямому напрямку. Передавач у цьому випадку виявляє всі помилки приймача і при відсутності обмежень на затримку завжди має можливість виправити їх.

У реальних системах при повному інформаційному зворотному зв'язку не виявляються лише так звані дзеркальні помилки, тобто помилки, що відбуваються в тому самому символі двічі: при передачі по прямому й зворотному каналах.

Недоліком описаної системи є мала продуктивність, оскільки на передачу кожної інформаційної комбінації витрачується принаймні (при відсутності повторних передач) час $\Delta t = 2t_k + 2t_p$, де t_k – час передачі кодової комбінації; t_p – час розповсюдження сигналів між станціями.

Більш високу пропускну здібність забезпечують системи з ІЗЗ, у яких здійснюється неперервна передача інформаційних комбінацій.

У таких системах тільки у випадку незбігу комбінацій прямого і зворотного каналів передавач припиняє передачу чергових комбінацій і робить повторення помилкової комбінації. У буферних запам'ятовуючих пристроях передавача і приймача при цьому зберігається не одна, а M інформаційних комбінацій. З прийняттям сигналу незбігу стираються всі M комбінацій, а по прямому каналу слідом за сигналом стирання повторюються всі M комбінацій.

Визначимо для розглянутої системи ймовірність відсутності перекручувань і ймовірності виявлення та не виявлення помилок в одному циклі. У системі з ІЗЗ перекручування не виявляється тільки в тому випадку, якщо кодова комбінація B_i перетворилася в прямому каналі в B_j , а у зворотному каналі відбувається інверсна трансформація B_j в B_i . Тому, що використовується безнадмірне кодування, то ймовірність правильного приймання по прямому каналу

$$P_{\text{пр}}^{(\text{прям})} = (1 - P_{\text{пом}}^{(\text{прям})})^n,$$

а по зворотному каналу

$$P_{\text{пр}}^{(\text{звор})} = (1 - P_{\text{пом}}^{(\text{звор})})^n.$$

Ймовірність відсутності перекручувань після повного циклу передачі

$$P_{\text{пр}} = P_{\text{пр}}^{(\text{прям})} P_{\text{пр}}^{(\text{звор})} = (1 - P_{\text{пом}}^{(\text{прям})})^n (1 - P_{\text{пом}}^{(\text{звор})})^n. \quad (1.96)$$

Імовірність перетворення в прямому каналі кодової комбінації V_i у V_j обчислюється за формулою

$$P_{\text{пом}}^{(\text{прям})} = 1 - P_{\text{пр}}^{(\text{прям})} = 1 - (1 - P_{\text{пом}}^{(\text{прям})})^n.$$

Імовірність зворотної трансформації V_j у V_i за умови, що передача $N = 2^n$ комбінацій рівноймовірна, обчислюється за формулою

$$P_{\text{пом}}^{(\text{звор})} = \frac{1 - P_{\text{пр}}^{(\text{звор})}}{N} = \frac{1 - (1 - P_{\text{пом}}^{(\text{звор})})^n}{N}.$$

Загальна ймовірність не виявлення помилки при одноразовій передачі

$$P_{\text{нв}} = P_{\text{пом}}^{(\text{прям})} P_{\text{пом}}^{(\text{звор})} = \frac{1}{N} \left[1 - (1 - P_{\text{пом}}^{(\text{прям})})^n \right] \left[(1 - P_{\text{пом}}^{(\text{звор})})^n \right] \dots \quad (1.97)$$

Формули (1.96) і (1.97) описують два можливих результати одного циклу обміну інформацією. Третій результат, що доповнює правильну передачу й помилку, що не виявляється, до повної групи подій – помилка, що виявляється. Імовірність такої події

$$P_{\text{вп}} = 1 - P_{\text{нвп}} - P_{\text{пр}}.$$

В системі з інформаційним зворотним зв'язком ситуації, що призводять до не виявлення перекручування кодової комбінації, точно такі ж, як і в системі з КЗЗ, внаслідок чого залишкова ймовірність помилкового прийняття кодової комбінації може бути обчислена за формулою (1.95).

Основною проблемою при побудові системи з інформаційним зворотним зв'язком є захист від трансформації службових сигналів у комбінацію основного повідомлення або навпаки. Тому завжди необхідно вводити певну надмірність, щоб захистити службові сигнали від таких трансформацій. Для цього застосовують завадостійке кодування.

Звичайно по прямому каналу передають тільки інформаційні символи, а перевіірочні запам'ятовуються на передавальній стороні. Прийняті інформаційні символи також піддаються кодуванню на приймальній стороні й по зворотному каналу посиляють тільки перевіірочні символи. На передавальній стороні прийняті по зворотному каналу перевіірочні символи порівнюються з перевіірочними символами, що зберігаються, на передавальній стороні. Якщо вони не збігаються, то по

прямому каналу посиляється сигнал заперечення і здійснюється повторення.

Системи з різними зворотними зв'язками мають певні переваги і недоліки, які треба врахувати при їх використанні. Перевага систем з ІЗЗ полягає в можливості використання безнадмірних кодів. Це обумовлює простоту їх технічної реалізації. Для них характерна простота роботи і висока достовірність передачі. Недоліки цих систем полягають у тому, що видача споживачеві навіть правильно отриманої інформації затримується доти, доки після аналізу інформації з передавальної сторони не буде отриманий сигнал підтвердження. В системах з ІЗЗ однаково завантажений як прямий, так і зворотний канали передачі інформації.

Головна перевага систем з КЗЗ полягає в тому, що правильна інформація видається споживачеві практично без затримки. Ці системи мають високу достовірність і низьку завантаженість зворотного каналу. До недоліків цих систем слід віднести необхідність використання надмірного завадостійкого коду. Це обумовлює необхідність використання кодуєчих і декодуєчих пристроїв, що ускладнює побудову і роботу апаратури. В них значно завантажений прямий канал.

Система з комбінованим зворотним зв'язком мають певні комбінації властивостей, притаманних системам з інформаційним і керуючим зворотним зв'язком. Досліди підказують, що коли завади в прямому і зворотному каналах однакові, то як інформаційний, так і керуючий зворотний зв'язок забезпечують приблизно однакову достовірність. Канали передачі інформації в цілому також завантажені однаково. Системи з ІЗЗ доцільно розробляти і використовувати в тих випадках, коли потрібна максимальна простота побудови системи і допускається затримка використання отриманої інформації. Якщо така затримка недопустима, то слід використати систему з КЗЗ.

Практична реалізація розглянутих видів зворотного зв'язку дає можливість забезпечити потрібну достовірність передачі повідомлень системами при незначному їх ускладненні і припустимих економічних витратах.

Теорія систем зі зворотним зв'язком досить складна та ще далека до завершення; багато досліджень виконані з істотними допущеннями й ідеалізацією. Проте, використання каналу зворотного зв'язку у всіх випадках, де це можливо, є однією з основних тенденцій розвитку систем передачі інформації.

Контрольні запитання до теми 1

1. Дайте визначення кількості інформації. Як вона розраховується?
2. Дайте визначення ентропії.
3. Яким умовам задовольняє ентропія випадкового експерименту?
4. Поясніть порядок розрахунку ентропії випадкового експерименту з рівноймовірними наслідками.
5. Дайте визначення одиниці виміру невизначеності.
6. Поясніть порядок розрахунку ентропії випадкового експерименту з нерівноймовірними наслідками.
7. Поясніть умови максимізації ентропії джерел повідомлень.
8. Поясніть порядок обчислення ентропії експерименту при взаємозалежності подій.
9. Поясніть порядок обчислення ентропії об'єднання.
10. Поясніть суть надмірності джерел повідомлень.
11. Дайте визначення диференціальної ентропії джерел неперервних повідомлень.
12. Поясніть порядок обчислення диференціальної ентропії джерел з нормальним законом розподілу.
13. Поясніть порядок обчислення диференціальної ентропії процесу з рівномірним розподіленням.
14. Поясніть порядок визначення максимальної диференціальної ентропії.
15. Дайте визначення та поясніть класифікацію каналів передачі інформації.
16. Поясніть модель дискретного каналу передачі інформації.
17. Визначте кількість інформації, що передається по дискретному каналу передачі інформації.
18. Визначте пропускну здібність дискретного каналу передачі інформації.
19. Визначте пропускну здібність двійкового симетричного каналу.
20. Поясніть призначення методів ефективного кодування інформації.
21. Поясніть суть ефективного кодування Шеннона – Фано.
22. Поясніть суть ефективного кодування Хаффмена.
23. Дайте класифікацію завадостійких кодів.
24. Поясніть загальні принципи завадостійкого кодування.
25. Що таке відстань по Хеммінгу та мінімальна кодова відстань?

26. Що таке вага кодового слова. Чому дорівнює мінімальна кодова відстань для лінійних кодів?
27. Поясніть порядок розрахунку характеристик завадостійких кодів.
28. Поясніть порядок вибору породжувальної матриці лінійного коду.
29. Поясніть порядок побудови лінійного коду за допомогою породжувальної матриці .
30. Що таке синдром кодового слова. Від чого він залежить?
31. Поясніть порядок побудови перевіркової матриці лінійного коду.
32. Поясніть порядок декодування лінійного коду за допомогою перевіркової матриці.
33. Поясніть принципи побудови кодуючих і декодуючих пристроїв лінійних групових кодів.
34. Дайте визначення лінійних циклічних кодів.
35. Назвіть вимоги до породжувального многочлена циклічних кодів.
36. Поясніть принципи побудови пристроїв кодування циклічних кодів на схемах множення.
37. Поясніть принципи побудови пристроїв кодування циклічних кодів на схемах розподілу многочленів.
38. Поясніть принципи побудови пристроїв декодування циклічних кодів.
39. Поясніть принцип побудови згортувальних кодів.
40. Поясніть принцип декодування згортувальних кодів за алгоритмом Витербі.
41. Поясніть принципи багаторазової передачі кодових комбінацій.
42. Поясніть сутність систем передачі інформації зі зворотним зв'язком.
43. У чому суть систем передачі інформації з керуючим зворотним зв'язком?
44. У чому суть систем передачі інформації з інформаційним зворотним зв'язком?

Використана література до теми 1

1. Северінов О.В. Теорія інформації та кодування: Навч. посібник / О.В. Северінов, О.С. Жученко. – Х.: ХУПС, 2007. – 186 с.
2. Основи теорії інформації та кодування: Навчальний посібник / Л.С. Сорока, О.В. Северінов, О.С. Жученко, С.Г. Рассомахін, І.В. Рубан. – Х.: ХНУ імені В.Н. Каразіна, 2008. – 264 с.
3. Теорія інформації: підручник для слухачів, курсантів та студентів вищих навчальних закладів / І.В. Рубан, С.І. Хмелевський, О.В. Петров, С.М. Балакірева, Ю.В. Данюк, Ю.С. Долгий, М.А. Павленко, О.В. Северінов, О.О. Писарчук, Г.А. Пухальська, С.В. Смеляков. – Харків: ХНУПС, 2018. – 274 с.
4. Berlekamp E. Algebraic coding Theory / E. Berlekamp – World Scientific Publishing Co. Ltd., 2015. – 501 p.
5. Blahut R.E. Algebraic Codes for Data Transmission / R. E. Blahut. - Cambridge University Press, 2003, -567 pp.
6. Peterson W.W. Error-Correcting Codes, second edition / [W. Wesley Peterson](#). – MIT Press, 1972. 575 p.
7. Clark G.C. Error-Correction Coding for Digital Communications / G.C. Clark, J.B. Cain. - Springer, 1981. – 432 p.
8. Кузнецов О.О. Сигнали і коди. Алгебраїчні методи синтезу. Монографія. О.О. Кузнецов, С.П. Євсєєв, С.В. Кавун, О.Г. Король – Харків: Вид. ХНЕУ, 2009. – 400 с.

Тема 2. Криптографічні механізми захисту інформації в інформаційних системах

2.1. Математичні основи сучасної теорії захисту інформації

Загальнотеоретичною основою побудови більшості сучасних криптографічних методів захисту інформації є положення теорії скінченних полів, систем лишкових класів, теорії чисел, методів булевої алгебри, елементів кореляційного та спектрального аналізу [1–10]. Нижче викладаються деякі математичні положення, що є необхідними для засвоювання сучасних принципів побудови криптографічних механізмів захисту інформації.

2.1.1. Математичні положення теорії скінченних полів та систем класів лишків

Означення. Нехай a та b – цілі числа. Нехай a дорівнює b за модулем n , позначається через $a \equiv b \pmod{n}$, якщо $a - b$ ділиться на n .

Приклад. $23 \equiv 3 \pmod{5}$, тому що $23 - 3 = 5 * 4$;

$-25 \equiv 3 \pmod{7}$, тому що $-25 - 3 = 7 * -4$;

Властивості. Нехай a, a_1, b, b_1, c – цілі числа.

1. $a \equiv b \pmod{n}$, тільки тоді, коли a та b дають рівні лишки при діленні на n .

2. Рефлексивність. $a \equiv a \pmod{n}$.

3. Симетрія. Якщо $a \equiv b \pmod{n}$, то $b \equiv a \pmod{n}$.

4. Транзитивність. Якщо $a \equiv b \pmod{n}$ і $b \equiv c \pmod{n}$, то $a \equiv c \pmod{n}$.

5. Якщо $a \equiv a_1 \pmod{n}$ та $b \equiv b_1 \pmod{n}$,

то $a + b \equiv a_1 + b_1 \pmod{n}$ і $a * b \equiv a_1 * b_1 \pmod{n}$.

Означення. Нехай n – ціле додатне число. Позначимо через C_t клас, у який об'єднано усі цілі числа, які при діленні на n дають один і той же залишок t . Всі цілі числа розбіються на n класів $C_0, C_1, \dots, C_{n-1}$, які називаються класами лишків за модулем n .

Приклад. Нехай $n = 7$. Тоді до класу C_2 належать числа виду $7 * x + 2$, де $x \in \mathbb{Z}$.

Твердження. Два числа є порівнюваними за модулем n , якщо вони належать одному класу лишків за модулем n .

Означення. Якщо з кожної системи лишків за модулем n взяти по одному представнику, то отриману систему чисел називають повною системою лишків за модулем n . Якщо повну систему лишків будувати з найменших невід'ємних лишків, то вона прийме вигляд: $0, 1, 2, \dots, n - 1$. Її будемо позначати через Z_n . Арифметичні операції над елементами цієї множини відбуваються за модулем n . Повна система лишків утворює групу з операцією додавання.

Приклад. Повною системою лишків за модулем 5 буде множина чисел $Z_5 = \{0, 1, 2, 3, 4\}$.

Приклад. $Z_{12} = \{0, 1, 2, \dots, 11\}$. В класі Z_{12} : $11 + 6 = 5$, тому що $11 + 6 = 17 \equiv 5 \pmod{12}$. $10 * 3 = 6$, тому що $10 * 3 = 30 \equiv 6 \pmod{12}$.

Теорема про лишки лінійної форми. Якщо у лінійній формі $ax + b$ число x пробігає усі значення з повної системи лишків за модулем n при $\text{НСД}(a, n) = 1$ та довільному b , тоді $ax + b$ пробігає усі значення повної системи лишків за модулем n .

Доведення. Отримана система складається з n чисел, оскільки замість x у формі $ax + b$ підставляються n різних значень. Доведемо від супротивного, що всі ці n отриманих чисел різні. Нехай x_1 та x_2 не порівнювані за модулем n , але $ax_1 + b \equiv ax_2 + b \pmod{n}$. Тоді $ax_1 \equiv ax_2 \pmod{n}$. Але оскільки $\text{НСД}(a, n) = 1$, то $x_1 \equiv x_2 \pmod{n}$. Отримали суперечність.

□

Приклад. Нехай $n = 6$, $a = 5$, $b = 1$, при цьому $\text{НСД}(a, n) = 1$. Підставимо до форми $5 * x + 1$ значення x із повної системи лишків $Z_6 = \{0, 1, 2, \dots, 5\}$.

x	$5 * x + 1 \pmod{6}$
0	1
1	0
2	5
3	4
4	3
5	2

У правому стовпчику таблиці всі числа різні.

Означення. Якщо з кожної системи лишків C_t ($t = 0, 1, \dots, n - 1$) за модулем n , для якої $\text{НСД}(t, n) = 1$ взяти по одному представнику, то отриману систему чисел називають зведеною системою лишків за

модулем n і позначають через Z_n^* . Зведена система лишків утворює групу з операцією множення.

Якщо p – просте, то $Z_p^* = \{1, 2, \dots, p-1\}$.

Означення. Порядком множини A будемо називати кількість її елементів і позначати через $|A|$.

Приклад. Зведеною системою лишків для $n = 10$ буде множина чисел $Z_{10}^* = \{1, 3, 7, 9\}$, $|Z_{10}^*| = 4$.

1. Якщо p – просте число, то $\varphi(p) = p - 1$ та $\varphi(p^a) = p^a \cdot (1 - 1/p)$ для довільного a .

2. Якщо m та n взаємно прості, то $\varphi(m \cdot n) = \varphi(m) \cdot \varphi(n)$.

3. Якщо $n = p_1^{a_1} p_2^{a_2} \dots p_k^{a_k}$, то $\varphi(n) = n \cdot (1 - 1/p_1) \cdot (1 - 1/p_2) \cdot \dots \cdot (1 - 1/p_k)$.

4. $\varphi(n) = |Z_n^*|$.

5. $\sum_{d|n} \varphi(d) = n$.

Приклад. Обчислити $\varphi(728)$, $\varphi(10)$.

$728 = 7 \cdot 8 \cdot 13 = 2^3 \cdot 7 \cdot 13$, $10 = 2 \cdot 5$.

$\varphi(728) = 728 \cdot (1 - 1/2) \cdot (1 - 1/7) \cdot (1 - 1/13) = 728 \cdot (1/2) \cdot (6/7) \cdot (12/13) = 288$.

$\varphi(10) = 10 \cdot (1 - 1/2) \cdot (1 - 1/5) = 10 \cdot (1/2) \cdot (4/5) = 4$.

Твердження. Порядком групи Z_n^* будемо називати кількість елементів у ній та позначати $|Z_n^*|$. При цьому:

$$|Z_n^*| = \varphi(n).$$

Приклад. $Z_{10}^* = \{1, 3, 7, 9\}$, $|Z_{10}^*| = \varphi(10) = 4$.

Твердження. Порядком групи Z_n^* будемо називати кількість елементів у ній та позначати $|Z_n^*|$. При цьому:

$$|Z_n^*| = \varphi(n)$$

Приклад. $Z_{10}^* = \{1, 3, 7, 9\}$, $|Z_{10}^*| = \varphi(10) = 4$.

Теорема 2.1. (про лишки лінійної форми). Якщо у лінійній формі $a \cdot x$, число x пробігає усі значення зі зведеної системи лишків за модулем n при НСД $(a, n) = 1$, тоді $a \cdot x$ пробігає всі значення зведеної системи лишків за модулем n .

Доведення. Підставивши замість змінної x у лінійну форму $a \cdot x \cdot \varphi(n)$ чисел, отримаємо $\varphi(n)$ різних чисел, оскільки вони належать за модулем n різним класам (це впливає з першої теореми про лишки лінійної форми для $b = 0$). Оскільки x – лишок зведеної системи, то НСД $(x, n) = 1$. За

умовою теореми НСД $(a, n) = 1$. З останніх двох рівностей випливає, що $\text{НСД}(a * x, n) = 1$, тобто числа $a * x$ взаємно прості з n .

□

Приклад. Розглянемо множину чисел $\{1, 3, 7, 9\}$, яка є зведеною системою лишків для $n = 10$. Нехай $a = 7$, $\text{НСД}(7, 10) = 1$. Тоді мають місце співвідношення:

$$7 * 1 \pmod{10} \equiv 7 \pmod{10} \equiv 7$$

$$7 * 3 \pmod{10} \equiv 21 \pmod{10} \equiv 1$$

$$7 * 7 \pmod{10} \equiv 49 \pmod{10} \equiv 9$$

$$7 * 9 \pmod{10} \equiv 63 \pmod{10} \equiv 3$$

Доведення. Оберненням числа a за модулем n (позначається a^{-1}) називається таке число $x \in Z_n$, що $ax \equiv 1 \pmod{n}$.

Приклад. Обчислити $5^{-1} \pmod{7}$. Знайдемо всі значення $5 * x \pmod{7}$, $x = 0, \dots, 6$.

$$5 * 0 \pmod{7} \equiv 0 \pmod{7} \equiv 0$$

$$5 * 1 \pmod{7} \equiv 5 \pmod{7} \equiv 5$$

$$5 * 2 \pmod{7} \equiv 10 \pmod{7} \equiv 3$$

$$5 * 3 \pmod{7} \equiv 15 \pmod{7} \equiv 1$$

$$5 * 4 \pmod{7} \equiv 20 \pmod{7} \equiv 6$$

$$5 * 5 \pmod{7} \equiv 25 \pmod{7} \equiv 4$$

$$5 * 6 \pmod{7} \equiv 30 \pmod{7} \equiv 2$$

Оскільки $5 * 3 \pmod{7} \equiv 1$, то $5^{-1} \pmod{7} \equiv 3$.

Твердження. Обернене число a^{-1} за модулем n існує тільки тоді, коли $\text{НСД}(a, n) = 1$.

Якщо $\text{НСД}(a, n) = k > 1$, то для довільного елемента $x \in Z_n$ вираз $a * x \pmod{n}$ буде ділитися на k і ніколи не буде дорівнювати 1 (тому що 1 не ділиться на k при $k > 1$).

Квадратичні лишки

Означення. Число $a \in Z_n^*$ називається квадратичним лишком або квадратом за модулем n , якщо існує таке $x \in Z_n^*$, що $x^2 \equiv a \pmod{n}$. Якщо такого x не існує, то число a називається *квадратичним нелишком*. Множина всіх квадратичних лишків за модулем n позначається через Q_n , нелишків – через $\overline{Q}_n$. За означенням $0 \notin Z_n^*$, отже $0 \notin Q_n$ та $0 \in \overline{Q}_n$.

Теорема 2.2. Нехай p – непарне просте число, g – генератор Z_p^* . Тоді число a є квадратичним лишком за модулем p тільки тоді, коли $a = g^i \pmod{p}$, де i – парне ціле.

Доведення. Якщо $a = g^{2k} \pmod{p}$, то $a = b^2 \pmod{p}$, де $b = g^k \pmod{p}$.

Нехай $a = g^k \pmod{p}$ – елемент Z_p^* . Піднесемо його до квадрата:

$a^2 = g^{2k} \pmod{p} \equiv g^i \pmod{p}$. Оскільки $2^k \pmod{p-1} = i$ – парне число, то звідси i впливає твердження про те, що квадрат довільного елемента $a \in Z_p^*$ представляється у вигляді $g^i \pmod{p}$ лише для парного i .

□

Наслідок. $|Q_p| = (p-1)/2$, $|\overline{Q_p}| = (p-1)/2$.

Тобто половина елементів Z_p^* є квадратичними лишками, а половина – ні.

Приклад. Число $a = 3$ є генератором Z_7^* . Степені a наведені у наступній таблиці.

i	0	1	2	3	4	5	6
$a^i \pmod{7}$	1	3	2	6	4	5	1

Звідси $Q_7 = \{1, 2, 4\}$, $\overline{Q_7} = \{3, 5, 6\}$.

Схема множення квадратичних лишків та нелишків аналогічна схемі додавання парних та непарних цілих чисел:

лишок * лишок = лишок

лишок * нелишок = нелишок

нелишок * нелишок = лишок

Приклад. Дослідимо операції множення лишків та нелишків у групі Z_7^* .

$2 \in Q_7, 4 \in Q_7 : 2 * 4 = 8 \equiv 1 \in Q_7$

$2 \in Q_7, 5 \in \overline{Q_7} : 2 * 5 = 10 \equiv 3 \in \overline{Q_7}$

$5 \in \overline{Q_7}, 6 \in \overline{Q_7} : 5 * 6 = 30 \equiv 2 \in Q_7$

Твердження. Нехай n – добуток двох різних простих чисел p та q , $n = p * q$. Тоді число $a \in Z_n^*$ є квадратичним лишком за модулем n тільки тоді, коли $a \in Q_p$ та $a \in Q_q$. Звідси

$|Q_n| = |Q_p| * |Q_q| = (p-1)(q-1)/4$ та $|\overline{Q_n}| = 3(p-1)(q-1)/4$.

Приклад. Нехай $n = 21$. Тоді $|Q_{21}| = (2 * 6)/4 = 3$, $Q_{21} = \{1, 4, 16\}$,

$|\overline{Q_{21}}| = (3 * 2 * 6)/4 = 9$, $\overline{Q_{21}} = \{2, 5, 8, 10, 11, 13, 17, 19, 20\}$.

Означення. Нехай $a \in Q_n$. Якщо $x \in Z_n^*$ задовольняє $x^2 \equiv a \pmod{n}$, то x називається *квадратним коренем числа a за модулем n* .

Теорема 2.3. Нехай p – просте, $p \equiv 3 \pmod{4}$, $a \in Q_p$. Тоді розв'язком рівняння

$$x^2 \equiv a \pmod{p}$$

будуть числа r та $-r$, де $r = a^{\frac{p+1}{4}} \pmod{p}$.

Доведення. $r^2 \equiv a^{\frac{p+1}{2}} \pmod{p} \equiv \sqrt{a^{p+1}} \pmod{p} \equiv \sqrt{a^{p-1} \cdot a^2} \pmod{p} \equiv \sqrt{1 \cdot a^2} \pmod{p} \equiv a \pmod{p}$, оскільки за теоремою Ферма $a^{p-1} \pmod{p} \equiv 1$.

□

Доведення теореми можна провести, використовуючи критерій Ейлера. Оскільки a – квадратичний лишок за модулем p , то

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p} = 1.$$

Враховуючи, що число p можна подати у вигляді $p = 4m + 3$ для деякого натурального m , то $\frac{p-1}{2} = 2m + 1$. Тобто $a^{\frac{p-1}{2}} = a^{2m+1} \equiv 1 \pmod{p}$, $a^{2m+2} \equiv a \pmod{p}$. Візьмемо квадратний корінь лівої та правої частин останньої рівності:

$$a^{m+1} \equiv \sqrt{a} \pmod{p}.$$

Приклад. Обчислити $\sqrt{5}$ та $\sqrt{3}$ в Z_{11}^* .

$p = 11$ – просте, $p \equiv 3 \pmod{4}$, $\frac{p+1}{4} = 3$.

$$\sqrt{5} : 5^3 \pmod{11} \equiv 4. \quad -4 \equiv 7 \pmod{11}.$$

$$\text{Перевірка: } 4^2 \pmod{11} \equiv 5, \quad 7^2 \pmod{11} \equiv 5.$$

$$\sqrt{3} : 3^3 \pmod{11} \equiv 5. \quad -5 \equiv 6 \pmod{11}.$$

$$\text{Перевірка: } 5^2 \pmod{11} \equiv 3, \quad 6^2 \pmod{11} \equiv 3.$$

Теорема. 2.4. Нехай $n = p * q$, де p, q – непарні прості числа. Число $a \in Z_n^*$ є квадратичним лишком за модулем n тільки тоді, коли a є квадратичним лишком за модулем p та q . Тобто

$$a \in Q_n \Leftrightarrow a \in Q_p \text{ та } a \in Q_q.$$

$$\text{Звідси } |Q_n| = |Q_p| * |Q_q| = (p-1)(q-1) / 4.$$

□

Приклад. Нехай $n = 21 = 3 * 7$. $a \in Q_{21} \Leftrightarrow a \in Q_3$ та $a \in Q_7$.

$Q_3 = \{1\}$, поширимо остачі до 21 за модулем 3: $\{1, 4, 7, 10, 13, 16, 19\}$.

$Q_7 = \{1, 2, 4\}$, поширимо остачі до 21 за модулем 7: $\{1, 2, 4, 8, 9, 11, 15, 16, 18\}$.

$|Q_{21}| = |Q_3| * |Q_7| = 1 * 3 = 3$. Числа, спільні в двох множинах поширених остач, і є квадратичними лишками за модулем 21.

$$Q_{21} = \{1, 4, 16\}.$$

Алгоритм обчислення оберненого числа. Якщо необхідно обчислити $a^{-1} \pmod{n}$, то знайдемо за розширеним алгоритмом Евкліда НСД $(a, n) = d$ та такі значення x та y , що $ax + ny = d$. Якщо $d > 1$, то оберненого значення не існує. Інакше $a^{-1} \pmod{n} = x$, тому що

$$ax \pmod{n} \equiv ax + ny \pmod{n} \equiv d = 1.$$

Приклад. Обчислити $2^{-1} \pmod{7}$.

НСД $(2, 7) = 1$, отже обернене значення існує.

За розширеним алгоритмом Евкліда матимемо:

$$2 * (-3) + 7 * 1 = 1, \text{ звідки } 2^{-1} \pmod{7} \equiv -3 \pmod{7} \equiv 4.$$

$$\text{Перевірка: } 2 * 4 \pmod{7} \equiv 8 \pmod{7} \equiv 1.$$

Означення. Діленням числа a на число b за модулем n називається множення a на b^{-1} за умови існування b^{-1} .

Приклад. Результатом $4 : 5 \pmod{7}$ буде

$$4 * 5^{-1} \pmod{7} \equiv 4 * 3 \pmod{7} \equiv 12 \pmod{7} \equiv 5.$$

$$\text{Перевірка: } 5 * 5 \pmod{7} \equiv 25 \pmod{7} \equiv 4$$

Подільність цілих чисел

Нехай m – деяке ціле позитивне число. Будь-яке ціле число b подамо у єдиному вигляді:

$$b = mq + r, \quad \text{якщо } (0 < r < m). \quad (2.1)$$

Число q (1.1) називається часткою, а ціле r – остатком від розподілу b на m , або найменшим від'ємним відрахуванням за модулем m . З виразу (2.1) також випливає, що якщо m ділить b , остаток $r = 0$. У цьому випадку будемо писати m/b . Для довільних цілих чисел a і b справедливі наступні рівності:

$$(a + b) \pmod{m} = (a \pmod{m} + b \pmod{m}) \pmod{m}; \quad (2.2)$$

$$(ab) \pmod{m} = (a \pmod{m} \times b \pmod{m}) \pmod{m}. \quad (2.3)$$

Найбільший спільний дільник і найменше спільне кратне

Спільним дільником цілих чисел $a_1, \dots, a_n$ називається будь-яке ціле число d , що ділить кожне із чисел $a_1, a_2, \dots, a_n$:

$$d/a_1, d/a_2, \dots, d/a_n.$$

Найбільшим спільним дільником цілих чисел $a_1, \dots, a_2, \dots, a_n$ називається такий додатний спільний дільник чисел $a_1, \dots, a_n$, що ділиться на будь-який інший спільний дільник цих чисел. Найбільший із спільних дільників будемо позначати:

$$\text{НСД}(a_1, \dots, a_2, \dots, a_n) \text{ або } [a_1, \dots, a_2, \dots, a_n].$$

Спільним кратним ненульових цілих чисел $a_1, \dots, a_2, \dots, a_n$ називається будь-яке ціле число, що ділиться на кожне із цих чисел.

Найменшим спільним кратним чисел $a_1, \dots, a_2, \dots, a_n$ називається найменше додатне число, кратне всім цим числам. Найменше спільне кратне чисел $a_1, a_2, \dots, a_n$ позначається як:

$$\text{НСК}(a_1, \dots, a_2, \dots, a_n) \text{ або } [a_1, \dots, a_2, \dots, a_n].$$

Числа $a_1, \dots, a_2, \dots, a_n$ називаються *взаємно простими*, якщо $\text{НСД}(a_1, \dots, a_2, \dots, a_n) = 1$, тобто найбільший спільний дільник цих чисел дорівнює 1. Для взаємно простих чисел *найбільшим спільним кратним* $[a_1, \dots, a_2, \dots, a_n]$ є добуток цих чисел:

$$[a_1, a_2, \dots, a_n] = \prod_{i=1}^n a_i = a_1 a_2 a_3 \dots a_n. \quad (2.4)$$

Китайська теорема про залишки

Теорема 2.5. (Китайська теорема про залишки). Нехай $n_1, n_2, \dots, n_k$ — взаємно прості числа. Тоді система порівнянь:

$$\begin{aligned} x &\equiv a_1 \pmod{n_1} \\ x &\equiv a_2 \pmod{n_2} \\ &\dots \dots \dots \\ x &\equiv a_k \pmod{n_k} \end{aligned}$$

має єдиний розв'язок за модулем $n = n_1 * n_2 * \dots * n_k$.

□

Алгоритм Гауса розв'язку системи лінійних порівнянь з китайської теореми про залишки

Значення x обчислюється наступним чином:

$$x = \sum_{i=1}^k a_i N_i M_i \pmod{n}, \text{ де } N_i = n / n_i, M_i = N_i^{-1} \pmod{n_i}.$$

Приклад. Розв'язати систему порівнянь:

$$\begin{cases} x = 5 \pmod{11} \\ x = 8 \pmod{13} \end{cases}$$

$n = 11 * 13 = 143$. Обчислимо N_i та їх обернені значення M_i :

$$N_1 = 143 / 11 = 13, N_2 = 143 / 13 = 11;$$

$$M_1 = 13^{-1} \pmod{11} = 6, M_2 = 11^{-1} \pmod{13} = 6.$$

Таким чином,

$$x = 5 * 13 * 6 + 8 * 11 * 6 \pmod{143} \equiv 390 + 528 \pmod{143} \equiv 60$$

Відповідь: $x = 60 \pmod{143}$.

Алгоритм Евкліда для цілих чисел

Нехай a і b – цілі числа, не рівні одночасно нулю, і послідовність чисел $a, b, r_1 > r_2 > r_3 > r_4 > \dots > r_n$ визначена тим, що кожне r_k – це залишок від розподілу передпопереднього числа на попереднє, а передостаннє ділиться на останнє націло, тобто:

$$a = bq_0 + r_1$$

$$b = r_1q_1 + r_2$$

$$r_1 = r_2q_2 + r_3$$

...

$$r_{k-2} = r_{k-1}q_{k-1} + r_k$$

...

$$r_{n-1} = r_nq_n.$$

Тоді НСД (a, b), найбільший загальний дільник a і b рівний r_n , останньому ненульовому члену цієї послідовності.

Існування таких $r_1, r_2, \dots$, тобто можливість ділення із залишком m на n для будь-якого цілого m і цілого $n \neq 0$, доводиться індукцією за m .

Коректність цього алгоритму впливає з наступних двох тверджень:

Нехай $a = bq + r$, тоді $\gcd(a, b) = \gcd(b, r)$.

Доведення:

1. Нехай k – будь-який спільний дільник чисел a і b не обов'язково максимальний, тоді

$$a = t_1 * k ; b = t_2 * k;$$

де t_1 і t_2 – цілі числа з визначення.

2. Тоді k також спільний дільник чисел b і r , тому що b ділиться на k відносно a .

$r = a - bq = (t_1 - t_2 * q) * k$ (вираз в дужках є цілим числом, отже k ділить r без залишку).

3. Зворотне твердження також правильне і доводиться аналогічно – будь-який дільник b і r так само є дільником a і b .

4. Всі спільні дільники пар чисел ab і br збігаються.

Отже, немає спільного дільника у чисел ab , який би був так само і дільником br і навпаки.

5. Зокрема максимальний дільник залишається тим же самим.

$$\gcd(0, r) = r \text{ для будь-якого ненульового } r.$$

□

Простіше сформулювати алгоритм Евкліда так: якщо дані натуральні числа a і b і, поки виходить позитивне число, по черзі віднімати від більшого менше, то в результаті вийде НСД.

Розширений алгоритм Евкліда і співвідношення Безу

Формули для r_i можуть бути переписані таким способом:

$$r_1 = a + b(-q_0)$$

$$r_2 = b - r_1q_1 = a(-q_1) + b(1 + q_1q_0)$$

...

$$\gcd(a, b) = r_n = as + bt,$$

де s і t – цілі числа.

Це представлення найбільшого спільного дільника є співвідношенням Безу, а числа s і t – коефіцієнтами Безу. Співвідношення Безу є ключовим у доведенні леми Евкліда і основної теореми арифметики.

Схема Горнера

Схема Горнера (правило Горнера) – алгоритм обчислення значення многочлена, записаного у вигляді суми мономів, при заданому значенні змінної. Метод Горнера дозволяє знайти корінь многочлена, а також

обчислити похідні полінома в заданій точці. Схема Горнера також є простим алгоритмом для розподілу многочлена на біном виду $x - c$.

Опис алгоритму. Заданий многочлен $P(x)$:

$$P(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \dots + a_n x^n, \quad a_i \in \mathbb{R}.$$

Нехай потрібно обчислити значення даного многочлена при фіксованому значенні $x = x_0$. Представимо многочлен $P(x)$ у вигляді:

$$P(x) = a_0 + x(a_1 + x(a_2 + \dots x(a_{n-1} + a_n x) \dots)).$$

Визначимо послідовність:

$$b_n = a_n$$

$$b_{n-1} = a_{n-1} + b_n x$$

...

$$b_i = a_i + b_{i+1} x$$

...

$$b_0 = a_0 + b_1 x.$$

Шукане значення $P(x_0) = b_0$. В отриману форму запису $P(x)$ підставимо $x = x_0$ і будемо обчислювати значення виразу, починаючи з внутрішніх дужок. Для цього будемо замінювати підвираз через b_i :

$$P(x_0) = a_0 + x_0(a_1 + x_0(a_2 + \dots x_0(a_{n-1} + a_n x_0) \dots))$$

$$= a_0 + x_0(a_1 + x_0(a_2 + \dots x_0(a_{n-1} + b_n x_0) \dots))$$

$$= a_0 + x_0(a_1 + x_0(a_2 + \dots x_0(b_{n-1}) \dots))$$

⋮

$$= a_0 + x_0(b_1).$$

Використання схеми Горнера для розподілу многочлена на біном.

При розподілі многочлена $a_0 x^n + a_1 x^{n-1} + \dots + a_{n-1} x + a_n$ на $X - c$ отримуємо многочлен $a_0 x^n + a_1 x^{n-1} + \dots + a_{n-1} x + a_n$ із залишком b_n .

Таким же способом можна визначити кратність кореня (використовувати схему Горнера для нового полінома) і схему можна використовувати для знаходження коефіцієнтів при розкладанні полінома за степенями $X - c$:

$$P(x) = A_0 + A_1(x - c) + A_2(x - c)^2 + \dots + A_n(x - c)^n.$$

Приклад. Розв'язати рівняння $x^4 + 8x^3 + 12x^2 - 12x - 9 = 0$.

Розв'язання. У даному випадку $a_0 = -9, a_n = a_4 = 1$, тому, якщо рівняння має раціональний корінь x_0 , то x_0 дорівнює ± 1 або ± 3 або ± 9 .

Легко перевірити, що $x_0 = 1$ – корінь рівняння. Застосовуючи схему Горнера, маємо:

	1	8	12	-12	-9
1	1	9	21	9	0

Отже, $x^4 + 8x^3 + 12x^2 - 12x - 9 = (x^3 + 9x^2 + 21x + 9)(x - 1)$. Далі шукаємо розв'язки рівняння $x^3 + 9x^2 + 21x + 9 = 0$. Його коренями можуть бути числа ± 1 або ± 3 або ± 9 . Підставляємо ці числа у рівняння і знаходимо корінь $x_1 = -3$. Знову застосовуємо схему Горнера:

	1	9	21	9
-3	1	6	3	0

Отже, $x^3 + 9x^2 + 21x + 9 = (x^2 + 6x + 3)(x + 3)$ і

$$x^4 + 8x^3 + 12x^2 - 12x - 9 = (x^2 + 6x + 3)(x + 3)(x - 1).$$

Залишилося розв'язати квадратне рівняння $x^2 + 6x + 3$. Його коренями є числа:

$$x_{2,3} = \frac{-6 \pm \sqrt{24}}{2} = -3 \pm \sqrt{6}.$$

Відповідь: $x_0 = 1, x_1 = -3, x_2 = -3 + \sqrt{6}, x_3 = -3 - \sqrt{6}$.

Приклад. Знайти $|x + y|$, якщо $\begin{cases} x^2 + y^2 = 3 \\ 2xy = 1 \end{cases}$.

Розв'язання. Додамо рівняння системи. Отримаємо $x^2 + y^2 + 2xy = 3 + 1 = 4$. Отже, $x^2 + y^2 = 4$ і $|x + y| = 2$.

Відповідь: 2.

Приклад. Визначити найменший цілий розв'язок рівняння

$$(x^2 + 2x)^2 - 4(x + 1)^2 + 7 = 0.$$

Розв'язання. Запишемо рівняння у вигляді:

$$\begin{aligned} (x^2 + 2x)^2 - 4(x + 1)^2 + 7 &= (x^2 + 2x + 1 - 1)^2 - 4(x + 1)^2 + 7 = \\ &= ((x + 1)^2 - 1)^2 - 4(x + 1)^2 + 7 = 0. \end{aligned}$$

Позначимо $(x + 1)^2 = t$. Отримаємо:

$$(t - 1)^2 - 4t + 7 = t^2 - 2t + 1 - 4t + 7 = t^2 - 6t + 8 = 0$$

Звідси $t_1 = 4, t_2 = 2$. Отже, маємо два випадки: 1) $(x + 1)^2 = 4$; 2) $(x + 1)^2 = 2$. В першому випадку розв'язку є числа $x_1 = 1, x_2 = -3$. У другому випадку цілих розв'язків немає. Найменший цілий розв'язок рівняння – $x_2 = -3$. *Відповідь:* – 3.

2.1.2. Математичні положення теорії чисел

Прості числа. Функція та теорема Ейлера

Прості числа – це ціле позитивне число, більше одиниці, яке не ділиться без залишку на жодне інше ціле позитивне число, окрім одиниці та самого себе. Усі інші числа, окрім одиниці, називаються складовими. Таким чином, усі натуральні числа, які більші одиниці, розбиваються на прості та складові.

Основна теорема арифметики стверджує, що кожне натуральне число, більше одиниці, представлене у вигляді добутку простих чисел, причому єдиним способом з точністю до порядку проходження співмножників. Таким чином, прості числа – елементарні “будівельні блоки” натуральних чисел [8; 10–12]. Простих чисел нескінченно багато. Найдавніший відомий доказ цього факту був даний Евклідом у “Початках” (книга IX, твердження 20). Його доказ може бути коротко відтворений наступним чином: представимо, що кількість простих чисел звичайна. Перемножимо їх і додамо до них одиницю. Отримане число не ділиться на жодне з остаточного набору простих чисел, тому що залишок від розподілу на кожне з них дає одиницю. Отже, число повинне ділитися на деяке просте число, не включене в цей набір.

Математики пропонували інші докази. Один з них (наведене Ейлером) показує, що сума всіх чисел, зворотних до простих, розходиться. Відома теорема про розподіл простих чисел затверджує, що кількість простих чисел, які менші n і позначувані $\pi(n)$, зростає як $n/\ln(n)$.

Найбільшим відомим простим числом за станом на червень 2009 року є $2^{43112609} - 1$. Воно містить 12 978 189 десяткових цифр і є простим числом Мерсенна ($M_{43112609}$). Його знайшли 23 серпня 2008 року на математичному факультеті університету UCLA у рамках проекту за розподіленням пошуком простих чисел Мерсенна GIMPS. Попереднє число також є простим числом Мерсенна $M_{37156667}$, було знайдено 6 вересня 2007 року учасником проекту GIMPS Гансом-Міхаелем Елвеніхом (Hans-Michael Elvenich). Числа Мерсенна вигідно відрізняються від інших наявністю ефективного тесту простоти: тесту Люка – Лемера. Завдяки йому прості числа Мерсенна давно втримують рекорд як найбільш відомі прості. За знаходження простих чисел з більш ніж 10^8 і 10^9 десяткових

цифр EFF призначила грошові призи відповідно в 150000 і 250000 доларів США. Перша тисяча простих чисел наведена в табл. 2.9.

Таблиця 2.9

Перша тисяча простих чисел

2	3	5	7	11	13	17	19	23	29	31	37	41	43	47	53	59	61	67	71	73	79	83	89	97	101
103	107	109	113	127	131	137	139	149	151	157	163	167	173	179	181	191	193								
197	199	211	223	227	229	233	239	241	251	257	263	269	271	277	281	283	293								
307	311	313	317	331	337	347	349	353	359	367	373	379	383	389	397	401	409								
419	421	431	433	439	443	449	457	461	463	467	479	487	491	499	503	509	521								
523	541	547	557	563	569	571	577	587	593	599	601	607	613	617	619	631	641								
643	647	653	659	661	673	677	683	691	701	709	719	727	733	739	743	751	757								
761	769	773	787	797	809	811	821	823	827	829	839	853	857	859	863	877	881								
883	887	907	911	919	929	937	941	947	953	967	971	977	983	991	997	1009									
1013	1019	1021	1031	1033	1039	1049	1051	1061	1063	1069	1087	1091	1093												
1097	1103	1109	1117	1123	1129	1151	1153	1163	1171	1181	1187	1193	1201												
1213	1217	1223	1229	1231	1237	1249	1259	1277	1279	1283	1289	1291	1297												
1301	1303	1307	1319	1321	1327	1361	1367	1373	1381	1399	1409	1423	1427												
1429	1433	1439	1447	1451	1453	1459	1471	1481	1483	1487	1489	1493	1499												
1511	1523	1531	1543	1549	1553	1559	1567	1571	1579	1583	1597	1601	1607												
1609	1613	1619	1621	1627	1637	1657	1663	1667	1669	1693	1697	1699	1709												
1721	1723	1733	1741	1747	1753	1759	1777	1783	1787	1789	1801	1811	1823												
1831	1847	1861	1867	1871	1873	1877	1879	1889	1901	1907	1913	1931	1933												
1949	1951	1973	1979	1987	1993	1997	1999	2003	2011	2017	2027	2029	2039												
2053	2063	2069	2081	2083	2087	2089	2099	2111	2113	2129	2131	2137	2141												
2143	2153	2161	2179	2203	2207	2213	2221	2237	2239	2243	2251	2267	2269												
2273	2281	2287	2293	2297	2309	2311	2333	2339	2341	2347	2351	2357	2371												
2377	2381	2383	2389	2393	2399	2411	2417	2423	2437	2441	2447	2459	2467												
2473	2477	2503	2521	2531	2539	2543	2549	2551	2557	2579	2591	2593	2609												
2617	2621	2633	2647	2657	2659	2663	2671	2677	2683	2687	2689	2693	2699												
2707	2711	2713	2719	2729	2731	2741	2749	2753	2767	2777	2789	2791	2797												
2801	2803	2819	2833	2837	2843	2851	2857	2861	2879	2887	2897	2903	2909												
2917	2927	2939	2953	2957	2963	2969	2971	2999	3001	3011	3019	3023	3037												
3041	3049	3061	3067	3079	3083	3089	3109	3119	3121	3137	3163	3167	3169												
3181	3187	3191	3203	3209	3217	3221	3229	3251	3253	3257	3259	3271	3299												
3301	3307	3313	3319	3323	3329	3331	3343	3347	3359	3361	3371	3373	3389												
3391	3407	3413	3433	3449	3457	3461	3463	3467	3469	3491	3499														

3511	3517	3527	3529	3533	3539	3541	3547	3557	3559	3571	3581	3583	3593
3607	3613	3617	3623	3631	3637	3643	3659	3671	3673	3677	3691	3697	3701
3709	3719	3727	3733	3739	3761	3767	3769	3779	3793	3797	3803	3821	3823
3833	3847	3851	3853	3863	3877	3881	3889	3907	3911	3917	3919	3923	3929
3931	3943	3947	3967	3989	4001	4003	4007	4013	4019	4021	4027	4049	4051
4057	4073	4079	4091	4093	4099	4111	4127	4129	4133	4139	4153	4157	4159
4177	4201	4211	4217	4219	4229	4231	4241	4243	4253	4259	4261	4271	4273
4283	4289	4297	4327	4337	4339	4349	4357	4363	4373	4391	4397	4409	4421
4423	4441	4447	4451	4457	4463	4481	4483	4493	4507	4513	4517	4519	4523
4547	4549	4561	4567	4583	4591	4597	4603	4621	4637	4639	4643	4649	4651
4657	4663	4673	4679	4691	4703	4721	4723	4729	4733	4751	4759	4783	4787
4789	4793	4799	4801	4813	4817	4831	4861	4871	4877	4889	4903	4909	4919
4931	4933	4937	4943	4951	4957	4967	4969	4973	4987	4993	4999	5003	5009
5011	5021	5023	5039	5051	5059	5077	5081	5087	5099	5101	5107	5113	
5119	5147	5153	5167	5171	5179	5189	5197	5209	5227	5231	5233	5237	5261
5273	5279	5281	5297	5303	5309	5323	5333	5347	5351	5381	5387	5393	5399
5407	5413	5417	5419	5431	5437	5441	5443	5449	5471	5477	5479	5483	5501
5503	5507	5519	5521	5527	5531	5557	5563	5569	5573	5581	5591	5623	5639
5641	5647	5651	5653	5657	5659	5669	5683	5689	5693	5701	5711	5717	5737
5741	5743	5749	5779	5783	5791	5801	5807	5813	5821	5827	5839	5843	5849
5851	5857	5861	5867	5869	5879	5881	5897	5903	5923	5927	5939	5953	5981
5987	6007	6011	6029	6037	6043	6047	6053	6067	6073	6079	6089	6091	6101
6113	6121	6131	6133	6143	6151	6163	6173	6197	6199	6203	6211	6217	6221
6229	6247	6257	6263	6269	6271	6277	6287	6299	6301	6311	6317	6323	6329
6337	6343	6353	6359	6361	6367	6373	6379	6389	6397	6421	6427	6449	6451
6469	6473	6481	6491	6521	6529	6547	6551	6553	6563	6569	6571	6577	6581
6599	6607	6619	6637	6653	6659	6661	6673	6679	6689	6691	6701	6703	6709
6719	6733	6737	6761	6763	6779	6781	6791	6793	6803	6823	6827	6829	6833
6841	6857	6863	6869	6871	6883	6899	6907	6911	6917	6947	6949	6959	6961
6967	6971	6977	6983	6991	6997	7001	7013	7019	7027	7039	7043	7057	7069
7079	7103	7109	7121	7127	7129	7151	7159	7177	7187	7193	7207	7211	7213
7219	7229	7237	7243	7247	7253	7283	7297	7307	7309	7321	7331	7333	7349
7351	7369	7393				7411	7417	7433		7451		7457	

7459	7477	7481	7487	7489	7499	7507	7517	7523	7529	7537	7541	7547
7549	7559	7561	7573	7577	7583	7589	7591	7603	7607	7621	7639	7643
7649	7669	7673	7681	7687	7691	7699	7703	7717	7723	7727	7741	7753
7757	7759	7789	7793	7817	7823	7829	7841	7853	7867	7873	7877	7879
7883	7901	7907	7919									

З визначення простого числа випливає, що будь-яке ціле число $v > 1$ завжди має хоча б два дільники 1 і v . Якщо окрім дільників 1 і v число v не має інших додатних дільників, то воно називається простим. Числа, більші від 1, що мають цілі додатні дільники, відмінні від 1 і самого себе, називаються складними.

Складене число – натуральне число, більше 1, що не є простим. Кожне складене число є добутком двох натуральних чисел, які більші 1.

Послідовність складених чисел починається так:

4, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, ... (послідовність A002808 в OEIS).

Основна теорема арифметики наголошує, що будь-яке складене число може бути єдиним способом розкладене в добуток простих множників.

Будь-яке довільне ціле число $v > 1$ єдиним чином розкладається на добуток попарно різних простих чисел q_i або їх степенів:

$$v = q_1^{\beta_1} q_2^{\beta_2} \dots q_k^{\beta_k}.$$

Розглянемо визначення функції Ейлера. Для будь-якого натурального числа α функція Ейлера $\varphi(\alpha)$ визначається як кількість чисел, обраних із сукупності 1, 2, 3, ..., α , що є взаємoprостими с α .

Приклад. Обчислити функцію Ейлера для чисел, які менші 8.

Розв'язання:

$\varphi(1) = 1$ ($X = 1$), $\varphi(2) = 1$ ($X = 1$); $\varphi(3) = 2$ ($X = 1, 2$), $\varphi(4) = 2$ ($X = 1, 3$);
 $\varphi(5) = 4$ ($X = 1, 2, 3, 4$), $\varphi(6) = 2$ ($X = 1, 5$); $\varphi(7) = 6$ ($X = 1, 2, 3, 4, 5, 6$).

Процедура обчислення значень функції Ейлера спрощується, якщо скористатися наступними теоремами.

Теорема 2.6. Функція Ейлера мультиплікативна, тобто:

$$\varphi(a, b) = \varphi(a) \varphi(b), \quad (2.5)$$

при взаємно простих a і b .

Теорема 2.7. Нехай p – просте число, $\alpha \geq 1$ – будь-яке натуральне.

Тоді:

$$\varphi(p^\alpha) = p^{\alpha-1}(p-1). \quad (2.6)$$

Наслідок виразу 2.56:

$$\varphi(p) = p - 1.$$

Теорема 2.8. При $m > 1$

$$\varphi(m) = \prod_{p|m} \left(1 - \frac{1}{p}\right). \quad (2.7)$$

Теорема 2.9.

$$\sum_j \varphi(d) = m. \quad (2.8)$$

З теореми випливає, що сума значень $\varphi(d)$ за всіма d , які ділять a , дорівнює a .

Теорема 2.10. Якщо $m = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_s^{\alpha_s}$ є розкладанням числа $\overline{m}$, тоді:

$$\varphi(m) = p_1^{\alpha_1-1} p_2^{\alpha_2-1} \dots p_s^{\alpha_s-1} = (p_1 - 1)(p_2 - 1) \dots (p_s - 1).$$

Приклад. Обчислити значення функції Ейлера для числа $L = 270$.

Розв'язання:

$$\varphi(270) = \varphi(2 \times 3^3 \times 5) = 3^2 (2 - 1) (3 - 1) (5 - 1) = 72.$$

Приклад. Обчислити значення функції Ейлера для числа $L = 700000$.

Розв'язання:

$$\varphi(700\,000) = \varphi(2^5 \times 5^5 \times 7) = 2^4 5^4 (5 - 1) (7 - 1) = 240\,000.$$

Теорема 2.11 (Ейлера). Якщо $m > 1$ і $(a, m) = 1$, то:

$$a^{\varphi(m)} \equiv 1 \pmod{m}, \quad (2.9)$$

таким чином, $a^{\varphi(m)} - 1$ ділиться на m .

□

Виставлення натурального числа у вигляді добутку простих чисел називається розкладанням на прості або факторизацією числа. На сьогоднішній день невідомі поліноміальні алгоритми факторизації чисел, хоча і не доведено, що таких алгоритмів не існує (тут і далі йдеться про поліноміальну залежність часу роботи алгоритму від логарифма числа,

що перевіряється, тобто від кількості його цифр). На передбачуваний завданням факторизації базується криптосистема RSA.

Тести на простоту та алгоритми розкладу на прості множники

Тест простоти – алгоритм, який за заданим натуральним числом визначає, чи просте це число. Розрізняють детерміновані та імовірнісні тести. Визначення простоти заданого числа в загальному випадку не таке вже й тривіальне завдання. Тільки в 2002 році було доведено, що вона поліноміально розв'язна. Проте тестування простоти значно легше факторизації заданого числа.

Розглянемо наступні тести простоти:

перебір дільників;

теорема Вільсона;

тест Міллера-Рабіна;

тест Люка-Лемера для чисел Мерсена;

тест Пепіна для чисел Ферма;

тест Агравала-Каяла-Саксени (Тест AKS), перебір дільників.

Перебір дільників – алгоритм факторизації або тестування простоти числа шляхом повного перебору всіх можливих потенційних дільників.

Звичайно перебір дільників полягає в переборі всіх цілих (як варіант: простих) чисел від 2 до квадратного кореня числа n , яке факторизується і в обчисленні залишку від розподілу n на кожне з цих чисел. Якщо залишок від розподілу на деяке число m дорівнює нулю, то m є дільником n . У цьому випадку або n оголошується складовим, і алгоритм закінчує роботу (якщо тестується простота n), або n скорочується на m і процедура повторюється (якщо здійснюється факторизація n). З вилучення квадратного кореня з n і неможливості скоротити n на жодне з менших чисел n оголошується простим.

Реалізація тестування простоти на C++

```
bool is_prime(int n)
{
    if (n <= 1)
        return false;
    for (int j = 2; j * j <= n; j++)
        if (n % j == 0) return false;
    return true;
}
```


Теорема Вільсона – теорема теорії чисел, яка затверджує, що p – просте число тоді, коли $(p - 1)! + 1$ ділиться на p .

Практичне використання теореми Вільсона для визначення простоти числа недоцільне через складність обчислення факторіалу.

Реалізація тестування простоти на C++

```
int factorial(int x) {  
    if( x == 0 ) return 1;  
    return x * factorial (x - 1);  
}  
bool simpleint (int p)  
{  
    return ((factorial (p-1)+1)%p==0);  
},
```

де функція `factorial ()` знаходить факторіал рекурсивним методом, а `simpleint ()` – вважає логічний вираз і повертає `false`, якщо число складене, `true`, якщо число просте.

Тест Міллера – Рабіна – імовірнісний поліноміальний тест простоти. Тест Міллера – Рабіна дозволяє ефективно визначати, чи є дане число складовим. Однак з його допомогою не можна суворо довести простоту числа. Проте тест часто використовується в криптографії для одержання більших випадкових простих чисел.

Алгоритм був розроблений Гарі Міллером у 1976 і модифікований Майклом Рабіном у 1980 році.

Нехай непарне число більше 1. Число $m - 1 = 2^s t$, де t непарне.

Ціле число a , $1 < a < m$ називається свідком простоти числа m , якщо виконуються дві вимоги:

m не ділиться на a ;

$a^t \equiv 1 \pmod{m}$ або є число k , таке що $a^{2^k t} \equiv -1 \pmod{m}$.

Теорема Рабіна стверджує, що складене непарне число m має не більше $\phi(m)/4$ різних свідків простоти, де $\phi(m)$ – функція Ейлера.

Алгоритм Міллера – Рабіна параметризується кількістю раундів r . Рекомендується брати r порядком величини $\log_2(m)$, де m – число, що перевіряється.

Для даного m є ціле число s і ціле непарне число t , що $m - 1 = 2^s t$. Обирається випадкове число a , $1 < a < m$. Якщо a не є свідком простоти числа m , то видається відповідь “ m складене”, і алгоритм завершується. Інакше, обирається нове випадкове число a і процедура

перевірки повторюється. Після знаходження r -свідків простоти видається відповідь “ m , імовірно просте”, і алгоритм завершується.

Алгоритм може бути записаний на псевдокоді в такий спосіб:

введення: $m > 2$, непарне натуральне число, яке необхідно перевірити на простоту;

r , параметр, що визначає точність тесту.

Висновок: складене, означає, що m точно складене;

імовірно просте, означає, що m з високою ймовірністю є простим.

Представити $m - 1$ у вигляді $2^s \cdot t$, де t непарне, можна зробити послідовним діленням $m - 1$ на 2.

цикл A: повторити r разів:

Вибрати випадкове a у діапазоні $[2, m - 2]$

$x \leftarrow at \bmod m$

якщо $x = 1$ або $x = m - 1$ то перейти на наступну ітерацію циклу A

для $r = 1 \dots s - 1$

$x \leftarrow x^2 \bmod m$

якщо $x = 1$, то повернути складене

якщо $x = m - 1$, то перейти на наступну ітерацію циклу A

повернути складене

повернути ймовірно просте.

З теореми Рабіна випливає, що якщо r з випадково обраних чисел виявилось свідком простоти числа m , то ймовірність того, що m складене, не перевершує $4 - r$.

Тест Люка – Лемера – ефективний тест простоти для чисел Мерсена. Цей тест був запропонований Люка в 1878 році та в 1930 році вдосконалений Лемером.

Число Мерсена – числа виду $M_p = 2^p - 1$, де p – натуральне число, названі на честь французького математика Мерсена.

Послідовність чисел Мерсена починається так:

1, 3, 7, 15, 31, 63, 127, 255, 511, 1023, ... (послідовність A000225 в OEIS).

Іноді числами Мерсена називають числа M_p із простими індексами p . Ця послідовність починається так:

3, 7, 31, 127, 2047, 8191, 131071, 524287, 8388607, 536870911, 2147483647...

Тест Люка – Лемера базується на тому, що простота числа Мерсенна $M_p = 2^p - 1$ тягне простоту числа p у твердженні, що p впливає.

Для простого числа p число M_p є простим тільки тоді, коли воно ділить число $L_p - 1$, де числа L_k визначаються рекурентним співвідношенням: $L_1 = 4$, $L_k = 1 + L_{k-2}^2$.

Для встановлення простоти M_p послідовність чисел $L_1, L_2, \dots, L_{p-1}$ достатньо обчислювати за модулем числа M_p (тобто обчислювати не самі числа L_k , довжина яких зростає експоненціально; а залишки від ділення L_k на M_p , довжина яких обмежена p -бітами). Останнє число в цій послідовності $L_{p-1} \bmod M_p$ називається відрахуванням Люка – Лемера. Таким чином, число Мерсенна M_p є простим тільки тоді, коли число p просте і відрахування Люка – Лемера дорівнює нулю.

Можливими значеннями L_1 є: 4, 10, 52, 724, 970, ... (послідовність A018844 в OEIS).

Завдяки тесту Люка – Лемера прості числа Мерсена втримують лідерство як найбільш відомі прості числа. Саме тест Люка – Лемера лежить в основі проекту розподілених обчислень GIMPS, що займається пошуком нових простих чисел Мерсена.

Тест Пепіна – тест простоти для чисел Ферма.

Тест Пепіна полягає у зведенні числа 3 у степінь $(F_n - 1)/2 = 2^{2^n - 1}$ (серією з $2^n - 1$ послідовних зведень у квадрат) за модулем F_n і порівнянні результату с – 1.

Псевдокод

$b \leftarrow 3$

від $i = 1$ до $2^n - 1$ виконати $b \leftarrow b^2 \bmod F_n$

якщо $b \equiv -1 \pmod{F_n}$, то

повернення “ F_n – просте”

інакше

повернення “ F_n – складене”

Тест Пепіна базується на наступному наслідку квадратичного закону взаємності: число 3 є первісним коренем за модулем кожного простого числа Ферма.

Тому число Ферма $F_n = 2^{2^n} + 1$ є простим тільки тоді, коли $3^{(F_n-1)/2} \equiv -1 \pmod{F_n}$.

Число 3 у тесті Пепіна може бути замінене на 5, 6, 7 або 10 (послідовність A129802 в OEIS), які також є первісними коренями за модулем кожного простого числа Ферма. Числа Ферма – числа виду $F_n = 2^{2^n} + 1$. Вивчення чисел такого виду почав Ферма, який висунув

гіпотезу, що всі вони прості. Однак ця гіпотеза була спростована Ейлером у 1732 році, який винайшов розкладання числа F_5 на прості дільники:

$$F_5 = 4294967297 = 641 \times 6700417.$$

Тест Агравала – Каяла – Саксені (або тест AKS)

В інформатиці тест Агравала – Каяла – Саксені (або тест AKS) – це поліноміальний детермінований тест простоти чисел, запропонований індійськими вченими Маніндрой Агарвалом, Ніраджем Каялом і Нітіном Саксеной та вперше опублікований 6 серпня 2002 року в статті “PRIMES is in P”. До цієї публікації завдання розпізнавання простоти класу P було відкритою проблемою. За своє відкриття автори одержали премію Годеля в 2006 році. У 2005 році був запропонований варіант алгоритму, який виконується за $O(\log^6 n + \epsilon(n))$ операцій, де n – випробуване число.

Теорема AKS. Нехай $n \geq 2$; n – ціле число; q, r – прості числа, причому

$$\forall m \in \{1, 2, \dots, r\} : \text{НСД}(m, n) = 1$$

$$q \mid (r-1)$$

$$q \geq 4 \sqrt{r} \log n$$

$$n^{(r-1)/q} \not\equiv 1 \pmod{r}$$

$$\forall a \in \{1, \dots, [2\sqrt{r} \log n] + 1\} : (x - a)^2 = (x^2 - a) \pmod{n, X^r - 1}$$

Алгоритми AKS

$$r = 2$$

while ($r < n$) {

 if ($\text{НСД}(r, n) \neq 1$) output складене;

 if (r – просте число, $r > 2$) {

$q =$ найбільший простий дільник u ($r - 1$);

 if ($(q \geq 4\sqrt{r} \log n)$ and $(n^{(r-1)/q} \not\equiv 1 \pmod{r})$ break;

 }

$r \leftarrow r + 1$;

 for $a = 1$ to $([2\sqrt{r} \log n] + 1)$

 if ($(x - a)^2 = (x^2 - a) \pmod{n, X^r - 1}$) output складене;

 if ($n = a^b$; a, b – цілі; $a, b \geq 2$) output складене;

 else output просте.

Розкладання числа на прості множники

Розкладання на множники числа (наприклад, $252601 = 41 \times 61 \times 101$) є однією з найдавніших проблем теорії чисел. Приступаючи до факторизації числа, слід спочатку переконатися, що воно не просте. Це можна зробити за допомогою одного з описаних вище тестів простоти. Їх трудомісткість, як правило, значно менше, ніж в алгоритмах факторизації [11-14]. Найкращими алгоритмами факторизації є:

Решето загального числового поля (найшвидший з відомих алгоритмів для чисел розміром 110 і більше десяткових розрядів).

Квадратичне решето для чисел розміром (для чисел менше 110 десяткових розрядів; в 1994 р. за 8 місяців на 1600 процесорах було розкладено 129-розрядне число).

Метод еліптичної кривої

Монте-Карло Поларда

Метод безперервних дробів (за часом не підходить для реалізації).

Перевірка розподілом (найдавніший алгоритм розкладання; полягає в перевірці на розподіл на кожне просте число, менше або рівне квадратному кореню із числа, що розкладається).

Майже всі ці алгоритми субекспоненціальної складності. Алгоритмів факторизації, що мають оцінки складності краще зазначеної, у цей час не існує. Розглянемо більш прості методи, що мають експонентну складність.

Метод Ферма. Даний метод заснований на пошуку чисел x і y , таких, що $x^2 \equiv y^2 \pmod{n}$, де n треба розкласти на множники.

Теорема 2.12. (Теорема Ейлера про подання числа у вигляді різниці квадратів). Якщо $n > 1$ – непарне, то існує взаємно однозначна відповідність між розкладаннями на множники $n = a \cdot b$ і виразом у вигляді різниці квадратів $n = x^2 - y^2$, $x > y > 0$.

Тут
$$x = \frac{a+b}{2}; y = \frac{a-b}{2}, a = x+y; b = x-y.$$

Метод Ферма полягає у тому, що при малих значеннях параметра y у виразі $n = x^2 - y^2$ можна знайти пару (x, y) , перебираючи в якості кандидатів на значення x числа $\lfloor \sqrt{n} + 1 \rfloor, \lfloor \sqrt{n} + 2 \rfloor, \dots$ і перевіряючи для кожного з них рівності $(\lfloor \sqrt{n} + j \rfloor)^2 - n = y^2$.

Алгоритм факторизації методом Ферма:

Вхід: n – непарне число, $p_1, \dots, p_k$ – невеликі прості числа.

Перевірити, чи ділять націло $p_i, i = \overline{1, k}$ число n . Якщо так, то дільник знайдений.

$$\forall x \in [\lfloor \sqrt{n} \rfloor + 1; \lfloor \sqrt{n} \rfloor + n_0] \text{ обчислити величини} \\ t \leftarrow x^2 - n, t_i \leftarrow t \bmod p_i, i = \overline{1, k}.$$

Якщо хоча б для одного $i = \overline{1, k}$ виконана одна з умов:

$t_i = 0$ і p_i^2 не ділить t

$$t_i \neq 0 \text{ і } \left(\frac{t_i}{p_i} \right) = -1,$$

перейти до наступного x на кроці 2. А якщо ні, то перейти до кроку 4.

Перевірити, чи є $t \leftarrow x^2 - n$ повним квадратом. Якщо $x^2 - n = y^2$, то n – складене; якщо $t \leftarrow x^2 - n$ – не повний квадрат, то перейти до наступного x на кроці 2.

Інший варіант методу Ферма запропонував Кнут (без операцій розподілу).

Нехай $n = UV$, де $U \leq V$.

Допустимо, що n – непарне, тоді U, V теж непарні. Тому можна покласти $x = \frac{U+V}{2}; y = \frac{U-V}{2}, n = x^2 - y^2, 0 \leq y - x \leq n$. Метод Ферма полягає в пошуку x та y задовольняючи цим співвідношенням. Тоді $n = (x - y)(x + y)$. Наступний алгоритм показує, як, не виконуючи операцій розподілу, можна реалізувати метод Ферма:

$x' \leftarrow 2\lfloor \sqrt{n} \rfloor + 1; y' \leftarrow 1; r \leftarrow \lfloor \sqrt{n} \rfloor^2 - n$, під час роботи алгоритму x', y', r відповідають відповідно $2x + 1, 2y + 1, x^2 + y^2 - n^2$.

Якщо $r \leq 0$, то перейти до кроку (4).

$r \leftarrow r - y'; y' \leftarrow y' + 2$ і повернутися на крок (2).

Якщо $r = 2$, то кінець циклу: маємо $m = \frac{x' - y'}{2} \cdot \frac{x' + y' - 2}{2}$ і значення

$\frac{x' - y'}{2}$ є найбільшим множником числа $m \leq \sqrt{n}$.

$r \leftarrow r + x'; x' \leftarrow x' + 2$ і повернутися на крок (3).

Кількість кроків, виконаних цим алгоритмом для знаходження U, V , пропорційно $\sqrt{n}$.

Метод Ферма ефективний, якщо дільники n близькі один до одного. Даний метод є, за своєю суттю, переборним, тобто неефективним.

Метод $(p-1)$ – факторизації Поларда. Допустимо, що n – непарне складене число, що не має невеликих простих дільників. Позначимо через p найменший простий дільник числа n . Наше завдання полягає в його знаходженні.

Допустимо, що число $(p-1)$ розкладається на добуток невеликих простих дільників. Оберемо число k , яке є параметром методу. Для успішної роботи алгоритму потрібно, щоб виконувалася умова ділення $M(k):(p-1)$, де $M(k) = \text{НЗК}(1, 2, \dots, k)$ (замість $M(k)$, наприклад, можна використовувати $k!$).

Відповідно до малої теореми Ферма виконується порівняння $2^{M(k)} \equiv 1 \pmod{p}$. Якщо при цьому $2^{M(k)} \not\equiv 1 \pmod{n}$, то p ділить $\text{НСД}(2^{M(k)} - 1, n)$, де $p > 1$, $\text{НСД}(2^{M(k)} - 1, n) < n$.

Таким чином, $\text{НСД}(2^{M(k)} - 1, n)$ є дільником числа n , кратним p .

Через те що число k невідоме, воно шукається способом за алгоритмом:

Алгоритм методу факторизації $(p-1)$ – факторизація Поларда:

Нехай k – ціле число, наприклад, $k < 10^6$, і c – невелике ціле з умовою $\text{НСД}(c, n) = 1$, наприклад, $c = 2$.

$\forall i = \overline{1, k}$ обчислюється $m_i \leftarrow c^{M(i)} \pmod{n}$ і перевіряється тест кроку 2.

Обчислити $d \leftarrow \text{НСД}(m_i - 1, n)$. Якщо $1 < d < n$, то знайдений нетривіальний дільник числа n . А якщо ні, то вважається $i \leftarrow i + 1$.

Оцінка складності даного методу в найгіршому разі становить $O(n^{1/2} \log^{\text{const}} n)$ арифметичних операцій. Однак у деяких випадках алгоритм може швидко видати дільник числа n .

На практиці $(p-1)$ метод Поларда звичайно використовують до застосування більш сильних алгоритмів факторизації для того, щоб відокремити невеликі прості дільники числа n .

Метод p-Поларда. Далі наведений алгоритм цього методу:

Вибираємо випадковим способом x_1 з множини $\{0, 1, \dots, n-1\}$; $y \leftarrow x_1$;
 $k \leftarrow 2$. $i \leftarrow i+1$ і обчислюємо наступний елемент послідовності
 $x_i \leftarrow f(x_{i-1}) \bmod n$.

Якщо $d \leftarrow \text{НЗД}(y - x_i, n)$: $d \neq 1$ і $d \neq n$, то d – дільник n , цикл завершений.

Якщо i дорівнює k , то $y \leftarrow x_i$, $k \leftarrow 2k$.

Можливо, що цикл значень за модулем n виявиться більше, ніж $\sqrt{n}$. Метод має евристичну оцінку складності $O(n^{1/4})$ арифметичних операцій. Він дуже популярний і звичайно використовується для відділення невеликих простих дільників числа n , яке факторизується.

Основна ідея даного методу дуже проста. Якщо період послідовності $x_i \bmod n$ може бути порядком n , то період послідовності $x_i \bmod p$ для простого дільника p числа n не перевершує p . Це значить, що y та x_i можуть бути різними за модулем n , але збігатися за модулем p .

Існує константа c така, що для будь-якого $\lambda > 0$ ймовірність не знайти нетривіальний дільник n за $c\sqrt{\lambda}n^{1/4} \log^3 n$ бітових операцій буде меншою, ніж $e^{-\lambda}$.

Метод Шермана – Лемана. Нехай n – непарне, $n > 8$.

Для $a = 2, 3, \dots, \lfloor n^{1/3} \rfloor$ перевірити, що a ділить n . Якщо на цьому кроці не розкладається n на множники, то переходимо до кроку 2.

Якщо на 1 кроці дільник не знайдений і n – складене, то $n = pq$, де p, q – прості числа, і $n^{1/3} < p \leq q < n^{2/3}$.

Тоді для всіх $k = 1, 2, \dots, \lfloor n^{1/3} \rfloor$ і всіх $d = 0, 2, \dots, \left\lfloor \frac{n^{1/6}}{4\sqrt{k}} \right\rfloor + 1$ перевірити, чи є

число $\left(\left\lfloor \sqrt{4kn} \right\rfloor + d \right)^2 - 4kn$ квадратом натурального числа. Якщо є, то для $A = \left\lfloor \sqrt{4kn} \right\rfloor + d$ і $B = \sqrt{A^2 - 4kn}$ виконане порівняння $A^2 \equiv B^2 \pmod{n}$.

У цьому випадку перевірити умову $1 < \text{НСД}(A \pm B, n) < n$.

Якщо ця умова виконана, то n розкладається на два множники і алгоритм зупиняється.

Якщо даний алгоритм не знайшов розкладання n на два множники, то n – просте число.

Даний алгоритм розкладає n на множники за $O(n^{1/3})$ арифметичних операцій.

Метод Ленстрі.

Теорема 2.13. Нехай r, s, n – натуральні числа, що задовольняють умовам $1 \leq r < s < n, n^{1/3} < s, \text{НСД}(r, s) = 1$.

У такому випадку існує не більше 11 дільників r_i числа n таких, що $r_i \equiv r \pmod{s}$. Є алгоритм, який знаходить усі ці r_i за $O(\log n)$ арифметичних операцій.

Алгоритм методу факторизації Ленстрі.

На вході задані числа $r, s, n \in \mathbb{N}$, що задовольняють умовам теореми.

За допомогою узагальненого алгоритму Евкліда знайти $r^* \in \mathbb{N}$ $r^* r \equiv 1 \pmod{s}$.

Знайти r' таке, що $r' \equiv r^* n \pmod{s}, 0 \leq r' < s$.

Для чергового значення $i = 0, 1, \dots$ знайти числа за наступними правилами:

$$a_0 = s, b_0 = 0, c_0 = 0,$$

$$a_1 \equiv r' r^* \pmod{s}, 0 < a_1 \leq s, b_1 = 1, c_1 \equiv \frac{n - r r'}{s} r^* \pmod{s}$$

$$\forall i \geq 2:$$

$$a_i = a_{i-2} - q_i a_{i-1}, b_i = b_{i-2} - q_i b_{i-1}, c_i \equiv c_{i-2} - q_i c_{i-1} \pmod{s}$$

$$\text{де } q_i = \begin{cases} 0 \leq q_i < a_{i-1}, i - \text{парне} \\ 0 < q_i \leq a_{i-1}, i - \text{непарне.} \end{cases}$$

Для чергового набору a_i, b_i, c_i знайти всі цілі числа c , що задовольняють умовам:

$$c \equiv c_i \pmod{s}$$

$$|c| < s, i - \text{парне}$$

$$2a_i b_i \leq c \leq \frac{n}{s^2} + a_i b_i, i - \text{непарне.}$$

Таких c буде не більше двох.

Для кожного c із кроку 3 розв'язати в цілих числах систему рівнянь:

$$\begin{cases} x a_i + y b_i = c \\ (x s + r)(y s + r') = n. \end{cases}$$

Якщо x та y виявляться не негативними цілими числами, то додати $(xs + r)$ до списку шуканих дільників.

Якщо $a_i = 0$, то алгоритм закінчує роботу. Інакше повертаємось на крок 2 до наступного значення $i \leftarrow i + 1$.

Проблема дискретного логарифмування

Проблема обчислення дискретного логарифма є не лише цікавою, а і вкрай корисною для систем захисту інформації. Ефективний алгоритм знаходження дискретного логарифма значною мірою знизив би безпеку систем ідентифікації користувача та схеми обміну ключів.

Означення. Нехай G – скінченна циклічна група порядку n . Нехай g – генератор G та $b \in G$. Дискретним логарифмом числа b за основою g називається таке число x ($0 \leq x \leq n - 1$), що $g^x = b$ та позначається $x = \log_g b$.

Проблема дискретного логарифма. Нехай p – просте число, g – генератор множини Z_p^* , $y \in Z_p^*$. Знайти таке значення x ($0 \leq x \leq p - 2$), що $g^x \equiv y \pmod{p}$. Число x називається дискретним логарифмом числа y за основою g та модулем p .

Узагальнена проблема дискретного логарифма. Нехай G – скінченна циклічна група порядку n , g – її генератор, $b \in G$. Необхідно знайти таке число x ($0 \leq x \leq n - 1$), що $g^x = b$.

Розширенням узагальненої проблеми може стати задача розв'язку рівняння $g^x = b$, коли знято умову циклічності групи G , а також умову того, що g – генератор G (у такому випадку рівняння може і не мати розв'язку).

Приклад. $g = 3$ є генератором Z_7^* : $3^1 = 3$, $3^2 = 2$, $3^3 = 6$, $3^4 = 4$, $3^5 = 5$, $3^6 = 1$.

$\log_3 4 = 4 \pmod{7}$, тому що розв'язком рівняння $3^x = 4$ буде $x = 4$.

Теорема. 2.14. Нехай a – генератор скінченної циклічної групи G порядку n . Якщо існує алгоритм, який обчислює дискретний логарифм за основою a , то цей алгоритм може також обчислити дискретний логарифм за будь-якою основою b , яка є генератором G .

Доведення. Нехай $k \in G$, $x = \log_a k$, $y = \log_b k$, $z = \log_a b$. Тоді $a^x = b^y = (a^z)^y$, звідки $x = zy \pmod{n}$. Підставимо в останню рівність замість змінних логарифмічні вирази:

$$\log_a k = (\log_a b) (\log_b k) \pmod{n}$$

або

$$\log_b k = (\log_a k) (\log_a b)^{-1} \pmod{n}.$$

З останньої рівності випливає справедливність теореми.

Примітивний алгоритм

Для знаходження $\log_g b$ (g – генератор G порядку n , $b \in G$) будемо обчислювати значення $g, g^2, g^3, g^4, \dots$, поки не отримаємо b . Часова оцінка алгоритму – $O(n)$. Якщо n – велике число, то час обчислення логарифму є достатньо великим і тому алгоритм є неефективним.

Алгоритм великого та малого кроку

Першим детермінованим алгоритмом для обчислення дискретного логарифма був алгоритм великого та малого кроку, запропонований Шанком (Shank).

Для обчислення $\log_g b$ в групі Z_n^* необхідно зробити наступні кроки:

1. Обчислити $a = \lfloor \sqrt{n} \rfloor$.
2. Побудувати список $L_1 = 1, g^a, g^{2a}, \dots, g^{a^2}$ (за модулем n).
3. Побудувати список $L_2 = b, bg, bg^2, \dots, bg^{a-1}$ (за модулем n).
4. Знайти число z , яке зустрілося в L_1 та L_2 .

Тоді $z = bg^k = g^{la}$ для деяких k та l . Звідси $b = g^{la-k} = g^x$; $x = la - k$.

Два питання постає при дослідженні роботи наведеного алгоритму:

1. Чи завжди знайдеться число, яке буде присутнім в обох списках?
2. Як ефективно знайти значення z ?

Запишемо $x = sa + t$ для деяких s, t таких, що $0 \leq s, t < a$. Тоді $b = g^x = g^{sa+t}$. Помножимо рівність на g^{a-t} , отримаємо: $bg^{a-t} = g^{s(a+1)}$. Значення зліва обов'язково зустрінесться в L_2 , а справа – в L_1 .

Відсортуємо отримані списки L_1 та L_2 за час $O(a * \log a)$. За лінійний час проглядаємо списки зліва направо, порівнюючи їх голови: якщо вони рівні, то значення z знайдене, якщо ні – то видалити менше число і продовжити перевірку.

Приклад. Розв'язати рівняння: $2x \equiv 11 \pmod{13}$.

$$a = \lfloor \sqrt{13} \rfloor = 4;$$

$$L_1: 1, 2^4 \equiv 3, 2^8 \equiv 9, 2^{12} \equiv 1, 2^{16} \equiv 3;$$

$$L_2: 11, 11 * 2 \equiv 9, 11 * 2^2 \equiv 5, 11 * 2^3 \equiv 10;$$

Число 9 зустрілось в обох списках. $11 * 2 \equiv 2^8$, $11 \equiv 2^7$, звідки $x = 7$.

Відповідь: $x = 7$.

Інший підхід до реалізації алгоритму великого та малого кроку можна отримати, якщо рівність $b = g^{sa+t}$ ($a = \lfloor \sqrt{n} \rfloor$, $0 \leq s, t < a$) переписати у вигляді $b * (g^{-a})^s = g^t$. Обчислимо g^{-a} та складемо таблицю значень g^t ,

$0 \leq t < a$. Далі починаємо знаходити значення $b \cdot (g^{-a})^s$, $s = 0, 1, \dots$ перевіряючи їх наявність у таблиці g^t . Як тільки знаходяться такі s та t , алгоритм зупиняється.

Приклад. Обчислити $\log_2 3$ в групі Z_{19}^* .

$3 = 2^x = 2^{sa+1}$, $3 \cdot (2^{-a})^s = 2^t$. Складемо таблицю 2^t , $0 \leq t < \lfloor \sqrt{19} \rfloor = 5$:

t	0	1	2	3	4
2^t	1	2	4	8	16

$2^{-1} \equiv 10 \pmod{19}$, оскільки $2 \cdot 10 \equiv 1 \pmod{19}$.

Тоді $3 \cdot (2^{-5})^s \pmod{19} \equiv 3 \cdot (10^5)^s \pmod{19} \equiv 3 \cdot 3^s \pmod{19}$.

Обчислюємо $3 \cdot 3^s$, $s = 0, 1, \dots$:

s	0	1	2
$3 \cdot 3^s$	3	9	8

Значення 8, яке отримали при $s = 2$, присутнє в таблиці 2^t , $0 \leq t < 5$.

Звідси $3 \cdot (2^{-5})^2 = 2^3$ або $3 = (2^5)^2 \cdot 2^3 = 2^{5 \cdot 2 + 3} = 2^{13}$.

Відповідь: $3 = 2^{13}$, тобто $\log_2 3 = 13$.

Алгоритм Полард – ро.

Нехай G – циклічна група з порядком n (n – просте). Розіб'ємо елементи групи G на три підмножини S_1 , S_2 та S_3 , які мають приблизно однакову потужність. При цьому необхідне виконання умови: $1 \notin S_2$. Визначимо послідовність елементів x_i наступним чином:

$$x_0 = 1, x_{i+1} = \begin{cases} b \cdot x_i, x_i \in S_1 \\ x_i^2, x_i \in S_2 \\ a \cdot x_i, x_i \in S_3 \end{cases}, i \geq 0. \quad (2.10)$$

Ця послідовність, у свою чергу, утворить дві послідовності c_i та d_i , що задовольняють умові $x_i = a^{c_i} b^{d_i}$ та визначаються наступним чином:

$$c_0 = 0, c_{i+1} = \begin{cases} c_i, x_i \in S_1 \\ (2 \cdot c_i) \bmod n, x_i \in S_2 \\ (c_i + 1) \bmod n, x_i \in S_3 \end{cases}, i \geq 0. \quad (2.11)$$

та

$$d_0 = 0, d_{i+1} = \begin{cases} (d_i + 1) \bmod n, x_i \in S_1 \\ (2 \cdot d_i) \bmod n, x_i \in S_2 \\ d_i, x_i \in S_3 \end{cases}, i \geq 0. \quad (2.12)$$

Алгоритм буде працювати циклічно, шукаючи таке значення i , для якого $x_i = x_{2i}$. Для таких значень матимуть місце рівність $a^{c_i} b^{d_i} = a^{c_{2i}} b^{d_{2i}}$ або $b^{d_i - d_{2i}} = a^{c_{2i} - c_i}$. Логарифмуючи останню рівність за основою a , матимемо:

$$(d_i - d_{2i}) * \log_a b \equiv (c_{2i} - c_i) \pmod{n}.$$

Якщо $d_i \not\equiv d_{2i} \pmod{n}$, то це рівняння може бути ефективно розв'язане для обчислення $\log_a b$.

Алгоритм

Вхід: генератор a циклічної групи G з порядком n та елемент $b \in G$.

Вихід: дискретний логарифм $x = \log_a b$.

1. $x_0 \leftarrow 1, c_0 \leftarrow 0, d_0 \leftarrow 0$.

2. for $i = 1, 2, \dots, d_0$

2.1. За значеннями $x_{i-1}, c_{i-1}, d_{i-1}$ та $x_{2i-2}, c_{2i-2}, d_{2i-2}$ обчислити значення x_i, c_i, d_i та x_{2i}, c_{2i}, d_{2i} в (2.10), (2.11), (2.12).

2.2. if $(x_i = x_{2i})$ then

$r \leftarrow (d_i - d_{2i}) \pmod{n}$;

if $(r = 0)$ then return (FALSE); // розв'язку не знайдено

$x \leftarrow r^{-1} (c_i - c_{2i}) \pmod{n}$.

return (x) .

Якщо алгоритм завершується невдачею (повертає FALSE), то можна запустити його, вибравши інші початкові значення c_0, d_0 з інтервалу $[1; n - 1]$ та поклавши $x_0 = a^{c_0} b^{d_0}$.

Приклад. Обчислити $\log_2 9$ в групі Z_{19}^* .

Побудуємо наступну таблицю значень послідовностей x_i, c_i, d_i :

i	x_i	a_i	b_i	x_{2i}	a_{2i}	b_{2i}
1	9	0	1	18	1	1
2	18	1	1	4	4	2
3	17	2	1	4	8	6
4	4	4	2	4	16	14
5	17	4	3	4	32	30
6	4	8	6	4	64	62

На 6 кроці отримали $x_6 = x_{12}$. Підставивши їх значення, отримаємо:

$$2^8 * 9^6 = 2^{64} * 9^{62} \text{ або } 2^{8-64} = 9^{62-6}, 2^{-56} = 9^{56}.$$

Логарифмуємо рівність: $-56 * \log_2 9 = 56 \pmod{18}$, оскільки $|Z_{19}^*| = 18$.

Враховуючи що $-56 \pmod{18} \equiv 16$, $56 \pmod{18} \equiv 2$, перепишемо рівність у вигляді $16 * \log_2 9 = 2 \pmod{18}$ або $8 * \log_2 9 = 1 \pmod{9}$.
 $\log_2 9 = 8^{-1} \pmod{9} = 8$.

Відповідь: $\log_2 9 = 8$.

Індексний алгоритм.

Алгоритм, базований на обчисленні індексів, є найпотужнішим при обчисленні дискретного логарифма. Необхідно побудувати відносно невелику підмножину S елементів групи G , яка називається множниковою основою. Ця підмножина повинна обиратися таким чином, щоб якомога більша частина елементів G могла бути подана у вигляді добутку її елементів. При обчисленні значення $\log_a b$ (a – генератор G , $b \in G$) спочатку обчислюються значення логарифмів елементів з S (які заносяться в тимчасову базу даних), а потім на їх основі обчислюється логарифм числа b .

Алгоритм

1. Вхід: генератор a циклічної групи G порядку n та елемент $b \in G$.

Вихід: дискретний логарифм $x = \log_a b$.

$S = \{p_1, p_2, \dots, p_t\}$. В якості значень p_i можна обрати, наприклад, i -те просте число.

2. Побудувати систему лінійних рівнянь, розв'язком якої будуть значення $\log_a p_i$. Для цього виконаємо наступні кроки:

2.1. Обрати деяке ціле k , $0 \leq k \leq n - 1$ та обчислити a^k .

2.2. Спробувати представити значення a^k у вигляді добутку чисел з S :

$$a^k = \prod_{i=1}^t p_i^{c_i}, \quad c_i \geq 0.$$

Якщо така рівність знайдена, то записати рівняння:

$$k = \sum_{i=1}^t c_i \log_a p_i \pmod{n}.$$

2.3. Повторювати кроки 2.1 та 2.2 доти, доки не отримаємо $t + c$ лінійних рівнянь. Невелике ціле число c ($1 \leq c \leq 10$) обирається таким чином, щоб складена система рівнянь мала єдиний розв'язок з великою ймовірністю (якщо скласти лише t рівнянь з t невідомими, то з великою ймовірністю два з цих рівнянь будуть залежними і тоді система матиме більше одного розв'язку).

3. Розв'язати утворену систему рівнянь, отримати значення $\log_a p_i$, $1 \leq i \leq t$.

4. Обчислення $\log_a b$.

4.1. Обрати деяке ціле k , $0 \leq k \leq n - 1$ та обчислити $b * a^k$.

4.2. Спробувати представити значення $b * a^k$ у вигляді добутку чисел з S :

$$b * a^k = \prod_{i=1}^t p_i^{d_i}, \quad d_i \geq 0.$$

Якщо такого представлення знайти не вдається, виконати знову 4.1. Прологарифмувавши другим способом останню рівність, отримаємо:

$$x = \log_a b = \left(\sum_{i=1}^t d_i \log_a p_i - k \right) \bmod n.$$

Приклад. Обчислити $\log_2 12$ в групі Z_{19}^* .

1. Нехай $S = \{2, 3, 5\}$ – множникова основа.

2. Будуємо систему рівнянь для знаходження значень $\log_2 p_i$, де $p_i \in S$. Оскільки множина S містить 3 елементи, то достатньо отримати 3 лінійно незалежних рівняння.

$k = 5$: $2^5 \pmod{19} \equiv 13$ – не подамо у вигляді добутку чисел з S .

$k = 7$: $2^7 \pmod{19} \equiv 14$ – не подамо у вигляді добутку чисел з S .

$k = 2$: $2^2 \pmod{19} \equiv 4 = 2^2$. Перше рівняння: $2 = 2\log_2 2$.

$k = 10$: $2^{10} \pmod{19} \equiv 17$ – не подамо у вигляді добутку чисел з S .

$k = 15$: $2^{15} \pmod{19} \equiv 12 = 2^2 * 3$. Друге рівняння: $15 = 2\log_2 2 + \log_2 3$.

$k = 11$: $2^{11} \pmod{19} \equiv 15 = 3 * 5$. Третє рівняння: $11 = \log_2 3 + \log_2 5$.

3. Система рівнянь за модулем 18 (порядок Z_{19}^* дорівнює 18) має вигляд:

$$\begin{cases} 2 = 2\log_2 2 \pmod{18} \\ 15 = 2\log_2 2 + \log_2 3 \pmod{18} \\ 11 = \log_2 3 + \log_2 5 \pmod{18} \end{cases}$$

Її розв'язком буде:

$\log_2 2 = 1$, $\log_2 3 = 13$, $\log_2 5 = 16$.

4. Обчислення $\log_2 12$.

$k = 3$: $12 * 2^3 \pmod{19} \equiv 1$ – не представимо у вигляді добутку чисел з S .

$k = 7$: $12 * 2^7 \pmod{19} \equiv 16 = 2^4$.

$\log_2 12 + 7 \equiv 4\log_2 2 \pmod{18}$, $\log_2 12 \equiv (4\log_2 2 - 7) \pmod{18} = 15$.

Відповідь: $\log_2 12 = 15$.

Алгоритм Поліга – Хелмана.

Алгоритм Поліга – Хелмана ефективно розв'язує задачу дискретного логарифма в групі G порядку n , якщо число n має лише малі прості дільники.

Нехай $g, h \in G$, $|G| = p^s$, p – просте. Тоді значення $x = \log_g h$ можна подати у вигляді:

$$x = x_0 + x_1p + x_2p^2 + \dots + x_{s-1}p^{s-1}.$$

Піднесемо рівняння $h = g^x$ до степеня p^{s-1} . Її розв'язком буде:

$$\begin{aligned} h^{p^{s-1}} &= (g^{p^{s-1}})^x = (g^{p^{s-1}})^{x_0 + x_1p + x_2p^2 + \dots + x_{s-1}p^{s-1}} = \\ &= (g^{p^{s-1}})^{x_0} * (g^{p^s})^{x_1} * (g^{p^s})^{px_2} * \dots * (g^{p^s})^{p^{s-2}x_{s-1}} = (g^{p^{s-1}})^{x_0}, \end{aligned}$$

оскільки $g^{p^s} = 1$ (g – генератор групи, p^s – її порядок).

Таким чином, з рівності $h^{p^{s-1}} = (g^{p^{s-1}})^{x_0}$ знаходимо x_0 .

Далі, маючи значення $x_0, x_1, \dots, x_{i-1}$, можна обчислити x_i з рівняння:

$$\left(h \cdot g^{-\sum_{j=0}^{i-1} x_j p^j} \right)^{p^{s-(i+1)}} = (g^{p^{s-1}})^{x_i}.$$

Приклад. Обчислити $\log_3 7$ в Z_{17}^* .

Необхідно розв'язати рівняння $3^x = 7$ в групі, порядок якої дорівнює $16 = 2^4$.

Подамо x у двійковій системі числення: $x = x_0 + 2x_1 + 4x_2 + 8x_3$.

1. Обчислення x_0 .

Піднесемо рівняння $3^x = 7$ до степеня $2^3 = 8$:

$$3^{8(x_0 + 2x_1 + 4x_2 + 8x_3)} = 7^8, \quad 3^{8x_0 + 16x_1 + 32x_2 + 64x_3} = -1,$$

$$3^{8x_0} * (3^{16})^{x_1} * (3^{16})^{2x_2} * (3^{16})^{4x_3} = -1.$$

Оскільки $3^{16} \pmod{17} \equiv 1$, то останнє рівняння прийме вигляд $3^{8x_0} = -1$.

Враховуючи, що $3^8 \pmod{17} \equiv -1$, маємо: $(-1)^{x_0} = -1$, $x_0 = 1$.

2. Обчислення x_1 .

Домножимо рівність $3^{x_0 + 2x_1 + 4x_2 + 8x_3} = 7$ на $3^{-x_0} = 3^{-1} \pmod{17} = 6$, отримаємо:

$$3^{2x_1 + 4x_2 + 8x_3} = 7 * 6 \text{ або } 3^{2x_1 + 4x_2 + 8x_3} = 8.$$

Піднесемо рівняння до степеня 4: $3^{8x_1+16x_2+32x_3} = 8^4$, $3^{8x_1} = -1$,
 $x_1 = 1$.

3. Обчислення x_2 .

Алгоритм Полард-ро факторизації числа

У 1974 році Джон Полард запропонував алгоритм знаходження нетривіального дільника натурального числа n . При цьому алгоритм використовує лише операції додавання, множення та віднімання модулярної арифметики.

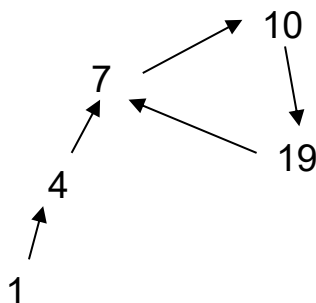
Ідея алгоритму Полард-ро полягає в ітеративному обчисленні деякої наперед заданої поліноміальної функції f з цілими коефіцієнтами. Побудуємо послідовність x_i наступним чином: x_0 оберемо довільним із Z_n , а $x_{i+1} = f(x_i) \bmod n$, $i \geq 0$. Оскільки x_i може приймати лише скінченний набір значень (цілі числа від 0 до n), то існують такі цілі n_1 та n_2 ($n_1 < n_2$), що $x_{n_1} = x_{n_2}$. Враховуючи поліноміальність f , для кожного натурального k маємо: $x_{n_1+k} = x_{n_2+k}$, тобто, починаючи з індексу $i = n_1$, послідовність $\{x_i \bmod n\}$ буде періодичною.

Приклад. Нехай $n = 21$, $x_0 = 1$, $x_{i+1} = x_i^2 + 3 \bmod 21$.

Тоді послідовність x_i має вигляд: 1, 4, 19, 7, 10, 19, 7, 10, ...

Таким чином, $x_3 = x_6$, період послідовності дорівнює 3.

Послідовність x_i можна відобразити у вигляді “кола з хвостом”: коло відповідає періодичній частині, а хвіст – доперіодичній. Картинка нагадує грецьку літеру ρ , тому метод, який застосовується в алгоритмі, називається ρ – евристикою. Послідовність із попереднього прикладу можна зобразити так:



Ідея алгоритму полягає в обчисленні для кожного $i > 0$ значення $d = \text{НСД}(x_{2i} - x_i, n)$. Якщо на деякому кроці $d > 1$, то це i є нетривіальний дільник числа n .

Побудуємо послідовність елементів x_i наступним чином:

$$x_0 = 2, x_{i+1} = f(x_i) = (x_i^2 + 1) \bmod n, i > 0.$$

Алгоритм

Вхід: натуральне число n , параметр $t > 1$.

Вихід: нетривіальний дільник d числа n .

1. $a = 2, b = 2$;

2. for $i \leftarrow 1$ to t do

2.1. Обчислити $a \leftarrow (a^2 + 1) \bmod n$; $b \leftarrow (b^2 + 1) \bmod n$;
 $b \leftarrow (b^2 + 1) \bmod n$;

2.2. Обчислити $d \leftarrow \text{НСД}(a - b, n)$;

2.3. if $1 < d < n$ return (d) ; // знайдено нетривіальний дільник;

3. return $(False)$; // дільник не знайдено.

Вважаємо, що функція $f(x) = (x^2 + 1) \bmod n$ генерує випадкові числа. Тоді для знаходження дільника числа n необхідно виконати не більше ніж $O(\sqrt{n})$ операцій модулярного множення.

Якщо алгоритм Поларда-ро не знаходить дільник за t ітерацій, то замість функції $f(x) = (x^2 + 1) \bmod n$ можна використовувати $f(x) = (x^2 + c) \bmod n$, для деякого цілого $c, c \neq 0, -2$.

Приклад. Нехай $n = 19939$.

Послідовність x_i : 2, 5, 26, 677, 19672, 11473, 12391, 6582, 15217, 5483, 15217, 5483, 15217, ...

a	b	d
2	2	1
5	26	1
26	19672	1
677	12391	1
19672	15217	1
11473	15217	1
12391	15217	157

Знайдено розклад $19939 = 157 * 127$.

Нехай $n = 143$. Послідовність x_i : 2, 5, 26, 105, 15, ...

a	b	d
2	2	1
5	26	НСД(21, 143) = 1
26	15	НСД(11, 143) = 11

Знайдено розклад $143 = 11 * 13$.

Ймовірнісний квадратичний алгоритм факторизації числа

Твердження. Нехай x та y – цілі числа, $x^2 \equiv y^2 \pmod{n}$ та $x \not\equiv y \pmod{n}$. Тоді $x^2 - y^2$ ділиться на n , причому жоден із виразів $x + y$ та $x - y$ не ділиться на n . Число $d = \text{НСД}(x^2 - y^2, n)$ є нетривіальним дільником n .

Теорема 2.15. Якщо n – непарне складене число, яке не є степенем простого числа, то завжди існують такі x та y , що $x^2 \equiv y^2 \pmod{n}$, причому $x \not\equiv y \pmod{n}$.

Доведення. Нехай $n = n_1 * n_2$ – добуток взаємно простих чисел. Оберемо таке y , що $\text{НСД}(y, n) = 1$. Далі розв'яжемо систему рівнянь:

$$\begin{cases} x \equiv y \pmod{n_1} \\ x \equiv -y \pmod{n_2} \end{cases}$$

Розв'язком системи буде $x \equiv 60 \pmod{143}$.

Має місце рівність $60^2 \equiv 5^2 \pmod{143}$, причому $60 \not\equiv 5 \pmod{143}$.

Тоді дільником числа n буде $d = \text{НСД}(60^2 - 5^2, 143) = 11$.

Формально ймовірнісний квадратичний алгоритм факторизації будується на наступній ідеї:

Нехай $F = \{p_0, p_1, p_2, \dots, p_t\}$ – множникова основа, p_i – різні прості числа, причому дозволяється обрати $p_0 = -1$. Побудуємо множину порівнянь $x_i^2 \equiv z_i$, таку що значення z_i є повністю факторизованими у множині F :

$$z_i = \prod_{j=0}^t p_j^{e_{ij}},$$

та добуток деякої підмножини значень z_i є повним квадратом:

$$z = \prod_{i=1}^k z_i^{f_i} = y^2, y \in \mathbb{Z}, f_i \in \{0, 1\}.$$

Якщо множина порівнянь із вказаними властивостями побудована, то, поклавши $x = \prod_{i=1}^k z_i^{f_i}$ і перевіривши виконання нерівності $x \equiv y \pmod{n}$, отримаємо $x^2 \equiv y^2 \pmod{n}$. Число $d = \text{НСД}(x^2 - y^2, n)$ є нетривіальним дільником n .

Приклад. Знайти дільник числа $n = 143$.

Обираємо випадково число $x \in [2, 142]$, обчислюємо $x^2 \pmod{143}$ та розкладаємо результат на множники:

1. $z_1 = 19^2 \pmod{143} = 75 = 3 * 5^2$.
2. $z_2 = 77^2 \pmod{143} = 66 = 2 * 3 * 11$.
3. $z_3 = 29^2 \pmod{143} = 126 = 2 * 3^2 * 7$.
4. $z_4 = 54^2 \pmod{143} = 56 = 2^3 * 7$.

Можна помітити, що добуток z_3 та z_4 є повним квадратом:

$$z = z_3 * z_4 = 2^4 * 3^2 * 7^2 = (2^2 * 3 * 7)^2 = 84^2.$$

Маємо рівність:

$$z_3 * z_4 = 29^2 * 54^2 \equiv 84^2 \pmod{143},$$

або, враховуючи, що $29 * 54 \pmod{143} \equiv 136$, маємо:

$$136^2 \equiv 84^2 \pmod{143}, \text{ причому } 136 \not\equiv 84 \pmod{143}.$$

Дільником числа $n = 143$ буде

$$d = \text{НСД}(136 - 84, 143) = \text{НСД}(52, 143) = 13.$$

Квадратичний алгоритм факторизації

Серед усіх існуючих алгоритмів факторизації найшвидшим є квадратичний. Він ефективно застосовується для чисел, кількість цифр яких менша за 100, та які не мають малих простих дільників. Евристичний аналіз, проведений Померансом у 1981 році, показав, що число N може бути розкладено на множники за час

$$e^{(1+o(1))\sqrt{\ln N \ln \ln N}}.$$

Нехай n – число, яке факторизується;

$$m = \sqrt{n}.$$

Розглянемо многочлен $q(x) = (x + m)^2 - n$.

Квадратичний алгоритм обирає $a_i = x + m$ ($x = 0, \pm 1, \pm 2, \dots$), обчислює значення $b_i = (x + m)^2 - n$ та перевіряє, чи факторизується b_i у множниковій основі $F = \{p_0, p_1, p_2, \dots, p_t\}$.

Зазначимо, що $a_i^2 = (x + m)^2 - n \equiv (x + m)^2 \pmod{n} \equiv b_i \pmod{n}$.

Алгоритм

Вхід: натуральне число n , яке не є степенем простого числа.

Вихід: нетривіальний дільник d числа n .

1. Обрати множникову основу $F = \{p_0, p_1, p_2, \dots, p_t\}$, де $p_0 = -1$, $p_i - i$ – просте число p , для якого n є квадратичним лишком за модулем p .

2. Обчислити $m = \lfloor \sqrt{n} \rfloor$.

3. Знаходження $t + 1$ пари (a_i, b_i) .

Значення x перебираються у послідовності $0, \pm 1, \pm 2, \dots$.

Покласти $i \leftarrow 1$. Поки $i \leq t + 1$ робити:

3.1. Обчислити $b = q(x) = (x + m)^2 - n$ та перевірити, чи розкладається b у множниковій основі F . Якщо ні, обрати наступне x та повторити цей крок.

3.2. Нехай $b = \prod_{j=1}^t p_j^{e_{ij}}$. Покласти $a_i = x + m$, $b_i = b$, $v_i = (v_{i1}, v_{i2}, \dots, v_{it})$, де $v_{ij} = e_{ij} \bmod 2$, $1 \leq j \leq t$.

3.3. $i \leftarrow i + 1$.

4. Знайти підмножину $T \subseteq \{1, 2, \dots, t + 1\}$ таку, що $\sum_{i \in T} v_i = 0$.

5. Обчислити $x = \prod_{i \in T} a_i \bmod n$.

6. Для кожного j , $1 \leq j \leq t$ обчислити $l_j = (\sum_{i \in T} e_{ij}) / 2$.

7. Обчислити $y = \prod_{j=1}^t p_j^{l_j} \bmod n$.

8. Якщо $x \equiv y \pmod{n}$, знайти іншу підмножину $T \in \{1, 2, \dots, t + 1\}$ таку, що $\sum_{i \in T} v_i = 0$ та перейти до кроку 5.

9. Обчислити дільник $d = \text{НСД}(x - y, n)$.

Приклад. Розкласти на множники $n = 24961$.

1. Побудуємо множникову основу: $F = \{-1, 2, 3, 5, 13, 23\}$

2. $m = \lfloor \sqrt{24961} \rfloor = 157$.

3. Побудуємо наступну таблицю:

i	x	$q(x)$	факторизація $q(x)$	a_i	v_i
1	0	-312	$-23 * 3 * 13$	157	(1, 1, 1, 0, 1, 0)
2	1	3	3	158	(0, 0, 1, 0, 0, 0)
3	-1	-625	-54	156	(1, 0, 0, 0, 0, 0)
4	2	320	$26 * 5$	159	(0, 0, 0, 1, 0, 0)
5	-2	-936	$-23 * 32 * 13$	155	(1, 1, 0, 0, 1, 0)
6	4	960	$26 * 3 * 5$	161	(0, 0, 1, 1, 0, 0)
7	-6	-2160	$-24 * 33 * 5$	151	(1, 0, 1, 1, 0, 0)

4. Оберемо $T = \{1, 2, 5\}$, оскільки $v_1 + v_2 + v_5 = 0$.

5. Обчислимо $x = (a_1 a_2 a_5) \pmod{n} = 936 = 2^6 * 3^4 * 13^2$.
 6. $l_1 = 1, l_2 = 3, l_3 = 2, l_4 = 0, l_5 = 1, l_6 = 0$.
 7. $y = -2^3 * 3^2 * 13 \pmod{n} = 24025$.
 8. Оскільки $936 \equiv -24025 \pmod{n}$, необхідно шукати іншу множину T .
 9. Оберемо $T = \{3, 6, 7\}$, оскільки $v_3 + v_6 + v_7 = 0$.
 10. Обчислимо $x = (a_3 a_6 a_7) \pmod{n} = 23405 = 2^{10} * 3^4 * 5^6$.
 11. $l_1 = 1, l_2 = 5, l_3 = 2, l_4 = 3, l_5 = 0, l_6 = 0$.
 12. $y = -2^5 * 3^2 * 5^3 \pmod{n} = 13922$.
 13. $23405 \equiv 13922 \pmod{n}$.
- $d = \text{НСД}(x - y, n) = \text{НСД}(9483, 24961) = 109$ – дільник.
Відповідь: 109 – дільник 24961.

Символи Лежандра та Якобі

Означення. Нехай p – просте, a – ціле число. Символ Лежандра $\left(\frac{a}{p}\right)$

визначається так:

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & \text{якщо } p \text{ ділиться на } a \\ 1, & \text{якщо } a \in \mathbb{Q}_p \\ -1, & \text{якщо } a \in \overline{\mathbb{Q}_p}. \end{cases}$$

Критерій Ейлера. Число a , яке не ділиться на непарне просте p , є квадратичним лишком за модулем p тільки тоді, коли $a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$, і квадратичним нелишком тільки тоді, коли $a^{\frac{p-1}{2}} \equiv -1 \pmod{p}$.

Доведення. За теоремою Ферма $a^{p-1} \equiv 1 \pmod{p}$ при $\text{НСД}(a, p) = 1$ та $\text{НСД}(2, p) = 1$. Або:

$$\left(a^{\frac{p-1}{2}} + 1\right) * \left(a^{\frac{p-1}{2}} - 1\right) \equiv 0 \pmod{p}.$$

Звідси вираз в одній із дужок ділиться на p . Обидві дужки не можуть ділитися на p , оскільки тоді на p ділилася б і їх різниця, яка дорівнює 2, а за умовою теореми p – непарне просте число. Якщо a є квадратичним лишком, то $a \equiv x^2 \pmod{p}$ для деякого такого x , що $\text{НСД}(x, p) = 1$. Маємо:

$a^{\frac{p-1}{2}} \equiv (x^2)^{\frac{p-1}{2}} \equiv x^{p-1} \equiv 1 \pmod{p}$, тобто $a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$ або $a^{\frac{p-1}{2}} \equiv -1 \pmod{p}$ ділиться на

р. Якщо a є квадратичним нелишком, то $a^{\frac{p-1}{2}} - 1$ не ділиться на p , звідки $a^{\frac{p-1}{2}} + 1$ повинно ділитися на p , або $a^{\frac{p-1}{2}} \equiv -1 \pmod{p}$.

Наслідок. $\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p}$. Якщо число a є квадратичним лишком

за модулем p , то за означенням символу Лежандра $\left(\frac{a}{p}\right) = 1$, а за

критерієм Ейлера $a^{\frac{p-1}{2}} \pmod{p} \equiv 1$. Відповідно, якщо число a є квадратичним нелишком за модулем p , то $\left(\frac{a}{p}\right) = -1$ і $a^{\frac{p-1}{2}} \pmod{p} \equiv -1$,

звідки і випливає твердження.

Приклад. Чи існує розв'язок рівняння $x^2 \equiv 5 \pmod{7}$.

Якщо існує розв'язок рівняння, то число 5 повинно бути квадратичним лишком за модулем 7. Перевіримо це за критерієм Ейлера:

$5^{\frac{7-1}{2}} \equiv 5^3 \pmod{7} \equiv 25 * 5 \pmod{7} \equiv 4 * 5 \pmod{7} \equiv 20 \pmod{7} \equiv -1 \pmod{7}$. Звідси випливає, що 5 є квадратичним нелишком за модулем 7 і рівняння розв'язків не має.

Властивості символу Лежандра.

1. $\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p}$. Вказана властивість є наслідком критерію

Ейлера.

Зокрема $\left(\frac{1}{p}\right) = 1$ та $\left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{2}}$.

Отже, $-1 \in Q_p$, якщо $p \equiv 1 \pmod{4}$ та $-1 \notin Q_p$, якщо $p \equiv 3 \pmod{4}$.

2. $\left(\frac{a * b}{p}\right) = \left(\frac{a}{p}\right) * \left(\frac{b}{p}\right)$. Властивість випливає з послідовності

очевидних порівнянь:

$$\left(\frac{a * b}{p}\right) (ab)^{\frac{p-1}{2}} \equiv a^{\frac{p-1}{2}} * b^{\frac{p-1}{2}} \left(\frac{a}{p}\right) * \left(\frac{b}{p}\right) \pmod{p}.$$

Зокрема, якщо $a \in Z_p^*$, то $\left(\frac{a^2}{p}\right) = 1$ та $\left(\frac{a^2 b}{p}\right) = \left(\frac{b}{p}\right)$.

3. Якщо $a \equiv b \pmod{p}$, то $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$. Властивість випливає з того, що

числа одного класу є одночасно або квадратичними лишками, або нелишками. Впливаючи з цієї властивості, можна записати:

$$\left(\frac{a}{p}\right) = \left(\frac{a+pt}{p}\right), t \in \mathbb{Z}.$$

4. $\left(\frac{1}{p}\right) = 1$. Одиниця є квадратичним лишком для довільного

непарного простого p . Ця властивість випливає з того, що порівняння $x^2 \equiv 1 \pmod{p}$ завжди має розв'язки $x = \pm 1 \pmod{p}$.

$$5. \left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}}.$$

Якщо $p = 8k \pm 1$, то $\frac{p^2-1}{8} = \frac{(8k \pm 1)^2 - 1}{8} = \frac{64k^2 \pm 16k}{8} = 8k^2 \pm 2k$ – парне

число.

Якщо $p = 8k \pm 3$, то $\frac{p^2-1}{8} = \frac{(8k \pm 3)^2 - 1}{8} = \frac{64k^2 \pm 48k + 8}{8} = 8k^2 \pm 6k +$

$+ 1$ – непарне число.

Отже $\left(\frac{2}{p}\right) = 1$, якщо $p \equiv 1$ або $7 \pmod{8}$ та $\left(\frac{2}{p}\right) = -1$, якщо $p \equiv 3$ або 5

$\pmod{8}$.

6. Закон взаємності непарних простих чисел. Якщо p – просте непарне число, відмінне від q , то

$$\left(\frac{p}{q}\right) * \left(\frac{q}{p}\right) = (-1)^{\frac{(p-1)(q-1)}{4}}.$$

Помноживши цю рівність на $\left(\frac{p}{q}\right)$, отримаємо:

$\left(\frac{q}{p}\right) = (-1)^{\frac{(p-1)(q-1)}{4}} * \left(\frac{p}{q}\right)$. Якщо виконується хоча б одна з рівностей $p \pmod{4} \equiv 1$ чи $q \pmod{4} \equiv 1$, то $\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right)$, інакше $\left(\frac{p}{q}\right) = -\left(\frac{q}{p}\right)$.

Символ Якобі є узагальненням символу Лежандра на випадок, коли n є непарним, але не обов'язково простим.

Означення. Нехай n – непарне ціле число, $n \in \mathbb{Z}$. Відомо, що $n = p_1^{k_1} p_2^{k_2} \dots p_t^{k_t}$, де p_i – прості числа. Символ Якобі $\left(\frac{a}{n}\right)$ визначається так:

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{k_1} \left(\frac{a}{p_2}\right)^{k_2} \dots \left(\frac{a}{p_t}\right)^{k_t}.$$

Зазначимо, що якщо n просте, то символ Якобі стає символом Лежандра.

Властивості символу Якобі

1. $\left(\frac{a}{n}\right)$ може приймати одне з трьох значень: -1 , 0 чи 1 . При цьому $\left(\frac{a}{n}\right) = 0$ тільки тоді, коли $\text{НСД}(a, n) \neq 1$.

2. $\left(\frac{a * b}{n}\right) = \left(\frac{a}{n}\right) * \left(\frac{b}{n}\right)$. Якщо $a \in \mathbb{Z}_n^*$, то $\left(\frac{a^2}{n}\right) = 1$.

3. $\left(\frac{a}{m * n}\right) = \left(\frac{a}{m}\right) * \left(\frac{a}{n}\right)$.

4. Якщо $a \equiv b \pmod{n}$, то $\left(\frac{a}{n}\right) = \left(\frac{b}{n}\right)$.

5. $\left(\frac{1}{n}\right) = 1$.

6. $\left(\frac{-1}{n}\right) = (-1)^{\frac{n-1}{2}}$. Отже $\left(\frac{-1}{n}\right) = 1$, якщо $n \equiv 1 \pmod{4}$ та $\left(\frac{-1}{n}\right) = -1$, якщо $n \equiv 3 \pmod{4}$.

7. $\left(\frac{2}{n}\right) = (-1)^{\frac{n^2-1}{8}}$.

Отже $\left(\frac{2}{n}\right) = 1$, якщо $n \equiv 1$ або $7 \pmod{8}$ та $\left(\frac{2}{n}\right) = -1$, якщо $n \equiv 3$ або $5 \pmod{8}$.

8. $\left(\frac{m}{n}\right) = \left(\frac{n}{m}\right) (-1)^{\frac{(m-1)(n-1)}{4}}$.

З властивостей символу Якобі випливає, що якщо n непарне, а число a подати у вигляді $a = 2^k a_1$, де a_1 – непарне число, то

$$\left(\frac{a}{n}\right) = \left(\frac{2^k}{n}\right) \left(\frac{a_1}{n}\right) = \left(\frac{2^k}{n}\right) \left(\frac{n \bmod a_1}{a_1}\right) (-1)^{\frac{(a_1-1)(n-1)}{4}}.$$

Ця формула дає можливість обчислити значення символу Якобі не маючи розкладу числа n на прості множники.

На відміну від символу Лежандра символ Якобі $\left(\frac{a}{n}\right)$ не визначає, чи є число a квадратичним лишком за модулем n . Справді, якщо $a \in Q_n$, то $\left(\frac{a}{n}\right) = 1$, але з того, що $\left(\frac{a}{n}\right) = 1$, не випливає $a \in Q_n$.

Означення. Нехай n – непарне ціле число, $n \equiv 3$. Число a назовемо псевдопростим за модулем n , якщо $\left(\frac{a}{n}\right) = 1$. Множину псевдопростих чисел позначатимемо через $\hat{Q}_n = J_n - Q_n$, де $J_n = \{a \in Z_n^* \mid \left(\frac{a}{n}\right) = 1\}$.

2.1.3. Методи булевої алгебри, елементи кореляційного та спектрального аналізу

При оцінці криптографічних властивостей більшості симетричних шифрів використовуються методи булевої алгебри, елементи кореляційного й спектрального аналізу [7; 10; 15–17]. Введемо необхідні терміни і визначення, розглянемо основні показники та критерії оцінки криптографічних властивостей.

Булевою функцією f від n змінних є функція, що здійснює відображення з поля $GF(2^n)$ всіх двійкових векторів $x = (x_1, \dots, x_n)$ довжини n в полі $GF(2)$.

Звичайно булеві функції представляються в алгебраїчній нормальній формі й розглядаються як сума добутків складових координат. Поле $GF(2^n)$ складається з 2^n векторів $\alpha_i: \alpha_0 = (0, \dots, 0, 0), \alpha_1 = (0, \dots, 0, 1), \dots, \alpha_{2^n-1} = (1, \dots, 1, 1)$, $\alpha_i \in V_n$, де V_n – векторний простір в $GF(2^n)$.

Послідовністю функції f називається $(1, -1)$ -послідовність, визначена як $((-1)^{f(\alpha_0)}, (-1)^{f(\alpha_1)}, \dots, (-1)^{f(\alpha_{2^n-1})})$.

Таблицею істинності функції f називається $(0, 1)$ -послідовність, певна як $(f(\alpha_0), f(\alpha_1), \dots, f(\alpha_{2^n-1}))$.

Послідовність функції $f \in \text{збалансованою}$, якщо її $(0, 1)$ -послідовність $(1, -1)$ -послідовність) містить однакову кількість нулів і одиниць (одиниць і мінус одиниць). Функція $f \in \text{збалансованою}$, якщо збалансована її послідовність.

Еквівалентне визначення збалансованості: функція f над $GF(2^n)$ є збалансованою, якщо її вихідні значення є рівноймовірними:

$$|\{x \mid f(x) = 0\}| = |\{x \mid f(x) = 1\}| = 2^n - 1.$$

Афінною функцією f називається функція виду

$$f = a_1x_1 \oplus \dots \oplus a_nx_n \oplus c,$$

де $a_j, c \in GF(2)$, $j = 1, 2, \dots, n$.

Функція f називається лінійною, якщо $c = 0$.

Вагою Хемінга вектора α ($(0,1)$ -послідовності α), позначуваним як $W(\alpha)$, є кількість одиниць у векторі (послідовності).

Відстанню Хемінга $d(f, g)$ між послідовностями двох функцій f і g є кількість позицій, у яких різні послідовності цих функцій.

Нелінійність N_s перетворення – мінімальна відстань Хемінга між вихідною послідовністю S й усіма вихідними послідовностями афінних функцій над деяким полем [9; 21; 36]:

$$N_s = \min\{d(S, \varphi)\},$$

де φ – множина афінних функцій.

Нелінійність функції N_f – мінімальна відстань Хемінга між функцією f й усіма афінними функціями над $GF(2^n)$ [16]:

$$N_f = \min\{d(f, \varphi)\},$$

де φ – множина афінних функцій.

Для довільної функції f нелінійність N_f над $GF(2^n)$ може досягати:

$$N_f \leq 2^{n-1} - 2^{\frac{n}{2}-1}.$$

Для збалансованої функції f над $GF(2^n)$ ($n \geq 3$) нелінійність N_f може досягати [9]:

$$N_f \leq \begin{cases} 2^{n-1} - 2^{n/2-1} - 2, n = 2k, \\ \lfloor \lfloor 2^{n-1} - 2^{n/2-1} \rfloor \rfloor, n = 2k + 1, \end{cases}$$

де $\lfloor \lfloor x \rfloor \rfloor$ – максимальне парне ціле, менше або рівне x .

Функція f має кореляційний імунітет порядку k , якщо вихідна послідовність функції $y \in Y$ статистично не залежить від будь-якої підмножини із k вхідних координат:

$$\forall \{x_1, \dots, x_k\} P(y \in Y / \{x_1, \dots, x_k\} \in X) = P(y \in Y).$$

Еквівалентне визначення кореляційного імунітету в термінах перетворення Уолша [94]: функція f над полем $GF(2^n)$ має кореляційний імунітет порядку k , $KI(k)$, якщо її перетворення Уолша задовольняє рівності $F(\omega) = 0$ для всіх $\omega \in V_n$ таких, що $1 \leq W(\omega) \leq k$:

$$\forall \omega \in V_n \quad F(\omega) = 0 \quad KI(f) = k.$$

Перетворення Уолша $F(\omega)$ функції f над полем $GF(2^n)$ визначається, якщо ухвалює дійсні значення функція

$$F(\omega) = 2^{-n} \sum_x (-1)^{f(x) \oplus \langle \omega, x \rangle},$$

де $\omega \in V_n$, $f(x)$, $\langle \omega, x \rangle \in \mathbb{N}$ ($\langle \omega, x \rangle$ – скалярний добуток $w_1 x_1 \oplus \dots \oplus w_n x_n$).

Кореляційно-імунна функція k -го порядку – функція, що володіє кореляційним імунітетом порядку k . Збалансовані кореляційно-імунні функції називаються *еластичними функціями*.

Функція f над полем $GF(2^n)$ задовольняє: критерію поширення щодо вектора α , $KP(\alpha)$, якщо функція $f(x) \oplus f(x \oplus \alpha)$ є збалансованою, $x \in V_n$, де $x = (x_1, x_2, \dots, x_n)$:

$$P(f(x) = f(x \oplus \alpha)) = \frac{1}{2};$$

критерію поширення степеня k , $KP(k)$, якщо задовольняється критерій поширення щодо всіх векторів $\alpha \in V_n$ при $1 \leq W(\alpha) \leq k$:

$$P(f(x) = f(x \oplus \alpha)) = \frac{1}{2} \quad \forall \alpha: 1 \leq W(\alpha) \leq k;$$

суворому лавинному критерію, СЛК, якщо f задовольняє критерію поширення степеня 1:

$$P(f(x) = f(x \oplus \alpha)) = \frac{1}{2} \forall \alpha: W(\alpha) = 1.$$

Алгебраїчний степінь $\deg(f)$ є степенем найдовшого доданка функції, представленого в алгебраїчній нормальній формі. Алгебраїчною нормальною формою називається вираз виду:

$$f(x_1, \dots, x_n) = a_0 \oplus \bigoplus_{i=1}^n a_i x_i \oplus \bigoplus_{1 \leq i < j \leq 1} a_{ij} x_i x_j \oplus \dots \oplus a_{12\dots n} x_1 x_2 \dots x_n.$$

Функція f над $GF(2^n)$ називається *бент-функцією* [9; 21; 36; 55; 61; 64; 68; 73], якщо

$$2^{-n/2} \sum_{x \in V_n} (-1)^{f(x) \oplus \langle \beta, x \rangle} = \pm 1.$$

для всіх $\beta \in V_n$.

Послідовність бент-функції називається *бент-послідовністю*. Для бент-функцій справедливі наступні твердження [9]:

1. $\langle \xi, \ell \rangle = \pm 2^{n/2}$ для будь-якої афінної послідовності ℓ довжини 2^n ;
2. $f(x) \oplus f(x \oplus \alpha)$ збалансована $\forall \alpha \in V_n, W(\alpha) \neq 0$;
3. $f(x) \oplus \langle \alpha, x \rangle$ ухвалює значення одиниці $2^{n-1} \pm 2^{n/2-1}$ раз $\forall \alpha \in V_n$;
4. $f(x) \oplus h(x)$, де $h(x)$ – афінна функція, також є бент-функцією, де ξ – послідовність бент-функції.

Автокореляційна функція $\hat{r}(s)$ для $s \in 0 \dots 2^n - 1$ визначена як

$$\hat{r}(s) = \sum_{x=0}^{2^n-1} \hat{f}(x) \hat{f}(x \oplus s).$$

Існує думка, що функція f задовольняє характеристиці поширення m , якщо

$$(1 \leq |s| \leq m) \Rightarrow |\hat{r}(s)| = 0.$$

Аналогічно, автокореляція $AC(f)$ функції f визначається як модуль найбільшого значення $\hat{r}(s)$:

$$AC(f) = \max_{s \neq 0} \left| \sum_{x \in V_n} \hat{f}(x) \hat{f}(x \oplus s) \right| = \max_{s \neq 0} |\hat{r}(s)|.$$

Автокореляція $\hat{r}(s)$ забезпечує витік інформаційного потоку із входу на вихід функції. У деяких випадках дуже сильний взаємозв'язок може бути поданий як лінійна структура (для якої справедливі рівності $f(x) \oplus f(x \oplus s)$).

= 1 або $f(x)$ ($f(x(s)) = 0$). Кryptoаналітична значущість даного факту (для блокових шифрів) показана Евертсом [61]. Вони, як правило, уникнені. У фундаментальній роботі Майєра і Штаффельбаха мінімальна відстань функції до будь-якої функції з лінійною структурою визначена як *міра нелінійності*.

Автокореляція $AC(f)$ була запропонована Цзенгом і Цзангом як глобальна лавинна характеристика, оскільки вона включає всі автокореляції, що не тільки відповідають s із низькою вагою Хемінга. В аналогічному ракурсі визначена й інша глобальна характеристика: сума квадратів різних автокореляцій, тобто

$$\sigma_f = \sum_s \hat{r}(s)^2.$$

Бент-функції досягають нульової автокореляції (і, більше того, максимальної нелінійності).

Алгебраїчний степінь кожної змінної x_i , $\deg(f, x_i)$ визначається степенем найдовшого терма функції, що містить x_i [9]:

$$\deg(f, x_i) = \max_{0 \leq i < n} \deg(\text{term}_{x_i}).$$

Кількість термів функції $\text{term}(f)$ визначається кількістю термів функції, з яких полягає функція, подана в алгебраїчній нормальній формі:

$$\text{term}(f) = \min_{0 \leq j < 2^n} \text{term}_j.$$

Кількість термів функції, що містять змінну x_i , $\text{term}_{x_i}(f)$, визначається кількістю термів функції, що містять змінну x_i :

$$\text{term}_{x_i}(f) = \min_j \text{term}_j(x_i).$$

Введений коефіцієнт рівномірної мінімізації крос-кореляції характеризує рівномірну мінімізацію коефіцієнтів крос-кореляції

$$k_{pm} = \frac{k_{gp}}{r \cdot c_{cp}},$$

де k_{gp} – граничний коефіцієнт крос-кореляції;

r – удільна вага ненульових значень коефіцієнтів крос-кореляції;

c_{cp} – середнє значення коефіцієнта крос-кореляції.

Наведені значення мають такий вигляд:

$$r = \left| \frac{B}{2^n} - 2^{n-1} \right|, c_{cp} = \frac{\sum_{i=1}^n c_i}{2^n},$$

$$k_{rp} = \frac{|1 - 2^n| \cdot (c_n + c_{n+2})}{2},$$

для V_{n+1} і

$$k_{rp} = |1 - 2^n| \cdot c_n$$

для V_n

$$c_i(f, Lw) = 2^{-n} \sum_x (-1)^{f(x)} (-1)^{wx} = 2^{-n} \hat{F}(w),$$

де B – загальна кількість ненульових значень коефіцієнтів крос-кореляції;

c_i – значення коефіцієнта крос-кореляції функції з множиною всіх афінних функцій L_w ;

c_n – коефіцієнт кореляції бент-функції над V_n – з множиною всіх афінних функцій над V_n ;

c_{n+2} – коефіцієнт кореляції бент-функції над V_{n+2} з множиною всіх афінних функцій над V_{n+2} .

Існує теоретичне обґрунтування введення коефіцієнта рівномірності мінімізації кореляції k_{rm}^* , який визначається як середнє значення суми всіх ненульових значень коефіцієнтів кореляції

$$k_{rm}^* = \frac{\sum_{i=1}^B c_i(f, L_w)}{B},$$

де

$$c_i(f, Lw) = 2^{-n} \sum_x (-1)^{f(x)} (-1)^{wx} = 2^{-n} \hat{F}(w),$$

і B – загальна кількість ненульових значень коефіцієнтів кореляції.

Абсолютне значення функції кореляції C_f визначається як максимальне значення всіх коефіцієнтів кореляції:

$$C_f = \max |c(f, \ell_i)|,$$

і визначає максимальне значення коефіцієнта кореляції функції f над V_n , $\ell_i \in L$, де ℓ_i , L – лінійна функція і множина усіх лінійних функцій відповідно.

Кількість векторів num_1 і num_2 , при яких функція не задовольняє/задовольняє критерію поширення, визначається як [33]:

$$\text{num}_1 = n\text{PC}(k),$$

$$\text{num}_2 = \text{PC}(k).$$

Кількість векторів num_3 і num_4 , при яких функція не задовольняє/задовольняє кореляційному імунітету, визначається як

$$\text{num}_3 = n\text{KI}(k),$$

$$\text{num}_4 = \text{KI}(k).$$

2.2. Компоненти криптосистеми та їх функціональні характеристики

Розглянемо загальну схему симетричної, або традиційної, криптографії (рис. 2.1).

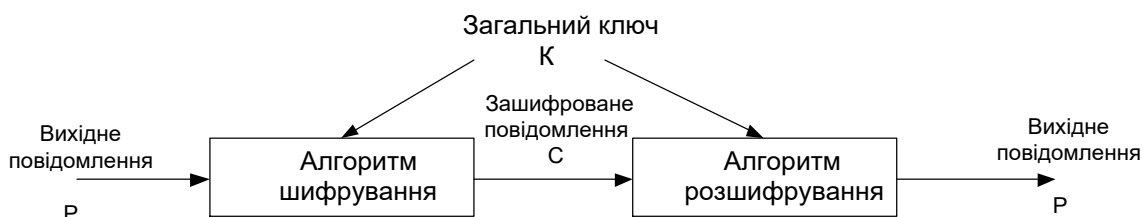


Рис. 2.1. Загальна схема симетричного шифрування

У процесі шифрування використовується певний алгоритм шифрування, на вхід якому подаються вихідне незашифроване повідомлення, називане також plaintext, і ключ. Виходом алгоритму є зашифроване повідомлення, що називається також ciphertext. Ключ є значенням, що не залежить від шифрованого повідомлення. Зміна ключа повинна призводити до зміни зашифрованого повідомлення [18–20].

Зашифроване повідомлення передається одержувачу. Одержувач перетворює зашифроване повідомлення у вихідне незашифроване повідомлення за допомогою алгоритму дешифрування і того ж самого ключа, який використовувався при шифруванні, або ключа, який можна легко одержати із ключа шифрування.

Незашифроване повідомлення будемо позначати P або M , від слів plaintext і message. Зашифроване повідомлення будемо позначати C , від слова ciphertext.

Безпека, забезпечувана традиційною криптографією, залежить від декількох факторів.

По-перше, криптографічний алгоритм повинен бути досить сильним, щоб передане зашифроване повідомлення неможливе було розшифрувати без ключа, використовуючи тільки різні статистичні закономірності зашифрованого повідомлення або які-небудь інші способи його аналізу.

По-друге, безпека переданого повідомлення повинна залежати від таємності ключа, але не від таємності алгоритму. Алгоритм повинен бути проаналізований фахівцями, щоб виключити наявність слабких місць, при яких погано схований взаємозв'язок між незашифрованим і зашифрованим повідомленнями. До того ж при виконанні цієї умови виробники можуть створювати дешеві апаратні чипи і вільно розповсюджувати програми, що реалізують даний алгоритм шифрування.

По-третє, алгоритм повинен бути таким, щоб не можна було довідатися ключ, навіть знаючи досить багато пар (зашифроване повідомлення, незашифроване повідомлення), отриманих при шифруванні з використанням даного ключа.

Клод Шенон увів поняття дифузії і конфузії для опису стійкості алгоритму шифрування.

Дифузія – це розсіювання статистичних особливостей незашифрованого тексту в широкому діапазоні статистичних особливостей зашифрованого тексту. Це досягається тим, що значення кожного елемента незашифрованого тексту впливає на значення багатьох елементів зашифрованого тексту або, що те ж саме, будь-який елемент зашифрованого тексту залежить від багатьох елементів незашифрованого тексту.

Конфузія – це знищення статистичного взаємозв'язку між зашифрованим текстом і ключем.

Якщо X – це вихідне повідомлення й K – криптографічний ключ, то зашифрований переданий текст можна записати у вигляді $Y = E_K[X]$.

Одержувач, використовуючи той же ключ, розшифровує повідомлення $X = D_K[Y]$.

Супротивник, не маючи доступу до K і X , повинен спробувати довідатися X , K , або й те й інше.

Області застосування

Стандартний алгоритм шифрування повинен бути застосований в багатьох додатках:

Шифрування даних. Алгоритм повинен бути ефективний при шифруванні файлів даних або великого потоку даних.

Створення випадкових чисел. Алгоритм повинен бути ефективний при створенні певної кількості випадкових бітів.

Гешування. Алгоритм повинен ефективно перетворюватися в однобічну геш-функцію.

Платформи

Стандартний алгоритм шифрування повинен бути реалізований на різних платформах, які, відповідно, висувають різні вимоги.

Алгоритм повинен ефективно реалізовуватися на спеціалізованій апаратурі, призначеній для виконання шифрування/дешифрування.

Більші процесори. Хоча для найбільш швидких додатків завжди використовується спеціальна апаратура, програмні реалізації застосовуються частіше. Алгоритм повинен допускати ефективну програмну реалізацію на 32-бітних процесорах.

Процесори середнього розміру. Алгоритм повинен працювати на мікроконтролерах й інших процесорах середнього розміру.

Малі процесори. Повинна існувати можливість реалізації алгоритму на смарт-картах, нехай навіть із урахуванням твердих обмежень на використовувану пам'ять.

Додаткові вимоги

Алгоритм шифрування повинен, по можливості, задовольняти деяким додатковим вимогам.

Алгоритм повинен бути простим для написання коду, щоб мінімізувати ймовірність програмних помилок.

Алгоритм повинен мати плоский простір ключів і допускати будь-який випадковий рядок бітів потрібної довжини як можливого ключа. Наявність слабких ключів небажана.

Алгоритм повинен легко модифікуватися для різних рівнів безпеки й задовольняти як мінімальним, так і максимальним вимогам.

Усі операції з даними повинні здійснюватися над блоками, кратними байту або 32-бітному слову.

Формальне математичне визначення криптосистеми.

Критерії та показники ефективності

Абстрактно *секретна система* визначається як деяка множина відображень одного простору (множини можливих повідомлень) в інший простір (множину можливих криптограм) [8, 9, 21].

Зафіксуємо множину можливих повідомлень $M = \{M_1, M_2, \dots, M_m\}$ і множину криптограм $E = \{E_1, E_2, \dots, E_n\}$. Зафіксуємо також множину відображень $\varphi = \{\varphi_1, \varphi_2, \dots, \varphi_k\}$, де $\varphi : M \rightarrow E, i = 1, 2, \dots, k$.

Якщо множини M і E рівнопотужні, тобто $n = m$, а відображення $\varphi_i \in \varphi$ сюрєктивне й ін'єктивне, то існує зворотне відображення $\varphi_i^{-1} : E \rightarrow M$, що кожному елементу множини E ставить у відповідність елемент множини M .

Очевидно, що φ_i та φ_i^{-1} задають взаємооднозначне відображення множин M і E .

Зафіксуємо тепер множину ключів $k = \{K_1, K_2, \dots, K_k\}$ так, що для всіх $i = 1, 2, \dots, k$ відображення $\varphi_i \in \varphi$ однозначно задається ключем K_i , тобто: $\varphi_i : M \xrightarrow{K_i} E$.

Кожне конкретне відображення φ_i з множини відповідає способу шифрування за допомогою конкретного ключа K_i . На рис. 2.2 схематично подане відображення $\varphi_i \in \varphi$, задане ключем K_i .

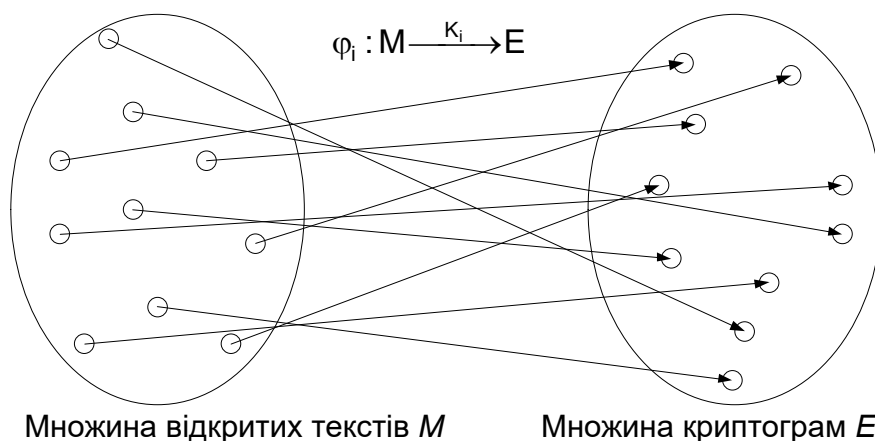


Рис. 2.2. Відображення $\varphi_i : M \xrightarrow{K_i} E$ множини відкритих текстів у множині криптограм

Зафіксуємо множину ключів $K^* = \{K_1^*, K_2^*, \dots, K_k^*\}$ у загальному випадку $K \neq K^*$. Всі елементи множини зворотних відображень $\varphi^{-1} = \{\varphi_1^{-1}, \varphi_2^*, \dots, \varphi_k^*\}$ задаються відповідним ключем: $\varphi_i^{-1} : E \xrightarrow{K_i^*} M$.

Кожне конкретне відображення φ_i^{-1} з множини φ^{-1} відповідає способу розшифрування за допомогою ключа K_i^* .

Якщо відомо ключ K_i^* , то в результаті розшифрування можлива єдина відповідь – елемент множини M .

Таким чином, в абстрактне *визначення криптосистеми* входять такі множини: $M, E, \varphi, \varphi^{-1}, K, K^*$ (множини відкритих текстів і криптограм, множини прямих і зворотних відображень, множини ключів).

Якщо при цьому $K \neq K^*$, то система *асиметрична*. Навпаки, якщо $K = K^*$ – *симетрична*. Відповідно до принципу Кірхгофа стійкість секретної системи повинна базуватися тільки на збереженні в таємниці від противника ключа.

На рис. 2.3 подана структурна схема секретної системи [22, 23].

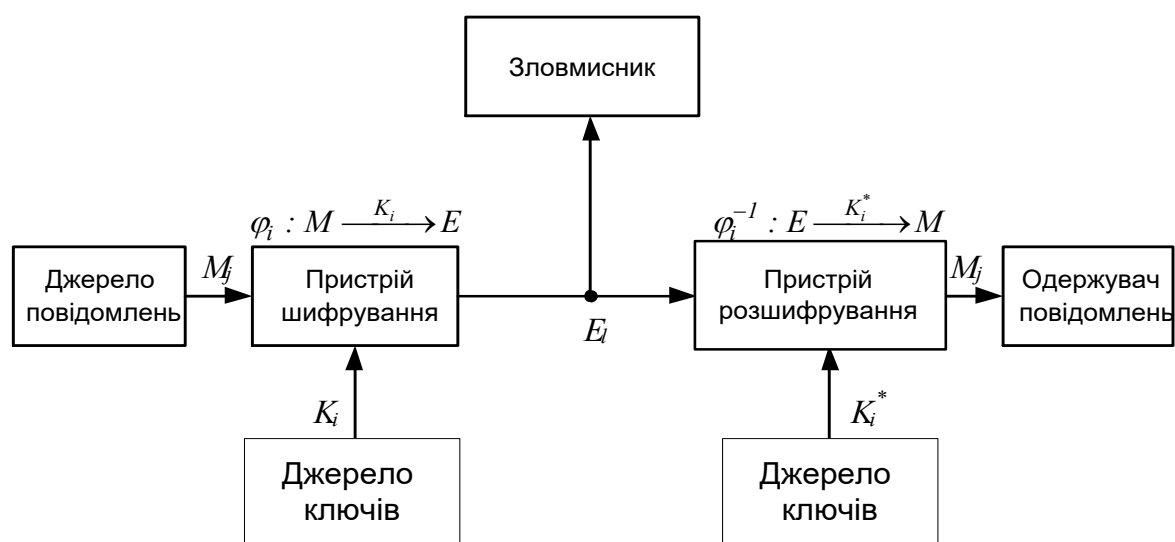


Рис. 2.3. Структурна схема секретної системи

Джерело повідомлень породжує потік повідомлень із множини M . Кожне повідомлення зображується конкретною реалізацією деякого випадкового процесу, що описує роботу джерела повідомлень. Кожному повідомленню $M_i \in M = \{M_1, M_2, \dots, M_m\}$ відповідає ймовірність $P(M_i)$. Розподіл ймовірностей випадкового процесу задається сукупним розподілом ймовірностей випадкових величин, тобто множиною ймовірностей:

$$P_M = \{P(M_1), P(M_2), \dots, P(M_m)\}. \quad (2.13)$$

Джерело ключів породжує потік ключів з множини K і/або K^* . Кожному ключу $K_i \in K = \{K_1, K_2, \dots, K_k\}$ відповідає деяка ймовірність $P(K_i)$, а кожному $K_i^* \in K^* = \{K_1^*, K_2^*, \dots, K_k^*\}$ відповідає ймовірність $P(K_i^*)$. Випадковий процес вироблення ключів задається множинами ймовірностей

$$P_K = \{P(K_1), P(K_2), \dots, P(K_k)\}, \quad (2.14)$$

$$P_{K^*} = \{P(K_1^*), P(K_2^*), \dots, P(K_k^*)\}. \quad (2.15)$$

Множини значень апіорних ймовірностей (3.63 – 3.65) утворюють апіорні знання супротивника про джерело повідомлень і джерело ключів, відповідно. Фактично ці множини характеризують апіорні знання супротивника щодо можливої “слабкості” секретної системи.

Вибір ключа K_i визначає конкретне відображення φ_i з множини відображень Φ . За допомогою відображення φ_i , що відповідає обраному ключу K_i , за повідомленням, що M_i надійшло, формується криптограма:

$$E_i = \varphi_i(K_i, M_i).$$

Криптограма E_i передається в точку прийому по деякому каналу й може бути перехоплена противником. На прийомному кінці за допомогою зворотного відображення φ_i^{-1} (заданого ключем K_i^*) із криптограми E_i відновлюється первісне повідомлення:

$$M_i = \varphi_i^{-1}(K_i^*, E_i).$$

Якщо противник перехопить криптограму E_i , він може з її допомогою спробувати обчислити апостеріорні ймовірності різних можливих повідомлень

$$P_{M|E_i} = \{P(M_1|E_i), P(M_2|E_i), \dots, P(M_m|E_i)\} \quad (2.16)$$

і різних можливих ключів

$$P_{K|E_i} = \{P(K_1|E_i), P(K_2|E_i), \dots, P(K_k|E_i)\}, \quad (2.17)$$

які могли бути використані при формуванні криптограми E_i .

Множини апостеріорних імовірностей (2.16 – 2.17) утворюють апостеріорні знання противника про ключі $K = \{K_1, K_2, \dots, K_k\}$ й повідомлення $M = \{M_1, M_2, \dots, M_m\}$ після перехоплення криптограми E_1 . Фактично, множини $P_{K|E_1}$ і $P_{M|E_1}$ становлять множини припущень, яким приписані відповідні ймовірності.

Основними показниками ефективності секретних систем прийнято вважати наступні:

1. *Криптографічна стійкість* (кількість таємності), яку оцінюють як складність рішення завдання криптоаналізу найкращим відомим методом. Деякі секретні системи є зробленими в тому розумінні, що положення противника не полегшується в результаті перехоплення будь-якої кількості повідомлень. Інші системи, хоча й дають супротивникові деяку інформацію при перехопленні чергової криптограми, але не допускають єдиного “рішення”. Системи, що допускають єдине рішення, дуже різноманітні як за витратою часу й сил, необхідних для одержання цього рішення, так і за кількістю матеріалу, який необхідно перехопити для одержання єдиного рішення.

2. *Обсяг ключових даних*. Якщо система симетрична, то ключ загальний для всіх користувачів системи і його потрібно передавати по закритих каналах зв'язку, тобто він не повинен бути занадто великим, щоб його було легко доставити, і не повинен бути занадто малим, щоб його не можна було зламати повним перебором. Якщо система несиметрична, то один із ключів можна зробити загальнодоступним і передавати його по відкритих каналах зв'язку.

3. *Складність виконання прямого й зворотного криптографічного перетворення* (шифрування і розшифрування повідомлень). Ці операції повинні бути по можливості простими і легко реалізованими на практиці.

4. *Розростання кількості помилок*. У деяких типах шифрів помилка в одній букві, допущена при шифруванні або передачі, призводить до великої кількості помилок у розшифруванні тексту. Такі помилки розростаються в результаті операції розшифрування, викликаючи значну втрату інформації, часто вимагаючи повторної передачі криптограми. Бажано мінімізувати це зростання кількості помилок.

5. Збільшення обсягу повідомлення. У деяких типах секретних систем обсяг повідомлення збільшується в результаті операції шифрування. Цей небажаний ефект потрібно мінімізувати.

2.3. Прості шифри

З поширенням писемності з'явилась потреба в обміні листами і повідомленнями, що викликало необхідність приховування вмісту письмових повідомлень від сторонніх. Методи приховування вмісту письмових повідомлень можна розділити на три групи. До першої групи відносяться методи маскування або стеганографії, які здійснюють приховання самого факту наявності повідомлення; другу групу становлять різні методи тайнопису або криптографії (від грецьких слів *kryptos* – таємний і *grapho* – пишу); методи третьої групи орієнтовані на створення спеціальних технічних обладнань, засекречування інформації.

Історія криптографії – порівнюється з історією людської мови. Більше того, спочатку писемність сама по собі була своєрідною криптографічною системою, тому що в прадавніх суспільствах нею володіли тільки обрані.

Розвитку тайнопису сприяли війни. Письмові накази й повідомлення обов'язково шифрувалися, щоб полон кур'єрів не дозволив противнику одержати важливу інформацію. Наприклад, римський імператор Цезар користувався у своїй військовій і особистій переписці шифром, сутність якого полягала в заміні кожної букви латинської мови на наступну букву алфавіту. Тоді знаменита фраза: “VENI, VIDI, VICI” (“Прийшов, побачив, переміг”), якою Цезар сповістив одного зі своїх друзів у Римі про швидко здобуту їм перемогу, у зашифрованому вигляді буде: “XFOJ, XJEJ, XJDJ”.

Практично одночасно із криптографією став розвиватися й криптоаналіз – наука про розкриття шифрів (ключів) за шифртекстом.

В історії криптографії умовно можна виділити чотири етапи: наївний, формальний, науковий, комп'ютерний.

Для наївної криптографії (до початку XVI ст.) характерне використання будь-яких, зазвичай примітивних, способів заплутування противника щодо змісту шифрованих текстів. На початковому етапі для захисту інформації використовувалися методи кодування й стеганографії, які схожі, але не тотожні криптографії.

Більшість із використовуваних шифрів зводилися до перестановки або моноалфавітної підстановки. Одним з перших зафіксованих прикладів є шифр Цезаря, що полягає в заміні кожної букви вихідного тексту на іншу, що знаходиться від неї в алфавіті на конкретну кількість позицій. Інший шифр, полібіанський квадрат, авторство якого приписується грецькому письменникові Полібію, є загальною моноалфавітною підстановкою, яка проводиться за допомогою випадково заповненої алфавітом квадратною таблицею (для грецького алфавіту розмір становить 5×5). Кожна буква вихідного тексту замінюється на букву, що стоїть у квадраті знизу від неї.

Етап формальної криптографії (кінець XV – початок XX ст.) пов'язаний з появою формалізованих стійких до ручного криптоаналізу шифрів. В європейських країнах це відбулося в епоху Відродження, коли розвиток науки й торгівлі викликав попит на надійні способи захисту інформації. Важлива роль на цьому етапі належить Леону Батісту Альберту, італійському архітектору, який одним з перших запропонував багатоалфавітну підстановку. Даний шифр, що одержав ім'я дипломата XVI ст. Блеза Віжінера, полягав у послідовному “додаванні” букв вихідного тексту із ключем (процедуру можна полегшити за допомогою спеціальної таблиці). Його робота “Трактат про шифр” (1466 р.) вважається першою науковою працею з криптології.

Однією з перших друкованих праць, у якій узагальнені і сформульовані відомі на той момент алгоритми шифрування, є праця “Поліграфія” (1508 р.) німецького абата Іоганна Трісемуса. Йому належать два невеликих, але важливих відкриття: спосіб заповнення полібіанського квадрата (перші позиції заповнюються за допомогою ключового слова, що легко запам'ятовується, інші – буквами, що залишилися в алфавіті) і шифрування пари букв (біграм).

Простим, але стійким способом багатоалфавітної заміни (підстановки біграм) є шифр Плейфера, який був відкритий на початку XIX ст. Чарльзом Уїтстоном. Йому належить і важливе вдосконалення – шифрування “подвійним квадратом”. Шифри Плейфера і Уїтстона використовувалися аж до Першої світової війни, тому що із труднощами піддавалися ручному криптоаналізу.

У XIX ст. голландець Керкхоф сформулював головну вимогу до криптографічних систем, яка залишається актуальним і на сьогодні: таємність шифрів повинна бути заснована на таємності ключа, але не алгоритмі.

Нарешті, останнім словом у донаучній криптографії, яке забезпечило ще більш високу криптостійкість, а також дозволило автоматизувати (у змісті механізувати) процес шифрування, стали роторні криптосистеми.

Однією з перших подібних систем стала винайдена в 1790 р. Томасом Джеферсоном, майбутнім президентом США, механічна машина. Багатоалфавітна підстанова за допомогою роторної машини реалізується варіацією взаємного положення обертових роторів, кожний з яких здійснює “прошиту” у ньому підстановку.

Практичне поширення роторні машини одержали тільки на початку ХХ ст. Однією з перших практично використовуваних машин стала німецька Enigma, розроблена в 1917 р. Едвардом Хеберном і вдосконалена Артуром Кірхом. Роторні машини активно використовувалися під час Другої світової війни. Крім німецької машини Enigma використовувалися також обладнання Sigaba (США), Турех (Великобританія), Red, Orange і Purple (Японія). Роторні системи – вершина формальної криптографії, тому що відносно просто реалізовували дуже стійкі шифри. Успішні крипто-атаки на роторні системи стали можливі тільки з появою ЕОМ на початку 40-х рр.

Головна відмінна риса наукової криптографії (1930 – 60-ті рр.) – поява криптосистем зі строгим математичним обґрунтуванням криптостійкості. До початку 30-х рр. остаточно сформувалися розділи математики, що є науковою основою криптології: теорія ймовірностей і математична статистика, загальна алгебра, теорія чисел, почали активно розвиватися теорія алгоритмів, теорія інформації, кібернетика. Своєрідним вододілом стала робота Клода Шенона “Теорія зв’язку в секретних системах” (1949 р.), яка підвела наукову базу під криптографію й криптоаналіз. Відтоді почали говорити про криптологію (від грецького *kryptos* – таємний і *logos* – повідомлення) – науку про перетворення інформації для забезпечення її таємності. Етап розвитку криптографії і криптоаналіза до 1949 р. стали називати донаучною криптологією. Шенон увів поняття “розсіювання” і “перемішування”, обґрунтував можливість створення як завгодно стійких криптосистем.

У 1960-х рр. провідні криптографічні школи підійшли до створення блокових шифрів, ще більш стійких порівняно з роторними криптосистемами, однак тих, що допускають практичну реалізацію тільки у вигляді цифрових електронних обладнань.

Комп’ютерна криптографія (з 1970-х рр.) зобов’язана своєю появою обчислювальним засобам із продуктивністю, достатньою для реалізації

криптосистем, що забезпечують при великій швидкості шифрування для кількох порядків більш високу криптостійкість, ніж “ручні” і “механічні” шифри.

Першим класом криптосистем, практичне застосування яких стало можливе з появою потужних і компактних обчислювальних засобів, стали блокові шифри. В 70-х рр. був розроблений американський стандарт шифрування DES (прийнятий в 1978 р.). Один з його авторів, Хорст Фейстель (співробітник IBM), описав модель блокових шифрів, на основі якої були побудовані інші, більш стійкі симетричні криптосистеми, у тому числі вітчизняний стандарт шифрування ГОСТ 28147–89.

З появою DES збагатився й криптоаналіз. Для атак на американський алгоритм було створено кілька нових видів криптоаналізу (лінійний, диференціальний і т.інш.), практична реалізація яких знову ж була можлива тільки з появою потужних обчислювальних систем.

У середині 70-х рр. XX сторіччя відбувся справжній прорив у сучасній криптографії – поява асиметричних криптосистем, які не вимагали передачі секретного ключа між сторонами. Тут відправною точкою прийнято вважати роботу, опубліковану Уїтфілдом Діфі і Мартіном Хелманом у 1976 р. за назвою “Нові напрямки в сучасній криптографії”. В роботі вперше сформульовані принципи обміну шифрованою інформацією без обміну секретним ключем. Незалежно до ідеї асиметричних криптосистем підійшов Ральф Мерхнулі. Декількома роками пізніше Рон Рівест, Аді Шамір і Леонард Адлеман відкрили систему RSA, першу практичну асиметричну криптосистему, стійкість якої була заснована на проблемі факторизації більш простих чисел. Асиметрична криптографія відкрила відразу кілька нових прикладних напрямків, зокрема систему електронного підпису (ЕП) і електронних грошей.

У 1980 – 90-ті рр. з’явилися зовсім нові напрямки криптографії: імовірнісне шифрування, квантова криптографія та інші. Усвідомлення їх практичної цінності ще попереду. Актуальним залишається і завдання вдосконалення симетричних криптосистем. У цей же період були розроблені нефайстелевські шифри (SAFER, RC6 і ін.), а в 2000 р. після відкритого міжнародного конкурсу був прийнятий новий національний стандарт шифрування США – AES.

Криптографія є одним з найбільш потужних засобів забезпечення конфіденційності й контролю цілісності інформації. В багатьох відносинах вона займає центральне місце серед програмно-технічних регуляторів безпеки. Наприклад, для портативних комп’ютерів, фізично захистити які

вкрай важко, тільки криптографія дозволяє гарантувати конфіденційність інформації навіть у випадку крадіжки.

Метод криптографії можна визначити як деяку множину відображень одного простору (простору можливих повідомлень) в інший простір (простір можливих криптограм). Кожне конкретне відображення із цієї множини відповідає шифруванню за допомогою конкретного ключа. Далі подані визначення цих понять.

Повідомлення – текст, який необхідно зробити незрозумілим для сторонніх, тобто порушити значеннєву логіку. Такі повідомлення називатимемо вихідними повідомленнями або відкритим текстом

Криптограма – повідомлення з перекрученим логічним змістом.

Шифрування даних – процес перетворення відкритих даних у зашифровані дані (шифротекст, криптограму) за допомогою шифру.

Шифр – сукупність оборотних перетворень множини можливих відкритих даних у множину можливих шифротекстів, здійснюваних за певними правилами із застосуванням ключів.

Ключ – конкретний секретний стан деякого параметра (параметрів), що забезпечує вибір одного перетворення із сукупності можливих для використовуваного методу шифрування.

Криптоаналіз – галузь криптології, у якій розглядаються питання зламу шифрів з метою відновлення інформації у відкритому виді або фальсифікації шифрованої інформації, яка згодом повинна бути прийнята як справжня.

Симетричні алгоритми поділяються на дві категорії. Одні з них обробляють текст побітно (іноді побайтно) і називаються потоковими алгоритмами або потоковими шифрами. Ті ж, які працюють із групами бітів відкритого тексту, називаються блоковими алгоритмами (шифрами).

Найпростіші симетричні шифри засновані на найпростіших арифметичних операціях підстановки, перестановки, елементах модульної арифметики. Тому всі прості симетричні шифри можна розділити на дві групи шифрів:

Перестановчі та підстановчі шифри: одноалфавітні, поліалфавітні.

Перестановчі шифри

Всі перестановчі шифри засновані на алгебраїчній операції, називаної перестановкою, внаслідок виконання якої здійснюється шифрування відкритого тексту і розшифрування криптограми.

Перестановкою називається розташування в певному порядку k елементів упорядкованої множини M .

Перестановчий шифр із ключовим словом

Букви відкритого тексту записуються в клітки прямокутної таблиці у її рядки. Букви ключового слова пишуться над стовпцями й указують порядок цих стовпців (по зростанню номерів букв в алфавіті). Щоб одержати зашифрований текст, треба виписувати букви по стовпцях з обліком їх алфавітного порядку:

Відкритий текст: захист інформації

Ключ: шифр

ш	и	ф	р
4	1	3	2
з	а	щ	и
т	а		и
н	ф	о	р
м	а	ц	и
и			

Криптограма: аафа_ири_щ_оц_зтнми

Ключове слово (послідовність стовпців) відоме адресату, який легко зможе розшифрувати повідомлення.

Матрична перестановка

Матрична перестановка становить ускладнену перестановку. Для цього відкритий текст записується в матрицю за певним ключем $k_1 = \{1, 2, \dots, n\}$, який залежить від довжини тексту. Криптограма виходить при зчитуванні із цієї матриці за ключем $k_2 = \{1, 2, \dots, m\}$. Розмірність матриці $n \times m$.

Відкритий текст: "ШИФРУВАННЯ_ПЕРЕСТАНОВКОЮ".

Ключі: k_1 5-3-1-2-4-6; k_2 4-2-3-1.

Матриця складається із шести рядків і чотирьох стовпців 6×4 .

Запис в рядках відповідно до ключа k_1 , читання по стовпцях відповідно до ключа k_2 .

1	и	е	_	п
2	е	р	е	с
3	о	в	а	н
4	т	а	н	о
5	ш	и	ф	р
6	в	к	о	й
k_1/ k_2	1	2	3	4

Криптограма:

“ПСНОРЙЕРВАИК_ЕАНФОИЕОТШВ”.

Через те що символи криптотексту ті ж, що й у відкритому тексті, частотний аналіз покаже, що кожна буква зустрічається приблизно з тією ж частотою, що й звичайно. Це дає криптоаналітику інформацію про те, що це перестановчий шифр. Застосування до криптотексту іншого переставного фільтра значно підвищить безпеку. Існують й більш складні перестановчі шифри. Із застосуванням комп'ютера можна розкрити майже увесь їх зміст.

Хоча багато сучасних алгоритмів використовують перестановку, з цим зв'язана проблема використання великого обсягу пам'яті, а також іноді потрібна робота з повідомленнями певного розміру. Тому частіше використовують підстановчі шифри.

Підстановчі шифри

Підстановкою називається взаємно-однозначне відображення f деякої кінцевої множини m на множину m' або на себе. При цьому в підстановці беруть участь всі елементи множини m та m' .

Одноалфавітні шифри

Шифр простої заміни

У процесі шифрування здійснюється перетворення відкритого тексту m довжини l таким чином, що кожний символ замінюється на деякий інший символ. При цьому однаковим символам у відкритому тексті відповідають однакові символи криптотекста, а різним символам – різні. Ключем є таблиця, в якій установлюється правило заміни символів, тобто кожному символу a алфавіту A , ($a \in A$) ставиться у відповідність символ $b \neq a$ із цього ж алфавіту A . Позначимо операцію переходу $\rightarrow$. Тоді правило переходу можна подати у вигляді:

$$\alpha_i, \alpha_j \in A, \alpha_i \rightarrow \alpha_j, \text{ при } i \neq j', \text{ де } i, j \in [0, \dots, |A|],$$

де $|A|$ – потужність алфавіту, тобто кількість елементів в алфавіті.

Візьмемо алфавіт української мови й позначимо його як алфавіт $A_{\text{укр}}$.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$A_{\text{укр}}$	а	б	в	г	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я
К	й	ц	у	к	є	н	г	ш	щ	з	х	є	ф	и	в	а	п	р	о	л	д	ж	ї	я	ч	с	м	ь	т	і	б	ю

Відкритий текст: захист_інформації

Криптограма: щйчздж_хпярласхе

Шифр Цезаря

У шифрі Цезаря кожна буква замінюється на букву, що перебуває на k символів правіше за модулем, рівним кількості букв в алфавіті ($k = 3, n = 32$)

$C_k(j) = (j + k) \pmod{n}$, n – кількість букв в алфавіті.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
А _{укр}	а	б	в	г	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я

Очевидно, що зворотною підстановкою є:

$$C_{k^{-1}}(j) = C_{n-k} = (j + n - k) \pmod{n}.$$

Афіна криптосистема

Узагальненням системи Цезаря є афінна криптосистема. Вона визначається двома числами a й b , де $0 \leq a, b \leq n - 1$, n – потужність алфавіту A . Числа a і n повинні бути взаємно прості, $\text{НСД}(a, n) = 1$. Відповідними замінами є:

$$\begin{aligned} A_{a,b}(j) &= (a*j + b) \pmod{n} \\ A^{-1}_{a,b}(j) &= (j - b)*a^{-1} \pmod{n}. \end{aligned}$$

Зворотну заміну також можна одержати просто помінявши місцями рядки в таблиці замін.

Взаємна простота a і n необхідна для бієктивності відображення, а якщо ні, то можливі відображення різних символів в один, що призведе до неоднозначності розшифрування.

Шифр Цезаря із ключовим словом

У даному різновиді шифру Цезаря ключ задається числом k ($0 \leq k \leq n - 1$) і коротким ключовим словом або пропозицією. Випишується алфавіт, а під ним, починаючи з k -ої позиції, ключове слово. Букви, що залишилися, записуються за абеткою після ключового слова. У підсумку одержуємо підстановку для кожної букви. Вимога, щоб усі букви ключового слова були різними, не обов'язкова – можна записувати ключове слово без повторення однакових букв. Кількість ключів у системі Цезаря із ключовим словом рівна $n!$.

Шифр Плейфера

Полігамна заміна формується з одного алфавіту за допомогою спеціальних правил. Як приклад розглянемо шифр Плейфера.

У цьому шифрі алфавіт розташовується в матриці. Відкритий текст M розбивається на пари символів m_i, m_{i+1} . Кожна пара символів відкритого тексту замінюється на парі символів з матриці в такий спосіб:

якщо символи перебувають в одному рядку, то кожний із символів пари замінюється на наступний, який знаходиться праворуч від нього (за останнім символом у рядку впливає перший);

якщо символи перебувають в одному стовпці, то кожний символ пари замінюється на символ, розташований нижче його в стовпці (за останнім нижнім символом впливає верхній);

якщо символи пари перебувають у різних рядках і стовпцях, то вони вважаються протилежними кутами прямокутника. Символ, що перебуває в лівому куті, замінюється на символ, що стоїть в іншому лівому куті; заміна символу, що перебуває в правому куті, здійснюється аналогічно;

якщо у відкритому тексті зустрічаються два однакових символи підряд, то перед шифруванням між ними вставляється спеціальний символ (наприклад, тире).

Відкритий текст: "ШИФР_ПЛЕЙФЕРА_".

Матриця алфавіту подана на рис. 2.4.

А	Х	Б	М	Ц	В
Ч	Г	Н	Ш	Д	О
Е	Щ	,	Х	У	П
.	З	Ъ	Р	И	Й
С	Ь	К	Э	Т	Л
Ю	Я		Ы	Ф	-

ШИ	РД
ФР	ЫИ
_П	,-
ЛЭ	СТ
ЙФ	-И
ЕР	.Х
А_	ЧС

Рис. 2.4. Матриця алфавіту

Шифротекст: "РДЫИ,-СТ-І.ХЧС"

Багатоалфавітні шифри

Багатоалфавітні підстановочні шифри були винайдені Ліном Батістом (LeonBattista) в 1568 році. Основна ідея багатоалфавітних систем полягає в тому, що протягом усього тексту та сама буква може бути зашифрована по-різному. Тобто заміни для букви вибираються з багатьох алфавітів залежно від положення в тексті. Це є гарним захистом від простого підрахунку частот, тому що не існує єдиного маскування для

кожної букви в криптотексті. У даних шифрах використовуються множинні однобуквенні ключі, кожний з яких використовується для шифрування одного символу відкритого тексту. Першим ключем шифрується перший символ відкритого тексту, другим – другий і т. д. Після використання всіх ключів вони повторюються циклічно.

Шифр Віженера

Однією з найстарших і найбільш відомих багатоалфавітних криптосистем є система Віженера, названа на честь французького криптографа Блейза Віженера (Vigenere). Цей метод був уперше опублікований в 1586 році. У даному шифрі ключ K задається набором з d букв, приналежних алфавіту A .

$$K = k_0, k_1 \dots k_{d-1}, \text{ де } k_i \in A.$$

Далі набори підписуються з повторенням під повідомленням M :

$$\begin{aligned} M &= m_0 m_1 \dots m_{d-1} m_d \dots m_t \\ K &= k_0 k_1 \dots k_{d-1} k_d \dots k_{t(\text{mod } d)}. \end{aligned}$$

Отриману послідовність складають з відкритим текстом за модулем потужності алфавіту $n = |A|$ за виразом:

$$C_i(M, K_d) = (m_i + k_i \text{ mod } d) \text{ mod } n.$$

Також букву шифртексту можна знаходити з наступної таблиці, як перетинання стовпця, обумовленого буквою відкритого тексту, і рядка, обумовленого буквою ключа (рис. 2.5).

У випадку, якщо ключ K складається з одного символу, $d = 1$, то одержуємо класичний шифр Цезаря.

Повторне застосування двох або більше шифрів Віженера буде називатися складеним шифром Віженера. Він має рівняння:

$$C_i^*(M, K_j^{d_j}) = (m_i + k_{0, i \text{ mod } d_0} + k_{1, i \text{ mod } d_1} + \dots + k_{j, i \text{ mod } d_j}) \text{ mod } n,$$

де $K_j^{d_j} = \{k_0^{d_0}, k_1^{d_1}, \dots, k_j^{d_j}\}$ – множина ключів різних періодів, що мають $d_0, d_1, \dots, d_j$ відповідно.

Букви відкритого тексту

ключи	Букви	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
		Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А
		В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б
		Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В
		Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г
		Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д
		Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е
		З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж
		И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З
		Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И
		К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й
		Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К
		М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л
		Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М
		О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н
		П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О
		Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
		С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р
		Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С
		У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т
		Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У
		Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф
		Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х
		Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц
		Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч
		Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш
		Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ
		Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ
		Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы
		Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь
		Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
		Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю

Рис. 2.5. Таблица Віженера

Шифр з автоключем з використанням відкритого тексту.

Подальшою модифікацією системи Віженера є система шифрів з автоключем (auto-key), приписувана математику XVI ст. Дж. Кардано, AUTOCLAVE. Шифрування починається за допомогою “первинного ключа” (який є справжнім ключем у нашому змісті) і триває за допомогою повідомлення, зміщеного на довжину первинного ключа, потім проводиться додавання за модулем n, рівному потужності алфавіту A.

Перетворення відкритого тексту в криптограму здійснюється за формулою:

$$C_i^*(M^l, K_j^{dj}) = \begin{cases} m_i + k_i \bmod n, i < d; \\ m_i + k_{i-d} \bmod n, i \geq d \end{cases}$$

Розшифрування проводиться за формулою:

$$M_i^*(C^l, K_j^{dj}) = \begin{cases} c_i + k_i \bmod n, i < d; \\ c_i + k_{i-d} \bmod n, i \geq d \end{cases}$$

Наприклад:

Повідомлення М	П Р И В Е Т П Р И М А Т У
Первинний ключ К	У Г П У
Автоключ	П Р И В Е Т П Р И
Шифротекст	С У Ч Х Ф У Ч Т Н Ю П У Ї

Легальне розшифровування не представляє праці: за первинним ключем виходить початок повідомлення, після чого знайдена частина вихідного повідомлення використовується як ключ.

Шифр із автоключем з використанням криптограми.

Як і в попередньому алгоритмі, шифрування здійснюється за допомогою “первинного ключа” К, а потім триває за допомогою отриманої криптограми $c_0, c_1, \dots$ при використанні “первинного ключа”. Для цього алгоритму формула перетворення відкритого тексту в криптограму здійснюється за формулою:

$$C_i^*(M^l, K_j^{dj}) = \begin{cases} m_i + k_i \bmod n, i < d; \\ m_i + k_{i-d} \bmod n, i \geq d \end{cases}$$

Розшифрування проводиться за формулою:

$$M_i^*(C^l, K_j^{dj}) = \begin{cases} c_i + k_i \bmod n, i < d; \\ c_i + k_{i-d} \bmod n, i \geq d \end{cases}$$

Поліалфавітна заміна

Даний клас шифрів об'єднав у собі два види математичних перетворень перестановки f і підстановки g .

За допомогою операції перестановки формується ключова матриця підстановки, що складається з декількох різних функцій f_i .

За допомогою підстановки g проводиться шифрування відкритого тексту, формування ключа, вибираються випадкові перестановки базового алфавіту.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Арос	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ы	ь	э	ю	я	
f ₀ (x)	й	ц	у	к	е	н	г	ш	щ	з	х	ъ	ф	ы	в	а	п	р	о	л	д	ж	э	я	ч	с	м	и	т	ь	б	ю
f ₁ (x)	м	и	т	ь	б	ю	й	ц	у	к	е	н	г	ш	щ	з	х	ъ	ф	ы	в	а	п	р	о	л	д	ж	э	я	ч	с
f ₂ (x)	р	о	л	д	ж	э	я	ч	с	м	и	т	ь	б	ю	й	ц	у	к	е	н	г	ш	щ	з	х	ъ	ф	ы	в	а	п
f ₃ (x)	ш	щ	з	х	ъ	ф	ы	в	а	п	р	о	л	д	м	и	т	ь	б	ю	й	ц	у	к	е	н	г	ж	э	я	ч	с

Ключ шифрування підстановкою $g = \{1, 3, 0\}$ довжиною $S = 3$.
Шифрування повідомлення.

$$C_i(M, F, g) = fg(imods)(m_i).$$

Розшифрування:

$$M_i(C, F, g) = fg(imods)(c_i).$$

2.4. Криптографічні примітиви й типи структур симетричної криптосистеми

2.4.1. Основні криптографічні примітиви

Алгоритми симетричного шифрування різняться способом, яким обробляється вихідний текст. Можливе шифрування блоками або шифрування потоком. Блок тексту розглядається як ненегативне ціле число, або як кілька незалежних ненегативних цілих чисел. Довжина блоку завжди вибирається рівною степеню двійки.

В більшості блокових алгоритмів симетричного шифрування використовуються наступні типи операцій:

таблична підстановка, при якій група бітів відображається в іншу групу бітів (це так звані S-box);

переміщення, за допомогою якого біти повідомлення переупорядковуються (це так звані P-box);

операція додавання за модулем 2, позначувана XOR або $\oplus$;

операція додавання за модулем 2^{32} або за модулем 2^{16} ;

циклічний зсув на деяку кількість бітів.

Ці операції циклічно повторюються в алгоритмі, утворюючи так звані раунди. Входом кожного раунду є вихід попереднього раунду і ключ, який отриманий за певним алгоритмом із ключа шифрування K .

Ключ раунду називається підключем. Кожний алгоритм шифрування може бути поданий способом, наведеним на рис. 2.6.

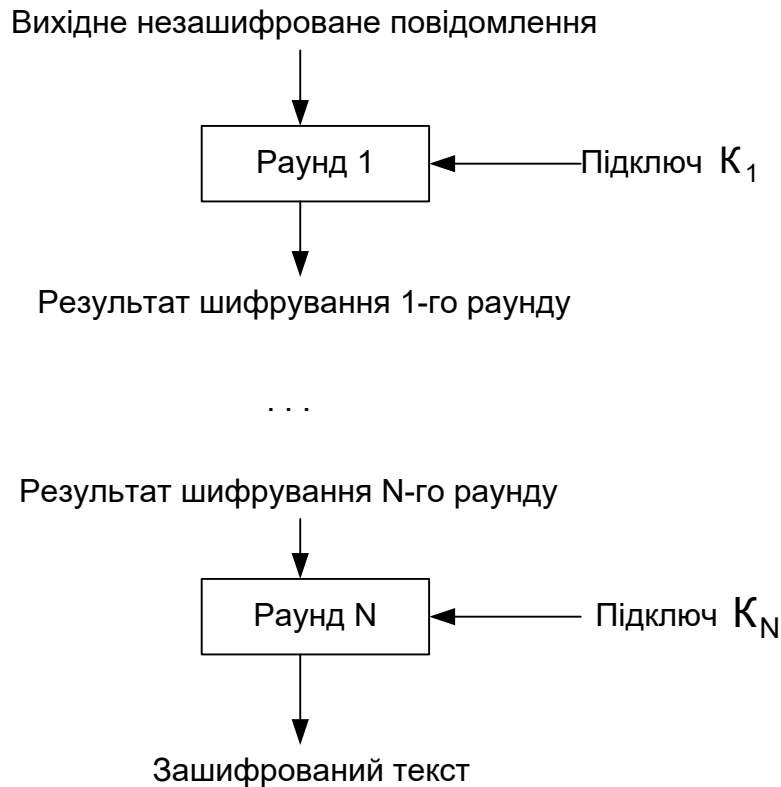


Рис. 2.6. Структура алгоритму симетричного шифрування

Використовувані критерії при розробці алгоритмів

Беручи до уваги перераховані вимоги, звичайно вважається, що алгоритм симетричного шифрування повинен:

- маніпулювати даними в більших блоках, переважно розміром 16 або 32 біти;

- мати розмір блоку 64 або 128 бітів;

- мати масштабований ключ до 256 бітів.

Використовувати прості операції, які ефективні на мікропроцесорах, тобто, що виключає додавання, табличні підстановки, або множення за модулем. Не повинні використовуватися зсув змінної довжини, побітних перестановок або умовних переходів.

Повинна бути можливість реалізації алгоритму на 8-бітному процесорі з мінімальними вимогами до пам'яті.

Використовувати заздалегідь обчислені підключі. На системах з більшою кількістю пам'яті ці підключі можуть бути заздалегідь обчислені для прискорення роботи. У випадку неможливості завчасного обчислення підключів повинне відбутися тільки вповільнення виконання. Завжди повинна бути можливість шифрування даних без яких-небудь попередніх обчислень.

Складатися зі змінного числа ітерацій. Для додатків з маленькою довжиною ключа недоцільно застосовувати велику кількість ітерацій для протистояння диференціальним й іншим атакам. Отже, повинна бути можливість зменшити число ітерацій без втрати безпеки (не більше ніж зменшений розмір ключа).

По можливості не мати слабких ключів. Якщо це неможливо, то кількість слабких ключів повинна бути мінімальною, щоб зменшити ймовірність випадкового вибору одного з них. Проте усі слабкі ключі повинні бути заздалегідь відомі, щоб їх можна було відбракувати в процесі створення ключа.

Задіяти підключі, які є однобічним ґешем ключа. Це дає можливість використовувати більші парольні фрази як ключ без шкоди для безпеки.

Не мати лінійних структур, які зменшують комплексність і не забезпечують вичерпний пошук.

Використовувати просту для розуміння розробку. Це дає можливість аналізу й зменшує закритість алгоритму.

Більшість блокових алгоритмів заснована на використанні мережі Файстеля, всі мають плоский простір ключів з можливим виключенням декількох слабких ключів.

2.4.2. Структура Файстеля алгоритму блокового симетричного шифрування

Блоковий алгоритм перетворює n -бітний блок незашифрованого тексту в n -бітний блок зашифрованого тексту. Кількість блоків довжини n рівна 2^n . Для того щоб перетворення було оборотним, кожний з таких блоків повинен перетворюватися у свій унікальний блок зашифрованого тексту. При маленькій довжині блоку така підстановка погано приховує статистичні особливості незашифрованого тексту. Якщо блок має довжину 64 біти, то він уже добре приховує статистичні особливості вихідного тексту. Але в цьому випадку перетворення тексту не може бути довільним у силу того, що ключем буде саме перетворення, що виключає ефективну як програмну, так і апаратну реалізацію.

Найбільш широке поширення одержали мережі Файстеля, тому що, з одного боку, вони задовольняють усім вимогам до алгоритмів симетричного шифрування, а з іншого – сторони, досить прості і компактні.

Мережа Файстеля має наступну структуру. Вхідний блок ділиться на підблоки рівної довжини, названі множинами. У випадку, якщо блок має довжину 64 біти, використовуються дві множини по 32 біти кожна. Кожна

множина обробляється незалежно від іншої, після чого здійснюється циклічний зсув всіх множин вліво. Таке перетворення виконується кілька циклів або раундів. У випадку двох множин кожний раунд має структуру, показану на рис. 2.7.

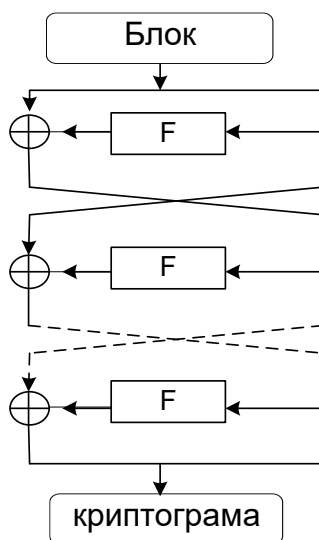


Рис. 2.7. I-й раунд мережі Файстеля

Функція F називається утворюючою. Кожний раунд складається з обчислення функції F для однієї множини і побітового виконання операції XOR результату F з іншою множиною. Після цього множини міняються місцями. Вважається, що оптимальна кількість раундів – від 8 до 32. Важливо те, що збільшення кількості раундів значно збільшує криптостійкість алгоритму. Можливо, ця особливість і вплинула на настільки активне поширення мережі Файстеля, тому що для більшої криптостійкості досить просто збільшити кількість раундів, не змінюючи сам алгоритм. Останнім часом кількість раундів не фіксується, а лише вказуються допустимі межі.

Мережа Файстеля є оборотною навіть у тому випадку, якщо функція F не є такою, тому що для розшифрування не потрібно обчислювати F^{-1} . Для розшифрування використовується той же алгоритм, але на вхід подається зашифрований текст, і ключі використовуються у зворотному порядку.

На сьогодні усе частіше використовуються різні різновиди мережі Фастеля для 128-бітного блоку із чотирма множинами. Збільшення кількості множин, а не розмірності кожної множини, пов'язане з тим, що найбільш популярними дотепер залишаються процесори з 32-розрядними словами, отже, оперувати 32-розрядними словами ефективніше, ніж з 64-розрядними.

Основною характеристикою алгоритму, побудованого на основі мережі Фейстеля, є функція F . Різні варіанти стосуються також початкового і кінцевого перетворень. Подібні перетворення, називані забілюванням (whitening), здійснюються для того, щоб виконати початкову рандомізацію вхідного тексту.

Перевага конструкції Файстеля полягає, у першу чергу, у тому, що пряме і зворотне криптографічні перетворення для методу блокового перетворення інформації мають ідентичну структуру, внаслідок чого апаратна реалізація буде більш компактною порівняно з методами, для яких пряме і зворотне перетворення відрізняються. Крім того, конструкція Файстеля порівняно з SPN-конструкцією дозволяє використовувати у функції, що шифрує, більш широкий набір перетворень, тому, що до них не пред'являється вимога оборотності.

2.4.3. SPN-структура алгоритму блокового симетричного шифрування

SPN-конструкція допускає послідовне застосування функції, що шифрує, яка працює не з частинами інформаційного блоку, а відразу з усім блоком (рис. 2.8).

Усі перетворення в SPN-конструкціях повинні бути оборотними, тому, що розшифрування проводиться шляхом виконання зворотних перетворень у зворотному порядку.

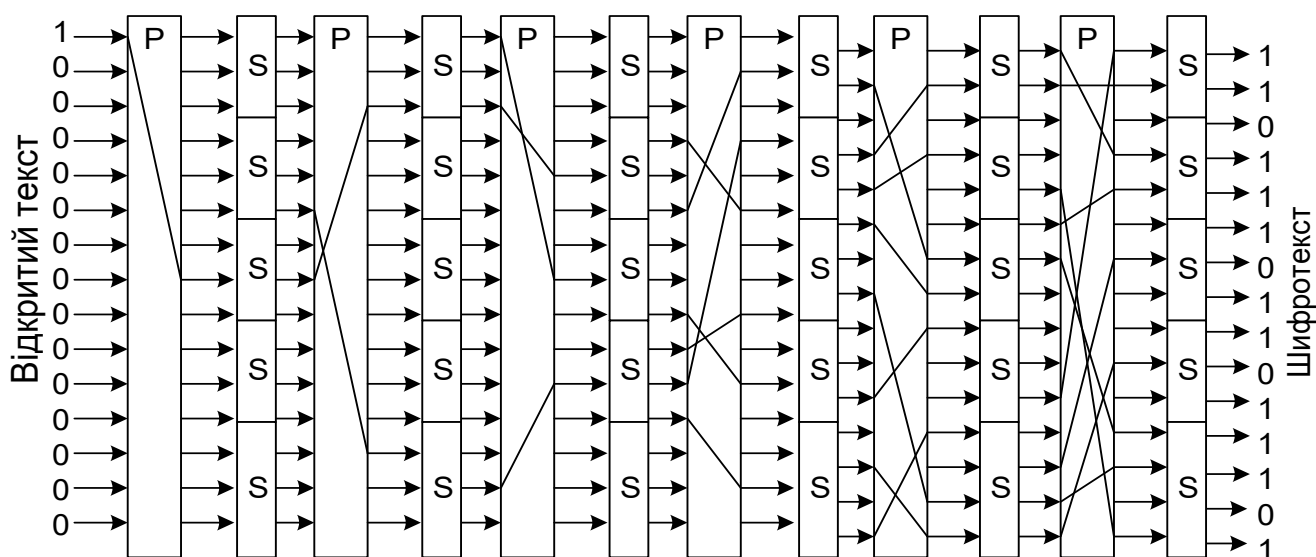


Рис. 2.8. Схеми SPN

Характерною рисою обох конструкцій блокового криптографічного перетворення інформації є використання шифруючих функцій операцій нелінійного перемішування (нелінійних S-блоків або нелінійних вузлів заміни). Зазначимо, що в якості функцій, що шифрують, можуть також додатково використовуватися операції розсіювання (бітові Р-перестановки) і операції введення таємності (додавання з ключем).

На рис. 2.9 представлені шифруючі функції для методів з 16-бітними інформаційними блоками, побудованими з використанням класичного ланцюга Файстеля (рис. 2.9 а – бієктивна функція, що шифрує, рис. 2.9 б (з використанням зжимаючо-розширювальних перетворень) і за схемою SPN (рис. 2.9 в).

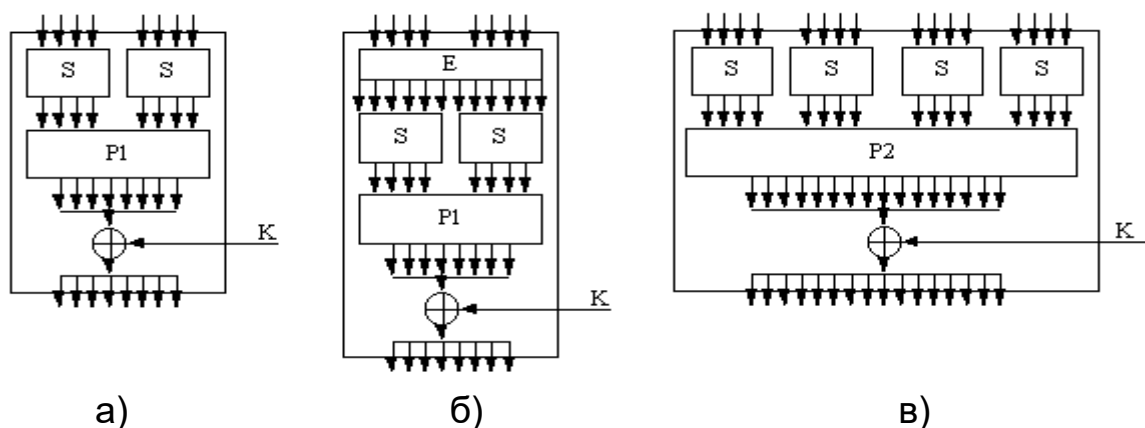


Рис. 2.9. Шифруючі функції блокових методів

Як видно з наведених даних, нелінійні вузли замін є вузлами, що визначають стійкість перетворення інформації і їх використання характерне як для методів, що використовують ланцюги Файстеля, так і для методів, що використовують SPN-структури. Стійкість методів блокового криптографічного перетворення інформації базується, насамперед, на стійкості нелінійних вузлів замін.

2.4.4. Типи структур поточкових алгоритмів шифрування

Серед методів потокового перетворення інформації найбільш широке поширення одержали методи, засновані на використанні схем з рівномірним рухом регістрів і без пам'яті.

Перевагами таких методів є простота як програмної, так і апаратної реалізацій, висока стійкість перетворень і висока швидкодія.

Більша частина методів потокового перетворення інформації підрозділяється на два більших класи методів (рис. 2.10, 2.11):

методи, побудовані на основі використання фільтрувальних генераторів;

методи, побудовані на використанні комбінувальних генераторів.

Методи потокового перетворення інформації, засновані на використанні фільтрувальних генераторів, припускають у загальному випадку використання лінійного рекурентного регістру (ЛРР), вихідна послідовність якого надходить на вхід нелінійного вузла ускладнення, що безпосередньо здійснює криптоперетворення.

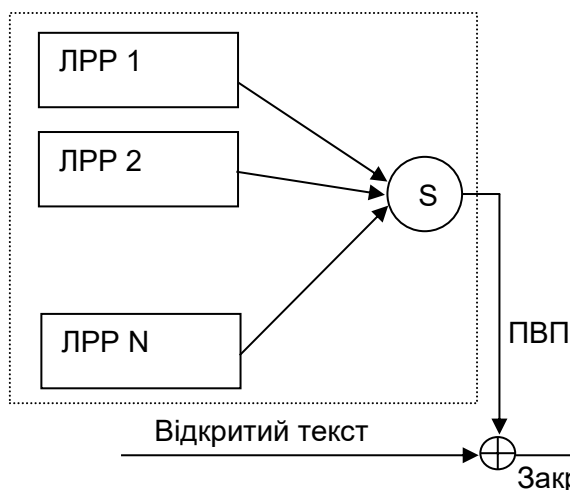


Рис. 2.10. Комбінувальний генератор

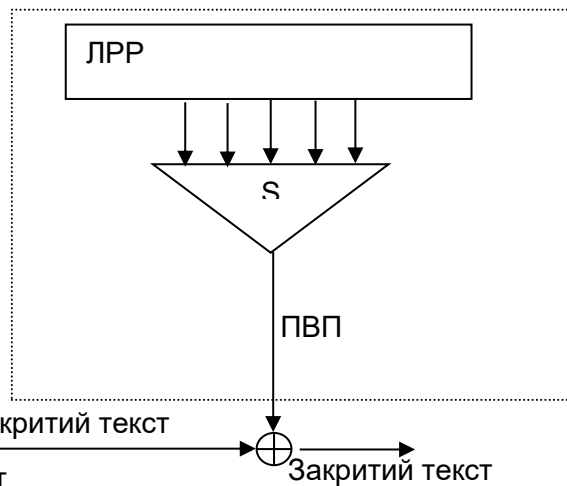


Рис. 2.11. Фільтрувальний-генератор

На кожному циклі перетворення здійснюється знімання даних з певних гнізд ЛРР, після чого виділені дані зазнають перетворення за допомогою криптографічного перетворення S (нелінійного вузла ускладнення). Результат операції підсумовується за модулем 2 (операція XOR) з відкритим тестом, і отримана в такий спосіб послідовність є шуканим закритим тестом.

Класичною схемою фільтрувального генератора вважається схема з використанням бітових ЛРР довжиною не менше 128 бітів, подана на рис. 2.10. Перевага фільтрувального генератора полягає, у першу чергу, у тому, що використання одного регістру зсуву робить просту апаратну реалізацію, що, в свою чергу, дозволяє добитися високої швидкості криптоперетворень. Саме даний клас методів використовується в додатках, до яких пред'являються тверді вимоги до швидкості перетворення/передачі даних.

Відмінність методів потокового перетворення інформації, заснованих на використанні комбінувальних генераторів від методів, заснованих на використанні фільтрувальних генераторів, полягає в тому, що вони припускають використання декількох паралельно працюючих

лінійних рекурентних регістрів, вихідні послідовності яких аналогічно надходять на вхід нелінійного вузла ускладнення. На кожному циклі перетворення здійснюється знімання даних з виходів ЛРР, після чого виділені дані зазнають перетворення за допомогою криптографічного перетворення S (нелінійного вузла ускладнення). Результат операції також підсумовується за модулем 2 (операція XOR) з відкритим тестом, і отримана в такий спосіб послідовність є шуканим закритим тестом.

Як і у випадку з методами блокового перетворення даних, стійкість методів потокового перетворення даних визначається стійкістю нелінійних вузлів заміни.

2.4.5. Математичні моделі нелінійних вузлів заміни у термінах булевої алгебри

Нелінійні вузли заміни блокових симетричних криптографічних засобів захисту інформації подаються у вигляді обладнання перетворення, реалізованого за допомогою двох комутаторів (рис. 2.12). При цьому один комутатор перетворює набір з n бітів $(a_1, a_2, \dots, a_n)$ в одну цифру за підставою 2^n , інший комутатор виконує зворотнє перетворення, тобто утворює набір з n бітів $(b_1, b_2, \dots, b_n)$. Таке обладнання потенційно може замінити будь-який вхідний набір даних $(a_1, a_2, \dots, a_n)$ на будь-який вихідний набір $(b_1, b_2, \dots, b_n)$.

Нелінійний вузол заміни (S -блок), схема перетворення даних якого подана на рис. 2.12, містить 2^n внутрішніх станів комутаторів, які можуть бути виконані $2^n!$ різними способами. Це означає, що існує $2^n!$ різних варіантів відповідних нелінійних вузлів (таблиць) заміни.

Формалізуємо процес перетворення даних у нелінійному вузлі заміни за допомогою математичної моделі, що дозволяє описати його внутрішню структуру і адекватно оцінити основні показники його ефективності.

Як видно з рис. 2.12, процес перетворення даних у нелінійному вузлі заміни у загальному випадку можна представити у вигляді деякого відображення:

$$\varphi: A \rightarrow B, \quad (2.18)$$

де A – множина можливих наборів $(a_1, a_2, \dots, a_n)$, що надходять на вхід S -блоку;

B – множина можливих наборів $(b_1, b_2, \dots, b_n)$, що надходять на вихід S -блоку.

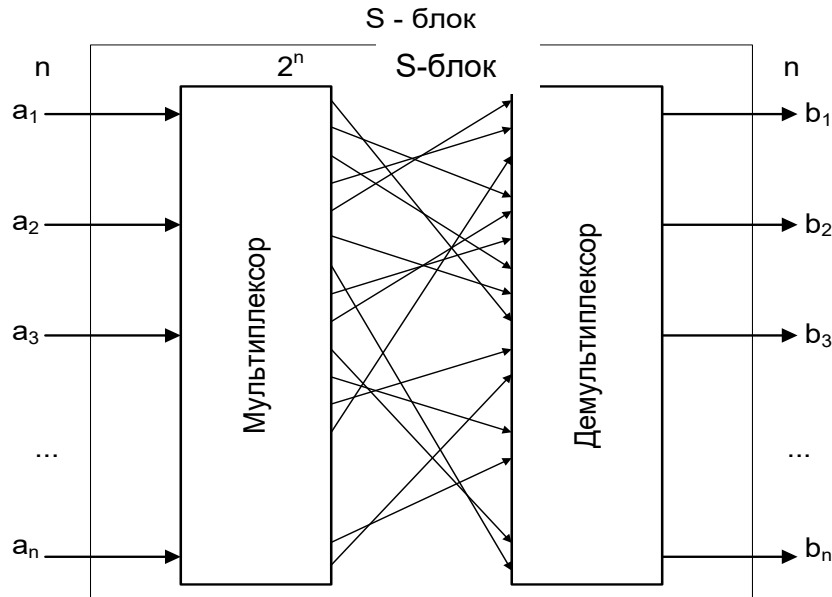


Рис. 2.12. Схема перетворення даних у нелінійному вузлі заміни

Властивості відображення (2.18) задаються внутрішньою структурою вузла заміни, тобто набором конкретних комутацій (з'єднань) виходів мультиплексора і входів демультиплексора.

Для формального аналітичного опису внутрішньої структури й оцінки властивостей нелінійного вузла заміни пропонується математична модель як абстрактна сукупність наступних елементів (математичної символіки):

1. Множина вхідних векторів:

$$A = \{A_1, A_2, \dots, A_i, \dots, A_{2^m}\}, A_i = (a_1^{(i)}, a_2^{(i)}, \dots, a_m^{(i)}) \in A, a_j^{(i)} \in GF(2), i = 1, \dots, 2^m, \\ j = 1, \dots, m, |A| = 2^m.$$

2. Множина вихідних векторів:

$$B = \{B_1, B_2, \dots, B_i, \dots, B_{2^n}\}, B_i = (b_1^{(i)}, b_2^{(i)}, \dots, b_n^{(i)}) \in B, b_j^{(i)} \in GF(2), i = 1, \dots, 2^m, j = 1, \dots, m, \\ , |B| = 2^n.$$

3. Множина відображень $\Phi = \{\varphi_1, \varphi_2, \dots, \varphi_i, \dots, \varphi_{2^m!}\}$ вхідних векторів з множини A в множину вихідних векторів B , тобто:

$$\varphi_i : A \rightarrow B, \varphi_i \in \Phi, i = 1, \dots, 2^m!, |\Phi| = 2^m!.$$

4. Кожне відображення $\varphi_i : A \rightarrow B$ з множини Φ параметризується сукупністю компонентних криптографічних булевих функцій:

$$F = \{f_1(x_1, \dots, x_m), f_2(x_1, \dots, x_m), \dots, f_u(x_1, \dots, x_m), \dots, f_n(x_1, \dots, x_m)\}, u = 1, \dots, n, |F| = n.$$

Кожний елемент множини запишемо в алгебраїчній нормальній формі:

$$F \left\{ \begin{array}{l} f_1(x_1, \dots, x_m) = c_{1,0} \oplus \bigoplus_{i=1}^m c_{1,i} x_i \oplus \bigoplus_{1 \leq i < j \leq m} c_{1,ij} x_i x_j \oplus \dots \oplus c_{1,12\dots m} x_1 x_2 \dots x_m; \\ f_2(x_1, \dots, x_m) = c_{2,0} \oplus \bigoplus_{i=1}^m c_{2,i} x_i \oplus \bigoplus_{1 \leq i < j \leq m} c_{2,ij} x_i x_j \oplus \dots \oplus c_{2,12\dots m} x_1 x_2 \dots x_m; \\ \dots \\ f_u(x_1, \dots, x_m) = c_{u,0} \oplus \bigoplus_{i=1}^m c_{u,i} x_i \oplus \bigoplus_{1 \leq i < j \leq m} c_{u,ij} x_i x_j \oplus \dots \oplus c_{u,12\dots m} x_1 x_2 \dots x_m; \\ \dots \\ f_n(x_1, \dots, x_m) = c_{n,0} \oplus \bigoplus_{i=1}^m c_{n,i} x_i \oplus \bigoplus_{1 \leq i < j \leq m} c_{n,ij} x_i x_j \oplus \dots \oplus c_{n,12\dots m} x_1 x_2 \dots x_m. \end{array} \right. \quad (2.19)$$

5. Система обмежень:

$$\left\{ \begin{array}{l} Cб_f = Cб_{f_1} \wedge Cб_{f_2} \wedge \dots \wedge Cб_{f_n} = Cб_{тр} \\ N_f = \min\{N_{f_1}, N_{f_2}, \dots, N_{f_n}\} \geq N_{тр} \\ Kl_f(k) = \min\{Kl_{f_1}(k), Kl_{f_2}(k), \dots, Kl_{f_n}(k)\} \geq Kl_{тр}(k) \\ КП_f(k) = \min\{КП_{f_1}(k), КП_{f_2}(k), \dots, КП_{f_n}(k)\} \geq КП_{тр}(k) \\ AC_f = \max\{AC_{f_1}, AC_{f_2}, \dots, AC_{f_n}\} \leq AC_{тр} \end{array} \right\}, \quad (2.20)$$

де: $Cб_f, N_f, Kl_f(k), КП_f(k), AC_f$ – значення показників збалансованості, нелінійності, кореляційного імунітету, критерію поширення і автокореляції, відповідно оцінені за критерієм мінімального ризику;

$Cб_{f_i}, N_{f_i}, Kl_{f_i}(k), КП_{f_i}(k), AC_{f_i}$ – значення показників збалансованості, нелінійності, кореляційного імунітету, критерію поширення і автокореляції булевої функції $f_i(x_1, \dots, x_m)$, $i = 1, 2, \dots, n$; $Cб_{тр}, N_{тр}, Kl_{тр}(k), КП_{тр}(k), AC_{тр}$ – необхідні значення показників збалансованості, нелінійності, кореляційного імунітету, критерію поширення і автокореляції відповідно.

6. Множина компонентних нелінійних булевих функцій $\bar{F} = \{\bar{f}_1(x_1, \dots, x_m), \bar{f}_2(x_1, \dots, x_m), \dots, \bar{f}_u(x_1, \dots, x_m), \dots, \bar{f}_{2^n-1}(x_1, \dots, x_m)\}$ як повна множина

функцій, отриманих лінійною комбінацією булевих функцій $f_i(x_1, \dots, x_m)$ з множини $F, u = 1, \dots, 2^n, |\bar{F}| = n$:

$$\left\{ \begin{array}{l} \bar{f}_1(x_1, \dots, x_m) = f_1(x_1, x_2, \dots, x_m); \\ \bar{f}_2(x_1, \dots, x_m) = f_2(x_1, x_2, \dots, x_m); \\ \dots \\ \bar{f}_n(x_1, \dots, x_m) = f_n(x_1, x_2, \dots, x_m); \\ \bar{f}_{n+1}(x_1, \dots, x_m) = f_1(x_1, x_2, \dots, x_m) \oplus f_2(x_1, x_2, \dots, x_m); \\ \dots \\ \bar{f}_{2^n-1}(x_1, \dots, x_m) = f_1(x_1, x_2, \dots, x_m) \oplus f_2(x_1, x_2, \dots, x_m) \oplus \dots \oplus f_n(x_1, x_2, \dots, x_m). \end{array} \right. \quad (2.21)$$

7. Система обмежень:

$$\left\{ \begin{array}{l} C\bar{b}^*_f = C\bar{b}_{\bar{f}_1} \wedge C\bar{b}_{\bar{f}_2} \wedge \dots \wedge C\bar{b}_{\bar{f}_{2^n-1}} = C\bar{b}^*_{tr} \\ N^*_f = \min\{N_{\bar{f}_1}, N_{\bar{f}_2}, \dots, N_{\bar{f}_{2^n-1}}\} \geq N^*_{tr} \\ KI^*_f(k) = \min\{KI_{\bar{f}_1}(k), KI_{\bar{f}_2}(k), \dots, KI_{\bar{f}_{2^n-1}}(k)\} \geq KI^*_{tr}(k) \\ КП^*_f(k) = \min\{КП_{\bar{f}_1}(k), КП_{\bar{f}_2}(k), \dots, КП_{\bar{f}_{2^n-1}}(k)\} \geq КП^*_{tr}(k) \\ AC^*_f = \max\{AC_{\bar{f}_1}, AC_{\bar{f}_2}, \dots, AC_{\bar{f}_{2^n-1}}\} \leq AC^*_{tr} \end{array} \right\}, \quad (2.22)$$

де: $C\bar{b}^*_f, N^*_f, KI^*_f(k), КП^*_f(k), AC^*_f$ – значення показників збалансованості, нелінійності, кореляційного імунітету, критерію поширення і автокореляції компонентних булевих функцій, відповідно оцінені за критерієм міні-мального ризику;

$C\bar{b}_{\bar{f}_i}, N_{\bar{f}_i}, KI_{\bar{f}_i}(k), КП_{\bar{f}_i}(k), AC_{\bar{f}_i}$ – значення показників збалансованості, нелінійності, кореляційного імунітету, критерію поширення і автокореляції компонентної булевої функції $\bar{f}_i(x_1, \dots, x_m), i = 1, 2, \dots, 2^n - 1$;

$C\bar{b}^*_{tr}, N^*_{tr}, KI^*_{tr}(k), КП^*_{tr}(k), AC^*_{tr}$ – необхідні значення показників збалансованості, нелінійності, кореляційного імунітету, критерію поширення і автокореляції до компонентних булевих функцій відповідно.

Всі функції з множини F задовольняють системі обмежень (2.15). Вузли заміни, відповідно до запропонованої моделі, задаються значеннями функцій $f_u(x_1, \dots, x_m)$, $u = 1, \dots, n$ з множини F , тобто значення на виході S -блоку задаються сукупністю рівностей:

$$\left\{ \begin{array}{l} b_1^{(l)} = f_1(a_1^{(l)}, \dots, a_m^{(l)}) = c_{1,0} \oplus \bigoplus_{i=1}^m c_{1,i} a_i^{(l)} \oplus \bigoplus_{1 \leq i < j \leq m} c_{1,ij} a_i^{(l)} a_j^{(l)} \oplus \dots \oplus c_{1,12\dots m} a_1^{(l)} a_2^{(l)} \dots a_m^{(l)}; \\ b_2^{(l)} = f_2(a_1^{(l)}, \dots, a_m^{(l)}) = c_{2,0} \oplus \bigoplus_{i=1}^m c_{2,i} a_i^{(l)} \oplus \bigoplus_{1 \leq i < j \leq m} c_{2,ij} a_i^{(l)} a_j^{(l)} \oplus \dots \oplus c_{2,12\dots m} a_1^{(l)} a_2^{(l)} \dots a_m^{(l)}; \\ \dots \\ b_u^{(l)} = f_u(a_1^{(l)}, \dots, a_m^{(l)}) = c_{u,0} \oplus \bigoplus_{i=1}^m c_{u,i} a_i^{(l)} \oplus \bigoplus_{1 \leq i < j \leq m} c_{u,ij} a_i^{(l)} a_j^{(l)} \oplus \dots \oplus c_{u,12\dots m} a_1^{(l)} a_2^{(l)} \dots a_m^{(l)}; \\ \dots \\ b_n^{(l)} = f_n(a_1^{(l)}, \dots, a_m^{(l)}) = c_{n,0} \oplus \bigoplus_{i=1}^m c_{n,i} a_i^{(l)} \oplus \bigoplus_{1 \leq i < j \leq m} c_{n,ij} a_i^{(l)} a_j^{(l)} \oplus \dots \oplus c_{n,12\dots m} a_1^{(l)} a_2^{(l)} \dots a_m^{(l)}. \end{array} \right. \quad (2.23)$$

Будь-яка функція

$$\bar{f}_u(x_1, \dots, x_m) \in \bar{F} = \{\bar{f}_1(x_1, \dots, x_m), \bar{f}_2(x_1, \dots, x_m), \dots, \bar{f}_{2^n-1}(x_1, \dots, x_m)\}, u = 1, 2, \dots, 2^n - 1,$$

отримана лінійною комбінацією функцій:

$$\begin{aligned} \bar{f}_l(x_1, x_2, \dots, x_m) &= f_i(x_1, x_2, \dots, x_m) \oplus f_j(x_1, x_2, \dots, x_m) \oplus \dots \oplus f_s(x_1, x_2, \dots, x_m), \\ f_i(x_1, x_2, \dots, x_m), f_j(x_1, x_2, \dots, x_m), \dots, f_s(x_1, x_2, \dots, x_m) &\in F, \end{aligned}$$

задовольняє системі обмежень (2.17) із граничними значеннями $СБ^*_{тр}, N^*_{тр}, KI^*_{тр}(k), КП^*_{тр}(k), АС^*_{тр}$.

Таким чином, математична модель нелінійних вузлів заміни блокових симетричних засобів захисту інформації на основі аналітичного опису основних структурних компонентів системи, що накладає обмеження з нелінійності, збалансованості, кореляційного імунітету, критерію поширення і автокореляції, дозволяє термінами булевої алгебри описувати внутрішню структуру нелінійних вузлів заміни, і оцінювати основні показники їх ефективності.

2.5. Алгоритми блокового симетричного шифрування DES, ГОСТ-28147

2.5.1. Алгоритм блокового симетричного шифрування DES

Принципи розробки

Найпоширенішим і найбільш відомим алгоритмом симетричного шифрування є DES (Data Encryption Standard). Алгоритм був розроблений у 1977 році, в 1980 році був прийнятий NIST (National Institute of Standards and Technology США) у якості стандарту (FIPS PUB 46).

DES є класичною мережею Фейштеля з двома галузями (рис. 2.13).

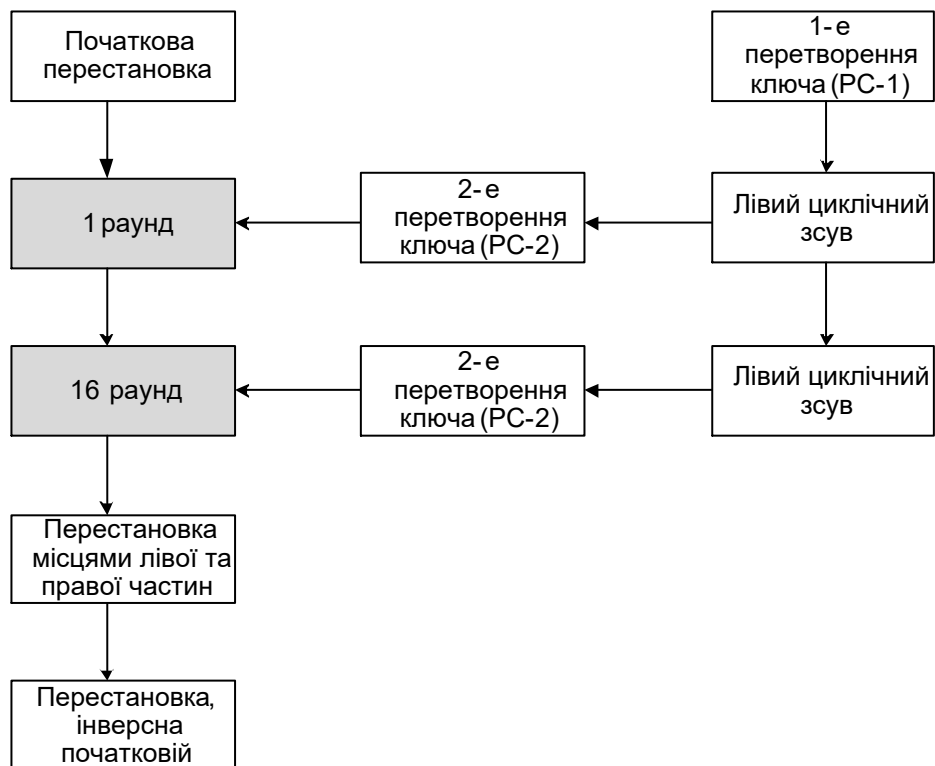


Рис. 2.13. Загальна схема DES

Дані шифруються 64-бітними блоками, використовуючи 56-бітний ключ. Алгоритм перетворює за кілька раундів 64-бітний вхід в 64-бітний вихід. Довжина ключа рівна 56 бітам.

Процес шифрування складається із чотирьох етапів. На першому з них виконується початкова перестановка (IP) 64-бітного вихідного тексту (забілювання), під час якої біти переупорядковуються відповідно до стандартної таблиці. Наступний етап складається з 16 раундів однієї й тієї ж функції, яка використовує операції зсуву і підстановки. На третьому

етапі ліва і права половини виходу останньої (16-ї) ітерації міняються місцями. Нарешті, на четвертому етапі виконується перестановка IP^{-1} результату, отриманого на третьому етапі. Перестановка IP^{-1} інверсна початковій перестановці.

На рис. 2.14 показаний спосіб, у якому використовується 56-бітний ключ.

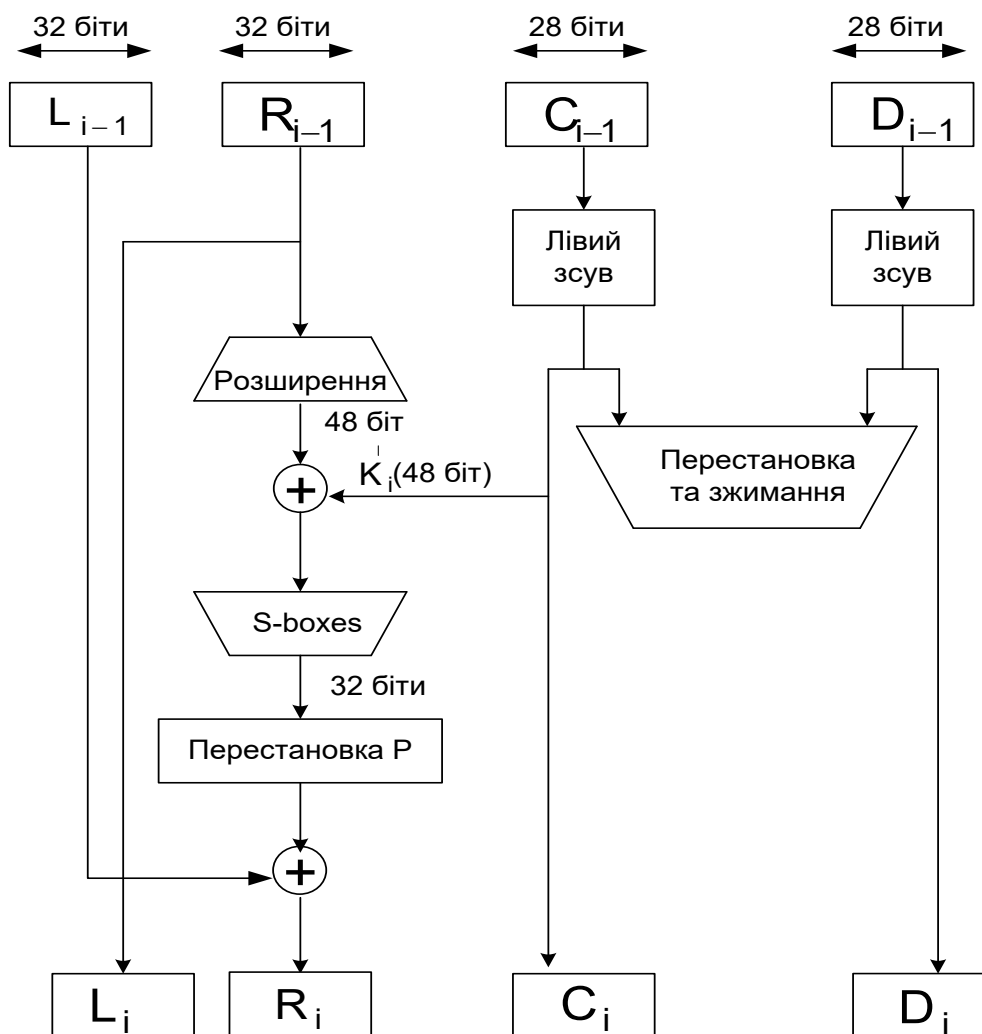


Рис. 2.14. І-ий раунд DES

Спочатку ключ подається на вхід функції перестановки. Потім для кожного з 16 раундів підключ K_i є комбінацією лівого циклічного зсуву і перестановки. Функція перестановки однакова для кожного раунду, але підключі K_i для кожного раунду виходять різні, внаслідок повторюваного зсуву бітів ключа.

Шифрування. Початкова перестановка

Початкова перестановка і її інверсія визначаються стандартною таблицею. Якщо M – це довільні 64 біти, то $X = IP(M)$ – переставлені 64

біти. Якщо застосувати зворотну функцію перестановки $Y = IP^{-1}(X) = IP^{-1}(IP(M))$, то вийде первісна послідовність бітів.

Послідовність перетворень окремого раунду

Тепер розглянемо послідовність перетворень, яка використовується в кожному раунді. 64-бітний вхідний блок проходить через 16 раундів, при цьому на кожній ітерації виходить проміжне 64-бітне значення. Ліва і права частини кожного проміжного значення трактуються як окремі 32-бітні значення, позначені L і R . Кожну ітерацію можна описати в такий спосіб:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i), \text{ де } \oplus \text{ позначає операцію XOR.}$$

Таким чином, вихід лівої половини L_i дорівнює входу правої половини R_{i-1} . Вихід правої половини R_i є результатом застосування операції XOR до L_{i-1} і функції F , що залежить від R_{i-1} і K_i .

Розглянемо функцію F більш докладно.

R_i , яке подається на вхід функції F , має довжину 32 біти. Спочатку R_i розширюється до 48 бітів, використовуючи таблицю, яка визначає перестановку і розширення на 16 бітів. Розширення відбувається в такий спосіб: 32 біти розбиваються на групи по 4 біти і потім розширюються до 6 бітів, приєднуючи крайні біти із двох сусідніх груп. Наприклад, якщо частина вхідного повідомлення

... efgh ijkl mnop ... ,

то в результаті розширення виходить повідомлення

... defghi hijklm lmnoqr ...

Після цього для отриманого 48-бітного значення виконується операція XOR з 48-бітним підключем K_i . Потім отримане 48-бітне значення подається на вхід функції підстановки, результатом якої є 32-бітне значення.

Підстановка складається з восьми S-boxes, кожний з яких на вході одержує 6 бітів, а на виході створює 4 біти. Ці перетворення визначаються спеціальними таблицями. Перший і останній біти вхідного значення S-box визначають номер рядка в таблиці, середні 4 біти визначають номер стовпця. Перетинання рядка і стовпця визначає 4-бітний вихід. Наприклад, якщо входом є 011011, то номер рядка рівний 01 (рядок 1) і номер стовпця рівний 1101 (стовпець 13). Значення в рядку 1 і стовпці 13 рівне 5, тобто виходом є 0101.

Далі отримане 32-бітне значення обробляється за допомогою перестановки P , метою якої є максимальне переупорядкування бітів, щоб у наступному раунді шифрування з великою ймовірністю кожний біт оброблявся іншим S-box.

Створення підключів

Ключ для окремого раунду K_i складається з 48 бітів. Ключі K_i виходять за наступним алгоритмом. Для 56-бітного ключа, який використовується на вході алгоритму, спочатку виконується перестановка відповідно до таблиці Permuted Choice 1 (PC-1). Отриманий 56-бітний ключ розділяється на дві 28-бітні частини, позначувані як C_0 і D_0 відповідно. На кожному раунді C_i і D_i незалежно циклічно зрушуються вліво на 1 або 2 біти, залежно від номера раунду. Отримані значення є входом наступного раунду. Вони також становлять вхід у Permuted Choice 2 (PC-2), який створює 48-бітне вихідне значення, що є входом функції $F(R_{i-1}, K_i)$.

Розшифрування

Процес розшифрування аналогічний процесу шифрування. На вході алгоритму використовується зашифрований текст, але ключі K_i використовуються у зворотній послідовності. K_{16} використовується на першому раунді, K_1 на останньому раунді. Нехай виходом i -го раунду шифрування буде $L_i || r_i$. Тоді відповідний вхід $(16-i)$ -ого раунду розшифрування буде $R_i || l_i$.

Після останнього раунду процесу розшифрування дві половини виходу міняються місцями так, щоб вхід заключної перестановки IP^{-1} був $R_{16} || L_{16}$. Виходом цієї стадії є незашифрований текст.

Перевіримо коректність процесу розшифрування. Використаємо зашифрований текст і ключ й використаємо їх як вхід в алгоритм. На першому кроці виконаємо початкову перестановку IP і одержимо 64-бітне значення $Ld_0 || rd_0$. Відомо, що IP і IP^{-1} протилежні. Отже,

$$Ld_0 || rd_0 = IP \text{ (зашифрований текст).}$$

$$\text{Зашифрований текст} = IP^{-1}(R^{16} || L^{16})$$

$$Ld_0 || rd_0 = IP(IP^{-1}(R_{16} || L_{16})) = R_{16} || L_{16}.$$

Таким чином, вхід першого раунду процесу розшифрування еквівалентний 32-бітному виходу 16-ого раунду процесу шифрування, у якого ліва і права частини записані у зворотному порядку.

Тепер ми повинні показати, що вихід першого раунду процесу розшифрування еквівалентний 32-бітному входу 16-ого раунду процесу шифрування. По-перше, розглянемо процес шифрування.

$$L_{16} = R_{15}$$

$$R_{16} = L_{15} \oplus F(R_{15}, K_{16}).$$

При розшифруванні:

$$Ld_1 = Rd_0 = L_{16} = R_{15}$$

$$Rd_1 = Ld_0 \oplus F(Rd_0, K_{16}) = R_{16} \oplus F(Rd_0, K_{16}) = (L_{15} \oplus F(R_{15}, K_{16})) \oplus F(R_{15}, K_{16}).$$

XOR має наступні властивості:

$$(A \oplus B) \oplus C = A \oplus (B \oplus C)$$

$$D \oplus D = 0$$

$$E \oplus 0 = E.$$

Таким чином, маємо $Ld_1 = R_{15}$ і $Rd_1 = L_{15}$. Отже, вихід першого раунду процесу розшифрування є $L_{15}||R_{15}$, який є перестановкою входу 16-го раунду шифрування. Легко показати, що дана відповідність виконується всі 16 раундів. Можна описати цей процес у загальних термінах. Для i -ого раунду, що шифрує алгоритму:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i).$$

Ці рівності можна записати по-іншому:

$$R_{i-1} = L_i$$

$$L_{i-1} = R_i \oplus F(R_{i-1}, K_i) = R_i \oplus F(L_i, K_i).$$

Таким чином, описані входи i -ого раунду як функція виходів.

Вихід останньої стадії процесу розшифрування є $R_0||L_0$. Щоб входом IP^{-1} стадії було $R_0||L_0$, необхідно поміняти місцями ліву і праву частини.

Але

$$IP^{-1}(R_0||L_0) = IP^{-1}(IP(\text{незашифрований текст})) = \text{незашифрований текст}.$$

Тобто одержуємо незашифрований текст, що і демонструє можливість дешифрування DES.

Проблеми DES

Через те що довжина ключа рівна 56 бітам, існує 2^{56} можливих ключа. На сьогодні така довжина ключа недостатня, оскільки допускає успішне застосування лобових атак. Альтернативою DES можна вважати потрійний DES, IDEA, а також алгоритм Rijndael, прийнятий у якості нового стандарту на алгоритми симетричного шифрування [34; 35; 40; 44; 88; 93].

Також без відповіді поки залишається питання, чи можливий криптоаналіз із використанням існуючих характеристик алгоритму DES. Основою алгоритму є вісім таблиць підстановки, або S-boxes, які застосовуються в кожній ітерації. Існує небезпека, що ці S-boxes конструювалися таким чином, що криптоаналіз можливий для противника,

який знає слабкі місця S-boxes. Протягом багатьох років обговорювалася як стандартна, так і несподівана поведінка S-boxes, але все-таки нікому не вдалося виявити їх фатально слабкі місця.

Алгоритм потрійний DES

На сьогоднішній день основним недоліком DES вважається маленька довжина ключа, тому вже давно почали розроблятися різні альтер-нативи цьому алгоритму шифрування. Один з підходів полягає в тому, щоб розробити новий алгоритм, і успішний тому приклад – IDEA. Інший підхід допускає повторне застосування шифрування за допомогою DES з використанням декількох ключів.

Недоліки подвійного DES. Найпростіший спосіб збільшити довжину ключа полягає в повторному застосуванні DES із двома різними ключами. Використовуючи незашифроване повідомлення P і два ключі K_1 і K_2 , зашифроване повідомлення C можна одержати в такий спосіб:

$$C = E_{K_2} [E_{K_1} [P]].$$

Для розшифрування потрібно, щоб два ключі застосовувалися у зворотному порядку:

$$P = D_{K_1} [D_{K_2} [C]].$$

У цьому випадку довжина ключа рівна $56 * 2 = 112$ бітів.

Атака “зустріч посередині”

Для наведеного вище алгоритму подвійного DES існує так звана атака “зустріч посередині”. Вона заснована на наступній властивості алгоритму. Маємо: $C = E_{K_2} [E_{K_1} [P]]$.

Тоді: $X = E_{K_1} [P] = D_{K_2} [C]$.

Атака полягає в наступному. Потрібно, щоб атакуючий знав хоча б одну пару незашифрованого тексту і відповідний йому зашифрований текст: (P, C) . У цьому випадку, по-перше, шифрується P для всіх можливих 2^{56} значень K_1 . Цей результат запам'ятовується в таблиці, і потім таблиця впорядковується за значенням X . Наступний крок полягає в дешифруванні C із застосуванням усіх можливих 2^{56} значень K_2 . Для кожного виконаного дешифрування шукається рівне йому значення в першій таблиці. Якщо відповідне значення знайдене, то вважається, що ці ключі можуть бути правильними, і вони перевіряються для наступної відомої пари незашифрованого тексту, зашифрованого тексту.

Якщо відома тільки одна пара значень незашифрованого тексту – зашифрованого тексту, то може бути отримана досить велика кількість неправильних значень ключів. Але якщо противник має можливість перехопити хоча б дві пари значень (незашифрований текст –

зашифрований текст), то складність зламу подвійного DES фактично стає рівною складності зламу звичайного DES, тобто 2^{56} .

Потрійний DES із двома ключами

Очевидна протидія атаці “зустріч посередині” полягає у використанні третьої стадії шифрування із трьома різними ключами. Це піднімає вартість лобової атаки до 2^{168} , яка на сьогоднішній день вважається вищою практичних можливостей. Але при цьому довжина ключа рівна $56 * 3 = 168$ бітам, що іноді буває громіздко.

У якості альтернативи пропонується метод потрійного шифрування, що використовує тільки два ключі. У цьому випадку виконується послідовність зашифрування-розшифрування-зашифрування (EDE) (рис. 2.15, рис. 2.16).

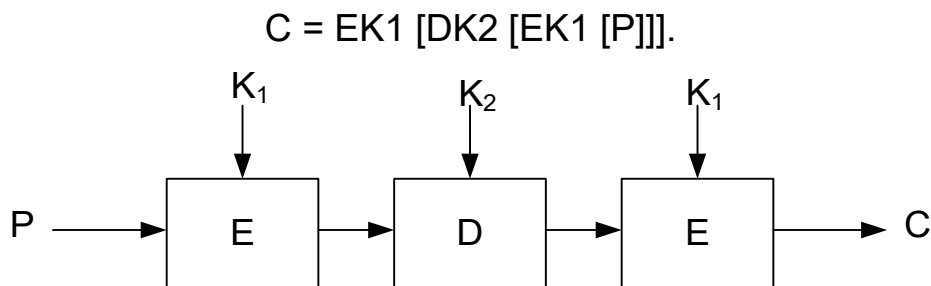


Рис. 2.15. Шифрування потрійним DES

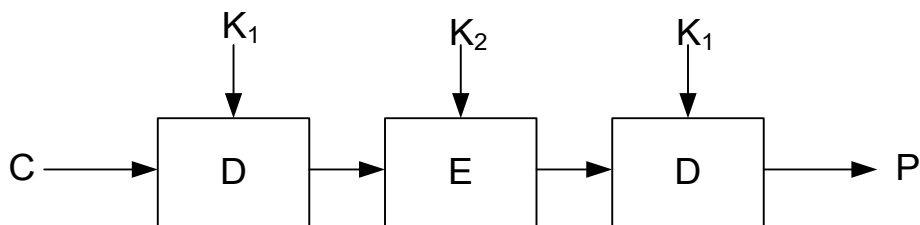


Рис. 2.16. Розшифрування потрійним DES

Немає великого значення, що використовується на другій стадії: шифрування або дешифрування. У випадку використання дешифрування існує тільки та перевага, що можна потрійний DES звести до звичайного одиночного DES, використовуючи $K_1 = K_2$:

$$C = EK_1 [DK_1 [EK_1 [P]]] = EK_1 [P].$$

Потрійний DES є досить популярною альтернативою DES і використовується при керуванні ключами в стандартах ANSI X9.17 і ISO 8732 і в PEM (Privacy Enhanced Mail). Відомих криптографічних атак на потрійний DES не існує. Ціна добору ключа в потрійному DES дорівнює 2^{112} .

2.5.2. Алгоритм блокового симетричного шифрування ГОСТ-28147

Алгоритм ГОСТ 28147 є вітчизняним стандартом для алгоритмів симетричного шифрування. ГОСТ 28147 розроблений в 1989 році і є блоковим алгоритмом шифрування. Довжина блоку дорівнює 64 бітам, довжина ключа дорівнює 256 бітам, кількість раундів – 32. Алгоритм становить класичну мережу Файстеля.

$$L_i = R_{i-1}$$
$$R_i = L_i \oplus f(R_{i-1}, K_i).$$

Функція F проста. Спочатку права половина та i -ий підключ складаються за модулем 2^{32} . Потім результат розбивається на вісім 4-бітових значень, кожне з яких подається на вхід S-box. ГОСТ 28147 використовує вісім різних S-boxes, кожен з яких має 4-бітовий вхід і 4-бітовий вихід. Виходи всіх S-boxes поєднуються в 32-бітне слово, яке потім циклічно зсувається на 11 бітів вліво. Нарешті, за допомогою XOR результат поєднується з лівою половиною, у результаті чого виходить нова права половина (рис. 2.17).

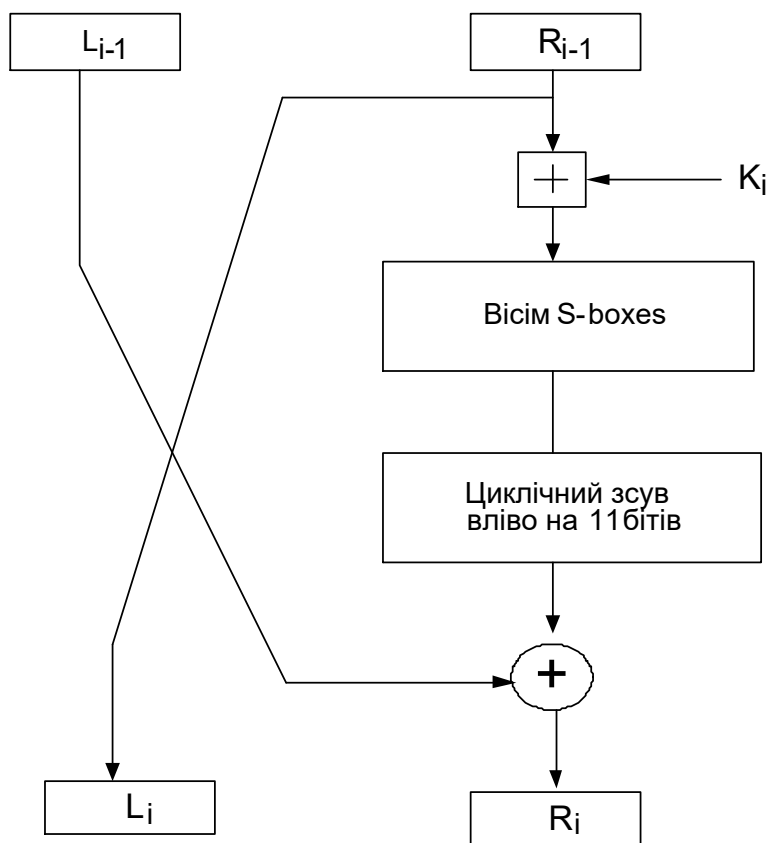


Рис. 2.17. i -й раунд ГОСТ 28147

Генерація ключів проста. 256-бітний ключ розбивається на вісім 32-бітних підключів. Алгоритм має 32 раунди, тому кожний підключ використовується в чотирьох раундах за наступною схемою:

Раунд	1	2	3	4	5	6	7	8
Підключ	1	2	3	4	5	6	7	8
Раунд	9	10	11	12	13	14	15	16
Підключ	1	2	3	4	5	6	7	8
Раунд	17	18	19	20	21	22	23	24
Підключ	1	2	3	4	5	6	7	8
Раунд	25	26	27	28	29	30	31	32
Підключ	8	7	6	5	4	3	2	1

Вважається, що стійкість алгоритму ГОСТ 28147 багато в чому визначається структурою S-boxes. Довгий час структура S-boxes у відкритій пресі не публікувалась.

На сьогодні відомі S-boxes, які використовуються в додатках Центрального Банку РФ і вважаються досить сильними. Входом і виходом S-box є 4-бітні числа, тому кожний S-box може бути поданий у вигляді рядка чисел від 0 до 15, розташованих у деякому порядку. Тоді порядковий номер числа буде вхідним значенням S-box, а саме число – вихідним значенням S-box.

1-ий S-box	4	10	9	2	13	8	0	14
	6	11	1	12	7	15	5	3
2-ий S-box	14	11	4	12	6	13	15	10
	2	3	8	1	0	7	5	9
3-ий S-box	5	8	1	13	10	3	4	2
	14	15	12	7	6	0	9	11
4-ий S-box	7	13	10	1	0	8	9	15
	14	4	6	12	11	2	5	3
5-ий S-box	6	12	7	1	5	15	13	8
	4	10	9	14	0	3	11	2
6-ий S-box	4	11	10	0	7	2	1	13
	3	6	8	5	9	12	15	14
7-ий S-box	13	11	4	1	3	15	5	9
	0	10	14	7	6	8	2	12
8-ий S-box	1	15	13	0	5	7	10	4
	9	2	3	14	6	11	8	12

Основні відмінності між DES і ГОСТ 28147 наступні:

DES використовує більш складну процедуру створення підключів, ніж ГОСТ 28147. В ГОСТ ця процедура дуже проста. В DES застосовується 56-бітний ключ, а в ГОСТ 28147 – 256-бітний.

При виборі сильних S-boxes ГОСТ 28147 вважається дуже стійким. В S-boxes DES 6-бітові входи й 4-бітові виходи, а в S-boxes ГОСТ 4-бітові входи й виходи. В обох алгоритмах використовується по вісім S-boxes, але розмір S-box ГОСТ суттєво менший від розміру S-box DES.

В DES застосовуються нерегулярні перестановки P, в ГОСТ використовується 11-бітний циклічний зсув вліво. Перестановка DES збільшує лавинний ефект. В ГОСТ зміна одного вхідного біта впливає на один S-box одного раунду, який потім впливає на два S-boxes наступного раунду, три S-boxes наступного і т. д.

У ГОСТ 28147 потрібно 8 раундів перш ніж зміна одного вхідного біта вплине на кожний біт результату; DES для цього потрібно тільки 5 раундів. У DES 16 раундів, у ГОСТ – 32 раунди, що робить його більш стійким до диференціального і лінійного криптоаналізу.

2.6. Типові режими роботи криптосистеми

Для будь-якого симетричного блокового алгоритму шифрування визначено чотири режими виконання.

ECB – Electronic Codebook – кожний блок з 64 бітів незашифрованого тексту шифрується незалежно від інших блоків із застосуванням алгоритму симетричного шифрування. Типові додатки – безпечна передача одиночних значень (наприклад, криптографічного ключа).

CBC – Cipher Block Chaining – вхід криптографічного алгоритму є результатом застосування операції XOR до наступного блоку незашифрованого тексту і попереднього блоку зашифрованого тексту. Типові додатки – загальна блок-орієнтована передача, автентифікація.

CFB – Cipher Feedback – при кожній ітерації алгоритму обробляється J бітів вхідного значення. Попередній зашифрований блок використовується як вхід в алгоритм; до J бітів виходу алгоритму і наступного незашифрованого блоку з J бітів застосовується операція XOR, результатом якої є наступний зашифрований блок з J бітів. Типові додатки – потокоорієнтована передача, автентифікація.

OFB – Output Feedback – аналогічний CFB, за винятком того, що на вхід алгоритму при шифруванні наступного блоку подається результат шифрування попереднього блоку; тільки після цього виконується операція XOR із черговими J бітами незашифрованого тексту. Типові додатки – потокоорієнтована передача за зашумленим каналом (наприклад, супутниковий зв'язок).

2.6.1. Шифрування в режимі *Electronic Code Book, ECB*

Даний режим є найпростішим режимом, при якому незашифрований текст обробляється послідовно, блок за блоком. Кожний блок шифрується, використовуючи той самий ключ. Якщо повідомлення довше, ніж довжина блоку відповідного алгоритму, то воно розбивається на блоки відповідної довжини, причому останній блок доповнюється, якщо буде потреба, фіксованими значеннями. При використанні даного режиму однакові незашифровані блоки будуть перетворені в однакові зашифровані блоки.

ECB-режим ідеальний для невеликої кількості даних, наприклад, для шифрування ключа сесії.

Далі будемо використовувати такі позначення. Символ E будемо використовувати для позначення операції шифрування n -бітного блокового шифру, де n – кількість бітів у блоках відкритого й закритого текстів, а символом D будемо позначати операцію дешифрування для того ж шифру.

Перетворення відкритого тексту P у закритий текст C здійснюється за формулою:

$$C = E_k(P),$$

де C – n -бітний блок шифротексту;

K – секретний ключ для блокового шифру;

P – n -бітний блок відкритого тексту.

Аналогічним чином маємо зворотне перетворення:

$$P = D_k(C),$$

також справедливе співвідношення виду:

$$P = D_k(E_k(P)).$$

Режим електронної кодової книги. Схема режиму подана на рис. 2.18.

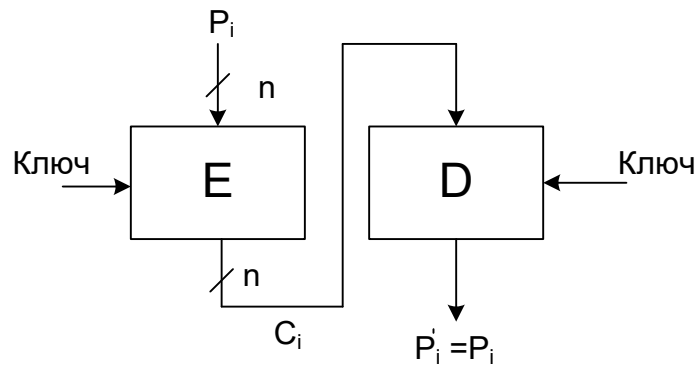


Рис. 2.18. Режим електронної кодової книги

На вхід алгоритму надходить k -бітний ключ K і n -бітні блоки відкритого тексту $P = p_1, p_2, \dots, p_t \dots$. На виході формуються n -бітні блоки шифротексту $C = c_1, c_2, \dots, c_t \dots$.

Шифрування: для $1 \leq i \leq t$:

$$c_i = E_k(p_i).$$

Дешифрування: для $1 \leq i \leq t$:

$$p_i = D_k(c_i).$$

Зазначимо наступні властивості *ECB* режиму:

1. Шифрування ідентичних відкритих текстів. Блокові шифри в такому режимі не забезпечують приховання шаблонів даних – ідентичні блоки відкритого тексту (при тому самому ключі) перетворюються в ідентичний шифротекст.

2. Вплив зчеплення. Блоки шифруються незалежно один від одного. Переупорядкування блоків шифротексту призведе до переупорядкування блоків відкритого тексту. Оскільки блоки шифротексту незалежні, навмисна підстановка *ECB*-блоків (наприклад, вставка блоку, що часто зустрічається) не має впливу на дешифрування суміжних блоків.

3. Розмноження помилок. Один і більше помилкових бітів в одному блоці шифротексту впливають на дешифрування тільки цього блоку. Результат дешифрування блоку з помилками є випадковою величиною, близько 50 % відновлювальних бітів відкритого тексту.

4. Істотним недоліком *ECB* є те, що той самий блок незашифрованого тексту, що з'являється більше одного разу в повідомленні, завжди має той самий зашифрований вигляд. Внаслідок цього для більших повідомлень

ECB-режим вважається небезпечним. Якщо повідомлення має багато однакових блоків, то при криптоаналізі дана закономірність буде виявлена.

Із перерахованих особливостей випливає і використання *ECB*-режиму. *ECB*-режим не рекомендується використовувати для шифрування повідомлення більше, ніж один блок, або якщо повторно використовуються ключі для шифрування більше, ніж одного одноблокового повідомлення. Таємність ключа може бути забезпечена шляхом включення випадкових додаткових бітів у кожний блок.

2.6.2. Шифрування в режимі Cipher Block Changing, CBC

Для подолання недоліків ECB використовують спосіб, при якому однакові незашифровані блоки перетворюються в різні зашифровані. Для цього на вхід алгоритму подається результат застосування операції XOR до поточного незашифрованого блоку і попереднього зашифрованого блоку.

Режим зчеплення блоків шифру включає використання n -розрядного вектора ініціалізації, позначуваного IV. У схему додається регістр або буфер зворотного зв'язку, у який спочатку записується значення IV, а потім – наступні значення блоків шифротексту. На рис. 2.19 показана схема CBC.

Для одержання першого блоку зашифрованого повідомлення використовується вектор ініціалізації (IV), для якого виконується операція XOR з першим блоком незашифрованого повідомлення. При розшифруванні для IV виконується операція XOR з виходом алгоритму розшифрування для одержання першого блоку незашифрованого тексту.

IV повинен бути відомий як відправнику, так і одержувачу. Для максимальної безпеки IV повинен бути захищений так само, як і ключ.

На вхід алгоритму надходить k -бітний ключ K , n -бітний IV, послідовність n -бітних блоків відкритого тексту $P = p_1, p_2, \dots, p_i, \dots, p_t \dots$

Послідовність блоків шифротексту $C = c_1, c_2, \dots, c_t$ обчислюється за правилом:

$$c_i = E_k(p_i \oplus c_{i-1}); \quad i = 1, \dots, t; c_0 = IV.$$

Розшифрування здійснюється як: $p_i = c_{i-1} \oplus D_k(c_i); \quad i = 1, \dots, t; c_0 = IV.$

Властивості режиму:

1. Шифрування ідентичних відкритих текстів. Ідентичні блоки шифротексту формуються тоді, коли шифрується той самий відкритий

текст на тих самих ключах і векторі ініціалізації. Зміна вектора ініціалізації, ключа або першого блоку відкритого тексту (наприклад, використовуючи лічильник або випадкову величину) призводить до різних шифротекстів.

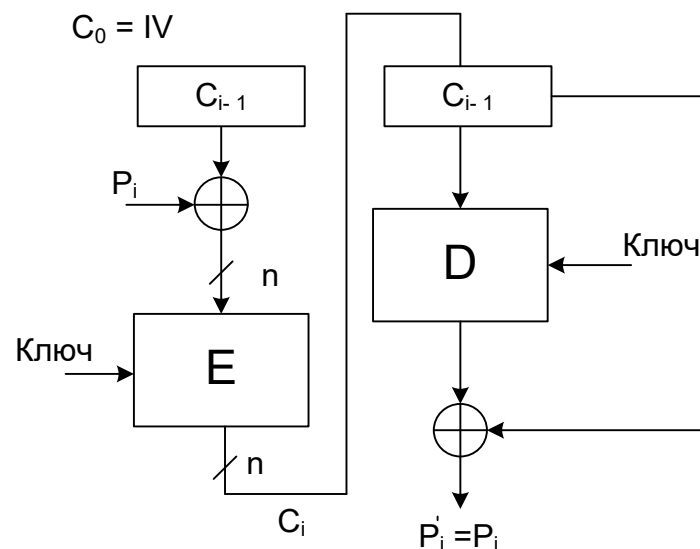


Рис. 2.19. Режим зчеплення блоків шифру

2. Вплив зчеплення. Механізм зчеплення приводить до того, що шифротекст c_i залежить від p_i і всіх попередніх блоків відкритого тексту. У цьому випадку вхідна залежність попередніх блоків утримується в значенні попереднього блоку шифротексту. Отже, перевпорядкування порядку проходження блоків шифротексту впливає на розшифрування. Для того щоб правильно розшифрувати блок шифротексту, необхідно мати правильний попередній блок шифротексту.

3. Розмноження помилок. Одиначна бітова помилка в блоці шифротексту c_i впливає на дешифрування як мінімум двох блоків c_i і c_{i+1} (оскільки p_i залежить від c_i і c_{i+1}). Блок p'_i відновлений із c_i звичайно є повністю випадковим (50 % бітів помилкові), у той час як відновлений відкритий текст p'_{i+1} матиме неправильних бітів рівно стільки, скільки їх має блок c_i . Таким чином, противник може викликати передбачувані зміни бітів у p'_{i+1} , змінюючи відповідні біти блоку c_i .

4. Відновлення помилок. CBC-режим є реалізацією самосинхронізуючого шифру або шифрування з автоключем за шифротекстом в тому розумінні, що якщо має місце помилка в блоці c_i (включаючи втрату одного або більше вхідних блоків), але немає помилок в c_{i+1} , c_{i+2} , то можливе коректне дешифрування p_{i+2} .

З погляду практичного використання блокових шифрів у даному режимі необхідно також урахувати наступні рекомендації:

1. Розмноження помилок при шифруванні. Хоча розшифрування в CBC-режимі дозволяє відновити відкритий текст із помилок у блоках шифротексту, модифікація блоку відкритого тексту p_i під час шифрування змінює всі наступні блоки шифротексту. Це ускладнює (якщо не виключає) застосовуваність режимів зчеплення для додатків, що вимагають випадкового доступу (читання/запису) до шифрованих даних. Альтернативою є режим ECB.

2. Самосинхронізація і групування помилок. Незважаючи на те, що самосинхронізація можлива при наявності помилок у бітах, самосинхронізація при “загублених” бітах, що є причиною помилки визначення меж блоку (помилка цілісності кадру), неможлива ні в CBC-режимі, ні в інших режимах.

3. Цілісність IV в CBC. Незважаючи на те, що вектор ініціалізації IV в CBC-режимі не є секретним, повинна бути забезпечена цілісність, оскільки навмисна модифікація IV дозволяє противнику виконувати передбачувані зміни бітів у відновлюваному першому блоці відкритого тексту. Використання секретних IV є одним з методів запобігання таких модифікацій.

2.6.3. Шифрування в режимі *Electronic Feedback, CFB*

Блоковий алгоритм призначений для шифрування блоків встановленої довжини. Однак можна перетворити блоковий алгоритм у потоковий алгоритм шифрування, використовуючи два останні режими. Потоковий алгоритм шифрування усуває необхідність розбивати повідомлення на цілу кількість блоків досить великої довжини, отже, він може працювати в реальному часі. Таким чином, якщо передається потік символів, кожний символ може шифруватися і передаватися відразу, з використанням символно-орієнтованого режиму блокового алгоритму шифрування. У CFB-режимі на вхід подається k -бітний ключ K , n -бітний IV, послідовність r -бітних блоків відкритого тексту $P = p_1, p_2, \dots, p_u$, де $1 \leq r \leq n$. На виході формуються r -бітні блоки шифротексту $C = c_1, c_2, \dots, c_u \dots$

Шифрування здійснюється таким способом. Спочатку здійснюється запис у буфер зворотного зв'язку, записується значення IV— $I_1=IV$. Потім для всіх $1 \leq i \leq u$ виконуються наступні операції:

1. Обчислення функції шифрування

$$O_i = E_k(I_i).$$

2. Вибірка r крайніх ліворуч бітів результату шифрування O_i

$$t_i = O_i \sim r,$$

де $O_i \sim r$ означає операцію вибірки.

3. Формування і передача r -бітного блоку шифротексту c_i

$$c_i = p_i \oplus t_i.$$

4. Формування змінного зворотного зв'язку із блоку шифротексту, відповідно до виразу

$$l_{i+1} = 2^r l_i \oplus c_i \bmod 2^n.$$

Ця операція еквівалентна зсуву значення буферного регістра l_i на r -бітів вправо і запису в кінець регістра r -бітного блоку c_i . Розшифрування виконується відповідно:

$$P_i = c_i \oplus t_i; \quad i=1, \dots, u,$$

де t_i , O_i і l_i обчислюються відповідно до виразів, наведених вище і $l_1 = IV$.

На рис. 2.20 подана схема CFB-режиму.

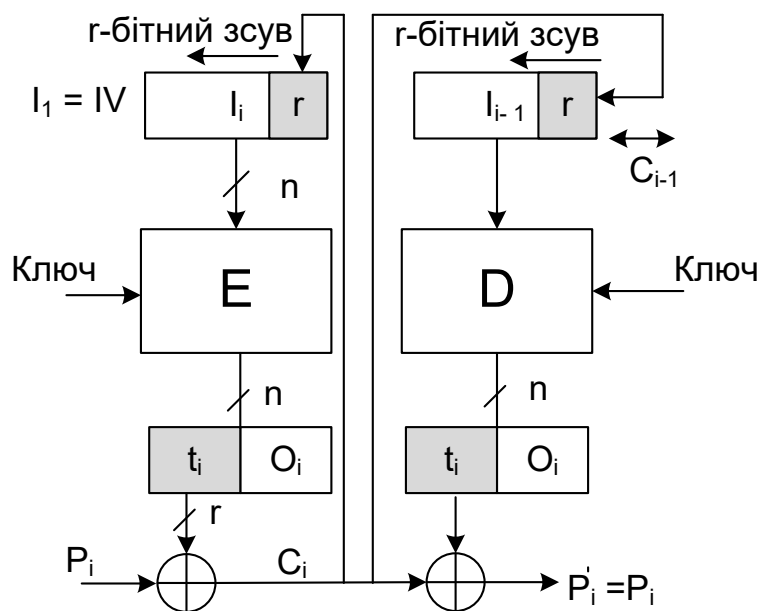


Рис. 2.20. Режим зворотного зв'язку за шифротекстом

Відзначимо, що фактично необхідно мати ще й буфер зворотного зв'язку для зберігання c_{i-1} , що звичайно вибирається рівним n .

CFB-режим має наступні властивості:

1. Шифрування ідентичних відкритих текстів. Як і в CBC-режимі шифрування, зміна IV приводить до різних результатів шифрування того самого відкритого тексту. Значення IV не обов'язково повинне бути секретним, хоча непередбачуваність IV є бажаною в деяких додатках.

2. Вплив зчеплення. Аналогічно з CBC-режимом шифрування механізм зчеплення приводить до того, що блок шифротексту c_i залежить від двох блоків відкритого тексту p_i і p_{i-1} . Отже, перепорядкування блоків шифротексту впливає на дешифрування. Правильне дешифрування правильних блоків шифротексту вимагає одержання $\lceil n/k \rceil$ правильних попередніх блоків шифротексту для того, щоб регістр зворотного зв'язку містив правильне значення.

3. Розмноження помилок. Одна або більше помилок у будь-якому одиничному r -бітному блоці шифротексту c_i впливає на дешифрування наступних $\lceil n/k \rceil$ блоків шифротексту, тобто поки не буде закінчена обробка n бітів шифротексту, після яких помилковий блок c_i буде висунутий з регістра зворотного зв'язку. Відновлений відкритий текст p'_i буде відрізнятися від p_i точно в тих же позиціях, у яких відбулися помилки в c_i . Інші некоректно відновлені блоки відкритого тексту звичайно будуть випадковими векторами, тобто матимуть 50 % помилкових бітів. У такий спосіб противник може здійснити передбачувані зміни бітів у p_i шляхом зміни відповідних бітів у c_i .

4. Відновлення помилок. CFB-режим, як і CBC-режим, що самосинхронізується, але вимагає одержання $\lceil n/k \rceil$ блоків шифротексту для відновлення синхронізації.

5. Продуктивність. Для $r < n$ продуктивність зменшується в n/r разів (порівняно з CBC), тому що кожне шифрування E забезпечує формування тільки r бітів шифротексту на виході.

Розглянутий режим є режимом роботи з r -бітними вхідними символами і r -бітним зворотним зв'язком. Цей режим схожий з режимом CFB, стандартизованим для DES (NBS FIPS Pub 81 і ANSI X3.106) і позначається як CFB r -бітний символ/ r -бітний зворотний зв'язок. Стандарт ISO/IEC 10118:1991 визначає більш загальний CFB-режим. При використанні цього режиму може здійснюватися обробка j -бітних блоків відкритого тексту при r -бітному зворотному зв'язку, причому $j \leq r$. Таким чином, для реалізації режиму необхідно вибрати величину зворотного зв'язку r ($1 \leq r \leq n$) і величину блоку відкритого тексту j ($1 \leq j \leq r$).

На вхід надходить k -бітний ключ K , n -бітний IV , послідовність j -бітних блоків відкритого тексту $P = p_1, p_2, \dots, p_u$. На виході алгоритму формуються j -розрядні блоки шифротексту $C = c_1, c_2, \dots, c_u$.

Шифрування здійснюється відповідно до наступних операцій.

Перед шифруванням здійснюється запис у регістр зворотного зв'язку значення вектора ініціалізації $I_1 = IV$.

Для всіх $1 \leq i \leq u$ виконуються наступні операції:

1. Обчислення функції шифрування:

$$O_i = E_k(I_i).$$

2. Вибірка j лівих бітів величини O_i :

$$t_i = O_i \sim j.$$

3. Формування блоку шифротексту:

$$c_i = p_i + t_i.$$

4. Формування змінного зворотного зв'язку із блоку шифротексту c_i шляхом його доповнення ліворуч $r-j$ одиницями:

$$F_i = F^{(1)}(r-j) || c_i,$$

де $F^{(1)}(r-j)$ – блок з $r-j$ одиниць;

$||$ – операція конкатенції.

5. Зсув регістра I_i на r -бітів вліво і запис F_i у регістр:

$$I_{i+1} = 2^r I_i + F_i \bmod 2^n.$$

Зазначимо, що для $i = u$ операції 4 і 5 не виконуються.

Розшифрування виконується в аналогічному порядку, за винятком того, що:

$$P_i = c_i \oplus t_i.$$

На рис. 2.21 подано схематичне зображення функціонування алгоритму шифрування в CFB-режимі за ISO/IEC 10118: 1991.

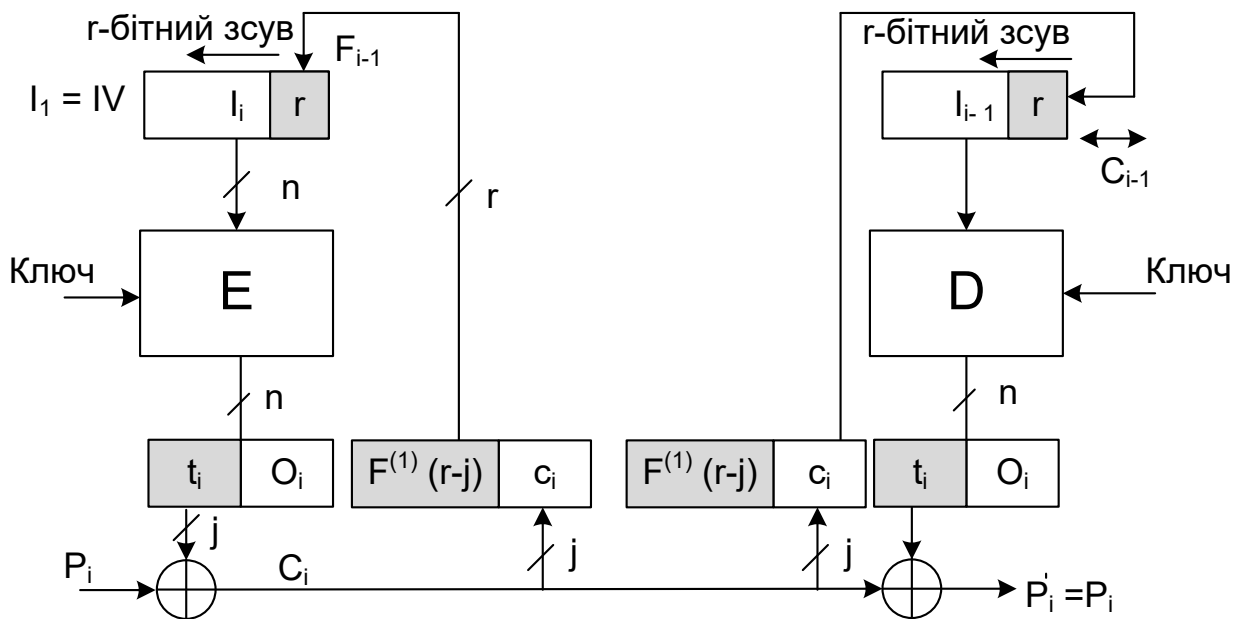


Рис. 2.21. Режим CFB – j -бітний відкритий текст/ r -бітний зворотний зв'язок (j/r CFB-режим)

І, нарешті, у другій редакції стандарту ISO/IEC 10116: 1997 була запропонована узагальнена версія CFB-режиму. Стандартизований режим забезпечує “конвеєрну обробку даних”.

У раніше розглянутих версіях CFB-режиму результат шифрування одного блоку відкритого тексту є входом для шифрування наступного блоку. Це означає, що неможливі “конвеєрні” обчислення, тобто неможливо приступити до шифрування одного блоку до закінчення обробки попереднього блоку. Щоб уникнути цього недоліку, у нову версію CFB-режиму уведений уже m -розрядний буфер зворотного зв'язку, де $2^n \geq m \geq n$. Також для роботи необхідний m -розрядний IV.

На вхід алгоритму надходять: k -розрядний ключ K , m -розрядний IV, послідовність j -розрядних блоків відкритого тексту $P = p_1, p_2, \dots, p_u$. На виході алгоритму формується послідовність j -розрядних блоків закритого тексту $C = c_1, c_2, \dots, c_u$. Спочатку здійснюється запис у буфер зворотного зв'язку (регістр зсуву L_i). Значення вектора ініціалізації $I_1 = IV$.

Далі для всіх $i = \overline{1, u}$ виконуються наступні операції:

1. Вибірка n кратних ліворуч біт з регістра зсуву L :

$$X_i = L_i \sim n.$$

2. Обчислення функції шифрування E :

$$O_i = E_k(X_i).$$

3. Вибірка j кратних ліворуч бітів з величини O_i :

$$t_i = O_i \sim j.$$

4. Формування і передача блоку закритого тексту:

$$C_i = p_i \oplus t_i.$$

5. Формування змінного зворотного зв'язку із блоку шифротексту c_i шляхом його доповнення ліворуч $r - j$ одиницями:

$$F_i = F^{(1)}(r - j) \parallel c_i.$$

6. Зрушення вмісту буфера зворотного зв'язку I на r розрядів вліво і запис F_i у регістр:

$$I_{i+1} = 2^r I_i + F_i \bmod 2^n.$$

Дешифрування виконується аналогічним чином, за винятком операції 4, а саме:

$$P_i = c_i \oplus t_i.$$

На рис. 2.22 схематично поданий CFB-режим за ISO/IEC 10116. Зазначимо також, що в другій редакції ISO/IEC 10116 рекомендується обирати $j = r$.

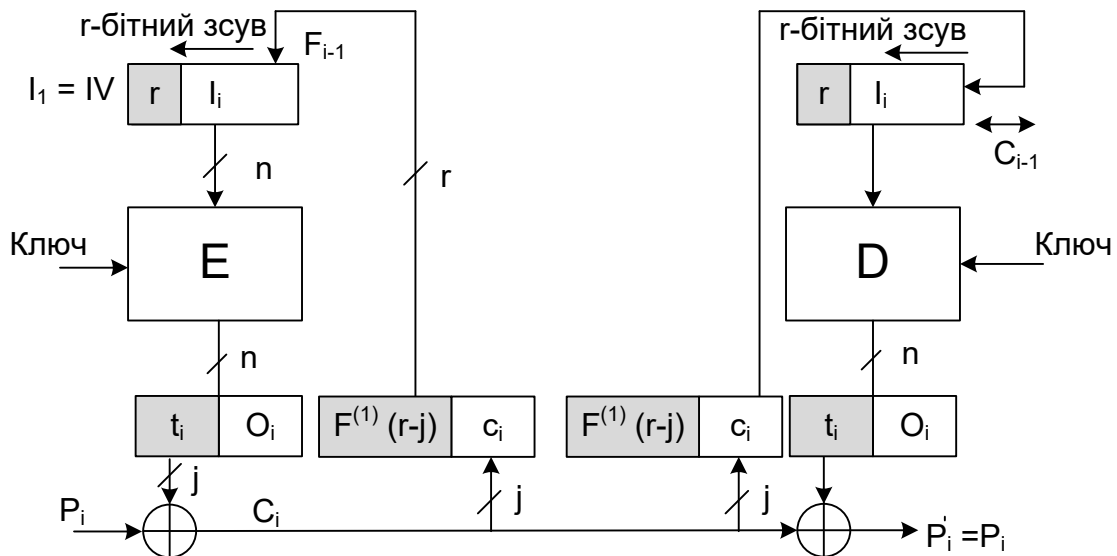


Рис. 2.22. Режим CFB за ISO/IEC 10116 з конвеєрною обробкою даних

3.6.4. Шифрування в режимі *Output Feedback, OFB*

Даний режим подібний режиму CFB. Різниця полягає в тому, що вихід алгоритму в режимі OFB подається назад у регістр зсуву, тоді як у режимі CFB у регістр зсуву подається результат застосування операції XOR до незашифрованого блоку і результату алгоритму.

Основна перевага режиму OFB полягає в тому, що якщо при передачі відбулась помилка, то вона не поширюється на наступні зашифровані блоки, і тим самим зберігається можливість дешифрування наступних блоків. Наприклад, якщо з'являється помилковий біт у C_i , то це призведе тільки до неможливості дешифрування цього блоку і одержання P_i . Подальша послідовність блоків буде розшифрована коректно. При використанні режиму CFB C_i подається в якості входу в регістр і є причиною наступного викривлення потоку.

Недолік OFB у тому, що він більш уразливий до атак модифікації потоку повідомлень, ніж CFB.

Режим OFB може бути використаний для додатків, у яких повинне бути виключене будь-яке розмноження помилок. Режим схожий з режимом CFB і дозволяє шифрувати блоки різної довжини, але як зворотний зв'язок використовується не блок шифротексту, а шифрований блок з виходу функції E .

Поширено дві версії *OFB*-режиму роботи n -розрядного блокового шифру. Версія ISO/IEC 10116: 1997 (ISO/IEC 10118: 1991) вимагає n -розрядного зворотного зв'язку (повний зворотній зв'язок) і є більш стійкою. Раніше була прийнята версія NBS FIPS 81, що дозволяє $r \leq n$ розрядний зворотний зв'язок.

У цій версії *OFB*-режиму на вхід алгоритму надходять: k -бітний ключ K , n -розрядний IV , послідовність r -розрядних блоків відкритого тексту $P = p_1, p_2, \dots, p_u, 1 \leq r \leq n \dots$. На виході формується послідовність r -розрядних блоків шифротексту. При дешифруванні відновлюється відкритий текст.

Шифрування здійснюється в такий спосіб.

Спочатку в регістр I_i записується значення IV .

Потім для $1 \leq i \leq u$ виконуються:

1. Обчислення шифрованого блоку:

$$O_i = E_k(I_i).$$

2. Вибірка r крайніх ліворуч бітів величини O_i . Допускають, що крайній біт визначений як 1.

$$t_i = O_i \sim r.$$

3. Передача r -розрядного блоку шифротексту c_i :

$$c_i = p_i \oplus t_i.$$

4. Відновлення змісту регістра зсуву l_i :

$$l_{i+1} = O_i.$$

На рис. 2.23 подана схема OFB-режиму за ISO/IEC 10116: 1997.

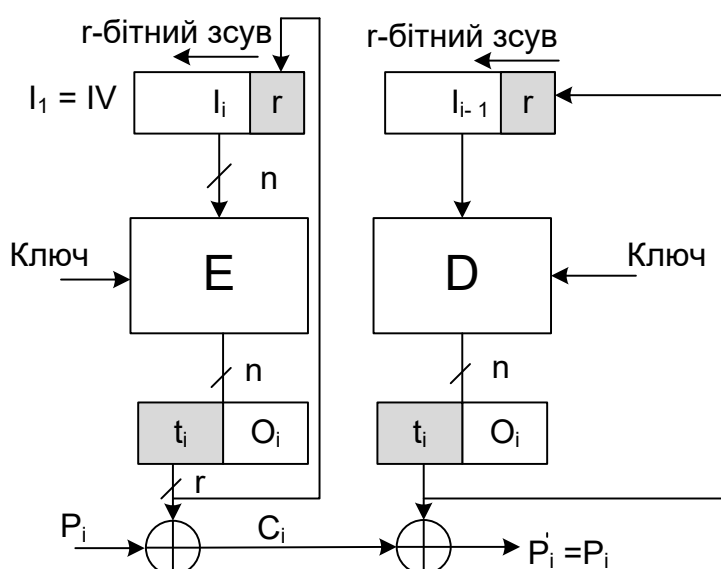


Рис. 2.23. Режим OFB r -бітний відкритий текст/ n -бітний зворотний зв'язок (r/n – OFB-режим)

При розшифруванні встановлюють $l_1 = IV$ і для всіх $1 \leq i \leq u$ після одержання блоку c_i здійснюють ті ж дії за винятком операції 3, що має вигляд:

$$p_i = c_i \oplus t_i.$$

При реалізації режиму OFB за NBS FIPS B1 на вхід алгоритму надходять: k -бітний ключ K , n -розрядний IV , послідовність r -розрядних блоків відкритого тексту $P = p_1, p_2, \dots, p_u$, де $1 \leq r \leq n$. На виході формується послідовність r -розрядних блоків шифротексту $C = c_1, c_2, \dots, c_u$.

Алгоритм шифрування і розшифрування аналогічний розглянутим вище, за винятком того, що операція відновлення регістра $I_{i+1} = O_i$ замінюється на операцію зсуву змісту регістра і запису в нього значення t_i :

$$I_{i+1} = 2^r I_i + t_i \bmod 2^n.$$

Розглянемо властивості OFB-режиму:

1. Шифрування ідентичних відкритих текстів. Як і в CBC і в CFB-режимах шифрування, зміна IV приводить до різних результатів шифрування того самого відкритого тексту.

2. Вплив зчеплення. Ключовий потік є незалежним від відкритого тексту.

3. Розмноження помилок. Одна або більше бітових помилок у будь-якому символі шифротексту c_i , впливає на дешифрування тільки цього символу, і точно в тих же бітових позиціях, у яких виявилися помилки в c_i . У такий спосіб забезпечується повне відновлення бітів відкритого тексту.

4. Відновлення синхронізації. OFB-режим відновлюється після помилок у бітах шифротексту, але не може самосинхронізуватися після втрати бітів шифротексту, які руйнують вирівнювання (синхронізацію) ключового потоку, що розшифровує (у цих випадках необхідна пересинхронізація шифру).

5. Продуктивність. Для $r < n$ продуктивність зменшується в n/r разів (як і в CFB-режимі). Однак у всіх випадках, оскільки ключовий потік не залежить від відкритого й закритого тексту, він може бути обчислений заздалегідь за заданим ключем й IV.

6. Зміна IV в OFB. Вектор ініціалізації, що не є секретним, повинен змінюватися якщо в OFB-режимі повторно використовується ключ K. У протилежному випадку буде сформований ідентичний ключовий потік, і шляхом додавання за модулем 2 (XOR) відповідних шифртекстів противник може прийти до криптоаналізу, у якому шифр буде з нескінченним ключем, де як ключ використовується відкритий текст.

У такий спосіб, виходячи із властивостей режиму, на практиці OFB-режим використовується тільки для забезпечення конфіденційності. У спрощеному OFB-режимі виводять відновлення вхідного блоку як функцію лічильника, тобто: $I_{i+1} = I_i$.

Це дозволяє уникнути проблеми так званого короткого циклу, і забезпечити відновлюваність після помилок в обчисленні E. Більше того, це забезпечує властивість випадкового доступу: не обов'язково

дешифрувати i -ий блок шифротексту для того, щоб дешифрувати $i + 1$ -ий блок.

Раніше версія ISO-режиму була більш стійка, ніж версія NBS FIPS. В OFB-режимі з повним n -розрядним зворотним зв'язком ключовий потік генерується з використанням ітеративної функції $O_i = E_k(O_i - 1)$. Оскільки E_k є перестановкою й при припущенні, що k є випадковою величиною, E_k дійсно є випадковим вибором з множини $(2^n)!$ перестановок за n елементами. Може бути показано, що для фіксованих (випадкових) значень ключа й вектора ініціалізації очікувана довжина циклу перед повторенням будь-якого значення O_i дорівнюватиме $2^n - 1$. З іншого боку, якщо кількість бітів зворотного зв'язку дорівнює $r < n$, як визначено в FIPS 81, то ключовий потік формуватиметься з використанням ітерації $O_i = f(O_i - 1)$, де f не є функцією перестановки. Допускаючи її поводження як випадкової функції, очікувана довжина циклу складе величину близько $2^{n/2}$. Отже, кращим є використання OFB-режиму з повним n -розрядним зворотним зв'язком.

Нарешті необхідно відзначити, що і OFB-режим з повним зворотним зв'язком і режим лічильника забезпечують можливість застосування блокового шифру як генератора ключового потоку для потокового шифру. Аналогічним чином і в CFB-режимі здійснюється шифрування потоку символів, при цьому блоковий шифр використовується як генератор ключового потоку (залежного від відкритого тексту).

Режим CBC також можна розглядати як потоковий шифр із n -розрядними блоками, що відіграють роль дуже великих символів. У такий спосіб режими роботи дозволяють побудувати потокові шифри на основі блокових шифрів.

2.6.5. Шифрування в режимах удосконаленого OFB і PCBC

Відомі недоліки привели до появи вдосконаленого варіанта шифрування в режимі OFB. Основні зміни стосуються методу генерації незалежної послідовності блоків: для одержання чергового блоку пропонується шифрувати не s_i , а $s_i + IV(\text{mod } 264)$, де IV – деякий вектор ініціалізації.

Режим шифрування PCBC (Propagating Cipher Block Chaining) застосовується в протоколі Kerberos (версія 4) і дозволяє виявляти помилки. Даний режим шифрування не є федеральним або міжнародним

стандартом. Режим PCBC – варіант режиму CBC, що володіє специфічною властивістю, – у результаті розшифрування одинична помилка поширюється на весь шифротекст (вирішується зворотне завдання з погляду режиму OFB). Дана властивість дозволяє з високою надійністю виявляти помилки, що виникають при передачі повідомлень каналами з шумом. Шифрування в режимі PCBC виконується за правилом:

$$c_i = E_k(p_i \oplus p_{i-1} \oplus c_{i-1}),$$

розшифрування:

$$p_i = D_k(c_i) \oplus c_{i-1} \oplus p_{i-1},$$

де $p_0 \oplus c$ – вектор ініціалізації.

2.6.6. Інші режими шифрування

Інтуїтивні міркування приводять до висновку, що подвійне шифрування фіксованого відкритого тексту тим самим блоковим шифром (на одному або двох різних ключах) дозволяє підвищити криптостійкість. Однак це не завжди так. Розглянемо деякі режими багаторазового шифрування. Основна ідея полягає в багаторазовому (два, три та більше разів) застосуванні алгоритму шифрування до фіксованого блоку відкритого тексту на декількох секретних ключах. Розшифрування блоку шифротексту виконується у зворотному порядку. Для початку розглянемо режим подвійного шифрування. Позначимо через C , P , K шифротекст, відкритий текст і секретний ключ відповідно. Для процедури шифрування і розшифрування на ключі K введемо позначення E_k і D_k . Тоді подвійне шифрування/розшифрування на двох різних ключах виконується за правилом:

$$\begin{aligned} C &= E_{K_2}(E_{K_1}(P)), \\ P &= D_{K_2}(D_{K_1}(C)). \end{aligned}$$

У випадку якщо криптографічне перетворення має властивість $C = E_{K_2}(E_{K_1}(P)) = E_{K_3}(P)$, подвійне шифрування на двох різних ключах не має ніяких переваг порівняно з однократним. Доказ даного факту (групових властивостей блокового шифру) вимагає теоретичних досліджень. Трудоемність силової атаки – повного перебору ключів при наявності пари C і P та відбору варіантів за результатами шифрування-

розшифрування – зросте в 2^n раз і складе 2^{2n} спроб (де n – довжина ключа в бітах). Дійсно, спочатку фіксується перший претендент на ключ K_1 і виконується шифрування P , потім результат шифрується на будь-яких ключах – претендентах на K_2 доти, доки не буде отриманий шифротекст C . Існує 2^n претендентів на ключ K_1 і стільки ж на K_2 . Отже, обсяг перебору при пошуку ключів складе в найгіршому разі $2^n \times 2^n = 2^{2n}$ спроб шифрування. Меркль (R.C. Merkle) і Хеллман (M.E. Hellman) подали свій варіант силової атаки, що дозволяє розкривати ключі за 2^{n+1} спроб замість 2^{2n} . Ідея атаки полягає в обміні часу (кількості спроб) на пам'ять. Передбачається, що криптоаналітику противника відомі дві пари “відкритий текст/шифротекст” – P_1, C_1 і P_2, C_2 такі, що:

$$C_1 = E_{K_2}(E_{K_1}(P_1)) \text{ і } C_2 = E_{K_2}(E_{K_1}(P_2)).$$

Спочатку виконується шифрування блоку відкритого тексту P_1 на всіляких ключах-претендентах на K_1 . Отримані в результаті блоки шифротексту зберігаються в пам'яті. Їх зберігання буде потрібно $2^n b$ комірок пам'яті, де b – довжина блоку в бітах. Далі для кожного претендента на ключ K_2 виконується розшифрування блоку шифротексту C_1 . Потім виконується пошук результату розшифрування C_1 серед збережених раніше в пам'яті блоку шифротексту. Якщо результат виявлений, то, можливо, наступний претендент K_2 на ключ, і той, що відповідає шифротексту в пам'яті, – дійсні ключі K_1 і K_2 . Для перевірки претендентів виконується шифрування P_2 . Якщо отриманий у результаті шифротекст ідентичний C_2 – ключі розкриті, якщо ні – пошук ключів триває. Для пошуку результату в пам'яті застосовується ґеш-функція (не плутати із криптографічними додатками ґеш-функцій), що відображає аргумент на деяку адресу пам'яті. Кількість операцій при реалізації ґеш-функції не залежить від обсягу пам'яті і тому не враховується при оцінці трудоемності атаки. Максимальна кількість спроб, яку, можливо, доведеться провести, дорівнює 2^{2n} або 2^{n+1} . Для двох пар імовірність успіху оцінюється як $2^{-(2n - 2b)}$. Природно, чим більше пара використовується для перевірки ключів, тим вища ймовірність успіху. Таким чином, імовірність успіху для m пар – $2^{-(2m - 2b)}$. Описаний варіант атаки вимагає великого обсягу пам'яті. Так, при 128-бітному ключі для зберігання проміжних результатів буде потрібно 10^{39} байтів пам'яті. У своїй книзі “Прикладна криптографія” Шнайер (B. Schneier) приводить наступне порівняння: “Якщо припустити, що існує

спосіб зберігання біта інформації, використовуючи один-єдиний атом алюмінію, обладнання пам'яті, необхідне для виконання атаки такого роду, буде становити алюмінієвий куб з ребром в 1 км^2 [35].

У 1979 р. Тачмен (W. Tuchman) запропонував метод потрійного шифрування на двох різних ключах:

$$C = E_{K_3}(D_{K_2}(E_{K_1}(P))), \quad P = D_{K_3}(E_{K_2}(D_{K_1}(C))).$$

Такий режим називають “шифрування – розшифрування – шифрування” (encrypt-decrypt-encrypt, EDE). При n -бітних ключах дана схема шифрування еквівалентна схемі з 2 n -бітним ключем. Операція $D_{K_2}(E_{K_1}(P_1))$ при шифруванні (відповідно $E_{K_1}(P_1)$ при розшифруванні) не впливає на криптостійкість і необхідна для сумісності з режимом однократного шифрування.

Дійсно, при $K_1 = K_2$, $C = E_{K_1}(D_{K_1}(E_{K_1}(P))) = E_{K_1}(P)$. При рівних ключах трудоемність атаки зростає у два рази – можна обчислити і зберегти в пам'яті результати розшифрування $D_{K_1}(E_{K_1}(P))$ для всіх претендентів на K_1 , для цього буде потрібно 2^{n+2} спроб. При двох різних ключах обсяг перебору в найгіршому разі становить 2^{2n} спроб. Однак Меркль і Хеллман розробили варіант силової атаки на схему потрійного шифрування. У ході атаки необхідно виконати 2^{n+1} спроб дешифрування; для зберігання проміжних результатів буде потрібно $2^n b$ комірок пам'яті.

Спочатку нульовий блок (блок, що полягає з послідовності двійкових нулів) дешифрується на будь-яких претендентах на ключ K_2 , результати зберігаються в пам'яті, нульовий блок дешифрується на будь-яких претендентах на K_1 з відстеження в якості результату. Потім виконується потрійне шифрування знайденого P_1 на ключ-претендент. Отриманий шифротекст C_1 дешифрується на ключі-претенденти K_1 . Якщо отриманий результат збігається з деяким значенням у пам'яті (результати дешифрування нульового блоку), то з деякою ймовірністю поточна пара ключів є дійсними K_1 і K_2 . Перевірка завершується шифруванням P_2 , якщо результат рівний C_2 – ключі знайдені, якщо ні – пошук триває.

Очевидне вдосконалення схеми потрійного шифрування полягає у використанні трьох різних ключів:

$$C = E_{K_3}(D_{K_2}(E_{K_1}(P))), \quad P = D_{K_3}(E_{K_2}(D_{K_1}(C))).$$

Силова атака для такої схеми зводиться до 2^n спроб дешифрування і комірки пам'яті 2^{nb} [66; 96].

Існує крипостійкий метод потрійного шифрування із двома ключами, відомий під назвою TEMK — Triple Encryption with Minimum Key. Ідея полягає в тому, щоб одержати три різні ключі K_1 , K_2 , K_3 із двох секретних ключів X_1 і X_2 . Для цього необхідно виконати наступні обчислення:

$$K_1 = E_{X_1}(D_{X_2}(E_{X_1}(T_1))),$$

$$K_2 = E_{X_1}(D_{X_2}(E_{X_1}(T_2))),$$

$$K_3 = E_{X_1}(D_{X_2}(E_{X_1}(T_3))).$$

де T_1 , T_2 і T_3 становлять константи, які необов'язково зберігати в секреті.

Ще один відомий спосіб шифрування на трьох різних ключах зводиться до попереднього підсумовування за модулем 2 (операція XOR) ключа K_1 і блоку відкритого тексту, шифрування на ключі K_2 з наступним підсумовуванням за модулем 2, отриманому на попередньому кроці блоку шифротексту і ключа K_3 ;

$$C = K_3 \oplus E_{K_2}(P \oplus K_1),$$

$$P = K_1 \oplus D_{K_2}(C \oplus K_3).$$

Розмір ключів, що маскують K_1 і K_3 , повинен збігатись з розміром блоку і може не збігатися з ключем K_2 .

Вперше цей метод був застосований Рівестом у криптоалгоритмі DESX, запропонований компанією RSA Data Security, Inc. для посилення федерального стандарту DES, а потім блокових шифрах Khufu і Khafre. Зміст описаних дій полягає в тому, щоб заважати криптоаналітику одержати пару “відкритий текст/шифротекст”. Метод змушує криптоаналітика вгадувати не тільки ключ K_2 , але й один з ключів, що маскує. Операція підсумовування за модулем 2 (операція XOR), яка виконується і до і після шифрування, забезпечує методу досить високу крипостійкість.

Якщо $K_1 = K_2$, то для силової атаки буде потрібно $(2^{n+b})/p$ спроб дешифрування, де n — розрядність ключа, b — розрядність блоку і p — кількість відомих відкритих текстів. Якщо $K_1 \neq K_2$, то для атаки із трьома відомими відкритими текстами потребує 2^{n+b+1} спроб. З погляду диференціального і лінійного криптоаналізу такі заходи забезпечують

захист тільки для декількох бітів ключа. Але з обчислювальної точки зору це дуже дешевий спосіб підвищити криптостійкість блокового шифру.

Очевидно, що кратність шифрування можна збільшувати. Так, п'ятикратне шифрування демонструє високий рівень криптостійкості стосовно описаного вище алгоритму. Причому аргументи, аналогічні тим, що були наведені для подвійного шифрування, показують, що чотириразове шифрування порівняно з потрійним лише незначно підвищує криптостійкість:

$$C = E_{K_1}(D_{K_2}(E_{K_3}(D_{K_2}(E_{K_1}(P))))), \quad P = D_{K_1}(E_{K_2}(D_{K_3}(E_{K_2}(D_{K_1}(C)))))$$

Схема сумісна з потрійним шифруванням при $K_1 = K_2$ і однократним шифруванням при $K_1 = K_2 = K_3$. Звичайно, вона буде ще надійнішою, якщо використовувати п'ять незалежних ключів.

2.7. Атаки на блокові шифри

Процес, при якому вживається спроба отримати X , K , або й те, і інше, називається *криптоаналізом*. Однією з можливих атак на алгоритм шифрування є лобова атака, тобто простий перебір усіх можливих ключів. Якщо кількість ключів досить велика, то підібрати потрібний неможливо. При довжині ключа n бітів кількість можливих ключів дорівнює 2^n . Таким чином, ніж довший ключ, тим більш стійким вважається алгоритм для лобової атаки.

Існують різні типи атак, засновані на тому, що противнику відома певна кількість пар незашифрованих повідомлень – зашифроване повідомлення. При аналізі зашифрованого тексту противник часто застосовує статистичні методи аналізу тексту. При цьому він має загальну уяву про тип тексту, наприклад, англійський або російський текст, здійснений файл конкретної ОС, вихідний текст на деякій конкретній мові програмування і т. п. У багатьох випадках криптоаналітик має багато інформації про вихідний текст і має можливість перехоплення одного або декількох незашифрованих повідомлень разом з їх зашифрованим виглядом. Також криптоаналітик може знати основний формат або основні характеристики повідомлення. Вважають, що криптографічна схема абсолютно безпечна, якщо зашифроване повідомлення не містить ніякої інформації про вихідне повідомлення, і що криптографічна схема обчислювально-безпечна, якщо:

1. Ціна розшифрування повідомлення більша від ціни самого повідомлення.

2. Час, необхідний для розшифрування повідомлення, більший періоду життя повідомлення.

2.7.1. Диференціальний криптоаналіз

Розглянемо загалом основний підхід, який використовується при диференціальному і лінійному криптоаналізах. І в тому і в іншому випадку передбачається, що відома досить велика кількість пар (незашифрований текст, зашифрований текст).

Поняття диференціального криптоаналізу було введено Елі Біхамом (Biham) і Аді Шаміром (Shamir) в 1990 році. Кінцеве завдання диференціального криптоаналізу – використовуючи властивості алгоритму, в основному властивості S-box, визначити підключ раунду. Конкретний спосіб диференціального криптоаналізу залежить від розглянутого алгоритму шифрування.

Якщо в основі алгоритму лежить мережа Файстеля, то можна вважати, що блок m складається із двох половинок – m_0 і m_1 . Диференціальний криптоаналіз розглядає відмінності, які відбуваються в кожній половині при шифруванні. (Для алгоритму DES “відмінності” визначаються за допомогою операції XOR, для інших алгоритмів можливий інший спосіб). Обирається пара незашифрованих текстів з фіксованою відмінністю. Потім аналізуються відмінності, що вийшли після шифрування одним раундом алгоритму, і визначаються ймовірності різних ключів. Якщо для багатьох пар вхідних значень, що мають ту саму відмінність X , при використанні того самого підключа однаковими (Y) виявляються і відмінності відповідних вихідних значень, то можна казати, що X тягне Y з певною ймовірністю. Якщо ця ймовірність близька до одиниці, то можна вважати, що підключ раунду знайдений з даною ймовірністю. Через те що раунди алгоритму незалежні, ймовірності визначення підключа кожного раунду слід перемножувати. Вважається, що результат шифрування даної пари відомий. Результати диференціального криптоаналізу використовуються як при розробці конкретних S-box, так і при визначенні оптимальної кількості раундів.

Диференціальний криптоаналіз на основі відмов пристрою

У вересні 1996 р. група фахівців з компанії Bellcore оголосила про новий метод криптоаналізу, що дозволяє ефективно розкривати секретний ключ, який зберігається в пам'яті портативного обладнання, що шифрує або

дешифрує, наприклад, Smart Card або PCMCLA Зберігання ключа в подібних обладнаннях забезпечується за рахунок унікальних характеристик технології TEMPEST. Вперше метод був успішно застосований для розкриття ключа криптосистеми RSA. Подальші дослідження нового методу, що одержав назву диференціального криптоаналізу на основі збоїв обладнання (Differential Fault Analysis DFA), продемонстрували його ефективність при атаці на DES й інші блокові шифри. Так, для розкриття ключа DES знадобилося проаналізувати двісті шифротекстів, отриманих у результаті шифрування відкритих текстів, що зберігаються в пам'яті обладнання. Показано, що застосування потрібного DES-шифрування не впливає на трудоемність атаки.

Відомо, що деякі види випромінювання (наприклад, що іонізує або радіоактивне) призводять до збоїв у роботі електронного устаткування. Саме ця ідея і лягла в основу криптоаналітичної атаки, розробленої криптоаналітиками компанії Bellcore. Передбачається, що криптоаналітик має необмежений доступ до обладнання, що шифрує/дешифрує. Відкритий текст і секретний ключ зберігаються в пам'яті обладнання і недоступні. Криптоаналітик штучно викликає збої в роботі обладнання, піддаючи його до опромінення. Опромінення приводить до інверсії біта (або бітів) в одному з регістрів на деякому проміжному етапі криптографічного перетворення. Для блокових шифрів конструкції Файстеля інверсія виникає на одному із циклів перетворення. Позиція інвертованого біта і номер циклу криптоаналітику також не відомі. Збій у роботі призводить до викривлення шифротексту на виході обладнання. Криптоаналітик намагається розкрити секретний ключ, аналізуючи спотворені і неспотворені шифротексти.

Розглянемо описаний метод на прикладі криптоаналізу DES. Припустимо, що є два різні шифротексти, отримані при шифруванні того самого відкритого тексту на фіксованому ключі. Відомо, який шифротекст отриманий у результаті інверсії одиночного біта в процесі шифрування. Під *інверсією* мається на увазі наступне: біт правої половини вхідного блоку на одному з 16 циклів DES-перетворення міняє вихідне значення 0 на значення 1, або навпаки. Причому позиція біта і номер циклу перетворення вибираються випадково і мають рівномірний розподіл.

На першому етапі атаки необхідно встановити номер циклу перетворення, на якому відбулась інверсія. Припустимо, що інверсія відбулась на останньому, шістнадцятому циклі DES-перетворення. Якщо інверсія відбулась у правій половині блоку (до заключної перестановки), то

два шифротексти будуть різнитися в одному біті, локалізованому в правій половині блоку. Відмінність лівих половин блоків шифротексту визначається виходами тих S-блоків (одного або двох), на вході яких з'явився інвертований біт. Застосування методу диференціального криптоаналізу дозволяє розкрити шість бітів ключа для кожного такого S-блоку.

Розглянемо випадок, коли інверсія виникає на п'ятнадцятому циклі. Для розкриття бітів ключа необхідна участь більше двох S-блоків в останньому циклі перетворення: відмінність у правій половині блоків шифротексту еквівалентна відмінностям на виході F-функції передостаннього циклу. За зафіксованими вихідними змінами можна встановити позицію одиночного інвертованого біта. Крім того, можна визначити, яка зміна правої половини блоку шифротексту приводить до відповідної зміни виходу F-функції на останньому циклі (суми за модулем 2 лівої половини шифротексту й інвертованого біта).

Для розкриття підключа останнього циклу перетворення досить проаналізувати менше 200 шифротекстів. Подальший розвиток атаки можливий у двох напрямках. Перший варіант – пошук секретного ключа шляхом вичерпної перевірки $2^8 = 256$ кандидатів при заданому підключі (нагадаємо, що розрядність підключа – 48 бітів). Альтернативний варіант полягає у використанні знання про підключ, отриманий на останньому циклі для аналізу попередніх циклів. Останній варіант дозволяє успішно атакувати DES у режимі *EDE*-шифрування на трьох різних ключах. Описаний метод працює й у тих випадках, коли інверсія виникає усередині F-функції або процедури генерації підключів.

Метод диференціального криптоаналізу на основі відмов обладнання можна застосовувати для атаки на такі блокові шифри, як IDEA, RC5 і Feal. Деякі блокові шифри, наприклад, Khufu, Khafre і Blowfish використовують заданий секретний ключ для генерації 5 блоків. У цьому випадку описаний метод дозволяє розкривати не тільки ключі, але і самі S-блоки.

Розглянутий вище криптоаналітичний метод дозволяє ефективно розкривати секретний ключ, що зберігається в пам'яті обладнання, навіть тоді, коли сам алгоритм криптографічного перетворення невідомий. Так, наприклад, деталі криптоалгоритму становлять державну таємницю і засекречені Агентством національної безпеки США. Однак апаратна реалізація криптоалгоритму у вигляді мікросхеми Clippers входить до

складу багатьох комерційних обладнань шифрування, наприклад, Fortezza PCMCIA.

Передбачається, що криптографічні ключі зберігаються в пам'яті з асиметричною динамікою збоїв. Це означає, наприклад, що перехід зі стану 1 у стан 0 відбувається з більшою ймовірністю, ніж протилежна подія. Гнізда CMOS мають симетричну динаміку збоїв – усі переходи рівноймовірні. Однак для деяких видів енергонезалежної пам'яті це не так. Наприклад, при впливі ультрафіолетового випромінювання гнізда EEPROM пам'яті ухвалюють значення 0 з більшою ймовірністю, ніж значення 1.

Передбачається також, що в криптографічне обладнання можна вводити деякий відкритий текст m і в результаті шифрування на ключі k , що зберігається в енергонезалежній пам'яті, одержувати на виході шифротекст c . Оскільки пам'ять відкрита на запис (але закрита на читання), в обладнання можна ввести новий i -бітний ключ k^1 замість старого ключа k . Ще раз зазначимо, що основне завдання криптоаналітика зводиться не до дешифрування деякого шифротексту (зробити це дуже просто, тому що є доступ до відповідного обладнання), а до розкриття секретного ключа, що зберігається в пам'яті обладнання шифрування/розшифрування.

Криптоаналітична атака включає два етапи:

1) на першому етапі невідомий секретний ключ k , що зберігається в пам'яті обладнання, використовується для серії шифрувань фіксованого відкритого тексту m_0 . Після кожного шифрування обладнання зазнає опромінення. В результаті послідовність шифротекстів на виході обладнання складатиметься з декількох різних послідовностей $c, c_1, c_2, \dots, c_f$. Зміна шифротексту – перехід від c_i до c_{i+1} обумовлена спрямованою інверсією (з 1 у 0) деяких бітів ключа, іншими словами – заміною ключа k_i на k_{i+1} , $k_i \neq k_{i+1}$. Оскільки $w_f(k) = n/2$, можна припустити, що $f = n/2$ і шифротекст c_f становлять результат шифрування m_0 на нульовому ключі k_f . Відсутність змін у шифротексті після опромінення – природна ознака виродження ключа до нульового стану;

2) на другому етапі виконується відновлення вихідного секретного ключа k_0 за нульовим ключем k_f . Нехай відомий деякий проміжний ключ k_{i+1} , тоді в моделі збоїв можна припустити, що ключ k_i відрізняється від k_{i+1} в одній позиції, $w_f(k_i \oplus k_{i+1}) = 1$. Тоді для розкриття ключа k_i досить зашифрувати m_0 на послідовності ключів k_{i+1} , одержуваних з k_{i+1}

послідовною заміною на нулі. Якщо результат шифрування рівний c_i – ключ відновлений.

Через те що алгоритм шифрування невідомий, то для перевірки ключів доведеться скористатись обладнанням шифрування. Як було зазначено вище, обладнання дозволяє різні ключі. Отже, для відновлення невідомого ключа k_0 за відомим k_f знадобиться перевірити $O(n)$ ключів на кожному із $O(n)$ кроків. Таким чином, складність методу оцінюється як $O(n^2)$ операцій шифрування.

Окрім розкриття секретного ключа, що зберігається в пам'яті обладнання шифрів шифрування, можна спробувати розв'язати і більш складне завдання – ідентифікувати невідомий криптоалгоритм, включаючи визначення функцій циклу, S-блоки та підключі.

Розглянемо метод ідентифікації, виходячи із припущення, що реалізований обладнанням криптоалгоритм відповідає конструкції Файстеля.

Спочатку необхідно встановити, які біти шифротексту отримані в результаті обробки правого, а які – лівого підблоку на останньому циклі криптографічного перетворення. Для цього необхідно виконати серію шифрувань деякого відкритого тексту. У процесі шифрування до обладнання застосовують опромінення, що приводить відповідно до описаної вище моделі збоїв до інверсії деяких бітів. Аналіз відстані Хемінга пари шифротекстів – перекрученого і неспотвореного – дозволяє встановити позиції бітів, інвертованих на останньому або передостанньому циклі перетворення. В якості критерію для вибору шифротекстів використовується деяке граничне значення, як правило, що не перевищує чверті розміру блоку. Таким чином, перекручені шифротексти, для яких відстань Хемінга пари не перевищує заданого граничного значення, отримані в результаті збою на останньому або передостанньому циклі перетворення. Збій на останньому циклі приводить до зміни одного біта в правому і декількох бітів у лівому підблоці. В 32-бітному підблоці можливо не більше 32 інверсій. Отже, для аналізу знадобиться не більше 32 перекручених шифротекстів. Перекручені шифротексти, отримані в результаті збою на останньому циклі, будуть різнитися в одному біті в правому підблоці та декількох бітах у лівому підблоці.

Після визначення бітів правого та лівого підблоків можна встановити їх функціональну залежність. Допустимо, що в криптоалгоритмі застосовується F -функція, що складається з набору S -блоків. Деякі біти правого підблоку надходять на вхід S -блоку; вихідне значення формує біти

лівого підблоку. Для визначення вхідних і вихідних бітів, а також номерів кожного S-блоку досить вибрати всі пари шифротекстів, що різняться в одному біті в правому підблоці. Інформації про відмінності в лівих підблоках для кожного номера позиції в правому підблоці досить для визначення номерів S-блоків, їх розмірів, а також вхідних і вихідних бітів.

Зміст S-блоків можна відновити, побудувавши таблицю $T(x) = S(x \oplus k) \oplus u$, де k – підключ, а невідоме значення u отримане із правого підблоку попереднього циклу. Наприклад, якщо в якості невідомого криптоалгоритму використовується DES, можна відновити зміст S-блоків, за винятком $6+4 = 10$ з $64 \times 4 = 256$ невідомих бітів у кожному блоці. Для LOKI вдається відновити $2^{12} \times 8 - (12+8) = 32\,748$ невідомих бітів у кожному блоці. Відсутні біти можуть бути відновлені в результаті обробки даних, отриманих на останньому циклі перетворення. Описаний метод дозволяє відновити не самі S-блоки, а їх суму за модулем 2 (операція XOR) з підключами. Відновлення S-блоків і підключів можливо шляхом аналізу відмінностей результатів шифрування на декількох ключах і даних з таблиці T . Описаний метод був успішно застосований для реконструкції S-блоків і алгоритму генерації підключів для DES і LOKI.

У деяких блокових шифрах нелінійна функція F може бути реалізована не у вигляді набору констант – S-блоків (як в DES і LOKI), а як послідовність різних арифметичних операцій, зсуву, логічних операцій I , АБО, НІ і т. д. Подібні функції також піддаються відновленню. Можна відновити F -функцію більш загального виду з довільно заданою операцією підсумування (XOR) правої половини бітів підключу і деякою функцією на виході. З метою спрощення аналізу можна інтерпретувати таку F -функцію, як один великий S-блок. Незважаючи на високу трудоемність, завдання відновлення такого S-блоку практично реалізоване – кількість вхідних і вихідних бітів S-блоку не перевищує половини розміру блоку шифротексту. Ефективність такого підходу очевидна – інший метод ідентифікації блокового шифру на основі повного перебору відомих шифрів і можливих ключів має практично необмежену трудоемність. Описана атака застосовувалась для ідентифікації DES у якості невідомого блокового шифру. Зазначимо, що трудоемність відновлення змісту S-блоків суттєво залежить від їх розмірів. Оскільки S-блок DES має 6-бітний вхід, то для його реконструкції досить проаналізувати не більше 64 вхідних значень, отриманих з перекручених блоків шифротексту. Трудоемність аналогічної реконструкції для LOKI значно вище – S-блоки мають 12-бітний вхід (4096 різних значень).

2.7.2. Лінійний криптоаналіз

Метод лінійного криптоаналізу вперше був застосований для атаки блокового FEAL і потім DES. Даний метод використовує лінійні наближення. Це означає, що якщо виконати операцію XOR над деякими бітами відкритого тексту, потім над деякими бітами шифротексту, а потім виконати XOR результатів попереднього підсумування, можна одержати біт, який дорівнює сумі деяких бітів ключа. Це називається лінійним наближенням, яке може бути правильним з деякою ймовірністю p . Якщо $p \neq 1/2$, то цей факт можна використовувати для розкриття бітів ключа. Розглянемо даний метод.

У загальному вигляді лінійний криптоаналіз матиме вигляд:

$$\sum_{i=1}^{i=a} P \oplus \sum_{j=1}^{j=b} C = \sum_{l=1}^{l=c} K,$$

де $i_1, i_2, \dots, i_a, j_1, \dots, j_b$ та $l_1, l_2, \dots, l_c$ – позиції деяких бітів відкритого тексту P , шифротексту C_i ключа K .

При заданій ефективності виразу лінійного криптоаналізу можна визначити один біт ключа K , скориставшись наступним алгоритмом максимуму правдоподібності.

Алгоритм 1.

Крок 1. Нехай T задає кількість відкритих текстів, таких, що ліва частина рівняння дорівнює нулю.

Крок 2. Тоді якщо $T > N/2$ (N – загальна кількість відкритих текстів), розпізнаємо:

$$\sum_{l=1}^{l=c} K = 0 \quad \left(\text{при } p > \frac{1}{2} \right),$$
$$\sum_{l=1}^{l=c} K = 1 \quad \left(\text{при } p < \frac{1}{2} \right),$$

або розпізнаємо:

$$\sum_{l=1}^{l=c} K = 1 \quad \left(\text{при } p > \frac{1}{2} \right),$$
$$\sum_{l=1}^{l=c} K = 0 \quad \left(\text{при } p < \frac{1}{2} \right).$$

Імовірність успіху збільшується зі зростанням N або $p \rightarrow 1/2$.

Приклад. Розглянемо криптоалгоритм DES з n -циклами криптографічного перетворення з урахуванням 48-бітного ключа K_n і F -функції нелінійності. Вираз лінійного криптоаналізу має наступний вигляд:

$$\sum_{i=1}^{i=a} P \oplus \sum_{j=1}^{j=b} C \oplus \sum_{k=1}^{k=b} F_n(C^R, K_n) = \sum_{l=1}^{l=c} K,$$

де C^R – права половина блоку шифротексту.

Оцінюючи ефективність даного виразу при підстановці різних кандидатів-ключів можна відновити K_n і $\sum_{l=1}^{l=c} K_n$. Для цього слід скористатись наступним алгоритмом максимальної правдоподібності.

Алгоритм 2.

Крок 1. Нехай T задає кількість відкритих текстів, таких, що ліва частина рівняння дорівнює нулю для кожного кандидата $K_n^{(i)}$ ($i = 1, 2, \dots$).

Крок 2. Позначимо через T та $T_{\min}$ максимальне і мінімальне значення серед T_i .

Тоді якщо $1/2 T_{\max} - N/2 > 1/2 T_{\min} - N/2$, скориставшись кандидатом для $T_{\max}$, розпізнаємо:

$$\sum_{i=1}^{l=c} k = 0 \text{ (при } p > 1/2),$$

$$\sum_{i=1}^{l=c} k = 1 \text{ (при } p < 1/2).$$

Якщо $1/2 T - N/2 < 1/2 T_{\min} - N/2$ скориставшись кандидатом для $T_{\min}$, розпізнаємо:

$$\sum_{i=1}^{l=c} k = 0 \text{ (при } p > 1/2),$$

$$\sum_{i=1}^{l=c} k = 1 \text{ (при } p < 1/2).$$

Розглянемо лінійні наближення для S -блоків. Завдання полягає в дослідженні ймовірності того, що деякий біт на вході S -блоку рівняється деякому біту на виході. Однак у більш загальному випадку корисно розглянути залежності сум бітів на деяких вхідних і вихідних позиціях.

Для заданого S -блоку S , ($a = 1, 2, \dots, 8$) і чисел $1\alpha, \dots, 63\alpha$ та $1\beta, \dots, 15\beta$ визначимо $NS_a(a, \beta)$ – кількість вихідних значень для 64 можливих значень

на вході S_a таких, що результат логічної операції “I” суми за модулем 2 деяких бітів значення на вході S_a і числа a дорівнює результату логічної операції “I” суми за модулем 2 деяких бітів значення на виході S_a і числа b . Дану умову можна записати у вигляді:

$$NS_a(a, \beta) = \left\{ x / 1 \leq x \leq 64, \sum_{s=0}^5 (x_s \vee a_s) = \sum_{t=0}^3 S_a(x)_t \vee \beta_t \right\},$$

де $S_a(x)_t$ – t -й біт вихідного значення S_a при заданому на вході значенні x .

Якщо $NS_a(a, \beta)$ не дорівнює 32, можна зробити висновок про кореляційну залежність вхідних і вихідних бітів S_a .

Приклад. Рівняння, наведене вище, показує, що четвертий біт вхідних значень S_5 дорівнює сумі всіх бітів вихідних значень із імовірністю $12/64 = 0,19$:

$$NS_5(16, 15) = 12.$$

Наступне рівняння виконується з імовірністю 0,19 для випадково обраного X і зафіксованого K (з урахуванням перестановки в E - і P -блоках):

$$X_{15} \oplus \sum_{7,18,24,29} F(X, K) = K_{22}.$$

З визначення S -блоків випливає, що:

1) $NS_a(a, \beta)$ парне;

2) якщо $a = 1, 32$ або 33 , то $NS_a(a, \beta) = 32$ для всіх S_a і β .

Приклад. Розглянемо криптоалгоритм DES із трьома циклами перетворення. Наступне рівняння виконується для першого циклу з імовірністю 12/64:

$$\sum_{7,18,24,29} X_2 \oplus \sum_{7,8,24,29} P^L \oplus P_{15}^R = K_{22}^1,$$

де $P^R P^L$ – права і ліва половини відкритого тексту.

Аналогічне рівняння виконується для останнього третього циклу перетворень:

$$\sum_{7,18,24,29} X_2 \oplus \sum_{7,8,24,29} C^L \oplus C_{15}^R = K_{22}^3,$$

де $C^R C^L$ – права і ліва половини шифротексту.

Таким чином, лінійна апроксимація для третього циклу має вигляд:

$$\sum_{7,18,24,29} P^L \oplus \sum_{7,8,24,29} C^L \oplus P_{15}^R \oplus C_{15}^R = K_{22}^1 \oplus K_{22}^3.$$

Описаний метод можна узагальнити на повний DES із 16 циклами криптографічного перетворення.

2.7.3. Силова атака на основі розподілених розв'язань

Наприкінці січня 1997 р. компанія RSA Data Security, Inc. анонсувала новий криптографічний конкурс (табл. 2.10). Ціль конкурсу – оцінка криптостійкості федерального стандарту США DES і симетричних криптосистем зі змінною довжиною ключа кратною 8 криптоалгоритмам RC5. Відомо, що 56-бітний ключ стандарту DES не гарантує адекватної криптостійкості у випадку силової атаки (повний перебір ключа). В 1993 р. Вінер (M. J. Wiener) з Bell-Northern Research розробив для силової атаки на DES спеціалізований паралельний комп'ютер, що дозволяє розкривати ключ протягом декількох годин. Спроби усунення основного недоліку DES, пов'язаного з коротким ключем, призвели до появи різних *DES*-модифікацій – методу *DES*-шифрування, методу DESX, запропонованого Рівестом (R. Rivest), а також ряду альтернативних криптосистем, таких, як IDEA і RC5. Відомо також, що окрім криптостійкості існують й інші фактори, що виявляють вплив на вибір довжини ключа симетричної криптосистеми.

Таблиця 2.10

Умови і результати конкурсу компанії RSA

Криптосистема	Приз (тис. дол.)	Початок (9.00)	Час завершення	Тривалість
DES	10	28.01.97	17.06.97 22.40	140 діб
RC-32/12/5	1	28.01.97	28.01.97 00.30	3,5 год.
RC-32/12/6	5	28.01.97	10.02.97 10.00	313 год.
RC-32/12/7	10	28.01.97	20.10.97 11.18	210 діб
RC-32/12/8	10	28.01.97	—	—
RC-32/12/9	10	28.01.97	—	—
RC-32/12/10	10	28.01.97	—	—
RC-32/12/11	10	28.01.97	—	—
RC-32/12/12	10	28.01.97	—	—
RC-32/12/13	10	28.01.97	—	—
RC-32/12/14	10	28.01.97	—	—
RC-32/12/15	10	28.01.97	—	—
RC-32/12/16	10	28.01.97	—	—
RC-32/12	10	28.01.97	—	—

Довжини ключа симетричної криптосистеми. Відповідно до законодавства довжина ключа для експортованих за межі США і Канади криптографічних додатків на основі симетричного криптоалгоритму обмежено 40 бітами. Необхідність теоретичних обґрунтувань у цьому випадку відпадає, неспроможність криптографії з 40-бітним ключем переконливо продемонстрована першими результатами – криптосистема KC5-32/12/5 була “зламана” через 3,5 години після оголошення в Internet умов криптографічного конкурсу (див. табл. 2.1).

Параметри криптосистеми RC5 у табл. 2.1 позначаються як RC 5-X/Y/Z, де X – розрядність блоку в бітах, Y – кількість циклів криптографічного перетворення, Z – довжина ключа в байтах ($Z \times 8$ – розрядність у бітах). Силова атака зводиться, по суті, до дешифрування фіксованого шифротексту на різні ключі при заданому векторі ініціалізації. Так, для атаки на RC5-32/12/5 був заданий шифротекст у шістнадцятиричному вигляді):

12	35	13	64	78	d3	da	08	d9	15	ed	20	89	30	5f	50	68	5c
6c	b4	bf	5b	00	38	ff	44	4e	d9	d4	9b	46	16	fb	12	92	62
9b	d4	7f	1a	8d	48	fe	b6	63	d1	d4	c2	eb	19	0d	86	3f	f4
43	75	9a	58	06	2c	8b	a5	9e	6a	33	30	c3	3e	a8	ab	24	25

IV вектор ініціалізації 8a 16 2f 69 e8 37 98.

Секретний ключ вважається розкритим, якщо результат дешифрування на поточному ключі приводить Кзмістовного тексту англійською мовою. Умови конкурсу, а також усі необхідні вихідні дані, можуть бути отримані з Web-сервера компанії RSA Data Secutky, Inc. за адресою <http://www.rsa.com>.

Зазначимо, що всі результати табл. 3.1, на відміну від ідеї Вінера, отримані за допомогою програмної реалізації криптоалгоритму і масової паралельної атаки із залученням десятків, а іноді і тисяч робочих станцій. Мережа Internet як приклад практичного додатка ідеї розподілених обчислень стала потужним інструментом силової атаки в руках криптоаналітиків і хакерів. Силова атака на основі розподілених обчислень має свою динаміку, пов'язану з активністю мережних користувачів. Проаналізуємо можливості розподіленої силової атаки на прикладі криптосистем RC 5-32/12/5 і RC/12/6.

Для розв'язання ряду завдань практичної криптографії необхідні значні обчислювальні ресурси. При цьому деякі із цих завдань піддаються розпаралелюванню. Таким чином, Internet, який поєднує багато тисяч

робочих станцій, дозволяє ефективно вирішувати трудомісні завдання шляхом координованої і одночасної роботи великої кількості комп'ютерів. Такий підхід, що реалізує можливості комп'ютерних мереж, відомий під назвою *методу розподілених обчислень*. Одна з перших успішних спроб використання Internet для факторизації RSA-модуля з 129 десяткових розрядів почата в 1994 р. Об'єднані зусилля 600 людей і 1600 комп'ютерів, включаючи дві факс-машини, координувалися за допомогою електронної пошти. Недавня факторизація RSA-модуля з 130 десяткових розрядів продемонструвала виняткову ефективність розв'язку, заснованого на об'єднанні методу розподілених обчислень, алгоритму узагальненого числового решета і координації роботи комп'ютерів Web-технології.

Проблема пошуку ключа симетричної криптосистеми шляхом повного перебору ставиться до класу завдань, що допускають розпаралелювання. У цей час має достатні ресурси для успішної силової атаки на DES (або будь-яку іншу криптосистему, порівняну з DES за довжиною ключа) протягом декількох тижнів. Таким чином, аналіз потенційних можливостей методу розподілених обчислень при криптографічних завдань цікавий як для криптоаналітиків, так і для розроблювачів криптографічних додатків.

У 1992 р. Кароні (G. Caronni) і Алемсбергер (W. Alemsberger) розробили програмне забезпечення для координації одночасної роботи мережних комп'ютерів "поганих" паролів локальних UNIX-серверів. Програма дозволила розв'язати поставлене завдання об'єднаними зусиллями 350 робочих станцій. Оголошений у січні 1997 р. криптографічний конкурс відкрив нову область додатка для програмного Кароні – Алемсбергер. Протягом січня програма була перероблена під нову. Програмна реалізація криптоалгоритму RC5, оптимізована для комп'ютера Ultra 1/170, дозволяла перевіряти мільйон ключів криптосистеми RC5-32/12/5 за 6,15 с.

За тиждень до початку атаки організатори розіслали в ряд університетів США і Європи пропозиції про участь у проекті із проханням залучення всіх зацікавлених осіб. До початку атаки стало зрозумілим, що об'єднане загальною метою мережне співтовариство, є достатнім потенціалом для розв'язку поставленого завдання. Запуск проекту з 18:07 за західноєвропейським часом (9:07 PST). Помилки в роботі програмного забезпечення сервера, виявлені Шнайдером (C. Schneider) в 20:30, привели до перезапуску 40 серверів, які брали участь у проекті.

Незважаючи на збої, завдання було вирішено об'єднаними зусиллями – 800 комп'ютерів у Швейцарії, Німеччині, Норвегії і США.

Секретний ключ криптосистеми RC5-32/12/5 знайдений за 231 хв. після перевірки 51 % усіх ключів із середньою продуктивністю 40 млн ключів/с. Відомо, що аналогічна атака, почата раніше Голдбергом (I. Goldberg) з Університету Берклі, успішно завершила перебір 31 % ключів за 210 хв. Зазначимо, що з методологічної точки зору в організації атаки Кароні і Голдберга мало істотні.

У проекті Голубова задіяно 259 комп'ютерів (287 процесорів):

- 97 Ultrasparc;
- 4 Ultrasparc (8 процесорів);
- 120 робочих станцій HP;
- 8 процесорів Pentium;
- 30 Sparcstation 20s.

Усі ресурси були повністю доступні до моменту запуску і залишались незмінними в період здійснення проекту. Середня швидкість перевірки становила 27 млн ключів/с. Керування перебором ключів здійснювалось за допомогою центрального сервера. Одержавши чергову порцію ключів для перевірки, спеціальний клієнт оцінював поточну продуктивність свого комп'ютера і повідомляв результати серверу.

Сервер, у свою чергу, передавав клієнту сегмент ключового простору, що забезпечує 20-хвилинне завантаження з урахуванням поточної продуктивності комп'ютера. Після завершення перевірки сегмента клієнт сповіщав сервер про результати, і цикл повторювався. При цьому всі перевірені, а також неперевірені ключі, які перебувають у роботі, враховувались сервером при організації подальшої перевірки.

Блискавична “розправа” із криптосистемою RC5-32/12/5 стала безумовним стимулом для силової атаки криптосистеми RC5-32/12/6 з 48-бітним ключем (далі в тексті – проект RC5-32/12/6). Проект був запущений 29 січня о 18:00 після переробки програмного забезпечення і попереднього шестигодинного тестування. Існує два підходи до організації силової атаки на основі методу розподілених обчислень. Перший – централізоване керування процесом пошуку за допомогою сервера. Другий – випадковий і незалежний вибір стартової точки в ключовому просторі кожним комп'ютером із всіх, які брали участь у проекті та оповіщення у випадку розкриття ключа. Перший підхід кращий, але пов'язаний з певними труднощами. Так, не виключена можливість

перевантаження сервера потоком запитів з боку клієнтів. Дотого ж деякі ненавмисні дії клієнтів можуть призвести до збоїв у роботі центрального сервера. Не виключений також ризик зловмисного деструктивного впливу. Для відомості до мінімуму зазначених ризиків необхідні додаткові заходи. Відомий метод забезпечення відмовостійкості допускає введення додаткової надмірності “серверів”, а також принцип ієрархії в проектуванні структури зв’язків з усунення деяких ризиків, які є обмеженнями наслідків збоїв і відмов. Застосування автентифікації гарантує дійсність усіх повідомлень при передачі як від клієнта до сервера, так і у зворотному напрямку. Ведення реєстраційного журналу спрощує аналіз і відстеження наслідків відмов.

Організатори проекту навмисно запропонували спрощену схему взаємодії за технологією “клієнт-сервер”. Для реєстрації клієнта та одержання чергового завдання від сервера використовувався найпростіший протокол.

“Запит реєстрація” включає номер версії, тип комп’ютера та іншу необхідну для початку роботи інформацію. “Запит завдання” містить відомості про обсяг сегмента ключового простору і оцінку часу перебору сегмента зазначеного обсягу. Клієнт може відмовитись від участі або призупинити виконання завдання. Факт відмови з боку клієнта встановлюється за нульовим обсягом сегмента в “запиті завдання”. Сервер підтверджує відмову клієнта повідомленням “скидання завдання”. У випадку розкриття ключа або поновлення роботи клієнт повідомляє сервер за допомогою спеціально передбачених повідомлень. Якщо сервер за якимись причинами завершує свою роботу і замість нього включається новий, то у відповідь на повідомлення клієнта “завдання виконане” новий сервер відповідає повідомленням “запит реєстрації”, ніж ініціює реєстрацію клієнта на новому сервері. Для негайної зупинки роботи клієнтів передбачене повідомлення “зупинка”. Сервер управляє сегментуванням ключового простору, періодично записуючи поточний стан на диск з метою оперативного відновлення програмного забезпечення у випадку перезавантаження операційної системи. Сервер динамічно відслідковує час роботи всіх зареєстрованих клієнтів. Якщо час перебору ключового сегмента клієнтом у “запиті завдання” минає, сервер передає сегмент іншому клієнту віртуальної мережі. Після завершення процедури перевірки поточного ключового сегмента клієнту, відповідно до запиту, передається новий сегмент.

Криптоалгоритм RC5 для програмного забезпечення клієнта був реалізований мовою асемблера для процесорів Intel і RS6000. Оптимізація на рівні компілятора мови C виконувалась для наступних операційних систем: AIX, Freebsd, HPUX, IRIX, LINUX, NEXTSTEP, Netbsd, OS2, OSF1, Openbsd, SCO _SV, Sunos, ULTRIX, Windows (NT&95), AMIGA MacOS MIPS, Alpha, (Ultra) SPARC, 486/586 Intel, Pentium Pro, Paragons, ОС для комп'ютерів з паралельною архітектурою (до 16 000 процесорів).

Лавиноподібне залучення учасників у силову атаку на криптосистему RC5-32/12/6 дозволило задіяти значні мережні ресурси. Зазначимо, що пікова продуктивність тестування ключів досягала 440 млн ключів/с, а максимальна кількість одночасно працюючих комп'ютерів – 4 500 одиниць.

Зростання продуктивності в ході силової атаки на криптосистему RC5-32/12/6 порівнюємо із продуктивністю атаки Гольдберга на криптосистему RC5-32/12/5 (27 млн ключів/с), у якій була задіяна фіксована кількість комп'ютерів.

Розглянута організація силової атаки на основі методу розподілених обчислень породжує наступне питання. Чи спроможний один-єдиний центральний сервер упоратись зі зростаючим потоком запитів, і як таке зростання впливає на ефективність розподілених обчислень? Виходячи з даних про пікове навантаження останніх днів проекту, можна припустити, що центральний сервер обслуговував 5 000 клієнтів кожні 30 хв. Це, у свою чергу, означає, що щосекунди до сервера зверталось більше двох клієнтів. Продуктивність сучасних серверів дозволяє легко справлятися з подібним навантаженням при інтенсивності трафіка порядку 300 байт/с. Так, сервер проекту RC5-32/12/6, скомпонований на Sun Ultra 1/170, був завантажений менше, ніж на 2 %, при цьому потреба в оперативній пам'яті склала менше 6 Мбайтів.

Таким чином, експериментальні дані, отримані в ході проекту, підтверджують, що один сервер без перевантаження може обслужити мільйон клієнтів. Відзначимо, що ефективність описаного підходу може бути невисокою через додаткові затримки, що виникають у зв'язку з низькою пропускною здатністю каналів зв'язку .

Оцінимо обчислювальну продуктивність проекту. Оскільки для перевірки одного ключа криптоалгоритму RC₅ потрібно близько 2 000 ключів, то для знаходження методом силової атаки знадобиться $3,24 \times 10^{17}$ ключів. Так як прийнята одиниця виміру обчислювальної продуктивності в один мільйон ключів/с (MIPS/Year, MY) оцінюється як $3,15 \times 10^{13}$ ключів/с, сумарна продуктивність склала 10^4 MY протягом 10

календарних днів. Це свідчить про зростання обчислювальної продуктивності Internet. Для порівняння: обчислювальна продуктивність при факторизації 129-розрядного *RSA*-модуля в 1994 р. склала 5×10^3 Мґ за дев'ять календарних місяців.

За результатами проекту RC5-32/12/5 і RC5-32/12/6 поданий прогноз майбутнього силової атаки на основі розподілених обчислень. У першу чергу зазначимо, що продуктивність перевірки ключів, яка досягнута за останні дні проекту RC5-32/12/6 (330 млн ключів/с), недостатня для ефективної силової атаки на криптосистему RC-332/12/7. Враховуючи продуктивність проекту RC5-32/12/6 для силової атаки криптосистеми RC5-32/12/7, потрібно близько $2^{55}/330 \times 2^{20}$ с – приблизно три з половиною роки безперервної роботи. Однак експериментальні дані свідчать про подвоювання продуктивності кожні три дні. Таким чином, очікувана продуктивність при збереженні тенденції може зрости на порядок порівняно з піковою продуктивністю проекту RC5-32/12/6. У цьому випадку для здійснення атаки знадобиться не більше півроку.

Результати розподіленої силової атаки на криптосистеми RC5-32/12/5 і RC3 32/12/6 продемонстрували слабкість 40- і 48-бітних ключів. Відомо, що в ході обговорення експортного законодавства США було запропоновано обмежити довжину ключа в експортованих криптографічних додатках 56 бітами. Однак, враховуючи зростання продуктивності комп'ютерів (подвоюється кожні 18 місяців за законом Мура) і динаміку розвитку Internet, можна припустити, що 56-бітний ключ також не гарантує адекватної криптостійкості. За оцінками ведучих криптографів і фахівців з комп'ютерних технологій мінімальна довжина ключа симетричної криптосистеми, що гарантує криптостійкість стосовно силової атаки, повинна коливатися від 75 до 90 бітів. Багато фахівців з консервативними поглядами дотримуються думки, що довжина ключа повинна бути не менше 128 бітів.

2.8. Інші відомі блокові шифри

2.8.1. RC2

Криптоалгоритм RC2 становить блоковий шифр із ключем змінної довжини, розроблений Рівестом на замовлення компанії RSA Data Security, Inc. Аббревіатура "BP" означає "код Рональда" (Ron's Code), або "шифр Рівеста" (Rivest's Cipher). Криптоалгоритм розроблявся як альтернатива криптостандарту DES. RC2 працює із блоками по 64 біти, про-грамна реалізація криптоалгоритму приблизно у два – три рази швидша, ніж DES.

Змінна довжина ключа дозволяє домагатись адекватної криптостійкості з урахуванням можливостей силової атаки. Криптоалгоритм RC2 дозволяє виконувати шифрування в різних режимах – ECB, CBC, CFB. Уряд США не забороняє експортувати апаратні і програмні реалізації криптоалгоритмів RC2 і RC4 – при дотриманні обмеження на довжину ключа (40 бітів). Американські компанії за межами США і Канади можуть використовувати криптоалгоритми з довжиною ключа 65 бітів. При шифруванні за криптоалгоритмом RC2 до секретного ключа методом конкатенації додається деякий допоміжний ключ від 40 до 88 бітів. Для виконання дешифрування допоміжний ключ передається одержувачу зашифрованого повідомлення у відкритому вигляді.

2.8.2. RC5

Криптоалгоритм RC5 також розроблений Рівестом на замовлення компанії RSA Data Security, Inc. Це блоковий шифр зі змінною довжиною блоку (32, 64 і 128 бітів), ключа і кількістю циклів криптографічного перетворення (від 0 до 255 бітів). Довжина ключа варіюється від 0 до 2048 бітів. Можливість параметризації дозволяє гнучко набудовувати криптоалгоритм із урахуванням конкретних вимог з криптостійкості та ефективності реалізації. Криптоалгоритм RC5 складається із трьох основних процедур: розширення ключа, шифрування і розшифрування.

У процедурі розширення ключа заданий секретний ключ зазнає спеціальне перетворення з метою заповнення ключової таблиці, причому розмір таблиці залежить від кількості циклів криптографічного перетворення. Ключова таблиця використовується потім для шифрування і розшифрування. Процедура шифрування складається із трьох основних операцій: цілочисельного підсумування, підсумування за модулем 2 (операція XOR) і циклічного зсуву. Безумовна перевага криптоалгоритму полягає в простоті реалізації. Непередбачуваність результату операції циклічного зсуву, що залежить від конкретних вхідних даних при шифруванні, забезпечує необхідний рівень криптостійкості. Дослідження криптостійкості RC5 показали, що варіант криптоалгоритму з розрядністю блоку 64 біти та 12 (і більше) циклами перетворення гарантує адекватну криптостійкість стосовно диференціального та лінійного криптоаналізу.

2.8.3. IDEA

IDEA (International Data Encryption Algorithm) є блоковим симетричним алгоритмом шифрування, розробленим Сюдзя Гавкіт (Xuejia Lai) і Джеймсом Массей (James Massey) зі швейцарського федерального інституту технологій. Перша версія була опублікована в 1990 році. Переглянута версія алгоритму, посилена засобами захисту від диференціальних криптографічних атак, була подана в 1991 році і докладно описана в 1992 році.

IDEA є одним з декількох симетричних криптографічних алгоритмів, якими спочатку передбачалось замінити DES.

Принципи розробки

IDEA є блоковим алгоритмом, який використовує 128-бітовий ключ для шифрування даних блоками 64 біти.

Метою розробки IDEA було створення стійкого криптографічного алгоритму з досить простою реалізацією.

Криптографічна стійкість

Наступні характеристики IDEA характеризують його криптографічну стійкість:

1. Довжина блоку повинна бути достатньою, щоб сховати всі статистичні характеристики вихідного повідомлення. З іншого боку, складність реалізації криптографічної функції зростає експоненціально відповідно до розміру блоку. Використання блоку розміром в 64 біти в 90-і роки означало достатню силу. Більше того, використання режиму шифрування CBC говорить про подальше посилення цього аспекту алгоритму.

2. Довжина ключа повинна бути досить великою для того, щоб запобігти можливості простого перебору ключа. При довжині ключа 128 бітів IDEA вважається досить безпечним.

3. Конфузія: зашифрований текст повинен залежати від ключа складним і заплутаним способом.

4. Дифузія: кожний біт незашифрованого тексту повинен впливати на кожний біт зашифрованого тексту. Поширення одного незашифрованого біта на велику кількість зашифрованих бітів приховує статистичну структуру незашифрованого тексту. Визначити, як статистичні характеристики зашифрованого тексту залежать від статистичних харак-

теристик незашифрованого тексту досить складно. IDEA із цього погляду є дуже ефективним алгоритмом.

В IDEA два останні пункти виконуються за допомогою трьох операцій. Це відрізняє його від DES, де все побудоване на використанні операції XOR і маленьких нелінійних S-boxes.

Кожна операція виконується над двома 16-бітними входами і створює один 16-бітний вихід.

Цими операціями є:

1. Побітова, що виключає XOR, позначуване як $\oplus$.
2. Сума цілих за модулем 2^{16} (за модулем 65536), при цьому входи і виходи трактуються як беззнакові 16-бітні цілі. Цю операцію позначимо як $+$.
3. Множення цілих за модулем $2^{16} + 1$ (за модулем 65537), при цьому входи і виходи трактуються як беззнакові 16-бітні цілі, за винятком того, що блок з одних нулів трактується як 2^{16} . Цю операцію позначимо як $\cdot$.

Ці три операції є несумісними в тому розумінні, що:

1. Не існує пари із трьох операцій, що задовольняють дистрибутивному закону. Наприклад:

$$a \cdot (b + c) \neq (a \cdot b) + (a \cdot c).$$

2. Не існує пари із трьох операцій, що задовольняють асоціативному закону. Наприклад:

$$a + (b \oplus c) \neq (a + b) \oplus c.$$

Використання комбінації із цих трьох операцій забезпечує комплексну трансформацію входу, роблячи криптоаналіз більш важким, ніж в алгоритмі DES, заснованому винятково на функції XOR.

Шифрування

Розглянемо загальну схему шифрування IDEA. Як і в будь-якому алгоритмі шифрування існує два входи: незашифрований блок і ключ. У цьому випадку незашифрований блок має довжину 64 біти, ключ – 128 бітів.

Алгоритм IDEA складається з восьми раундів, за якими іде заключне перетворення. Алгоритм розділяє блок на чотири 16-бітних підблоки. Кожний раунд одержує на вході чотири 16-бітних підблоки і створює чотири 16-бітних вихідних підблоки. Заключне перетворення також

одержує на вході чотири 16-бітних підблоки і створює чотири 16-бітних підблоки. Кожний раунд використовує шість 16-бітних ключів, заключне перетворення використовує чотири підключа, тобто всього в алгоритмі використовується 52 підключа (рис. 2.24).

Послідовність перетворень окремого раунду

Розглянемо послідовність перетворень окремого раунду.

Одним з основних елементів алгоритму, що забезпечують дифузію, є структура, називана МД (множення/додавання).

На вхід цієї структури подаються два 16-бітних значення і два 16-бітних підключа, на виході створюються два 16-бітних значення. Повна комп'ютерна перевірка показує, що кожний біт виходу цієї структури залежить від кожного біта входів незашифрованого блоку і від кожного біта підключів.

Дана структура повторюється в алгоритмі вісім разів, забезпечуючи високоефективну дифузію (рис. 2.25).

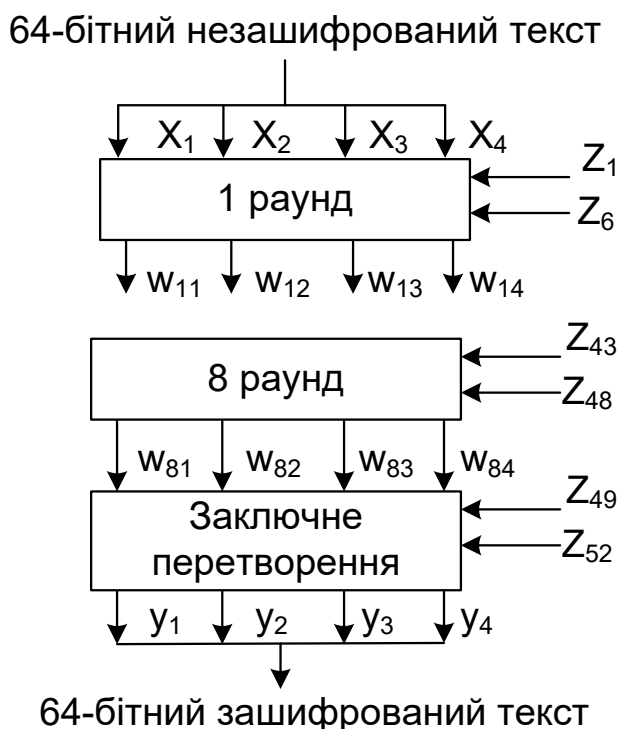


Рис. 2.24. Алгоритм IDEA

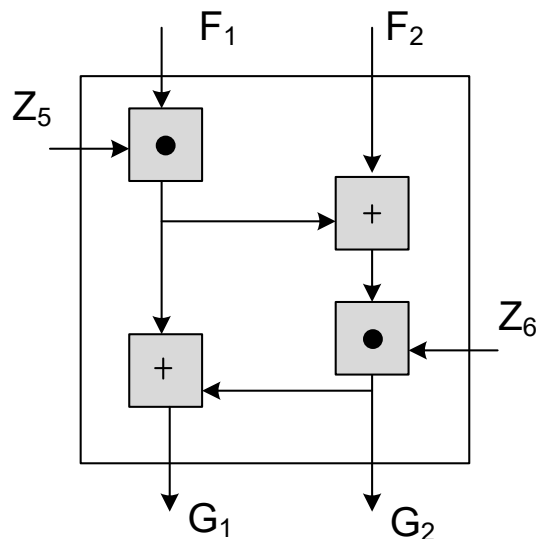


Рис. 2.25. Структура МД (множення/додавання)

Раунд починається з перетворення, яке комбінує чотири вхідних підблоки із чотирма підключами, використовуючи операції додавання і множення. Чотири вихідні блоки цього перетворення комбінуються, використовуючи операцію XOR для формування двох 16-бітних блоків, які є входами МД-структури. Крім того МД-структура має на вході ще два підключи і створює два 16-бітних виходи.

На виході чотири вихідних підблоки першого перетворення комбінуються із двома вихідними підблоками МД-структури, використовуючи XOR для створення чотирьох вихідних підблоків даної ітерації.

Зазначимо, що два виходи, які частково створюються другим і третіми входами (X_2 і X_3), міняються місцями для створення другого і третього виходів (W_{12} і W_{13}). Це збільшує перемішування бітів і робить алгоритм більш стійким для диференціального криптоаналізу.

Розглянемо дев'ятий раунд алгоритму, позначений як заключне перетворення. Це та ж структура, що була описана вище. Єдина різниця полягає в тому, що другий і третій входи міняються місцями.

Це зроблено для того, щоб дешифрування мало ту ж структуру, що і шифрування. Зазначимо, що дев'ята стадія вимагає тільки чотири вхідні підключі, у той час як для перших восьми стадій для кожної з них необхідно шість вхідних підключів (рис. 2.26, 2.27).

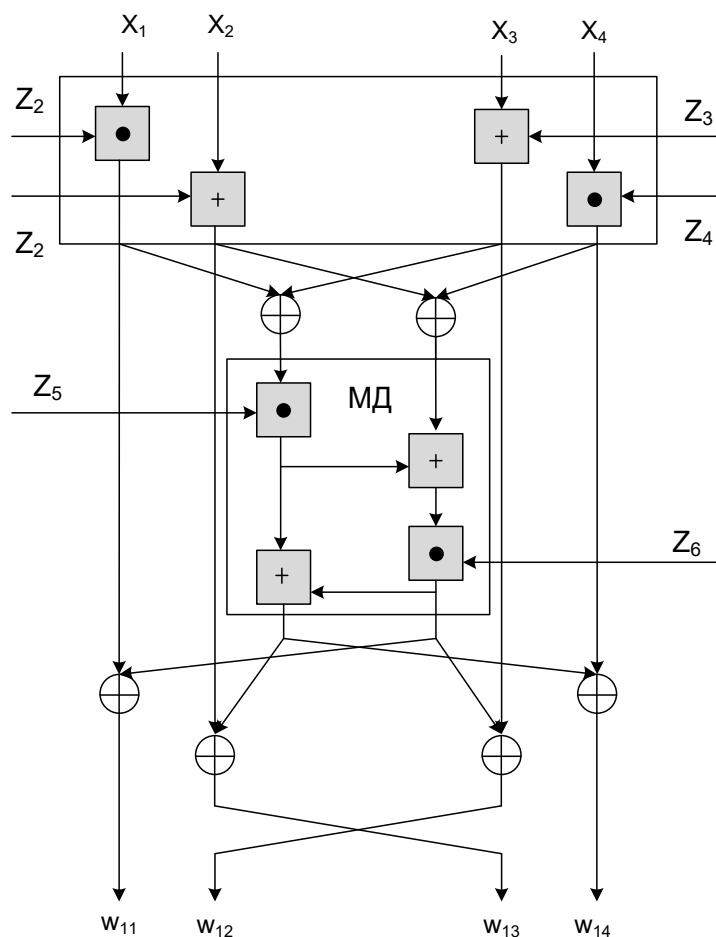


Рис. 2.26. I-й раунд IDEA

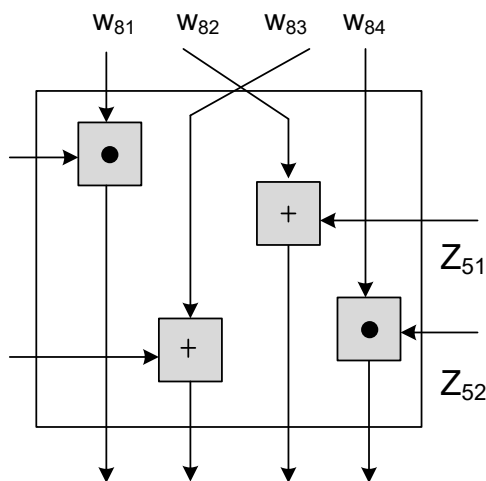


Рис. 2.27. Заключне перетворення

Створення підключів

П'ятдесят два 16-бітних підключа створюються з 128-бітного ключа шифрування в такий спосіб. Перші вісім підключів, які позначено як Z_1 , Z_2 , ..., Z_8 , виходять безпосередньо із ключа, при цьому Z_1 рівний першим 16

бітам, Z_2 рівний наступним 16 бітам і т. д. Потім відбувається циклічний зсув ключа вліво на 25 бітів, і створюються наступні вісім підключів. Ця процедура повторюється доти, доки не будуть створені всі 52 підключа.

Зазначимо, що кожний перший підключраунду отриманий зі своєї підмножини бітів ключа. Якщо весь ключ позначити як $Z_{[1..128]}$, то першими ключами у восьми раундах будуть:

$$\begin{aligned} Z_1 &= Z_{[1..16]} & Z_{25} &= Z_{[76..91]} \\ Z_7 &= Z_{[97..112]} & Z_{31} &= Z_{[44..59]} \\ Z_{13} &= Z_{[90..105]} & Z_{37} &= Z_{[37..52]} \\ Z_{19} &= Z_{[83..98]} & Z_{43} &= Z_{[30..45]} \end{aligned}$$

На кожному раунді за винятком першого і восьмого використовуються тільки 96 бітів підключа, множина бітів ключа на кожній ітерації не перетинається, і не існує відносини простого зсуву між підключами різних раундів. Це відбувається через те, що на кожному раунді використовується тільки шість підключів, у той час як при кожній ротації ключа виходить вісім підключів.

Розшифрування

Процес розшифрування аналогічний процесу шифрування і полягає у використанні зашифрованого тексту як входу в ту ж саму структуру IDEA, але з іншим набором ключів. Ключі, які використовуються при розшифруванні, $U_1, \dots, U_{52}$ виходять із ключів, які шифрують, у такий спосіб:

1. Перші чотири підключа i -ого раунду розшифрування виходять із перших чотирьох підключів 10-го раунду шифрування, де стадія заключного перетворення вважається 9-м раундом. Перший і четвертий ключі розшифрування еквівалентні мультиплікативній інверсії за модулем $2^{16} + 1$ відповідно першого і четвертого підключів шифрування. Для раундів з 2 по 8 другий і третій підключі розшифрування еквівалентні адитивній інверсії за модулем 2^{16} відповідно третього та другого підключів шифрування. Для раундів 1 і 9 другий і третій підключі розшифрування еквівалентні адитивній інверсії за модулем 2^{16} відповідно другого і третього підключів шифрування.

2. Для перших восьми раундів останні два підключа і раунди розшифрування еквівалентні останнім двом підключам 9-го раунду шифрування. Для мультиплікативної інверсії використовується нотація Z_{j-1} , тобто:

$$Z_j \cdot Z_{j-1} = 1 \bmod (2^{16} + 1).$$

Через те що $2^{16} + 1$ є простим числом, кожне ненульове ціле $Z_j \leq 2^{16}$ має унікальну мультиплікативну інверсію за модулем $2^{16} + 1$. Для адитивної інверсії використовується нотація $-Z_j$, таким чином маємо:

$$-Z_j + Z_j = 0 \bmod 2^{16}.$$

Для доказу того, що алгоритм розшифрування з відповідними підключачами має коректний результат, розглянемо одночасно процеси шифрування і розшифрування. Кожний з восьми раундів розбитий на дві стадії перетворення, перша з яких називається трансформацією, а друга – шифруванням (рис. 2.28, 2.29).

Розглянемо перетворення, виконувані в прямокутниках на обох рисунках. При шифруванні підтримуються наступні співвідношення на виході трансформації:

$$\begin{aligned} Y_1 &= W_{81} \cdot Z_{49} & Y_3 &= W_{82} + Z_{51} \\ Y_2 &= W_{83} + Z_{50} & Y_4 &= W_{84} \cdot Z_{52}. \end{aligned}$$

Перша стадія першого раунду процесу дешифрування підтримує наступні співвідношення:

$$\begin{aligned} J_{11} &= Y_1 \cdot U_1 & J_{13} &= Y_3 + U_3 \\ J_{12} &= Y_2 + U_2 & J_{14} &= Y_4 \cdot U_4. \end{aligned}$$

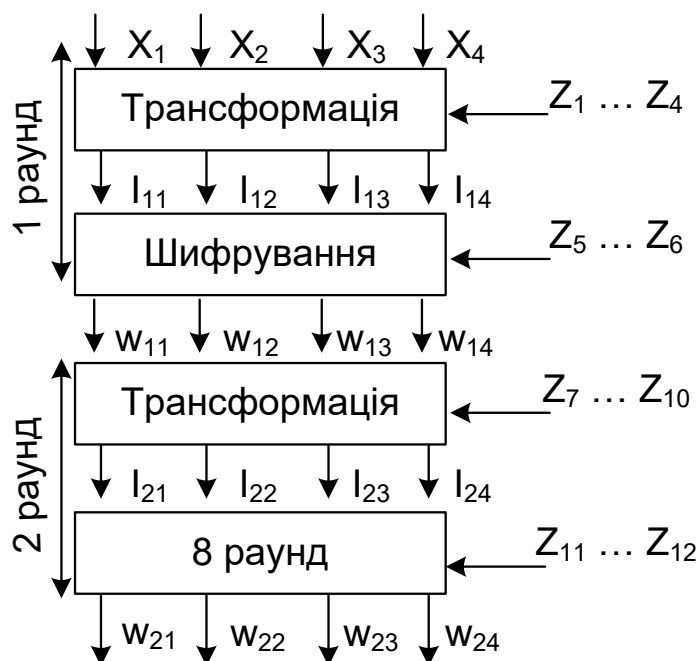


Рис. 2.28. Шифрування IDEA

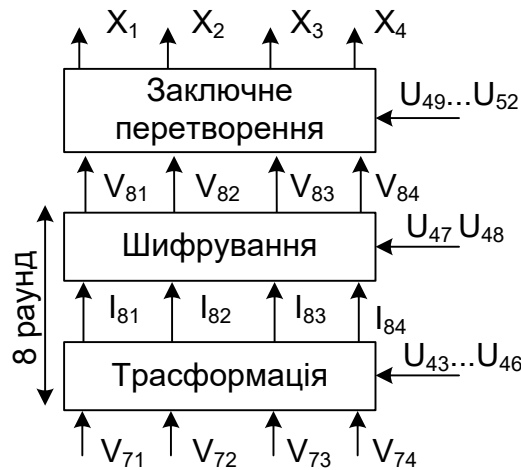


Рис. 2.29. Розшифрування IDEA

Підставляючи відповідні значення, одержимо:

$$\begin{aligned}
 J_{11} &= Y_1 \cdot Z_{49-1} = W_{81} \cdot Z_{49} \cdot Z_{49-1} = W_{81} \\
 J_{12} &= Y_2 + -Z_{50} = W_{83} + Z_{50} = W_{83} + Z_{50} + -Z_{50} = W_{83} \\
 J_{13} &= Y_3 + -Z_{51} = W_{82} + Z_{51} + -Z_{51} = W_{82} \\
 J_{14} &= Y_4 \cdot Z_{52-1} = W_{84} \cdot Z_{52} \cdot Z_{52-1} = W_{84}.
 \end{aligned}$$

Таким чином, вихід першої стадії процесу розшифрування еквівалентний входу останньої стадії процесу шифрування за винятком чергування другого і третього блоків. Тепер розглянемо наступні відносини:

$$\begin{aligned}
 W_{81} &= I_{81} \oplus \text{MAR}(I_{81} \oplus I_{83}, I_{82} \oplus I_{84}) \\
 W_{82} &= I_{83} \oplus \text{MAR}(I_{81} \oplus I_{83}, I_{82} \oplus I_{84}) \\
 W_{83} &= I_{82} \oplus \text{MAL}(I_{81} \oplus I_{83}, I_{82} \oplus I_{84}) \\
 W_{84} &= I_{84} \oplus \text{MAL}(I_{81} \oplus I_{83}, I_{82} \oplus I_{84}),
 \end{aligned}$$

де $\text{MAR}(X, Y)$ – правий вихід МД-структури із входами X та Y ;

$\text{MAL}(X, Y)$ – лівий вихід МД-структури із входами X та Y .

Одержимо:

$$\begin{aligned}
 V_{11} &= J_{11} \oplus \text{MAR}(J_{11} \oplus J_{13}, J_{12} \oplus J_{14}) = W_{81} \oplus \text{MAR}(W_{81} \oplus W_{82}, W_{83} \oplus W_{84}) = \\
 &= I_{81} \oplus \text{MAR}(I_{81} \oplus I_{83}, I_{82} \oplus I_{84}) \oplus \text{MAR}[I_{81} \oplus \text{MAR}(I_{81} \oplus I_{83}, I_{82} \oplus I_{84}) \oplus I_{83} \oplus \\
 &\oplus \text{MAR}(I_{81} \oplus I_{83}, I_{82} \oplus I_{84}), I_{82} \oplus \text{MAL}(I_{81} \oplus I_{83}, I_{82} \oplus I_{84}) \oplus I_{84} \oplus \text{MAL}(I_{81} \oplus I_{83}, I_{82} \oplus \\
 &\oplus I_{84})] = I_{81} \oplus \text{MAR}(I_{81} \oplus I_{83}, I_{82} \oplus I_{84}) \oplus \text{MAR}(I_{81} \oplus I_{83}, I_{82} \oplus I_{84}) = I_{81}.
 \end{aligned}$$

Аналогічно ми маємо:

$$V_{12} = I_{83}, V_{13} = I_{82}, V_{14} = I_{84}.$$

Через те що вихід другої стадії процесу розшифрування еквівалентний входу передостанньої стадії процесу шифрування за винятком чергування другого і третього підблоків, аналогічно можна показати, що

$$V_{81} = I_{11}, V_{82} = I_{13}, V_{83} = I_{12}, V_{84} = I_{14}.$$

Нарешті, через те що вихід трансформації процесу розшифрування еквівалентний першій стадії процесу шифрування за винятком чергування другого і третього підблоків, виходить, що вихід всього процесу шифрування еквівалентний входу процесу шифрування.

2.8.4. Алгоритм SAFER

Криптоалгоритм SAFER (Secure And Fast Encryption Routine) становить блоковий шифр, розроблений на замовлення корпорації Cylink [54]. Довжина секретного ключа в одній з версій дорівнює 64 біти. Криптоалгоритм орієнтований на байтову обробку блоків по 64 біти, і має змінну кількість циклів криптографічного перетворення (до 10, 6). На відміну від більшості блокових шифрів має різні процедури шифрування і розшифрування. Перша версія SAFER з довжиною ключа 64 біти відома за назвою SAFER K-64. Друга версія – SAFER K-128 має 128-бітний ключ. Результати криптоаналізу продемонстрували криптостійкість SAFER K-64 стосовно диференціального та лінійного криптоаналізів при дотриманні однієї умови – кількість циклів перетворення повинна бути більше шести. В результаті досліджень було виявлено слабе місце процедури перетворення ключа. У нових модифікаціях SAFER SK-64 і SAFER SK-128 виявлений недолік усунутий. Модифікація SAFER SK-40 має 40-бітний ключ і п'ять циклів перетворення та вимагає більшого обсягу обчислень при диференціальному і лінійному криптоаналізі порівняно із силовою атакою (повним перебором множини ключів).

2.8.5. FEAL

Криптоалгоритм FEAL був розроблений як альтернатива DES [66]. Оригінальний криптоалгоритм (FEAL-4) був розрахований на програмну реалізацію і мав чотири цикли перетворення при обробці 64-бітного блоку та 64-бітного секретного ключа. Криптоаналітичне дослідження продемонструвало слабкість криптоалгоритму – для розкриття ключа при

атаці на основі вибіркового відкритого тексту досить 20 зразків [44]. Успіхи в криптоаналізі FEAL-8 (вісім циклів перетворення) привели до появи модифікації зі змінною кількістю циклів перетворення FEAL-N. Опубліковані результати успішного диференціального криптоаналізу FEAL-N при $N \leq 31$. Для розкриття секретного ключа FEAL-8 методом лінійного криптоаналізу досить 225 різних відкритих текстів.

2.8.6. Skipjack

Криптоалгоритм Skipjack розроблений Агентством національної безпеки США і є невід'ємною частиною мікросхеми Clipper. Розрахований на обробку 64-бітних вхідних блоків даних (32 цикли перетворення на кожний блок) і має 80-бітний ключ. Очевидно, криптоалгоритм має високу криптостійкість – для порівняння: DES має 56-бітний ключ і 16 циклів перетворення на блок. Однак деталі криптоалгоритму засекречені. Таким чином, відкриті криптоаналітичні дослідження Skipjack неможливі. На замовлення уряду США незалежна група експертів провела обмежені дослідження криптостійкості Skipjack і опублікувала результати у звіті [54; 66]. Програмна реалізація Skipjack неможлива через його таємність.

2.8.7. Blowfish

Blowfish є мережею Файстеля, у якій кількість ітерацій рівна 16. Довжина блоку рівна 64 бітам, ключ може мати будь-яку довжину в межах 448 бітів. Хоча перед початком будь-якого шифрування виконується складна фаза ініціалізації, саме шифрування даних виконується досить швидко. Алгоритм призначений в основному для додатків, у яких ключ міняється нечасто, до того ж існує фаза початкового рукостискання, під час якої відбувається автантифікація сторін і узгодження загальних параметрів та секретів. Класичним прикладом подібних додатків є мережна взаємодія. При реалізації на 32-бітних мікропроцесорах з більшим кешем даних Blowfish значно швидше DES.

Алгоритм складається із двох частин: розширення ключа і шифрування даних. Розширення ключа перетворює ключ довжиною, принаймні, 448 бітів у кілька масивів підключів загальною довжиною 4168 байтів. В основі алгоритму лежить мережа Файстеля з 16 ітераціями. Кожна ітерація складається з перестановки, що залежить від ключа, і підстановки, що залежить від ключа і даних. Операціями є XOR і додавання 32-бітних слів.

Blowfish використовує велику кількість підключів. Ці ключі повинні бути обчислені заздалегідь, до початку будь-якого шифрування або розшифрування даних. Елементи алгоритму:

1. P – масив, що складається з вісімнадцяти 32-бітних підключів:

$P_1, P_2, \dots, P_{18}$.

2. Чотири 32-бітних S -boxes з 256 входами кожний. Перший індекс означає номер S -box, другий індекс – номер входу:

$S_{1,0}, S_{1,1}, \dots S_{1,255};$

$S_{2,0}, S_{2,1}, \dots S_{2,255};$

$S_{3,0}, S_{3,1}, \dots S_{3,255};$

$S_{4,0}, S_{4,1}, \dots S_{4,255}.$

Метод, який використовується для обчислення цих підключів, буде описаний нижче.

Шифрування

Входом є 64-бітний елемент даних X , який ділиться на дві 32-бітні половини: X_l і X_r .

$X_l = X_l \text{ XOR } P_i$

$X_r = F(X_l) \text{ XOR } X_r$

Swap X_l and X_r .

Функція F

Розділити X_l на чотири 8-бітних елементи A, B, C, D .

$F(X_l) = ((S_{1,A} + S_{2,B} \bmod 232) \text{ XOR } S_{3,C}) + S_{4,D} \bmod 232.$

Розшифрування відрізняється від шифрування тим, що P_i використовуються у зворотному порядку.

Генерація підключів

Підключі обчислюються з використанням самого алгоритму Blowfish.

1. Ініціалізувати перший P -масив і чотири S -boxes фіксованим рядком.

2. Виконати операцію XOR P_1 з першими 32 бітами ключа, операцію XOR P_2 із другими 32 бітами ключа і т. д. Повторювати цикл доти, доки увесь P -масив не буде побітово підсумований з усіма бітами ключа. Для коротких ключів виконується конкатенація ключа із самим собою.

3. Зашифрувати нульовий рядок алгоритмом Blowfish, використовуючи підключі, описані в пунктах (1) і (2).

4. Замінити P_1 і P_2 виходом, отриманим на кроці (3).

5. Зашифрувати вихід кроку (3), використовуючи алгоритм Blowfish з модифікованими підключами.

6. Замінити P_3 і P_4 виходом, отриманим на кроці (5).

7. Продовжити процес, замінюючи всі елементи P -масиву, а потім всі чотири S -boxes виходами відповідним чином модифікованого алгоритму Blowfish.

Для створення всіх підключів потрібна 521 ітерація.

2.8.8. REDOC

Криптоалгоритм REDOC розроблений для компанії Cryptech, Inc. [54; 66]. В шифрі використовуються 20-байтовий (160-бітний) ключ і 80-бітний блок (по 10 циклів криптографічного перетворення на кожний блок). REDOC орієнтований на байтову операцію і ефективний при прог-рамній реалізації. Використовує мінливі табличні функції. На відміну від DES, що має фіксований набір таблиць підстановок і перестановок, REDOC II використовує залежні від ключа і відкритого тексту набори таблиць. Особливість криптоалгоритму – використання спеціальних масок – чисел, отриманих із ключів і необхідних для вибору табличних функцій. З погляду силової атаки REDOC дуже надійний: для розкриття ключа потрібно 2^{160} спроб дешифрування. Використовуючи диференціальний криптоаналіз, Біхам і Шамір досягли успіху в криптоаналізі одного циклу криптографічного перетворення REDOC за допомогою 2 300 обраних відкритих текстів [54; 66]. Спроби криптоаналізу двох і більше циклів перетворення закінчилися невдачею.

2.8.9. LOKI

Криптоалгоритм LOKI був поданий в 1990 р. у якості можливої альтернативи DES. В шифрі використовуються 64-бітовий блок і 64-бітовий ключ. Порівняно із силовою атакою метод диференціального криптоаналізу дозволяє ефективніше розкривати ключі LOKI з 11 циклами криптографічного перетворення [66]. Криптоалгоритм має властивість компліментарності, що зменшує трудоємність силової атаки в 256 разів. У дослідженні [54; 66] показано, що LOKI з 14 і менше циклами чутливий до диференціального криптоаналізу.

2.8.10. Khufu

Криптоалгоритм Khufu розроблений Р. Мерклем (R. Merkl) і становить 64-бітний шифр із 512-бітним ключем і змінною кількістю циклів

криптографічного перетворення. Автор криптоалгоритму зазначив, що Khufu з 8 циклами менш криптостійкий при атаці на основі вибіркового відкритого тексту, ніж варіант із 16, 24 або 32 циклами [54]. Відомо, що Khufu стійкий до атаки методом диференціального криптоаналізу. Відомий результат успішної атаки на Khufu з 16 циклами. Для здійснення атаки знадобилося 2^{31} зразків вибіркового відкритого тексту. Результат, однак, не вдалось поширити на більшу кількість циклів перетворення. Силова атака на Khufu безнадійна: 2^{512} спроб дешифрування – такий обсяг перебору не реалізуємий ні за яких умов.

2.8.11. Khafre

Khafre – ще один криптоалгоритм, запропонований Мерклем. Khufu (Хуфу) і Khafre (Хафр) – імена єгипетських фараонів.) За конструкцією цей алгоритм схожий на Khufu. При реалізації Khafre менш ефективний, ніж Khufu, за рахунок використання 64- або 128-бітних ключів і більшої кількості циклів перетворення. Атака на основі методу диференціального криптоаналізу дозволила розкрити ключ Khafre з 16 циклами після години роботи на персональному комп'ютері.

2.8.12. Алгоритм Rijndael

Обґрунтування розробки

При розробці алгоритму враховувалися наступні критерії:

- протидія всім відомим атакам;
- швидкість і компактність коду для широкого кола платформ;
- простота розробки.

У більшості алгоритмів шифрування перетворення кожного раунду має структуру мережі Файстеля. У цьому випадку звичайно частина бітів у кожному проміжному стані просто переміщається без зміни в іншу половину. Перетворення раунду алгоритму Rijndael не має структуру мережі Файстеля. Замість цього перетворення кожного раунду складається із чотирьох різних перетворень, які називаються *шарами*.

Кожний шар розроблявся з урахуванням протидії лінійному та диференціальному криптоаналізу. В основу кожного шару покладена своя власна функція:

1. Нелінійний шар складається з паралельного застосування S-boxes для оптимізації нелінійних властивостей у найгіршому випадку.

2. Шар лінійного переміщення рядків гарантує високий степінь дифузії для декількох раундів.

3. Шар лінійного переміщення стовпців також гарантує високий степінь дифузії для декількох раундів.

4. Додатковий шар ключа складається із простого XOR проміжного стану із ключем раунду.

Перед першим раундом застосовується додаткове забілювання з використанням ключа. Причина цього полягає в наступному. Будь-який шар після останнього або до першого додавання ключа може бути просто знятий без знання ключа і тим самим не додає безпеки в алгоритм (наприклад, початкова і кінцева перестановки в DES). Початкове або кінцеве додавання ключа застосовується також у деяких інших алгоритмах, наприклад, IDEA, SAFER і Blowfish. Для того щоб зробити структуру алгоритму більш простою, шар лінійного переміщення останнього раунду відрізняється від шару переміщення інших раундів. Можна показати, що це в кожному разі не підвищує і не знижує безпеку. Це аналогічно відсутності операції *swar* в останньому раунді DES.

Специфікація алгоритму

Rijndael є блоковим алгоритмом шифрування зі змінною довжиною блоку і змінною довжиною ключа. Довжина блоку і довжина ключа можуть бути незалежно встановлені в 128, 192 або 256 бітах.

Стан, ключ шифрування і кількість раундів

Різні перетворення виконуються над проміжним результатом, названим станом. Стан можна розглядати як двовимірний масив байтів. Цей масив має чотири рядки і різну кількість стовпців, позначуваних як N_b , рівну довжині блоку, ділену на 32. Ключ також можна розглядати як двовимірний масив із чотирма рядками. Кількість стовпців ключа шифрування, позначувана як N_k , дорівнює довжині ключа, діленій на 32. На рис. 2.30 наведений приклад стану (з $N_b = 6$) і ключа шифрування (з $N_k = 4$).

У деяких випадках ці блоки також розглядаються як одновимірні масиви чотирибайтних векторів, де кожний вектор складається з відповідного стовпця. Такі масиви мають довжину 4, 6 або 8 відповідно, і індекси в діапазонах $0 \dots 3$, $0 \dots 5$ або $0 \dots 7$. Чотирибайтні вектори іноді будемо називати словами. Якщо необхідно вказати чотири окремих байти в чотирибайтному векторі, будемо використовувати нотацію (a, b, c, d) , де a, b, c і d є байтами в позиціях 0, 1, 2 і 3, відповідно, у розглянутому стовпці, векторі або слові.

A _{0,0}	A _{0,1}	A _{0,2}	A _{0,3}	A _{0,4}	A _{0,5}
A _{1,0}	A _{1,1}	A _{1,2}	A _{1,3}	A _{1,4}	A _{1,5}
A _{2,0}	A _{2,1}	A _{2,2}	A _{2,3}	A _{2,4}	A _{2,5}
A _{3,0}	A _{3,1}	A _{3,2}	A _{3,3}	A _{3,4}	A _{3,5}

K _{0,0}	K _{0,1}	K _{0,2}	K _{0,3}
K _{1,0}	K _{1,1}	K _{1,2}	K _{1,3}
K _{2,0}	K _{2,1}	K _{2,2}	K _{2,3}
K _{3,0}	K _{3,1}	K _{3,2}	K _{3,3}

Рис. 2.30. Приклад стану (з $N_b = 6$) і ключа шифрування (з $N_k = 4$)

Входи і виходи Rijndael вважаються одномірними масивами з 8 байтів, пронумерованими від 0 до $4 \cdot N_b - 1$. Отже, ці блоки мають довжину 16, 24 або 32 байти, і масив індексується в діапазонах 0 ... 15, 0 ... 23 або 0 ... 31. Ключ вважається одномірним масивом 8-бітних байтів, пронумерованих від 0 до $4 \cdot N_k - 1$. Отже, ці блоки мають довжину 16, 24 або 32 байти, і масив індексується в діапазонах 0 ... 15, 0 ... 23 або 0 ... 31.

Вхідні байти алгоритму відображаються в байти стану в наступному порядку: A_{0,0}, A_{1,0}, A_{2,0}, A_{3,0}, A_{0,1}, A_{1,1}, A_{2,1}, A_{3,1}, ... Байти ключа шифрування відображаються в масив у наступному порядку: K_{0,0}, K_{1,0}, K_{2,0}, K_{3,0}, K_{0,1}, K_{1,1}, K_{2,1}, K_{3,1}, ... Після виконання операції шифрування вихід алгоритму виходить із байтів стану аналогічним чином.

Отже, якщо однорозмірний індекс байта в блоці є n , і двомірний індекс є (i,j) , то маємо:

$$I = n \bmod 4, J = \lfloor n/4 \rfloor, N = i + 4 \cdot j.$$

Більше того, індекс i є також номером байта в чотирибайтному векторі або слові, j є індексом вектора або слова у вкладеному блоці.

Кількість раундів позначається N_r і залежить від значень N_b і N_k , як показано в наступній таблиці 2.11.

Таблиця 2.11

Кількість раундів як функція від довжини блоку і довжини ключа

N_r	$N_b = 4$	$N_b = 6$	$N_b = 8$
$N_k = 4$	10	12	14
$N_k = 6$	12	12	14
$N_k = 8$	14	14	14

Перетворення раунду

Перетворення раунду складається із чотирьох різних перетворень.

Перетворення Bytesub

Перетворення Bytesub є нелінійною байтовою підстановкою, що виконується для кожного байта стану незалежно. Таблиця підстановки є оборотною і сконструйована у вигляді композиції двох перетворень:

1. Спочатку, береться мультиплікативна інверсія в $GF(2^8)$ з визначеним вище представленням, "00" відображається сам у себе.

2. Потім застосовується афінне (в $GF(2)$) перетворення, обумовлене в такий спосіб:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ & & 1 & 1 & 0 & 0 & 0 & 1 \\ & & & 1 & 1 & 1 & 0 & 0 & 1 \\ & & & & 1 & 1 & 1 & 1 & 0 & 0 \\ & & & & & 1 & 1 & 1 & 1 & 0 \\ & & & & & & 0 & 1 & 1 & 1 & 1 & 1 \\ & & & & & & & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Застосування описаного S-box до всіх байтів стану позначається як: Bytesub (State)

На рис. 2.31 показаний результат застосування перетворення Bytesub до State.

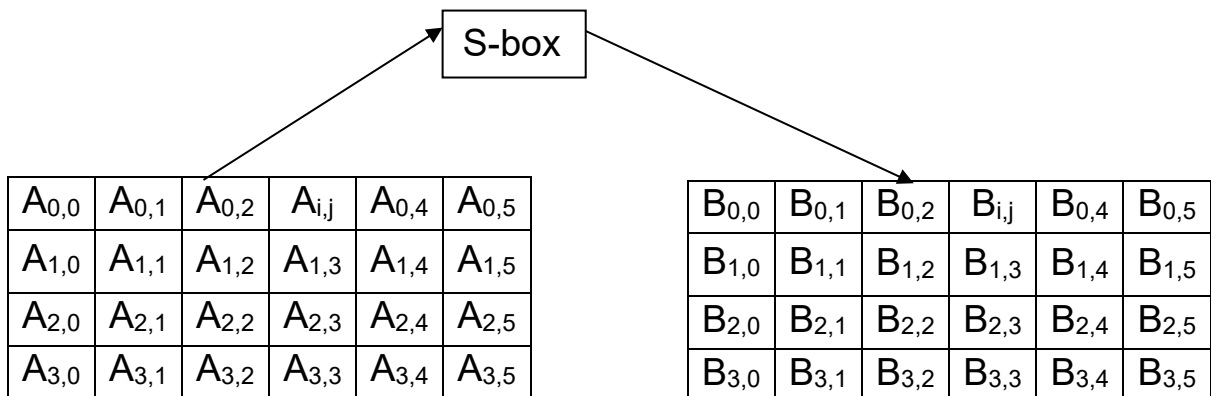


Рис. 2.31. Застосування Bytesub для кожного байта в State

Інверсія Bytesub є застосуванням байтової підстановки відповідно до інверсної таблиці. Це виходить інверсією афінного відображення і мультиплікативною інверсією в $GF(2^8)$.

Перетворення Shiftrow

У Shiftrow рядок стану циклічно зсувається на різні значення. Нульовий рядок не зсувається. Рядок 1 зсувається на C_1 байтів, рядок 2 –

на C_2 байтів, рядок 3 – на C_3 байтів. Величини C_1 , C_2 і C_3 залежать від N_b . Значення наведені в табл. 2.12.

Таблиця 2.12

Величина зсуву залежно від довжини блоку

N_b	C_1	C_2	C_3
4	1	2	3
6	1	2	3
8	1	3	4

Операція зсуву рядків на зазначенні значення позначається як Shiftrow (State)

Інверсією для Shiftrow є циклічний зсув трьох нижніх рядків відповідно на $N_b - C_1$, $N_b - C_2$ і $N_b - C_3$ байтів, щоб байт у позиції j у рядку i переміщався в позицію $(j + N_b - C_i) \bmod N_b$.

Перетворення Mixcolumn

У Mixcolumn стовпці стану розглядаються як поліноми в $GF(2^8)$ і множаться за модулем $x^4 + 1$ на фіксований поліном:

$$c(x) = "03" x^3 + "01" x^2 + "01" x + "02".$$

Даний поліном є взаємнопростим з $x^4 + 1$ і, отже, інвертуємий. Як було описано вище, це може бути записане у вигляді множення матриці. Нехай $b(x) = c(x) / a(x)$

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}.$$

Застосування даної операції до всіх стовпців стану позначається як: Mixcolumn (State).

Інверсія Mixcolumn є аналогічною Mixcolumn. Кожний стовпець перетвориться множенням його на поліном $d(x)$, обумовлений у такий спосіб:

$$("03" x^3 + "01" x^2 + "01" x + "02") \oplus d(x) = "01".$$

У результаті одержуємо:

$$d(x) = "0B" x^3 + "0D" x^2 + "09" x + "0E".$$

Додавання із ключем раунду

Виконується операція побітового XOR ключа раунду з поточним станом. Довжина ключа раунду дорівнює довжині блоку N_b . Дане перетворення позначається як:

Addroundkey (State, Roundkey).

Addroundkey є інверсією самого себе.

Створення ключів раунду

Ключі раунду виходять із ключа шифрування за допомогою перетворення, що складається із двох компонентів: розширення ключа і вибір ключа раунду. Основний принцип полягає в наступному:

Загальна кількість бітів ключа раунду дорівнює довжині блоку, помноженій на кількість раундів плюс 1. Наприклад, для довжини блоку 128 бітів і 10 раундів необхідно 1408 бітів ключа раунду.

Ключ шифрування розширюється в Expandedkey.

Ключі раунду виходять із цього Expandedkey у такий спосіб: перший ключ раунду складається з перших N_b слів, другий складається з наступних N_b слів і т. д.

Розширення ключа

Expanded Key є лінійним масивом чотирибайтних слів і позначається як $W [N_b * (N_r + 1)]$. Перші N_k слів складаються із ключа шифрування. Інші слова визначаються рекурсивно. Функція розширення ключа залежить від значення N_k : існує версія функції для N_k , рівним або меншим 6, і версія для N_k , більше 6.

У цьому випадку Subbyte (W) є функцією, яка повертає чотирибайтне слово, у якому кожний байт є результатом застосування S-box Rijndael до байта у відповідній позиції вхідного слова. Функція Rotbyte (W) повертає слово, у якому байти циклічно переставлені таким чином, що для вхідного слова (a, b, c, d) створюється вихідне слово (b, c, d, a).

Можна зазначити, що перші N_k слів заповнюються ключем шифрування. Кожне наступне слово $W[i]$ рівне XOR попереднього слова $W[i - 1]$ і позиціям слова N_k до $W[i - N_k]$. Для слів у позиціях, які кратні N_k , спочатку застосовується перетворення XOR до $W[i - 1]$ і константою раунду. Дане перетворення складається із циклічного зсуву байтів у слові Rotbyte, за яким слідує застосування табличної підстановки для всіх чотирьох байтів у слові (Subbyte).

Відмінність у схемі для $N_k > 6$ полягає в тому, що для і 4-кратних N_k , Subbyte застосовується для $W[i - 1]$ перед XOR.

Константи раунду не залежать від N_k і визначаються в такий спосіб:

$Rcon[i] = (RC[i], '00', '00', '00')$.

$RC[i]$ є елементами в $GF(2^8)$ зі значенням $x^{(i-1)}$ таким, що:

$RC[1] = 1$ (тобто '01').

$RC[i] = x$ (тобто '02') $\cdot (RC[i-1]) = x^{(i-1)}$.

Вибір ключа раунду

Ключ раунду i виходить зі слів буфера ключа раунду $W[N_b * i]$ до $W[N_b * (i + 1)]$.

Алгоритм шифрування

Алгоритм шифрування Rijndael складається з:

- початкового додавання із ключем;
- $N_r - 1$ раундів;
- заключного раунду.

Зауваження: розширений ключ завжди виходить із ключа шифрування і ніколи не специфікується безпосередньо. Проте на вибір самого ключа шифрування обмежень не існує.

Переваги алгоритму

Переваги, що відносяться до аспектів реалізації:

Rijndael може виконуватись швидше, ніж звичайний блоковий алгоритм шифрування. Виконана оптимізація між розміром таблиці і швидкістю виконання.

Rijndael можна реалізувати в смарт-карті у вигляді коду, використовуючи невеликий RAM і маючи невелику кількість циклів. Виконана оптимізація розміру ROM і швидкості виконання.

Перетворення раунду допускає паралельне виконання, що є важливою перевагою для майбутніх процесорів і спеціалізованої апаратури.

Алгоритм шифрування не використовує арифметичні операції, тому тип архітектури процесора не має значення.

Простота розробки

Алгоритм шифрування повністю “самопідтримуваний”. Він не використовує інших криптографічних компонентів, S-boxів, взятих з добре відомих алгоритмів, бітів, отриманих зі спеціальних таблиць, чисел типу p і тому подібних криптоперетворень.

Алгоритм не засновує свою безпеку на ітераціях арифметичних операцій, що погано розуміються.

Компактна розробка алгоритму не дає можливості сховати люки.

Змінна довжина блоку

Довжини блоків від 192 до 256 бітів дозволяють створювати ґеш-функції без колізій, що використовують Rijndael у якості функції стиску. Довжина блоку 128 бітів сьогодні вважається для цієї мети недостатньою.

Розширення

Розробка дозволяє специфікувати варіанти довжини блоку і довжини ключа в діапазоні від 128 до 256 бітів із кроком в 32 біти.

Хоча кількість раундів Rijndael зафіксована в даній специфікації, у випадку виникнення проблем з безпекою він може модифікуватися і мати кількість раундів як параметр.

Розширення. Різна довжина блоку і ключа шифрування

Обробка ключа підтримує довжину ключа, яка була б кратна 4 байтам. Єдиним параметром, який необхідний для визначення іншої довжини ключа, відмінної від 128, 192 або 256 бітів, є кількість раундів алгоритму.

Структура алгоритму допускає довільну довжину блоку, кратну 4 байтам, з мінімумом у 16 байтів. Додавання ключа, Bytesub і Mixcolumn перетворення не залежать від довжини блоку. Єдиним перетворенням, яке залежить від довжини блоку, є Shiftrow. Для кожної довжини блоку повинен бути визначений спеціальний масив C_1 , C_2 , C_3 .

Можна визначити розширення Rijndael, яке також підтримує довжини блоку і ключа між 128 і 256 бітами зі збільшенням в 32 біти. Кількість раундів визначається так:

$$N_r = \max(N_k, N_b) + 6.$$

Це розширює правило для кількості раундів для альтернативних довжин блоку і ключа. Додаткові значення C_1 , C_2 і C_3 визначені в табл. 2.13.

Таблиця 2.13

Величина зсуву залежно від довжини блоку

N_b	C_1	C_2	C_3
5	1	2	3
7	1	2	4

Інші можливості

Обговоримо функції, відмінні від шифрування, які можуть виконуватися алгоритмом Rijndael.

MAC-коди

Rijndael може застосовуватись в якості алгоритму MAC. Для цього слід використовувати блоковий алгоритм у режимі CBC-MAC.

Геш-функція

Rijndael може використовуватись в якості ітераційної геш-функції. При цьому Rijndael застосовується в якості функції раунду. Існує одна можлива реалізація. Рекомендується використовувати довжину блоку і ключа, рівної 256 бітам. Блок повідомлення подається на вхід як ключ шифрування. Іншим входом є змінна, вихід алгоритму XOR з даною змінною, і отримане значення є новим значенням цієї змінної.

Генератор псевдовипадкових чисел

Існує багато способів, за допомогою яких Rijndael можна використовувати в якості генератора псевдовипадкових чисел. Розглянемо один з них, у якому застосовуються довжина блоку і довжина ключа в 256 бітів.

Існує три операції:

Reset:

Ключ алгоритму шифрування і стан встановлюються в нуль.

Seeding (і reseeding):

“Seed біти” вибираються таким чином, щоб забезпечувати мінімальну ентропію. Вони доповнюються нулями доти, доки результуючий рядок не матиме довжину, кратну 256 бітам.

Обчислюється новий ключ шифрування шифруванням за допомогою Rijndael-блоку бітів seed, використовуючи поточний ключ шифрування. Це виконується рекурсивно доти, доки всі блоки seed не будуть оброблені. Стан змінюється шляхом застосування Rijndael з новим ключем шифрування.

Генератор псевдовипадкового числа

Стан змінюється шляхом застосування Rijndael з новим ключем шифрування. Перші 128 бітів стану розглядаються як псевдовипадкове число. Даний крок може бути повторений багато разів.

2.9. Потоківі шифри

Міцні шифри перетворюють відкритий текст у шифротекст по одному біту за операцію.

Генератор потоку ключів видає потік бітів: $k_1, k_2, k_3, \dots, k_i$. Цей потік бітів (іноді називаний ключем, що біжить) і потік бітів відкритого тексту $p_1, p_2, p_3, \dots, p_i$ зазнають операції XOR, в результаті якої виходить потік бітів шифротексту:

$$c_i = p_i \oplus k_i.$$

При розшифруванні для відновлення бітів відкритого тексту операція XOR виконується над бітами шифротексту і тим самим потоком ключів:

$$p_i = c_i \oplus k_i.$$

Безпека системи повністю залежить від властивостей генератора потоку ключів. Якщо генератор потоку ключів видає нескінченний рядок нулів, шифротекст збігатиметься відкритим текстом і перетворення буде безглуздим. Якщо генератор потоку ключів видає повторюваний 16-бітовий шаблон, криптостойкість буде зневажливо мала. У випадку нескінченного потоку випадкових бітів криптостойкість потокового шифру буде еквівалентна криптостійкості одноразового блокноту. Генератор потоку ключів створює бітовий потік, який схожий на випадковий, але в дійсності детермінований і може бути безпомилково відтворений при розшифруванні. Чим ближче вихід генератора потоку ключів до випадкового, тим вища трудоемність криптоаналітичної атаки.

Блокові і потокові шифри реалізуються по-різному. Потокові шифри, що розшифровують дані за одним бітом, не дуже підходять для програмних реалізацій. Блокові шифри легше реалізовувати програмно, тому що вони дозволяють уникнути трудоемних маніпуляцій з бітами і оперують зручними для комп'ютера блоками даних. З іншого боку, потокові шифри більше підходять для апаратної реалізації [24].

2.9.1. Регістри зсуву зі зворотнім зв'язком

Більшість реальних потокових шифрів заснована на *регістрах зсуву зі зворотним зв'язком*.

Регістр зсуву застосовують для генерації ключової послідовності. Регістр зсуву зі зворотним зв'язком складається із двох частин: регістру зсуву і функції зворотнього зв'язку (рис. 2.32).

Регістр зсуву становить послідовність бітів, кількість яких визначається довжиною зсуву регістру. Якщо довжина рівна n -бітам, то регістр називається n -бітовим регістром зсуву. Щоразу, коли потрібно витягти біт, всі біти регістру зсуву зсуваються вправо на 1 позицію. Новий крайній лівий біт є функцією всіх інших бітів регістру.

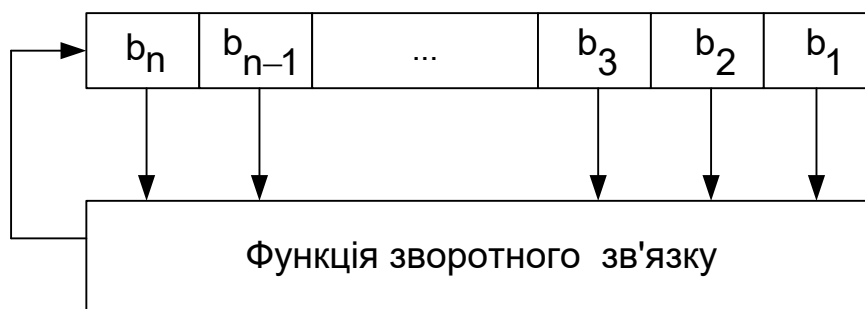


Рис. 2.32. Регістр зсуву зі зворотнім зв'язком

На виході регістру виявляється один, зазвичай молодший значущий біт. Періодом регістру називається довжина одержуваної послідовності до початку її повторення. Найпростішим видом регістру зсуву зі зворотним зв'язком є *регістр зсуву з лінійним зворотним зв'язком (РЗЛЗЗ)* (рис. 2.33).

Зворотний зв'язок становить XOR деяких бітів регістру; ці біти називаються *відповідною послідовністю*.

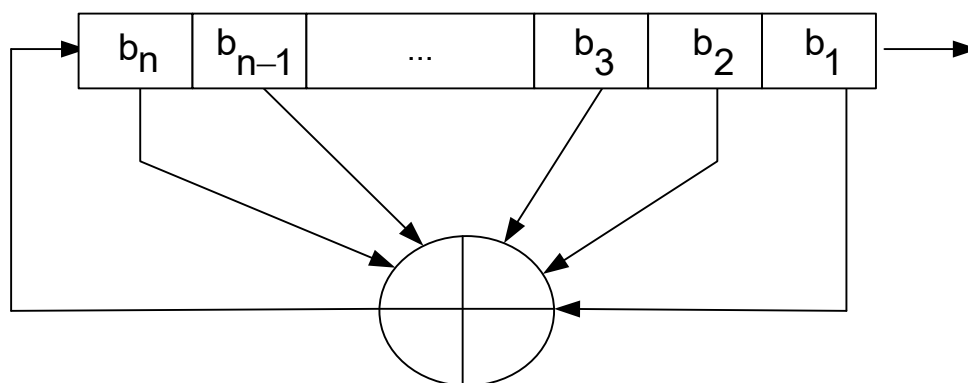


Рис. 2.33. РЗЛЗЗ

На рис. 2.34 показаний 4-бітовий РЗЛЗЗ з відводом від першого і четвертого бітів.

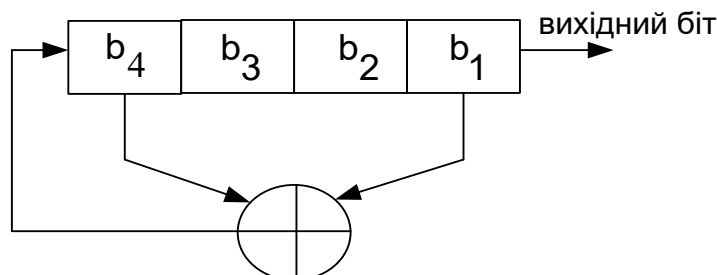


Рис. 2.34. 4-бітний РЗЛЗЗ

Якщо за його допомогою проаналізувати значення 1111, то до повторення реєстр буде приймати наступні внутрішні стани:

1111 0111 1011 0101 1010 1101 0110 0011 1001 0100 0010 0001 1000 1100 1110.

Вихідною послідовністю буде рядок молодших значущих бітів: 111101011001000 ...

РЗЛЗЗ (n -бітовий) може перебувати в одному з $2^n - 1$ внутрішніх станів. Це означає, що теоретично такий реєстр може генерувати псевдовипадкову послідовність із періодом $2^n - 1$ бітів. Кількість внутрішніх станів і періодів рівна $2^n - 1$, тому що заповнення РЗЛЗЗ нулями призведе до того, що реєстр зсуву видаватиме нескінченну послідовність нулів, що абсолютно даремно. Тільки при визначених відвідних послідовностях РЗЛЗЗ циклічно пройде через всі $2^n - 1$ внутрішніх станів. Такі РЗЛЗЗ мають максимальний період, а результат, що вийшов, називається M -послідовністю. Для того щоб конкретний РЗЛЗЗ мав максимальний період, многочлен, асоційований з відвідною послідовністю, повинен бути примітивним за модулем 2, – тобто не розкладатись на добуток двійкових многочленів меншого степеня. Відповідну математичну теорію можна знайти в [42; 43].

Наприклад, многочлен $x^{32} + x^7 + x^5 + x^3 + x^2 + x + 1$ примітивний за модулем 2. Розглянемо цей многочлен у термінах РЗЛЗЗ із максимальним періодом. Степінь многочлена задає довжину РЗЛЗЗ. Вільний член многочлена завжди рівний 1, і його можна вилучити. Степені формальної змінної многочлена задають довжину РЗЛЗЗ виключенням 0-ї, задають відвідну послідовність, яка відлічується від лівого краю реєстру зсуву. Тобто члени многочлена з меншим степенем відповідають позиціям, розташованим ближче до правого краю реєстру. Тоді для взятого 32-бітового зсувного реєстру новий біт генерується за допомогою XOR тридцять другого, сьомого, п'ятого, третього, другого і першого бітів (рис. 2.35).

Тоді РЗЛЗЗ матиме максимальну довжину, циклічно проходячи до повторення через $2^{32} - 1$ різних значень.

Самі по собі РЗЛЗЗ є гарними генераторами псевдовипадкових послідовностей, але вони мають деякі небажані невивадкові властивості. Для РЗЛЗЗ довжини n внутрішній стан становить попередні n вихідних бітів генератора. Навіть якщо схема зворотного зв'язку зберігається в секреті, вона може бути визначена за 2^n вихідними бітами генератора за допомогою алгоритму Берлекемпа – Месі.

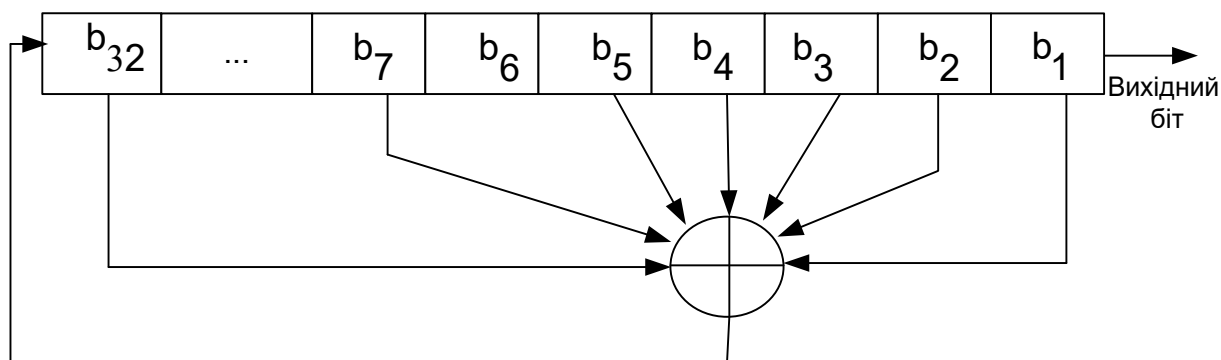


Рис. 2.35. **32-бітний РЗЛЗЗ з максимальним періодом**

Крім того, більші випадкові числа, які генеруються з використанням РЗЛЗЗ, формують підряд біти послідовності з сильнокорельованими властивостями і для деяких типів додатків зовсім не є випадковими.

Незважаючи на це, РЗЛЗЗ часто використовуються при розробці алгоритмів шифрування.

2.9.2. Алгоритм А5

А5 – потоковий шифр, який використовується для шифрування GSM (Group Special Mobile) – європейського стандарту цифрових стільникових мобільних телефонів. А5 складається із трьох РЗЛЗЗ довжиною 19, 22 і 23. Виходом є XOR трьох РЗЛЗЗ. В А5 використовується змінюване керування тактуванням. Кожний регістр тактується залежно від свого біта, потім виконується XOR зі зворотною граничною функцією середніх бітів всіх трьох регістрів. Звичайно на кожному етапі тактується два РЗЛЗЗ. Існує тривіальна атака на відкритий текст, яка заснована на припущенні того, що зміст перших двох РЗЛЗЗ відомий і спробі визначення третього РЗЛЗЗ за ключовою послідовністю. Але ідеї, що лежать в основі А5, дозволяють проектувати надійні потокові шифри. Алгоритм ефективний і задовольняє всім відомим статистичним тестам. Єдина його слабкість – короткі регістри. Варіанти А5 з більш довгими регістрами зсуву і більш щільними многочленами зворотнього зв'язку дозволяють протистояти силовій атаці.

2.9.3. RC4

RC 4 – потоковий шифр зі змінним розміром ключа, розроблений в 1987 р. Рівестом для RSA Data Security, Inc. Алгоритм працює в режимі

OFB. Для формування ключів використовується S-блок розміром 8×8 : $S_0, S_1, \dots, S_{255}$. Елементи становлять перестановку чисел від 0 до 255, а перестановка є функцією ключа змінної довжини. В алгоритмі застосовуються два лічильники, i та j з нульовими початковими j . Для генерації випадкового байта виконуються наступні обчислення:

$$i = (i + 1) \bmod 256;$$

$$j = (j + S_i) \bmod 256.$$

Поміняти місцями S_i та S_j .

$$t = (S_i + S_j) \bmod 256;$$

$$K = S_t.$$

K використовується в операції XOR з відкритим текстом для одержання шифротексту або в XOR із шифротекстом для одержання відкритого тексту. Шифрування виконується приблизно в 10 разів швидше, ніж у DES. Також нескладна й ініціалізація S-блоку. Спочатку S-блок заповнюється за правилом: $S_0 = 0, S_1 = 1, \dots, S_{255} = 255$. Після цього ключ записується в масив: $K_0, K_1, \dots, K_{255}$. Потім при початковому значенні $j = 0$ у циклі виконуються афінне обчислення:

for $l = 0$ to 255:

$$j = (j + S_i + K_i) \bmod 256.$$

Поміняти місцями S_i та S_j .

Компанія RSA Data Security, Inc. затверджує, що алгоритм стійкий до диференціального і лінійного криптоанализу, та що він у високому степені нелінійний. S-блок повільно змінюється при використанні: i і j забезпечується випадкова зміна кожного елементу. C.4 входить у десятки комерційних продуктів, включаючи Lotus Notes, AOCE компанії Apple Computer і Oracle Secure SQL. Цей алгоритм також є частиною специфікації стандарту Стільникової цифрової пакетної передачі даних CDPD (Cellular Digital Packet Data).

2.9.4. SEAL

SEAL – це ефективний потоковий шифр, розроблений в IBM Рогевеєм (P. Rogaway) і Копперсмітом (D. Coppersmith). Алгоритм оптимізований для 32-бітових процесорів. Для нормальної роботи йому потрібно вісім 32-бітових регістрів і пам'ять на декілька кілобайтів. У SEAL передбачений ряд попередніх дій із ключем зі збереженням результатів у декількох таблицях. Таблиці використовуються для прискорення процедур шифрування і розшифрування. Особливістю SEAL є те, що він

дійсно не є традиційним потоковим шифром, а становить сімейство псевдовипадкових функцій. При 160-бітовому ключі k і 32-бітовому регістрі n -SEAL розтягує L в L -бітовий рядок $k(n)$. L може приймати будь-яке значення, менше 64 Кбайтів. SEAL використовує наступне правило: якщо k вибирається випадковим чином, то $k(n)$ повинне відрізнятися від випадкової L -бітової функції n . Практичний ефект того, що SEAL є сімейством псевдовипадкових функцій, полягає в тому, що він зручний у ряді додатків, де незастосовані традиційні потокові шифри. При використанні більшості потокових шифрів створюється односпрямована послідовність бітів: єдиним способом визначити i -й біт (знаючи ключ і позицію i) є генерування всіх бітів до i -го. Відмінність сімейства псевдовипадкових функцій полягає в тому, що можна легко одержати доступ до будь-якої позиції ключової послідовності.

Наприклад, для шифрування жорсткого диска, що складається з множини 512-байтових секторів, можна скористатись сімейством псевдовипадкових функцій, подібних SEAL, і виконати XOR кожного сектору з $k(n)$. Це те ж саме, що якби була виконана операція XOR всього диска з довгою псевдовипадковою функцією, причому будь-яка частина цієї довгої послідовності бітів може бути обчислена незалежно. Сімейство псевдовипадкових функцій також спрощує проблему синхронізації, що зустрічається в стандартних потокових шифрах – можна зашифрувати на ключі k n -е передане повідомлення x_n , виконавши XOR x_n і $k(n)$. Одержувачу не потрібно зберігати стан шифру для відновлення x_n , йому не доводиться турбуватись і про загублені повідомлення, що впливають на процес розшифрування.

2.10. Принципи побудови криптосистем з відкритим ключем

Створення алгоритмів асиметричного шифрування є найбільшим і, можливо, єдиним революційним досягненням в історії криптографії.

Алгоритми шифрування з відкритим ключем розроблявся для того, щоб розв'язати два найбільш важких завдання, що виникли при використанні симетричного шифрування.

Першим завданням є розподіл ключа. При симетричному шифруванні потрібно, щоб обидві сторони вже мали загальний ключ, який якимось чином повинен бути їм заздалегідь переданий. Діффі, один з

основоположників шифрування з відкритим ключем, помітив, що ця вимога заперечує всю суть криптографії, а саме можливість підтримувати загальну таємність при комунікаціях.

Другим завданням є необхідність створення таких механізмів, при використанні яких неможливо було б підмінити кого-небудь із учасників, тобто потрібен електронний підпис. При використанні комунікацій для розв'язку широкого кола завдань, наприклад, у комерційних і приватних цілях, електронні повідомлення і документи повинні мати еквівалент підпису, що втримується в паперових документах. Необхідно створити метод, при використанні якого всі учасники будуть переконані, що електронне повідомлення було послане конкретним учасником. Це більш сильна вимога, ніж автентифікація.

Діффі і Хеллман досягли значних результатів, запропонувавши спосіб розв'язку обох завдань, який радикально відрізняється від усіх попередніх підходів до шифрування.

Спочатку розглянемо загальні риси алгоритмів шифрування з відкритим ключем і вимоги до цих алгоритмів. Зазначимо вимоги, яким повинен відповідати алгоритм, що використовує один ключ для шифрування, інший ключ – для розшифрування, і при цьому обчисленні неможливо визначити ключ, який використовується при розшифруванні, знаючи тільки алгоритм шифрування і ключ, що шифрує.

Крім того, деякі алгоритми, наприклад, RSA, мають наступну характеристику: кожний із двох ключів може використовуватись як для шифрування, так і для розшифрування.

Спочатку розглянемо алгоритми, які володіють обома характеристиками, а потім перейдемо до алгоритмів відкритого ключа, які не мають другої властивості.

При описі симетричного шифрування і шифрування з відкритим ключем будемо використовувати наступну термінологію. Ключ, який використовується у симетричному шифруванні, називатимемо *секретним ключем*. Два ключі, використовувані при шифруванні з відкритим ключем, називатимемо *відкритим ключем* і *закритим (особистим) ключем*. Закритий ключ тримається в секреті і позначається – K_R , відкритий ключ – K_U .

Будемо припускати, що всі учасники мають доступ до відкритих ключів один одного, а закриті ключі створюються локально кожним учасником і, отже, розподіляться не повинні.

У будь-який час учасник може змінити свій закритий ключ і опублікувати складову пари відкритого ключа, замінивши ним старий відкритий ключ.

Діффі і Хеллман описують вимоги, яким повинен задовольняти алгоритм шифрування з відкритим ключем.

1. Обчислювально легко створювати пару ключів (відкритий ключ KU, закритий ключ KR).

2. Обчислювально легко, маючи відкритий ключ і незашифроване повідомлення M, створити відповідне зашифроване повідомлення:

$$C = E_{KU}[M].$$

3. Обчислювально легко розшифрувати повідомлення, використовуючи закритий ключ:

$$M = D_{KR}[C] = D_{KR}[E_{KU}[M]].$$

4. Обчислювально неможливо, знаючи відкритий ключ KU, визначити закритий ключ KR.

5. Обчислювально неможливо, знаючи відкритий ключ KU і зашифроване повідомлення C, відновити вихідне повідомлення M.

Можливо додати шосту вимогу, хоча вона не виконується для всіх алгоритмів з відкритим ключем.

6. Функції, що шифрують і розшифровують, можуть застосовуватись в будь-якому порядку:

$$M = E_{KU}[D_{KR}[M]].$$

Це досить сильні вимоги, які вводять поняття *однобічної криптографічної функції з лямбдою*. *Однобічною криптографічною функцією* називається така функція, у якої кожний аргумент має єдине зворотне значення, при цьому обчислити саму функцію легко, а обчислити зворотну функцію – важко.

$$Y = f(X) \text{ – легко,}$$

$$X = f^{-1}(Y) \text{ – важко.}$$

Звичайно “легко” означає, що проблема може бути вирішена за поліноміальний час від довжини входу. Таким чином, якщо довжина входу має n бітів, то й час обчислення функції пропорційно $n \times \alpha$, де α – фіксована константа. Таким чином, говорять, що алгоритм належить класу

поліноміальних алгоритмів P . Термін “важко” означає більш складне поняття. У загальному випадку будемо вважати, що проблему розв’язати неможливо, якщо зусилля для її розв’язку більше поліноміального часу від величини входу. Наприклад, якщо довжина входу n бітів, і час обчислення функції пропорційний 2^n , то це вважається обчислювально неможливим завданням. На жаль, важко визначити, чи проявляє конкретний алгоритм таку складність. Більше того, традиційні погляди про обчислювальну складність фокусуються на гіршому випадку або на середньому випадку складності алгоритму. Це неможливо для криптографії, де потрібна неможливість інвертувати функцію для всіх або майже всіх значень входів.

Повернемось до визначення однобічної криптографічної функції з люком, яку, подібно однобічній функції, легко обчислити в одному напрямку і важко обчислити у зворотному напрямку доти, доки недоступна деяка додаткова інформація. При наявності цієї додаткової інформації інверсію можна обчислити за поліноміальний час. Таким чином, однобічна криптографічна функція з люком належить сімейству однобічних функцій f_k , таких, що:

$Y = f_k(X)$ – легко, якщо k і X відомі;

$X = f_k^{-1}(Y)$ – легко, якщо k і Y відомі;

$X = f_k^{-1}(Y)$ – важко, якщо Y відомо, але k невідомо.

Розробка конкретного алгоритму з відкритим ключем залежить від відкриття відповідної однобічної криптографічної функції з люком.

Криптоаналіз алгоритмів з відкритим ключем

Як і у випадку симетричного шифрування, алгоритм шифрування з відкритим ключем уразливий для лобової атаки. Контрзахід стандартний: використовувати більші ключі.

Криптосистема з відкритим ключем застосовує визначені неінвертовані математичні функції. Складність обчислень таких функцій не є лінійною від кількості бітів ключа, а зростає швидше, ніж ключ. Таким чином, розмір ключа повинен бути набагато більшим, щоб зробити лобову атаку непрактичною, і досить маленьким для можливості практичного шифрування. На практиці розмір ключа робиться таким, щоб лобова атака була непрактичною, але в результаті швидкість шифрування виявилася досить повільною для використання алгоритму в загальних цілях. Тому

шифрування з відкритим ключем у цей час в основному обмежується додатками керування ключем і підписом, у яких потрібне шифрування невеликого блоку даних.

Інша форма атаки полягає в тому, щоб знайти спосіб обчислення закритого ключа, знаючи відкритий ключ. Неможливо математично довести, що дана форма атаки виключена для конкретного алгоритму відкритого ключа. Таким чином, будь-який алгоритм, включаючи широко використовуваний алгоритм RSA, є підозрілим.

Нарешті, існує форма атаки, специфічна для способів використання систем з відкритим ключем. Це атака ймовірного повідомлення. Припустимо, наприклад, що повідомлення, яке посиляється, складається винятково з 56-бітного ключа сесії для алгоритму симетричного шифрування. Противник може зашифрувати всі можливі ключі, використовуючи відкритий ключ, і може дешифрувати будь-яке повідомлення, що відповідає переданому зашифрованому тексту.

Таким чином, незалежно від розміру ключа схеми відкритого ключа атака зводиться до лобової атаки на 56-бітний симетричний ключ. Захист від подібної атаки полягає в додаванні визначеної кількості випадкових бітів у прості повідомлення.

Основні способи використання алгоритмів з відкритим ключем є шифруванням/розшифруванням, створенням і перевіркою електронного підпису та обміном ключами симетричного шифрування (рис. 2.36).

Шифрування з відкритим ключем складається з наступних кроків:

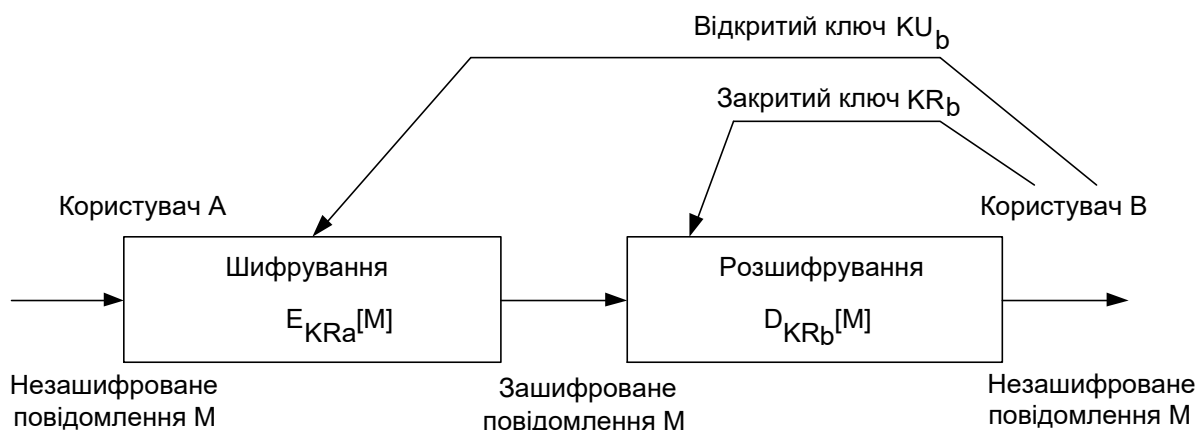


Рис. 2.36. Шифрування з відкритим ключем

1. Користувач В створює пару ключів KU_b і KR_b , які використовуються для шифрування і розшифрування переданих повідомлень.

2. Користувач В робить доступним деякий надійний спосіб свого ключа шифрування, тобто відкритий ключ KU_b . Складова пари закритого ключа KR_b тримається в секреті.

3. Якщо абонент А прагне послати повідомлення абоненту В, він шифрує повідомлення, використовуючи відкритий ключ абонента В KU_b .

4. Коли В одержує повідомлення, він розшифровує його, використовуючи свій закритий ключ KR_b . Ніхто інший не зможе розшифрувати повідомлення, тому що цей закритий ключ знає тільки абонент В.

Якщо користувач (кінцева система) надійно зберігає свій закритий ключ, ніхто не зможе підглянути передані повідомлення.

Створення і перевірка підпису наведена на рис. 2.37 і складається з наступних кроків:

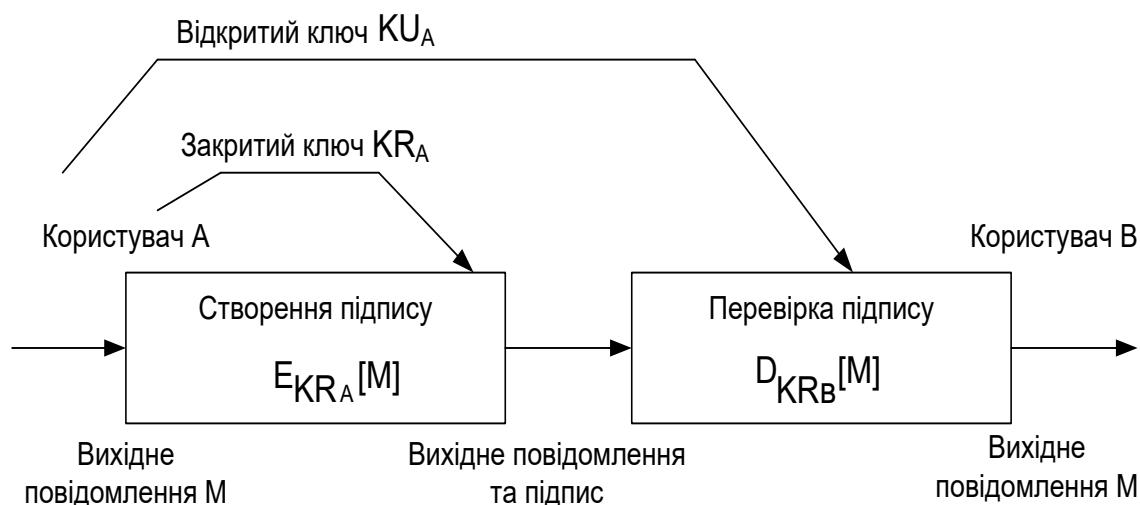


Рис. 2.37. Створення і перевірка підпису

1. Користувач А створює пару ключів KR_A і KU_A , які використовуються для створення і перевірки електронного підпису переданих повідомлень.

2. Користувач А робить доступним деякий надійний спосіб свого ключа перевірки, тобто відкритий ключ KU_A . Складова пари закритого ключа KR_A тримається в секреті.

3. Якщо А прагне послати підписане повідомлення абоненту В, він створює підпис $E_{K_{RA}}[M]$ для цього повідомлення, використовуючи свій закритий ключ K_{RA} .

4. Коли В одержує підписане повідомлення, він перевіряє підпис $D_{K_{UA}}[M]$, використовуючи відкритий ключ А K_{UA} . Ніхто інший не може підписати повідомлення, тому що цей закритий ключ знає тільки абонент А.

Доти, доки користувач або прикладна система надійно зберігають свій закритий ключ, їх підписи достовірні.

Крім того, неможливо змінити повідомлення, не маючи доступу до закритого ключа абонента А; тим самим забезпечується автентифікація і цілісність даних.

У цій схемі все повідомлення підписується, причому для підтвердження цілісності повідомлення потрібно багато пам'яті. Кожне повідомлення повинне зберігатись у незашифрованому вигляді для використання в практичних цілях. Крім того, копія повідомлення також повинна зберігатися в зашифрованому вигляді, щоб можна було перевірити, якщо буде потреба в електронному підписі. Більш ефективним способом є шифрування невеликого блоку бітів, який є функцією від повідомлення. Такий блок, називаний автентифікатором, повинен мати властивість неможливості зміни повідомлення без зміни автентифікатора. Якщо автентифікатор зашифрований закритим ключем відправника, він є електронним підписом, за допомогою якого можна перевірити вихідне повідомлення. Далі ця технологія буде розглядатись в деталях.

Описаний процес створення підпису не забезпечує конфіденційність. Це очевидно в тому випадку, якщо підпис заснований на автентифікаторі, тому що саме повідомлення передається в явному вигляді. Але навіть якщо здійснюється шифрування всього повідомлення, конфіденційність не забезпечується, тому що кожний абонент може розшифрувати повідомлення, використовуючи відкритий ключ відправника.

Обмін ключів: дві сторони взаємодіють для обміну ключем сесії, який надалі можна використовувати в алгоритмі симетричного шифрування. Деякі алгоритми можна задіяти трьома способами, у той час як інші можуть використовуватись одним або двома способами.

Перелічимо найбільш популярні алгоритми з відкритим ключем і можливі способи їх застосування.

Алгоритм	Шифрування / розшифрування	Електронний підпис	Обмін ключів
RSA	Так; непридатний для більших блоків	Так	Так
DSS	Немає	Так	Немає
Діффі -Хеллман	Немає	Немає	Так

2.10.1. Криптосистема шифрування даних RSA

Діффі і Хеллман визначили новий підхід до шифрування, що дозволив розробку алгоритмів шифрування, які задовольняють вимогам систем з відкритим ключем. Одним з перших результатів був алгоритм, розроблений в 1977 році Ріфом Рівестом, Аді Шаміром і Леоном Адлеманом і опублікований в 1978 році. З тих часів алгоритм Rivest-Shamir-Adleman (RSA) широко застосовується практично у всіх додатках, що використовують криптографію з відкритим ключем.

Алгоритм заснований на використанні того факту, що завдання факторизації є важким, тобто легко перемножити два числа, у той час як не існує поліноміального алгоритму знаходження простих співмножників великого числа.

Алгоритм RSA становить блоковий алгоритм шифрування, де зашифровані і незашифровані дані є цілими між 0 і $n - 1$ для деякого n .

Процедури шифрування і розшифрування даних у криптосистемі RSA.

Алгоритм, розроблений Рівестом, Шаміром і Адлеманом, використовує вирази з експонентами. Дані шифруються блоками, кожний блок розглядається як число, менше деякого числа n . Шифрування і розшифрування мають такий вигляд для деякого незашифрованого блоку M і зашифрованого блоку C .

$$C = M^e \pmod{n}$$

$$M = C^d \pmod{n} = (M^e)^d \pmod{n} = M^{ed} \pmod{n}.$$

Як відправник, так і одержувач повинні знати значення n . Відправник знає значення e , одержувач знає значення d . Таким чином, відкритий ключ є $KU = \{e, n\}$ і закритий ключ є $KR = \{d, n\}$. При цьому повинні виконуватись наступні умови:

1. Можливість знайти значення e , d і n такі, що $M^{ed} = M \pmod{n}$ для всіх $M < n$.

2. Відносна легкість обчислення M^e і C^d для всіх значень $M < n$.

3. Неможливість визначити d , знаючи e та n .

Спочатку розглянемо першу умову. Нам необхідне виконання рівності:

$$M^{ed} = M \pmod{n}.$$

Розглянемо деякі математичні поняття, властивості і теореми, які дозволять нам визначити e , d і n .

1. Якщо $(a \cdot b) \equiv (a \cdot c) \pmod{n}$, то $b \equiv c \pmod{n}$, якщо a і n взаємопрості, $\gcd(a, n) = 1$.

2. Позначимо Z_p – всі числа, взаємопрості з p і менші p . Якщо p – просте число, то Z_p – це всі залишки. Позначимо w^{-1} таке число, що $w \cdot w^{-1} \equiv 1 \pmod{p}$.

Тоді $\forall w \in Z_p \rightarrow z: w \times z \equiv 1 \pmod{p}$. Доказ цього випливає з того, що w і p взаємопрості, то при множенні всіх елементів Z_p на w залишками будуть всі елементи Z_p , можливо, переставлені. Таким чином, хоча б один залишок буде рівний 1.

3. Визначимо функцію Ейлера в такий спосіб: $\Phi(n)$ – кількість позитивних чисел, менших n і взаємопростих з n . Якщо p – просте, то $\Phi(p) = p - 1$.

Якщо p і q – прості, то $\Phi(p \times q) = (p - 1) \times (q - 1)$.

У цьому випадку $Z_{p \times q} = \{0, 1, j, (p \times q - 1)\}$.

Перелічимо залишки, які не є взаємопрості з $p \cdot q$:

$\{p, 2 \times p, j, (q - 1) \times p\}$

$\{q, 2 \times q, j, (p - 1) \times q\}$.

У такий спосіб

$$\Phi(p \times q) = p \times q - [(q - 1) + (p - 1) + 1] = p \times q - (p + q) + 1 = (p - 1) \times (q - 1).$$

Розглянемо приклад:

Нехай два простих числа: $p = 7$, $q = 17$.

Обчислити $n = p \times q = 7 \times 17 = 119$.

Обчислити $\Phi(n) = (p - 1) \times (q - 1) = 96$.

Обираємо e так, щоб e було взаємопростим з $\Phi(n) = 96$ і менше, ніж $\Phi(n)$: $e = 5$.

Знайдемо d так, щоб $d \times e \equiv 1 \pmod{96}$ і $d < 96$.

$d = 77$, тому що $77 \times 5 = 385 = 4 \times 96 + 1$.

Результуючі ключі: відкритий $KU = \{5, 119\}$ і закритий $KR = \{77, 119\}$.

Наприклад, потрібно зашифрувати повідомлення $M = 19$.

$19^5 = 66 \pmod{119}$; $C = 66$. Для розшифрування обчислюється $66^{77} \pmod{119} = 19$.

Безпека та швидкодія криптосистеми RSA

Існує багато публікацій, що розглядають тему апаратних реалізацій RSA [8; 16; 17; 24; 27; 34; 35; 40 – 44; 53; 78; 82]. Швидкодія апаратної реалізації RSA приблизно в 1000 разів нижча, ніж реалізації DES.

Швидкодія реалізації RSA з 512-бітовим модулем – 64 Кбіт/с. Існують також мікросхеми RSA, які оперують із 1024-бітовими числами. На сьогодні розробляються мікросхеми, які, використовуючи 512-бітовий модуль, наблизяться до рубежу 1 Мбіт/с.

Виробники також реалізують RSA в інтелектуальних картках, однак продуктивність цих реалізацій невисока. Програмна реалізація DES приблизно в 100 разів швидша програмної реалізації RSA на псевдокоді С.

У табл. 2.14 наведені приклади продуктивності програмної реалізації RSA для 8-бітової експоненти шифрування і різної розрядності модуля.

Таблиця 2.14

Ефективність програмної реалізації для 8-бітової експоненти шифрування і різної розрядності модуля (в секундах)

Операція	512	768	1024
Шифрування	0,03	0,05	0,08
Розшифрування	0,16	0,48	0,93
Обчислення підпису	0,16	0,52	0,97
Перевірка підпису	0,02	0,07	0,08

Створення ключів

Створення ключів включає наступні завдання:

1. Визначити два прості числа p та q .
2. Вибрати e і обчислити d .

Насамперед, розглянемо проблеми, пов'язані з вибором p і q . Через те що значення $n = p \times q$ буде відомо будь-якому потенціальному противнику, для запобігання розкриття p і q ці прості числа повинні бути

обрані з більшої множини, тобто p і q повинні бути більшими числами. З другого боку, метод, який використовується для пошуку більшого простого числа, повинен бути достатньо ефективним.

На сьогодні невідомі алгоритми, які створюють більші прості числа. Процедура, яка використовується для цього, обирає випадкове непарне число з необхідного діапазону і перевіряє, чи є воно простим. Якщо число не є простим, то знову обирається випадкове число доти, доки не буде знайдено просте.

Були розроблені різні тести для визначення того, чи є число простим. Ці тести імовірнісні, тобто тест показує, що дане число ймовірно є простим. Незважаючи на це, вони можуть виконуватись таким чином, що зроблять імовірність близькою до 1. Якщо n “провалює” тест, то воно не є простим. Якщо n “пропускає” тест, то n може як бути, так і не бути простим. Якщо n пропускає багато таких тестів, то можна з високим ступенем вірогідності сказати, що n є простим. Це досить довга процедура, але вона виконується відносно рідко: тільки при створенні нової пари (K_U, K_R) .

На складність обчислень також впливає те, яка кількість чисел буде відкинута перед тим, як буде знайдено просте число. Результат з теорії чисел, відомий як теорема простого числа, дає знати, що простих чисел, розташованих близько n , у середньому по одному на кожні $\ln(n)$ чисел.

Таким чином, у середньому потрібно перевірити послідовність із $\ln(n)$ цілих, перш ніж буде знайдено просте число. Через те що всі парні числа можуть бути відкинуті без перевірки, то потрібно виконати приблизно $\ln(n)/2$ перевірок.

Наприклад, якщо просте число шукається в діапазоні величин 2^{200} , то необхідно виконати близько $\ln(2^{200}) / 2 = 70$ перевірок.

Вибравши прості числа p і q , далі слід обрати значення e так, щоб $\gcd(\Phi(n), e) = 1$ і обчислити значення d , $d = e^{-1} \bmod \Phi(n)$. Існує єдиний алгоритм, називаний розширеним алгоритмом Евкліда, який за фіксований час обчислює найбільший загальний дільник двох цілих чисел і якщо цей загальний дільник дорівнює одиниці, визначається інверсне значення одного за модулем іншого. Таким чином, процедура полягає в генерації серії випадкових чисел і перевірці кожного відносно $\Phi(n)$ доти, доки не буде знайдено число, взаємопросте з $\Phi(n)$. Виникає питання, скільки випадкових чисел доведеться перевірити доти, доки не знайдеться потрібне число, яке буде взаємопростим з $\Phi(n)$. Результати показують, що ймовірність того, що два випадкові числа є взаємопростими, рівна 0,6.

Крипостійкість RSA

Можна визначити чотири можливі підходи для криптоаналізу алгоритму RSA:

1. Лобова атака: перебрати всі можливі закриті ключі.
2. Розкласти n на два прості співмножники. Це дасть можливість обчислити $\Phi(n) = (p - 1)(q - 1)$ і $d = e^{-1} \pmod{\Phi(n)}$.
3. Визначити $\Phi(n)$ безпосередньо, без початкового визначення p і q . Це також дасть можливість визначити $d = e^{-1} \pmod{\Phi(n)}$.
4. Визначити d безпосередньо, без початкового визначення $\Phi(n)$.

Захист від лобової атаки для RSA і йому подібних алгоритмів полягає у використанні великої довжини ключа. Таким чином, ніж більше бітів в e і d , тим краще. Однак через те що обчислення необхідні як при створенні ключів, так і при шифруванні/розшифруванні, ніж більше розмір ключа, тим повільніше працює система.

Більшість дискусій про криптоаналіз RSA фокусується на завданні розкладання n на два прості співмножники. У цей час невідомі алгоритми, за допомогою яких можна було б розкласти число на два простих множники для дуже великих чисел (тобто кілька сотень десяткових цифр). Кращий з відомих алгоритмів дає результат, пропорційний:

$$L(n) = \text{esqrt}(\ln n * \ln(\ln n)).$$

Доки не розроблені кращі алгоритми розкладання числа на прості множники, можна вважати, що величина n від 100 до 200 цифр у цей час є досить безпечною. На сучасному етапі вважається, що число з 100 цифр може бути розкладене на множники приблизно за два тижні. Для дорогих конфігурацій (тобто порядку \$10 млн.) число зі 150 цифр може бути розкладене приблизно за рік. Розкладання числа з 200 цифр перебуває за межами обчислювальних можливостей. Наприклад, навіть якщо обчислювальний рівень в 10^{12} операцій/с досяжний, що вище можливостей сучасних технологій, то буде потрібно понад 10 років для розкладання на множники числа з 200 цифр із використанням існуючих алгоритмів.

Для відомих у цей час алгоритмів завдання визначення (n) за даними e і n , принаймні, порівняна за часом із завданням розкладання числа на множники.

Для того щоб уникнути вибору значення n , яке могло б легко розкладатися на співмножники, на p і q повинне бути накладено багато

додаткових обмежень: p і q повинні один від одного відрізнятись по довжині тільки декількома цифрами. Таким чином, обидва значення p і q повинні бути від 10^{75} до 10^{100} .

Обидва числа $(p - 1)$ і $(q - 1)$ повинні містити великий простий співмножник.

$\gcd(p - 1, q - 1)$ повинен бути маленьким.

Розглянемо деякі з відомих видів атак на криптосистему RSA докладніше.

Мала експонента e

Розглянемо ситуацію, коли є мала експонента e ($e = 3, 5, 7$ і т. д.). Нехай $e = 3$. Допустимо, що A бажає послати повідомлення трьом іншим абонентам, що мають відкриті ключі $(3, n_1), (3, n_2), (3, n_3)$. A шифрує повідомлення і одержує $c_i = m^3 \pmod{n_i}$ для $i = 1, 2, 3$. Противник перехоплює всі 3 повідомлення і становить систему:

$$x = c_1 \pmod{n_1}; x = c_2 \pmod{n_2}; x = c_3 \pmod{n_3}.$$

Оскільки $0 \leq x < n_1 n_2 n_3$ і $m^3 < n_1 n_2 n_3$, то за китайською теоремою про залишки противник може знайти $x = m^3$ і, обчисливши кубічний корінь із x , знайде m .

Таким чином, одержуємо, що не можна використовувати малі експоненти при відправленні великої кількості однакових повідомлень. Або ж слід використовувати добавки ("salting") у повідомлення, у якості яких можуть виступати додані в кінець повідомлення випадкові вектори або ж час відправлення.

При використанні малих експонент також виникає проблема з маленькими повідомленнями, тобто з такими, для яких $m < n^{1/e}$, оскільки в цьому випадку m може бути отримане із шифртексту $c = m^e \pmod{n}$ шляхом обчислення кореня e -го степеня з C . Однак використання добавок у повідомлення допомагає уникнути цієї проблеми.

Атака перебором можливих відкритих текстів.

Якщо повідомлення не велике, противник може спробувати підібрати відкритий текст шляхом перебору всіх можливих варіантів і шифрування їх на відкритому ключі абонента e доти, доки не буде отриманий перехоплений шифртекст c . Такої атаки також можливо запобігти за допомогою добавок у повідомлення.

Використання загальних модулів

Схема RSA непридатна при використанні загальних модулів. Допустимо, є абоненти А і В з відкритими ключами (e_1, n) і (e_2, n) . Центр (загальний сервер або циркуляр) бажає послати обоїм абонентам однакові повідомлення. Центр отримує $m^{e_1} = c_1 \pmod{n}$ і $m^{e_2} = c_2 \pmod{n}$ і посилає c_1 і c_2 А і В відповідно. Протівник перехоплює ці повідомлення. Потім, якщо $(e_1, e_2) = 1$, за розширеним алгоритмом Евкліда можна знайти такі k_1 і k_2 , для яких $e_1 k_1 + e_2 k_2 = 1$. Відповідно, $m^{e_1 k_1} m^{e_2 k_2} = m$. Тобто знайшовши такі k_1 і k_2 (це він може зробити, адже відкриті ключі йому відомі), протівник може обчислити $(c_1)^{k_1} (c_2)^{k_2} = m$.

Якщо повідомлення не велике, протівник може спробувати підібрати відкритий текст шляхом:

Циклічна атака (безключове читання RSA).

Суть її полягає в наступному: знайти найменше ціле u таке, щоб $f = \text{НСД}(c^{eu} - c, n) > 1$.

Якщо $c^{eu} \equiv c \pmod{p}$ і $c^{eu} \equiv c \pmod{q}$, то $f = p$.

Якщо $c^{eu} \equiv c \pmod{p}$ і $c^{eu} \equiv c \pmod{q}$, то $f = q$.

В обох цих випадках протівник одержує або p , або q і факторизує n . Якщо ж $c^{eu} \equiv c \pmod{p}$ і $c^{eu} \equiv c \pmod{q}$, тоді $f = n$ і $c^{eu} \equiv c \pmod{n}$. Однак цей випадок менш імовірний, ніж перші два. Тому узагальнену циклічну атаку можна вважати одним з методів факторизації n .

Мультиплікативні властивості

Система RSA має мультиплікативні властивості. Нехай m_1 і m_2 – два різних відкритих тексти, а c_1 і c_2 – відповідні їм шифртексти. Зазначимо, що $(m_1 m_2)^e = m_1^e m_2^e = c_1 c_2 \pmod{n}$.

Інакше кажучи, шифртекст відкритого тексту $m = m_1 m_2 \in c = c_1 c_2 \pmod{n}$. Ця властивість називається також *гомоморфною властивістю* RSA, яка дозволить здійснити атаку за обраним шифртекстом. Його потрібно враховувати при сполученні схем шифрування на основі RSA і електронному підписі RSA.

Практичні рекомендації

У цей час рекомендується вибирати n розміром як мінімум 768 бітів, а для довготривалого зберігання інформації 1024 або навіть 2048 бітів.

Вибір простих чисел

Оскільки прості числа повинні вибиратись таким чином, щоб їх добуток було неможливо факторизувати, рекомендується брати їх більшими та однакової довжини. Так, для n довжини 1024 бітів p і q повинні бути довжиною 512 бітів.

Різниця чисел p і q ($p - q$) також не повинна бути маленькою, оскільки в цьому випадку $p \cong q$, отже, $p \cong \sqrt{n}$. Таким чином, розкладання n може бути знайдене простим розподілом на всі числа порядку $\sqrt{n}$. Числа p і q повинні бути також “стійкими” простими числами.

Визначення 2.1.1. Число p є *стійким (strong)*, якщо воно задовольняє 3 умовам:

1. $p - 1$ має великий простий дільник, позначимо його як r (тобто $p = 1 \pmod{r}$);

2. $p + 1$ має великий простий дільник, позначимо як s (тобто $p = s^{-1} \pmod{s}$);

3. $r - 1$ має великий простий дільник, позначимо його як t (тобто $r = 1 \pmod{t}$).

Умова 1 не дозволить успішно факторизувати $n(p - 1)$ методом Полларда, який дозволяє швидко розкласти число n на множники, якщо його дільник p має невеликі (скажемо, менше мільйона) прості дільники.

Умова 2 дозволить уникнути $p + 1$ методу Уільямса, що дозволяє розкласти n за умови, що $p + 1$ має невеликі дільники.

Умова 3 дозволить уникнути методу безключового читання RSA (циклическої атаки).

Якщо p вибирається випадково і має досить великий розмір, то як правило, $p - 1$ і $p + 1$ будуть мати великі прості дільники. Однак вибір стійких простих чисел не захищає систему від атаки алгоритмом факторизації на основі еліптичних кривих.

Одержати *стійкі прості числа* можна в такий спосіб. Генеруємо більші прості числа s і t . Потім одержуємо таке число r , що $r - 1$ ділиться на t (для цього розглядаємо непарні числа виду $kt + 1$, де k – послідовні натуральні числа, і перевіряємо їх на простоту, доки не знайдемо просте). Потім, обчислюючи $p = ((sr^{-1}rs^{-1}) \pmod{rs}) + xrs$, де x – деяке ціле число і перевіряючи p на простоту, знаходимо стійке просте число p .

Вибір експоненти e і прискорення роботи

Якщо число e обирається випадковим чином, то одним зі способів прискорення обчислення (тобто зменшення числа зведень у степінь) є наступний алгоритм швидкого зведення (на прикладі):

$$a^{25} = (a^2 a)^{2^2)^2} a.$$

Таким чином, замість 25 множень виконується всього 6 (тобто 4 зведення у квадрат і 2 множення на a). Формально алгоритм виглядає так:

Вхід: $a \in Z_n$ і ціле $0 \leq k < n$, де двійкове представлення $k = \sum_{i=0}^t k_i 2^i$.

Вихід: $a^k \bmod n$.

1. $b=1$. If $k=0$ then return b .
2. $A=a$.
3. If $k_0=1$ then $b=A$.
4. For $i=1$ to t do.
 - 4.1 $A=A^2 \bmod n$.
 - 4.2 If $k_i=1$ then $b=A*b \bmod n$.
5. Return b .

На практиці, як правило, використовується $e = 3$, у цьому випадку необхідно, щоб ні $p - 1$, ні $q - 1$ не ділилися на 3. При такому відкритому ключі операція зашифрування отримується дуже швидко і вимагає одне зведення у квадрат і одне модульне множення (або 2 модульних множення – залежно від реалізації). Іншим значенням, яке часто використовується, є $e = 2^{16} + 1 = 65\,537$. Це число має одну одиницю у двійковому записі і вимагає при використанні описаного алгоритму 16 зведень у квадрат і одне модульне множення. Така експонента має перевагу порівняно з $e = 3$, оскільки в цьому випадку атака, описана раніше, не здійсниться, тому що дуже мала ймовірність, що одне і теж повідомлення буде надіслане $2^{16} + 1$ абонентам.

2.10.2. Криптосистема Ель Гамала

Національний інститут стандартів і технології США (NIST) розробив федеральний стандарт електронного підпису DSS. Для створення електронного підпису використовується алгоритм DSA (Digital Signature Algorithm).

У якості ґеш-алгоритму стандарт передбачає використання алгоритму SHA-1 (Secure Hash Algorithm). DSS спочатку був запропонований в 1991

році і переглянутий в 1993 році у відповідь на публікації, що стосуються безпеки його схеми.

Підхід DSS

DSS використовує алгоритм, який розроблявся для використання тільки в якості електронного підпису. На відміну від RSA, його не можна використовувати для шифрування або обміну ключами. Проте це технологія відкритого ключа. Підхід DSS також використовує сильну ґеш-функцію. Ґеш-код є входом функції підпису разом з випадковим числом k , створеним для цього конкретного підпису. Функція підпису також залежить від закритого ключа відправника KR_A і множини параметрів, відомих всім учасникам. Можна вважати, що ця множина складається із глобального відкритого ключа KU_G . Результатом є підпис, що складається із двох компонентів, позначених як s і r .

Для перевірки підпису одержувач також створює ґеш-код отриманого повідомлення. Цей ґеш-код разом з підписом є входом у функцію верифікації. Функція верифікації залежить від глобального відкритого ключа KU_G і від відкритого ключа відправника KU_A . Виходом функції верифікації є значення, яке повинне рівнятись компоненту r підпису, якщо підпис коректний. Функція підпису така, що тільки відправник, який знає закритий ключ, може створити коректний підпис.

Тепер розглянемо деталі алгоритму, який використовується в DSS.

Опис алгоритму

DSS заснований на труднощах обчислення дискретних логарифмів і базується на схемі, яка спочатку була подана Elgamal і Schnorr.

Загальні компоненти групи користувачів

Існує три параметри, які є *відкритими* і можуть бути загальними для великої групи користувачів.

160-бітне просте число q , тобто $2^{159} < q < 2^{160}$.

Просте число p довжиною між 512 і 1024 бітами повинне бути таким, щоб q було дільником $(p - 1)$, тобто $2^{L-1} < p < 2^L$, де $512 < L < 1024$ і $(p - 1)/q$ є цілим:

$$g = h^{(p-1)/q} \bmod p, \text{ де } h \text{ є цілим між } 1 \text{ і } (p - 1).$$

Знаючи ці числа, кожний користувач обирає закритий ключ і створює відкритий ключ.

Закритий ключ відправника

Закритий ключ x повинен бути числом між 1 і $(q - 1)$ і повинен бути обраний випадково або псевдовипадково.

x – випадкове або псевдовипадкове ціле, $0 < x < q$.

Відкритий ключ відправника

Відкритий ключ обчислюється із закритого ключа як $y = g^x \bmod p$. Обчислити y за відомим x досить просто. Однак, маючи відкритий ключ y , обчисленням неможливо визначити x , який є дискретним логарифмом y на основі g .

$$y = g^x \bmod p.$$

Випадкове унікальне число для кожного підпису.

k – випадкове або псевдовипадкове ціле, $0 < k < q$ – унікальне для кожного підписування.

Підписування. Для створення підпису відправник обчислює дві величини r і s , які є функцією від компонентів відкритого ключа (p, q, g) , закритого ключа користувача (x) , ґеш-коду повідомлення $H(M)$ і цілого k , яке повинне бути створене випадково або псевдовипадково і повинне бути унікальним при кожному підписуванні.

$$\begin{aligned} r &= (g^k \bmod p) \bmod q \\ s &= [k^{-1} (H(M) + x^r)] \bmod q. \end{aligned}$$

Підпис = (r, s) .

Перевірка підпису

Одержувач виконує перевірку підпису з використанням наступних формул. Він створює величину v , яка є функцією від компонентів загального відкритого ключа, відкритого ключа відправника і ґеш-коду отриманого повідомлення. Якщо ця величина дорівнює компоненту r у підписі, то підпис вважається дійсним.

$$\begin{aligned} w &= s^{-1} \bmod q, \\ u_1 &= [H(M) w] \bmod q, \\ u_2 &= r w \bmod q, \\ v &= [(g^{u_1} y^{u_2}) \bmod p] \bmod q, \end{aligned}$$

підпис коректний, якщо $v = r$.

Доведемо, що $v = r$ у випадку коректного підпису.

Лема 1. Для будь-якого цілого t , якщо

$$g = h^{(p-1)/q} \bmod p;$$

то $gt \bmod p = gt^{\bmod q} \bmod p$.

За теоремою Ферма якщо h є взаємпростим з p , тоді $h^{p-1} \bmod p = 1$.
Отже, для будь-якого невід'ємного цілого n :

$$g^{nq} \bmod p = (h^{(p-1)/q} \bmod p)^{nq} \bmod p = h^{((p-1)/q) nq} \bmod p = h^{(p-1)n} \bmod p = \\ = ((h^{(p-1)} \bmod p)^n) \bmod p = 1^n \bmod p = 1.$$

Таким чином, для невід'ємних цілих n і z маємо:

$$g^{nq+z} \bmod p = (g^{nq} g^z) \bmod p = ((g^{nq} \bmod p) (g^z \bmod p)) \bmod p = g^z \bmod p.$$

Будь-яке невід'ємне ціле t може бути презентоване єдиним чином як $t = nq + z$, де n і z є невід'ємні цілі та $0 < z < q$. У такий спосіб $z = t \bmod q$.

Лема 2. Для невід'ємних чисел a і b :

$$g^{(a \bmod q + b \bmod q)} \bmod p = g^{(a+b) \bmod q} \bmod p.$$

За лемою 1 маємо:

$$g^{(a \bmod q + b \bmod q)} \bmod p = g^{(a \bmod q + b \bmod q) \bmod q} \bmod p = g^{(a+b) \bmod q} \bmod p.$$

Лема 3. $y^{(rw) \bmod q} \bmod p = g^{(xrw) \bmod q} \bmod p$.

За визначенням $y = gx \bmod p$. Тоді:

$y^{(rw) \bmod q} \bmod p = (gx \bmod p)^{(rw) \bmod q} \bmod p = gx^{((rw) \bmod q)} \bmod p$ – за правилами модульної арифметики:

$$y^{(rw) \bmod q} \bmod p = g^{(x((rw) \bmod q))} \bmod p = g^{(xrw) \bmod q} \bmod p – за лемою 1$$

Лема 4. $((H(M) + xr) w) \bmod q = k$.

За визначенням $s = (k^{-1} (H(M) + xr)) \bmod q$. Крім того, через те що q є простим, будь-яке невід'ємне ціле, менше q , має мультиплікативну інверсію. Тобто $(k^{k-1}) \bmod q = 1$. Тоді:

$$(ks) \bmod q = (k((k^{-1}(H(M) + xr)) \bmod q)) \bmod q = (k(k^{-1}(H(M) + xr))) \bmod q = \\ = ((k^{k-1}) \bmod q) ((H(M) + xr) \bmod q) \bmod q = (H(M) + xr) \bmod q$$

За визначенням $w = s^{-1} \bmod q$, отже, $(ws) \bmod q = 1$. Отже:

$$((H(M) + xr) w) \bmod q = (((H(M) + xr) \bmod q) (w \bmod q)) \bmod q = \\ = (((ks) \bmod q) (w \bmod q)) \bmod q = (kws) \bmod q = \\ = (k \bmod q) ((ws) \bmod q) \bmod q = k \bmod q$$

Через те що $0 < k < q$, то $k \bmod q = k$.

Теорема 2.16. Використовуючи визначення для v і r , доведемо, що $v = r$.

$$\begin{aligned}
 v &= ((gu^1 yu^2) \bmod p) \bmod q = ((g^{(H(M) w)} \bmod q) y^{(rw)} \bmod q) \bmod p) \bmod q = \\
 &= ((g^{(H(M) w)} \bmod q) g^{(xrw)} \bmod q) \bmod p) \bmod q = ((g^{(H(M) w)} \bmod q + (xrw) \bmod q) \bmod p) \bmod q = \\
 &= ((g^{(H(M) w + xrw)} \bmod q) \bmod p) \bmod q = ((g^{w (H(M) + xr)} \bmod q) \bmod p) \bmod q = \\
 &= (g^k \bmod p) \bmod q = r.
 \end{aligned}$$

Приклад. Оберемо $p = 11$, $g = 2$, секретний ключ $x = 8$. Обчислюємо $y = g^x \bmod p = 2^8 \bmod 11 = 256 \bmod 11 = 3$.

Отже, відкритий ключ $y = 3$.

Нехай повідомлення $M = 5$. Оберемо деяке випадкове число $k = 9$.

Переконаємося, що $\text{gsd}(k, p - 1) = 1$. Дійсно, $\text{gsd}(9, 10) = 1$.

Обчислюємо пару чисел r і s :

$$r = g^k \bmod p = 2^9 \bmod 11 = 512 \bmod 11 = 6,$$

$$s = y^k M \bmod p = 3^9 * 5 \bmod 11 = 19683 * 5 \bmod 11 = 9.$$

Одержимо шифртекст $(r, s) = (6, 9)$.

Виконаємо розшифрування цього шифртексту. Обчислюємо повідомлення M , використовуючи секретний ключ x :

$$M = s/r^x \bmod p = 9/6^8 \bmod 11.$$

Вираз $M = 9/6^8 \bmod 11$ можна подати у вигляді:

$$68 * M = 9 \bmod 11;$$

$$\text{або } 1679616 * M = 9 \bmod 11.$$

Вирішуючи дане порівняння, знаходимо $M = 5$.

У реальних схемах шифрування необхідно використовувати в якості модуля p велике ціле просте число, що має у двійковій виставі довжину 512... 1024 бітів.

При програмній реалізації схеми Ель Гамала швидкість її роботи (на SPARC-M) у режимах шифрування і розшифрування при 160-бітовому показнику степеня для різних довжин модуля p визначається значеннями, наведеними в табл. 2.15.

Таблиця 2.15

Швидкості роботи схеми Ель Гамала

Режим роботи	Довжина модуля, бітів		
	512	768	1024
Шифрування	0,33 с	0,80 с	1.09 с
Розшифрування	0,24 с	0,58 с	0,77 с

2.10.3. Комбінований метод шифрування

Головною перевагою криптосистем з відкритим ключем є їх потенційно висока безпека: немає необхідності ні передавати, ні повідомляти будь-кому значення секретних ключів, ні переконуватись в їх дійсності.

У симетричних криптосистемах існує небезпека розкриття секретного ключа під час передачі.

Однак алгоритми, що лежать в основі криптосистем з відкритим ключем, мають наступні недоліки:

генерація нових секретних і відкритих ключів заснована на генерації нових великих простих чисел, а перевірка простоти чисел займає багато процесорного часу;

процедури шифрування і розшифрування, пов'язані зі зведенням у степінь багатозначного числа, досить громіздкі.

Тому швидкодія криптосистем з відкритим ключем звичайно на 2 – 5 порядків разів менша швидкодії симетричних криптосистем з секретним ключем.

Комбінований (гібридний) метод шифрування дозволяє поєднувати переваги високої таємності, надавані асиметричними криптосистемами з відкритим ключем, з перевагами високої швидкості роботи, властивими симетричним криптосистемам із секретним ключем. При такому підході криптосистема з відкритим ключем застосовується для шифрування, передачі і наступного розшифрування тільки секретного ключа симетричної криптосистеми. А симетрична криптосистема застосовується для шифрування і передачі вихідного відкритого тексту. У результаті криптосистема з відкритим ключем не заміняє симетричну криптосистему із секретним ключем, а лише доповнює її, дозволяючи підвищити в цілому захищеність інформації, яка передається. Такий підхід іноді називають *схемою електронного цифрового конверту*.

Якщо користувач А прагне передати зашифроване комбінованим методом повідомлення М користувачу В, то порядок його дій буде такий.

1. Створити (наприклад, згенерувати випадковим чином) симетричний ключ, називаний у цьому методі сеансовим ключем K_s .

2. Зашифрувати повідомлення М на сеансовому ключі K_s .

3. Зашифрувати сеансовий ключ K_s на відкритому ключі K_B користувача В.

4. Передати відкритим каналом зв'язку на адресу користувача В зашифроване повідомлення разом із зашифрованим сеансовим ключем.

Дії користувача В при одержанні зашифрованого повідомлення і зашифрованого сеансового ключа повинні бути зворотними:

5. Розшифрувати на своєму секретному ключі K_B сеансовий ключ K_s .

6. За допомогою отриманого сеансового ключа K_s розшифрувати і прочитати повідомлення М.

При використанні комбінованого методу шифрування можна бути впевненим у тому, що тільки користувач В зможе правильно розшифрувати ключ K_s і прочитати повідомлення М.

Таким чином, при комбінованому методі шифрування застосовуються криптографічні ключі як симетричних, так і асиметричних криптосистем. Очевидно, вибір довжин ключів для кожного типу криптосистеми слід здійснювати таким чином, щоб противнику було однаково важко атакувати будь-який механізм захисту комбінованої криптосистеми.

У табл. 2.16 наведені розповсюджені довжини ключів симетричних і асиметричних криптосистем, для яких труднощі атаки повного перебору приблизно дорівнюють труднощам факторизації відповідних модулів асиметричних криптосистем.

Таблиця 2.16

Довжини ключів для симетричних і асиметричних криптосистем при їх однаковій криптостійкості

Довжина ключа симетричної криптосистеми, біти	Довжина ключа асиметричної криптосистеми, біти
56	384
64	512
80	768
112	1792
128	2304

Комбінований метод допускає можливість виконання процедури автентифікації, тобто перевірки дійсності переданого повідомлення. Для цього користувач А на основі функції ґешування повідомлення і свого секретного ключа K_A за допомогою відомого алгоритму електронному електронному підпису генерує свій підпис і записує її, наприклад, у кінець переданого файла.

Користувач В, прочитавши прийняте повідомлення, може переконатися в дійсності електронного підпису абонента А. Використовуючи той же алгоритм УЕП і результат ґешування прийнятого повідомлення, користувач В перевіряє отриманий підпис. Комбінований метод шифрування є найбільш раціональним, поєднуючи в собі високу швидкодію симетричного шифрування та високу криптостійкість, яка гарантується системами з відкритим ключем.

2.10.4. Керування криптографічними ключами

Керування ключами відіграє фундаментальну роль у криптографії і є основою криптографічних методів і засобів, що забезпечують конфіденційність, автентифікацію об'єктів і джерел повідомлень, цілісність даних а, що також реалізують електроний підпис.

При взаємодії об'єкти комунікації входять між собою в ключові взаємини, об'єкти використовують загальні дані або ключовий матеріал для використання криптографічних методів і засобів. Ключовий матеріал може включати відкриті і секретні ключі, вектори ініціалізації та інші ініціалізуючі величини, додаткові несекретні параметри. Ключові взаємини в комунікаційному устаткуванні виникають між двома сторонами – відправником і одержувачем. При зберіганні може існувати тільки одна сторона, яка бере на зберігання ключі і видає дані в різні моменти часу.

Фахівці ANSI займались розробкою стандартів в області керування ключами для банківських установ ще з кінця 70-х років. У результаті проведених робіт з'явилися такі банківські стандарти керування ключами, як X9.17, X9.24, X9.28, X9.30 і X9.31.

На міжнародному рівні роботи в цьому напрямку здійснюються технічним комітетом з розробки банківських стандартів TC68. Даний комітет розробив такі міжнародні стандарти, як ISO 8732 – керування ключами в оптових банківських операціях, що базується на X9.17, ISO 11568 – керування ключами для роздрібних банківських операцій, ISO 11649 – стандарт, аналогічний X9.28 і ISO 11166 – керування ключами на основі використання несиметричних алгоритмів, що базується на X9.30 і X9.31.

У 1996 році підкомітетом SC27 був розроблений загальний стандарт ISO/IEC 11770. В 1993 році, у ході розробки даного стандарту, виникли серйозні розбіжності між ISO TC68 і ISO/IEC SC27/WG2 щодо несумісності між ISO 11166 і CD-версією ISO/IEC 11770-3. Ці розбіжності були

дозволені через створення двох груп “угоди за розбіжностями”. В результаті робота ISO/IEC JTC1/SC27 була в основному сфальсифікована на механізмах запровадження в дію ключів (key establishment).

У даному підрозділі будуть розглянуті два міжнародних стандарти – ISO/IEC 11770 і ISO 11166.

Міжнародний стандарт ISO/IEC 11770 складається із трьох частин.

ISO/IEC 11770-1:1996 – *Інформаційні технології. – Методи забезпечення безпеки. – Керування ключами. – Ч. 1. Основні положення керування ключами.*

ISO/IEC 11770-2:1996 – *Інформаційні технології. – Методи забезпечення безпеки. – Керування ключами. – Ч. 2. Механізми, що використовують методи симетричної криптографії.*

ISO/IEC 11770-3:1998 – *Інформаційні технології. – Методи забезпечення безпеки. – Керування ключами. – Ч. 3. Механізми, що використовують методи несиметричної криптографії.*

У першій частині стандарту вводяться основні визначення і поняття в сфері керування ключами, утримуються загальні та практичні рекомендації з керування ключами, які пов’язані з рекомендаціями стандарту ISO/IEC 10181. У другій частині розглядаються механізми розподілу ключів на основі використання методів симетричної криптографії. Нарешті, в третій частині представлені механізми розподілу ключів (як для симетричних, так і для несиметричних алгоритмів) на основі використання методів несиметричної криптографії. Зазначимо, що область застосування стандарту ISO/IEC 11770 досить широка, вона охоплює широкий діапазон різних інформаційних систем. У зв’язку із цим даний документ носить більш загальний, фундаментальний характер, ніж інші стандарти в цій галузі.

Стандарт ISO 11166 є прикладним стандартом і його положення орієнтовані на банківське застосування. Він був підготовлений технічним комітетом ISO/TC 68 “Банківська справа і фінансова послуги”. Стандарт складається із двох частин. Перша частина ISO 11166-1 – *Керування ключами на основі несиметричних алгоритмів. – Ч.1. Принципи, процедури і формати* – визначає основні принципи керування ключами, описує процедури для здійснення безпечного керування ключами, вводить термінологію в області керування ключами, визначає формати повідомлень у протоколах, які використовуються при керуванні ключами.

Друга частина ISO 11166-1 – *Керування ключами на основі*

несиметричних алгоритмів. Ч. 2. Затверджений алгоритм на основі криптосистеми RSA – описує алгоритм RSA, дозволений для застосування в протоколах керування ключами.

Основні положення керування ключами (ISO/IEC 11770-1)

Перша частина стандарту ISO/IEC 11770-1 присвячена розгляду наступних основних питань:

- мета, погрози і політика керування ключами;
- основні поняття і визначення керування ключами;
- життєвий цикл ключів і функції керування ключами;
- класифікація ключів;
- моделі розподілу ключів і методи захисту ключів;
- сертифікація ключів;

інформаційні об'єкти керування ключами (Key Management Information Objects), структури даних, що містять ключ (ключі) і пов'язану з ними інформацію (такі структури також визначені в ASN.1), життєвий цикл сертифіката керування ключами.

Під *керуванням ключами* розуміють множину методів і процедур, що здійснюють встановлення і керування ключовими взаєминами між авторизованими об'єктами. Керування ключами включає методи і процедури, які підтримують:

- ініціалізацію системних користувачів усередині домену безпеки;
- генерацію, розподіл й інсталяцію ключового матеріалу;
- керування і контроль використання ключового матеріалу;
- відновлення, анулювання і знищення ключового матеріалу;
- зберігання, резервування/відновлення та архівування ключового матеріалу.

Метою керування ключами є запобігання наступних основних погроз:

- компрометація конфіденційності секретних ключів;
- компрометація автентичності секретних і відкритих ключів;
- неавторизоване використання секретних і відкритих ключів.

Керування ключами здійснюється на основі спеціальної політики безпеки, яка прямо або побічно визначає погрози безпеки. Політика також визначає:

- заходи і процедури, що впливають із застосування технічних і

організаційних аспектів керування ключами як автоматичного, так і ручного;

права, обов'язки і відповідальність кожної зі сторін, що брали участь у керуванні ключами;

тип і зміст записів, внесених у контрольні журнали (журнали контролю безпеки) щодо настання яких-небудь подій, пов'язаних з безпекою керування ключами.

Для подальшого розгляду положень стандарту наведемо деякі найбільш важливі визначення, які вводяться ISO/IEC 11770-1.

Орган (адміністратор) сертифікації (certification authority) – довірчий центр створення й закріплення сертифікатів відкритих ключів. Додатково орган сертифікації може створювати і призначати ключі об'єктам.

Неявна автентифікація ключа об'єкта А – завірення одного об'єкта А в тому, що тільки інший ідентифікований об'єкт може мати правильний (відповідний) ключ.

Ключ – послідовність символів, які управляють процесом криптографічного перетворення.

Центр розподілу ключів (ЦРК) (key distribution centre) – довірна сторона, яка генерує або здобуває, а потім розподіляє ключі об'єктам, що мають загальний ключ з ЦРК.

Підтвердження приймання ключа (key confirmation) – гарантії одного об'єкта того, що інший ідентифікований об'єкт володіє правильним ключем.

Контроль над ключем (key control) – здатність вибрати ключ або параметри, які використовуються при обчисленні ключів.

Центр передачі ключів (ЦПК) (key translation center) – довірна сторона передачі ключів між об'єктами, кожний з яких має загальний ключ з ЦПК.

Запровадження в дію ключа (key establishment) – процес, що робить доступним загальний секретний ключ одному або більше об'єктам. Запровадження в дію ключа включає угоду про ключ і доставку ключа.

Угода про ключ (key agreement) – процес уведення в дію загального секретного ключа між об'єктами таким шляхом, що ніхто з них не може визначити значення цього ключа. Це значить, що жоден з об'єктів не управляє ключем.

Доставка ключа (key transport) – процес (можливо захищений) передачі ключа від одного об'єкта іншому.

Класифікація ключів, ієрархія ключів і криптоперіод

Стандарт ISO/IEC 10770 здійснює класифікацію ключів за наступними класифікаційними ознаками (рис. 2.38).

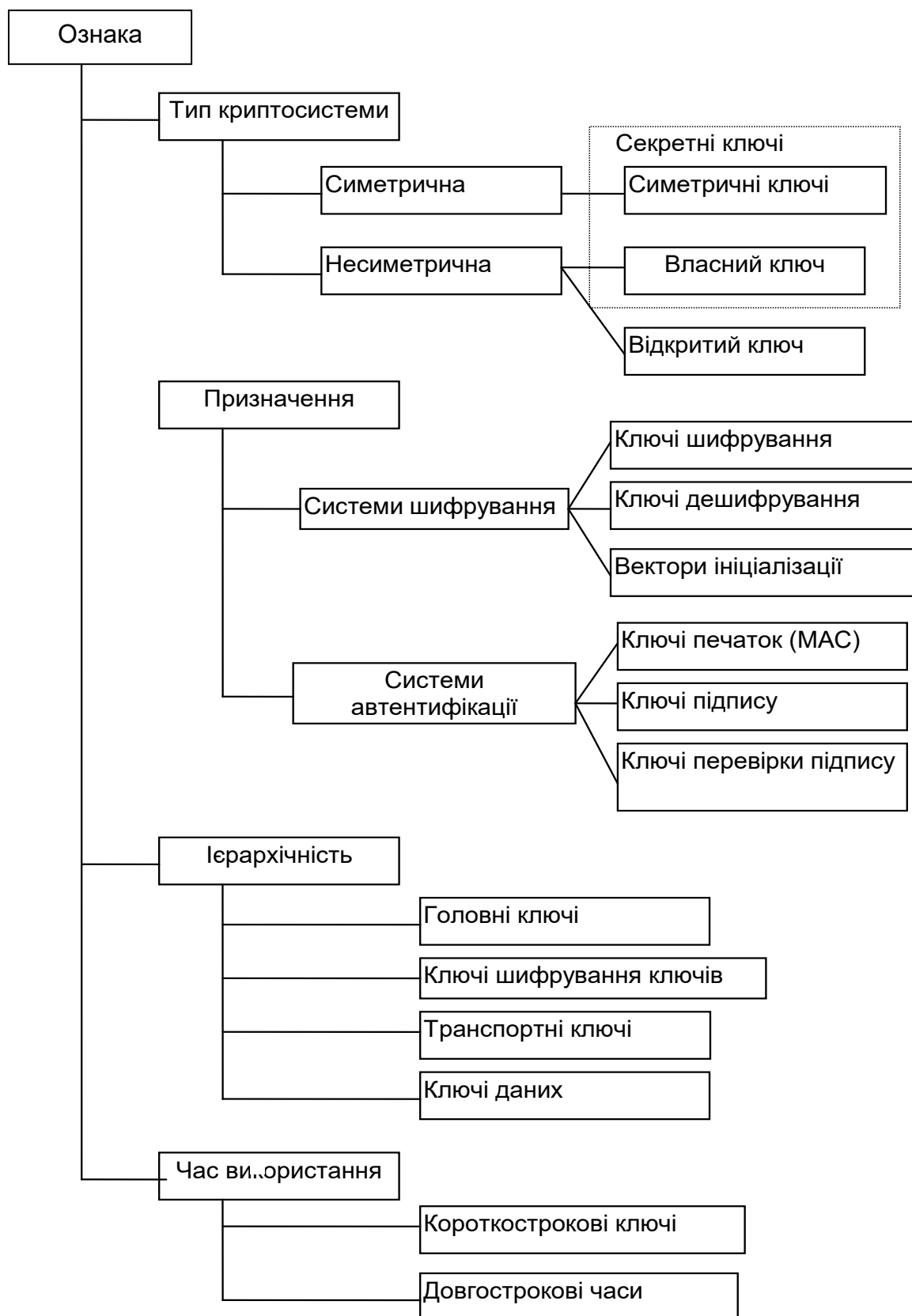


Рис. 2.38. Класифікація ключів

Ключі розділяються залежно від того, у якій криптографічній системі вони використовуються – симетричній або несиметричній. Як відомо, в симетричних криптографічних системах здійснюється два перетворення: одне на передавальній стороні, інше – на прийомній стороні, які використовують той самий ключ.

У несиметричних криптосистемах одне перетворення (відкрите) виконується на одному (відкритому) ключі, а закрите перетворення виконується на іншому (особистому) ключі. У зв'язку з цим усі ключі можна розділити на наступні типи.

Особистий ключ (private key) – ключ із пари несиметричних ключів об'єкта, який повинен використовуватися тільки даним об'єктом.

Відкритий ключ (public key) – ключ із пари несиметричних ключів об'єкта, який зроблений загальнодоступним.

Особистий і відкритий ключі використовуються в несиметричних криптографічних системах.

Симетричний ключ (symmetric key) – ключ, що використовується у симетричних (одноключових) криптографічних системах.

Секретний ключ (secret key) – ключ, який тримається в секреті і використовується тільки певною множиною об'єктів. Секретними ключами звичайно є симетричні й особисті ключі.

Як симетрична, так і несиметрична криптосистема в загальному випадку реалізують послуги конфіденційності (системи шифрування) і автентифікації об'єктів і повідомлень (системи автентифікації). В цьому випадку можна розглядати ключі, які потрібні для реалізації завдань цих систем.

Системи автентифікації містять у собі механізми забезпечення цілісності повідомлень (MAC-коди, MDC-коди та ін.) та електроні підписи.

Дані механізми допускають виконання таких дій, як:

постановка електронної печатки (seal), тобто обчислення симетричної криптографічної контрольної суми (MAC-код);

підпис – обчислення електронного підпису;

перевірка електронної печатки – повторне обчислення криптографічної контрольної функції;

перевірка підпису – перевірка несиметричного електронного підпису.

Таким чином, системи автентифікації використовують три основних типи ключів:

ключ печатки (sealing key) – симетричний секретний ключ;

ключ підпису (signature key) – особистий ключ;

ключ перевірки (verification key) – відкритий ключ (для несиметричних систем), або симетричний секретний ключ (для симетричних систем).

Системи шифрування включають симетричні й несиметричні системи і основними типами ключів є:

ключ шифрування – секретний або відкритий ключ;

ключ розшифрування – секретний або особистий ключ.

Завдання забезпечення конфіденційності можна розбити на дві підзадачі, за типом підлягаючої закриттю інформації – забезпечення конфіденційності даних користувача й забезпечення конфіденційності ключового матеріалу. У зв'язку з цим ключі звичайно організовуються в ієрархію ключів (key hierarchies).

На верхньому рівні ієрархії розташовується *головний* або *майстер-ключ* (master key). Головні ключі не можуть бути криптографічно захищені. Вони захищаються прийняттям відповідних організаційних заходів при їх розподілі, зберіганні, запровадженні в дію. Розподіл здійснюється ручним способом через довірених кур'єрів або спеціальною поштою, чи шляхом безпосередньої інсталяції в криптографічне встаткування при його виготовленні або поставці. Захист у цьому випадку здійснюється через фізичну або електричну ізоляцію обладнання зберігання ключа.

На наступному рівні ієрархії перебувають *ключі шифрування ключів* (key-encrypting keys). Це симетричні або відкриті ключі шифрування, які використовуються для зберігання і передачі інших ключів. Якщо такі ключі застосовуються в протоколах доставки ключів, то такі ключі ще називаються *транспортними ключами* (key-transport keys).

Нарешті на нижньому рівні ієрархії розташовуються *ключі даних* (data keys), які застосовуються для реалізації різних криптографічних операцій над даними користувача. Найчастіше до таких ключів відносяться симетричні короточасні ключі. До таких ключів можна віднести і ключі генерації електронного підпису, які найчастіше є довгочасними ключами.

Для захисту ключів одного рівня ієрархії можуть використовуватися ключі тільки вищого рівня ієрархії. Безпосередньо для реалізації послуг забезпечення безпеки даних використовуються ключі тільки найнижчого рівня ієрархії. Такий підхід дозволяє обмежити використання конкретного ключа, тим самим зменшити ризик розкриття ключа і затруднити проведення криптоаналітичних атак. Наприклад, компрометація одного ключа даних (тобто ключа на нижньому рівні ієрархії) приведе до компрометації тільки захищених цим ключем даних. Розкриття головного

ключа потенційно дає можливість розкрити або маніпулювати всіма захищеними цим ключем ключами (тобто всіма ключами ієрархії). Отже, бажано мінімізувати доступ до цього ключа. Можливо побудувати систему таким чином, щоб жоден з користувачів не мав доступу до значення головного ключа.

Наступною ознакою класифікації ключів є строк їх дії або криптоперіод.

Криптоперіод – це певний період часу, протягом якого конкретний криптографічний ключ затверджується до використання, або протягом якого криптографічні ключі залишаються в силі для даної системи й придатні для використання авторизованими сторонами.

Криптоперіод служить для того, щоб:

- обмежити кількість інформації, пов'язаної із ключами, доступної для криптоаналізу;

- зменшити ризик порушення безпеки у випадку компрометації ключа;

- обмежити використання конкретних методів і технологій для ефективної оцінки значення ключа протягом його життєвого циклу;

- обмежити доступний обчислювальний час для проведення інтенсивних криптоаналітичних атак (це стосується тільки короткочасних ключів).

За часовою ознакою ключі можна розділити на два класи – довгострокові і короткострокові.

До *довгострокових ключів* відносяться головні ключі, а також можуть бути віднесені ключі шифрування ключів і ключі, використовувані в схемах угоди про ключі.

Короткостроковими ключами є ключі, що вводяться в дію транспортними ключами або шляхом застосування схем угоди про ключі. Найчастіше короткочасні ключі використовуються як ключі даних або сеансових ключів (*session key*), використовуваних на один сеанс зв'язку.

З погляду застосування короткострокові ключі звичайно використовують у комунікаційних додатках, тоді як довгочасні ключі використовуються в додатках, які орієнтовані на зберігання даних, а також для захисту короткострокових ключів.

Терміни коротко- і довгостроковий відносяться тільки до безпосереднього використання ключа для реалізації криптографічної операції, виконуваної авторизованими сторонами. Ці терміни не можна віднести до більш загальної системної характеристики ключа, а саме – до життєвого циклу ключа. Так ключ шифрування, використовуваний для

одного сеансу зв'язку, тобто короточасний ключ, може вимагати забезпечення такої захищеності, щоб протистояти криптонападу досить довгий період. З іншого боку, якщо сигнатура буде перевірена негайно й у подальшій перевірці вона не матиме потреби, то і ключ підпису може бути захищений тільки на відносно короткий строк.

Життєвий цикл ключа, функції керування ключами, життєвий цикл керування ключами

Криптографічний ключ може перебувати в різних станах, які визначають його життєвий цикл. Стандарт ISO/IEC 11770 розрізняє основні перехідні стани. Основними станами є:

стан очікування (черговий стан) (pending active) – стан, у якому ключ не використовується для звичайних операцій;

активний стан (active) – стан, у якому ключ використовується для криптографічної обробки інформації;

постактивний стан (post active) – стан, у якому ключ може використовуватися тільки для дешифрування або верифікації. Якщо буде потреба використання ключа за призначенням, він переводиться з постактивного стану в активний. Ключ, про який відомо, що він скомпрометований, повинен бути негайно переведений у постактивний стан.

При переході з одного основного стану в інший ключ може перебувати в одному з перехідних станів (transition). Такими перехідними станами є:

генерація – процес генерації ключа, у ході якого відповідно до запропонованих правил генерується ключ;

активізація (activation) – процес або сукупність процесів, у ході яких ключ робиться придатним для використання, тобто переводиться зі стану очікування в активний стан;

деактивізація (deactivation) – процес або сукупність процесів, що обмежують використання ключа, наприклад, через закінчення терміну дії ключа або його анулювання, що і переводять ключ із активного в постактивний стан;

реактивізація (reactivation) – процес або сукупність процесів, які дозволяють перевести ключ з постактивного в активний стан для повторного використання;

знищення (destruction) – завершує життєвий цикл ключа.

На рис. 2.39 схематично поданий взаємозв'язок основних і перехідних станів.

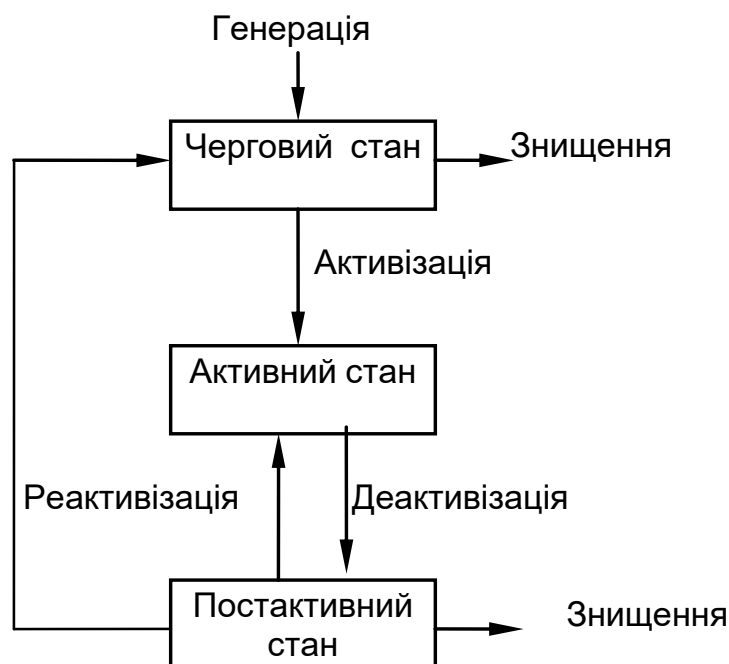


Рис. 2.39. **Взаємозв'язок основних і перехідних станів**

Життєвий цикл ключа підтримується одинадцятьма функціями керування ключами (key management services). Коротко охарактеризуємо ці функції [30].

1. *Генерація ключа* – забезпечує генерацію криптографічного ключа із заданими властивостями для конкретних криптографічних додатків.

2. *Реєстрація ключа* – зв'язує ключ із об'єктом (звичайно тільки відповідні секретні ключі). Об'єкт, що бажає зареєструвати ключ, контактує з адміністратором реєстрації.

3. *Створення сертифіката ключа* – гарантує взаємозв'язок відкритого ключа з об'єктом і забезпечується вповноваженим органом сертифікації (certification authority), який генерує відповідні сертифікати.

4. *Розподіл ключа* (distribute key) – множина процедур безпечного (секретного) забезпечення ключами і пов'язаної з ними інформації вповноважених об'єктів.

5. *Інсталяція ключа* (install-key) – розміщення ключа в устаткуванні керування ключами безпечним чином і готовим до використання.

6. *Зберігання ключа* (store-key) – безпечне зберігання ключів для подальшого використання або відновлення ключа для повторного використання.

7. *Похідна ключа* (derive-key) – формування великої кількості ключів, які називаються похідними ключами, шляхом комбінування секретного вихідного ключового матеріалу, називаного *ключем деривації*, з несекретними даними на основі використання необоротних процесів.

8. *Архівування ключа* (archive-key) – забезпечення безпечного зберігання ключів після їх використання. Дана функція використовує функцію зберігання ключа й інші засоби, наприклад, зовнішні сховища.

9. *Скасування (анулювання) ключа* (revoke-key) (відоме як видалення ключа (delete key)) – у випадках компрометації ключа функція забезпечує безпечну деактивізацію ключа.

10. *Дереєстрація ключа* (deregister-key) – функція реалізується повноважним органом реєстрації, який забирає запис про те, що даний секретний ключ пов'язаний з об'єктом.

11. *Знищення ключа* (destroy-key) – забезпечує безпечне знищення ключів, у яких минув термін дії. Ця функція включає і знищення всіх архівних копій ключа.

На рис 2.40 поданий життєвий цикл керування ключами і його взаємозв'язок з основними станами ключів.

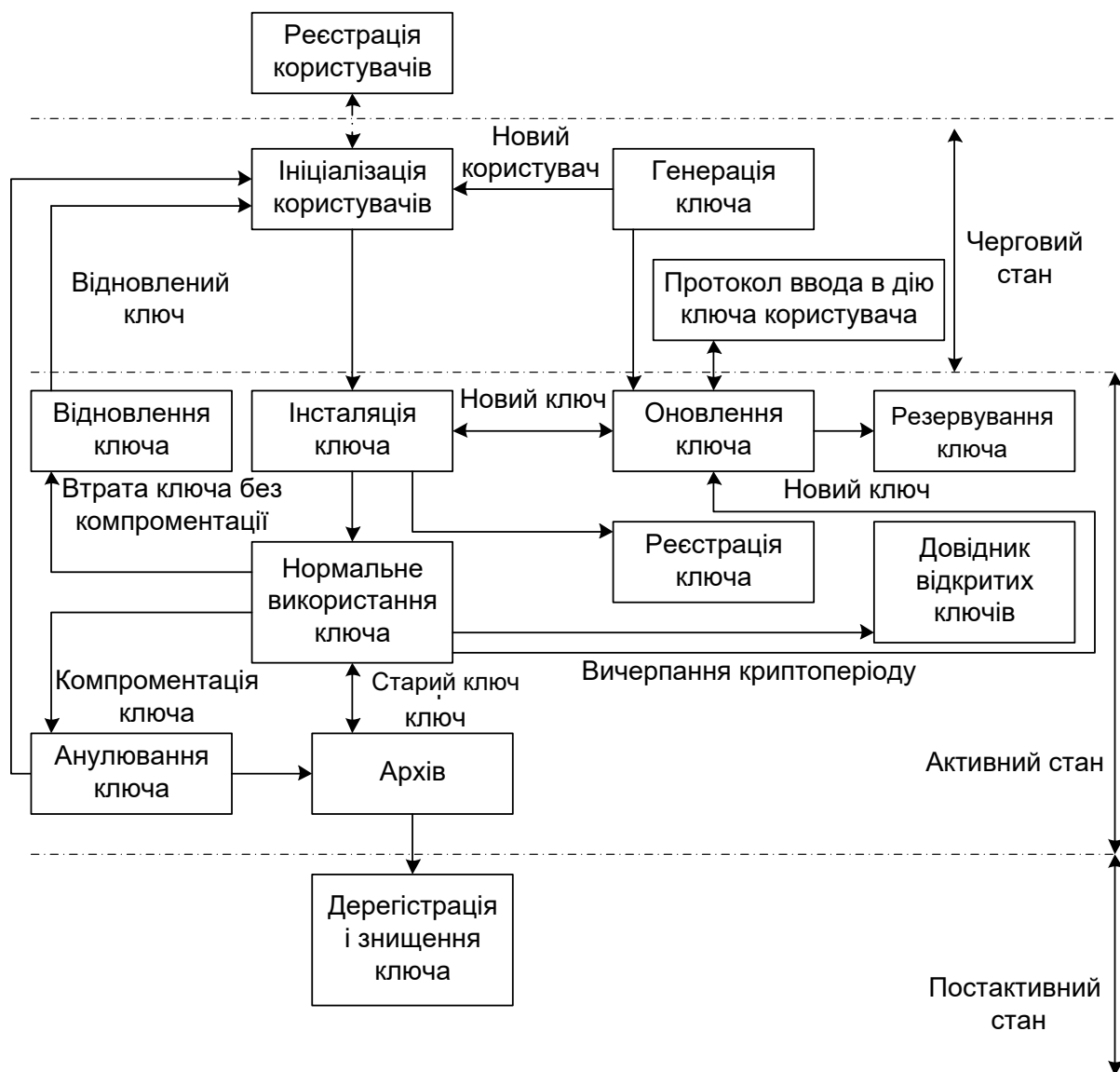


Рис. 2.40. Життєвий цикл керування ключами

Послідовність станів, у яких може перебувати ключовий матеріал, функції керування ключами, інші функції і процеси, що протікають протягом усього життєвого циклу ключів, утворюють життєвий цикл керування ключами.

Життєвий цикл керування ключами містить у собі наступні основні етапи й процеси [1; 2].

1. *Реєстрація користувача* – процес, у ході якого об'єкт стає авторизованим членом домену безпеки. Це допускає придбання (створення) і обмін первинним ключовим матеріалом між користувачем і доменом безпеки, наприклад, поділюваним паролем або персональним ідентифікаційним номером (PIN). Всі дії в ході реєстрації здійснюються безпечними одноразовими способами, наприклад, через особистий обмін, замовлення поштою, довіреним кур'єром.

2. *Ініціалізація користувача* – процес, у ході якого об'єкт ініціалізує свій криптографічний додаток (наприклад, інсталує та ініціалізує програмне або апаратне забезпечення), включаючи використання або інсталяцію первинного ключового матеріалу, який отриманий під час реєстрації користувача.

3. *Генерація ключа*. Генерація криптографічних ключів обов'язково повинна включати заходи, спрямовані на забезпечення відповідних властивостей ключа і його випадковості. Ці властивості забезпечуються шляхом використання методів генерації випадкових або псевдовипадкових чисел. Об'єкт може генерувати собі ключі або самостійно, або запитувати їх у довірчої сторони.

4. *Інсталяція ключа*. Ключовий матеріал інсталується в програмне або апаратне забезпечення об'єкта за допомогою різних методів, наприклад, ручне введення пароля або PIN, передача даних з використанням диска, постійного запам'ятовувального пристрою, чіп-карти або інших апаратних обладнань (наприклад, завантажника ключа). Первинний ключовий матеріал може служити для організації безпечного on-line сеансу зв'язку, за допомогою якого вводяться в дію основні робочі криптографічні ключі. Надалі, відновлення новим ключовим матеріалом замість використовуваного повинне здійснюватися за допомогою безпечних on-line-методів відновлення.

5. *Реєстрація ключа* здійснюється в тісному зв'язку з інсталяцією ключа і полягає в тому, що ключовий матеріал офіційно заноситься під унікальним іменем об'єкта в закриту базу ключів адміністратором реєстрації. Для відкритих ключів адміністратор сертифікації створює

сертифікати відкритих ключів, які виступають у ролі гарантів дійсності й цілісності ключа. Сертифікати містяться в довідниках відкритих ключів і є загальнодоступними.

6. Нормальне використання. На даному етапі ключі перебувають в оперативній доступності для стандартних криптографічних додатків, включаючи контроль використання криптографічних ключів. За нормальних умов функціонування системи в період нормального використання ключів збігається із криптоперіодом ключів. У несиметричних криптосистемах ключі з однієї пари можуть перебувати на різних етапах свого існування. Наприклад, у деякий момент часу відкритий ключ шифрування може вважатися недійсним, у той час як закритий особистий ключ залишається в активному стані і може використовуватися для розшифрування.

7. Резервування ключа становить резервування ключового матеріалу в незалежному, безпечному сховищі з метою здійснення, якщо буде потреба, відновлення ключа. Резервні копії ключа в основному відправляються на короткострокове зберігання під час нормального використання ключа.

8. Відновлення ключа. Наприкінці криптоперіоду оперативний ключовий матеріал замінюється новим. Це допускає використання комбінації функції генерації ключів, вироблення або установки нових ключів, реалізації двосторонніх протоколів запровадження в дію ключів або організації зв'язку із залученням довірчої третьої сторони. Для відкритих ключів

відновлення й реєстрація нових ключів звичайно допускає реалізацію безпечних комунікаційних протоколів за участю адміністратора сертифікації.

9. Архівування ключового матеріалу, який надалі не буде використовуватися для здійснення криптографічних операцій, здійснюється з метою забезпечення можливості пошуку ключа при виникненні особливих умов, наприклад, вирішення спорів, включаючи реалізацію функції причетності. Архівування допускає довгострокове автономне зберігання ключа, який виводиться при цьому в постактивний стан.

10. Перед знищенням ключа здійснюється його *дереєстрація*, тобто видалення відповідного запису з довідника, у результаті чого знищується взаємозв'язок значення конкретного ключа з об'єктом. Якщо здійснюється знищення секретного ключа, необхідно забезпечити безпечне і надійне знищення всіх його слідів (залишків).

11. *Відновлення ключа* здійснюється у випадку, якщо ключовий матеріал був загублений у результаті не примусової (ненавмисної) компрометації (збої, несправність устаткування, забування (втрата) пароля). У цьому випадку використовуються резервні копії ключа.

12. *Видалення (анулювання) ключа* має на увазі виведення ключа з активного стану в постактивний в результаті, наприклад, його компрометації. При цьому безпосередньо ключ не знищується. Для відкритих ключів у цьому випадку здійснюється видалення (анулювання) сертифіката.

Представлений життєвий цикл керування ключами є найбільш загальним і більш застосовуваним до несиметричних криптосистем. У симетричних криптосистемах керування ключами в загальному випадку менш складне. Так сеансові ключі можуть не реєструватися, не резервуватися, не віддалятися й не архівуватися.

Моделі керування ключами

Стандарт розглядає три варіанти взаємозв'язку між об'єктами, для яких необхідна підтримка керування ключами:

безпосередній зв'язок між об'єктами;

зв'язок між об'єктами, що належать тому самому домену безпеки, через третю сторону;

зв'язок між об'єктами, що належать різним доменам безпеки.

У зв'язку із цим стандарт уводить наступні моделі керування ключами:

модель розподілу ключів між об'єктами, що безпосередньо зв'язуються;

модель розподілу ключів усередині одного домену безпеки;

модель розподілу ключів між доменами безпеки.

Перша модель розглядається в тому випадку, коли два об'єкти комунікації встановлюють ключ без прямої off-line допомоги довірчої третьої сторони (наприклад, органу сертифікації, ЦРК або ЦПК). Такий розподіл ключів можливий при використанні закритих каналів зв'язку з використанням уже існуючих секретних ключів або через взаємний автентичний обмін копіями відкритих ключів об'єктів.

При розподілі ключів усередині одного домену безпеки для встановлення ключів можливе використання методів як симетричної, так і несиметричної криптографії. У першому випадку здійснюється розподіл безлічі сертифікатів відкритих ключів із залученням адміністратора сертифікації. Для того щоб одержати сертифікат відкритого ключа іншого

об'єкта, обидва об'єкти повинні зв'язатись зі своїм адміністратором безпеки. Проте не виключається можливість і безпосереднього обміну сертифікатами.

У випадку використання методів симетричної криптографії розподіл ключів здійснюється із залученням ЦРК або ЦПК, тобто реалізується централізоване керування ключами.

На рис. 2.41 подані варіанти використання ЦПК усередині одного домену безпеки. Нехай ЦПК має загальний секретний (симетричний) ключ як з об'єктом А, так і з об'єктом В. У схемі передаються наступні повідомлення.

ЦПК одержує ключ від А, який зашифрований на загальному ключі $K_{AЦПК}$ об'єкта А і ЦПК. Центр передачі ключів дешифрує ключ і перешифрує його, використовуючи секретний загальний ключ $K_{ВЦПК}$ об'єкта В і ЦПК.

Далі можна діяти за одним з наступних варіантів.

Перший варіант (рис 2.41(а)).

Перешифрований ключ ЦПК відправляє об'єкту А.

Об'єкт А, у свою чергу, посилає ключ об'єкту В.

Другий варіант (рис 2.41(б)).

ЦПК посилає перешифрований ключ безпосередньо об'єкту В.

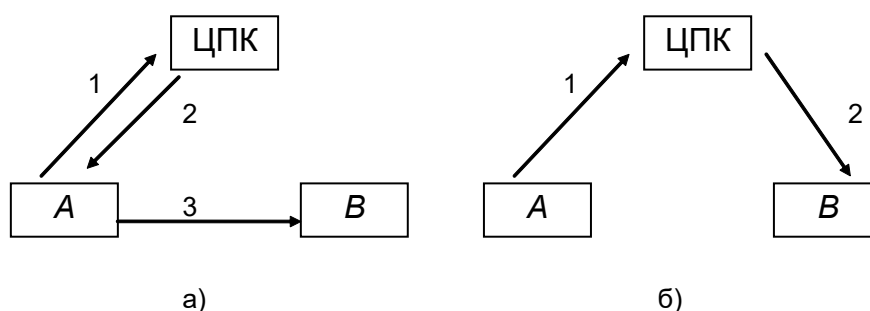


Рис. 2.41. Розподіл ключів через ЦПК

На рис. 2.42 подані варіанти використання ЦРК усередині одного домену безпеки.

Нехай ЦРК має загальний секретний ключ і з А, і з В. Крім того, нехай ЦРК може зв'язатись тільки з об'єктом А (рис. 2.42а). Об'єкт А в цьому випадку є відповідальним за “пересилання” ключа об'єкту В. Порядок обміну повідомленнями має такий вигляд.

1. Об'єкт А запитує ЦРК згенерувати і розподілити секретний ключ

K_{AB} , який буде загальним для A і B.

2. ЦРК посилає дві копії знову згенерованого секретного ключа об'єкту A: одна копія зашифрована на загальному ключі об'єкта A і ЦРК, а інша – на загальному ключі об'єкта B і ЦРК.

3. Об'єкт A пересилає другу зашифровану копію об'єкту B.

На рис. 2.42 б поданий варіант розподілу ключів, коли ЦРК має зв'язок з обома об'єктами. Тоді схема має вигляд.

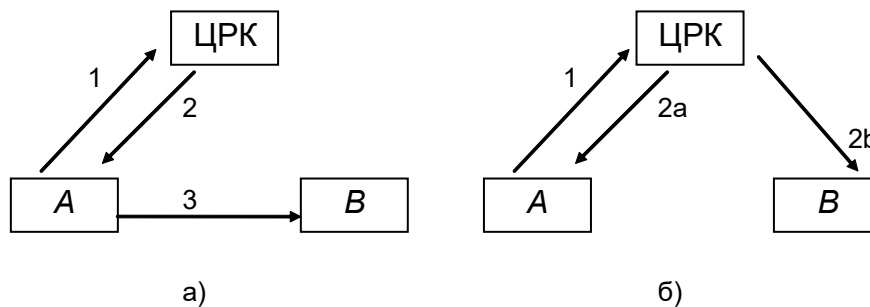


Рис. 2.42. Використання ЦРК

1. Об'єкт A запитує ЦРК згенерувати і розподілити секретний ключ, який буде загальним для A і B.

2а. ЦРК посилає знову згенерований секретний ключ об'єкту A, зашифрований на загальному з об'єктом A секретному ключі.

2б. ЦРК посилає згенерований секретний ключ об'єкту B, зашифрований на загальному з об'єктом B секретному ключі.

Нарешті розглянемо випадок, коли два об'єкти A і B, які належать різним доменам безпеки (D_A і D_B) і довіряють тільки своїм адміністраторам безпеки T_A і T_B , бажають установити між собою секретний зв'язок або, наприклад, об'єкт A бажає одержати доступ до ресурсів домену безпеки D_B .

Нагадаємо, що під *доменом безпеки* розуміють систему під керуванням одного адміністратора безпеки, якому довіряють усі об'єкти, що належать даній системі. На всі об'єкти домену безпеки поширюється єдина політика безпеки, побічно або прямо обумовлена адміністратором безпеки даного домену.

У такій ситуації необхідно розглянути два різних типи завдань:

об'єкт A бажає одержати сертифікат відкритого ключа об'єкта B, який потім буде використаний для встановлення загального ключа K_{AB} ;

об'єкти A і B бажають установити загальний секретний ключ K_{AB} .

Кожний із цих варіантів здійснюється із залученням адміністраторів безпеки T_A і T_B , які мають у своєму розпорядженні або надійні відкриті ключі, або загальний секретний ключ.

Якщо T_A і T_B мають між собою безпечний канал зв'язку, тобто довіряють один одному, то здійснюється попарний обмін ключами між об'єктами (A, T_A) , (T_A, T_B) і (T_B, B) , що дозволяє організувати взаємозв'язок між абонентами (A, B) . Якщо ж адміністратори безпеки не мають між собою довірчого зв'язку, то для організації взаємодії об'єктів різних доменів безпеки необхідно залучати третю довірчу сторону T_C , якій довіряють і T_A і T_B .

На рис. 2.43 подані два можливі варіанти взаємодії між об'єктами. Зазначимо, що можливі й інші схеми обміну повідомленнями.

Перший варіант опирається на використання методів несиметричної криптографії. Довірчість при одержанні відкритого ключа досягається шляхом застосування різних методів забезпечення автентифікації джерела повідомлення, наприклад, електронного підпису або коду дійсності повідомлення.

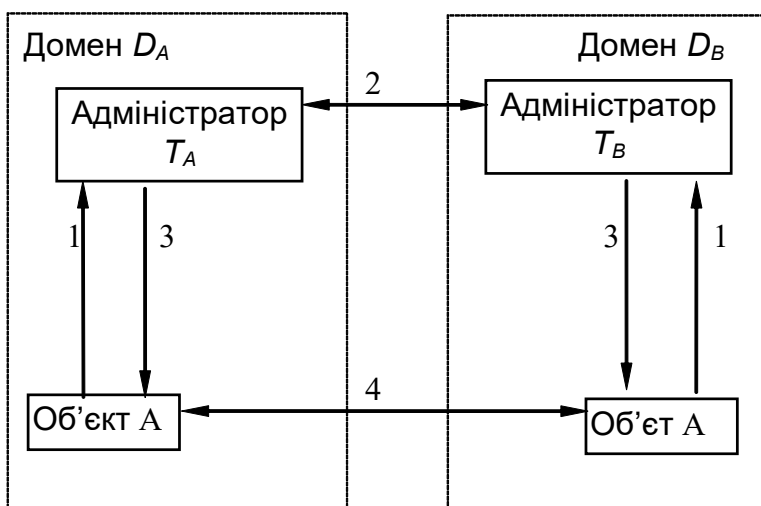


Рис. 2.43. Керування ключами між об'єктами, що належать різним доменам безпеки

Порядок дії сторін виглядає так:

1. Об'єкти A і B запитують у своїх адміністраторів T_A і T_B копії сертифікатів відкритих ключів один одного.

2. Адміністратори здійснюють обмін сертифікатами відкритих ключів об'єктів A і B .

3. Кожний адміністратор на відповідних відкритих ключах пересилає сертифікати об'єктам A і B .

4. Об'єкти A і B установлюють прямий секретний зв'язок між собою.

Другий варіант орієнтований на використання методів симетричної криптографії. У цьому випадку довірчість досягається через використання автентичних методів запровадження в дію ключів. Порядок дій об'єктів:

1. Об'єкти А і В запитують у своїх адміністраторів симетричний ключ, який буде встановлений між ними.

2. Адміністратори безпеки T_A і T_B узгоджують секретний строковий ключ K_{AB} .

3. Секретний ключ K_{AB} розподіляється об'єктам А і В з використанням загальних ключів кожного об'єкта зі своїм адміністратором.

4. Об'єкти А і В установлюють прямий секретний зв'язок між собою на ключі K_{AB} .

У цьому випадку, з погляду об'єктів А і В, об'єднання адміністраторів T_A , T_B і довірчий зв'язок між ними може розглядатися як єдиний адміністратор, за допомогою якого А зв'язується з В і який виступає в ролі ЦПК або ЦРК.

Сертифікація ключів

Стандартами керування ключами сертифікація ключів розглядається як основний засіб забезпечення автентифікації ключів. В ISO/IEC 11770-1 обговорюються наступні питання сертифікації ключів:

роль і місце органу сертифікації (Certification Authority);

модель процесу сертифікації;

розподіл, використання та анулювання сертифікатів.

Сертифікат відкритого ключа (public key certificate) – список даних, пов'язаних з окремим користувачем, включаючи відкритий ключ (або ключі) цього користувача, підписаний органом сертифікації. Таким чином, сертифікат складається із двох полів – поля даних і поля підпису. Поле даних містить, як мінімум, відкритий ключ користувача й ідентифікаційну інформацію користувача (наприклад, ідентифікатор користувача). Поле підпису містить підпис адміністратора сертифікації, яка є поручництвом за автентичність відкритого ключа користувача. Сертифікат також може містити й іншу додаткову інформацію, наприклад, із вказівкою того, яким чином може бути використаний той або інший ключ. Кожний користувач повинен бути приписаний до конкретного адміністратора сертифікації і мати довірену копію ключа перевірки підпису цього адміністратора.

Для виконання своїх функцій адміністратор повинен мати свою ключову пару формування/перевірки підпису. Відкритий ключ із цієї пари

доступний для всіх зареєстрованих авторизованих користувачів системи і дозволяє всім користувачам системи перевірити автентичність відкритих ключів у будь-якому сертифікаті адміністратора. Дійсність відкритого ключа АС може бути забезпечена як криптографічними, так і не криптографічними методами, наприклад, може бути використана замовлена пошта, кур'єр або особистий контакт. Кожного разу забезпечення дійсності, на відміну від конфіденційності, є обов'язковим.

Стандарт визначає коло відповідальності адміністратора (органу) сертифікації. Зокрема АС відповідає за:

- визначення об'єктів, інформація і відкриті ключі повинні бути включені в сертифікат;

- якість власної ключової пари, яка використовується для генерації сертифіката;

- безпека процесу генерації сертифіката й особистого ключа, який використовується в процесі генерації сертифікатів.

На сьогодні існує кілька стандартів, що визначають структуру і зміст сертифіката. Найбільш важливими з них є рекомендації ITU-T X.509 – 1993 і відповідний міжнародний стандарт ISO/IEC 9594-8:1995. Відносно недавно в ці стандарти були внесені виправлення, у результаті чого з'явилась так звана третя версія формату сертифіката X.509.

Звичайно сертифікат містить у собі наступну інформацію:

- ім'я користувача;

- дату закінчення терміну дії сертифіката і ключа (період повноважень);

- порядковий номер або ідентифікатор ключа, що ідентифікують сертифікат або ключ;

- один або більше відкритих ключів, що належать користувачу;

- алгоритм присвоєння ідентифікатора (ідентифікаторів) відкритому ключу (ключам);

- інформація, що стосується політики безпеки, під керуванням якої був створений сертифікат;

- додаткова інформація про користувача (наприклад, адреса або мережна адреса);

- додаткова інформація про ключ (наприклад, алгоритм і передбачуване використання);

- інформація, що сприяє верифікації підпису (наприклад, ідентифікатор алгоритму підпису);

статус відкритого ключа.

Сертифікати відкритих ключів служать для гарантії дійсності відкритого ключа, але не сертифікують додаткової інформації. Для сертифікації додаткової інформації можуть використовуватись *сертифікати атрибутів*. Сертифікат атрибутів схожий із сертифікатом відкритого ключа й дозволяє передавати специфічну інформацію (атрибути), пов'язану з АС, об'єктами або відкритими ключами. Сертифікати атрибутів можуть бути асоційовані з конкретним відкритим ключем через зв'язок атрибутивної інформації із ключем. Такий зв'язок забезпечується за допомогою методів ідентифікації ключа, наприклад, можуть використовуватися порядковий номер відповідного сертифіката відкритого ключа, ґеш-код відкритого ключа або сертифіката. Сертифікат атрибутів підписується адміністратором сертифікації атрибутів і створюється разом з адміністратором реєстрації атрибутів.

Модель процесу сертифікації

Модель процесу сертифікації характеризує взаємозв'язок між основними логічними об'єктами, які беруть участь у формуванні і керуванні сертифікатами. Модель, запропонована в ISO/IEC 11770-1, визначає наступні логічні об'єкти, які можуть бути залучені в процес сертифікації (рис. 2.44):



Рис. 2.44. Об'єкти процесу сертифікації

орган сертифікації або *адміністратор сертифікації* відповідає за організацію та здійснення процесу сертифікації відкритих ключів користувачів, а також ручається за дійсність відкритих ключів. Це включає виконання таких дій, як призначення відкритим ключам різних імен за допомогою підписаних сертифікатів, керування призначенням порядкових номерів сертифікатам і анулювання сертифікатів;

сервер імен (name server) відповідає за керування простором імен користувачем з метою забезпечення кожного користувача унікальним іменем;

довідник сертифікатів (certificate directory) забезпечує підтримку сертифікатів в on-line режимі, тобто в стані повної готовності до використання їх користувачами. Звичайно це база даних або сервер, доступні для користувачів у режимі тільки для читання. Поповнення й підтримку довідника здійснює адміністратор сертифікації. Користувачі також можуть мати свої довідники сертифікатів. У цьому випадку за їх підтримку відповідає сам користувач;

центр генерації ключів здійснює генерацію пар відкритого/закритого ключа, а також генерацію симетричних ключів і паролів. Центр генерації ключів може бути частиною встаткування користувача, якщо користувач самостійно генерує собі ключ, або частиною органу сертифікації, або взагалі може бути окремою довірчою системою;

орган реєстрації або *адміністратор реєстрації* (registration authority) відповідає за авторизацію об'єктів, що відрізняються унікальними іменами, у якості члена домену безпеки. Реєстрація користувача звичайно допускає зв'язок ключового матеріалу з об'єктом.

Зовсім не обов'язково, що в конкретній реалізації системи сертифікації ці об'єкти будуть роздільними. В окремих випадках деякі з них взагалі можуть не існувати.

Створення, використання, верифікація, розподіл і анулювання сертифікатів відкритих ключів

Перед створенням сертифіката відкритого ключа для деякого об'єкта А з метою перевірки дійсності об'єкта і факту приналежності відкритого ключа об'єкта, який сертифікується, АС повинен здійснити ряд заходів, які часто носять не криптографічний, а організаційний характер.

Генерація ключової пари об'єкта А може здійснюватися як довіреною стороною, так і самостійно.

У першому випадку довірена сторона генерує ключову пару, співвідносить її з конкретним об'єктом і включає відкритий ключ й

ідентифікаційну інформацію об'єкта в сертифікат. Об'єкт одержує копію відповідного особистого ключа безпечним (автентичному і конфіденційному) каналом після того, як доведе свою дійсність (наприклад, особисто показавши паспорт). Всі сторони, що надалі використовують сертифікат, по суті делегують відповідальність за встановлення дійсності об'єкта довірчій стороні.

У другому випадку після генерації ключової пари об'єкт безпечним чином передає відкритий ключ довірчій стороні або особисто, або використовуючи захищений канал. За результатами перевірки дійсності джерела відкритого ключа довірена сторона генерує сертифікат відкритого ключа. При цьому адміністратор сертифікації повинен зажадати докази знання об'єктом відповідного особистого ключа, для того щоб запобігти шахрайству серед легітимних користувачів, що прагнуть для злочинних дій одержати на своє ім'я сертифікат відкритого ключа іншого користувача.

Для того щоб об'єкт В міг використовувати автентичний відкритий ключ об'єкта А, він повинен виконати наступні дії.

1. Одержати автентичний відкритий ключ адміністратора сертифікації (це однократна операція, яка здійснюється при реєстрації об'єкта органом сертифікації).

2. Одержати ідентифікаційну інформацію, яка однозначно ідентифікує передбачуваний об'єкт А.

3. Одержати по незахищеному каналу (наприклад, з довідника сертифікатів відкритих ключів або безпосередньо від А) сертифікат відкритого ключа об'єкта А, погоджений з ідентифікаційною інформацією об'єкта А.

4. Перевірити поточне значення часу і дати та співвіднести їх з терміном дії ключа, зазначеним у сертифікаті. Перевірити дійсність відкритого ключа АС. Перевірити підпис під сертифікатом, використовуючи відкритий ключ АС. Перевірити, що сертифікат не є анульованим.

Існує три моделі розподілу сертифікатів.

Якщо розглядати довідник сертифікатів як відкриту базу даних, то реалізується запитальна модель розподілу сертифікатів або pull-модель. У цьому випадку користувачі витягають сертифікати з бази даних по мірі необхідності.

Інша модель розподілу сертифікатів носить примусовий характер (push-модель). У даній моделі сертифікати примусово розподіляються

всім користувачам або періодично, або по мірі створення сертифіката. Така модель найбільш прийнятна для реалізації в закритих системах.

Нарешті третій підхід до розподілу сертифікатів полягає в тому, що окремі користувачі за необхідності самостійно забезпечують своїми сертифікатами інших користувачів, наприклад, розподіляють сертифікати відкритих ключів для перевірки підпису.

Оскільки довідник сертифікатів звичайно розглядається як незахищена третя сторона, то необхідно реалізувати систему керування доступом до довідника. Для всіх користувачів дозволена тільки функція зчитування з довідника, а для авторизованих об'єктів найчастіше такими є адміністратори сертифікації й реєстрації, реалізуються функції підтримки і відновлення довідника. Крім того, самі сертифікати захищаються підписом, у результаті чого вони й можуть розподілятися по відкритих каналах зв'язку. Виключенням є on-line сертифікати, які формуються адміністратором сертифікації в режимі реального часу за запитом користувача. Такі сертифікати мають короткий термін дії й розподіляються через довірчі сторони, які надають гарантії того, що даний сертифікат не анульований.

Сертифікати можуть бути анульовані ще до закінчення строку своєї дії. Це відбувається при компрометації ключів, за запитом будь-якого об'єкта про анулювання сертифіката, по витіканню терміну дії повноважень об'єкта, при зміні функціональних обов'язків об'єкта в організації й з інших причин. У зв'язку з цим необхідні засоби, які дозволяють ефективно інформувати інших користувачів про недійсні сертифікати. Це особливо важливо при компрометації ключів, оскільки зменшити можливий збиток можна шляхом виключення подальшого використання скомпрометованого ключа. Існують різні методи розв'язання завдання анулювання ключів і сертифікатів.

1. Включення дати закінчення терміну дії ключа (сертифіката) в одне з полів сертифіката. Крайнім випадком є on-line-сертифікати, які анулюються відразу ж після їх використання.

2. Попередження ручним способом, коли всі користувачі системи інформуються про анулювання ключа з використанням бездротових засобів або спеціальними каналами. Такий метод зручний у малих або закритих системах.

3. Використання відкритих файлів анульованих ключів, у яких утримуються ідентифікатори анульованих ключів. Перед використанням ключа всі користувачі перевіряють відсутність ідентифікатора даного

ключа у файлі.

4. Створення списку анульованих сертифікатів (Certificate Revocation List (CRL)). CRL є списком порядкових номерів або інших ідентифікаторів сертифікатів, які були анульовані визначеним органом сертифікації.

5. Використання сертифікатів анулювання, що є альтернативою CRL. Сертифікат анулювання може розглядатися як сертифікат відкритого ключа, що містить час і прапор анулювання, і службовець для скасування відповідного сертифіката. Оригінальний сертифікат виключається з довідника сертифікатів і замінюється сертифікатом анулювання.

Стандарти рекомендують вирішувати завдання анулювання ключів через ведення списку анульованих сертифікатів. Даний список обов'язково повинен бути позначений міткою часу і підписується відповідним органом сертифікації. Крім списку порядкових номерів анульованих сертифікатів список може містити й іншу інформацію, наприклад, причину анулювання того або іншого сертифіката. Підпис списку гарантує його автентичність і формується тим адміністратором сертифікації, який відповідає за формування сертифікатів. Звичайно CRL містить ім'я цього АС. Відновлення списку повинне здійснюватися регулярно, навіть якщо список не був змінений, з обов'язковою установкою дати відновлення. Це необхідно для того, щоб надати можливість користувачам перевірити "свіжість" списку. Крім того необхідно передбачити засоби і механізми ефективного й своєчасного розподілу списків анульованих сертифікатів.

Механізми керування ключами на основі використання симетричних методів (ISO/IEC 11770-2)

Друга частина стандарту ISO/IEC 11770-2 була опублікована в 1996 році і визначає механізми встановлення ключа, що використовують методи симетричної криптографії. В основному використовується симетричне шифрування, але також розглядається використання й криптографічних контрольних функцій. Здебільшого розглянуті в ISO/IEC 11770-2 механізми засновані на використанні протоколів автентифікації, розглянутих в ISO/IEC 9798-2.

ISO/IEC 11770-2 включає 13 механізмів встановлення ключа для:
розподілу сеансового ключа між парою об'єктів із
передвстановленим загальним головним ключем;

розподілу ключа між парою сторін через ЦРК;

розподілу ключа між парою сторін через ЦПК.

Таким чином, визначені механізми для всіх трьох моделей, визначених в ISO/IEC 11770-1 для випадку, коли два об'єкти належать тому самому домену безпеки. Механізми для випадку, коли об'єкти довіряють різним адміністраторам, можуть бути отримані з механізмів, викладених ISO/IEC 11770-2.

У даному підрозділі розглядаються п'ять механізмів встановлення ключа, характерні для кожної із трьох введених стандартом моделей керування ключами.

При розгляді механізмів будемо використовувати наступні позначення:

A і B – два об'єкти, що бажають встановити новий секретний ключ;

ID_A і ID_B – ідентифікатори об'єктів;

$E_K(X)$ позначає шифрування блоку даних X з використанням секретного ключа K (зазначимо, що спосіб шифрування в стандарті не визначається). Якщо спосіб шифрування забезпечує цілісність даних і автентифікацію джерела даних, то мається на увазі, що за даними будуть обчислені MDC- або MAC-коди і приєднані до даних перед шифруванням;

$X||Y$ позначає конкатенцію полів даних X і Y у точно встановленому порядку;

поля Text містять додаткову допоміжну інформацію.

Розглянемо конкретні механізми.

Механізм встановлення ключа № 3.

Умовою реалізації механізму № 3 є те, що об'єкти A і B уже мають у своєму розпорядженні загальний секретний ключ K_{AB} . A і B також повинні підтримувати синхронізацію в системі, використовуючи синхронізований годинник або синхропослідовність. Даний механізм реалізує модель об'єктів, що безпосередньо зв'язуються, засновану на однопрохідному (однобічному) протоколі автентифікації (ISO/IEC 9798-2 п. 5.11.) і забезпечує однобічну автентифікацію об'єкта A об'єктом B , а також неявну автентифікацію ключа об'єктом A . Об'єкт A вибирає ключ i , отже, має контроль над ключем.

Механізм має наступний протокол.

Резюме. Об'єкт A взаємодіє з об'єктом B .

Результат. Об'єкт A передає секретний ключовий матеріал (наприклад, сеансовий ключ) об'єкта B .

1. Початкові умови. Об'єкти A і B мають загальний ключ K_{AB} .
2. Повідомлення протоколу.

$$A \rightarrow B : E_{K_{AB}}(T / N \parallel ID_B \parallel K \parallel \text{Text}).$$

3. Дії за протоколом.

а) об'єкт A генерує ключовий матеріал і формує пакет даних, у якому поле T/N містить або мітку часу T, або порядковий номер повідомлення N, поле K містить новий ключовий матеріал, поле Text містить іншу додаткову інформацію, наприклад, тип алгоритму шифрування. Сформований пакет об'єкта шифрує на загальному секретному ключі K_{AB} і відправляє зашифроване повідомлення об'єкту B;

б) при одержанні шифрованого повідомлення об'єкт B розшифровує його, потім перевіряє наявність свого ідентифікатора ID_B , правильність мітки часу або порядкового номера і витягає необхідний ключовий матеріал з поля K.

Механізм установки ключа № 6

Механізм № 6 використовує випадкові непередбачені числа, тобто застосовується механізм "запит – відповідь". Він опирається на трипрохідний протокол автентифікації (ISO/IEC 9798-2 п.5.2.2) і також реалізує модель об'єктів, що безпосередньо зв'язуються. Механізм забезпечує

взаємну автентифікацію між A і B. Причому жоден з об'єктів не має контролю над ключем.

Протокол механізму установки ключа № 6.

Резюме. Об'єкта взаємодіє з об'єктом B.

Результат. Об'єкт A передає секретний ключовий матеріал (наприклад, сеансовий ключ) об'єкту B.

1. Початкові умови. Об'єкти A і B мають загальний ключ K_{AB} .
2. Повідомлення протоколу.

$$B \rightarrow A : ID_B \parallel R_B$$

$$A \rightarrow B : E_{K_{AB}}(R_A \parallel R_B \parallel ID_B \parallel K_A \parallel \text{Text } 1)$$

$$B \rightarrow A : E_{K_{AB}}(R_B \parallel R_A \parallel K_B \parallel \text{Text } 2).$$

3. Дії по протоколу.

а) об'єкт B формує непередбачене випадкове число R_B і відправляє його об'єкту A разом зі своїм ідентифікатором ID_B ;

б) об'єкт А генерує непередбачене випадкове число R_A і ключовий матеріал, який зберігається, в поле K_A . Потім формує пакет, шифрує його на загальному ключі K_{AB} і відправляє об'єкту В;

в) об'єкти А і В обчислюють свій новий загальний ключ як необоротну функцію від ключового матеріалу K_A і K_B . Стандарт допускає або K_A , або K_B встановити в нуль. Однак якщо використовуються і K_A , і K_B , то функція, яка використовується для комбінації цих величин, повинна мати властивості, при яких жоден з об'єктів не матиме керування над ключем.

Механізм встановлення ключа № 9

Механізм заснований на п'ятипрохідному протоколі автентифікації (ISO/IEC 9798-2 п. 6.2) і реалізує модель встановлення ключа через ЦРК, коли об'єкт А посилає ключ об'єкту В. У цьому протоколі вводиться довірча третя сторона ЦРК, яка довіряє і А і В. ЦРК має загальні секретні ключі K_{ACPK} і K_{BCPK} з об'єктами А й В та має контроль над ключами. Механізм забезпечує взаємну автентифікацію між А і В.

Протокол механізму встановлення ключа № 9.

Резюме. Об'єкт А взаємодіє з ЦРК і об'єктом В.

Результат. А передає секретний ключовий матеріал K_{AB} об'єкту В.

1. Початкові умови. Об'єкти А і В мають відповідні загальні ключі K_{ACPK} і K_{BCPK} із центром розподілу ключів.

2. Повідомлення протоколу.

$B \rightarrow A : R_B.$

$A \rightarrow \text{ЦРК} : ID_A \parallel R_A \parallel R_B \parallel ID_B.$

$\text{ЦРК} \rightarrow A : E_{K_{ACPK}}(R_A \parallel K_{AB} \parallel ID_B \parallel \text{Text 1}) \parallel E_{K_{BCPK}}(R_B \parallel K_{AB} \parallel ID_A \parallel \text{Text 2})$
 $B : E_{K_{BCPK}}(R_B \parallel K_{AB} \parallel ID_A \parallel \text{Text 2}) \parallel E_{K_{AB}}(R_A \parallel R_B \parallel \text{Text 3})$

$B \rightarrow A : E_{K_{AB}}(R_B \parallel R_A \parallel \text{Text 4})$

3. Дії по протоколу.

а) В генерує випадкове число R_B і відправляє його А;

б) А генерує випадкове число R_A і разом з R_B й ідентифікатором об'єкта В відправляє в ЦРК;

в) центр розподілу ключів формує ключ K_{AB} , який буде встановлений між А і В. Повідомлення із ключем K_{AB} ЦРК шифрує на ключах K_{ACPK} і K_{BCPK} й відправляє об'єкту А;

г) об'єкт А передає частину отриманого від ЦРК повідомлення, зашифрованого на ключі K_{BCPK} , об'єкту В. Крім того, він шифрує додаткове

повідомлення, що містить значення раніше отриманих чисел R_A і R_B на новому ключі K_{AB} ;

д) об'єкт В, при одержанні повідомлення від А, розшифровує його, витягає значення випадкових чисел, формує нове повідомлення і зашифровує на ключі K_{AB} ;

в) об'єкт А, одержавши і розшифрувавши повідомлення, порівнює значення випадкових чисел з наявними у нього. У випадку їх збігу він робить висновок про те, що ключ між ним і об'єктом В встановлений правильно.

Механізм встановлення ключа № 11

Даний механізм здійснює встановлення ключа з використання центру передачі ключів. ЦПК дозволяє об'єктам А і В, які не мають загального секретного ключа, встановити безпечний зв'язок через використання довгострокових ключів K_{ACPK} і K_{BCPK} і трипрохідного протоколу. Об'єкт А має контроль над ключем.

Протокол механізму встановлення ключа № 11.

Резюме. Об'єкт А взаємодіє зі ЦПК і об'єктом В.

Результат. А передає секретний ключовий матеріал K_{AB} об'єкту В.

1. Початкові умови. Об'єкти А і В мають відповідні загальні ключі K_{ACPK} і K_{BCPK} із центром розподілу ключів.

2. Повідомлення протоколу.

$$A \rightarrow \text{ЦПК} : ID_A \parallel E_{K_{ACPK}}(ID_B \parallel K_{AB})$$
$$\text{ЦПК} \rightarrow A : E_{K_{BCPK}}(K_{AB} \parallel ID_A \parallel \text{Text 1})$$
$$A \rightarrow B : E_{K_{BCPK}}(K_{AB} \parallel ID_A \parallel \text{Text 2}).$$

3. Дії по протоколу.

а) А шифрує новий встановлюваний ключ K_{AB} разом з ідентифікатором передбачуваного одержувача ID_B на загальному з ЦПК ключі K_{ACPK} і відправляє повідомлення ЦПК із власним ідентифікатором ID_A ;

б) при розшифруванні повідомлення ЦПК визначає, що воно призначене для об'єкта В, знаходить ключ K_{BCPK} і перешифровує повідомлення на цьому ключі для відправлення повідомлення об'єкту В;

в) ЦПК повертає повідомлення об'єкту А. Об'єкт А або відправляє повідомлення об'єкту В, або розміщає його на відкритому для В сайті. ЦПК може і самостійно відправити повідомлення об'єкту В, тобто:

$\text{ЦПК} \rightarrow \text{В} : E_{K_{\text{ВЦПК}}} (K_{\text{АВ}} \parallel \text{ID}_{\text{А}} \parallel \text{Text } 2).$

Механізм встановлення ключа № 12

Механізм використовує мітки часу і сумісний, за винятком деяких особливостей, з 4-прохідним протоколом автентифікації (ISO/IEC 9798-2 п. 6.1). Механізм, як і механізм № 11, реалізує модель розподілу ключів через ЦПК. ЦПК має загальні секретні ключі $K_{\text{АЦПК}}$ і $K_{\text{ВЦПК}}$ із А і В відповідно. Крім того ЦПК, А і В повинні підтримувати синхронізацію. Механізм забезпечує взаємну автентифікацію між А і В. Об'єкт А має контроль над ключем.

Протокол має вигляд:

Резюме. Об'єкт А взаємодіє з ЦПК і об'єктом В.

Результат. А передає секретний ключовий матеріал $K_{\text{АВ}}$ об'єкту В.

1. Початкові умови. Об'єкти А і В мають відповідні загальні ключі $K_{\text{АЦПК}}$ і $K_{\text{ВЦПК}}$ з центром розподілу ключів.

2. Повідомлення протоколу.

$\text{А} \rightarrow \text{ЦПК} : \text{ID}_{\text{А}} \parallel E_{K_{\text{АЦПК}}} (\text{TVPA} \parallel \text{ID}_{\text{В}} \parallel K_{\text{АВ}} \parallel \text{Text } 1) ,$

$\text{ЦПК} \rightarrow \text{А} : E_{K_{\text{АЦПК}}} (\text{TVPA} \parallel \text{ID}_{\text{В}} \parallel \text{Text } 2) \parallel E_{K_{\text{ВЦПК}}} (T_{\text{ЦПК}} / N_{\text{ЦПК}} \parallel K_{\text{АВ}} \parallel \text{ID}_{\text{А}}),$

$\text{А} \rightarrow \text{В} : E_{K_{\text{ВЦПК}}} (T_{\text{ЦПК}} / N_{\text{ЦПК}} \parallel K_{\text{АВ}} \parallel \text{ID}_{\text{А}} \parallel \text{Text } 3) \parallel E_{K_{\text{АВ}}} (\text{TA} / \text{NA} \parallel \text{ID}_{\text{В}} \parallel \text{Text } 4),$

$\text{В} \rightarrow \text{А} : E_{K_{\text{АВ}}} (\text{TB} / \text{NB} \parallel \text{ID}_{\text{А}} \parallel \text{Text } 5).$

3. Дії по протоколу.

а) А шифрує новий ключ $K_{\text{АВ}}$ разом з ідентифікатором передбачуваного одержувача $\text{ID}_{\text{В}}$ і параметром TVPA , що залежить від часу (випадкове число, мітка часу, порядковий номер) і обраний А для того, щоб порівняти відповідь від ЦПК із запитом до ЦПК (цей параметр не перевіряється ЦПК);

б) з отриманого повідомлення ЦПК визначає ідентифікатор об'єкта В, вибирає ключ $K_{\text{ВЦПК}}$ і шифрує на ньому новий загальний ключ $K_{\text{АВ}}$ разом з ідентифікатором об'єкта А та міткою часу (або порядковим номером) $T_{\text{ЦПК}}/N_{\text{ЦПК}}$. Потім ЦПК відправляє повідомлення об'єкту А;

в) об'єкт А пересилає отримане від ЦПК повідомлення об'єкту В разом з новим повідомленням, зашифрованим на новому загальному ключі $K_{\text{АВ}}$;

г) об'єкт В відправляє відгук об'єкту А на ключі $K_{\text{АВ}}$.

2.10.5. Механізми керування ключами на основі використання несиметричних методів (ISO/IEC 11770-3)

Третя частина стандарту ISO/IEC 11770-3 визначає механізми встановлення ключа, засновані на застосуванні методів несиметричної криптографії. Стандарт описує механізми, що забезпечують:

узгодження загального секретного ключа між об'єктами (сім механізмів узгодження ключа);

передачу секретного ключа від одного об'єкта до іншого (шість механізмів передачі ключа);

забезпечення доступності секретного ключа одного об'єкта іншому об'єкту способом, який перевіряється (три механізми передачі секретних ключів).

Перші два типи механізмів включають використання протоколів визначених в ISO/IEC 9798-3, а також включає різні методи несиметричної криптографії, наприклад, ключовий обмін Діффі – Геллмана.

Третій тип механізмів включає використання сертифікатів.

Всі сім механізмів узгодження ключа, визначених в ISO/IEC 11770-3, використовують спеціальну однобічну функцію F

$$F : H \times G \rightarrow G,$$

де G і H – множини.

Функція задовольняє наступним властивостям:

1. $F(h, g) = F(h', F(h', g))$ для будь-яких $h, h' \in H$ і $g \in G$.

2. За даними $F(h, g)$, $F(h', g)$ і g обчислювально неможливо визначити $F(h, F(h, g))$. Це означає, що $F(\bullet, g)$ є однобічною криптографічною функцією для будь-якого g . Об'єкти A і B повинні мати загальний елемент $g \in G$ (який може бути і секретним).

Для прикладу розглянемо механізм узгодження ключів № 5. Якщо об'єкти A і B бажають використовувати цей механізм для установки нового загального секретного ключа, то:

об'єкт A повинен мати особистий ключ $h_A \in H$, відомий тільки A , і відкритий ключ $p_A = F(h_A, g)$, відомий об'єкту B ;

об'єкт B повинен мати особистий ключ $h_B \in H$, відомий тільки B , і відкритий ключ $p_B = F(h_B, g)$, відомий об'єкту A .

Механізм забезпечує взаємну неявну автентифікацію ключа, але не дозволяє, однак, об'єктам A і B вибрати значення або форму K_{AB}

заздалегідь, тобто ні об'єкт А, ні об'єкт В не мають контролю над ключем. Отже, цей механізм не застосовується в системах, де необхідно мати спеціальний ключ виду – у таких випадках необхідно використовувати механізм передачі ключа.

Перед застосуванням механізму об'єкта вибирає випадкове (секретне) число $R_A \in H$, а В вибирає випадкове (секретне) число $R_B \in H$. Протокол обміну повідомленнями має вигляд:

$$A \rightarrow B : F(R_B, g) \parallel \text{Text 1}$$

$$B \rightarrow A : F(R_B, g) \parallel \text{Text 2.}$$

Після одержання першого повідомлення В обчислює загальний секретний ключ, як:

$$K_{AB} = \omega(F(hb, F(R_A, g)), F(R_B, pa)),$$

а об'єкт А, після одержання другого повідомлення, також обчислює загальний секретний ключ згідно з виразом:

$$K_{AB} = \omega(F(R_A, pb), F(R_B, g)).$$

Тут ω позначає загальну однобічну криптографічну функцію узгодження.

Одним із прикладів однобічної криптографічної функції $F(h, g)$, визначеної в стандарті, є дискретний логарифм. Дана функція лежить в основі ключового обміну Діффі – Геллмана, у якому:

$G = Z_p^*$ (мультиплікативна група цілих чисел за модулем простого числа p);

$$H = \{1, 2, \dots, p - 2\};$$

g – примітивний елемент Z_p^* ;

$$F(h, g) = gh \bmod p.$$

Реалізація механізму узгодження ключа № 5 з використанням дискретного логарифма має такий вигляд:

Узгодження ключів Діффі – Геллмана

Резюме. А і В обмінюються повідомленнями відкритим каналом зв'язку .

Результат. Між А і В узгоджується загальний секретний ключ K_{AB} .

1. Початкові умови. Обирається відповідне просте число p і

первісний елемент α групи Z_p^* . Дані параметри є відкритими.

2. Повідомлення протоколу.

$$A \rightarrow B: \alpha^x \bmod p$$

$$B \rightarrow A: \alpha^y \bmod p.$$

3. Дії за протоколом. Дані дії виконуються при кожному узгодженні ключа.

а) А вибирає випадкове число x , $1 \leq x \leq p - 2$, зберігає його в секреті і посилає повідомлення об'єкту В;

б) В вибирає випадкове число y , $1 \leq y \leq p - 2$, зберігає його в секреті і посилає повідомлення об'єкту А;

в) В одержує α^x і обчислює загальний ключ як $K_{AB} = (\alpha^x)^y \bmod p$;

г) А одержує α^y і обчислює загальний ключ як $K_{AB} = (\alpha^y)^x \bmod p$.

Розглянутий протокол забезпечує взаємну автентифікацію ключа. Фіксовані значення $\alpha^x \bmod p$ і $\alpha^y \bmod p$ виступають у ролі довгочасних відкритих ключів відповідних сторін, що розподіляються з використанням підписаних сертифікатів. Якщо такі сертифікати відомі в апріорі, то одержимо схему узгодження ключів без передачі секретної інформації. Однак інваріантність у часі загального ключа є недостою протоколу. Протокол Діффі – Геллмана може виконуватись на елементах мультиплікативної групи Z_p^* поля Z_p , а також на елементах мультиплікативної групи поля $GF(2^m)$ і на групі точок, визначених на еліптичній кривій над кінцевим полем. Більш докладно такі схеми викладені в стандарті IEEE P1363 [89; 90].

Передачу ключа від одного об'єкта до іншого реалізують за допомогою транспортних механізмів. Одним з таких механізмів є транспортний механізм № 5. Даний механізм передає ключ із використанням випадкових чисел і заснований на 3-прохідному протоколі автентифікації (ISO/IEC 9798-3 п. 5.2.2)

Механізм забезпечує взаємну автентифікацію і додаткове підтвердження одержання ключа, надаване об'єкту В (key confirmation). У ході реалізації протоколу встановлюються два ключа (передача у двох напрямках).

Протокол передачі ключа має вигляд.

Резюме. А взаємодіє з В.

Результат. Об'єкти А і В обмінюються новими секретними ключами K_A і K_B .

1. Початкові умови. А і В мають пари перетворення підпис/перевірки (S_A, Y_A) і (S_B, Y_B) відповідно. Обидві сторони повинні мати доступ до відкритого перевірного перетворення один одного. А і В мають пари перетворень шифрування/розшифрування (E_A, D_A) і (E_B, D_B) відповідно. Обидві сторони повинні мати доступ до відкритого перетворення, що шифрує, один одного.

2. Повідомлення протоколу.

$A \rightarrow B : ID_A \parallel R_A \parallel \text{Text } 1$

$B \rightarrow A : S_B(R_B \parallel R_A \parallel ID_A \parallel E_A(ID_B \parallel K_B \parallel \text{Text } 2) \parallel \text{Text } 3 \parallel \text{Text } 4$

$A \rightarrow B : S_A(R_A \parallel R_B \parallel ID_B \parallel E_B(ID_A \parallel K_A \parallel \text{Text } 5) \parallel \text{Text } 6 \parallel \text{Text } 7.$

3. Дії за протоколом.

а) Об'єкт А надсилає запит (Text 1) об'єкту В на обмін новими секретними ключами спільно зі своїм ідентифікатором і випадковим числом R_A .

б) об'єкт В генерує новий секретний ключ K_B , шифрує його на відкритому ключі шифрування об'єкта А, а потім зашифрований блок разом з іншою інформацією підписує, використовуючи свій особистий ключ підпису і алгоритм підпису S_B ;

в) об'єкт А генерує новий ключ шифрування K_A , шифрує його на відкритому ключі шифрування об'єкта В, а потім підписує його разом з іншою інформацією на своєму особистому ключі підпису. При цьому для підтвердження правильності приймання ключа K_B у поле Text 6 об'єкт А поміщає контрольну суму ключа K_B . Взаємний контроль ключів може бути досягнутий шляхом обчислення обох ключів K_A і K_B із використанням однобічної функції.

2.10.6. Безпека розподілу ключів

Жоден ключ шифрування не можна використовувати нескінченно. Час його дії повинен минати автоматично, подібно паспортам і ліцензіям. Причини цього в наступному:

ніж довше використовується ключ, тим більша ймовірність його компрометації. Якщо ключ використовується протягом року, то ймовірність його компрометації набагато вища, ніж якби його використовували тільки один день;

ніж довше використовується ключ, тим більші втрати при компрометації ключа. Якщо ключ використовується для шифрування одного документа, то втрата ключа означає компрометацію тільки цього документа. Якщо той же самий ключ використовується для шифрування більших обсягів інформації, то його компрометація призводить до значних втрат;

ніж довше використовується ключ, тим більша спокуса прикласти необхідні зусилля для його розкриття. Розкриття ключа, який використовується протягом доби, дозволить прочитати всі повідомлення, передані протягом доби;

розкриття ключа, який використовується протягом року, дозволить противнику одержати доступ до секретної інформації, переданої протягом року;

у ряді випадків трудомісткість криптоаналізу визначається кількістю шифротекстів, отриманих у результаті шифрування на одному ключі.

Для будь-якого криптографічного додатка необхідна стратегія, що визначає допустимий час життя ключа. У різних ключів можуть бути різні періоди життя. Наприклад, для криптографічного телефону має сенс використовувати ключ тільки протягом телефонної розмови, а для нової розмови – використовувати новий ключ. У ключів повинна бути відносно коротка тривалість життя, залежно від значимості і кількості даних, зашифрованих протягом заданого періоду. Ключ для каналу зв'язку зі швидкістю передачі 1 Гбіт/с, можливо, доведеться міняти частіше, ніж для модемного каналу швидкістю 9600 біт/с. Якщо існує ефективний метод передачі нових ключів, сеансові ключі повинні мінятися щодня. Ключі шифрування ключів так часто міняти не потрібно. При цьому для криптоаналітика утворюється небагато шифротексту, а відповідний відкритий текст (випадкові ключі) неможливо вгадати. Однак якщо ключ шифрування скомпрометований, потенційні втрати катастрофічні. У деяких додатках ключі шифрування ключів замінюються тільки раз на місяць або навіть раз у рік. Необхідно збалансувати ризики, пов'язані з заміною ключа або використанням фіксованого ключа. Ключі шифрування, які використовуються при шифруванні даних довгострокового зберігання, не можна міняти часто. Щоденне розшифрування й повторне шифрування на новому ключі може призвести до негативних наслідків – подібна процедура призводить до нагромадження криптоаналітиком робочого матеріалу для наступної атаки. Розв'язком може послужити шифрування даних на деякому

унікальному ключі (або ключах) з наступним його шифруванням на ключі шифрування ключів. Ключ шифрування ключів повинен зберігатися в безпечному місці. Втрата цього ключа означає втрату всіх ключів, які були на ньому зашифровані. Тривалість життя секретних ключів асиметричної криптографії залежить від додатків. Секретні ключі для електронних підписів можуть використовуватись роками (навіть протягом людського життя). У багатьох криптомережах термін дії секретних ключів обмежений двома роками. Старий ключ також повинен зберігатися в секреті на випадок підтвердження підпису, зробленого під час дії старого ключа. Але для підписання нових документів повинен використовуватись новий ключ. Такий підхід дозволяє зменшити кількість матеріалу, яку криптоаналітик зможе використовувати для розкриття ключа.

2.10.7. Криптографія на еліптичних кривих

Більшість продуктів і стандартів, у яких для шифрування і електронних підписів застосовується метод криптографії з відкритим ключем, базується на алгоритмі RSA. Кількість бітів ключа, необхідна для надійної захищеності RSA, за останні роки різко зросла, що обумовило відповідне зростання завантаження систем у додатках, що використовують RSA. Це породило множину проблем, особливо для вузлів, що спеціалізуються на електронній комерції, де потрібен захист великої кількості транзакцій. Але є підхід, який може конкурувати з RSA – це криптографія на основі еліптичних кривих (ECC – Elliptic curve cryptography). Уже зараз вживають спроби стандартизації цього підходу, наприклад, як у стандарті IEEE P1363 за криптографією з відкритим ключем.

Привабливість підходу на основі еліптичних кривих порівняно з RSA полягає в тому, що з використанням еліптичних кривих забезпечується еквівалентний захист при дуже невеликій кількості розрядів, внаслідок чого зменшується завантаження процесору [25; 26].

У цьому підрозділі подані основи використання еліптичних кривих і криптографії з їх застосуванням.

Еліптичні криві

Перевага підходу на основі еліптичних кривих порівняно із завданням факторизації числа, яке використовується в RSA, або завданням цілочисельного логарифмування, що застосовується в

алгоритмі Діффі – Геллмана і в DSS, полягає в тому, що в цьому випадку забезпечується еквівалентний захист при меншій довжині ключа.

У загальному випадку рівняння еліптичної кривої E має вигляд:

$$y^2 + axy + by = x^3 + cx^2 + dx + e.$$

Як приклад розглянемо еліптичну криву E , рівняння якої має вигляд:

$$y^2 + y = x^3 - x^2.$$

На цій кривій лежать тільки чотири точки, координати яких є цілими числами. Це числа $A(0, 0)$, $B(1, -1)$, $C(1, 0)$ і $D(0, -1)$ (рис. 2.45).

Для визначення операції додавання для точок на еліптичній кривій зробимо наступні припущення (рис. 2.46).

На площині існує нескінченно вилучена точка $O \in E$, у якій сходяться всі вертикальні прямі.

Будемо вважати, що дотична до кривої перетинає точку торкання два рази.

Якщо три точки еліптичної кривої лежать на прямій лінії, то їх сума дорівнює O .

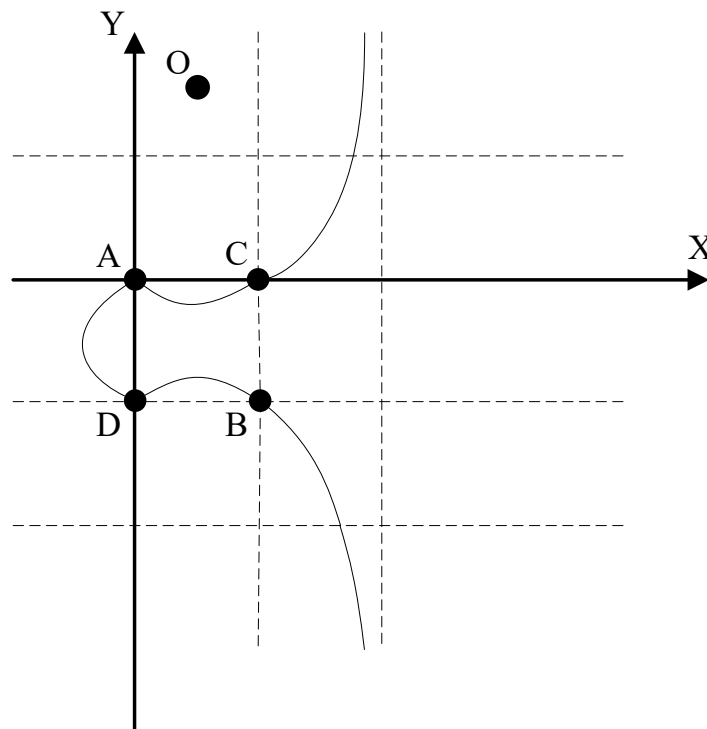


Рис. 2.45. Приклад еліптичної кривої із чотирма точками

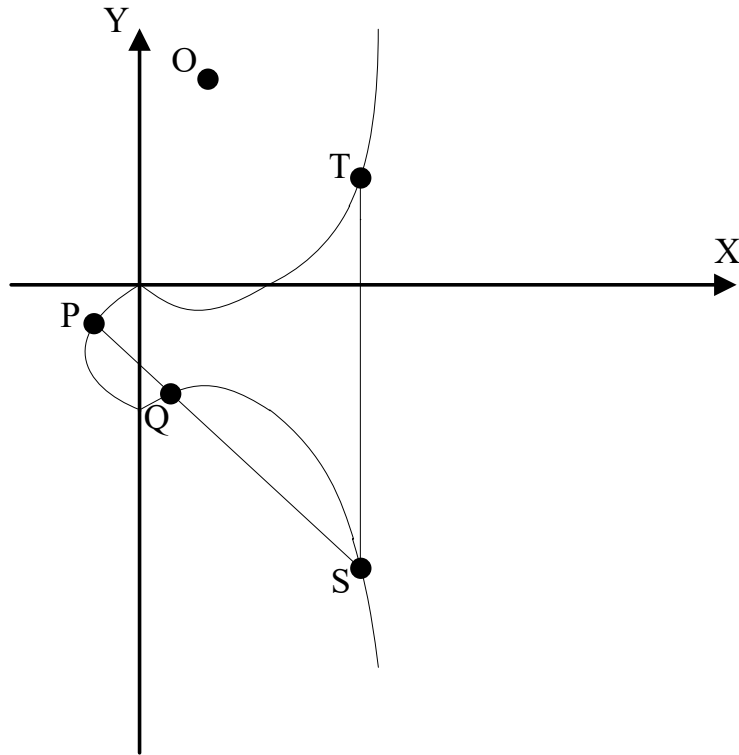


Рис. 2.46. Додавання точок на еліптичній кривій

Введемо наступні правила додавання точок на еліптичній кривій.

Точка O виступає в ролі нульового елемента. Так, $O = -O$ і для будь-якої точки P на еліптичній кривій $P + O = P$.

Вертикальна лінія перетинає криву у двох точках з однією і тією ж координатою x – скажемо, $S = (x, y)$ і $T = (x, -y)$. Ця пряма перетинає криву і в нескінченно вилученій точці. Тому $P_1 + P_2 + O = O$ і $P_1 = -P_2$.

Щоб скласти дві точки P і Q (див. рис. 3.46) з різними координатами x , слід провести через ці точки пряму і знайти точку перетинання її з еліптичною кривою. Якщо пряма не є дотичною до кривої в точках P або Q , то існує тільки одна така точка, позначимо її S . Згідно з нашим припущенням

$$P + Q + S = O.$$

Отже,

$$P + Q = -S \text{ або } P + Q = T.$$

Якщо пряма є дотичною до кривої в якій-небудь із точок P або Q , то в цьому випадку слід покласти $S = P$ або $S = Q$ відповідно.

Щоб подвоїти точку Q , слід провести дотичну в точці Q і знайти іншу точку перетинання S з еліптичною кривою. Тоді:

$$Q + Q = 2 \times Q = -S.$$

Введена таким способом операція додавання підкоряється всім звичайним правилам додавання, зокрема комутативному і асоціативному законам. Множення точки P еліптичної кривої на позитивне число k визначається як сума k точок P .

У криптографії з використанням еліптичних кривих всі значення обчислюються за модулем p , де p є простим числом. Елементами даної еліптичної кривої є пари невід'ємних цілих чисел, які менше p і задовольняють приватному виду еліптичної кривої:

$$y^2 \equiv x^3 + ax + b \pmod{p}.$$

Таку криву будемо позначати $E_p(a, b)$. При цьому числа a і b повинні бути менше p і задовольняти умові:

$$4a^3 + 27b^2 \pmod{p} \neq 0.$$

Множина точок на еліптичній кривій обчислюється в такий спосіб.

1. Для кожного такого значення x , що $0 \leq x \leq p$ обчислюється $x^3 + ax + b \pmod{p}$.

2. Для кожного з отриманих на попередньому кроці значень з'ясовується, чи має це значення квадратний корінь за модулем p . Якщо ні, то в $E_p(a, b)$ немає точок з цим значенням x . Якщо корінь існує, є два значення y , які відповідні до операції добування квадратного кореня (виключенням є випадок, коли єдиним значенням виявляється $y = 0$). Ці значення (x, y) і будуть точками $E_p(a, b)$.

Множина точок $E_p(a, b)$ має наступні властивості:

1. $P + 0 = P$.

2. Якщо $P = (x, y)$, то $P + (x, -y) = 0$. Точка $(x, -y)$ є від'ємним значенням точки P і означається $-P$. Зазначимо, що $(x, -y)$ лежить на еліптичній кривій і належить $E_p(a, b)$.

3. Якщо $P = (x_1, y_1)$ і $Q = (x_2, y_2)$, де $P \neq Q$, то $P + Q = (x_3, y_3)$ визначається за наступними формулами:

$$x_3 \equiv \lambda^2 - x_1 - x_2 \pmod{p};$$

$$y_3 \equiv \lambda(x_1 - x_3) - y_1 \pmod{p},$$

де $\lambda = (y_2 - y_1)/(x_2 - x_1)$, якщо $P \neq Q$, $\lambda = (3x_1^2 + a)/2y_1$, якщо $P = Q$.

Число λ є кутовим коефіцієнтом січної, яка проведена через точки $P = (x_1, y_1)$ і $Q = (x_2, y_2)$. При $P = Q$ січна перетворюється в дотичну, ніж і пояснюється наявність двох формул для обчислення λ .

Завдання, яке повинен розв'язати в цьому випадку атакуючий, є свого роду завданням “дискретного логарифмування на еліптичній кривій”, і формулюється в такий спосіб.

Дані точки P і Q на еліптичній кривій $E_p(a,b)$. Необхідно знайти коефіцієнт $k < p$ такий, що $P = k \times Q$.

Відносно легко обчислити P за даними k і Q , але досить важко обчислити k , знаючи P і Q .

Розглянемо три способи використання еліптичних кривих у криптографії.

Аналог алгоритму Діффі – Геллмана обміну ключами

Обмін ключами з використанням еліптичних кривих може бути виконаний у такий спосіб. Спочатку обирається просте число $p \approx 2^{180}$ і параметри a і b для рівняння еліптичної кривої. Це задає множину точок $E_p(a,b)$. Потім в $E_p(a,b)$ обирається генеруюча точка $G = (x_1, y_1)$. При виборі G важливо, щоб найменше значення n , при якому $n \times G = 0$, виявилось набагато більшим простим числом. Параметри $E_p(a,b)$ і G криптосистеми є параметрами, відомими всім учасникам [22].

Обмін ключами між користувачами A і B проводиться за наступною схемою.

1. Абонент A вибирає ціле число n_A , менше n . Це число є закритим ключем абонента A . Потім абонент A обчислює відкритий ключ $P_A = n_A \times G$, який становить деяку точку на $E_p(a,b)$.

2. Таким же чином абонент B обирає закритий ключ n_B і обчислює відкритий ключ P_B .

3. Абоненти обмінюються відкритими ключами, після чого обчислюють загальний секретний ключ K .

Абонент A : $K = n_A \times P_B$.

Абонент B : $K = n_B \times P_A$.

Слід зазначити, що загальний секретний ключ становить пару чисел. Якщо даний ключ передбачається використовувати в якості сеансового ключа для алгоритму симетричного шифрування, то із цієї пари необхідно створити одне значення.

Алгоритм електронного підпису на основі еліптичних кривих ECDSA

Алгоритм ECDSA (Elliptic Curve Digest Signature Algorithm), прийнятий у якості стандартів ANSI X9F1 і IEEE P1363.

Створення ключів:

1. Обирається еліптична крива $E_p(a,b)$. Число точок на ній повинне ділитися на велике ціле n .

2. Обирається точка $P \in E_p(a,b)$.

3. Обирається випадкове число $d \in [1, n-1]$.

4. Обчислюється $Q = d \times P$.

5. Закритим ключем є d , відкритим ключем – (E, P, n, Q) .

Створення підпису:

1. Обирається випадкове число $k \in [1, n-1]$.

2. Обчислюється:

$$k \times P = (x_1, y_1)$$

$$r = x_1 \pmod{n}.$$

Перевіряється, щоб r не дорівнювало нулю, тому що в цьому випадку підпис не залежатиме від закритого ключа. Якщо $r = 0$, то обирається інше випадкове число k .

Обчислюється:

$$k^{-1} \pmod{n}.$$

Обчислюється:

$$s = k^{-1} (H(M) + dr) \pmod{n}.$$

Перевіряється, щоб s не дорівнювало нулю, тому що в цьому випадку необхідного для перевірки підпису числа $s^{-1} \pmod{n}$ не існує.

Якщо $s = 0$, то обирається інше випадкове число k .

Підписом для повідомлення M є пара чисел (r, s) .

Перевірка підпису:

1. Перевірити, що цілі числа r і s належать діапазону чисел $[0, n-1]$. Якщо ні, то результат перевірки негативний, і підпис відкидається.

2. Обчислити $w = s^{-1} \pmod{n}$ і $H(M)$.

3. Обчислити:

$$u_1 = H(M) w \pmod{n}$$

$$u_2 = rw \pmod{n}.$$

4. Обчислити:

$$u_1 P + u_2 Q = (x_0, y_0)$$

$$v = x_0 \pmod{n}.$$

Підпис правильний тільки у тому випадку, коли $v = r$.

Шифрування/розшифрування з використанням еліптичних кривих

Розглянемо найпростіший підхід до шифрування/розшифрування з використанням еліптичних кривих. Завдання полягає в тому, щоб зашифрувати повідомлення M , яке може бути презентовано у вигляді точки на еліптичній кривій $P_m(x,y)$.

Як і у випадку обміну ключем, у системі шифрування/розшифрування в якості параметрів розглядається еліптична крива $E_p(a,b)$ і точка G на ній. Абонент B обирає закритий ключ n_B і обчислює відкритий ключ $P_B = n_B \times G$. Щоб зашифрувати повідомлення P_m , використовується відкритий ключ одержувача B P_B . Абонент A вибирає випадкове ціле позитивне число k і обчислює зашифроване повідомлення C_m , що є точкою на еліптичній кривій.

$$C_m = \{k \times G, P_m + k \times P_B\}.$$

Щоб розшифрувати повідомлення, учасник B множить першу координату точки на свій закритий ключ і віднімає результат із другої координати:

$$P_m + k \times P_B - n_B \times (k \times G) = P_m + k \times (n_B \times G) - n_B \times (k \times G) = P_m.$$

Абонент A зашифрував повідомлення P_m додаванням до нього $k \times P_B$. Ніхто не знає значення k , тому, хоча P_B і є відкритим ключем, ніхто не знає $k \times P_B$. Протівнику для відновлення повідомлення доведеться обчислити k , знаючи G і $k \times G$. Зробити це буде нелегко.

Одержувач також не знає k , але йому в якості підмови посилається $k \times G$. Помноживши $k \times G$ на свій закритий ключ, одержувач отримає значення, яке було додане відправником до незашифрованого повідомлення. Тим самим одержувач, не знаючи k , але маючи свій закритий ключ, може відновити незашифроване повідомлення.

Аналіз теоретико-числових задач

Сьогодні широко відомі та набули застосування в криптографічних додатках цілий ряд теоретико-числових задач, які дозволяють будувати односторонні функції з “секретом”, їх класифікація наведена на рис. 2.47.

Зазначимо, що при оцінці складності обчислення тієї або іншої задачі, за можливістю, буде вказуватися складність її обчислення з використанням квантового комп'ютера. Дане зауваження стало досить істотним у світлі останньої презентації канадської компанії D-Wave Systems в 2007 році реально діючого квантового комп'ютера. Розроблювачами було відзначено, що криптоаналіз є одним з основних напрямків його застосування. При описі практичної реалізації зробимо наступне припущення: оцінка виробляється з урахуванням реалізації на W -розрядному процесорі, при цьому операції присвоєння і керування потоком виконання команд не враховуються. Постійний розвиток обчислювальної техніки і збільшення її продуктивності, що

підтверджується законом Мура, а також удосконалення математичних методів криптоаналізу тягне до періодичного перегляду розмірів ключової інформації. На рис. 2.47 використовуються наступні аббревіатури: УЕП – електронний підпис, АСШ – асиметричне шифрування, ОК – обмін ключами.

Розглянемо окремо кожну теоретико-числову задачу.

Опис задачі

Нехай дане ціле число $N > 1$, що не є простим, тоді існує ціле число a , таке, що $1 < a < N$ і $a \mid N$. Завдання полягає в тому, щоб знайти таке a або довести, що воно дорівнює 1.

У криптографічних додатках використовують число N , що складається із двох простих множників сумарної величини, іншими словами, вони мають порівняні двійкові довжини.

Факторизація великого цілого числа

Складність криптоаналізу

Задача факторизації великого цілого числа N на множники належить до класу субекспоненціальних алгоритмів, складність яких становить $L_N(\alpha, \beta) = \exp((\beta + o(1))(\log N)^\alpha (\log \log N)^{1-\alpha})$.

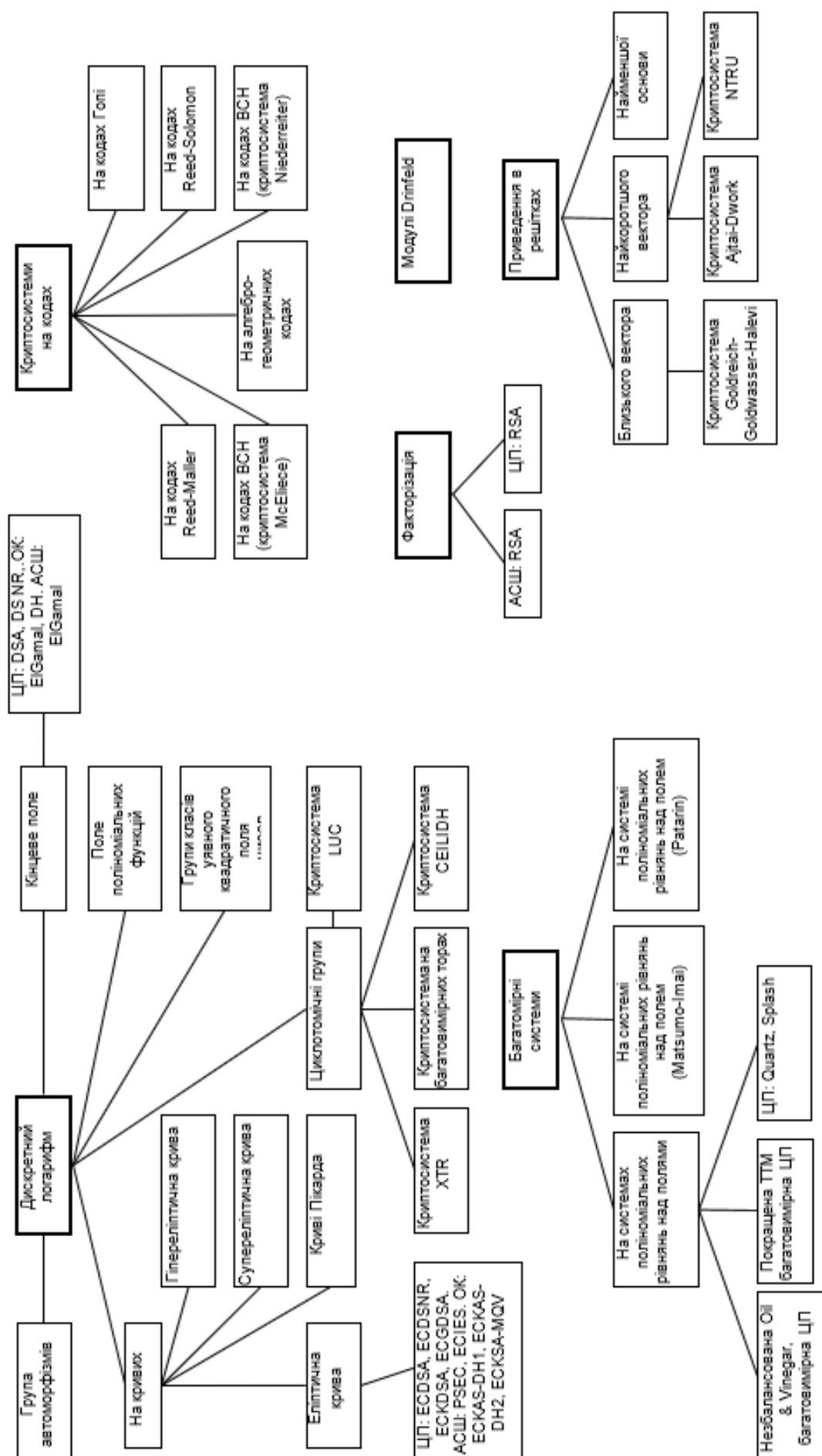
Складність алгоритму GNFS (General Number Field Sieve). Решета поля чисел загального виду складе $L_N\left(\frac{1}{3}, \sqrt[3]{\frac{64}{9}}\right)$.

Складність алгоритму SNFS (Special Number Field Sieve). Решета поля чисел спеціального виду $N = a^b + c$ складе $L_N\left(\frac{1}{3}, \sqrt[3]{\frac{32}{9}}\right)$.

Складність обчислення задачі з використанням квантового комп'ютера з кількістю $O(\log N)$ кубітів складе $O(\log N)$. Відомо, що для факторизації чисел двійкової довжини до 1024 бітів найбільш прийнятним вважається алгоритм ECM (Elliptic Curve Method) з використанням еліптичних кривих. Для чисел більшого розміру прийнято використовувати GNFS-алгоритм.

Продуктивність/складність реалізації

Примітив RSA. Даний примітив є першим серед реалізованих. Нехай k є секретним і нехай $N = pq$ є добутком двох великих простих множників двійковою довжиною $\frac{k}{2}$. Нехай також відкритий параметр – показник e , таке, що $\gcd(e, (p-1)(q-1)) = 1$ і відповідний особистий параметр $d \equiv e^{-1}(\bmod \lambda(N))$, де $\lambda(x)$ – функція Карлмайкла, $\lambda(N) = \text{lcm}(p-1, q-1)$.



Як видно з опису даного примітива, основними операціями в кільці цілих чисел є:

додавання/вирахування;

множення;

піднесення у квадрат;

мультиплікативне інвертування;

приведення за модулем;

піднесення в степінь, яке базується на операціях множення і піднесення у квадрат.

Відомо, що в даній ситуації найбільш доцільним є використання арифметики Montgomery. Основу зазначених перетворень становить операція множення. Відомо декілька модифікацій множення методом Montgomery:

SOS:separated operand scanning;

CIOS:coarsely integrated operand scanning;

FIOS:finely integrated operand scanning;

CIHS:coarsely integrated hybrid scanning.

Алгоритм CIOS є найбільш прийнятним для реалізації на процесорах загального призначення з погляду продуктивності.

У табл. 2.17 наведемо складності операцій у кільці цілих чисел, виражених в операціях процесора.

У табл. 2.17 подані такі позначення:

l – двійкова довжина показника. M_{Z_N} – операція множення в кільці чисел Z_N . S_{Z_N} – операція піднесення у квадрат у кільці чисел Z_N . MM_{Z_N} – операція множення методом Montgomery у кільці чисел Z_N . $k = \lceil l/W \rceil$ – кількість машинних слів двійковою довжиною W , необхідних для зберігання в пам'яті числа двійкової довжини l .

h – кількість блоків, на які розбивається показник в алгоритмі Lim-Lee, $a = \lceil l/h \rceil$ – розмір блоку, v – кількість підблоків, на які розбивається

кожний блок, розмір блоку $b = \lceil a/v \rceil$. Варіюючи значеннями h і v , можна досягти необхідної продуктивності при відомих обмеженнях на розмір пам'яті для зберігання передобчислень. Оптимальні значення для сучасних персональних комп'ютерів становлять $h = 8$ і $v = 3$ (або $v = 4$).

Зашифрування. На цьому етапі виробляється піднесення в степінь чисел довжиною k машинних слів за допомогою одного з методів,

зазначених у рядках 7 – 9 табл. 3.9. Аналітично дане перетворення має вигляд:

$$y = x^e \pmod{N},$$

де y – зашифроване повідомлення;

x – вихідне повідомлення;

N – складовий модуль;

e – ключ зашифрування.

Кількість операцій процесора, необхідних для виконання зашифровування інформаційного блоку, відповідає сумарній складності операції піднесення в степінь і приведення за модулем.

Розшифрування. На цьому етапі відбувається процес, аналогічний попередньому.

$$x = y^d \pmod{N},$$

де y – зашифроване повідомлення;

x – вихідне повідомлення;

N – складовий модуль;

d – ключ розшифрування.

Для піднесення в степінь $y^d \pmod{N}$ на стороні власника особистої інформації варто скористатися Китайською теоремою про залишки (CRT – Chinese Remainder Theorem). Знаючи розкладання $N = pq$, значення $y^d \pmod{N}$ може бути отримане як $y^d \pmod{N} = s_q + q(i_q(s_p - q_q) \pmod{p})$,
 $s_p = y^{d_p} \pmod{p}$, $s_q = y^{d_q} \pmod{q}$, $d_p = d \pmod{p-1}$, $d_q = d \pmod{q-1}$,
 $i_q = q^{-1} \pmod{p}$.

Припустивши, що двійкова довжина

$$\log_2 p = \log_2 q = \log_2 d_p = \log_2 d_q = \frac{1}{2}l,$$

це дозволяє знизити навантаження на процесор, тому що довжина операндів, з якими йому доведеться оперувати, складе $1/2l$ бітів, що дозволяє одержати обчислювальну складність, наведену в табл. 2.17, рядок 10.

Таблиця 2.17

Складності операцій у кільці цілих чисел, виражені в операціях процесора

№	Назва	Передобчислення	Складність
1	Класичне приведення за модулем*		$k(k+2)M_{CPU} + kD_{CPU}$
2	Приведення за модулем, метод Barrett*		$k(k+4)M_{CPU}$
3	Приведення за модулем, метод Montgomery*		$k(k+1)M_{CPU}$
4	Додавання без приведення за модулем		kA_{CPU}
5	Вирахування без приведення за модулем		kA_{CPU}
6	Множення Montgomery, множення методом CIOS [44; 73] без приведення за модулем		$(2k^3 + 2k)M_{CPU} + (4k^2 + 6k + 2)A_{CPU}$
7	Множення KCM (Karatsuba-Comba-Montgomery) без приведення за модулем		$(k^2 + k + X_1)M_{CPU} + (2k^2 + 4k + 1 + X_2)A_{CPU}$
8	Множення KCM із приведенням за модулем псевдо-мерсенові числа		$(\frac{3}{4}k^2 + k + 1)M_{CPU} + (4k^2 + 2k + 2)A_{CPU}$
9	Піднесення в степінь, метод "піднести до квадрата і помножити"		$\frac{1}{2}lM_{Z_N} + lS_{Z_N}$
10	Піднесення в степінь, проміжні множення, виконані методом Montgomery		$(2 + \frac{3}{2}l)MM_{Z_N}, k(2k + 1)(2 + \frac{3}{2}l)M_{CPU}$
11	Піднесення в степінь, метод Lim-Lee, складність передобчислень $(v + b + h - 1 + v(2^h - 2)(v + b))S_{Z_N} + vh2^{h-1}M_{Z_N}$	$v(2^h - 1)$	$(\frac{2^h - 1}{2^h}a - 2)M_{Z_N} + bS_{Z_N}$
12	Піднесення в степінь на стороні власника особистої інформації, проміжні множення, виконані методом Montgomery з деякими допущеннями $\log_2 p = \log_2 q = \log_2 d_p = \log_2 d_q = \frac{1}{2}l$		$(7 + \frac{3}{2}l)MM_{Z_p}, \frac{1}{4}k[(k + 1)(14 + 3l) + k]M_{CPU}$

Примітка: * – у даних роботах відсутнє посилання на кількість.

Існують також й інші варіанти реалізації описаних алгоритмів, які володіють набагато більшою продуктивністю, ніж представлені, але й більш складною логікою і потребують значних передобчислень (табл. 2.18).

Таблиця 2.18

Складність криптопримитива RSA, виражена в кількості операцій у кільці цілих чисел і кількості операцій процесора

Складність	Зашифровування	Розшифрування
Операцій в кільці	$(2 + \frac{3}{2}l)MM_{Z_N}$	$(7 + \frac{3}{2}l)MM_{Z_p}$
Операції процесора	$k(2k + 1)(2 + \frac{3}{2}l)M_{CPU}$	$\frac{1}{4}k[(k + 1)(14 + 3l) + k]M_{CPU}$
Операції в кільці за наявності передобчислень	$(\frac{2^h - 1}{2^h}a + b)MM_{Z_N}$	$(\frac{2^h - 1}{2^{h+1}}a + 5 + \frac{b}{2})MM_{Z_N}$
Операції процесора за наявності передобчислень	$(\frac{2^h - 1}{2^h}a + b)((\frac{3}{4}k^2 + k + 1)M_{CPU} + (4k^2 + 2k + 2)A_{CPU})$	$(\frac{2^h - 1}{2^h}a + 5 + b)((\frac{3}{4}k^2 + k + 1)M_{CPU} + (4k^2 + 2k + 2)A_{CPU})$

MM_{Z_N} – операція множення методом Montgomery у кільці чисел Z_N . M_{Z_N} – операція множення в кільці чисел Z_N .

При одержанні результатів, наведених у табл. 2.17, авторами було зроблене припущення про рівність складності операцій множення й піднесення у квадрат у кільці цілих чисел.

Дискретний логарифм

У цьому випадку під задачею дискретного логарифма мається на увазі цілий клас задач, які відрізняються одна від одної лише алгебраїчними структурами, що утворюють групи.

Найбільш загальне визначення даної задачі. Нехай дана кінцева циклічна група $\langle G \rangle$ порядку n , що утворить елемент P і довільний елемент групи Q . Задача знаходження дискретного логарифма складається з пошуку такого цілого числа x , що $Q = xP$. Зазначимо, що це число наведене за модулем n . Розглянемо задачі дискретного логарифмування (DLP) на конкретних алгебраїчних структурах, які утворюють групи.

Дискретний логарифм у мультиплікативній групі кінцевого поля (чисел, поліномів)

Опис задачі

Нехай дані групові елементи g (генератор поля) і h , тоді під рішенням DLP будемо розуміти обчислення рівняння $h = g^l$ відносно l або доказ того, що обчислення не існує. Необхідною умовою є $h \in \langle g \rangle$, тобто $l \in [0, \text{ord}(g)-1]$ й $h^{\text{ord}(g)} = 1$, де $g, h \in GF(q)$, n – порядок елемента, утворюючий групу $n = \text{ord}(g)$, l – секретний ключ, h – відкритий ключ.

Складність криптоаналізу

У табл. 2.19 наведемо оцінки складності обчислення DLP у поле за результатами огляду.

Таблиця 2.19

Оцінки складності обчислення DLP у полі

№	Назва алгоритмів	Поле	Складність	Тип
1	Узагальнена решітка поля чисел	$GF(p)$	Потрібно $O\left(\exp\left((64/9)^{1/3} + o(1)\right)(\ln p)^{1/3} (\ln \ln p)^{2/3}\right)$ групових операцій	Д
2	Узагальнена решітка поля чисел	$GF(2^m)$	Потрібно $O\left(\exp\left(1.587m^{1/3} (\ln m)^{2/3}\right)\right)$ групових операцій	Д
3	Алгоритм Adleman-Coppersmith	$GF(2^m)$	Потрібно $O\left(\exp\left((c + o(1))m^{1/3} (\log m)^{2/3}\right)\right)$ групових операцій, де $c \approx 1.4$	Д

Д – детермінований алгоритм, що дає точне обчислення.

Складність/продуктивність реалізації

У криптографії широкого поширення одержали поля $GF(p)$, p – просте число і їх розширення $GF(p^m)$. Розширені поля, у свою чергу, діляться за характеристикою на $GF(2^m)$ і $GF\left((2^m - c)^n\right)$ з характеристикою у вигляді псевдомерсенових чисел. За степенем розширення: на $GF(2^m)$, m – просте число і композитні $GF\left((2^m)^n\right)$. Також варто виділити прості поля з характеристикою у вигляді псевдомерсенових чисел і узагальнених Мерсенових чисел. Кожний з наведених полів знайшли своє застосування в криптографії, тому що є кращими при реалізації на різних платформах.

Одним з найцікавіших полів вважається Optimal Extension Field (OEF) через високопродуктивну арифметику. На рис. 2.48 наводиться їх класифікація.

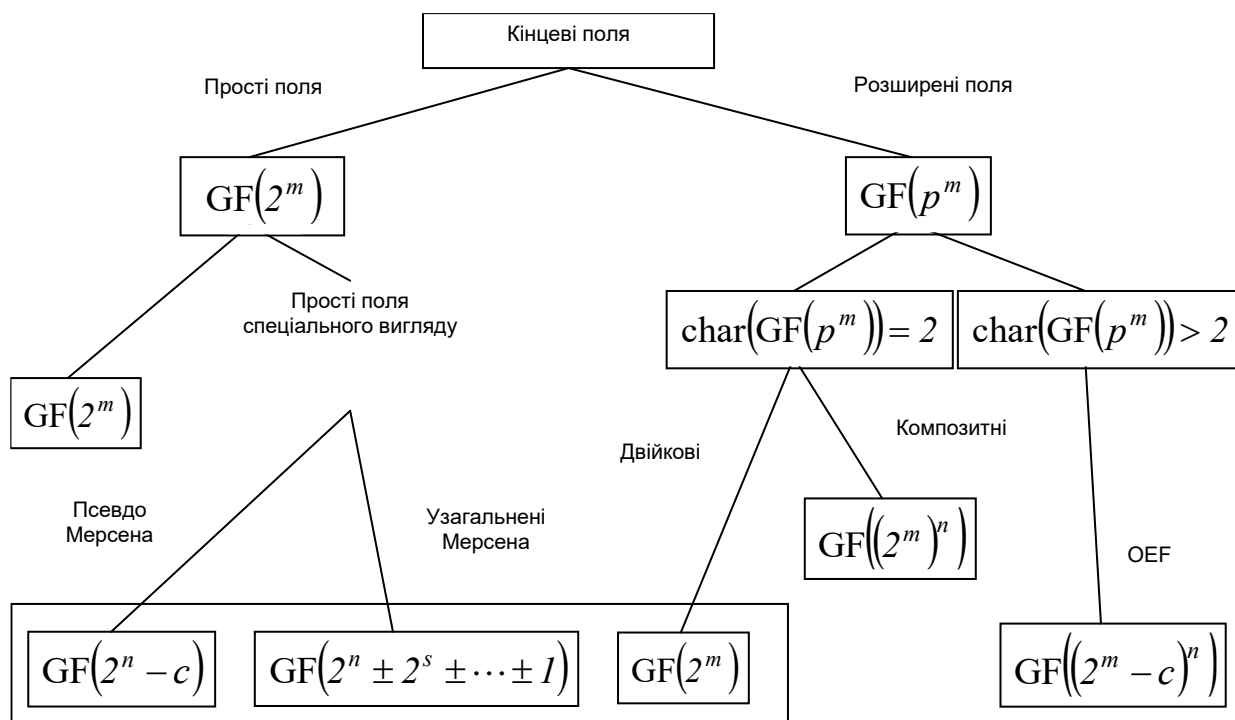


Рис. 2.48. Кінцеві поля, які пропонуються для використання в криптографічних додатках

Розгляд інших полів виходить за рамки даної роботи, існує досить велика кількість публікацій відносно реалізації арифметики в цих полях.

Прості поля. Розгляд почнемо із простих полів, модулем яких є псевдомерсенові і узагальнені Мерсеневі числа $p = \beta^t - c$, де c – невелике ціле. У табл. 2.20 наведемо складності операції приведення за відомими простими модулями в операціях процесора.

Таблиця 2.20

Теоретичні оцінки складності операції приведення

№ п/п	Модуль	Складність в операціях	Складність в операціях процесора
1	Псевдомерсеневе число загального виду, $p = \beta^t - c$		$(tl + l^2)M_{CPU}$, t – двійкова довжина модуля p , причому $l = \log_2 c \leq \frac{t}{2}$, k – двійкова довжина числа

№ п/п	Модуль	Складність в операціях	Складність в операціях процесора
2	Псевдомерсеневе число загального виду, $p = \beta^t - c$, коли c розміщується в одному машинному слові		$(t+1)M_{CPU}$, t – двійкова довжина модуля p , причому $l = \log_2 c \leq \frac{t}{2}$, k – двійкова довжина числа
3	Псевдомерсеневе число $p = 2^{192} - 2^{64} - 1$	$2A_{F_p}$	$12A_{CPU}$, $W = 32$
4	Псевдомерсеневе число $p = 2^{224} - 2^{96} + 1$	$7A_{F_p}$	$49A_{CPU}$, $W = 32$
5	Псевдомерсеневе число $p = 2^{256} - 2^{224} + 2^{192} + 2^{96}$	$15A_{F_p}$	$120A_{CPU}$, $W = 32$
6	Псевдомерсеневе число $p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$	$15A_{F_p}$	$180A_{CPU}$, $W = 32$
7	Псевдомерсеневе число $p = 2^{521} - 1$	$\frac{3}{2}A_{F_p} + 17X_{CPU} + 33Sh_{CPU}$	$25,5A_{CPU} + 17X_{CPU} + 33Sh_{CPU}$, $W = 32$
8	Додавання без приведення за модулем		kA_{CPU}
9	Вирахування без приведення за модулем		kA_{CPU}
10	Множення Montgomery без приведення за модулем		$(2k^3 + 2k)M_{CPU} + (4k^2 + 6k + 2)A_{CPU}$
11	Множення КСМ без приведення за модулем		$(k^2 + k + X_1)M_{CPU} + (2k^2 + 4k + 1 + X_2)A_{CPU}$
12	Множення без приведення за модулем		$(k(k-1)+t)(3A_{CPU} + M_{CPU})$
13	Піднесення у квадрат без приведення за модулем		$(k(2k-1)+t)(4A_{CPU} + \frac{1}{2}M_{CPU})$

Далі розглянемо складність операції піднесення в степінь. У табл. 2.21 подані аналітичні вирази.

Таблица 2.21

Теоретичні оцінки складності операції піднесення в степінь

№	Назва	Розмір передобчислень	Середня складність
1	Метод “Піднесення у квадрат і множення”	–	$\frac{1}{2}lM_{F_p} + lS_{F_p}$
2	Метод з фіксованою шириною вікна передобчислень	2^{w-l}	$\left(\left\lfloor \frac{l-1}{w} \right\rfloor (w+1-2^{-w}) + 1 - 2^{-((l-1) \bmod w)}\right)M_{F_p}$

№	Назва	Розмір передобчислень	Середня складність
3	Метод з шириною вікна передобчислень, яка адаптується	2^{w-l}	$\left(2^{w-l} + \frac{l}{w+l} + l - w\right)M_{F_p}$
4	Метод Lim-Lee	$v(2^h - 1)$	$\left(\frac{2^h-l}{2^h}a - 2\right)M_{F_p} + bS_{F_p}$

Остаточний вибір того або іншого алгоритму буде вироблятися на основі не тільки теоретичної оцінки складності, але й експериментальної оцінки.

Складність/продуктивність реалізації. У табл. 2.22 наведемо складність перетворень у поле $GF(p^6)$, де $p \equiv 2 \pmod{9}$, в операціях у поле $GF(p)$.

Таблиця 2.22

Складність операцій у поле $GF(p^6)$ в операціях у поле $GF(p)$

№	Операція в полі $GF(p^6)$	Складність
1	2	3
1	Обчислення a^p або a^{p^5}	$1A_{F_p}$
2	Обчислення a^{p^2} , a^{p^3} або a^{p^4}	$2A_{F_p}$
3	Обчислення a^2	$12M_{F_p}$
4	Обчислення $a \cdot b$	$18M_{F_p}$
5	Піднесення в степінь. Метод “піднести до квадрата й помножити”	$8lM_{F_p}$
6	Піднесення в степінь. Метод “піднести до квадрата й помножити” з поданням показника в JSF вигляді	$\approx 6.62lM_{F_p}$
7	Піднесення в степінь. Метод “піднести до квадрата й помножити”	$12 + 6lM_{F_p}$
8	Одночасне піднесення в степінь. Метод “піднести до квадрата й помножити” $a^m b^n$, варто зберігати 31 елемент $GF(p^6)$, обчислення алгоритму Піпенгера для $w = l/10$, $t = 5$	$2.93(l - 10)M_{F_p}$
9	Піднесення в степінь. Метод з адаптованою шириною вікна (змінне вікно) передобчислень, варто зберігати 2^{w-1} елементів $GF(p^6)$	$\left(2^{w-l} + \frac{l}{w+l} + l - w\right)l8M_{F_p}$
10	Піднесення в степінь. Метод “Lim-Lee”, варто зберігати $v(2^h - 1)$ елементів поля $GF(p^6)$	$\left(\frac{2^h-l}{2^h}a - 2\right)l8M_{F_p} + bl2M_{F_p}$

У табл. 2.22 надані позначення:

M_{F_p} – операція множення в поле $GF(p)$, $A_{GF(p)}$ – операція додавання в поле $GF(p)$. l – двійкова довжина показника.

У табл. 2.23 наведені особливості реалізації електронного підпису.

Таблица 2.23

Особливості реалізації електронного підпису

Умови проведення експериментальних оцінок часу виконання RSA, XTR DSA, ECDSA, криптосистема	Процесор	Операційна система	Компілятор	Особливості реалізації
ECDSA	Intel,PentiumII400MHz	MS Windows 2000	MS Visual C++ 6.0	3 асемблером
RSA, XTRDSA	Intel,PentiumII450MHz		C	

Наведемо в табл. 2.24 результати порівняння експериментальних оцінок часу формування й перевірки електронного підпису за допомогою RSA, XTR DSA, ECDSA.

Таблица 2.24

Експериментальні оцінки часу виконання RSA, XTR DSA, ECDSA

Перетворення	Довжина ключа	Час формування ключа, мс	Формування підпису	Перевірка підпису
RSA	1020	1224	5	40 (w/ CRT: 123)
XTRDSA	170	73	23	11
ECDSA	192		0,681	3,94

У табл. 2.25 наведемо теоретичну оцінку складності перетворень XTRDSA, ECDSA.

Таблица 2.25

Теоретична оцінка числа операцій множення в поле $GF(p)$ для XTR, EC, $\log_2 p = 170$

Назва примітива	Формування	Перевірка
XTRDSA	1700	2575
ECDSA	1360	2754

Іншим альтернативним поданням алгебраїчної структури є CEILIDH-представлення, що розшифровується як Compact, Efficient, Improves on LUC, and Improves on Diffie-Hellman [19; 48]. Зупинимося на ньому більш докладно.

Елементи групи можуть бути також представлені точками еліптичної кривої, що дозволяє вже говорити про ECDLP – задачу дискретного логарифма в групі точок еліптичної кривої. Зупинимося на цій задачі детальніше.

Дискретний логарифм у групі точок еліптичної кривої над кінцевим полем

Опис задачі

Нехай дані точки $P, Q \in E(\text{GF}(q))$, тоді рішенням ECDLP вважатимемо обчислення рівняння $Q = IP$ відносно I або докази того, що обчислення не існує. Необхідною умовою є $Q \in \langle P \rangle$, тобто $I \in [0, \text{ord}(P) - 1]$, причому $\text{ord}(P) \cdot Q = O$, де P – базова точка, що утворить групу, $n = \text{ord}(P)$ – порядок базової точки, I – секретний ключ, Q – відкритий ключ, O – точка на нескінченності.

Проведемо короткий теоретичний екскурс.

Еліптичною кривою E над полем K називається гладка крива, що описується множиною рішень $(x, y) \in K^2$ рівняння в узагальненій формі Вейерштраса:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad a_1, a_2, a_3, a_4, a_6 \in K,$$

і перебуває в афінній площини $A^2(\bar{K}) = \bar{K} \times \bar{K}$ із точкою O на ∞ .

Альтернативним є наступне визначення. Еліптична крива – гладка крива E з P^2 над полем K із сімейства кривих з P^2 над полем k , утворених двома кубами $x_0x_1x_2 = 0$ та $x_0^3 + x_1^3 + x_2^3 = 0$ у формі Хассе [22; 81]:

$$E_{(a,b)} : ax_0x_1x_2 + b(x_0^3 + x_1^3 + x_2^3) = 0, (a, b) \in P^1.$$

З погляду практичної реалізації варто розглядати конкретні криві, задані над полями парної й непарної характеристики.

Далі розглянемо більш специфічні представлення еліптичної кривої над різними полями:

Поле непарної характеристики:

Подання Дочі-Ікарт-Кохеля (Doche-Icart-Kohel):

$$E_{(a)} : y^2 = x^3 + ax^2 + 16ax, a \in k, a(a-64) \neq 0.$$

Нейтральним елементом у цьому випадку є точка O на ∞ в проєктивних координатах $(0:1:0)$.

Подання Дочі-Ікарт-Кохеля (Doche-Icart-Kohel) орієнтоване на побудову точок:

$$E_{(a)} : y^2 = x^3 + 3x^2 + 3a(x+1)^2, a \in k, a \neq 0, a \neq \frac{9}{4}.$$

Нейтральним елементом у цьому випадку є точка O на ∞ , які проєктуються на координатах $(0:1:0)$.

Подання Едвардса (Edwards).

$$E_{(c,d)} : x^2 + y^2 = c^2(1 + dx^2y^2), c, d \in k, d = 1.$$

Технічно крива Едвардса не є еліптичною, тому що є сингулярною. Нейтральним елементом у цьому випадку є точка O на ∞ в координатах $(0, c)$. Зазначимо, що точка $(0, -c)$ володіє порядком 2, у той час як точки $(0, c)$ і $(-c, 0)$ мають порядок 4.

Подання Хассе (Hasse) – кубічна крива в афінних координатах:

$$E_{(d)} : x^3 + y^3 + 1 = 3dxy, d \in k,$$

або в стандартних, які проєктуються в координатах:

$$E_{(d)} : X^3 + Y^3 + Z^3 = 3dXYZ, d \in k, d^3 \neq 1.$$

Нейтральним елементом у цьому випадку є точка O на ∞ в проєктивних координатах $(1:-1:0)$. Над полем з нетривіальним кубічним корнем w з 1 існує дві точки на нескінченності: $(1:-w:0)$ і $(1:-w^2:0)$.

Подання схрещування Якобі (Jacobi intersection).

$$E_{(a)} : s^2 + c^2 = 1, as^2 + d^2 = 1, a \in k, a \neq 0 \text{ і } a \neq 1.$$

Нейтральним елементом у цьому випадку є точка O на ∞ в проєктивних координатах $(0:1:1)$.

Подання у формі рівняння Якобі 4-ого степеня (Jacobi quartics) орієнтоване на подвоєння:

$$E_{(a)} : y^2 = x^4 + 2ax^2 + 1, a \in k, \text{char}(k) \neq 2, 3, a^2 \neq 1.$$

Також може бути подане в наступному приватному вигляді:

$$E_{(a)} : y^2 = (1 - x^2)(1 - a^2 x^2), \quad a \in k, \quad a^2 \neq 1.$$

Нейтральним елементом у цьому випадку є точка O на ∞ з координатами $(0, 1)$.

Подання Монтгомері (Montgomery).

$$E_{(a,b)} : by^2 = x^3 + ax^2 + x, \quad a, b \in k.$$

Нейтральним елементом у цьому випадку є точка O на ∞ в координатах $(0, 1)$.

Подання Вейерштрасса (Weierstrass).

$$E_{(a,b)} : y^2 = x^3 + ax^2 + b, \quad a, b \in k, \quad b \neq 0 \text{ і } a^3 + 27b^3 \neq 0$$

Нейтральним елементом у цьому випадку є точка O на ∞ які проєктуються в координатах $(0:1:0)$.

Поле парної характеристики:

Двійкове представлення Едвардса (Edwards).

$$E_{(d_1, d_2)} : d_1(x + y) + d_2(x^2 + y^2) = (x + x^2)(y + y^2), \quad d_1, d_2 \in k.$$

Нейтральним елементом, у цьому випадку є точка O на ∞ в проєктивних координатах $(0:1:0)$.

Подання Вейерштрасса (Weierstrass).

$$E_{(a,b)} : y^2 + xy = x^3 + ax^2 + b, \quad a, b \in k.$$

Нейтральним елементом у цьому випадку є точка O на ∞ в проєктивних координатах $(0, 0)$ для випадку $b \neq 0$, у протилежному випадку $(0, 1)$.

Складність криптоаналізу

Складність обчислення ECDLP для стандартизованих кривих за допомогою методу ρ -Pollard становить $O((\ln^c h) \sqrt{\pi h/2})$ групових операцій, де h – найбільший простий дільник $\text{ord}(P)$, c – невелика константа. Відповідність двійкової довжини ключів симетричних шифрів надана в табл. 2.26.

Відповідність двійкової довжини ключів симетричних шифрів, RSA і криптосистем, побудованих на еліптичних кривих над простими й двійковими полями

Двійкова довжина ключа, який симетричний алгоритму шифрування	Назва симетричного алгоритму	Двійкова довжина ключа RSA	Двійкова довжина простого числа p для базового поля $GF(p)$	Степінь розширення m для базового поля $GF(2^m)$
80	SKIPJACK	1248	192	163
112	Triple-DES	2432	244	233
128	AES Small	3248	256	283
192	AES Medium	7936	384	409
256	AES Large	15424	521	571

Найбільш активні дослідження виробляються в галузі криптоаналізу, що дозволить істотно знизити об'єми обчислень для зламу конкретної криптосистеми (програмно й/або апаратно реалізованої) за допомогою аналізу інформації, яка отримана непрямим шляхом (Side Channel Attacks):

Споживаної енергії (Differential power analysis).

Час відгуку (Time analysis).

Складність/продуктивність реалізації

Відомо, що у всіх криптографічних примітивах на еліптичній кривій основною операцією є операція скалярного множення. У свою чергу, скалярне множення може бути представлене у вигляді ієрархії (рис. 2.49).



Рис. 2.49. Узагальнена ієрархія операцій при скалярному множенні в групі точок еліптичної кривої

Більш докладна ієрархія операцій при скалярному множенні може бути подана у такому вигляді (рис. 2.50).

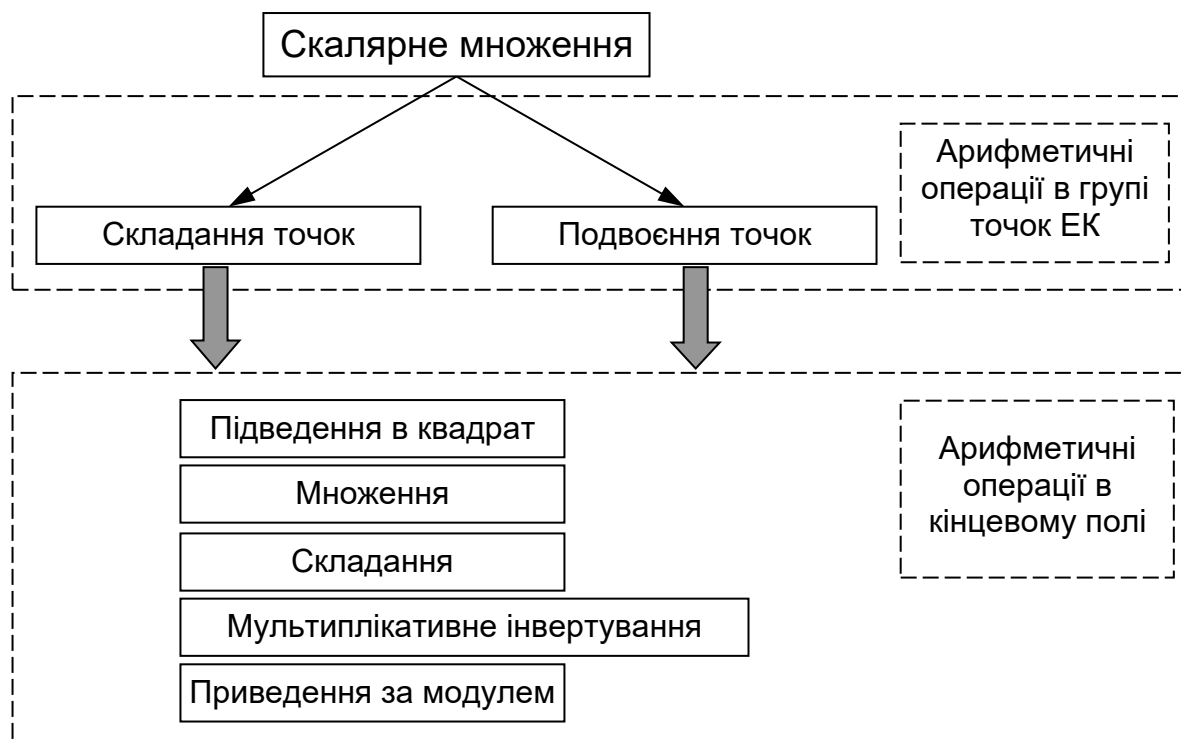


Рис. 2.50. Ієрархія операцій, використовуваних для скалярного множення точок еліптичної кривої

Розглянемо складності арифметики точок ЕК у різних формах і поданнях. В основу огляду покладені класифікація, результати і БД арифметик у групі точок ЕК, яка створена Т. Ланге.

Аналіз складності арифметики точок ЕК в різних формах і представленнях

Дискретний логарифм у яacobіані гіпереліптичної кривої над кінцевими полями (чисел, поліномів)

Опис задачі

Нехай дана гіпереліптична крива C ($GF(q)$) роду g й представленні дивізори $D_1, D_2 \in J(GF(q))$, під рішенням HCDLP, будемо розуміти обчислення рівняння $D_2 = I D_1$ відносно I або доказу, що обчислення не існує, де $J(GF(q)) = D^0/P$ – яacobіан, P – множина основних дивізорів, D^0 – множина дивізорів степеня 0, n – порядок дивізора D_1 , тобто $n = \text{ord}(D_1)$, I – секретний ключ, D_2 – відкритий ключ.

Складність криптоаналізу

Найбільш прийнятним для обчислення HECDLP вважається алгоритм Index-calculus. Деякі ідеї при рішенні DLP можуть бути застосовані й для обчислення HECDLP у яacobіані $J_C(k)$ гіпереліптичної кривої C , роду g над полем $k = F_q$ у мнимій квадратичній формі. Якщо рід g значно більший порівняно з q , тоді субекспоненціальним часом роботи імовірнісного алгоритму з відомим $n = q^g$ позначають $O(L_n[c])$, де $L_{q^g}[c] = \exp((c + o(1))\sqrt{g \log q \log g \log q})$ для деякої позитивної константи c .

Найбільш ефективним алгоритмом обчислення задачі дискретного логарифма в яacobіані ГЕК прийнято вважати алгоритм Theriault. Різні його варіанти дозволяють одержати складності $O(g^5 q^{2-2/(g+1)+\varepsilon})$ та $O(g^5 q^{2-4/(2g+1)+\varepsilon})$.

Продуктивність/складність реалізації. Відомо, що у всіх криптографічних примітивах на гіпереліптичній кривій основною операцією є операція скалярного множення в яacobіані. У свою чергу скалярне множення може бути подане у вигляді ієрархії (рис. 2.51).



Рис. 2.51. Узагальнена ієрархія операцій при скалярному множенні дивізорів у яacobіані ГЕК

Більш докладна ієрархія операцій при скалярному множенні може бути подана у вигляді рис. 2.52

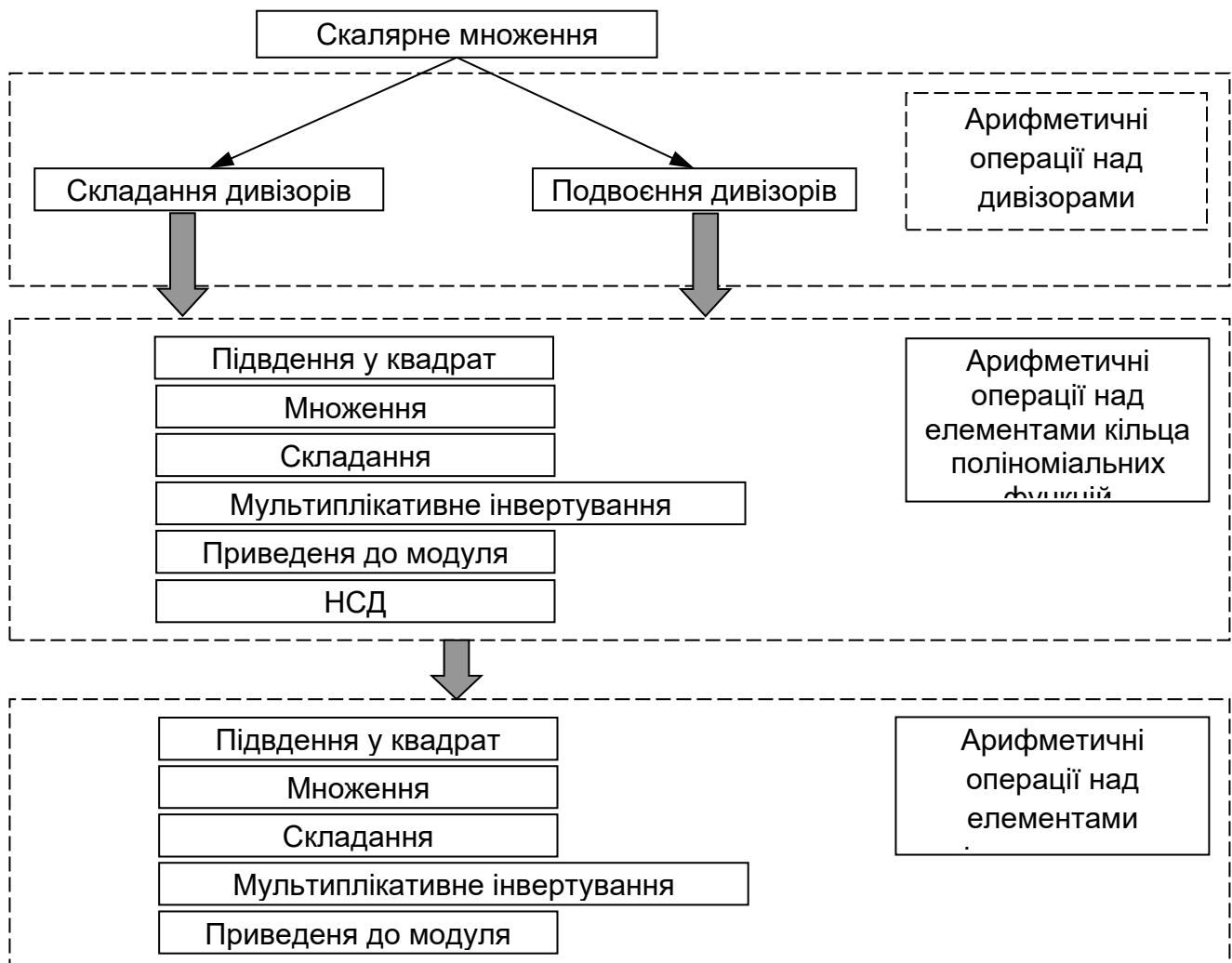


Рис. 2.52. Ієрархія операцій, використовуваних для скалярного множення дивізорів ГЕК

2.11. Методи та алгоритми забезпечення автентичності та цілісності даних

Саме питання забезпечення цілісності і автентичності інформації в сучасних банківських мережах є досить актуальним сьогодні. Оскільки методи криптоаналізу постійно вдосконалюються, потрібно проводити аналіз існуючих методів забезпечення автентифікації для усунення недоліків.

Розповсюджені наступні методи забезпечення автентичності повідомлення (рис. 2.53):

додавання до повідомлення коду автентифікації повідомлення (message authentication code, MAC-код) або зашифрованої контрольної суми;

введення електронних підписів;

контрольні суми, контроль CRC, ґешування і електронний підпис – базові засоби автентифікації при електронній передачі даних.

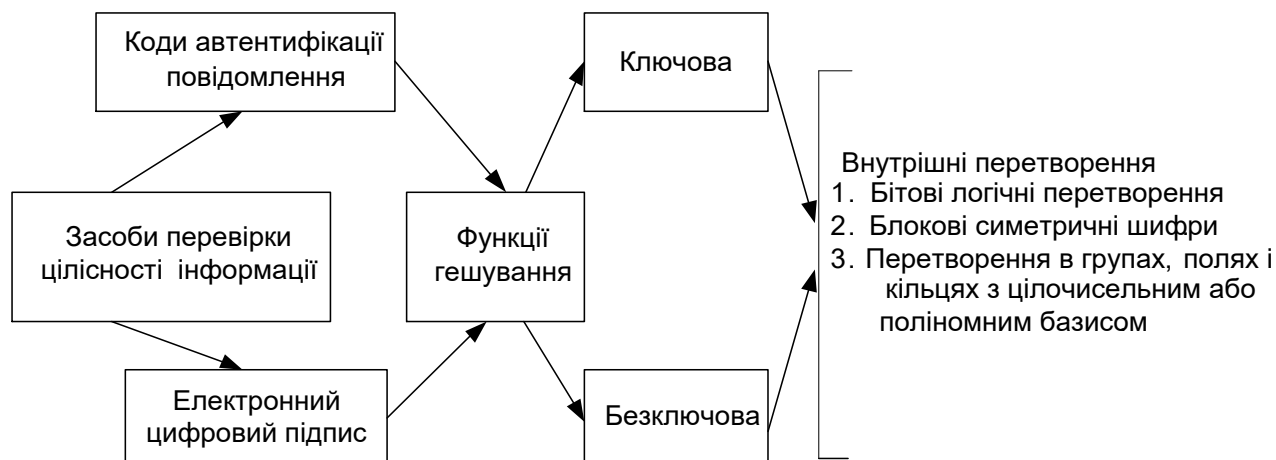


Рис. 2.53. Класифікація засобів перевірки цілісності інформації

2.11.1. Електронний підпис

При розгляді стандартів ISO 7498-2 та ISO / IEC 10181 визначено, що одним із найнадійніших способів вирішення завдань, пов'язаних з автентичністю (процедура встановлення достовірності твердження про те, що об'єкт (або суб'єкт) володіє заявленими властивостями) даних і джерел повідомлень, є процедури електронного підпису, побудовані на основі асиметричних криптографічних алгоритмів. УЕП становить рядок даних, який залежить від деякого секретного параметра (ключа), відомого тільки особі, яка підписує, і від вмісту повідомлення, яке вписується і представлене в цифровому вигляді.

Таким чином, удосконалений електронний підпис (УЕП) – реквізит електронного документа, призначений для захисту даного електронного документа від підробки, отриманий у результаті криптографічного перетворення інформації з використанням закритого ключа електронного підпису, що дозволяє ідентифікувати власника сертифіката ключа підпису, а також установити відсутність перекручування інформації в електронному документі.

На основі існуючих стандартів удосконаленого електронного підпису запропонована класифікація УЕП (рис. 2.54).

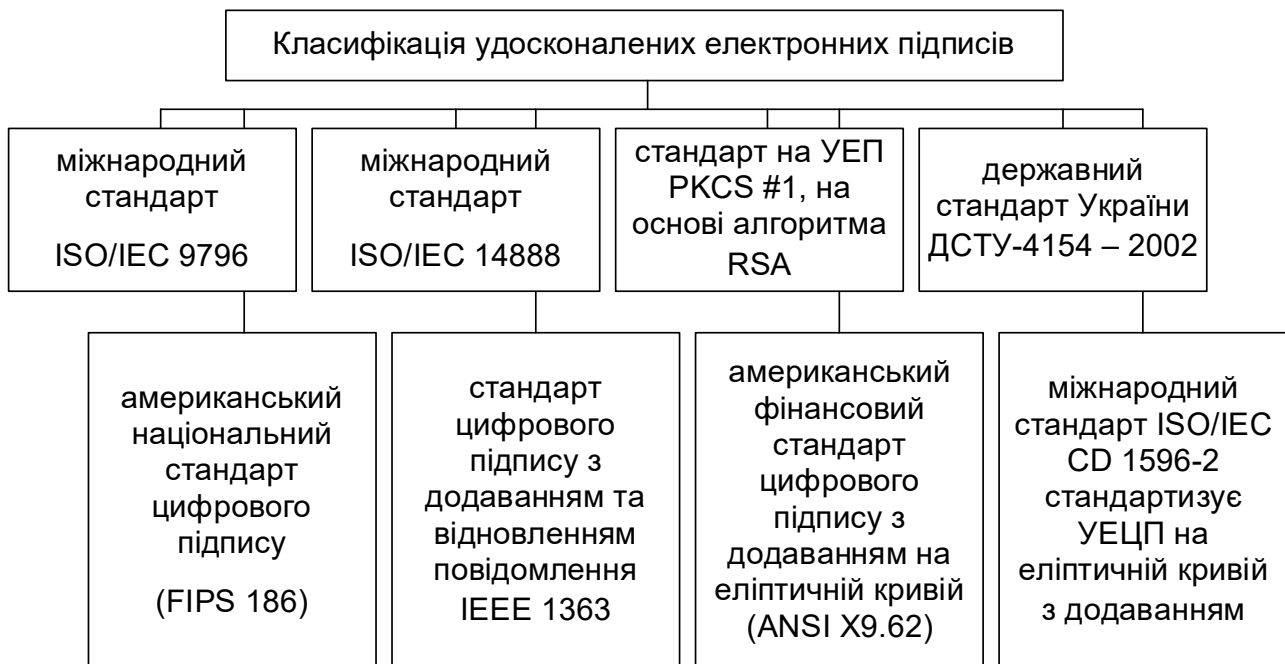


Рис. 2.54. Класифікація удосконалених електронних підписів

Для опису процесів обробки інформації з використанням механізмів УЕП скористаємося такою термінологією.

1. *Алгоритм генерації УЕП* – це метод формування УЕП.
2. *Алгоритм перевірки (верифікації) УЕП* – метод перевірки того, що підпис є автентичним, тобто дійсно створений конкретним об'єктом і не модифікований при передачі.
3. *Схема УЕП (або механізм УЕП)* – сукупність взаємозалежних алгоритмів генерації і верифікації електронного підпису.
4. *Процес (процедура) накладання УЕП* – сукупність математичного алгоритму генерації УЕП і методів представлення (форматування) даних, що підписуються.
5. *Процес (процедура) зняття УЕП* – сукупність алгоритму верифікації УЕП і методів відновлення даних.

Для побудови схеми УЕП необхідно визначити два алгоритми: алгоритм генерації УЕП і алгоритм верифікації УЕП. Алгоритм верифікації доступний для всіх потенційних одержувачів підписаних повідомлень, у той час як алгоритм генерації УЕП відомий тільки особі, яка підписує, що для деякого повідомлення $m \in M$ визначає відповідний підпис $s \in S$. Верифікатор, одержавши пари (m, s) і деяку відкриту інформацію про особу, що підписує, застосовує відповідний алгоритм верифікації УЕП. Даний алгоритм видає двійковий результат: “так”, якщо підпис правильний (автентична) і “ні” – у протилежному випадку.

Існуючі на сьогоднішній день схеми УЕП діляться на два класи (рис. 2.55):

схеми УЕП із відновленням повідомлення;
схеми УЕП із додаванням.

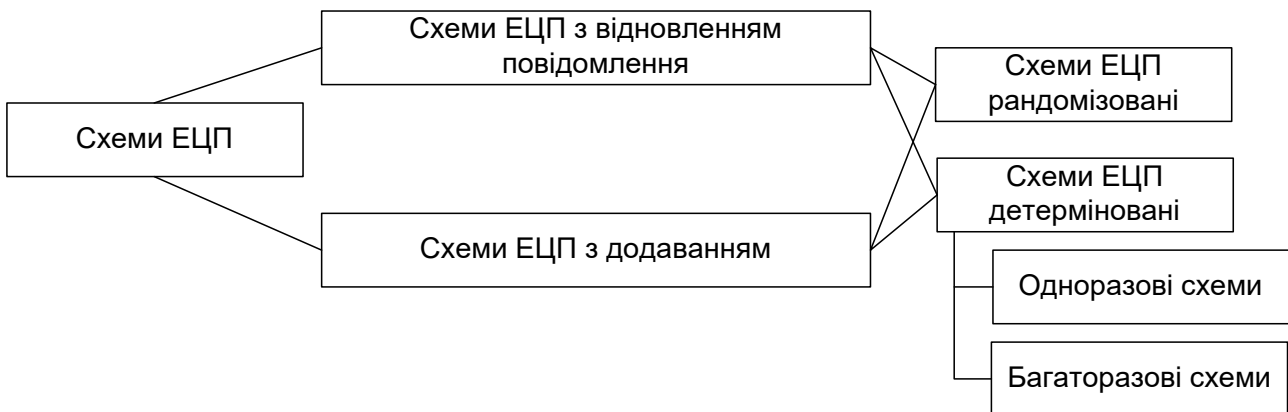


Рис. 2.55. Класифікація схем УЕП

У схемах УЕП із відновленням повідомлення всі або частина підписаного повідомлення може бути відновлена безпосередньо з електронного підпису. Таким чином, на вхід алгоритму верифікації надходить лише електронний підпис s .

У схемах УЕП із додаванням електронний підпис приєднується до повідомлення й у такому вигляді відправляється адресату. Для верифікації такого УЕП необхідно мати і підпис s , і відповідне повідомлення m .

Кожна з цих схем може являтися *детермінованою* або *рандомізованою*. Застосування детермінованих схем характеризується тим, що електронний підпис одного і того ж вхідного рядка даних призводить до формування однакових електронних підписів. У рандомізованій схемі при генерації підпису використовується деякий випадковий параметр (число), що призводить до формування різних підписів для однакових вхідних рядків (при використанні тих самих ключів). У рандомізованих схемах необхідно забезпечити непередбачуваність випадкових чисел.

У свою чергу детерміновані схеми діляться на схеми УЕП одноразового застосування (one-time) і схеми УЕП багаторазового застосування (multiple-use).

Модель електронного підпису з додаванням

Схема УЕП із додаванням є найбільш поширеною схемою в практичних додатках. До цього класу УЕП відносяться такі схеми підпису, як DSA, ElGamal, Schnorr, ДСТ 34.10. Розглянемо модель схеми УЕП із додаванням.

Нехай необхідно підписати деяке повідомлення довільної довжини $m \in M$, де M – простір повідомлень. Попереднє повідомлення m ґешується з використанням однобічної і вільної від колізій ґеш-функції:

$$\tilde{m} = h(m); \quad \tilde{m} \in M_h,$$

де M_h – простір ґеш-кодів.

Нагадаємо, що властивість однобічності означає, що за даним довільним рядком y (ґеш-кодом) обчислювально неможливо знайти двійковий рядок x такий, що $h(x) = y$, незважаючи на те, що такий рядок існує і у загальному випадку не один. Властивість вільності від колізій означає, що обчислювально неможливо знайти два рядки $x \neq x'$ таких, що $h(x) = h(x')$, незважаючи на те, що такі пари рядків існують.

Використовуючи особистий ключ $k \in K$, де K – простір ключів $\tilde{m}$, обраний алгоритм генерації підпису $SIG(\bullet)$ і ґеш-код повідомлення, відправник генерує підпис $s \in S$, де S – простір підписів:

$$s = SIG_k(\tilde{m}).$$

Повідомлення m і підпис, що додається до нього, s відправляється одержувачу.

Одержувач верифікує підпис таким чином: він одержує у своє розпорядження автентичну копію ключа верифікації $K_v \in K'$. Потім за отриманим повідомленням m обчислює ґеш-код $\tilde{m} = h(m)$ і, використовуючи алгоритм верифікації $VER(\bullet)$, приймає рішення відносності істинності або хибності підпису:

$$VER_{K_v}(\tilde{m}, s) \rightarrow \{\text{істина, хибність}\}.$$

До УЕП із додаванням пред'являються такі загальні вимоги:

- 1) для будь-якого ключа $k \in K$ алгоритм генерації $SIG_k(\bullet)$ повинен ефективно обчислюватися;
- 2) ключ верифікації підпису $K_v \in K'$ повинен бути захищений від підробки і для будь-якого K_v алгоритм верифікації підпису $VER_{K_v}(\bullet)$ також повинен ефективно обчислюватися;

3) будь-якому суб'єкту, за винятком відправника, обчислювально неможливо знайти $m' \neq m$, $m' \in M$ і $s' \neq s$, $s' \in S$ такі, що застосування алгоритму верифікації $VER_{k_v}(\tilde{m}, s')$, де $\tilde{m} = h(m')$ дасть значення "істина".

На рис. 2.56 наведена схема моделі УЕП із додаванням.

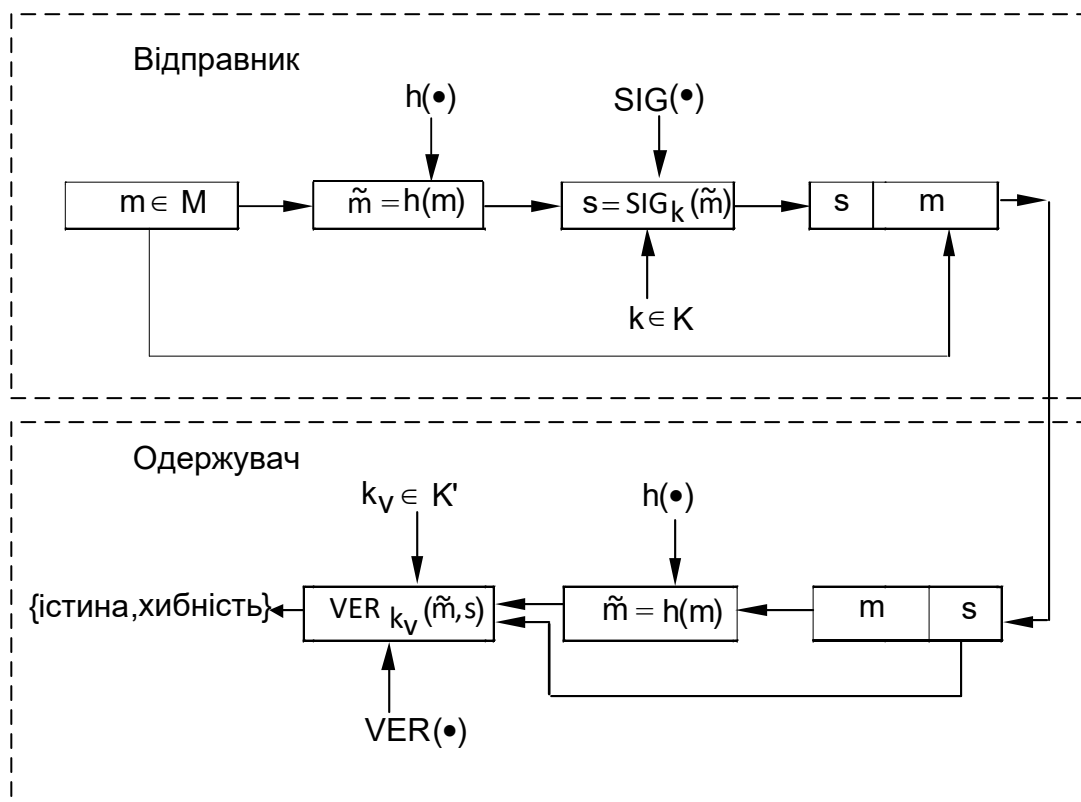


Рис. 2.56. Електроний підпис з додаванням

Модель електронного підпису з відновленням повідомлення

У схемах УЕП із відновленням повідомлення, що підписується, може бути відновлене безпосередньо з підпису.

На практиці УЕП із відновленням використовуються для роботи з короткими повідомленнями. Прикладами таких схем є схеми підпису RSA, Rabin, Nuberg-Rueppel (*NR*-схема).

Нехай відправник формує електроний підпис $s \in S$ для деякого повідомлення $m \in M$. При цьому надалі одержувач може відновити повідомлення m з s . Спочатку в повідомлення m вноситься надмірність шляхом застосування функції надмірності R :

$$\tilde{m} = R(m); \quad \tilde{m} \in M_R,$$

де M_R – простір надлишкових повідомлень або простір надмірності.

Потім формується електронний підпис відповідно до виразу

$$s = \text{SIG}_k(\tilde{m}).$$

Електронний підпис s відправляється одержувачу.

Для верифікації одержувач повинен одержати автентичний ключ верифікації $k_v \in K'$ відправника і обчислити надлишкове повідомлення з підпису відповідно до виразу:

$$\tilde{m} = \text{VER}_{k_v}(s).$$

У випадку, якщо $\tilde{m} \notin M_R$, то підпис відкидається як недійсний. Після цієї перевірки відновлюється оригінальне повідомлення m з $\tilde{m}$ шляхом обчислення $R^{-1}(m)$.

На рис. 2.57 подані процедури накладання і верифікації УЕП із відновленням повідомлення.

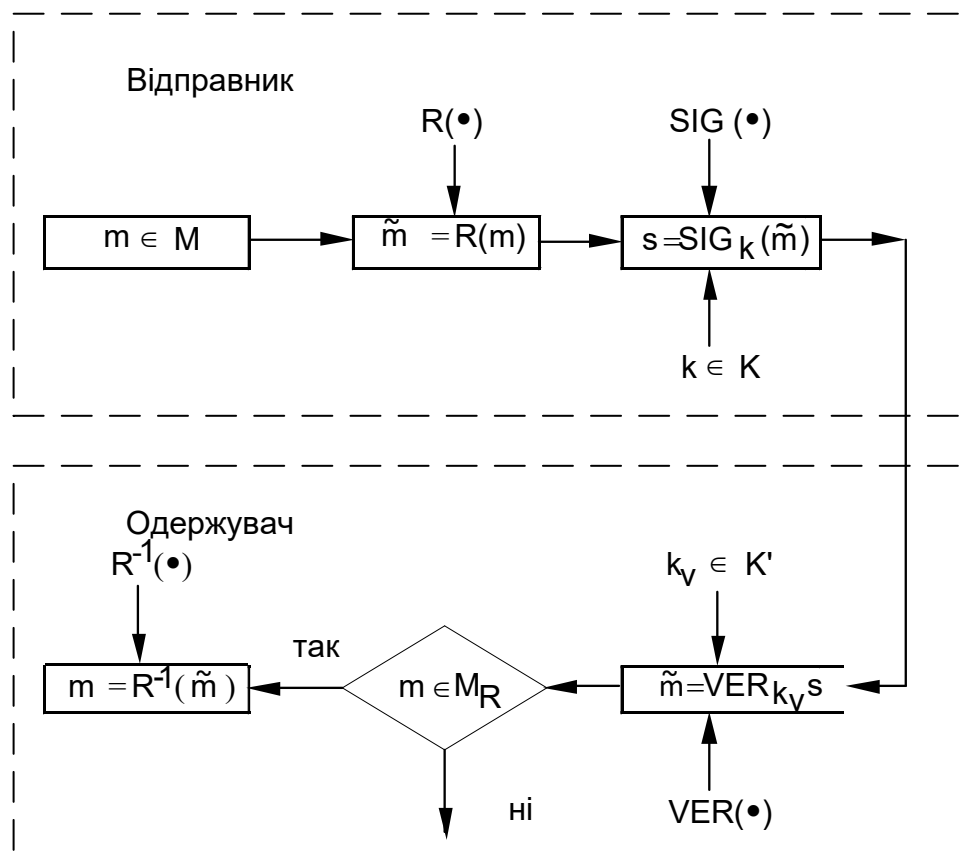


Рис. 2.57. Електронний підпис з відновленням повідомлення

До УЕП із відновленням пред'являються наступні вимоги:

1) для будь-якого ключа $k \in K$ алгоритм генерації $\text{SIG}_k(\bullet)$ повинен ефективно обчислюватися;

2) ключ верифікації підпису $k_v \in K'$ повинен бути захищений від підробки і для будь-якого k_v алгоритм верифікації підпису $VER_{k_v}(\bullet)$ також повинен ефективно обчислюватися;

3) будь-якому суб'єкту, за винятком відправника, обчислювально неможливо знайти $s' \neq s$, $s' \in S$ таке, що $VER_{k_v}(s) \in M_R$.

У електронному підписі з відновленням функція надмірності R і її інверсія R^{-1} є відкритими. Вибір відповідної функції R є критичною задачею з погляду забезпечення необхідної стійкості схеми УЕП. У УЕП із відновленням повідомлення потужності простору повідомлень $|M|$ менше потужності простору підписів $|S|$. Якби $|M| = |S|$, то пошук підпису s за відповідним m був би тривіальний. Функція надмірності здійснює зворотне відображення повідомлення m у надлишкове повідомлення $\tilde{m} \in M_R$. У свою чергу алгоритм генерації підпису $SIG_k(\bullet)$ застосовується до простору повідомлень, що підписуються M_S , причому $|M_S| > |M_R|$, тобто $M_R \subseteq M_S$. Вважається, що вдала функція надмірності повинна забезпечувати співвідношення потужностей таке, що $|M_R|/|M_S| = \left(\frac{1}{2}\right)^n$, де n – довжина повідомлення, яка підписується. У схемах УЕП із відновленням повідомлення функція надмірності є засобом виявлення модифікації електронного підпису, тобто є аналогом ґеш-функції, тільки зі зворотним знаком.

Функція надмірності для будь-якого повідомлення m породжує і вводить унікальну надмірність, що залежить від змісту інформації в повідомленні. Функція R повинна легко обчислюватися і найчастіше залежить від типу алгоритму генерації підпису.

Наприклад, функція введення надмірності, запропонована в міжнародному стандарті ISO/IEC 9796, застосовується тільки з перетвореннями RSA і Rabin.

Будь-яка схема з відновленням повідомлення може бути перетворена в схему УЕП із додаванням шляхом ґешування повідомлення, а потім застосування схеми УЕП із відновленням повідомлення до отриманого ґеш-коду. У цьому випадку алгоритм верифікації вимагає наявності і самого повідомлення.

Процедура накладання підпису для такого варіанта подана на рис. 2.58.

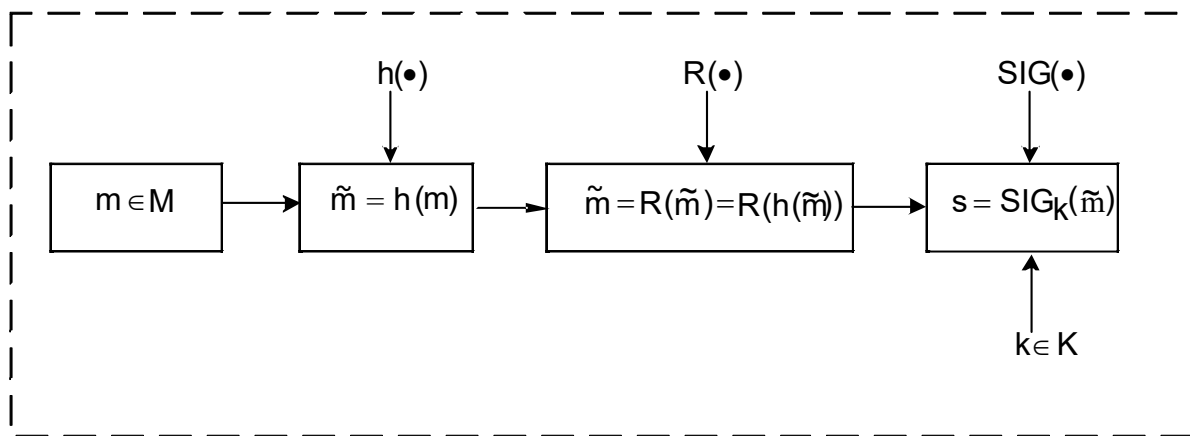


Рис. 2.58. **Комбінований електроний підпис**

У даному випадку до функції надмірності R пред'являються не такі жорсткі вимоги і вона може бути просто взаємно однозначною функцією відображення з множини M_h у множину M_s .

Електроні підписи з додаванням

Механізми електронного підпису з додаванням є об'єктом декількох стандартів. На міжнародному рівні ці механізми стандартизує стандарт ISO/IEC 14888. Американський національний стандарт FIPS 186 і російський національний стандарт ГОСТ 34.310 стандартизують схеми УЕП із додаванням, які є варіантами схеми El Gamal. Стандартизацією схем УЕП займаються і дослідницькі організації. Тут виділяється стандарт IEEE P1363, який визначає УЕП на основі несиметричної криптографії і промисловий стандарт PKCS № 1.

Стандарт ISO/IEC 14888 складається з трьох частин. Перша частина стандарту ISO/IEC 14888-1 уводить загальну модель схеми УЕП із додаванням.

Друга частина визначає схеми підпису на основі використання ідентифікаційної інформації.

Нарешті, у ISO/IEC 14888-3 описується два типи схем. По-перше, це схеми на основі використання проблеми дискретного логарифма, а саме: схема DSA, схема Pointcheval-Vaudenay і схема ECDSA (схема DSA на еліптичних кривих). По-друге, це схеми, стійкість яких заснована на проблемі факторизації. У стандарті подана: схема підпису за ISO/IEC 9796-1 з використанням ґешування і схема ESIGN.

Розглянемо більш докладно перераховані вище схеми.

Схеми підпису на основі використання ідентифікаційної інформації. Схема підпису Гіллоу-Куїскуотера (GQ-схема)

У схемах формування підпису на основі ідентифікаційної інформації відкритий ключ перевірки підпису формується з використанням ідентифікаційної інформації сторони, яка підписує. Таким чином, відпадає необхідність використання сертифікатів відкритих ключів. Особисті (секретні) ключі генерації підпису повинна генерувати довірча третя сторона. У таких схемах ДТС матиме доступ до всіх особистих ключів. У зв'язку з цим схема формування підпису на основі використання ідентифікаційної інформації може використовуватися не для всіх додатків. Найчастіше вона реалізується в закритих доменах безпеки, наприклад, усередині великої компанії або корпоративних закритих мереж, де існує “природна” довірча сторона.

ISO/IEC 14888-2 визначає три схеми підпису, що відносяться до класу рандомізованих схем і є різними варіантами схеми підпису Гіллоу – Куїскуотера (Guillou-Quisquater). В основу GQ-схеми підпису покладено використання ідентифікаційного протоколу GQ. Схема підпису, визначена в ISO/IEC 14888-2, допускає обов'язкове залучення ДТС.

ДТС генерує особисті ключі підпису, у той час як ключ верифікації може генеруватися верифікатором.

Генерація ключів для GQ-схеми підпису.

У результаті генерації ключів утворюється відкритий ключ (n, e, I_A) і відповідний особистий ключ a .

Довірча третя сторона виконує такі дії:

1. Обрати випадкові різні прості числа p і q , які зберігаються в секреті, і формує модуль $n = p \times q$.
2. Обрати ціле $e \in \{1, 2, \dots, n - 1\}$ таке, що $\text{НСД}(e, (p - 1)(q - 1)) = 1$.
3. Будь-яка сторона, що бажає використовувати GQ-схему підпису, надає ДТС унікальну ідентифікаційну інформацію у вигляді двійкового рядка I .

Ціле число I_A , $1 < I_A < n$ є ідентифікатором сторони A (тобто містить інформацію про ім'я, адресу, номер паспорта, PIN і т. д.). ДТС на основі інформації I_A формує величину

$$Y = f(I_A),$$

де f – функція введення надмірності (ISO/IEC 9796-1) така, що НСД (Y, n) = 1.

4. ДТС визначає ціле число $a \in Z_n$ таке, що $Ya^e \equiv 1 \pmod{n}$ відповідно до наступного алгоритму.

4.1. Обчислення $Y^{-1} \pmod{n}$.

4.2. Обчислення $d_1 = e^{-1} \pmod{p-1}$ і $d_2 = e^{-1} \pmod{q-1}$

4.3. Обчислення $a_1 = (Y^{-1})^{d_1} \pmod{p}$ і $a_2 = (Y^{-1})^{d_2} \pmod{q}$.

4.4. Пошук значення a , що задовольняє рівнянням:

$$a = a_1 \pmod{p};$$

$$a = a_2 \pmod{q}.$$

За відкритий ключ сторони А приймається вектор (n, e, I_A) . За особистий ключ сторони А приймається число a .

Процедури накладання і зняття GQ-підпису.

Процедура накладання GQ-підпису містить у собі виконання 3 кроків. Сторона А підписує двійкове повідомлення m довільної довжини. Для цього сторона А повинна:

1. Вибрати випадкове число k , $1 \leq k \leq n - 1$, і обчислити $r = k^e \pmod{n}$.

2. Обчислити ґеш-код $v = h(m||r)$.

Значення ґеш-коду повинно задовольняти нерівності $0 < v < e - 1$. Якщо v не задовольняє нерівності, то значення v приводять за модулем e .

3. Обчислити $s = r \cdot a^v \pmod{n}$.

Процедура зняття GQ-підпису. Для того щоб верифікувати електронний підпис (s, v) за повідомленням m , сторона В повинна одержати повідомлення m , підпис (s, v) і автентичний відкритий ключ сторони А (n, e, I_A) . Далі сторона В повинна обчислити:

$$u = s^{eI_A^v} \pmod{n};$$

$$v' = h(m||n).$$

Якщо $v = v'$, підпис вважається дійсним. Справедливість цього твердження впливає з того, що:

$$u \equiv s^{eI_A^v} \equiv (ka^v)^{eI_A^v} \equiv k^e(a^{eI_A})^v \equiv k^e \equiv r \pmod{n}.$$

Отже, $u = r$, звідси, $v = v'$.

Стійкість схеми GQ-підпису опирається на складність задачі факторизації складового модуля. Сучасні методи розкладання числа на складові множники і будуть визначати вимоги до параметрів схеми.

Останні досягнення в області факторизації говорять про те, що модуль n повинен бути, принаймні, 768-бітним. З метою забезпечення стійкості схеми УЕП до атак, що опираються на парадокс дня народження, довжина експоненти e повинна бути не менше 128 бітів. Розміри ґеш-кодів, що можуть застосовуватися в GQ-схемі підпису, рівні 128 або 160 бітам. Таким чином, при 768-бітному модулі n і 128-бітній експоненті e розмір відкритого ключа складе $896+u$ бітів, де u – кількість бітів, необхідна для представлення ідентифікатора I_A . Довжина особистого ключа a складе 768 бітів.

З погляду продуктивності GQ-схема вимагає виконання двох операцій зведення в степінь за модулем і однієї операції модульного множення. При використанні 768-бітного модуля n 128-бітної величини e і 128-бітного ґеш-коду v генерація підпису потребуватиме виконання приблизно такої ж кількості операцій. При схожих початкових умовах схема підпису RSA вимагає виконання 1152 модульних множень, а схема Фейга-Фіата-Шаміра (Feige-Fiat-Shamir) або схема FFS – 64 модульні множення. Але порівняно з останньою GQ-схема вимагає значно меншого місця для збереження ключів. (У схемі FFS довжина особистого ключа складе 98304 біти, а довжина відкритого ключа – 99072 біти).

Схеми підпису на основі використання властивостей дискретного логарифма

Третя частина стандарту ISO/IEC 14888-3 описує схеми підпису на основі використання дискретного логарифма (DSA і схема Pointcheval-Vaudenay), схеми DSA на еліптичних кривих – ECDSA і схеми підпису на основі використання проблеми факторизації; схеми підпису за ISO/IEC 9796-1 з ґешуванням і схеми підпису ESIGN.

Схема підпису DSA (Digital Signature Algorithm) була прийнята як федеральний стандарт США в 1993 році як складова частина стандарту електронного підпису DSS (Digital Signature Standard) FIPS 186. Стандарт FIPS 186 визначає використання ґеш-функції SHA-1 (FIPS 180-1, ISO/IEC 10118-3) разом з DSA.

DSA є розвитком схем електронних підписів Ель Гамала й К. Шнорра та УЕП із додаванням. Стійкість схеми підпису спирається на складність задачі взяття дискретного алгоритму.

Генерація ключів DSA.

У ході генерації ключів кожен об'єкт формує секретний ключ і відповідний особистий ключ.

Будь-який об'єкт А повинен виконати такі операції:

1. Обрати просте число q таке, що $2^{159} < q < 2^{160}$.
2. Обрати параметр t такий, що $0 \leq t \leq 8$ і обрати просте число p таке, що $2^{511+64t} < p < 2^{512+64t}$ і розкладання $(p - 1)$ має у своєму розкладанні простий співмножник q .
3. Визначити генератор (первісний елемент) Θ циклічної групи порядку q в Z_p^* . Для цього:

3.1. Обрати елемент $g \in Z_p^*$ і обчислити $\Theta = g^{(p-1)/q} \bmod p$.

3.2. Якщо $\Theta = 1$, то повернутися в 3.1.

4. Обрати випадкове ціле число a таке, що $1 \leq a \leq q - 1$.

5. Обчислити число $y = \Theta^a \bmod p$.

6. Відкритим ключем сторони А є вектор параметрів (p, q, Θ, y) , особистим ключем являється число a .

При формуванні простих чисел першим формується число q , а на його основі генерується число p . Алгоритм генерації простих чисел, що рекомендується викладений у FIPS 186.

Процедура накладання електронного підпису

Нехай об'єкт А бажає підписати двійкове повідомлення m довільної довжини. Для цього об'єкт А повинен виконати такі дії:

1. Обрати випадкове секретне ціле число k , $0 < k < q$. Для кожного знову обчисленого підпису обираються різні і непередбачені числа k . Таким чином, DSA – рандомізована схема.

2. Обчислити число $r = (\Theta^k \bmod p) \bmod q$.

3. Обчислити $k^{-1} \bmod q$.

4. За повідомленням m обчислити ґеш-код $h(m)$. Потім двійковий рядок $h(m)$ перетворити в ціле число.

5. Обчислити $s = k^{-1}\{h(m) + ar\} \bmod q$.

6. Підпис за повідомленням m є парою чисел (r, s) .

Процедура зняття електронного підпису

Для того щоб верифікувати електронний підпис (r, s) під повідомленням m , сторона В повинна виконати наступні дії.

1. Одержати автентичну копію відкритого ключа об'єкта А (p, q, Θ, y) .

2. Перевірити, що $0 < r < q$ та $0 < s < q$. У протилежному випадку підпис вважається недійсним.

3. Обчислити значення ґеш-коду $h(m)$ і

$$\begin{aligned}w &= s^{-1} \bmod q; \\u_1 &= w \cdot h(m) \bmod q; \\u_2 &= r \cdot w \bmod q; \\v &= (\Theta^{u_1} y^{u_2} \bmod p) \bmod q.\end{aligned}$$

4. Якщо $v = r$, то підпис приймається як дійсний.

Коректність підпису впливає з наступного. Якщо (r, s) є правильним підписом об'єкта A , тоді справедливе порівняння $h(m) \equiv -ar + ks \pmod{q}$.

Помноживши обидві частини порівняння на w , одержимо $w \cdot h(m) + arw \equiv k \pmod{q}$. Але з іншої сторони це порівняння не що інше як порівняння вигляду $u_1 + au_2 \equiv k \pmod{q}$.

Зведення первісного елемента Θ у степінь, показником якого виступають обидві частини порівняння, дає такий вираз:

$$(\Theta^{u_1} y^{u_2} \bmod p) \bmod q = (\Theta^k \bmod p) \bmod q.$$

Отже $v = r$, що і потрібно було довести.

Стійкість схеми DSA ґрунтується на проблемі дискретного логарифма в групі Z_p^* і циклічній підгрупі порядку q в Z_p^* .

Розмір числа q є фіксованим і становить 160 бітів, у той час як розмір числа p може приймати будь-яке значення, кратне 64 у межах від 512 до 1024 бітів. Використання 512-бітного числа є мінімально допустимою умовою реалізації схеми. Стандарти рекомендують застосовувати схему з довжиною модуля не менше 768 бітів. Стандарт FIPS 186 не дозволяє застосовувати прості числа p довжиною більше ніж 1024 бітів.

Процедура накладання підпису вимагає виконання однієї операції модульного зведення в степінь, який вимагає в середньому близько 240 модульних множень (при довжині модуля 768 бітів), однієї операції обчислення зворотного елемента з 160-бітним модулем, двох 160-бітних модульних множень і одного додавання. Основний внесок у часові витрати вносить операція модульного зведення в степінь. Однак перевагою DSA є можливість здійснення передобчислень. Випадкова величина k не залежить від повідомлення і може бути обрана заздалегідь.

Оскільки значення g є функцією від k , воно також може бути обчислене заздалегідь. Аналогічно можна знайти значення ag і k^{-1} . Таким чином, підпис може бути згенерований досить швидко. Усе що потрібно для генерації

підпису – це обчислити ґеш-код $h(m)$ + ag і здійснити модульне множення результату на k^{-1} . Основний обчислювальний внесок при знятті підпису виконується в дві операції: зведення в степінь за модулем p і з 160-бітним показником степеня. У середньому на виконання кожної з цих операцій буде потрібно 240 модульних множень. У цьому DSA уступає RSA, у якій як секретний ключ можливе використання малих показників, що значно спрощує верифікацію. Зовсім не обов'язково, щоб кожен об'єкт мав свої особисті числа p і q . Стандарт DSS дозволяє використання p , q і G як системних параметрів, хоча це більш привабливий варіант для противника.

У процесі накладання підпису необхідно перевіряти умову, що $s \neq 0$ і $r \neq 0$. При $s = 0$ процес верифікації буде неможливий, оскільки для такого s не існує зворотного елемента $s^{-1} \bmod q$. Якщо в процесі накладання підпису з'ясувалося, що $s = 0$ або $r = 0$, то необхідно повторити процедуру накладання підпису з новим значенням k .

Схема ESIGN (Efficient digital SIGNature) запропонована корпорацією Nippon (Японія) і є схемою, чия стійкість заснована на складності розв'язання задачі факторизації великого цілого числа.

Генерація ключів

Кожен об'єкт створює відкритий ключ і відповідний закритий ключ. Для цього об'єкт A повинен виконати такі операції.

1. Обрати два великих простих числа p і q таких, що $p \geq q$ і p, q приблизно рівних за довжиною.
2. Обчислити модуль $n = p^2q$.
3. Обрати позитивне ціле число $k \geq 4$.
4. Відкритим ключем сторони A є пара чисел (n, k) , закритим ключем – (p, q) .

Процедура накладання підпису

Алгоритм генерації підпису обчислює ціле число s таке, що $s^k \bmod n$ лежить у визначеному інтервалі, обумовленому повідомленням m . Для того, щоб підписати повідомлення m , що є двійковим рядком довільної довжини, об'єкт A повинен виконати такі дії.

1. Обчислити ґеш-код $v = h(m)$.
2. Обрати випадкове секретне ціле число x , $0 \leq x \leq pq$.
3. Обчислити:

$$\begin{aligned} w &= \lceil ((v - x^k) \bmod n) / (p \cdot q) \rceil; \\ y &= w \cdot (kx^{k-1})^{-1} \bmod p; \\ s &= x + ypq \bmod n. \end{aligned}$$

4. Електроним підписом повідомлення m є число s .

Процедура верифікації підпису

У ході верифікації демонструється, що величина $s^k \bmod n$ належить визначеному інтервалу. Для цього об'єкт В повинен виконати такі дії:

Одержати автентичний відкритий ключ (n, k) сторони А.

Обчислити:

$$\begin{aligned} u &= s^k \bmod n; \\ z &= h(m). \end{aligned}$$

Прийняти підпис, якщо:

$$z \leq u \leq z + 2^{\left\lceil \frac{2}{3} \lg n \right\rceil}.$$

У протилежному випадку підпис вважається недійсним.

Коректність підпису впливає з доказу.

Зазначимо, що:

$$s^k \equiv (x + ypq)^k \equiv \sum_{i=0}^k \binom{k}{i} x^{k-i} (ypq)^i \equiv x^k + kypqx^{k-1} \pmod{n}.$$

Але $kx^{k-1}y \equiv w \bmod p$ і, таким чином

$kx^{k-1}y \equiv w + lp$ для деяких $l \in \mathbb{Z}$.

Отже,

$$\begin{aligned} s^k &\equiv x^k + pq(w + lp) \equiv x^k + pqw \equiv x^k + pq \left\lceil \frac{(h(m) - x^k) \bmod n}{pq} \right\rceil \equiv \\ &\equiv x^k + pq \left(\frac{h(m) - x^k + jn + \varepsilon}{pq} \right) \pmod{n}, \text{ де } \varepsilon = (x^k - h(m)) \bmod pq. \end{aligned}$$

Звідси $s^k \equiv x^k + h(m) - x^k + \varepsilon \equiv h(m) + \varepsilon \pmod{n}$.

Оскільки $0 \leq \varepsilon < pq$, то

$h(m) \leq s^k \bmod n \leq h(m) + pq \leq h(m) + 2^{\lceil \frac{2}{3} \lg n \rceil}$, що і потрібно було довести.

У схемі підпису ESIGN використовується модуль $n = p^2q$, що відрізняється від RSA-модулів. Розроблювачі ESIGN стверджують, що за стійкістю дана схема не поступається схемам RSA і DSA. Однак не відомо, наскільки простіше або складніше розкладання модуля $n = p^2q$ на складові співмножники порівняно зі звичайним модулем $n = pq$.

Специфічний доказ коректності підпису ESIGN дозволяє реалізувати наступні атаки.

Для даного правильного підпису s за повідомленням m противник може підробити підпис для іншого повідомлення m' , якщо ґеш-код $h(m')$ приймає значення таке, що

$$h(m') \leq u \leq h(m') + 2^{\lceil \frac{2}{3} \lg n \rceil},$$

де $u = s^k \bmod n$.

Якщо буде знайдене повідомлення m' , що володіє такими властивостями, то підпис s буде дійсним і для нього. Така ситуація можлива, якщо $h(m)$ і $h(m')$ збігаються в $(\lg n)/3$ старших значущих бітах. Припускаючи, що h має властивості випадкової функції, противнику для пошуку такого повідомлення необхідно випробувати $2^{(\lg n)/3}$ варіантів.

Інший можливий варіант підробки підпису полягає в перебуванні пари повідомлень m і m' таких, що $h(m)$ і $h(m')$ збігатимуться в старших $(\lg n)/3$ бітах. Відповідно до парадокса дня народження цього необхідно випробувати $O(2^{(\lg n)/6})$ варіантів. Якщо противник здатний сформувати коректний підпис для повідомлення m , то цей же підпис буде дійсним і для повідомлення m' .

Розглянуті вище приклади атак і визначають вимоги до величини модуля.

Схема підпису ESIGN з погляду продуктивності перевершує схему підпису RSA. Крім того, схема ESIGN дозволяє виконувати перед-обчислення, що сприяють збільшенню обчислювальної ефективності алгоритму. Після вибору випадкового числа x сторона А може здійснити наступні передобчислення:

$$u_1 = x^k \bmod n;$$

$$u_2 = 1/(kx^{k-1}) \bmod p.$$

Потім визначити:

$$w = \lceil w - u_1/(pq) \rceil;$$

$$s = x + (wu_2 \bmod p) \cdot p \cdot q.$$

Використання передобчислювань дозволяє збільшити швидкість формування підпису до 10 разів. На ефективність обчислень підпису впливає значення параметра k . Розроблювачі рекомендують вибирати його з множини $k = \{4, 8, 16, 32, 64, 128, 256, 512, 1024\}$.

Так для $k = 4$ і 768-бітний модулі n генерації підпису ESIGN перевершує за швидкістю генерацію підпису RSA від 10 до 100 разів. Процедура верифікації також досить ефективна і порівняна зі схемою RSA з малими відкритими ключами. Даний алгоритм може бути реалізований і на еліптичних кривих.

Електроні підписи з відновленням повідомлення

Схеми УЕП із відновленням повідомлення є об'єктом стандартизації міжнародного стандарту ISO/IEC 9796. Даний міжнародний стандарт складається з трьох частин:

- ISO/IEC 9796-1 стандартизує схему УЕП на основі RSA-перетворення;
- ISO/IEC 9796-2 стандартизує схему УЕП аналогічну схемі, визначеній в першій частині, але використовуючи для введення надмірності ґеш-функцію. У даному варіанті схеми УЕП забезпечується часткове відновлення повідомлення і схема може бути використана для підпису повідомлень довільної довжини;
- ISO/IEC 9796-3 описує схему, в якій як алгоритм генерації підпису використовується дискретний логарифм (схема ElGamal).

Схема підпису за ISO/IEC 9796-1

Загальна ідея застосування електронного підпису з відновленням повідомлення полягає в тому, що вихідне коротке повідомлення подовжується, потім у нього вводиться надмірність, після чого воно

підписується. Схема УЕП, визначена в ISO/IEC 9796-1, забезпечує необхідні показники стійкості тільки при дотриманні всіх обмежень, визначених у стандарті. Схема УЕП за ISO/IEC 9796-1 заснована на використанні криптографії з відкритими ключами з використанням схеми RSA або Rabin. Алгоритм підпису повинен відображати k -бітний вхідний рядок у k -бітний рядок підпису. Схема не вимагає застосування ґеш-функції і може використовуватися для підпису коротких повідомлень, що потім відновлюються з підпису.

При описі схеми УЕП використовуються такі позначення:

s – двійковий рядок підпису;

$d \leq 8 \lfloor (s+3)/16 \rfloor$ – двійковий рядок повідомлення m , яке підписується, обирається так, що ;

$z = \lceil d/8 \rceil$ – кількість байтів у доповненому повідомленні;

$r = 8z - d + 1$ – число, більше на одиницю кількості доповнених бітів;

$t = \lceil (s-1)/16 \rceil$ – найменше ціле число таке, що рядок з $2t$ байтів включатиме принаймні $s - 1$ біт.

Приклад. Нехай необхідно підписати повідомлення m довжиною 150 бітів. При цьому довжина підпису повинна дорівнювати 1024 бітам. Тоді наведені вище параметри приймуть такі значення: $s = 1024$ біти, $d = 150$ бітів, $z = 19$ байтів, $r = 3$ біти, $t = 64$ байти.

Процес накладання УЕП за ISO/IEC 9796-1

1. *Доповнення.* Нехай m – повідомлення, які необхідно підписати. Доповнення полягає у формуванні доповненого повідомлення MP (message padding) шляхом конкатенації ліворуч вихідного повідомлення m з r нульовими бітами:

$$MP = 0^{r-1} \| m, z,$$

де $1 \leq r \leq 8$ обирається таким, що кількість бітів у MP буде кратним восьми.

Кількість байтів у MP позначатимемо символом:

$$MP = m_z \| m_{z-1} \| \dots \| m_2 \| m_1,$$

де m_i – байт.

2. *Розширення повідомлення.* З доповненого повідомлення формується розширене повідомлення ME (message extension) шляхом

ітеративної конкатенації ліворуч доповненого повідомлення із самим собою.

Повторення конкатенації здійснюється доти, доки не буде сформований рядок $ME = ME_t || ME_{t-1} || \dots || ME_2 || ME_1$, який містить рівно t байтів. Якщо число t не кратне числу z , то останні приєднані байти будуть частиною множини байтів з MP , що є послідовними праворуч доповненого повідомлення, тобто:

$$ME_i = m_{(i \bmod z)+1}, \forall i = \overline{0, t-1}.$$

Процес накладання УЕП містить у собі виконання п'яти кроків (рис. 2.54).

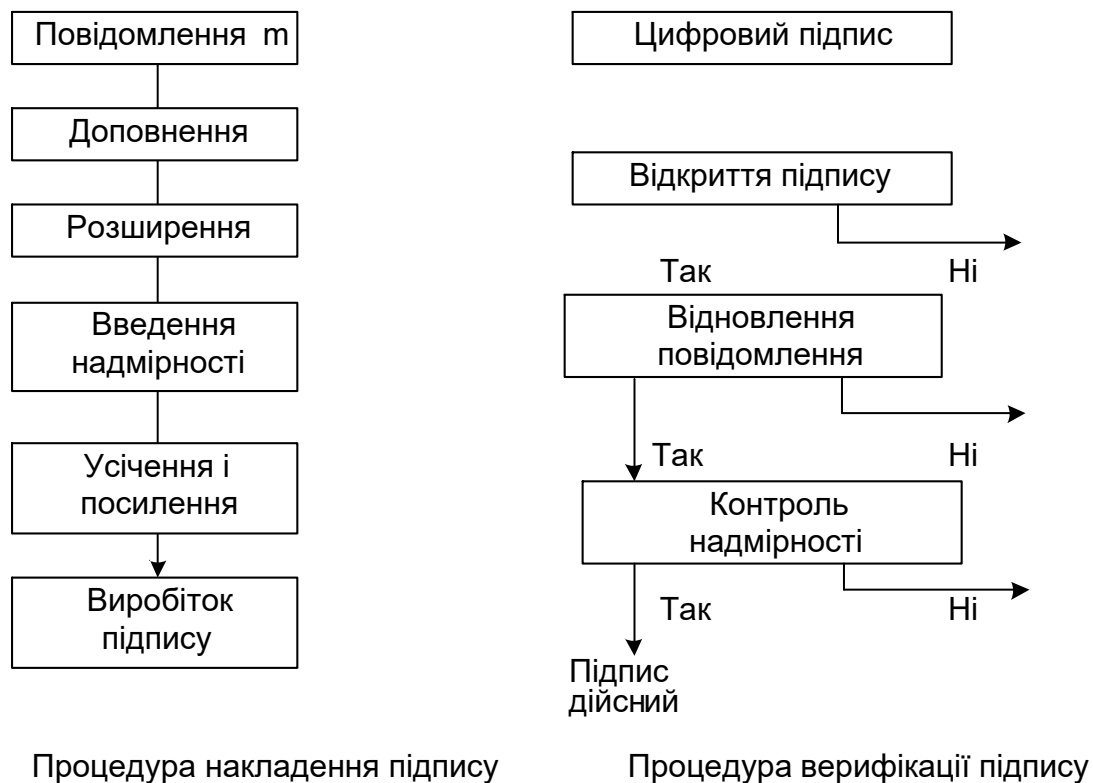


Рис.2.59. **Процедури накладання і верифікації підпису за ISO/IEC 9796-1**

3. Уведення надмірності. Надмірність вводиться в ME з метою одержання надлишкового повідомлення MR (message redundancy) вигляду $ME = ME_{2t} || ME_{2t-1} || \dots || ME_2 || ME_1$. Уведення надмірності здійснюється в такий спосіб. Надлишкове повідомлення формується шляхом перемноження t байтів розширеного повідомлення з t надлишковими байтами, з наступним узгодженням MR_{2z} -го байта

отриманого рядка. Усі байти надлишкового повідомлення формуються згідно з такими виразами:

$$MR_{2i-1} = ME_i, MR_{2i} = F(ME_i), \forall i = \overline{1, t}.$$

де $F(u)$ – функція перетворення байта u .

Функція $F(u)$ визначена в такий спосіб. Якщо байт $u = u_1 \| u_2$, де u_1 і u_2 – напівбайти, то

$$F(u) = \pi(u_2) \| \pi(u_1),$$

де π – перестановка вигляду:

$$\pi = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & A & B & C & D & E & F \\ E & 3 & 5 & 8 & 9 & 4 & 2 & F & 0 & D & B & 6 & 7 & A & C & 1 \end{pmatrix}.$$

Нарешті, останній байт MR формується шляхом заміни байта MR_{2z} на байт вигляду $r \oplus MR_{2z}$. Таке перетворення дозволяє верифікатору підпису відновити довжину повідомлення. Оскільки довжина повідомлення дорівнює $d = 8z - r + 1$, то для визначення значення d досить визначити значення z і r . Ці значення можуть бути відновлені з MR .

4. Усічення і посилення. На основі надлишкового повідомлення формується s -бітне проміжне ціле число IR . Формування IR здійснюється в такий спосіб:

а) до $s - 1$ молодшим значущим бітам надлишкового повідомлення ліворуч додається одиничний біт. Інші старші біти (якщо вони є) просто відкидаються;

б) здійснюється модифікація молодшого значущого байта $u_2 \| u_1$ шляхом заміни його на байт вигляду $u_1 \| 0110$.

Таким чином, $IR = 6 \pmod{16}$.

5. Генерація підпису. На цьому кроці формується електронний підпис шляхом застосування алгоритму генерації УЕП до рядка IR :

$$s = \text{SIG}_k(IR).$$

Рис. 2.60 пояснює процес усічення і посилення.

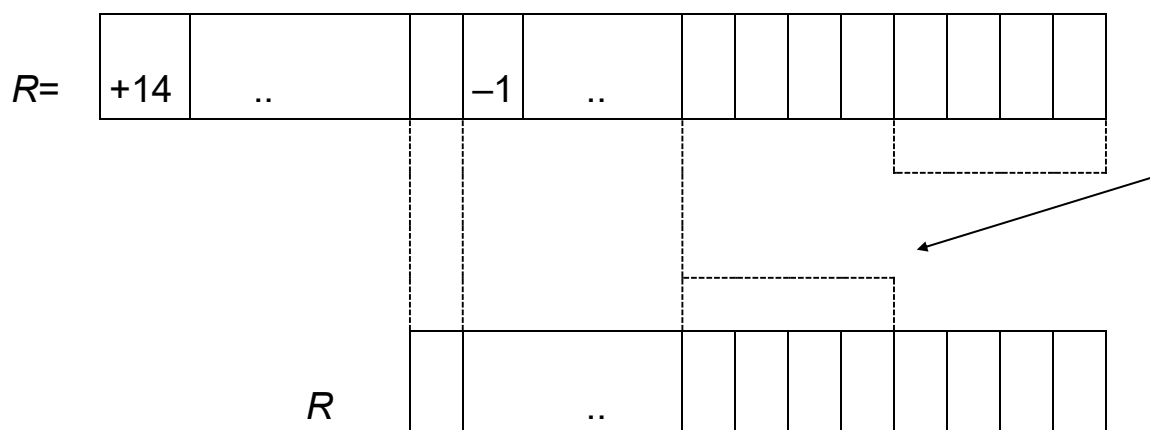


Рис. 2.60. Формування рядка *IR*

Схема УЕП ISO/IEC 9796-1 орієнтована на генерацію УЕП на основі алгоритмів RSA або Rabin. Розглянемо застосування даних алгоритмів.

Алгоритм генерації УЕП RSA

Спочатку сторона, яка підписується, формує необхідні ключі відповідно до наступного алгоритму.

1. Сформувати два великих відмінних один від одного простих числа p і q , приблизно однієї довжини. При цьому $p \not\equiv q \pmod{8}$.

2. Обчислити $n = pq$ і $\varphi(n) = (p - 1)(q - 1)$.

3. Вибрати випадкове ціле число v , при цьому $1 < v < \varphi(n)$;

$(e, p - 1) = 1$ і $(e, q - 1) = 1$ при e непарному;

$\left(e, \frac{p-1}{2}\right) = 1$ і $\left(e, \frac{q-1}{2}\right) = 1$ при e парному.

4. Використовуючи розширений алгоритм Евкліда, обчислити число v з рівнянь:

$$ve \equiv 1 \pmod{\varphi(n)}, \text{ при } e \text{ парному};$$

$$ve \equiv 1 \pmod{\varphi(n)/2}, \text{ при } e \text{ непарному}.$$

Відкритим ключем сторони, що підписує, є пара чисел (e, n) . Число e ще називають перевірочним компонентом або ключем верифікації. Особистим ключем є число d .

Обчислення підпису здійснюється в два етапи. Спочатку з рядка IR формується репрезентативний елемент RR. Формування RR здійснюється в такий спосіб:

- якщо e непарне, то $RR = IR$;
- якщо e парне і символ Якобі $J\left(\frac{IR}{n}\right) = 1$, то $RR = IR$;
- якщо e парне і символ Якобі $J\left(\frac{IR}{n}\right) = -1$, то $RR = IR/2$.

Зазначимо, що:

$$J\left(\frac{a}{n}\right) = \left(a^{(p-1)/2} \bmod p\right) \left(a^{(q-1)/2} \bmod q\right),$$

де $n = pq$.

Електроний підпис s обчислюється як

$$s = \min(RR^d \bmod n, n - (RR^d \bmod n)).$$

Алгоритм генерації УЕП Rabin

Схема Рабіна аналогічна схемі RSA, але як відкритий ключ e використовуються тільки парні числа і найчастіше $e = 2$.

Процес зняття УЕП за ISO/IEC 9796-1

Зняття електронного підпису здійснюється в три етапи (див. рис. 2.7).

- 1) відкриття підпису;
- 2) відновлення повідомлення;
- 3) контроль надмірності.

На кожному з цих трьох етапів підпис може бути відкинутий як недійсний, якщо він не пройде відповідної перевірки. У таких випадках обробка підпису припиняється з відповідним повідомленням при цьому зацікавлених сторін.

Розглянемо більш докладно кожний з цих етапів.

1. Відкриття підпису. На даному етапі здійснюється перетворення електронного підпису s у рядок IR' – відновлене проміжне ціле:

$$IR' = VER_{k_v}(s).$$

Для схеми RSA $k_v = e$; для схеми Rabin $k_v = 2$. Підпис відкидається, якщо рядок IR' не є s -бітним рядком зі старшим значущим бітом, рівним "1" і молодшими значущими бітами, рівними "0110". У випадку правильного УЕП справедлива рівність $IR = IR'$.

2. *Відновлення повідомлення.* У ході відновлення повідомлення з рядка IR' формується рядок надлишкового повідомлення MR' , що складається з $2t$ байтів. Відновлення здійснюється в такий спосіб:

а) нехай X буде рядком з $s - 1$ молодших значущих бітів рядка IR' ;

б) молодший значущий байт рядка X , що представляється як $u_1 \| u_3 \| u_1 \| 0110$, де u_i – напівбайт, замінюється відповідно до виразу $\pi^{-1}(u_4) \| u_1$;

в) відновлене повідомлення MR' формується шляхом доповнення ліворуч рядка X нульовими бітами (від 0 до 15 бітів), у результаті чого загальна довжина MR' складе $2t$ байтів.

Якщо електроний підпис правильний, то значення MR' буде збігатися із значенням рядка MR за винятком $16t - s + 1$ старших значущих бітів.

Далі здійснюється обчислення значення r і z ;

а) здійснюється порівняння парних і непарних байтів відновленого надлишкового повідомлення MR' . Для усіх $i = \overline{1, t}$ перевіряють виконання умови

$$MR'_{2i} \oplus F(MR'_{2i-1}) = 0.$$

Якщо для всіх $i = \overline{1, t}$ ця умова справедлива, то електроний підпис вважається недійсним;

б) нехай z – найменше позитивне ціле число i , при якому значення $f = MR'_{2i} \oplus F(MR'_{2i-1}) \neq 0$;

в) нехай r буде найменшим значущим напівбайтом числа f . Підпис вважається недійсним, якщо шістнадцятиричне значення r лежить поза інтервалом $1 \leq r \leq 8$.

Відновлене доповнене повідомлення MP' довжиною z байтів формується з MR' відповідно до виразів:

а) $MP'_{2i} = MR'_{2i-1}$ для усіх $i = \overline{1, z}$;

б) якщо $r - 1$ старших значущих бітів рядка MP' не дорівнюють нулю, то електроний підпис вважається недійсним.

Оригінальне повідомлення $M' \in 8z - r + 1$ молодших значущих бітів рядка MP' .

3. *Контроль надмірності.* Електроний підпис верифікується в такий спосіб:

а) з отриманого на попередньому етапі повідомлення M' формується рядок MR'' шляхом застосування операцій доповнення, розширен-

ня повідомлення і введення надмірності (аналогічно до процесу накладання електронного підпису);

б) електроний підпис вважається дійсним тільки тоді, коли $s - 1$ молодших значущих бітів рядка MR'' збігаються з $s - 1$ молодшими значущими бітами рядка MR' ;

Розглянута схема електронного підпису може бути модифікована в схему УЕП з додаванням i , отже, буде застосовна для підпису повідомлення довільної довжини. У цьому випадку можливе застосування однобічної вільної від колізій ґеш-функції.

Процедура накладання електронного підпису полягає в тому, що повідомлення довільної довжини m спочатку ґешується, а потім ґеш-код $h(m)$ є входом у процедуру накладання підпису за ISO/IEC 9796-1.

Електроний підпис з частковим відновленням повідомлення

Схема УЕП із частковим відновленням повідомлення визначена в ISO/IEC 9796-2 і має такі властивості.

По-перше, схема забезпечує часткове відновлення повідомлення і застосована для повідомлень довільної довжини. Тобто якщо повідомлення коротке, то воно може бути цілком відновлене з підпису. Якщо повідомлення занадто довге, тобто коли довжина повідомлення d не погодиться з довжиною підпису s , то частина повідомлення може бути відновлена з підпису, а інша частина повинна бути передана одержувачу будь-яким іншим способом.

По-друге, в раніше розглянутій схемі підпису для введення надмірності необхідно відводити не менше половини доступного простору блоку підпису. У ISO/IEC 9796-2 метод уведення надмірності допускає використання ґеш-функції, що дозволяє збільшити обсяг даних у блоці підпису. Так, якщо як алгоритм генерації підпису взяти 768-бітне перетворення RSA то, відповідно до процедури формування підпису за ISO/IEC 9796-1, безпосередньо для переданих даних буде доступно не більше 384 бітів. При застосуванні схеми, визначеної в другій частині стандарту, дозволяється використовувати для даних до 600 бітів.

Основна ідея схеми УЕП із частковим відновленням повідомлення полягає в тому, що:

1. Повідомлення m ґешується з використанням ґеш-функції, у результаті чого формується ґеш-код $h(m)$.

2. Якщо повідомлення занадто довге, щоб бути цілком включеним у підпис, то виділяється деяка його частина, що надалі буде відновлюватися з підпису.

3. До відновлюваної частини повідомлення додається прапорний біт (more data bit).

4. Відновлювана частина повідомлення, прапорний біт і ґеш-код разом з іншими бітами формату (наприклад, ідентифікатор ґеш-функції) конкатуються і є вхідними параметрами в алгоритм генерації підпису. Як алгоритм генерації підпису рекомендується використовувати схему RSA.

Схема електронного підпису Нуберга-Рюппела

Четверта частина стандарту ISO/IEC 9796-4 визначає схему Нуберга – Рюппела (Nyberg-Rueppel, NR-схема), у якій як алгоритм генерації підпису використовується варіант схеми підпису Ель Гамала. В даний момент запропоновані дві версії NR-схеми підпису: на основі обчислення дискретного логарифма в Z_p^* і на основі обчислення логарифма на еліптичних кривих.

NR-схема є рандомізованою схемою з відновленням повідомлення. Для даної схеми простір, що підписується, $M_s = Z_p^*$, де p – просте число. Простір підписів $S = Z_p \times Z_q$, де q – просте число і q ділить $(p - 1)$. Функція введення надмірності відображає множину повідомлень M у множину M_s . Алгоритм генерації ключів аналогічний алгоритму генерації ключів для схеми DSA, за винятком того, що на довжину чисел p і q не накладається ніяких обмежень.

Генерація ключів для NR-схеми підпису

Для формування відкритого і відповідного закритого ключа сторона A повинна виконати такі дії:

1. Вибрати прості числа p і q такі, що q ділить $(p - 1)$.
2. Визначити генератор Θ циклічної групи порядку q в Z_p^* . Для цього:
 - 2.1. Вибрати елемент $g \in Z_p^*$ і обчислити:

$$\Theta = g^{(p-1)/q} \bmod p.$$

- 2.2. Якщо $\Theta = 1$, повернутися в 2.1.

3. Вибрати випадкове ціле число a таке, що $1 \leq a \leq q - 1$.

4. Обчислити число $y = \Theta^a \bmod p$.

5. Відкритим ключем сторони A є вектор (p, q, Θ, y) , закритим ключем є число a .

Процедура накладання електронного підпису

Нехай сторона A бажає підписати повідомлення $m \in M$. Тоді необхідно:

1. Обчислити $\bar{m} = R(m)$ (уведення надмірності).

2. Вибрати випадкове секретне ціле число k , $1 \leq k \leq q - 1$ і обчислити параметр:

$$r = \Theta^{-k} \bmod p.$$

3. Обчислити:

$$e = \bar{m}r \bmod p;$$

$$s = ae + k \bmod q.$$

4. Електроним підписом повідомлення m є пари чисел (e, s) .

Процедура зняття електронного підпису

Для того щоб верифікувати УЕП і відновити повідомлення, об'єкт B повинен:

1. Одержати автентичну копію відкритого ключа (p, q, Θ, y) сторони A .

2. Перевірити, що $0 < e < p$. Якщо умова не виконується, то підпис відкидається.

3. Перевірити, що $0 < s < q$. Якщо умова не виконується, то підпис відкидається.

4. Обчислити:

$$v = \Theta^s y^{-e} \bmod p;$$

$$\tilde{m} = ve \bmod p.$$

5. Перевірити, що $\tilde{m} \in M_R$. Якщо $\tilde{m} \notin M_R$, то підпис відкидається.

6. Відновити повідомлення $m = R^{-1}(\tilde{m})$.

Коректність УЕП впливає зі справедливості наступного ствердження. Якщо абонент A сформував підпис, то

$$v \equiv \Theta^s y^{-e} \equiv \Theta^{s-ae} \equiv \Theta^k \pmod{p}.$$

У такий спосіб $ve \equiv \Theta^k \tilde{m} \Theta^{-k} \equiv \tilde{m} \pmod{p}$, що і потрібно було довести.

Стійкість NR -схеми ґрунтується на складності розв'язання задачі дискретного логарифма. Противник може спробувати підробити підпис шляхом вибору числа k і обчислення параметрів r . Така атака рівноцінна тривіальному випадковому підбору підпису s . Імовірність успіху такої

атаки складає $1/p$, що при великих значеннях p складає досить малу величину.

Для запобігання визначення особистого ключа необхідно на кожне повідомлення обирати різне k . Ці числа повинні формуватися з урахуванням спеціальних вимог, мати рівномірний закон розподілу і бути непередбачуваними. Для формування числа k можна використовувати генератор, визначений у ANSI X 9.17. Оскільки NR -схема підпису забезпечує відновлення повідомлення, то пред'являються тверді вимоги до функції введення надмірності $R(m)$. При виборі функції надмірності R необхідно враховувати мультиплікативну природу схеми підпису. Припустимо, що $m \in M$, $\tilde{m} = R(m)$ і (e, s) – електроний підпис для m . Тоді $e = \tilde{m}\Theta^{-k} \bmod p$ для деякого цілого k і $s = ae + k \bmod q$. Нехай $\tilde{m}^* = \tilde{m}\Theta^e \bmod p$ для деякого e . Якщо $s^* = s + e \bmod q$ і $\tilde{m}^* \in M_R$, то (e, s^*) є коректним підписом для нового повідомлення $m^* = R^{-1}(\tilde{m}^*)$. Це впливає з того, що:

$$\begin{aligned} v &\equiv \Theta^s y^{-e} \equiv \Theta^{s+e} \Theta^{-ae} \equiv \Theta^{k+e} \pmod{p}; \\ ve &\equiv \Theta^{k+e} \tilde{m} \Theta^{-k} \equiv \tilde{m} \Theta^e \equiv \tilde{m}^* \pmod{p}. \end{aligned}$$

Оскільки $\tilde{m}^* \in M_R$, то підроблений підпис (e, s^*) буде прийнятий як коректний для повідомлення m^* .

Нарешті, перевірка умови $0 < e < p$ є обов'язковою. Припустимо (e, s) є підписом сторони A для повідомлення m . Тоді $e = \tilde{m}r \bmod p$ і $s = ae + k \bmod q$. Противник може використовувати цей підпис для формування підпису під обраним ним самим повідомленням m^* . Він визначає e^* таке, що $e^* \equiv \tilde{m}^*r \pmod{p}$ і $e^* \equiv e \pmod{q}$. Якщо не перевірити умову, що $0 < e^* < p$, то пара (e, s^*) буде прийнята як справжній підпис.

Розглянемо основні стандарти УЕП, застосовані у комплексних системах захисту банківської інформації, проведемо зіставлення основних характеристик УЕП (довжину ключів, довжину електронного підпису, складність (час) обчислення і складність (час) перевірки дійсності електронного підпису) з метою виявлення їх переваг і недоліків за умови, що рівень стійкості підпису стосовно будь-яких методів фальсифікації не нижче, ніж 10^{21} (або 30 років безперервної роботи мережі з 1000 суперкомп'ютерів).

У якості “базової” довжини ключів і довжини самого електронного підпису будемо розглядати довжину в 64 байти.

Алгоритм RSA ґрунтується на *NP*-повній задачі факторизації числа (розкладання цілого параметра n у вигляді добутку двох різних простих чисел приблизно рівних один по одному величини, тобто $n = p \times q$ на ці прості множники). За сучасними оцінками складність задачі розкладання на прості множники при цілих числах n з 64 байтів становить порядку 10^{17} – 10^{18} операцій, тобто перебуває десь на грані досяжності для серйозного “противника”. Тому звичайно в системах електронного підпису на основі алгоритму RSA застосовують більш довгі цілі числа n (звичайно від 75 до 128 байтів). Це відповідно приводить до збільшення довжини самого електронного підпису відносно 64-байтного варіанта приблизно на 20 – 100 % (у цьому випадку її довжина збігається з довжиною запису числа n), а також від 70 до 800 % збільшує час обчислень при підписуванні і перевірці.

Крім того, при генерації і обчисленні ключів у системі RSA необхідно перевіряти велику кількість досить складних додаткових умов на прості числа p і q , а невиконання кожного з них може уможливити фальсифікацію підпису з боку того, хто виявить невиконання хоча б однієї із цих умов (при підписуванні важливих документів допускати, навіть теоретично, таку можливість небажано) (рис. 2.61).

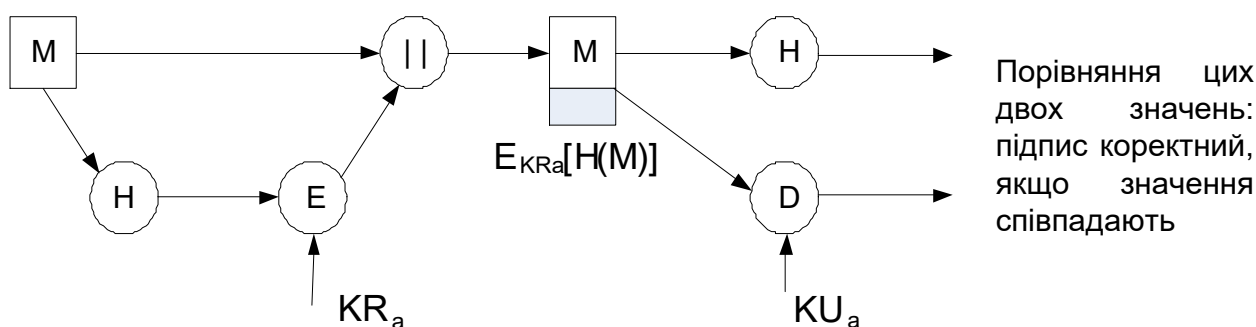


Рис. 2.61. Створення і перевірка підпису за алгоритмом RSA

Алгоритм EGSA [41]. Істотним кроком уперед у розробці сучасних алгоритмів електронного підпису був новий алгоритм Ель Гамала. У цьому алгоритмі ціле число n покладається рівним спеціально обраному великому простому числу p , за модулем якого і виробляються всі обчислення.

Такий вибір дозволяє підвищити стійкість підпису при ключах з 64 байтів приблизно в 1000 разів, тобто при такій довжині ключів забезпечується необхідний нам рівень стійкості порядку 10^{21} .

При цьому довжина самого електронного підпису збільшується у два рази і становить 128 байтів. Головна “заслуга” алгоритму Ель Гамала полягала в тому, що надалі він послужив основою для прийняття декількох стандартів електронного підпису, у тому числі національного стандарту США DSS і державного стандарту РФ ГОСТ Р-34.10-2001.

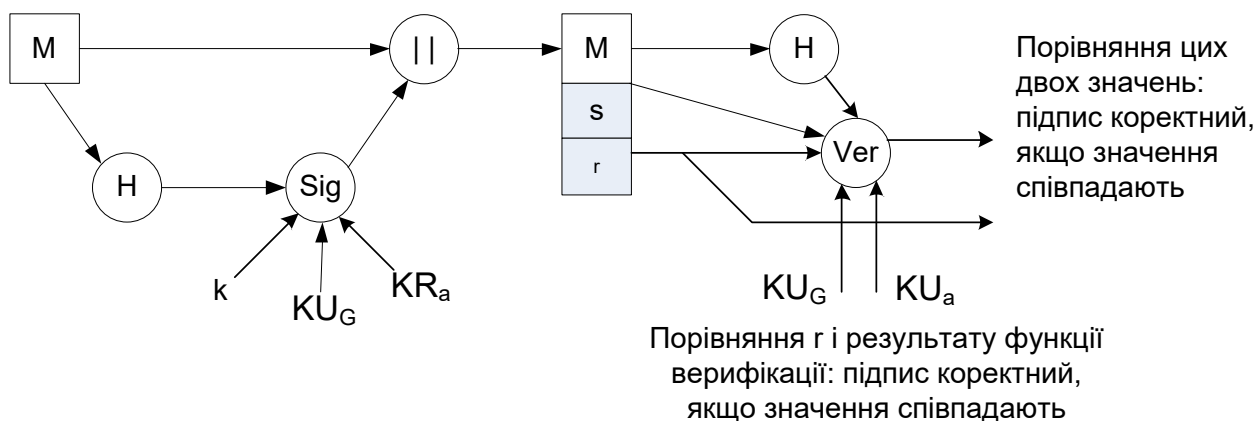


Рис. 2.62. Створення і перевірка підпису за алгоритмом DSA

Алгоритм DSA. Алгоритм DSA, що став надалі основою національного стандарту США на електронний підпис, має порівняно з алгоритмом RSA цілий ряд переваг:

- при заданому рівні стійкості електронного підпису цілі числа, з якими доводиться проводити обчислення, мають запис як мінімум на 20 % коротше, що відповідно зменшує складність обчислень не менше, ніж на 70 % і дозволяє помітно скоротити об'єм пам'яті, яка використовується;

- при виборі параметрів досить перевірити всього три умови;

- процедура підписування за допомогою цього методу не дозволяє обчислювати (як це можливо в RSA) електронні підписи під новими повідомленнями без знання секретного ключа.

Порівняно з оригінальним алгоритмом Ель Гамала метод DSA має одну важливу перевагу – при заданому в стандарті рівні стійкості числа, що беруть участь в обчисленні підпису, мають довжину 20 байтів кожне, скорочуючи загальну довжину підпису до 40 байтів.

Оскільки більшість операцій при обчисленні підпису і її перевірці також виробляється за модулем з 20 байтів, скорочується час обчислення підпису та об'єм пам'яті, яка використовується.

В алгоритмі Ель Гамала довжина підпису при такому рівні стійкості дорівнювала б 128 байтів.

Алгоритм НОТАРІУС-1 – аналог алгоритму Ель Гамала. Основна відмінність алгоритму полягає в тому, що замість звичайної операції множення цілих чисел за модулем великого простого p використовується множення за модулем простого числа, ця операція набагато ефективніше обчислюється на розповсюджених процесорах.

Процедури підписування електронних документів і перевірки електронних підписів за алгоритмом НОТАРІУС-1 виглядають аналогічно відповідним процедурам алгоритму Ель Гамала та забезпечують той же рівень стійкості підпису, але виконуються швидше. Алгоритм НОТАРІУС-S дозволив при збереженні стійкості підпису скоротити її довжину ще на 32,5 %. Для базового варіанта з ключами з 64 байтів довжина підпису скоротилася відносно DSA і НОТАРІУСА-D з 40 байтів до 27 байтів. Відповідно зменшився час обчислення і перевірки підпису. Стійкість залишилася на тому ж рівні – 10^{21} .

Алгоритм УЕП ДСТУ-4145. Алгоритм обчислення і перевірки УЕП заснований на властивостях груп точок еліптичної кривої над полями $GF(2^m)$. Стійкість алгоритму заснована на NP -повній задачі знаходження дискретного логарифма еліптичної кривої (знаходження значення k по базовій точці P і розташованій на кривій точці k). При цьому алгоритми на основі еліптичної кривої використовують ключі малих розмірів, що знижує вимоги до обчислювальних потужностей порівняно з вимогами до алгоритмів на основі RSA. У табл. 2.27 наведені порівняльні характеристики алгоритмів RSA і ECDSA (непарний випадок) при створенні і перевірці електронних підписів. Обидва алгоритми тестовано на паралельних процесорах Motorola 56303 DSP (66 МГц). При цьому функція перевірки підпису RSA використовує $e = 65\,537$.

Таблиця 2.27

**Порівняльні характеристики алгоритмів RSA і ECDSA
(непарний випадок) при створенні і перевірці електронного підпису**

Алгоритм (довжина ключа, біти)	Час виконання, мс	
	Створення підпису	Перевірка підпису
RSA (1024)	25	< 2
ECDSA (160)	32	33
RSA (2048)	120	5
ECDSA (216)	68	70

Як видно з табл. 2.27, при збільшенні розмірів ключа створення підписів за допомогою ECDSA виробляється значно швидше, ніж у системах RSA. Це розходження ще більшою мірою проявляється для однопроцесорних систем. З іншого боку, перевірка підпису за допомогою ECDSA робиться набагато повільніше, ніж ця ж процедура в системах RSA і знову ж розходження підсилюється для систем з одним процесором. Обробка ECDSA може трішки прискоритися в “парному” випадку. Потужність процесора, витрачена на перевірку підпису ECDSA, може сповільнити виконання інших додатків у системі. У деяких випадках системи RSA (навіть більші ключі, що використовують), можливо, будуть більш прийнятні, ніж криптосистеми на основі еліптичної кривої. Проте криптосистеми на основі еліптичної кривої одержують все більше поширення скоріше як альтернатива, а не заміна систем RSA, оскільки системи ECDLP мають деякі переваги, особливо при використанні в пристроях з малопотужними процесорами та маленькою пам'яттю. У табл. 2.28 наведені результати порівняльного аналізу найпоширеніших алгоритмів УЕП.

Таблица 2.28

Порівняльний аналіз найпоширеніших алгоритмів УЕП

Алгоритм	“Нотариус-S”			NIST DSA		
Р Довжина ключа (байти)	Л Довжина підпису (байти)	Q Складність підробки підпису без ключа	Т Час підробки підпису без ключа	Л Довжина підпису (байти)	Q Складність підробки підпису без ключа	Т Час підробки підпису без ключа
16	15	2*1014	10 днів	-	-	-
18	16	5*1014	1місяць	-	-	-
21	17	8*1015	1 рік	-	-	-
24	18	7*1016	10 років	-	-	-
36	21	5*1019	2 роки	-	-	-
48	24	1022	400 років	-	-	-
64	27	1024	2 тижня	40	1024	2 тижня
80	30	7*1026	25 років	40	1024	2 тижня
104	33	5*1029	18000 років	40	1024	2 тижня
128	36	2*1031	800000 років	40	1024	2 тижня
160	39	9*1034	3.9 млн. років	-	-	-
192	42	3*1037	107 років	-	-	-
224	45	5*1039	Більше 100 млрд.років			
256	48	6*1041				
304	51	4*1044				
352	54	1.5*1047				
400	57	3*1049				
448	60	4*1051				
512	63	2*1054				

Проведений аналіз даних табл. 2.28 дозволяє зробити висновок, що основним методом захисту УЕП є збільшення параметрів модулів перетворення порядку 512 і більше бітів. Але при цьому до такої ж довжини збільшуються і довжини ключів. Як наслідок збільшується обчислювальна складність криптографічних перетворень і зменшується швидкість. У той же час всі перетворення в банківських системах необхідно здійснювати в реальному масштабі часу, забезпечуючи необхідні показники конфіденційності та цілісності даних.

Система УЕП включає дві процедури: 1) процедуру створення підпису; 2) процедуру верифікації підпису (рис. 2.63).

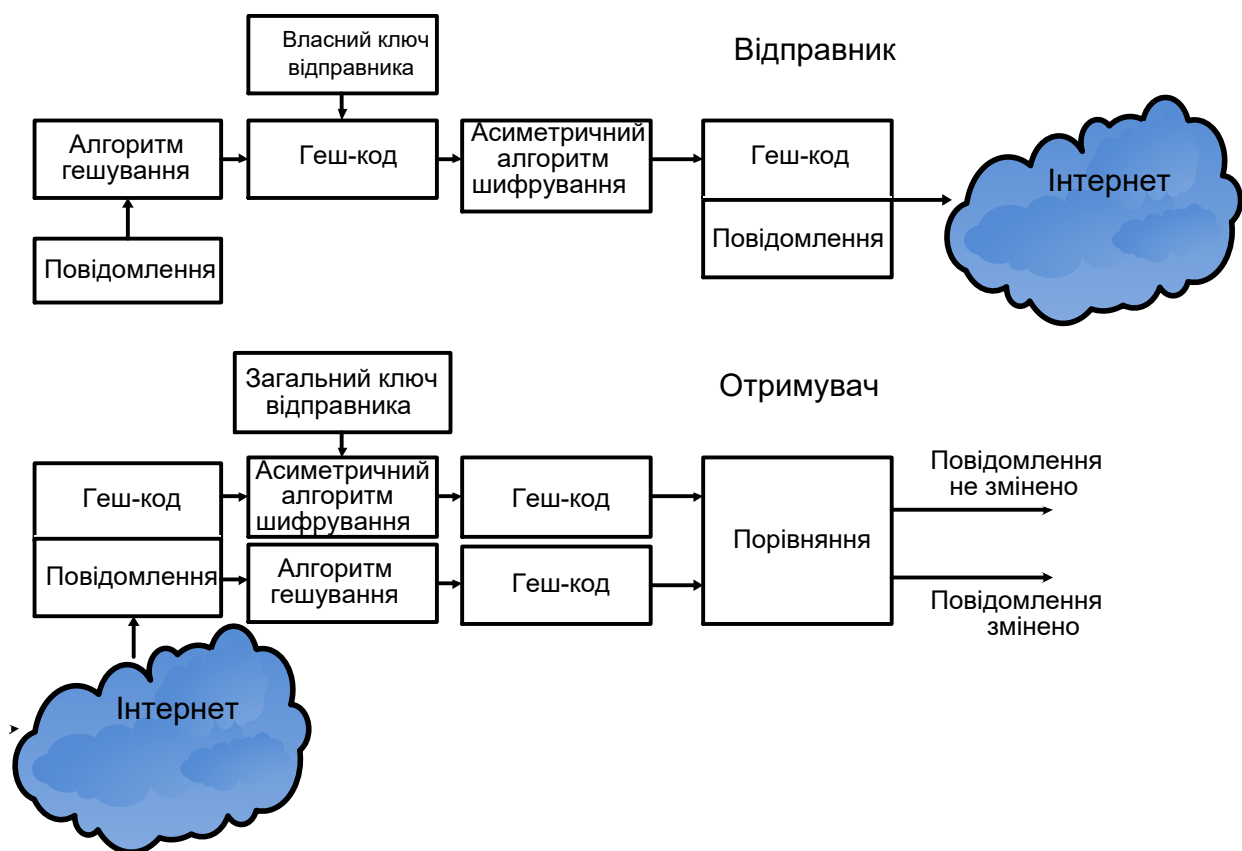


Рис. 2.63. Реалізація процесів створення та верифікації УЕП

У процедурі створення підпису використовується секретний ключ відправника повідомлення, у процедурі верифікації підпису – відкритий ключ відправника. При формуванні УЕП відправник обчислює ґеш-функцію $H(M)$ підписаного тексту M , призначену для стиснення і переміщення документа M , що підписується, до декількох десятків або сотень бітів (фіксованої довжини). При верифікації УЕП одержувач повідомлення знову обчислює ґеш-функцію $m' = H(M')$, прийнятого за допомогою вихідного тексту M (можливі зміни), після чого за допомогою відкритого ключа E

відправника перевіряє, чи відповідає отриманий підпис обчисленого значенням ґеш-функції $m' = m''$ [40].

Досить ефективним механізмом для забезпечення автентичності повідомлень є односпрямовані ґеш-функції. Частина з них будується на основі симетричного блокового алгоритму шифрування в режимі CBC або CPB, за допомогою фіксованого ключа і деякого вектора ініціалізації IV. Останній блок шифротексту і є ґеш-значенням повідомлення M. При такому підході не завжди можливо побудувати безпечну односпрямовану ґеш-функцію, але завжди можна одержати код автентифікації повідомлення MAC (Message Authentication Code). Основною перевагою цього механізму порівняно з УЕП є більш простий алгоритм генерації і верифікації, що дозволяє забезпечувати високу швидкодію алгоритмів автентифікації повідомлень у бездротових мережах передачі даних.

Для таких механізмів довжина блоку визначається довжиною ключа, а довжина ґеш-значення збігається з довжиною блоку. Чотири з найпоширеніших схем ґешування, що є безпечними при всіх атаках, наведені на рис. 2.64.

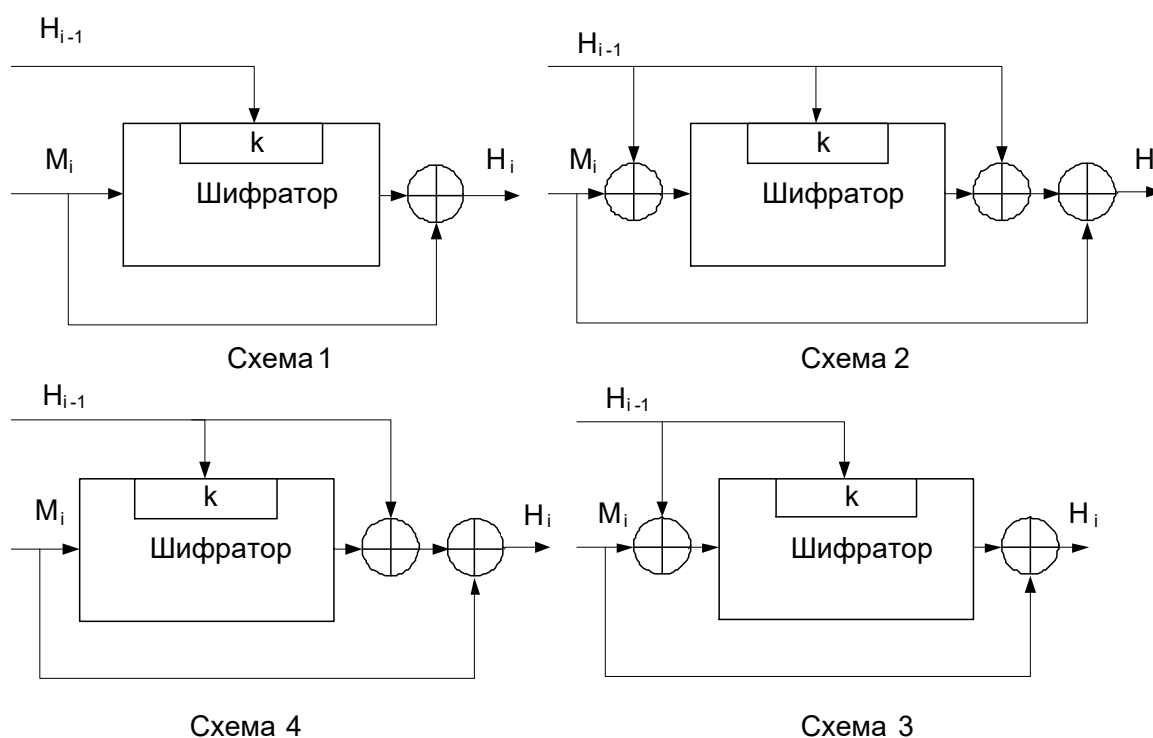


Рис. 2.64. Найпоширеніші варіанти схем ґешування

З поданих схем (рис. 2.64) при фіксованому шифрі більш стійкою є схема 2, однак вона є і більш складною в реалізації. Останнє обумовлене

результатом ґешування на попередньому кроці h_{i-1} , що подається як вхід циклової функції, як ключ на i -ому кроці k_{i-1} і складається за результатом h_i . Теоретичний доказ імовірності колізій у такому випадку є тривіальним [31; 57; 59; 61]. Ґешування повідомлення m виробляється в послідовності:

$$h \leftarrow IV, \quad h_i \leftarrow H(m_i, h_{i-1}) \quad \text{для } i=1, 2, \dots, l,$$

де H_i – функція стиску;

h_i – змінна зчеплення.

За необхідності останній блок заповнюється до довжини, яка кратна n . На відміну від стандартних передумов у ГОСТ 34.311 – 95 процедура ґешування очікує кінця повідомлення, а після робиться “набивання”, наведене на рис. 2.65.

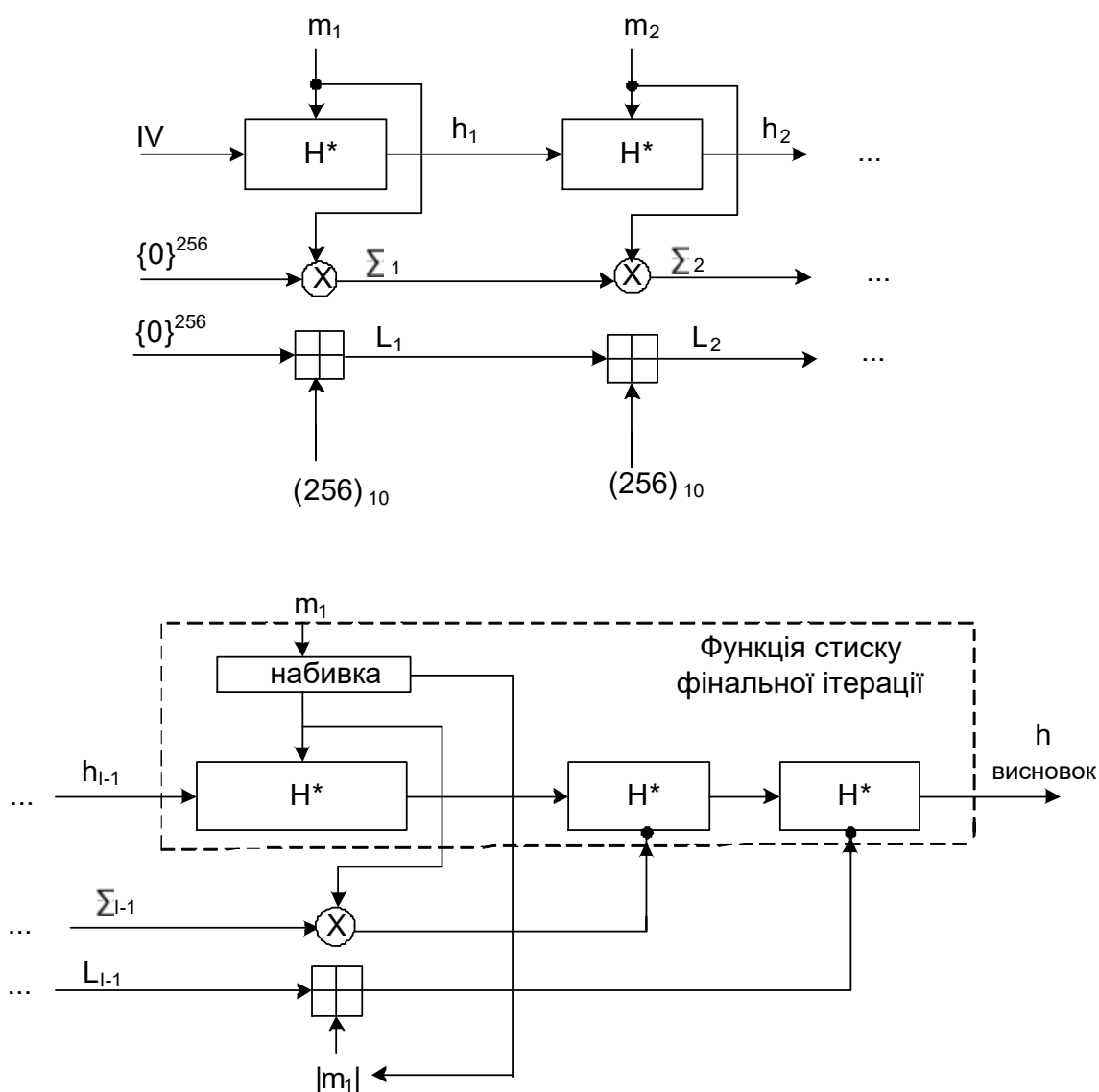


Рис. 2.65. Узагальнена структурна схема ґешування ГОСТ 34.311-95

Аналіз ґеш-функції за ГОСТ 34.311 – 95 дозволяє зробити наступні висновки:

булієва функція S-box лінійна і її застосування не виправдане – це призводить до зниження швидкості обробки даних через велику кількість повторень перетворення, що переміщає, для досягнення “заданого” рівня безпеки;

перетворення, що шифрує, при певних допущеннях неможливо атакувати в роздріб, а, отже, функцію стиску можна вважати стійкою до зіткнень;

алгоритм ґешування є методом послідовного ґешування з *MD*-посиленням (коефіцієнт стиску 2);

стійкість ґеш-функції значною мірою залежить від вибору блоків заміни у перетворенні, що шифрує, до того ж вони є одним з параметрів ініціалізації алгоритму ґешування;

IV у стандарті не фіксований, а це має бути на увазі, що необхідно виробити правила його використання, до того ж є великий клас атак на псевдозіткнення при нефіксованому IV;

швидкість обробки даних ґеш-функцією значно менша, ніж в аналогічних за зовнішніми параметрами HAVAL, SHA-256, а тим більше іншого *MD*-сімейства через спроби ліквідувати очевидні помилки конструювання ускладненням функції стиску;

приблизна швидкість реалізації – 4/5 від швидкості реалізації алгоритму шифрування, який лежить в основі;

з урахуванням парадоксу днів народження обчислювальна складність знаходження колізії становить $2^{256}/2$ операцій ґешування.

Це дозволяє заявити про більш високу криптографічну стійкість алгоритму порівняно з HAVAL, SHA-256 і потенційно високою стійкістю до колізій. Однак на сьогоднішній день знайдена колізія для функцій MD-5, SHA-1, які подібні до схеми ГОСТ 34.311 – 95. Somitra Kumar Sanadhyia і Palash Sarkary одержали новий метод визначення колізії на 22 кроці ґеш-функції SHA-2 з імовірністю 2^{-5} і 2^{-9} .

Незважаючи на те, що SHA-256 має 64 раунди, а SHA-512 – 80 раундів, задача визначення колізії не є тривіальною.

2.11.2. MDC-коди

Особливе місце серед механізмів забезпечення цілісності і автентичності займають функції ґешування: безключові та ключові, що дозволяють забезпечити широкий спектр послуг безпеки інформації згідно з ISO 7498. Односторонні ґеш-функції визначені в окремому міжнародному стандарті ISO/IEC 10118.

До безключових ґеш-функцій відносяться коди виявлення змін повідомлення (*MDC-код*, modification detection code), також відомі як коди виявлення маніпуляцій над повідомленнями або коди цілісності повідомлень. *MDC-коди* призначені для формування стислого образу або ґеш-коду повідомлення, який задовольняє спеціальним властивостям. Вони поділяються на блокові шифри та модульну арифметику. Зрештою *MDC-коди* забезпечують, спільно з іншими механізмами, цілісність даних. Безключові ґеш-функції є одним із складових елементів електронних підписів (рис. 2.66).

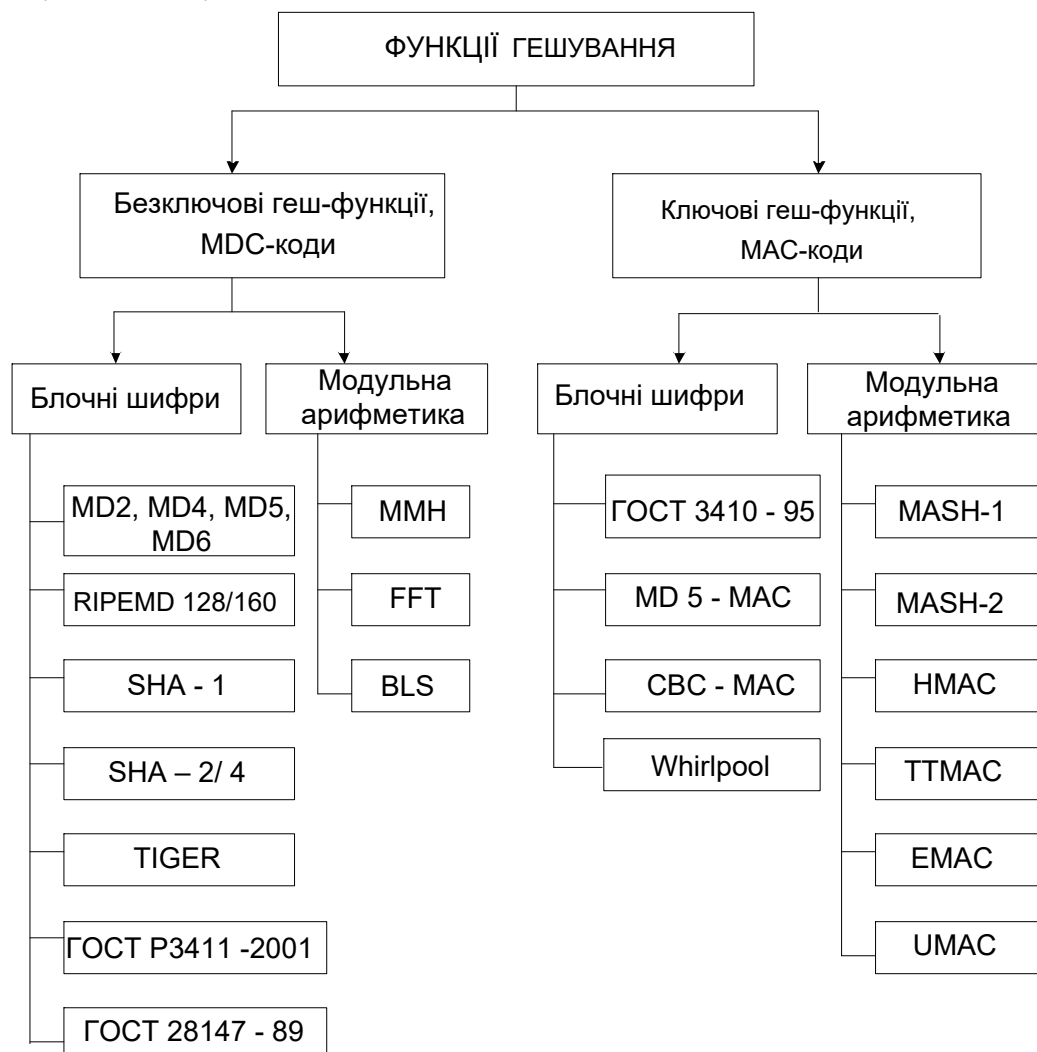


Рис. 2.66. Класифікація функцій ґешування

До ключових ґеш-функцій відносяться MAC-коди. Вони призначені для забезпечення цілісності даних і автентифікації повідомлень без використання будь-яких інших механізмів і дозволяють забезпечити автентифікацію повідомлення на основі використання методів симетричної криптографії. Алгоритми формування MAC-кодів розглядаються як ґеш-функції з двома вхідними параметрами, а саме повідомленням і секретним ключем. На виході такого алгоритму формується двійковий рядок фіксованої довжини. При цьому на практиці неможливо сформуванати точно такий же рядок без знання ключа.

У третій частині стандарту ISO/IEC 10118-3 визначені спеціалізовані ґеш-функції (dedicated hash function). Стандарт описує три типи таких ґеш-функцій: американський стандарт SHA, європейський стандарт RIPEMD-128, RIPEMD-160. Спеціальні ґеш-функції (Customized hash functions або dedicated hash functions) спеціально розроблені тільки для цілей ґешування і оптимізовані для виконання даного завдання. Ґеш-функції засновані на використанні принципів, закладених у ґеш-функціях сімейства MDx (MD2, MD4, MD5), які спеціально розроблялися для реалізації на 32-розрядних ЕОМ. На цей час ґеш-функції MD4, MD5 є найпоширенішими в практичних додатках ґеш-функціями. Однак деякі їх недоліки не дозволили стандартизувати їх [96]. Європейський консорціум RIPE, опираючись на свої дослідження властивостей цих алгоритмів, запропонував посилену версію MD4, що одержала назву RIPEMD. Ґеш-функція RIPEMD по суті складається із двох паралельно працюючих і модифікованих функцій MD4 (тобто функція має дві лінії). Суть модифікації полягає в зміні аргументів операторів циклічних зрушень і порядку проходження на вхід циклів ґеш-функції слів повідомлення, яке ґешується. Розглянемо ці алгоритми детальніше.

Алгоритм MDC-2. Даний метод виконує дві операції шифрування на обробку кожного вхідного блоку і, таким чином, швидкість ґеш-функції дорівнюватиме 1/2. Стандарт орієнтований на використання алгоритму DES як основи. Однак загальна конструкція може бути використана і для побудови ґеш-функції на основі інших блокових шифрів.

Ґеш-функція MDC-2 використовує такі компоненти:

- алгоритм DES як блоковий шифр Е з довжиною блоку $n = 64$ біти і 56-бітним ключем К;
- функції g і $\tilde{g}$, що відображають 64-бітні рядки в 56-бітні ключі для DES. Функції визначаються в такий спосіб. Нехай $U = u_1 u_2 \dots u_{64}$ буде 64-бітним рядком. Тоді в рядку U вилучається кожен восьмий біт, починаючи

з u_8 , а другий і третій біти встановлюються в "10" для g і "01" для $\tilde{g}$, тобто:

$$g(U) = u_1 10 u_4 u_5 u_6 u_7 u_9 u_{10} \dots u_{63}$$

$$\tilde{g}(U) = u_1 01 u_4 u_5 u_6 u_7 u_9 u_{10} \dots u_{63}.$$

При використанні алгоритму DES як блокового шифру ґеш-функція MDC-2 має наступний вигляд (рис. 2.67):

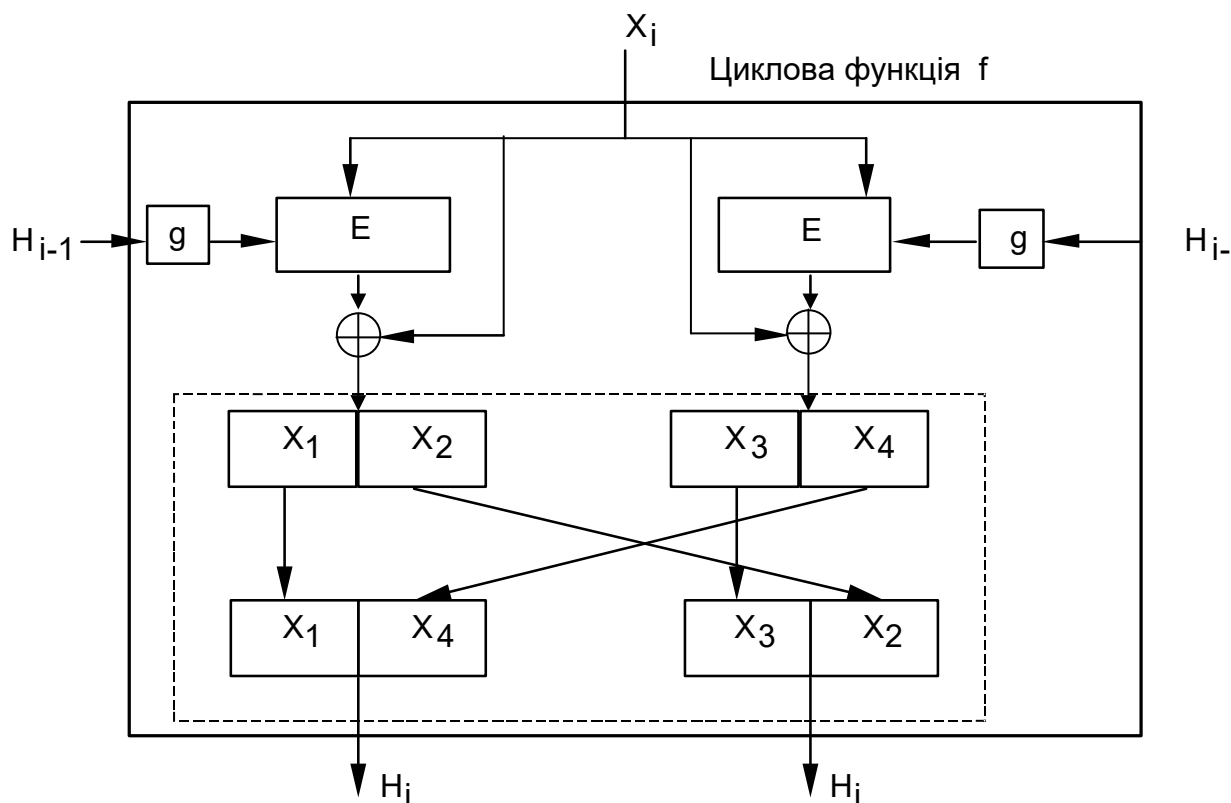


Рис. 2.67. Схема роботи ґеш-функції MDC-2

ґеш-функція MDC-2 (ISO/IEC 10118-2).

На вхід надходить рядок x довжиною $r = 64t$ бітів для $t \geq 2$.

На виході формується 128-бітний ґеш-код за рядком x .

1. Розбити рядок x на 64-бітні блоки x_i : $x = x_1 x_2 \dots x_t$.

2. Вибрати несекретні постійні IV і IV' . За замовчуванням

$$IV = 5252525252525252_{16}, IV' = 2525252525252525_{16}.$$

3. Нехай символ $\parallel$ позначає операцію конкатенації, символи C_i^L, C_i^R відповідно ліву і праву 32-розрядні частини результату шифрування C_i . Вихідне значення $h(x) = H_t \parallel \tilde{H}_t$ визначається в такий спосіб (для $1 \leq i \leq t$):

$$H_0 = IV; k_i = g(H_{i-1}); C_i = E_{k_i}(x_i) \oplus x_i; H_i = C_i^L \parallel \tilde{C}_i^R;$$

$$\tilde{H}_0 = IV'; k_i = \tilde{g}(\tilde{H}_{i-1}); \tilde{C}_i = E_{\tilde{k}_i}(x_i) \oplus x_i; \tilde{H}_i = \tilde{C}_i^L \parallel C_i^R.$$

На основі MDC-2 побудована більш складна ґеш-функція MDC-4. Ґеш-функція MDC-4 як циклову функцію використовує ґеш-функцію MDC-2. Одна ітерація циклової функції MDC-4 містить у собі послідовне виконання двох циклових функцій MDC-2. Таким чином, швидкість MDC-4 дорівнюватиме 1/4. Схема ґеш-функції MDC-4 зображена на рис. 2.68. Нижче наведений алгоритм MDC-4.

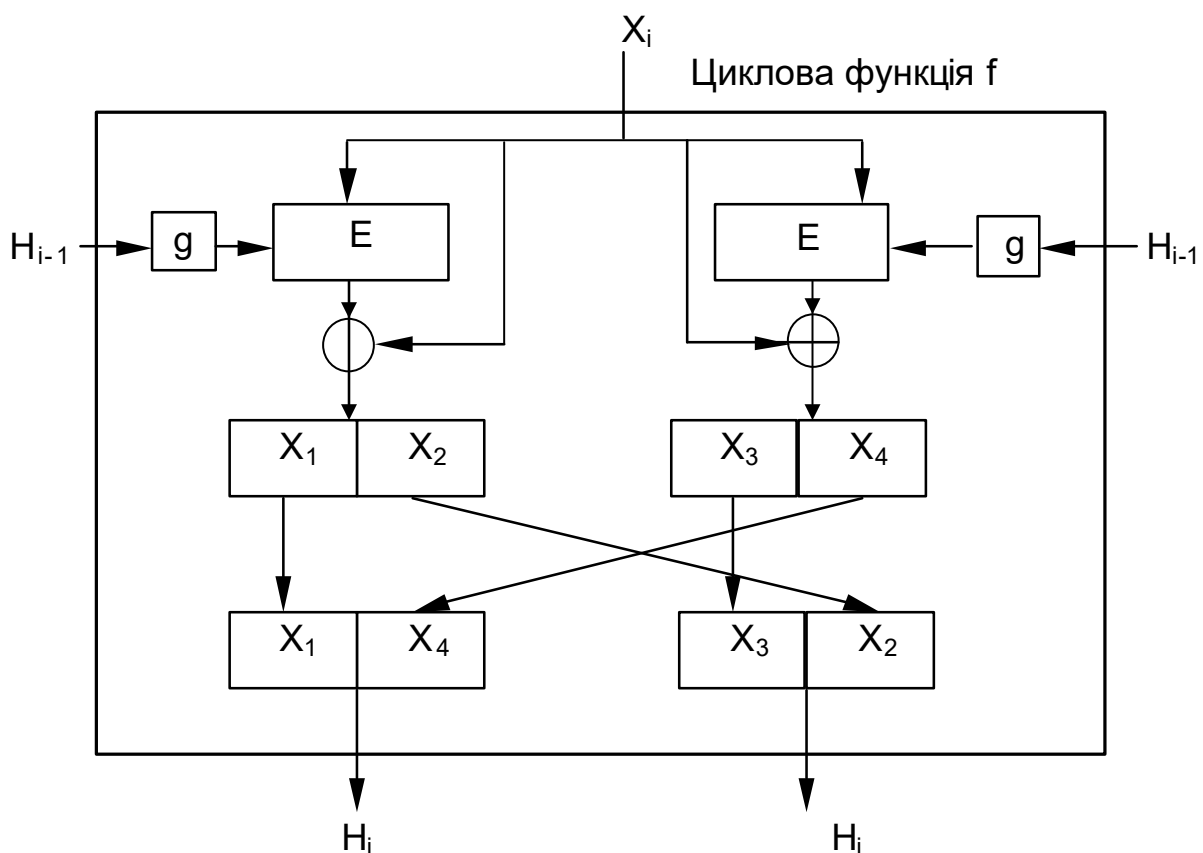


Рис. 2.68. Структура ґеш-функції MDC-4

Ґеш-функція MDC-4.

Вхід: рядок x довжиною $r = 64t$ біти для $t \geq 2$.

Вихід: 128-бітний ґеш-код за x .

1. Розбити рядок x на 64-бітні блоки x_i : $x = x_1x_2 \dots x_t$.

2. Вибрати несекретні постійні IV і IV' .

За замовчуванням

$$IV = 5252525252525252x, IV' = 2525252525252525x.$$

3. Нехай символ $\parallel$ позначає операцію конкатенації, а символи C_i^L, C_i^R відповідно ліву і праву 32-розрядні частини результату шифрування C_i . Вихідне значення $h(x) = G_t \parallel \tilde{G}_t$ визначається в такий спосіб (для $1 \leq i \leq t$):

$$\begin{aligned} G_0 &= IV; \tilde{G}_0 = IV'; \\ k_i &= g(G_{i-1}); G_i = E_{k_i}(x_i) \oplus x_i; H_i = C_i^L \parallel \tilde{C}_i^R; \\ \tilde{k}_i &= \tilde{g}(\tilde{G}_{i-1}); \tilde{G}_i = E_{\tilde{k}_i}(x_i) \oplus x_i; \tilde{H}_i = \tilde{C}_i^L \parallel C_i^R; \\ j_i &= g(H_i); D_i = E_{j_i}(\tilde{G}_{i-1}) \oplus \tilde{G}_{i-1}; G_i = D_i^L \parallel \tilde{D}_i^R; \\ \tilde{j}_i &= \tilde{g}(H_i); \tilde{D}_i = E_{\tilde{j}_i}(G_{i-1}) \oplus G_{i-1}; \tilde{G}_i = \tilde{D}_i^L \parallel D_i^R. \end{aligned}$$

Спеціалізовані ґеш-функції

Спеціалізовані ґеш-функції (Customised hash functions або dedi-cated hash functions) спеціально розроблені тільки для цілей ґешування й оптимізовані для виконання даної задачі. Третя частина стандарту ISO/IEC 10118-3 визначає три спеціалізовані ґеш-функції, а саме функції RIPEMD-128, RIPEMD-160 і SHA-1. Дані ґеш-функції засновані на використанні принципів побудови, закладені у ґеш-функції сімейства MDx (MD2, MD4, MD5), які спеціально розроблялися для реалізації на 32-розрядних ЕОМ. Алгоритм MD4 був запропонований Р. Райвестом у 1990 році, а в 1991 той же автор запропонував модифіковану версію алгоритму MD5. В даний час ґеш-функції MD4, MD5 є найбільш розповсюдженими в практичних додатках ґеш-функціями. Однак деякі їх недоліки не дозволили стандартизувати їх на міжнародному рівні.

Європейський консорціум RIPE, спираючись на свої дослідження властивостей цих алгоритмів, запропонував посилену версію MD4, що одержала назву RIPEMD. Ґеш-функція RIPEMD по суті складається з двох паралельно працюючих і модифікованих функцій MD4 (тобто функція має дві лінії). Суть модифікації полягає в зміні аргументів операторів циклічних зрушень і порядку проходження на вхід циклів ґеш-функції слів повідомлення, яке ґешується. Паралельні лінії, крім того, відрізняються використанням різних констант. Версії RIPEMD-128 і RIPEMD-160 виробляють ґеш-код довжиною, відповідно, 128 і 160 бітів.

Іншою альтернативою алгоритмам MDx є алгоритм SHA-1, розроблений спільно Агентством національної безпеки США і NIST й прийнятий як американський національний стандарт (FIPS 180-1).

Геш-функція SHA-1.

Алгоритм SHA-1 розроблений у 1992 році і формує за вхідним двійковим рядком довільної довжини 160-бітний геш-код. Алгоритм носить циклічний характер і у своїх циклах використовує тижий набір нелінійних функцій:

$$\begin{aligned}f(u, v, w) &= (u \wedge v) \vee (\sim u \wedge w); \\g(u, v, w) &= (u \wedge v) \vee (u \wedge w) \vee (v \wedge w); \\h(u, v, w) &= u \oplus v \oplus w,\end{aligned}$$

де u, v, w – 32-бітні змінні (слова);

$u \wedge v$ – логічне “І” за розрядами;

$u \vee v$ – логічне “АБО” за розрядами;

$\sim u$ – логічне “доповнення” за розрядами.

$\oplus$ – додавання за модулем 2.

Далі, при описі роботи алгоритму символ “+” позначає додавання за модулем 2^{32} , “<<s” – циклічний зсув уліво на s розрядів.

Алгоритм SHA-1.

Вхід. Двійковий рядок x довжиною $b \geq 0$.

Вихід. 160-бітний геш-код за рядком x .

1. *Ініціалізація*. Ініціалізувати п'ять векторів ініціалізації:

$h_1 = 67452301_x$; $h_2 = \text{efcdab89}_x$; $h_3 = 98badcfex$;

$h_4 = 10325476_x$; $h_5 = \text{c3d2e1f0}_x$.

Задати чотири додаткові константи.

$y_1 = 5a827999_x$; $y_2 = 6ed9eba1_x$;

$y_3 = 8f1bbcdc_x$; $y_4 = \text{ca62c1d6}_x$.

2. *Передобробка*. Доповнити рядок x так, щоб його довжина була кратна числу 512. Для цього додати в кінець рядка одиницю, а потім стільки нульових бітів, скільки необхідно для одержання рядка довжиною на 64 біти коротше довжини, кратної 512. Нарешті, додати останні два 32-розрядні слова, що містять двійкове представлення числа b . Нехай m – кількість 512-бітних блоків в отриманому доповненому рядку. Таким чином, форматований вхід буде складатися з $16m$ 32-розрядних слів $x_0x_1\dots x_{16m-1}$. Ініціалізувати вектор змінних зчеплення:

$$(H_1, H_2, H_3, H_4, H_5) = (h_1, h_2, h_3, h_4, h_5).$$

3. *Обробка*. Для всіх i від 0 до $m-1$ кожний i -й блок із шістнадцяти 32-бітних слів перетворити у вісімдесят 32-бітних слів і записати в

тимчасовий масив за таким алгоритмом:

$$X_i = x_{16i+j}, \text{ для } 0 \leq j \leq 15;$$

$$X_i = ((X_{j-3} \oplus X_{j-8} \oplus X_{j-14} \oplus X_{j-16})) \lll 1.$$

Зазначимо, що у початковій версії, що називалася SHA, алгоритм не містив лівого зсуву на один біт. Ця зміна введена для посилення ґеш-функції. У результаті отриманий алгоритм SHA-1.

Ініціалізувати робочі змінні

$$(A, B, C, D, E) = (H_1, H_2, H_3, H_4, H_5).$$

Обчислити для всіх i від 0 до $m-1$:

Для $j = 0$ до 19

$$t = ((A \lll 5) + f(B, C, D) + E + X_j + y_1);$$

$$(A, B, C, D, E) = (t, A, B \lll 30, C, D).$$

Для $j = 20$ до 39

$$t = ((A \lll 5) + h(B, C, D) + E + X_j + y_2);$$

$$(A, B, C, D, E) = (t, A, B \lll 30, C, D).$$

Для $j = 40$ до 59

$$t = ((A \lll 5) + g(B, C, D) + E + X_j + y_3);$$

$$(A, B, C, D, E) = (t, A, B \lll 30, C, D).$$

Для $j = 60$ до 79

$$t = ((A \lll 5) + h(B, C, D) + E + X_j + y_4);$$

$$(A, B, C, D, E) = (t, A, B \lll 30, C, D).$$

Обновити вектор змінних зчеплення:

$$(H_1, H_2, H_3, H_4, H_5) = (H_1 + A, H_2 + B, H_3 + C, H_4 + D, H_5 + E).$$

4. Завершення. За ґеш-код прийняти величину: $H_1 \parallel H_2 \parallel H_3 \parallel H_4 \parallel H_5$.

Порівняно з 128-розрядними ґеш-функціями 160-розрядний ґеш-код, який виробляється SHA-1, забезпечує більшу стійкість до силових атак. Ґеш-функції SHA-1 і RIPEMD-160 за стійкістю приблизно рівні й обидві перевершують MD5. Розширення блоку вхідного повідомлення введено з метою забезпечення більшої відмінності між вхідними блоками. Надмірність, що вводиться, сприяє підвищенню стійкості ґеш-функції.

Нижче наводяться тестові вектори для ґеш-функції SHA-1.

Вхідний рядок	Геш-код
""	0xda39a3ee5e6b4b0d3255bfe95601890afd80709
"a"	0x86f7e437faa5a7fce15d1ddcb9eaeaea377667b8
"abc"	0xa9993e364706816aba3e25717850c26c9cd0d89d
"abcdefghijklmnopqrstuvwxyz"	0x32d10c7b8cf96570ca04ce37f2a19d84240d3a89

Геш-функції RIPEMD-160, RIPEMD-128.

У 1995 році були виявлені деякі недоліки в першій версії геш-функції RIPEMD, що формує 128-розрядний геш-код. Зокрема були виявлені колізії для останніх двох із трьох циклів RIPEMD. Розроблена методика пошуку колізій була успішно застосована і для пошуку колізій у MD4. На основі отриманих результатів були зроблені припущення, що дана методика може бути використана для пошуку колізій у MD5 і в повній версії RIPEMD. Версія RIPEMD (разом з SHA-1) широко використовується в банківських технологіях. Однак ситуація з криптоаналізом геш-функцій сімейства MDx і перспективи розвитку обчислювальних потужностей призвели до висновку про необхідність відновлення RIPEMD. У результаті для стандартизації були запропоновані більш могутні версії геш-функції – RIPEMD-160 і RIPEMD-128. За оцінками розроблювачів геш-функція RIPEMD-160 повинна залишатися безпечною протягом десяти найближчих років. Удосконалена версія RIPEMD-128 замінила геш-функцію RIPEMD.

Розглянемо дані функції більш докладно.

Розмір геш-коду і перемінних зчеплень RIPEMD-160 дорівнює 160 бітам (п'ять 32-розрядних слів). Кількість циклів збільшена з трьох (як у RIPEMD) до п'яти. Крім того, внесені додаткові зміни в паралельні лінії алгоритму. Тепер права і ліва лінії відрізняються не тільки застосовуваними константами, але і порядком проходження булевих функцій. Для алгоритму RIPEMD-160 визначені такі бульові функції:

$$\begin{aligned}
 (u, v, w) &= u \oplus v \oplus w; \\
 g(u, v, w) &= (u \wedge v) \vee (\sim u \wedge w); \\
 h(u, v, w) &= (u \vee \sim v) \oplus w; \\
 k(u, v, w) &= (u \wedge w) \vee (v \wedge \sim w) \\
 l(u, v, w) &= u \oplus (v \vee \sim w).
 \end{aligned}$$

Усі позначення аналогічні раніше використаним позначенням при описі алгоритму SHA-1. На рис. 2.69 подана загальна структура геш-

функції RIPEMD-160.

Алгоритм RIPEMD-160.

Вхід. Двійковий рядок x довжиною $b \geq 0$.

Вихід. 160-бітний ґеш-код за рядком x .

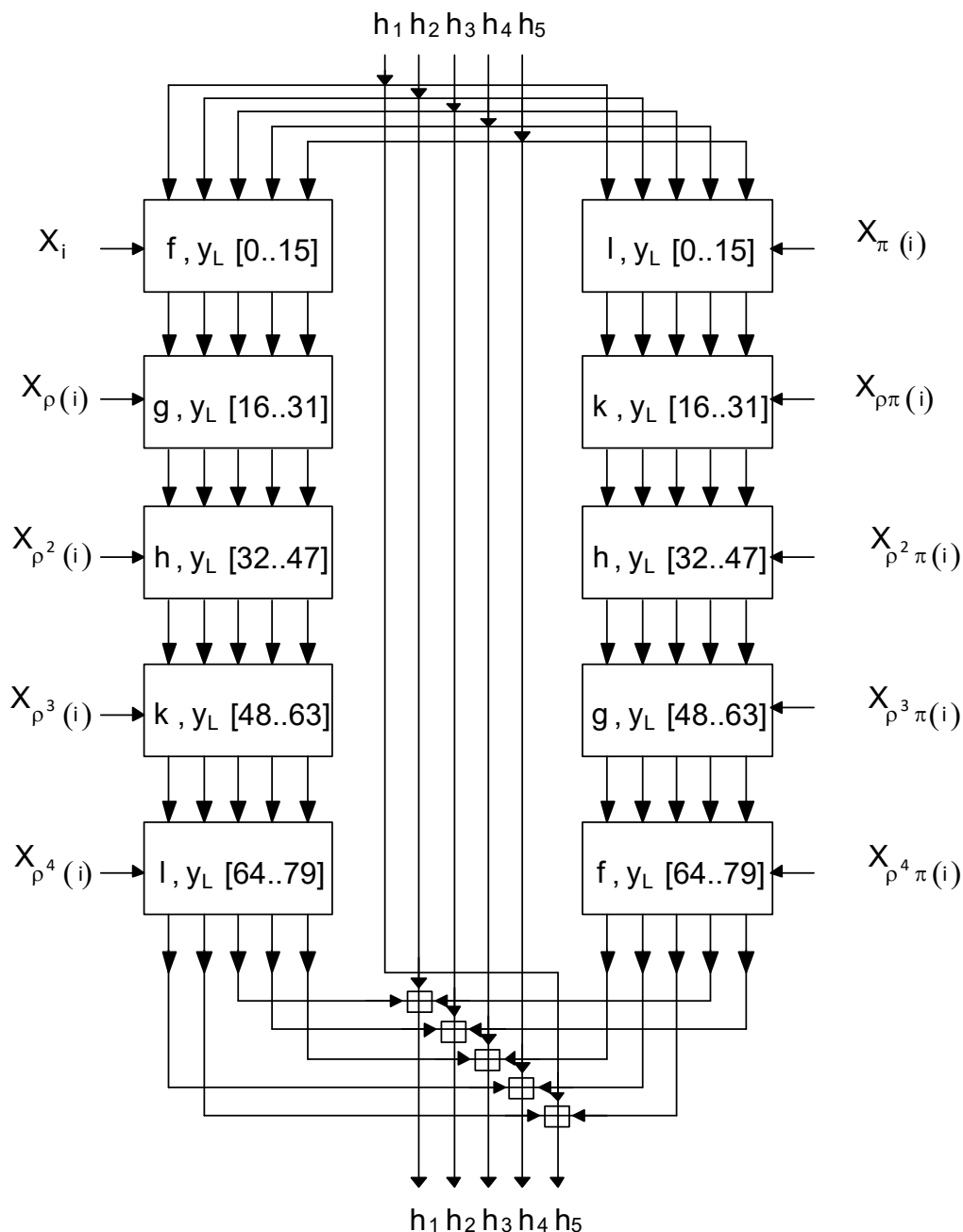


Рис. 2.69. Схема ґеш-функції RIPEMD-160

1. Ініціалізація

а) Ініціалізувати п'ять векторів ініціалізації:

$h_1 = 67452301_x$; $h_2 = \text{efcdab89}_x$; $h_3 = 98badcfex$;

$h_4 = 10325476_x$; $h_5 = \text{c3d2e1f0}_x$;

б) задати для лівої і правої ліній алгоритму додаткові константи:

– ліва лінія:

$$y_L[j] = 00000000_x, 0 \leq j \leq 15;$$

$$y_L[j] = 5a827999_x = \lfloor 20^{30} \cdot \sqrt{2} \rfloor, 16 \leq j \leq 31;$$

$$y_L[j] = 6ed9eba1_x = \lfloor 20^{30} \cdot \sqrt{3} \rfloor, 32 \leq j \leq 47;$$

$$y_L[j] = 8f1bbcdc_x = \lfloor 20^{30} \cdot \sqrt{5} \rfloor, 48 \leq j \leq 63;$$

$$y_L[j] = a953fd4e_x = \lfloor 20^{30} \cdot \sqrt{7} \rfloor, 64 \leq j \leq 79;$$

– права лінія:

$$y_R[j] = 0x50a28be6 = \lfloor 20^{30} \cdot \sqrt[3]{2} \rfloor, 0 \leq j \leq 15;$$

$$y_R[j] = 0x5c4dd124 = \lfloor 20^{30} \cdot \sqrt[3]{3} \rfloor, 16 \leq j \leq 31;$$

$$y_R[j] = 0x6d703ef3 = \lfloor 20^{30} \cdot \sqrt[3]{5} \rfloor, 32 \leq j \leq 47;$$

$$y_R[j] = 7ad76e9_x = \lfloor 20^{30} \cdot \sqrt[3]{7} \rfloor, 48 \leq j \leq 63;$$

$$y_R[j] = 00000000_x, 64 \leq j \leq 79;$$

в) задати порядок подачі слів повідомлення на ліву $z_L[j]$ і праву $z_R[j]$ лінії. Порядок подачі слів визначається на основі таких перестановок:

– для лівої лінії визначається перестановка ρ , а для циклів порядок слів визначається в такий спосіб:

i	$z_L[0..15]$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\rho(i)$	$z_L[16..31]$	7	4	13	1	10	6	15	3	12	0	9	5	2	14	11	8
$\rho^2(i)$	$z_L[32..47]$	3	10	14	4	9	15	8	1	2	7	0	6	13	11	5	12
$\rho^3(i)$	$z_L[48..63]$	1	9	11	10	0	8	12	4	13	3	7	15	14	5	6	2
$\rho^4(i)$	$z_L[64..79]$	4	0	5	9	7	12	2	10	14	1	3	8	11	6	15	13

– для правої лінії визначається перестановка $\pi(i) = 9i + 5 \pmod{16}$ і порядок подачі слів визначається з використанням перестановки ρ у такий спосіб:

$\pi(i)$	$z_R[0..15]$	5	14	7	0	9	2	11	4	13	6	15	8	1	10	3	12
$\rho\pi(i)$	$z_R[16..31]$	6	11	3	7	0	13	5	10	14	15	8	12	4	9	1	2
$\rho^2\pi(i)$	$z_R[32..47]$	15	5	1	3	7	14	6	9	11	8	12	2	10	0	4	13
$\rho^3\pi(i)$	$z_R[48..63]$	8	6	4	1	3	11	15	0	5	12	2	13	9	7	10	14
$\rho^4\pi(i)$	$z_R[64..79]$	12	15	10	4	1	5	8	7	6	2	13	14	0	3	9	11

г) визначити для лівої $s[j]$ і правої $s[j]$ ліній аргументи для операторів циклічного зсуву відповідно до таблиці:

$s_L[0..15]$	11	14	15	12	5	8	7	9	11	13	14	15	6	7	9	8
$s_L[16..31]$	7	6	8	13	11	9	7	15	7	12	15	9	11	7	13	12
$s_L[32..47]$	11	13	6	7	14	9	13	15	14	8	13	6	5	12	7	5
$s_L[48..63]$	11	12	14	15	14	15	9	8	9	14	5	6	8	6	5	12
$s_L[64..79]$	9	15	5	11	6	8	13	12	5	12	13	14	11	8	5	6
$s_R[0..15]$	8	9	9	11	13	15	15	5	7	7	8	11	14	14	12	6
$s_R[16..31]$	9	13	15	7	12	8	9	11	7	7	12	7	6	15	13	11
$s_R[32..47]$	9	7	15	11	8	6	6	14	12	13	5	14	13	13	7	5
$s_R[48..63]$	15	5	8	11	14	14	6	14	6	9	12	9	12	5	15	8
$s_R[64..79]$	8	5	12	9	12	5	14	6	8	13	6	5	15	13	11	11

2. *Передобробка.* Доповнити рядок x так, щоб його довжина була кратна числу 512. Для цього додати в кінець рядка одиницю, а потім стільки нульових бітів, скільки необхідно для одержання рядка довжиною на 64 біти коротше від довжини, кратної 512. Нарешті, додати останні два 32-розрядні слова, що містять двійкове представлення числа b . Нехай m – кількість 512-бітних блоків в отриманому доповненому рядку. У такий спосіб форматований вхід складається з $16m$ 32-розрядних слів $X_0 X_1 \dots X_{16m-1}$.

Ініціалізувати вектор змінних зчеплення

$$(H_1, H_2, H_3, H_4, H_5) = (h_1, h_2, h_3, h_4, h_5).$$

3. *Обробка.* Для всіх i від 0 до $m - 1$ кожен i -й блок з шістнадцяти 32-бітних слів записати в тимчасовий масив $X[j] = x_{16i+j}$, для $0 \leq j \leq 15$.

Потім:

а) виконати п'ять 16-крокових циклів лівої лінії.

Ініціалізувати робочі змінні:

$$(A_L, B_L, C_L, D_L, E_L) = (H_1, H_2, H_3, H_4, H_5).$$

Виконати:

для $j = 0$ до 15:

$$t = (A_L + f(B_L, C_L, D_L) + X[z_L[j]] + y_L[j]);$$

$$(A_L, B_L, C_L, D_L, E_L) = (E_L, E_L + (t \lll s_L[j]), B_L, C_L \lll 10, D_L);$$

для $j = 16$ до 31:

$$t = (A_L + g(B_L, C_L, D_L) + X[z_L[j]] + y_L[j]);$$

$$(A_L, B_L, C_L, D_L, E_L) = (E_L, E_L + (t \lll s_L[j]), B_L, C_L \lll 10, D_L);$$

для $j = 32$ до 47 :

$$t = (A_L + h(B_L, C_L, D_L) + X[z_L[j]] + y_L[j]);$$

$$(A_L, B_L, C_L, D_L, E_L) = (E_L, E_L + (t \lll s_L[j]), B_L, C_L \lll 10, D_L);$$

для $j = 48$ до 63 :

$$t = (A_L + k(B_L, C_L, D_L) + X[z_L[j]] + y_L[j]);$$

$$(A_L, B_L, C_L, D_L, E_L) = (E_L, E_L + (t \lll s_L[j]), B_L, C_L \lll 10, D_L);$$

для $j = 64$ до 79 :

$$t = (A_L + l(B_L, C_L, D_L) + X[z_L[j]] + y_L[j]);$$

$$(A_L, B_L, C_L, D_L, E_L) = (E_L, E_L + (t \lll s_L[j]), B_L, C_L \lll 10, D_L);$$

б) виконати паралельно з циклами лівої лінії п'ять 16-крокових циклів правої лінії.

Ініціалізувати робочі змінні

$$(A_R, B_R, C_R, D_R, E_R) = (H_1, H_2, H_3, H_4, H_5).$$

Виконати:

для $j = 0$ до 15 :

$$t = (A_R + l(B_R, C_R, D_R) + X[z_R[j]] + y_R[j]);$$

$$(A_R, B_R, C_R, D_R, E_R) = (E_R, E_R + (t \lll s_R[j]), B_R, C_R \lll 10, D_R);$$

для $j = 16$ до 31 :

$$t = (A_R + k(B_R, C_R, D_R) + X[z_R[j]] + y_R[j]);$$

$$(A_R, B_R, C_R, D_R, E_R) = (E_R, E_R + (t \lll s_R[j]), B_R, C_R \lll 10, D_R);$$

для $j = 32$ до 47 :

$$t = (A_R + h(B_R, C_R, D_R) + X[z_R[j]] + y_R[j]);$$

$$(A_R, B_R, C_R, D_R, E_R) = (E_R, E_R + (t \lll s_R[j]), B_R, C_R \lll 10, D_R);$$

для $j = 48$ до 63 :

$$t = (A_R + g(B_R, C_R, D_R) + X[z_R[j]] + y_R[j]);$$

$$(A_R, B_R, C_R, D_R, E_R) = (E_R, E_R + (t \lll s_R[j]), B_R, C_R \lll 10, D_R);$$

для $j = 64$ до 79 :

$$t = (A_R + l(B_R, C_R, D_R) + X[z_R[j]] + y_R[j]);$$

$$(A_R, B_R, C_R, D_R, E_R) = (E_R, E_R + (t \lll s_R[j]), B_R, C_R \lll 10, D_R);$$

в) оновити змінні зчеплення

$$t = H_1;$$

$$H_1 = H_2 + C_L + D_R;$$

$$H_2 = H_3 + D_L + E_R;$$

$$H_3 = H_4 + E_L + A_R;$$

$$H_4 = H_5 + A_L + B_R;$$

$$H_5 = t + B_L + C_R.$$

4. *Завершення.* Як ґеш-код прийняти величину: $H_1 || H_2 || H_3 || H_4 || H_5$.

Основні відмінності алгоритму RIPEMD-128 від RIPEMD-160 полягають в такому:

довжина ґеш-коду складає 128 бітів (чотири 32-бітних слова);

функція складається з чотирьох циклів i , відповідно, використовується чотири 32-розрядних змінних зчеплення;

основна операція циклу має вигляд:

$$t = (A + f(B, C, D) + X[z[j]] + y[j]);$$
$$(A, B, C, D) = (D, D + (t \lll s[j]), A, B, C);$$

визначені тільки чотири булеві функції f, g, h і k ;

з переліку констант виключаються значення $\lfloor 20^{30} \cdot \sqrt{7} \rfloor$ і $\lfloor 20^{30} \cdot \sqrt[3]{7} \rfloor$,

а з переліку векторів ініціалізації виключають h_5 .

Таким чином, принципи побудови ґеш-функцій сімейства RIPEMD можна сформулювати таким способом:

1. Побудова ґеш-функції здійснюється на основі застосування двох паралельних ліній ґешування, кожна з яких містить три (RIPEMD), чотири (RIPEMD-128) чи п'ять (RIPEMD-160) циклів.

2. Використання двох регістрів зсуву в основній операції.

3. Уведення розходжень між паралельними лініями алгоритму, а саме використання різного порядку подачі слів вихідного повідомлення і зворотне застосування нелінійних булевих функцій.

4. Використання перестановок спеціального вигляду. Перестановка ρ побудована таким чином, що якщо два слова вихідного повідомлення є "близькими" на першому-другому циклах, то вони будуть розрізнятися на другому-третьому циклах ґеш-функції і навпаки. Перестановка π побудована таким чином, що якщо два слова будуть близькі у своїй лівій частині, то в правій частині вони будуть відрізнятися принаймні в 7 позиціях.

5. Зсуви в ґеш-функціях обрані відповідно до таких правил:

значення зсуву повинно перебувати в межах від 5 до 15;

кожен блок повідомлення повинний зсуватись на різну кількість позицій на всіх циклах ґешування;

зсуви, які застосовуються на кожному регістрі, не повинні мати спеціальних шаблонів, наприклад, їх сума не повинна ділитися на 32;

не занадто багато значень зсуву повинні ділитися на чотири.

Деякі додатки, у яких використовують ґеш-функції, можуть вимагати виробітки більш довгих ґеш-кодів. Прямим шляхом розв'язання даної задачі є паралельне виконання однієї і тієї ж ґеш-функції з різними векторами ініціалізації. Однак у цьому випадку можлива поява залежностей між двома результатами ґешування, що і було виявлено при використанні MD4. Ґеш-функції RIPEMD-128 і RIPEMD-160 уже мають паралельні лінії і для одержання ґеш-кодів довжиною, відповідно, 256 і 320 бітів досить виключити кінцеву комбінацію результатів ґешування по кожній лінії.

У табл. 2.29 подана порівняльна характеристика безключових ґеш-функцій.

Таблиця 2.29

Порівняльна характеристика безключових ґеш-функцій

Характеристика	MD5	MD6	SHA-1	SHA-2 (256/512)	ГОСТ 28147	RIPEMD	
						128	160
Довжина дайджесту, біти	128	512	160	256/512	256	128	160
Розмір блоку обробки, біти	512	512	512	512/1024	512	512	512
Кількість ітерацій	64	168	80	64/80	32	128	160
Кількість елементарних логічних функцій	4	4	3	6/6	8	5	5
Кількість додаткових констант	64		4	64/64	—	4	4
Швидкість роботи на Pentium III 1000 MHz, Мбіт/с	574,64		344,43	135,5/68,7	315,27	63.8	39.8

Іншою альтернативою алгоритмам MDx є алгоритм SHA-1, розроблений спільно Агентством національної безпеки США та NITS і прийнятий у якості американського національного стандарту (FIPS 180-1). Порівняно з 128-розрядними ґеш-функціями 160-розрядний ґеш-код, який виробляється SHA-1, забезпечує більшу стійкість до силових атак. Ґеш-функції SHA-1 і RIPEMD-160 за стійкістю приблизно рівні та обидва перевершують MD5. Не менш важливим параметром є швидкість роботи алгоритму. SHA-1, RIPEMD-160 містить більшу кількість кроків і виконується на 160-бітовому буфері порівняно зі 128-бітовим буфером MD5. Таким чином, SHA-1 виконується на 25 % повільніше, ніж MD5 на тих

же апаратурах. RIPEMD-160 має 160 кроків, і виконується приблизно на 15 % повільніше, ніж SHA-1. Порівняльний аналіз ґеш-функцій подано в табл. 2.30.

Таблиця 2.30

Порівняльний аналіз ґеш-функцій

ґеш-функція	Клас функції	Базові перетворення	Довжина ґешу, біти
Whirlpool	однонаправлена	в кінцевих полях Галуа	512
SHA-2	однонаправлена	логічні та арифметичні	256, 384, 512
ГОСТ 34311-95	однонаправлена	БСШ	256
HAVAL	однонаправлена	логічні та арифметичні	128, 160, 192, 256
SHA-1	однонаправлена	логічні та арифметичні	160
RIPEMD-160	однонаправлена	логічні та арифметичні	160
MD5	однонаправлена	логічні та арифметичні	128
MD4	однонаправлена	логічні та арифметичні	128
UMAC	однонаправлена, MAC	в кінцевих полях Галуа	128, 64
Rijndael,CBC–MAC	виробка MAC	БСШ	128
ГОСТ28147-89(OFB)	виробка MAC	БСШ	64

Проведений аналіз табл. 2.30 показує, що використання *MDC*-коду за даним вхідним повідомленням ґеш-коду може обчислити будь-який суб'єкт, а при використанні *MAC*-коду обчислити ґеш-код за даним вхідним повідомленням може тільки суб'єкт, що володіє секретним ключем. Таким чином, використання *MAC*-кодів дозволяє інтегровано вирішувати завдання ґешування і забезпечення стійкості отриманих ґеш-кодів.

2.11.3. MAC-коди

Окремо можна виділити три підходи побудови *MAC*-кодів

1. *MAC*-коди, побудовані на основі БСШ (CBC-MAC).
2. *MAC*-коди, побудовані на основі безключових ґеш-функцій (HMAC, MDX-MAC).
3. *MAC*-коди, побудовані на основі сімейства універсальних ґеш-функцій.

Стійкість *MAC*-кодів, побудованих на основі БСШ у режимі CBC, базується на стійкості блокового шифру, який використовується в їх основі, що повинен мати необхідні властивості випадковості формованих

вихідних значень. Крім того, для даного алгоритму не повинні реалізовуватися атаки, які опираються на використання будь-яких особливих властивостей та особливостей структури алгоритму.

Алгоритм HMAC – алгоритм посилення криптостійкості інших алгоритмів, зокрема MD5. Код HMAC можна використати разом з будь-якою ітеративною криптографічною ґеш-функцією, наприклад, з MD5 або SHA-1, у сполученні із загальним секретним ключем. Стійкість коду HMAC залежить від властивостей лежачої в його основі ґеш-функції.

HMAC – алгоритм кодів автентифікації повідомлень, запропонований в 1996 р. М. Белає, У. Канетті і Х. Кравчук. Схема HMAC, використовуючи SHA-1 як основну функцію ґешування, був стандартизований NIST як FIPS-198, і входить до стандарту ISO (Міжнародна електротехнічна комісія) 9797-2. HMAC використовує функцію ґешування (або функцію стиснення) як чорний ящик таким чином, що існуючий код може використовуватися багато разів і основна ґеш-функція може бути легко замінена. NESSIE рекомендує для HMAC використовувати SHA-256 або SHA-512 як одних із найбільш стійких до колізії алгоритмів ґеш-функцій. Нехай дана ґеш-функція H з довжиною ґешу l -біт, і що ця функція заснована на повторенні функції стиснення F і використовує блоки повідомлення розміром b бітів.

Далі схема HMAC, що побудована на цій ґеш-функції, завантажує повідомлення M , що складається з довільного ряду бітів і розбивається на l -бітів блоків $HMAC(K, M)$, під дією ключа K працює до b -го біта (при цьому останній блок, якщо не рівний l -біт, доповнюється нулями). На рис. 2.70 зображено схему роботи алгоритму HMAC.

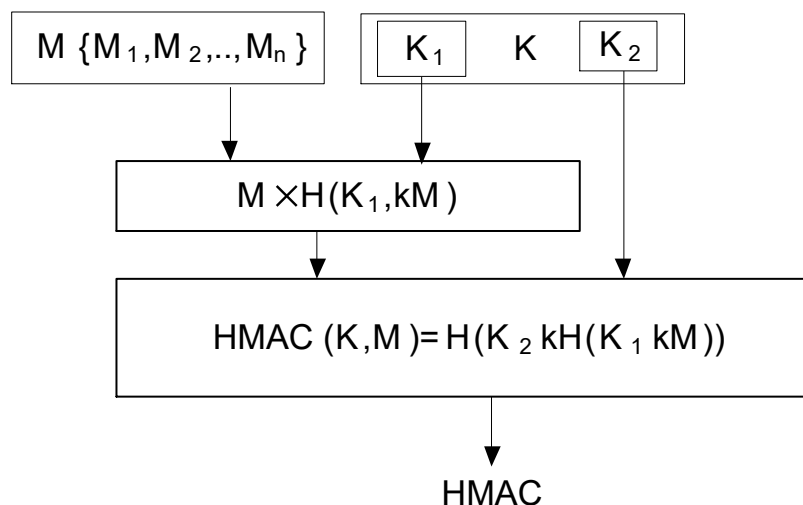


Рис. 2.70. Схема роботи алгоритму HMAC

Стійкість HMAC безпосередньо пов'язана із стійкістю функції ґешування H. Доведення стійкості для HMAC засноване на дослідженні функції ґешування H і функції компресії F (де H – повторюваний алгоритм, що застосовує F на блоках повідомлення фіксованої довжини).

Стійкість базується на таких властивостях: ґеш-функція H стійка для колізії, коли початкове значення секретне; функція компресії F, що змінюється за початковим значенням, становить стійкий алгоритм MAC (це означає, що його ґеш-послідовність складна для аналізу).

Рівень безпеки залежить від розміру внутрішнього блоку (розмір блоку функції ґешування), від довжини ключа і від довжини значення MAC-коду. Серед основних недоліків алгоритмів, розглянутих вище, є погане розсіювання та використання для генерації ключа алгоритму DES (AES). Щодо використання алгоритмів DES та AES для отримання стійкої ключової послідовності, то це накладає обмеження на швидкість ґешування самого алгоритму MAC-коду. У табл. 2.31 подано аналіз тестування швидкості роботи алгоритмів ґешування.

Таблиця 2.31

Аналіз тестування швидкості роботи алгоритмів ґешування

Функція ґешування	Кількість циклів	Мова реалізації	Швидкість роботи на Celeron 600 MHz	Швидкість роботи на Pentium III 1000 MHz
Whirlpool	10	C	28,013 Мбіт/с	46,961 Мбіт/с
SHA-2 (512)	80	C	41,159 Мбіт/с	68,701 Мбіт/с
SHA-2 (256)	64	C	81,308 Мбіт/с	135,557 Мбіт/с
ГОСТ 34311-95	-	C+Assembler	49,408 Мбіт/с	83,056 Мбіт/с
HAVAL	96(128, 160)	C	337,842 Мбіт/с	564,809 Мбіт/с
SHA-1	80	C, Assembler	206,285 Мбіт/с 361,581 Мбіт/с	344,433 Мбіт/с 605,558 Мбіт/с
RIPEMD-160	160	C	147,465 Мбіт/с	246,568 Мбіт/с
MD5	64	C	278,715 Мбіт/с	574,635 Мбіт/с
MD4	48	C	344,086 Мбіт/с	467,793 Мбіт/с
UMAC	-	C C+Assembler	989,371 Мбіт/с 3518,900 Мбіт/с	1648,953 Мбіт/с 5885,057 Мбіт/с
Rijndael CBC–MAC	14	C	139,376 Мбіт/с	231,255 Мбіт/с

У табл. 2.32 наведено можливі значення довжин ключа і ґеш-коду для різних алгоритмів сімейства MAC.

Таблиця 2.32

Довжина ключа k і довжина ґеш-коду n для MAC-кодів

Алгоритм	k (ключ)	n (ґеш-код)
UMAC	128	64
TTMAC	160	≤ 160
EMAC-AES	128, 192, 256	≤ 128
RMAC-AES	128, 192, 256	≤ 128
HMAC-SHA-1	≤ 512	≤ 160

У фінальному звіті міжнародного криптографічного конкурсу NESSIE відзначена функція ґешування Whirlpool (на основі власного убудованого блочно-симетричного шифру) і два алгоритми формування MAC-кодів: UMAC і Two-Track-MAC.

У CBC Whirlpool застосовується шифр W до поточного блоку повідомлення M_i як до відкритого тексту, так і з використанням H_i як ключа, розраховується побітове додавання за модулем 2 наступних трьох рядків: шифртексту, отриманого з W , рядка M_i і рядка H_i .

Алгоритм EMAC, заснований на AES CBC-MAC, вважається стандартом для входних у цю категорію шифрів. Схема використовує AES (Rijndael)-блоковий симетричний шифр і генерує величину MAC до 128 бітів. RMAC становить випадковий варіант схеми EMAC, що забезпечує кращу протидію атаці на основі внутрішніх помилок. Відмінність – у вихідному перетворенні (шифрування виробляється на ключі, отриманому порозрядним доповненням K_2 і R). Two-Track-MAC заснований на ґеш-функції RIPEMD-160 з модифікаціями. Алгоритм працює шляхом ітераційного використання функції стиску.

У табл. 2.33 подано основні результати оцінки швидкодії Two-Track-MAC-алгоритму для різних операційних платформ. Швидкість обчислень визначається кількістю циклів процесора, які затрачуються на один байт оброблюваного повідомлення. Як видно з аналізу табл. 2.33, Two-Track-MAC поступається за швидкістю тільки UMAC-алгоритму приблизно в 3 – 7 разів, але має в 2,5 раза більший розмір MAC-коду.

Швидкодія MAC-алгоритмів

Алгоритм	Довжина MAC-коду (біти)	Довжина ключа (біти)	Тип ПЕВМ				
			Pentium2	PIII/Linux	Pentium4	Xeon	AMD
TTMAC	160	160	21	21	40	37	21
UMAC-16	64	128	6.1	6.0	6.2	6.1	6.2
UMAC-32	64	128	2.5	2.9	6.7	6.6	1.9
HMAC-Whirlpool	512	512	86	72	98	103	100
HMAC-MD4	128	512	4.7	4.7	6.4	6.4	4.7
HMAC-MD5	128	512	7.2	7.3	9.4	9.4	7.4
HMAC-RIPE-MD	160	512	23	18	27	26	21
HMAC-SHA-0	160	512	16	15	23	23	13
HMAC-SHA-1	160	512	16	15	25	24	12
HMAC-SHA-2	256 384 512	512	40	39	40	39	33
			84	84	124	132	72
			84	84	124	132	72
HMAC-Tiger	192	512	24	21	28	26	20
CBC MAC-Rijndael (EMAC)	128	128	24	26	26	27	31
CBC MAC-DES	64	56	62	61	72	69	54
CBC MAC-Shacal	512	160	31	31	67	74	29

Аналіз табл. 2.33 показує, що схеми ключового ґешування UMAC на основі поліноміальних функцій дозволяють одержати найвищу швидкість ґешування, крім цього вони є переможцем Міжнародного криптографічного конкурсу NESSIE.

**Поліноміальні методи побудови універсальних
ґеш-функцій**

Розглянемо практичне застосування даних конструкцій, які використовуються у найшвидшому алгоритмі формування MAC-кодів – UMAC-16 і UMAC-32.

UMAC – код автентифікації повідомлення був розроблений Тедом Кроветцом (Ted Krovetz), Джоном Блеком (John Black), Шаї Халеві (Shai Halevi), Хьюго Кравцеком (Hugo Krawczyk) і Пилипом Рогевеєм (Phillip

Rogaway). Алгоритм заснований на сімействах універсальних ґеш-функцій і забезпечує доказову безпеку MAC-коду, безпека забезпечується стійкістю застосовуваного блочно-симетричного шифру (БСШ) AES у режимі CBC (зчеплення блоків відкритого тексту в другому шарі функції UHASH-16 або UHASH-32).

Таким чином, алгоритм дозволяє забезпечити більш високу швидкість (особливо для довгих повідомлень) і підтверджену безпеку ціною більшої складності.

Для опису процедур формування MAC-коду розглянемо UMAC (1999 року). Повідомлення M довільної довжини надходить на двошарову функцію UHASH, у якій на першому шарі за допомогою універсального класу ґеш-функцій NH повідомлення розбивається на блоки певної довжини (за винятком останнього блоку, що може мати меншу довжину). Кожен блок обробляється адитивним ключовим матеріалом (один ключ використовується для кожного блоку), процедура стиску здійснюється шляхом множення пари блоків повідомлення на ключ. Всі стислі блоки потім конкатенуються і довжина інформаційного блоку зростає до встановленого значення.

На другому шарі за допомогою функції PRF (псевдодовільної функції), використовуючи або режим CBC із БСШ AES, або використовуючи алгоритм HMAC, одержуємо MAC-код фіксованої довжини. Таким чином, сімейство універсальних ґеш-функцій NH використовується як прискорювач на HMAC або CBC-MAC.

Крім того, можна використати конструкцію Тойпліца (Toeplitz) для зниження ймовірності підробки (застосовуючи NH кілька разів із ключами, які перемішані між собою, і конкатенуючи результати), і/або використати дворівневе ґешування для зменшення кількості необхідних ключів. Існують також деякі інші варіації, що дозволяють оптимізувати деякі види структур.

Значна частина обмежень UMAC (1999 року) виходить із того, що повідомлення стискається до встановленого коефіцієнта, а не встановленої довжини. У першому випадку код автентифікації обчислюється шляхом обчислення функції PRF (ґеш||залишок), у другому випадку може бути обчислений шляхом ґеш PRF (залишок), що має деякі переваги (використання PRF обмежене мінімумом, у нього немає входу

необмеженої довжини). NH , сімейство універсальних ґеш-функцій, використаних в UMAC (1999 року), могло також бути стисле до встановленої вихідної довжини, у цьому випадку потрібний буде ключ (згенерований PRG) довжиною, рівною повідомленню (ціле повідомлення повинне розглядатися як єдиний блок у даному описі).

Нова версія UMAC (2000 року) є переможцем криптографічного європейського конкурсу NESSi, уводить додаткову складність для рішення проблеми стиску повідомлення до встановленої довжини так, що MAC може обчислюватися ґеш PRF (залишок). Частина PRF працює, кодуючи залишок блоковим шифром. Частина ґешу (також називана UHASH) стискає повідомлення, що складається із трьох різних шарів:

на першому шарі використовується швидке сімейство ґеш-функцій NH , для стиску повідомлення до встановленого коефіцієнта;

на другому шарі ґеш-коду установленої довжини використовується сімейство ґеш-функцій RP , яке не так швидко як NH , але генерує вихід установленої довжини з використанням ключа встановленої довжини (для цього використовується режим CBC БСШ AES);

на третьому шарі використовується сімейство ґеш-функцій IP , що зводить довжину свого входу до розміру, який більше підходить.

Універсальне сімейство ґеш-функцій RP ґрунтується на поліноміальних обчисленнях. Рядок, що складається з n шарів довжиною m бітів може розглядатися як поліном степеня n в множині значень, де кожне слово рядка служить коефіцієнтом. Для обчислення ґешу необхідно обчислити поліном для довільно обраної точки (ключа).

Для ефективності обчислення виконуються в області максимальних значень, однак менше, ніж 2^m . Функція застосовується для розширення області до довільного рядка, при цьому використовується гібридна схема, у якій RP становлять поліноміальні ґеш-коди: невеликий діапазон значень використовується для коротких повідомлень, а більший – для більш довгих повідомлень. З погляду безпеки, імовірність колізії підвищується порівняно із шаром універсального класу ґеш-функцій NH -алгоритму.

Шар з універсальним сімейством ґеш-функцій IP скорочує довжину свого входу, оскільки шар ґешу RP генерує виходи, значно більші порівняно з передбачуваною ймовірністю колізій (для найдовших

автентифікованих повідомлень більшість бітів вихідних рядків, шаруючи RP, будуть нулями).

Алгоритми UMAC16 і UMAC32 засновані на тришарових схемах ґешу UHASH 16 і UHASH 32 відповідно й використовуються для кодування залишку БСШ AES (Rijndael). При цьому функція PRG, що обчислюється із ключа користувача, одержує ключове значення, необхідне у внутрішній операції UHASH, і також заснована на AES у режимі зворотного зв'язку.

UHASH 16 використовує 16-бітові слова, представляючи їх як підписані цілі (рис. 2.71).

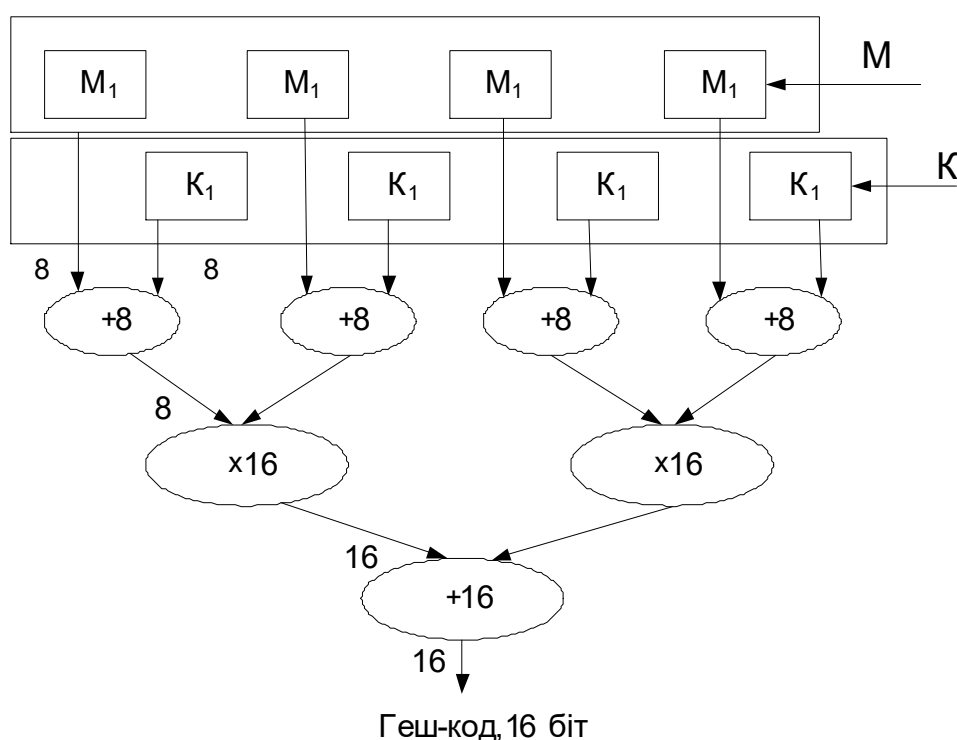


Рис. 2.71. Схема роботи алгоритму UMAC-16

Шар ґешу NH діє на блоках 2 Кб, стислих в 32-бітові величини (відповідає коефіцієнту стиску 512). Імовірність помилки виявилася не більше 2^{15} . Результат проходить шар ґешу RP, що обчислює вихідний рядок установленної довжини 128 бітів. Сімейство ґеш RP є конструкцією, що використовує три області значень із 32-бітовими, 64-бітовими і 128-бітовими первинними модулями відповідно. Довжина повідомлення обмежується максимумом 2^{64} бітів, і доведено, що цей шар значно збільшує ймовірність зіткнення (близько 2^{19}). Якщо автентифіковане повідомлення досить невелике, по-перше, відпадає необхідність у шарі RP і даний шар пропускається як оптимізація. Шар ґешу IP додає своє 128-бітове уведення

до 16-бітового виходу, створюючи ймовірність зіткнення майже 2^{15} . Тришарова конструкція повторюється множиною разів, з незалежними ключами для збільшення довжини коду автентифікації та зниження шансів підробки MAC-коду. За замовчуванням кількість повторень – чотири рази, і додавання 16-бітових вихідних величин дає 64-бітовий MAC-код з ймовірністю підробки 2^{60} .

Основні розходження з алгоритмом *UNHASH 32* полягають у використанні 32-бітових слів і дворазовому повторенні тришарових схем (за замовчуванням). Це дає різні можливості на практиці (використання більшої множини значень), але аналіз, головним чином, залишається колишнім. На рис. 2.72 зображено схему роботи алгоритму UMAC32.

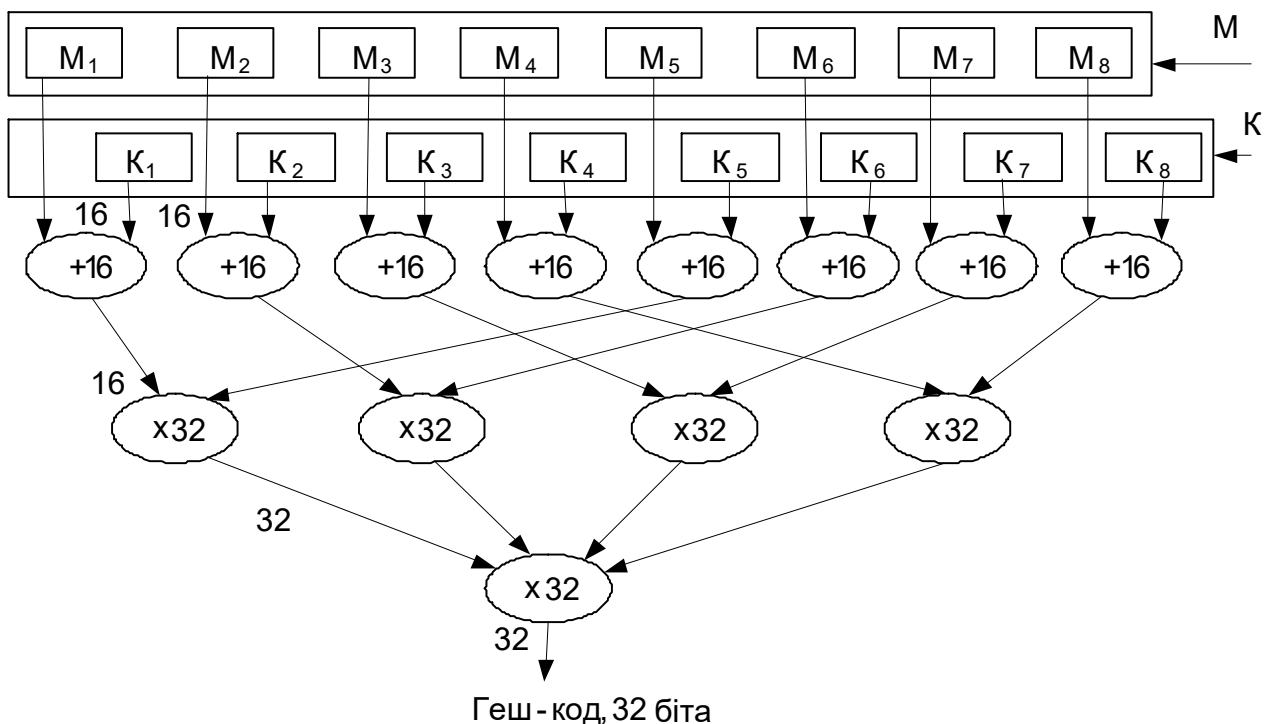


Рис. 2.72. **Схема роботи алгоритму UMAC32**

Переваги перед попередньою версією UMAC (1999 року) полягають у використанні мінімізованого шифрувального примітива (AES), що (у результаті) дає більшу ефективність для коротких повідомлень і забезпечує додаткову гнучкість верифікації: можна вибрати кількість паралельних ітерацій в обчисленні MAC, тим самим варіюючи обчислювальним часом для забезпечення заданого рівня безпеки.

Було розглянуто дві специфічних версії UMAC (UMAC32 і UMAC16). Інші версії UMAC можуть розглядатись згідно з параметрами, що описані нижче.

1. Довжина слова (розрядність).

Визначає розмір у бітах елементарної операції, яка лежить в основі роботи алгоритму. Дозволені значення складають 16 бітів (UMAC16) і 32 біти (UMAC32). Загалом робота UMAC буде ефективною, якщо процесор, на якому він працює, має хорошу підтримку для операцій над типами даних визначеного розміру (розрядність). Це випадок, якщо довжина слова вибрана рівною власній розрядності процесора, але в деяких випадках довжина слова, менша, ніж розрядність процесора (наприклад, коли це дозволяє експлуатувати невеликий векторний паралелізм, що підтримується процесором).

2. Довжина ключа.

Визначає розмір у бітах ключа, що вводиться користувачем. Як для UMAC32, так і UMAC16 довжина ключа складає 128 бітів. Альтернативно можливо вибрати ключ 256 бітів. У даному випадку функція KDF базується на AES з 256-бітному ключі. До того ж отриманий ключ K внаслідок використання для AES-кодування початкового заповнення повинен мати довжину 256 бітів (замість 128).

3. Довжина ґеш-коду.

Визначає розмір у бітах ґеш-коду UMAC. Довжина типового ґешу складає 64 біти, хоча можливі довжини ґеш-коду 32, 64, 96 або 128 бітів у специфікаціях UMAC32 і UMAC16 для того, щоб забезпечити різні рівні стійкості. Для UMAC можлива будь-яка довжина ґешу, аж до 256 бітів, кратна 8-ми бітам [66].

4. Довжина блоку.

Визначає бітовий розмір блоків, на які повідомлення ділиться на першому рівні HASH. Ця довжина рівна $2^{13} = 8192$ бітам для UMAC32 і UMAC16. Можливі також, $2^8, 2^9, \dots, 2^{28}$ біти.

Можливе середнє стиснення довжини блоку і вища продуктивність, але для цього необхідний більший об'єм пам'яті для зберігання внутрішніх ключів. Коли дані, що розбиваються, і ключ використовуються одночасно меншим за розрядністю процесором, продуктивність зменшується. Другий рівень HASH пропускається, коли повідомлення не довше, ніж один блок.

6. Наявність підпису рядка.

Конкретизує, чи були задіяні рядки в ґеш-функції. Вони розглядаються спочатку як підписані або не підписані цілі числа. Для UMAC16 використовується представлення підписаних чисел, для UMAC32 – не підписаних.

Проаналізувавши характеристики *MDC*-кодів і *MAC*-кодів, можна зробити висновок, що *MAC*-коди є більш стійкими і, на відміну від *MDC*-кодів, не потребують додаткових алгоритмів для шифрування повідомлення.

Аналіз методів побудови універсальних класів ґеш-функцій

Універсальні класи ґеш-функцій вперше були запропоновані Картером і Вегманом [41; 72], і використовуються в першому слої ґеш-функції *UNASH*-алгоритмів переможців міжнародного криптографічного конкурсу *NESSIE*. Розглянемо основні визначення для опису універсальних класів.

Нехай A і B будуть кінцевими множинами з відповідних елементів $a = |A|$ і $b = |B|$, де $a \geq b$. Функція $h: A \rightarrow B$ буде названа ґеш-функцією. Для ґеш-функції h і для $x, y \in A$, $x \neq y$, визначимо $\delta h(x, y) = 1$, якщо $h(x) = h(y)$ і $\delta h(x, y) = 0$ у зворотному випадку.

Тобто $\delta h(x, y) = 1$ тільки тоді, якщо ґешовані значення x і y викликають колізію. Для кінцевої множини H ґеш-функцій визначимо $\delta_{H(x,y)} = \sum_{h \in H} \delta h(x,y)$. Звідси $\delta_{H(x,y)}$ підраховує кількість ґеш-функцій у H , при яких значення елементів x і y викликають колізію.

Ідея універсального класу ґеш-функцій полягає у визначенні набору елементів кінцевої множини H ґеш-функцій таким чином, що випадковий вибір функції $h \in H$ забезпечував би низьку ймовірність того, що для будь-яких різних входів x і y імовірність того, що $h(x) = h(y)$ (імовірність колізії) не може бути більше ε : $P_{\text{кол}} = P(h(x) = h(y)) \leq \varepsilon$, і може бути підрахована як $\delta_{H(x,y)} / |H|$.

Визначення універсального класу ґеш-функцій еквівалентне визначенню такого алгоритму формування коду автентифікації, при якому виконується наступна умова: число різних правил формування коду автентифікації (кількість ключів), при якій існує колізія (збіг кодів автентифікації) для двох довільних вхідних послідовностей, обмежене. Кількість таких ключів не може перевищувати значення $P_{\text{кол}} \times H$, де $P_{\text{кол}}$ – імовірність колізії, H – кількість всіх правил (ключів).

Теорема 2.17. Для будь-якого класу H ґеш-функцій від A до B існують різні елементи $x, y \in A$, такі, що $\delta_{H(x,y)} \geq |H|(a - b)/(b(a - 1))$, де $a = |A|$ і $b = |B|$.

Наведемо два визначення класів ґеш-функцій.

1. Нехай $0 < \varepsilon < 1$. $H \in \varepsilon$ – універсальним ґеш-класом (скорочено $\varepsilon - U(H, a, b)$), якщо для двох різних елементів $x_1, x_2 \in X$ існує не більше ніж $H \times \varepsilon$ функцій $f \in H$ таких, що $h(x) = h(y)$, якщо $\delta_H(x, y) \leq \varepsilon|H|$ для всіх $x, y \in A, x \neq y$.

2. Нехай $0 < \varepsilon < 1$. $H \in \varepsilon$ – строго універсальним ґеш-класом (скорочено $\varepsilon - SU(H, a, b)$), якщо виконуються наступні умови:

2.1. Для кожного $x_1 \in A$ та для кожного $y_1 \in B$,

$$|\{h \in H : h(x_1) = y_1\}| = |H|/|B|.$$

2.2. Для кожного $x_1, x_2 \in A (x_1 \neq x_2)$ і для кожного $y_1, y_2 \in B$,

$$|\{h \in H : h(x_1) = y_1, h(x_2) = y_2\}| \leq \varepsilon|H|.$$

Визначення строго універсального класу ґеш-функцій еквівалентне визначенню алгоритму формування кодів автентифікації. Використовуючи строго універсальний клас ґеш-функцій, можна побудувати схему автентифікації, при якій будуть виконуватися наступні правила:

1. Кількість правил формування коду автентифікації (кількість ключів), при яких для довільної вхідної послідовності значення коду автентифікації не змінюється, обмежене. Кількість таких ключів не може перевершувати значення $\frac{H}{2^a}$, де H – кількість всіх ключів, a – кількість бітів коду автентифікації.

2. Кількість правил формування коду автентифікації (кількість ключів), при яких для двох довільних вхідних послідовностей відповідні їм значення коду автентифікації не змінюються, обмежена. Кількість таких ключів не може перевершувати значення $P_{\text{кол}} \cdot \frac{H}{2^a}$, де $P_{\text{кол}}$ – імовірність колізії, H – кількість всіх ключів, a – кількість бітів коду автентифікації.

Імовірність колізії кодів автентифікації в такій схемі буде визначатися наступним твердженням $P_{\text{кол}} \leq \varepsilon$.

З погляду того факту, що класи ґеш-функцій приводять до автентифікованих кодів, є інтерес порахувати нижні границі кількості необхідних ґеш-функцій. Розглянемо деякі раніше відомі границі.

Теорема 2.18. Якщо існує $\varepsilon - \text{SU}(H, a, b)$ (клас ґеш-функцій H від A до B), де $a = |A|$ і $b = |B|$, тоді:

$$|H| \geq \frac{a(b-1)}{a(\varepsilon b - 1) + b^2(1-\varepsilon)}.$$

Підставивши $\varepsilon = 1/b$ і $\varepsilon = (a-b)/(ab-b)$ у наведену вище границю, відповідно одержимо наслідок.

Наслідок 2.1. Припустимо, H – це клас ґеш-функцій від A до B , де $a = |A|$ і $b = |B|$. Якщо $H \in \varepsilon - \text{U}(H, a, b)$, тоді $|H| \geq a/b$. Якщо $H \in \varepsilon - \text{SU}(H, a, b)$, тоді $|H| \geq (a-1)/(b-1)$.

Далі представимо нижню границю на кількість ґеш-функцій в $\varepsilon - \text{SU}(H, a, b)$ – класі.

Теорема 2.19. Якщо існує $\varepsilon - \text{SU}(H, a, b)$ клас H ґеш-функцій від A до B , де $a = |A|$ і $b = |B|$, тоді:

$$|H| \geq 1 + \frac{a(b-1)^2}{b\varepsilon(a-1) + b - a}.$$

У випадку $\varepsilon = 1/b$ одержимо наступний наслідок.

Наслідок 2. Якщо існує $\varepsilon - \text{SU}(H, a, b)$ клас H ґеш-функцій від A до B , де $a = |A|$ і $b = |B|$, тоді $|H| \geq 1 + a(b-1)$.

Якщо $|H| = 1 + a(b-1)$, тоді існує $OA(b, a, \lambda)$, де $\lambda = (a(b-1)+1)/a^2$ ($OA(b, a, \lambda)$ – ортогональний масив).

Розглянемо деякі прямі та рекурсивні конструкції для універсальних класів ґеш-функцій.

Теорема 2.20. Нехай q буде степенем простого числа. Тоді існує $\varepsilon - \text{SU}(H, a, b)$ – клас H ґеш-функцій від A до B , де $|A| = q^2$, $|B| = q$ і $|H| = q$ (звідси $\varepsilon = 1/q$).

Доведення. Нехай $A = \text{GF}(q) \times \text{GF}(q)$, $B = \text{GF}(q)$ і $H = \{h_x : x \in \text{GF}(q)\}$, де $h_x(a, b) = b - ax$.

Теорема 2.21. Нехай q буде степенем простого числа. Тоді існує $\varepsilon - \text{SU}(H, a, b)$ – клас H ґеш-функцій від A до B , де $|A| = q^2$, $|B| = q$ і $|H| = q^3$ (звідси $\varepsilon = 1/q$).

Доведення. Нехай $A = \text{GF}(q) \times \text{GF}(q)$, $B = \text{GF}(q)$ і

$H = \{h_{xyz} : x, y, z \in \text{GF}(q)\}$, де $h_{xyz}(a, b) = x + ay + bx$.

Теорема 2.22. Нехай q буде степенем простого числа. Тоді існує $\varepsilon - SU(H, a, b)$ – клас H ґеш-функцій від A до B , де $|A| = q^2$, $|B| = q$ і $|H| = q^2$ (звідси $\varepsilon = 1/q$).

Доведення. Нехай $A = B = GF(q)$ і $H = \{h_{xy} : x, y \in GF(q)\}$, де $h_{xy}(a) = x + ay$.

Розглянемо деякі методи комбінування класів ґеш-функцій, які узагальнюють подібні конструкції.

Теорема 2.23. (Декартовий добуток). Якщо існує $\varepsilon - U(H, a, b)$ – клас H ґеш-функцій від A до B , тоді для будь-якого цілого $i \geq 1$ існує $\varepsilon - U(H, a, b)$ клас H^i ґеш-функцій від A^i до B^i з $|H| = |H^i|$.

Доведення: Для кожного $h \in H$ визначимо ґеш-функцію $h^i : A^i \rightarrow B^i$ за правилом $h^i : (a_1, \dots, a_i) = (h(a_1), \dots, h(a_i))$. Зазначимо $H^i = \{h^i : h \in H\}$.

Теорема 2.24. (Композиція 1). Для $i = 1, 2$ припустимо, що існує $\varepsilon - U(H, a, b)$ клас H_i ґеш-функцій від A_i до B_i , де $A_2 = B_1$. Тоді існує $\varepsilon - U(H, a, b)$ клас H ґеш-функцій від A_1 до B_2 , де $\varepsilon = \varepsilon_1 + \varepsilon_2$ і $|H| = |H_1| \times |H_2|$.

Доведення: Для кожних $h_i \in H_i, i = 1, 2$ визначимо ґеш-функцію $h : A_1 \rightarrow B_2$ за правилом $h(a) = h_2(h_1(a))$. Нехай H буде множиною всіх таких ґеш-функцій. Для двох будь-яких входів імовірність колізії не більше $\varepsilon_1 + (1 - \varepsilon_1)\varepsilon_2(\varepsilon_1 + \varepsilon_2)$.

Теорема 2.25 (Композиція 2). Припустимо, $H_1 \in \varepsilon - U(H, a, b)$ класом ґеш-функцій від A_1 до B_1 , і припустимо $H_2 \in \varepsilon - SU(H, a, b)$ класом ґеш-функцій від B_1 до B_2 . Тоді існує $\varepsilon - SU(H, a, b)$ клас H ґеш-функцій від A_1 до B_2 , де $\varepsilon = \varepsilon_1 + \varepsilon_2$ і $|H| = |H_1| \times |H_2|$.

Доведення. Для кожних $h_i \in H_i, i = 1, 2$ визначимо ґеш-функцію $h : A_1 \rightarrow B_2$ за правилом $h(a) = h_2(h_1(a))$. Нехай H буде множиною всіх таких ґеш-функцій. Нехай $x_1, x_2 \in A_1$ ($x_1 \neq x_2$) і нехай $y_1, y_2 \in B_2$. Скільки функцій у H відображають x_1 в v_1 і x_2 в v_2 ? Припустимо, що $v_1 = v_2$. Тоді максимальна кількість колізії не більше

$$P|H_1| \times \frac{|H_2|}{b} + (1 - P)|H_1| \times \frac{\varepsilon_2|H_2|}{b} \leq (\varepsilon_1 + \varepsilon_2)|H_1| \times \frac{|H_2|}{b}.$$

Якщо $v_1 \neq v_2$, тоді кількість колізій менше. Оскільки $P \leq \varepsilon_1$, треба, що $\varepsilon - SU(H, a, b)$ клас із $\varepsilon = \varepsilon_1 + \varepsilon_2$.

Розглянемо отримані конструкції для одержання автентифікованих кодів.

Теорема 2.26. Нехай q буде степенем простого числа та нехай $i \geq 1$ буде цілим числом. Тоді існує $\frac{i}{q} - \text{SU}(H, a, b)$ клас q^i ґеш-функцій від A до B , де $|A| = q^{2^i}$ і $|B| = q$.

Доведення. Застосуємо доказ теорем 2.16, 2.19 і 2.21.

Теорема 2.27. Нехай q буде степенем простого числа і нехай $i \geq 1$ буде цілим числом. Тоді існує $\frac{i+1}{q} - \text{SU}(H, a, b)$ клас q^{i+2} ґеш-функцій від A до B , де $|A| = q^{2^i}$ і $|B| = q$.

Доведення. Застосуємо доказ теорем 2.15 і 2.21, 2.23.

Теорема 2.28. Нехай q буде степенем простого числа і нехай $i \geq 1$ буде цілим числом. Тоді існує $\frac{i}{q} + \frac{1}{q} - \text{SU}(H, a, b)$ клас q^{2i+3} ґеш-функцій від A до B , де $|A| = q^{2^i}$ і $|B| = q$.

Доведення. Застосуємо доказ теореми 2.23 (із заміною q на q^2), 2.15 і 2.22.

Таким чином, представлені конструкції дозволяють будувати конструкції сімейств універсальних і строго універсальних класів.

Алгоритм на основі RSA

У четвертій частині стандарту ISO/IEC 10118-4 визначені ґеш-функції, що використовують модулярну арифметику. Стандарт уводить дві ґеш-функції MASH-1 і MASH-2, які використовують модульне зведення в степінь для побудови ґеш-коду. Основною ідеєю ґеш-функцій є використання ітеративної функції як циклової функції, яка використовує модулярну арифметику. Причинами стандартизації і застосування таких ґеш-функцій є, по-перше, можливість використання існуючих програмних і апаратних засобів модулярної арифметики, які застосовуються у несиметричних криптосистемах, і, по-друге, які застосовуються необхідного рівня стійкості. Основним недоліком функцій є низька швидкість формування ґеш-коду.

ґеш-функція MASH-1 (Modular Arithmetic Secure Hash-1) використовує RSA-подібні модулі N , довжина яких забезпечує необхідну стійкість.

Число N повинно важко розкладатися, на чому і ґрунтується стійкість алгоритму. Розмір модуля визначає довжину блоків оброблюваного повідомлення, а також розмір ґеш-коду (наприклад, 1025-бітний модуль забезпечує формування 1024-бітного ґеш-коду).

Алгоритм MASH-1.

Вхід. Двійковий рядок x довжиною $0 \leq b \leq 2^{n/2}$.

Вихід. n -розрядний ґеш-код за рядком x , довжина якого приблизно дорівнює довжині модуля N .

1. Системні установки і визначення констант. Установити RSA-подібний модуль $N = pq$ довжиною m -бітів, де p і q випадково обрані великі прості числа, що зберігаються в секреті. Визначити двійкову довжину n ґеш-коду як найбільший добуток числа 16 і довжини ґеш-коду, що задовольняє нерівності $16 \times n < m$.

Як вектор ініціалізації вибрати $H_0 = 0$. Визначити n -бітне ціле число як константу $A = f00...00_x$.

2. Передобробка. Доповнити, якщо необхідно, рядок x нульовими бітами, для того щоб одержати двійковий рядок довжиною $t \times n/2$ для найменшого $t \geq 1$. Розділити доповнений текст на $n/2$ -розрядні блоки $x_1, \dots, x_t$ і додати останній блок x_{t+1} , що містить $n/2$ -розрядне представлення числа b .

3. Розширення. Розширити кожен блок x_i у n -розрядний блок y_i шляхом вставки між 4-розрядними напівбайтами блоку x_i комбінації з чотирьох одиниць (1111). Останній блок y_{t+1} формується аналогічним чином за винятком того, що вставляється комбінація 1010.

4. Циклова функція. Для всіх $1 < i \leq t + 1$ відобразити два n -розрядних вхідних блоки (H_{i-1}, y_i) в один n -розрядний блок відповідно до виразу

$$H_i = (((H_{i-1} \oplus y_i) \vee A)^2 \bmod N) \perp n \oplus H_{i-1},$$

де $\vee$ – операція побітного логічного АБО;

$\oplus$ – додавання за модулем 2 (XOR);

$\perp$ – збереження молодших n -розрядів m -розрядного результату.

5. Закінчення. За ґеш-код приймається n -розрядний блок H_{t+1} .

Алгоритм MASH-2 відрізняється від алгоритму MASH-1 тільки показником степеня в цикловій функції, що має такий вигляд:

$$H_i = \left(\left(((H_{i-1} \oplus y_i) \vee A)^{2^8+1} \bmod N \right) \perp n \right) \oplus H_{i-1}.$$

Розмір модуля визначає довжину блоків оброблюваного повідомлення, а також розмір ґеш-коду (рис. 2.73).

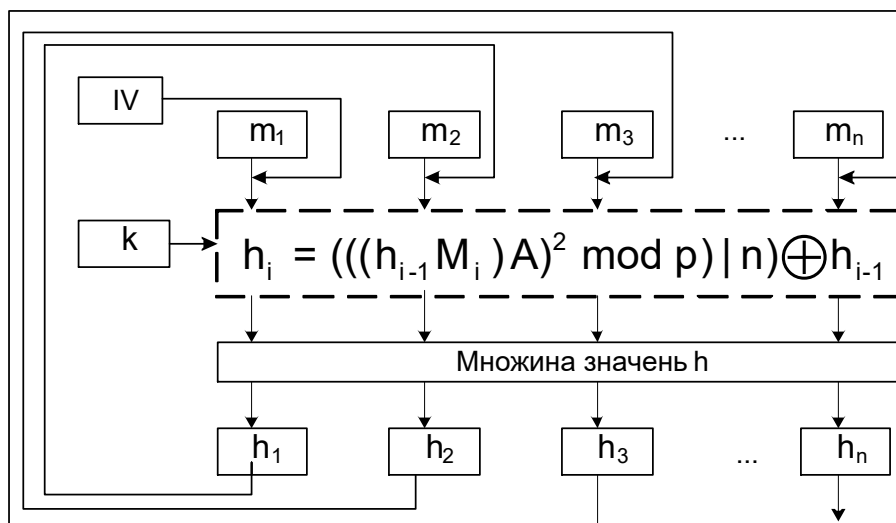


Рис. 2.73. Схема роботи алгоритму MASH-1

Алгоритм MASH-2 відрізняється від алгоритму MASH-1 тільки показником степеня в цикловій функції (рис. 2.74).

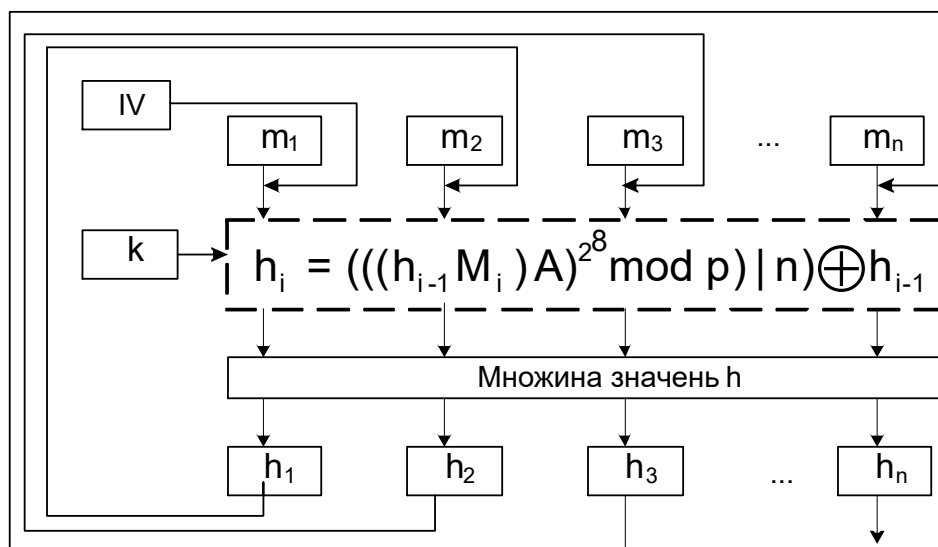


Рис. 2.74. Схема роботи алгоритму MASH-2

Контрольні запитання до теми 2

1. Механізми забезпечення конфіденційності на основі сучасних симетричних алгоритмів шифрування.
2. Принципи блокового шифрування.
3. Поняття секретності системи шифрування. Досконала секретність.
4. Ентропія і невизначеність. Інтенсивність і надмірність мови.
5. Відстань єдності й ідеальна секретність.
6. Режими роботи блочних шифрів.
7. Розподіл ключів. Сценарії розподілу ключів. Управління ключами.
8. Криптоаналіз симетричних сучасних схем шифрування.
9. Принципи побудови криптосистем з відкритим ключем.
10. Умови застосування методів криптографії з відкритим ключем.
11. Захищеність та обчислювальні аспекти алгоритму RSA.
12. Криптографічні перетворення на еліптичних кривих.
13. Механізми забезпечення автентичності на основі сучасних асиметричних процедур шифрування, MAC-кодів.
14. Сучасні алгоритми ґешування.
15. Автентифікація повідомлень і ґеш-функція ґешування.
16. Прості функції ґешування. Захист функцій ґешування.
17. Електроні підписи і протоколи автентифікації.
18. Стандарти електронного підпису.
19. Розподіл ключів за допомогою системи з відкритим ключем.
20. Структура Файстеля алгоритму БСШ.
21. SPN-структура алгоритму блокового симетричного шифрування.
22. Типи структур поточкових алгоритмів шифрування.
23. Алгоритм блокового симетричного шифрування DES.
24. Алгоритми блокового симетричного шифрування ГОСТ-28147.
25. Диференційний криптоаналіз на основі відмов пристрою.
26. Лінійний криптоаналіз.
27. Комбінований метод шифрування.

Використана література до теми 2

1. Кузнецов О. О. Захист інформації в інформаційних системах. Методи традиційної криптографії / О. О. Кузнецов, С. П. Євсєєв, О. Г. Король. – Харків: Вид. ХНЕУ, 2010. – 315 с. (Укр. мов.)
2. Кузнецов О. О. Захист інформації в інформаційних системах./ О. О. Кузнецов, С. П. Євсєєв, О. Г. Король. – Харків: Вид. ХНЕУ, 2010. – 513 с. (Укр. мов.)
3. Євсєєв С.П. Лабораторний практикум з основ криптографічного захисту: Навчальний посібник /С.П. Євсєєв, О.В. Мілов, О.Г. Король – Харків : Вид. ХНЕУ ім. С. Кузнеця, 2020. – 252 с. (Укр. мова)
4. Основи криптографії: Навчальний посібник / Остапов С.Е., Валь Л.О. – Чернівці: Книги – XXI, 2008. – 188 с.
5. Кузнецов О. О. Захист інформації та економічна безпека підприємства : монографія / О. О. Кузнецов, С. П. Євсєєв, С. В. Кавун. – Харків : Вид. ХНЕУ, 2008. – 360 с.
6. Algorithmic Number Theory: Lattices, Number Fields, Curves and Cryptography / J. P. Buhler, P. Stevenhagen. - Cambridge University Press, Mathematical Sciences Research Institute, 2008, - 665 p.
7. Hoffstein J. An Introduction to Mathematical Cryptography. – Springer, 2008. – 533 p.
8. An Introduction to Number Theory with Cryptography / Kraft J.S., Washington L.C. - Taylor & Francis Group, LLC, 2018. – 601 p.
9. Applied Algebra; Codes, Ciphers, and Discrete Algorithms / Hardy, D.W. - Taylor & Francis Group, LLC, 2009. – 424 p.
10. Boolean Functions and Their Applications in Cryptography / Chuan-Kun Wu, Dengguo Feng. - Springer-Verlag Berlin Heidelberg, 2016. – 267 p.
11. A Classical Introduction to Cryptography Exercise Book / Thomas Baignères, Pascal Junod, Yi Lu, Jean Monnerat, Serge Vaudenay. – Springer, 2006. – 262 p.
12. A Classical Introduction to Modern Cryptography. Applications for Communications Security / Serge Vaudenay. Swiss Fédérale Institute of Technologies (EPFL), 2006. – 344 p.
13. Schneier B. Applied Cryptography: protocols, algorithms and source code in C. - John Wiley & Sons Inc., 2015. – 784 p.
14. Cryptography in C and C++ / Michael Welschenbach. – APRESS, 2005. – 504 p.

15. Mollin, Richard A. An Introduction to Cryptography / Richard A. Mollin. Taylor & Francis Group, LLC, 2007. – 394 p. (Discrete mathematics and its applications)
16. Chaos-Based Cryptography. Theory, Algorithms and Applications / Ljupco Kocarev, Shiguo Lian (Eds.). - Springer-Verlag Berlin Heidelberg, 2011. – 408 p.
17. Sumeet Dua. Data Mining and Machine Learning in Cybersecurity / Sumeet Dua and Xian Du. - Taylor and Francis Group, LLC, 2011. 248 p.
18. Applied Cryptography for Cyber Security and Defense: Information Encryption and Cyphering / Hamid R. Nemati and Li Yang. - Information Science Reference, 2011.- 408 p.
19. William Stallings. Cryptography and Network Security: Principles and Practice / William Stallings. Pearson Education, Inc, 2014. – 760 p.
20. Cryptography Made Simple / Nigel P. Smart. - Springer International Publishing Switzerland, 2016. – 478 p.
21. Cryptography: theory and practice / Douglas R. Stinson and Maura B. Paterson. - CRC Press, Taylor & Francis Group, 2018. – 599 p.
22. Cryptography Engineering: Design Principles and Practical Applications / Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. Wiley Publishing, Inc., 2010. – 385 p.
23. Cybercryptography: Applicable Cryptography for Cyberspace Security / Song Y. Yan. - Springer Nature Switzerland AG, 2019. - 443 p.
24. Andreas Klein. Stream Ciphers / Klein A. - Springer-Verlag London 2013. – 400 p.
25. Hankerson. Darrel R. Guide to elliptic curve cryptography / Darrel Hankerson, Alfred J. Menezes, Scott Vanstone. - Springer-Verlag New York, Inc., 2004. – 332 p.
26. Handbook of elliptic and hyperelliptic curve cryptography / Scientific editors, Henri Cohen & Gerard Frey ; authors, Roberto M Avanzi ... [et. al.], Taylor & Francis Group, LLC, 2006. – 844 p.

Тема 3. Основи квантової криптографії

3.1. Небезпека криптографічних технологій

Немає жодного сумніву в тому, що криптографічний захист інформації є одним з наріжних каменів інформаційної безпеки. Особливо активно він використовується на усіх ділянках захисту фінансової інформації: для організації доступу, захисту конфіденційності та цілісності банківської інформації; для підтримки криптовалют та інших цілей інформаційної безпеки. Отже адекватне розгортання та підтримка криптографічного захисту інформації є надзвичайно важливим завданням, розрахованим на середньо- та довгострокову перспективу.

Для криптографічного захисту інформації розроблено багато криптосистем та криптографічних алгоритмів: симетричні та асиметричні системи шифрування для захисту конфіденційності, криптографічні ґеш-функції для процесів аутентифікації та захисту цілісності інформації, системи електронного електронного підпису для електронного документообігу, методи аутентифікації повідомлень на основі симетричних та асиметричних криптосистем та інші криптографічні процедури та протоколи. Ці криптографічні перетворення інформації довгий час безвідмовно служили людям і успішно використовувалися в усіх галузях людської діяльності, пов'язаних із інформаційними технологіями.

Тим не менше, не так давно, у світі з'явилися загрози, які поставили під сумнів цілісну картину криптографічних технологій, що успішно використовувалися до цього часу. Мова йде про квантові комп'ютери та сукупність алгоритмів для них, практичне впровадження яких сприятиме повному перегляду сучасних концепцій криптографічного захисту інформації.

Рух по цьому шляху вже розпочато. Так, у квітні 2016 року Національний інститут стандартів і технологій США (NIST) випустив документ NISTIR 8105 DRAFT Report on Post-Quantum Cryptography, в якому оцінив розвиток квантового комп'ютингу та його вплив на стан криптографічного захисту інформації в недалекому майбутньому. Висновки, до яких прийшли автори документу, скорочено подані у таблиці 3.1.

Вплив квантового комп'ютингу на загальні криптографічні алгоритми

Cryptographic Algorithm	Type	Purpose	Impact from large-scale quantum computer
AES-256	Symmetric key	Encryption	Larger key sizes needed
SHA-256, SHA-3		Hash functions	Larger output needed
RSA	Public key	Signatures, key establishment	No longer secure
ECDSA, ECDH (Elliptic Curve Cryptography)	Public key	Signatures, key exchange	No longer secure
DSA (Finite Field Cryptography)	Public key	Signatures, key exchange	No longer secure

Table 1 - Impact of Quantum Computing on Common Cryptographic Algorithms

Як видно з таблиці, квантовий комп'ютинг (звичайно, при впровадженні масштабованих квантових комп'ютерів) завдає відчутного удару по всіх напрямках криптографії. Зокрема, НІСТ вважає, що для збереження потрібного рівня захисту симетричні криптоалгоритми на зразок AES-256 потребують збільшення довжини криптографічного ключа, функції ґешування SHA-256 або SHA-3 будуть стійкими до зламу за допомогою квантових алгоритмів при збільшенні довжини ґеш-образу. Слід зауважити, що вплив квантового комп'ютингу не залежить (або слабо залежить) від архітектури функції ґешування, оскільки SHA-256 та SHA-3 ґрунтуються на різних підходах: структурі Меркля-Дамгарда та “криптографічної губки” відповідно.

Що стосується асиметричної криптографії, зокрема криптосистем RSA, а також електронного електронного підпису як на основі кінцевих полів, так і на еліптичних кривих, НІСТ вважає їх використання небезпечним.

Звичайно, це не означає, що ми повинні вже зараз ламати встановлені системи захисту, вони ще довгі роки будуть стійкими для практичного використання. Прикладом такого підходу можна вважати відношення до криптографічних ґеш-функцій MD-5, SHA-1, SHA-2, використання перших двох з яких сьогодні вважається небезпечним і

строго не рекомендується, хоча остання залишається практично стійкою, не дивлячись на те, що архітектура її подібна до SHA-1.

Виходячи з наведеного, можна зробити висновок про те, що ми сьогодні є свідками одного з найбільших (якщо не найбільшого) перевороту у криптографічному світі, який можна порівняти, хіба що, з винаходом асиметричної криптографії на рубежі 70-80 років XX сторіччя.

Які ж досягнення квантового комп'ютингу призвели до такої “революції”? Чому квантові комп'ютери можуть зробити те, що вважається технічно неможливим у класичному комп'ютингу? Які квантові алгоритми дозволяють це зробити? І взагалі, які особливості квантових обчислень сприяють такому зростанню обчислювальних потужностей?

Відповіді на ці та інші питання ми спробуємо отримати в наступному розділі, хоча їх складно отримати, не розібравшись детально, що ж таке квантові обчислення. Тому спочатку ми розглянемо основні поняття квантових обчислень, які дозволять нам сформулювати потрібний термінологічний та алгоритмічний апарат.

3.2. Фізичні основи квантових обчислень

В електроніці відомий *закон Мура* [1], згідно якого кількість транзисторів на кристалі інтегральних схем подвоюється кожні 24 місяці (див. рис.3.1). Відповідно, зменшуються розміри цих транзисторів, вдосконалюється технологічний процес. Сьогодні налагоджене масове виробництво мікропроцесорів на основі 20-22-нанометрових технологічних процесів, будуються заводи (наприклад, Fab42, штат Аризона) для 14-нанометрових виробництв, розробляються основи 10-нанометрового процесу. Чи значить це, що обчислювальні потужності, які доступні людству, будуть зростати необмежено? На жаль, відповідь на це питання негативна. Справа в тому, що сучасна технологія вже дуже близько підібралася до розміру атомів речовини. Дійсно, зараз йдеться про 10-нанометровий технологічний процес, розмір елемента в якому (10 нм) усього в 100 разів більший за розмір атома (0.1 нм). Це означає, що кожен елемент транзистора, наприклад, затвор, буде складатися з ~100 атомів. А що далі? Відомо, що побудувати транзистор (та й будь-який інший елемент мікросхем) з одного атома неможливо.

Однак, це ще не все. Сучасні комп'ютери використовують класичні алгоритми та способи збереження інформації. А ці методи для ряду дуже

важливих задач вимагають астрономічних розмірів пам'яті і таких самих запасів часу.

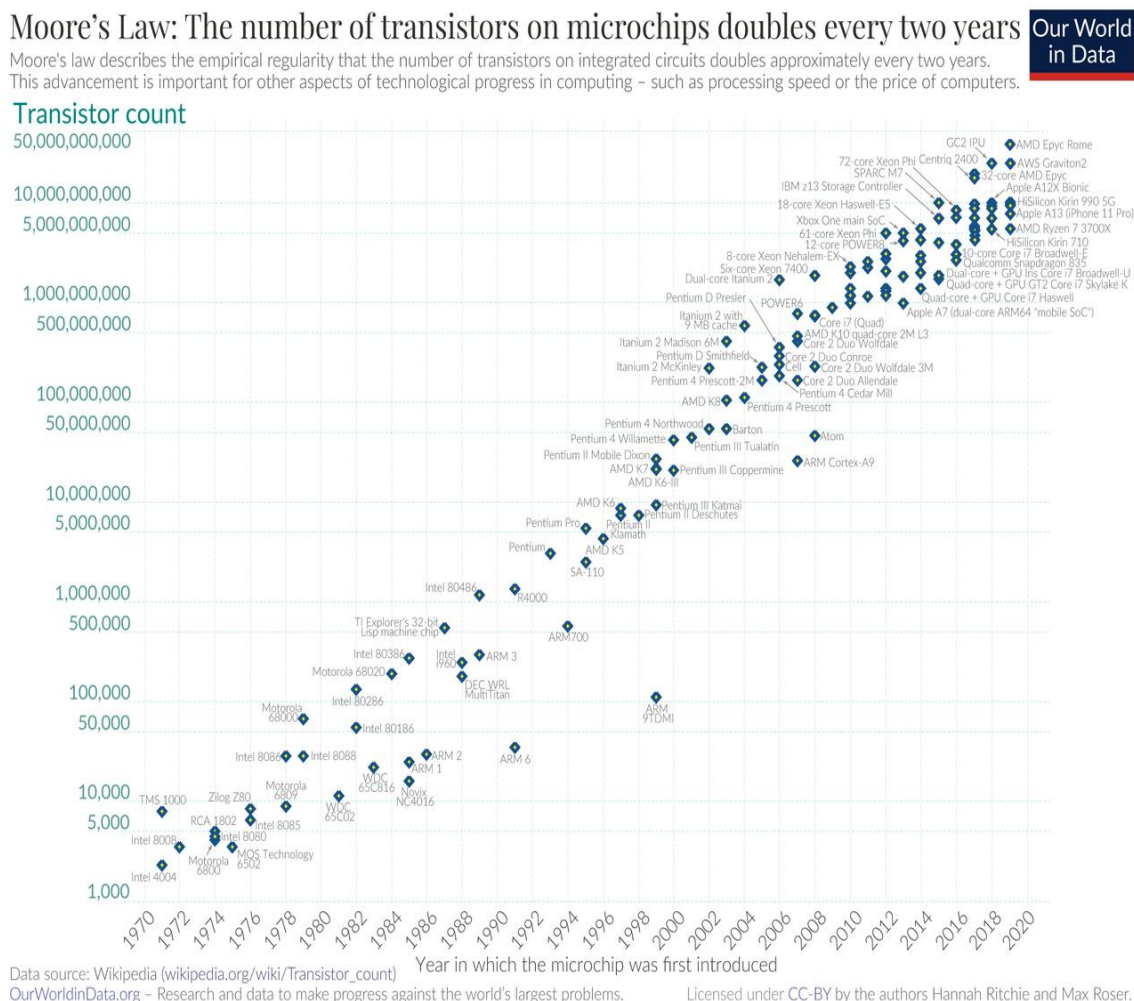


Рис. 3.1. Закон Мура, який демонструє подвоєння кількості транзисторів у процесорах кожні 24 місяці

Наприклад, якщо ми хочемо з максимальною точністю змодельювати еволюцію молекули ДНК, що складається з кількох сотень мільярдів елементарних частинок, то ми повинні будемо керувати одночасно такою кількістю елементів пам'яті, що перевищує кількість елементарних частинок у Всесвіті. Так що навіть якщо би ми навчилися створювати транзистори з одного атома, ряд принципових задач однаково будуть недоступними для наших комп'ютерів. Ще один приклад. Вже зараз сучасні суперкомп'ютери, які моделюють ядерні вибухи, урагани та інші погодні явища, використовують 1,5-2 тисячі процесорів та терабайти оперативної пам'яті, споживаючи енергію, якою могли би користуватися

жителі невеликого міста. Отже, необхідно змінювати принципи виконання обчислень.

Які тут можливі варіанти? Перший, який зараз досить успішно працює, це розпаралелювання обчислень. Сьогодні ми використовуємо багатоядерні процесори, розподілені обчислювальні системи тощо. Однак, й тут існують принципові обмеження. Наприклад, відомий *закон Амдала* [2] стверджує, що час виконання обчислень паралельною системою не може бути меншим за час виконання найдовшого фрагменту коду.

Більше того, якщо подивитися на рисунок 3.2, який ілюструє закон Амдала, то можна побачити, що навіть якби ми розпаралелили 95% усього коду, ми не досягнемо більше як 20-разового прискорення обчислень.

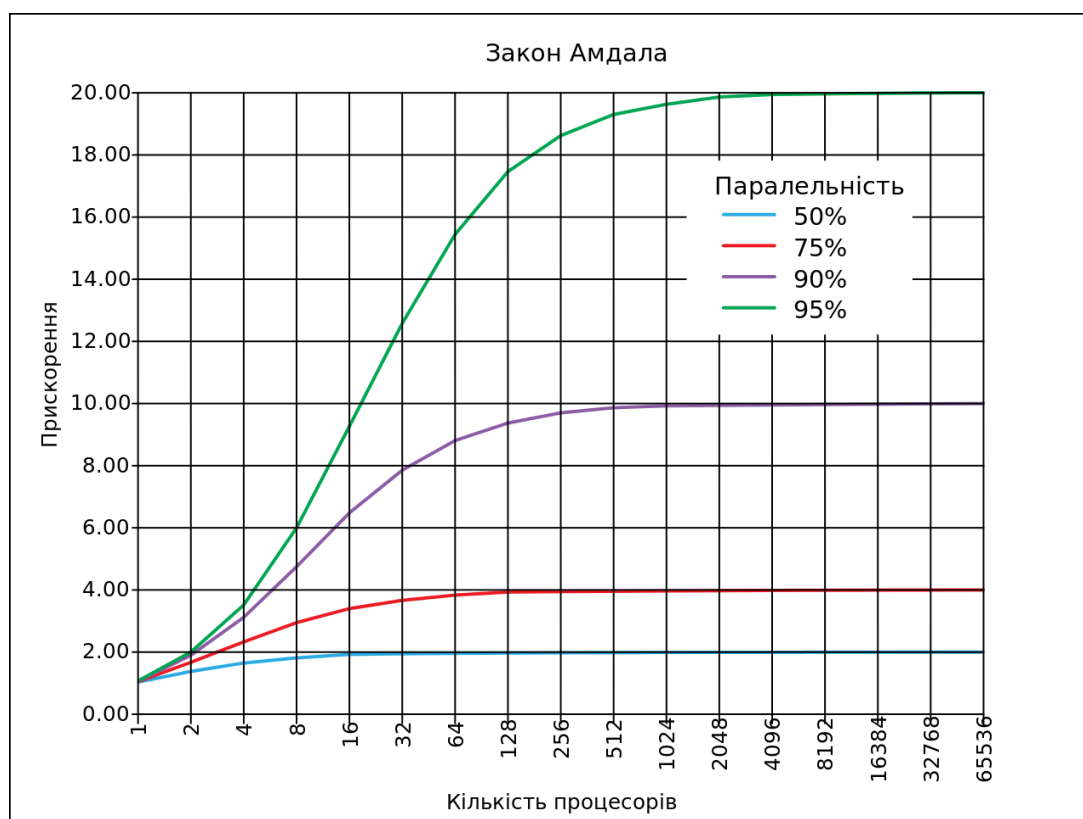


Рис. 3.2. Ілюстрація закону Амдала

Ще одним, правда тимчасовим фактором, який сповільнює прогрес швидкості обчислень, є те, що розробники сучасного програмного забезпечення не зовсім дбають про готовність своїх програм до паралельних обчислень. Свідченням цього можуть служити синтетичні тести процесорів, які час від часу виконують комп'ютерні журнали на кшталт iXBT. На рисунку 3.3 показано результати тестування процесорів 8-ядерного Core i7-5960X та 12-ядерного Xeon E5-4650 v3. З рисунку

видно, що процесор з меншою кількістю ядер, але на вищій тактовій частоті виграє у багатоядерного Xeon.

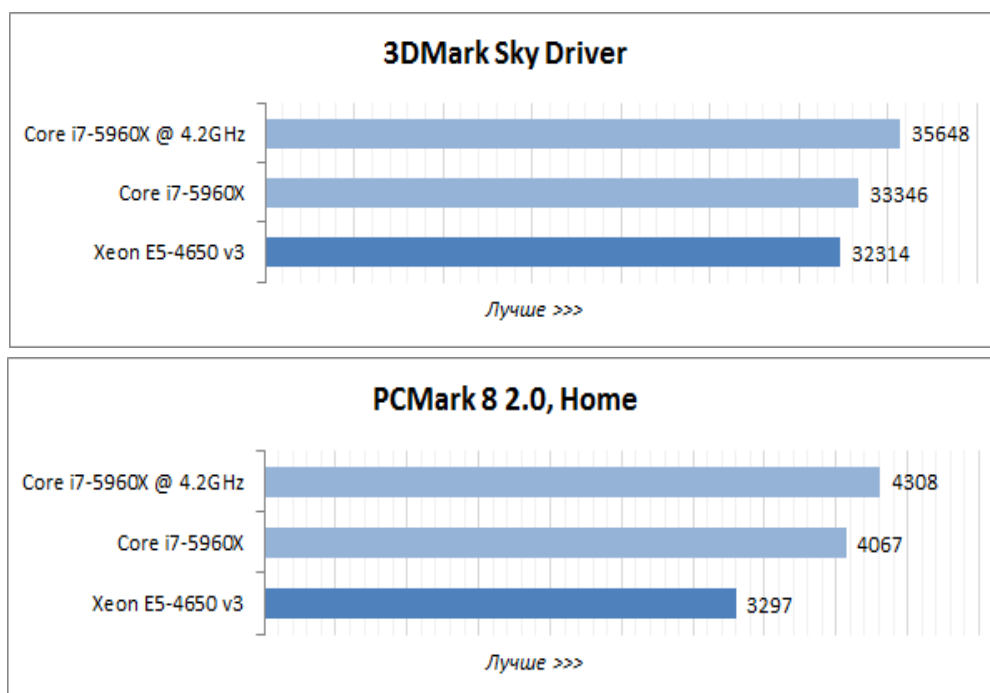


Рис. 3.3. Результати тестування швидкодії процесорів Core i7 порівняно з Xeon E5

І знов ми повертаємось до часу виконання послідовних команд.

Іншим, досить перспективним, варіантом виходу з такого становища могло би бути використання принципів квантових обчислень, які, як ми побачимо далі, мають дивовижні властивості квантового прискорення.

Однак, не все так просто. По-перше, поки квантова система виконує обчислення, доступ до результатів обмежено. Ми не можемо зупинити систему, подивитись проміжні результати і продовжити обчислення. Таке принципово неможливо. По-друге, надзвичайні проблеми виникають сьогодні при практичній реалізації квантових комп'ютерів. І хоча тут досягнуто деяких важливих результатів, говорити про повний успіх в побудові квантових комп'ютерів зарано. Сьогодні існують лише окремі зразки частково-квантових обчислювальних систем *Orion*, які випускає канадська фірма D-Wave, але вони призначені лише для виконання окремих чітко визначених задач. Тим не менше, пригадаємо, що перші комп'ютери, ENIAC, EDVAC та інші також використовувалися лише для однієї задачі.

І усе ж таки, значні переваги квантових комп'ютерів та обчислювальних алгоритмів на основі квантової парадигми, змушують дослідників серйозно займатися цими новітніми технологіями, які можуть спричинити чергову революцію в галузі інформаційних технологій.

В 1890-х роках німецький фізик-теоретик Макс Планк вивчав випромінювання абсолютно чорного тіла. Його теоретичні розрахунки свідчили, що процес випромінювання відбувається не неперервно, а деякими порціями, енергію яких можна описати простою формулою: $E=h\nu$. Тут ν — частота хвилі, що випромінюється, а h — деяка постійна, $h=6,62 \times 10^{-34}$ Дж·с. Цю постійну пізніше назвали постійною Планка, а величину енергії, яку описує запропонована формула — квантом електромагнітної хвилі. Результати досліджень було опубліковано у статті, яка вийшла в друці 1900 року [3]. Таким чином, було доведено, що класичної фізики зовсім недостатньо для того, щоби описати явища мікросвіту. Більше того, мікросвіт, тобто світ елементарних частинок та окремих атомів не підкоряється законам класичної фізики. Це було першими кроками становлення нової фізичної теорії, яка отримала назву квантової механіки.

Таким чином, виходить, що світло поглинається та випромінюється порціями, квантами, які називаються фотонами, а одночасно поводить себе як хвиля, тобто виявляє властивості інтерференції та дифракції.

У 20-х роках ХХ сторіччя французький фізик-теоретик Луї де Бройль довів, що корпускулярно-хвильовий дуалізм (тобто явище, коли частинка, корпускула, поводить себе як хвиля) властивий усім іншим елементарним частинкам. Луї де Бройль увів поняття *хвилі де Бройля*, яка характеризує певну частинку: $\lambda = h/mv$, де λ — довжина хвилі; h — постійна Планка; m , v — маса та швидкість частинки [4]. Сьогодні, правда, поняття корпускулярно-хвильового дуалізму має лише історичне значення, оскільки в 1948 році воно було замінено Річардом Фейнманом описанням елементарних об'єктів за допомогою інтегралів по траєкторіях, що розв'язало проблему корпускулярно-хвильового дуалізму [5]. Тим не менше, для наших цілей це поняття досить корисне, тому ми будемо його використовувати.

Перелічені поняття є одними з основних у квантовій механіці, яка описує поведінку мікрооб'єктів. Ця поведінка настільки не схожа на таку класичних об'єктів, що до неї дуже важко звикнути. Більше того, коли стараннями Нільса Бора, Вернера Гейзенберга, Ервіна Шредінгера та багатьох інших фізиків було створено основи нової теорії, про яку Лев

Ландау пізніше сказав, що “... людина здатна зрозуміти ті речі, які вона вже не в змозі уявити”, більшість фізиків не змогли її сприйняти. Серед них був і один з найвизначніших фізиків за всю історію науки, Альберт Ейнштейн, який якимось відзначив, обговорюючи поняття квантової суперпозиції: “...Бог не буде грати в кості...”, вирішуючи яке значення надати змінній. Альберт Ейнштейн висунув уявний експеримент, який отримав назву парадоксу Ейнштейна-Подольські-Розена, та досі не отримав однозначного пояснення. Цей парадокс ми розглянемо пізніше.

Історія квантових обчислень

У 1970 році аспірант Колумбійського університету Стівен Візнер подав до журналу IEEE Information Theory статтю, але вона не була прийнята до публікації оскільки редакція вважала ідеї, висунуті автором, антинауковими. Крім того, технологічний рівень тих років не дозволяв навіть мріяти про такий рівень захисту. І тільки 1983 року робота Візнера “Conjugate coding” була опублікована в журналі SIGACT News та отримала високу оцінку в наукових колах [6].

Суть ідеї С.Візнера полягає в тому, що кожна грошова банкнота повинна мати 20 “світлових пасток”, в кожній з яких розміщено по одному фотону з строго визначеним станом поляризації, як це показано на рис. 3.4.

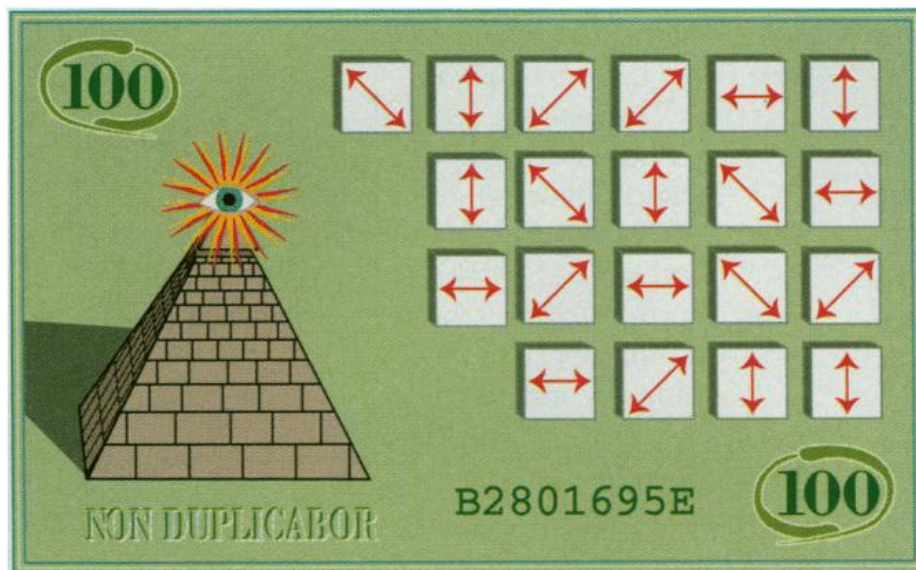


Рис. 3.4. Ескіз банкноти за С.Візнером

Така банкнота маркувалася би спеціальним серійним номером, який би містив деяку інформацію про напрями поляризації кожного фотона. Оскільки фальшивомонетник не знає, яку комбінацію фільтрів було застосовано для запису в пастки, він не зможе правильно зчитати інформацію в пастках, і тим більше, повторити таку комбінацію на фальшивій банкноті. Банк навпаки, зберігає цю послідовність у сховищі, і в будь-який час може перевірити коректність квантового запису на банкноті. Концепція С. Візнера довгий час була (та й зараз лишається) чисто теоретичною, оскільки створити пастки для тривалого зберігання квантового стану не вдається. Сьогодні можуть зберігати квантові частинки у стані суперпозиції на протязі секунд, що надзвичайно мало для такої задачі.

Тим не менше, робота С.Візнера була однією з перших, яка мала відношення до захисту інформації з використанням квантових уявлень.

У 1981 році лауреат Нобелівської премії 1965 року, визначний фізик-теоретик Річард Фейнман (рис. 3.5) на Першому конгресі по обчисленнях, який проходив у Масачусетському технологічному інституті, вперше запропонував використовувати способи обчислень на основі уявлень квантової механіки для моделювання квантових систем [7].



Рис. 3.5. Річард Фейнман

У тому ж році професор Масачусетського технологічного інституту Томмазо Тоффолі запропонував квантовий вентиль Тоффолі, який використовується й сьогодні для побудови квантової логіки [8]. У 1982 році В.Вуттерсом, В.Зуреком та Д.Діксом було висунуто так звану “теорему про заборону клонування” у квантових обчисленнях, яка доводить, що

невідомий квантовий стан частинки неможливо скопіювати, не зруйнувавши самий стан [9].

Приблизно в ті ж роки дослідники почали розробляти принципи квантового зв'язку та квантові протоколи передавання даних. Звичайно, до їхньої практичної реалізації було ще далеко, але основи закладалися саме тоді. Наприклад, у 1984 році було запропоновано перший квантовий протокол узгодження криптографічного ключа, який отримав назву BB84 за першими літерами розробників, Чарлза Беннетта з дослідницької лабораторії IBM та Жіля Brassara з університету Монреалю [10]. Сьогодні цей та інші квантові протоколи узгодження криптографічного ключа разом з поточковими шифрами складають основу квантової криптографії. Ч.Беннетт та С.Візнер у 1992 році запропонували також принципи квантового надщільного кодування, - один з перших квантових протоколів, який надає можливість вибору однієї з чотирьох альтернатив за допомогою передавання одного кубіта [11]. У цьому ж році Чарлз Беннетт розробив модифікований протокол квантового узгодження ключа BB84, який отримав назву B92 [12]. Цей рік був дуже плідотворним для квантової інформатики: Девід Дойч також запропонував алгоритм, який вперше продемонстрував квантове прискорення обчислень [13].

Оскільки існує квантова теорема про заборону клонування, дослідникам необхідно було розробити альтернативний метод перенесення невідомого квантового стану, оскільки без таких операцій створення обчислювальної техніки неможливе. У 1993 році Ч.Беннетт та Ж.Брассар запропонували зовсім нову концепцію, яка отримала назву квантової телепортації [14]. Розроблений протокол дозволяє перенести невідомий квантовий стан на новий квантовий об'єкт, тоді як квантовий стан оригіналу, тобто об'єкта, стан якого переносився, повністю руйнувався. Цей процес не має нічого спільного з телепортацією реальних об'єктів у творах фантастів, оскільки йдеться не про фізичний об'єкт, а про його квантовий стан. Квантову телепортацію сьогодні успішно реалізовано на відміну від "фантастичної" телепортації.

Настала черга складніших алгоритмів, які безпосередньо впливають на стан сучасної криптографічної науки. Зокрема, алгоритм, розроблений американським математиком Пітером Шором у 1994 році, дозволяє факторизувати великі цілі числа майже за той самий час, який потрібен для їх множення [15]. Зрозуміло, що після розробки цього алгоритму усі існуючі асиметричні криптоалгоритми незалежно від того, на якій математичній проблемі ґрунтується їх криптостійкість, можна вважати

скомпрометованими. Це знайшло своє відображення у документах, які ми вже згадували, випущених Національним інститутом стандартів і технологій США у квітні 2016 року (NISTIR 8105 Draft Report), де вважається, що асиметрична криптографія в тому вигляді, як вона застосовується зараз, не повинна більше використовуватися [16].

Через рік, у 1996 році, американський математик індійського походження, Лов Гровер, запропонував алгоритм швидкого пошуку у невпорядкованому масиві, який легко можна застосувати для пошуку криптографічного ключа будь-якого симетричного криптоалгоритму [17]. Тут важливий саме невпорядкований масив, оскільки у впорядкованому ефективний пошук організувати легше класичними алгоритмами, наприклад, методом дихотомії. За оцінками фахівців, існування цього алгоритму змушує збільшити довжину ключа симетричних алгоритмів у 2-3 рази.

Коротко подамо досягнення у технічній реалізації квантових алгоритмів та протоколів:

- у 1997 році групами фізиків під керівництвом Антона Цайлінгера (Інсбрук, Австрія) та Франческо де Мартіні (Рим, Італія) було реалізовано протокол квантової телепортації стану одного квантового біту на відстань 1 метра. У 2012 році відстань, на яку телепортували квантовий стан, досягла вже 143 км;
- у 2001 році спеціалістами IBM реалізовано алгоритм П.Шора на прототипі квантового комп'ютера. Було розкладено число 15 на прості множники, 3 та 5.
- 2005 рік – в Японії продемонстровано програмований квантовий комп'ютер на основі квантового оперативного запам'ятовуючого пристрою з двох квантових бітів.
- 2009 рік – такий же за параметрами квантовий комп'ютер продемонструвало Агентство національної безпеки США.
- 2012 рік – спільнота з чотирьох американських університетів, Південно-Каліфорнійського, Каліфорнійського в Санта-Барбарі, технологічного університету Дельфта та університету штату Айова, реалізувала квантовий комп'ютер з двобітовим оперативним запам'ятовуючим пристроєм, який працює при кімнатних температурах. Пристрій складався з квантових комірок пам'яті – кристалів алмазу з домішками азоту. Повідомляють, що було реалізовано на такому комп'ютері алгоритм Гровера.

Окремим рядком в історії квантових обчислень стоїть канадська компанія D-Wave. У лютому 2007 року керівництвом цієї компанії було оголошено про реалізацію квантового комп'ютера з шістнадцятибітовим процесором (назва комп'ютера – *Orion*). В листопаді 2007 році, буквально через півроку після *Orion*'у було продемонстровано роботу прототипу 28-бітового квантового комп'ютера *Leda* на конференції, присвяченій суперкомп'ютерам, а вже у грудні 2009 року науковий співробітник Google Хартмут Невен продемонстрував на комп'ютері D-Wave роботу програми розпізнавання образів. У травні 2011 року компанія випустила квантовий комп'ютер *D-Wave One* з 128-кубітовим процесором. Планується створення комп'ютера з 1024-кубітовим процесором.

Зараз компанія D-Wave успішно випускає квантові комп'ютери “під замовлення”. Сьогодні це обчислювальні системи, орієнтовані під конкретну задачу, які працюють під глибоким охолодженням (рис. 3.6). Вартість таких систем становить приблизно \$10 млн. долларів США, отже дозволити собі такий комп'ютер можуть лише окремі транснаціональні корпорації. Серед них – Google, який експериментує з розпізнаванням образів за допомогою квантового комп'ютера; Martin-Lockheed, яка використовує квантовий комп'ютер для аналізу критичного програмного забезпечення свого винищувача-бомбардувальника F-35.



Рис. 3.6. Квантовий комп'ютер D-Wave One

Квантові комп'ютери можуть застосовуватись для моделювання ураганів та інших кліматичних явищ, аналізу ДНК та складних ліків, аналізу

складних криптоалгоритмів та інших задач, які не під силу класичним комп'ютерним системам, навіть розподіленим.

Тим не менше, компанія дуже довго знаходилася під вогнем критики з боку експертів у галузі квантових обчислень. Справа в тому, що інженери D-Wave ніколи не демонстрували свої комп'ютери світовій спільноті. Компанія продавала лише машинний час, відповідаючи відмовою на вимоги науковців дозволити дослідити роботу цих комп'ютерів. Відповідно, у науковців виникли підозри про те, чи насправді компанія D-Wave розробила комп'ютер, що задовольняє усі вимоги до квантових обчислювальних систем. Основною вимогою є те, що під час обчислень основні вузли комп'ютера повинні знаходитися у стані квантової суперпозиції.

І лише в травні 2013 року компанія запросила канадського професора Кетрін МакГ'ю для порівняльних тестів швидкодії квантового комп'ютера *D-Wave One* на процесорі *Vesuvius* з тодішнім флагманом фірми Intel, *Core i7*. Оскільки *D-Wave One* був спеціально спроектований для розв'язку задач дискретної оптимізації, тести проводили саме на них. В цих задачах *D-Wave One* виконав тести у 3600 разів швидше за свого конкурента. Прискорення обчислень на інших задачах було незначним. І нарешті, в січні 2014 року інженери D-Wave експериментально довели існування квантової суперпозиції в процесорі під час обчислень, для чого використали кубітову тунельну спектроскопію, що було підтверджено інженерами Google, які виявили, що у 1000-кубітовому комп'ютері D-Wave, який вони використовували, кубіти дійсно організовані у кластери по вісім кубітів. Це дозволило досягти зростання швидкодії до 100 млн разів в одному з алгоритмів порівняно зі звичайним комп'ютером.

У вересні 2016 року на конференції у Сан-Франциско компанія показала свій новий комп'ютер *D-Wave 2X*, який містить 2000 кубітів. Як і раніше, він оптимізований для виконання задач дискретної оптимізації [18].

Таким чином, можна стверджувати, що наприкінці ХХ сторіччя на основі розвитку квантової механіки, теорії алгоритмів та теорії інформації було створено *новий розділ науки: квантову інформатику*, яка вивчає загальні принципи та закони, що керують динамікою складних квантових систем. Моделлю таких систем є *квантовий комп'ютер*. Сьогодні концептуальна математична та алгоритмічна бази квантової інформатики створена й успішно розвивається. Що стосується повноцінного універсального квантового комп'ютера, то вважається, що його ще не

створено. Існують лише квантові комп'ютери, розроблені та оптимізовані для окремих задач. Таку ситуацію людство вже проходило зі звичайними класичними комп'ютерами у 40-х-50-х роках XX сторіччя, коли вони виконували лише одну задачу, для якої їх було розроблено.

Можна очікувати створення універсального квантового програмованого комп'ютера найближчим часом.

Дослід з “кулеметом”

Згадані алгоритми та протоколи квантових обчислень ми розглянемо пізніше. Зараз же опишемо кілька простих прикладів, що дозволять нам частково зрозуміти дивну поведінку квантових об'єктів, яку абсолютно неможливо пояснити класичним чином. Явище квантової суперпозиції, яке нам дуже потрібне надалі, містить саму суть квантової механіки. Однак, ми досі не можемо його однозначно пояснити, тобто не можемо відповісти на питання: “Чому це так?”. Просто розповімо про те, як воно працює [19].

Спочатку розглянемо пристрій, схожий на кулемет (рис. 3.7). Це поганий кулемет. “Кулі”, що він випускає, дуже сильно розсіюються.

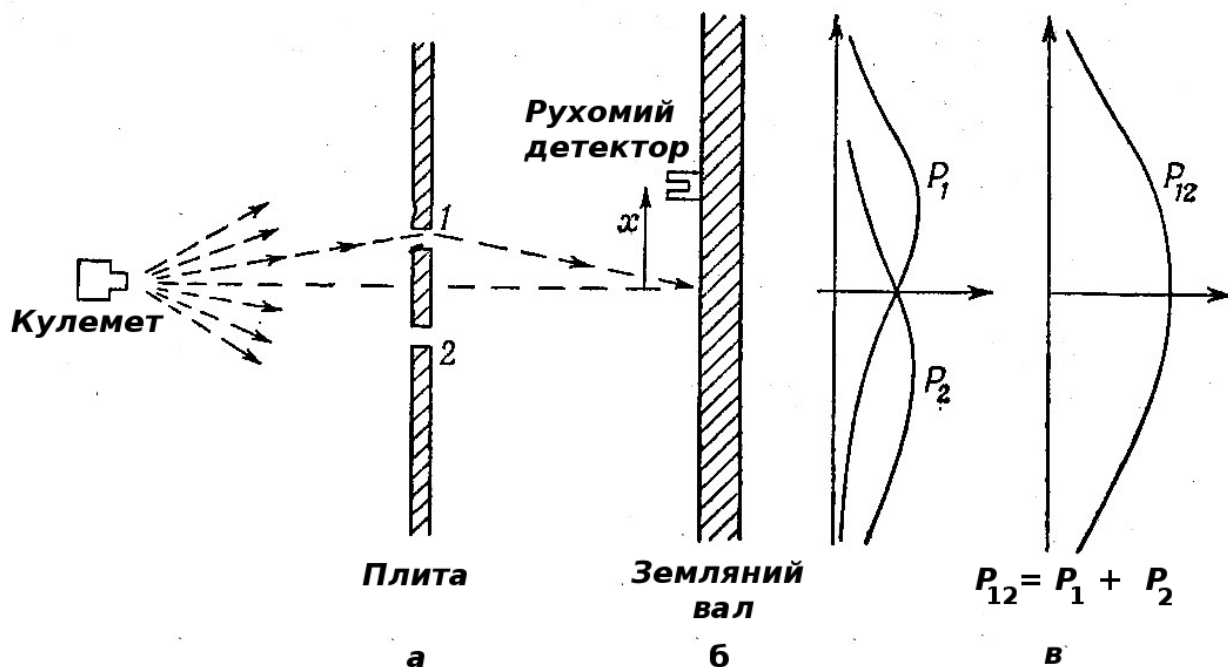


Рис. 3.7. Ілюстрація дослід з "кулеметом": а – схема “установки”; б - ймовірність “влучання” від кожної щілини; в – ймовірність “влучання від обох щілин

Перед кулеметом розташована броньова плита, а в ній є два отвори, 1 і 2, крізь які кулі проходять вільно. За плитою знаходиться земляний вал, куди влучають кулі, що пройшли крізь отвори. Перед земляним валом рухається “детектор”, наприклад, скриня з піском. Будь-яка куля, що влучає в “детектор”, лишається в ньому. Коли дослід закінчено, скриню відкривають та рахують кулі, що влучили у нього. “Детектор” рухається в обох напрямках вісі x і дозволяє відповісти на питання: “Яка ймовірність того, що куля, що пройшла крізь плиту, влучить у земляний вал на відстані x від центру?” Відзначу, що ми говоримо лише про ймовірність, тому що неможливо знати наперед, куди влучить куля. Цю ймовірність можна обчислити, підрахувавши, скільки куль влучило в “детектор” за певний проміжок часу, а потім розділити це число на загальну кількість куль, що влучали у земляний вал за цей же проміжок часу. Якщо швидкість стрільби на протязі експерименту не змінювалася, можна вважати ймовірність пропорційною кількості куль, що влучили у “детектор” за певний час.

Ми будемо вважати, що кулі влучають лише цілими. Не буває половинок куль або четвертих частинок. Це означає, що кулі “приходять” лиш дискретними однаковими порціями, по одній штуці. Навіть якщо швидкість стрільби стає дуже малою, в детекторі завжди знаходиться ціла кількість куль. Розмір порції не залежить від швидкості стрільби.

Результати такого уявного експерименту подані на рисунку 3.7в. Ми позначили ймовірність влучання кулі у детектор через P_{12} , щоби підкреслити, що вони можуть проходити і крізь перший отвір, і крізь другий. Очевидно, що ймовірність у центрі графіку буде більшою, а по його краях — меншою. Тим не менше, не так очевидно, що максимум ймовірності повинен бути при $x=0$, тобто у самому центрі графіку.

Це можна легко зрозуміти, якщо повторити дослід, закривши отвори 1 або 2. У першому випадку ми отримаємо криву P_2 , а в другому - P_1 (рис.3.7б). Максимуми цих кривих, як і слід було чекати, розташовані просто навпроти відкритого отвору. Якщо порівняти рисунки 3.7б та 3.7в, то стає зрозумілим, що у випадку двох відкритих отворів, ймовірності P_1 та P_2 просто додаються, утворюючи максимум рівно в початку координат $x=0$, тобто: $P_{12}=P_1+P_2$. Такий результат ми назвем “відсутністю інтерференції” з причин, про які ви дізнаєтеся пізніше. На цьому ми закінчимо експеримент з “кулеметом”, сформулювавши такі висновки:

- Кулі прибувають лише цілими, «квантуються»; не буває 1,5 або 0,25 куль.

- Закриваючи отвори, ми отримали відповідні ймовірності P_1 або P_2 .
- Якщо обидва отвори відкрито, то загальна ймовірність $P_{12} = P_1 + P_2$.
- Інтерференція відсутня.

Такі висновки абсолютно традиційні для макроскопічних об'єктів, що й не дивно, оскільки кулі і є макроскопічними об'єктами.

Дослід з хвилями

Розглянемо аналогічний дослід з хвилями на воді, схема якого подана на рисунку 3.8.

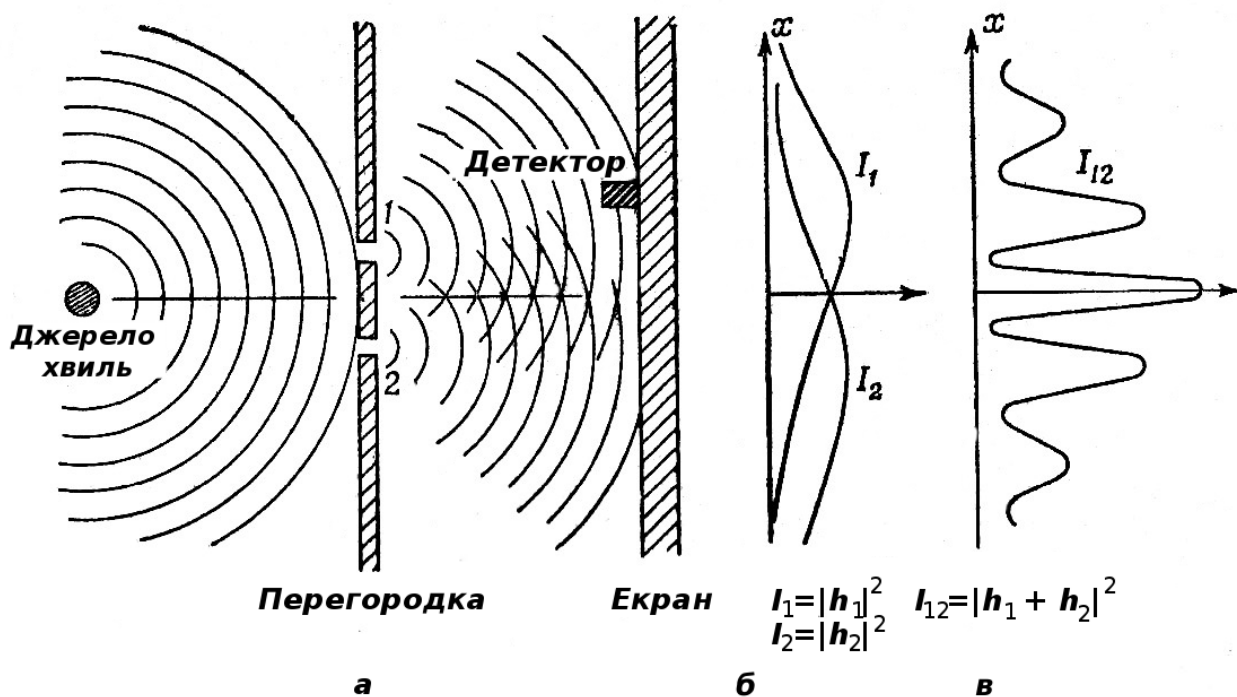


Рис. 3.8. Ілюстрація дослід з хвилями на воді: а - схема установки; б - інтенсивність від кожної щілини; в - інтерференція хвиль від двох щілин

Устаткування являє собою мілку посудину, заповнену водою. Предмет, позначений як "Джерело хвиль", коливаючись за допомогою маленького моторчика вгору-вниз, викликає кругові хвилі. Перегородка затримує хвилі, однак має два отвори, які, згідно з принципом Френеля, є джерелами вторинних хвиль, які досягають екрану та взаємодіють там, викликаючи картину, зображену на рисунку 3.8б та 3.8в. Перед екраном,

як і в першому досліді, розміщено рухомий детектор. Якщо його “шкалу” відкалібрувати пропорційно квадрату висоти хвилі h , то відлік буде показувати інтенсивність хвилі. Таким чином, детектор буде вимірювати енергію, що її переносить хвиля, або, що точніше, ту частину енергії, яку передано саме детектору.

Перше, що можна стверджувати за допомогою такого хвильового апарата, це те, що інтенсивність хвиль може бути довільною. Коли джерело рухається поволі, і детектор демонструє ледве помітний рух. Якщо інтенсивність руху джерела зростає, детектор показує більшу інтенсивність. Оскільки інтенсивність може бути будь-якою, ми вже не можемо стверджувати, що вона має хоча б якусь “порціонність”, квантується.

Давайте тепер змусимо джерело хвиль працювати стабільно та будемо вимірювати інтенсивність хвиль для різних значень координати x . Ми отримаємо таку криву, яку подано на рисунку 3.8в.

Як бачимо, хвилі взаємодіють між собою і, як це відомо зі шкільного курсу фізики, утворюють так звану інтерференційну картину. Саме таку картину у вигляді райдужних кілець ми бачимо на поверхні калюж, куди розлито трохи бензину або олії. Зрозуміло, що первинна хвиля від джерела дифрагує на отворах 1 та 2, в результаті чого ми маємо дві когерентні вторинні хвилі, які й утворюють інтерференційну картину. Це типовий результат для хвильових процесів, чи то електромагнітних, чи механічних хвиль.

Тепер, якщо ми будемо закривати отвори по черзі, то отримаємо криві інтенсивності I_1 та I_2 , зображені на рисунку 3.8б. Очевидно, що сумарна інтенсивність, I_{12} не дорівнює сумі інтенсивностей від кожної щілини, $I_1 + I_2$. В цьому й полягає суть явища інтерференції. На деяких ділянках (де спостерігаються максимуми графіка I_{12}) хвиля від одного отвору підсилює хвилю від другого. Утворюється так звана конструктивна інтерференція, забезпечуючи більшу інтенсивність. Це буває тоді, коли окремі хвилі приходять до деякої точки “у фазі”, тобто відстань від цієї точки до однієї з щілин більше (або менше) відстані до другої на ціле число довжин хвилі. Якщо ж різниця відстаней певної точки екрану до кожної зі щілин кратна не цілому числу довжин хвилі, виникає “деструктивна інтерференція”, тобто загальна інтенсивність зменшується. У цих точках можна побачити мінімуми кривої I_{12} .

Якщо згадати шкільну фізику, ми зможемо записати, що наш хвильовий процес можна описати такою формулою:

$$H_1 = h_1 e^{i\omega t}; H_2 = h_2 e^{i\omega t},$$

де $H_{1,2}$ — миттєва висота хвилі, t — час, ω — частота коливань.

Якщо відкриті обидві щілини, тоді будемо мати:

$$H_{1,2} = (h_1 + h_2) e^{i\omega t},$$

що символізує той факт, що загальна висота хвилі є сумою висот хвиль від окремих щілин.

Відомо, що інтенсивність пропорційна квадрату висоти хвилі, тобто:

$$I_1 = |h_1|^2; I_2 = |h_2|^2; I_{12} = |h_1 + h_2|^2.$$

Як бачимо, нічого спільного з тим, що було з кулями, не відбувається.

Якщо ми розкриємо модуль $|h_1 + h_2|^2$ отримаємо:

$$I_{12} = |h_1|^2 + |h_2|^2 + 2|h_1||h_2|\cos(\delta),$$

де δ — різниця фаз між h_1 та h_2 . Уводячи інтенсивності, можемо отримати:

$$I_{12} = I_1 + I_2 + 2\sqrt{I_1 I_2} \cos(\delta).$$

Як бачимо, останній вираз суттєво відрізняється від випадку кулемета. Останній доданок цієї формули — так званий “інтерференційний доданок”, якого немає у випадку класичних об’єктів.

На цьому ми закінчимо приклад з хвилями, зробивши такі висновки:

- інтенсивність хвиль може бути довільною — “квантування” у цьому випадку не існує;
- закриваючи отвори по чергово, ми отримали інтенсивності хвиль I_1 та I_2 ;
- якщо обидва отвори відкрито, маємо загальну інтенсивність $I_{12} \neq I_1 + I_2$.
- спостерігається явище інтерференції хвиль.

Такі висновки характерні для хвильових процесів.

Дослід з електронами

І, нарешті, розглянемо ще один дослід, схему якого подано на рисунку 3.9.

Цей уявний експеримент виконується з електронами, які генерує електронна гармата. Найпростіша за конструкцією електронна гармата складається з вольфрамового дроту, розміщеного у металічній коробочці з отвором. Дріт нагрівається електричним струмом.

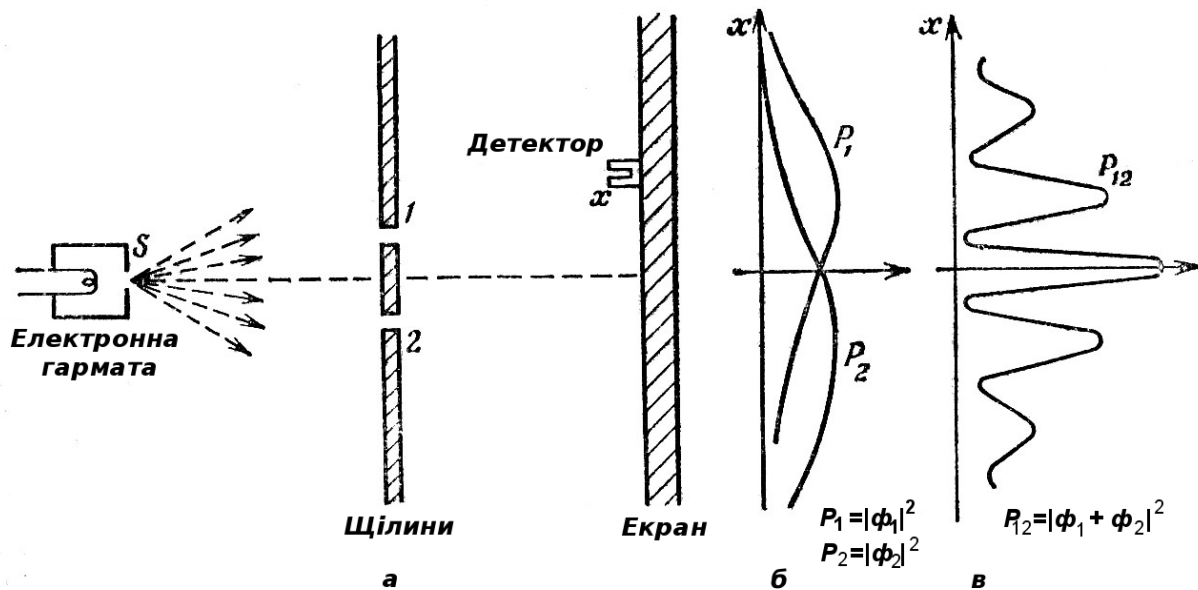


Рис. 3.9. Ілюстрація дослід з електронами: а - схема "установки"; б - інтенсивність від кожної щілини; в - інтенсивність від обох щілин

Якщо на нього подати негативний потенціал, а на коробочку — додатній, то електрони, що їх випромінює дріт, будуть розганятися різницею потенціалів, і деякі з них вискочать назовні крізь отвір. Такі електрони будуть мати приблизно однакову енергію. Перед гарматою розміщено металічний екран з двома щілинами, а за ним — ще один екран, який поглинає електрони. Перед поглинаючим екраном розташовано рухомий детектор електронів, скажімо лічильник Гейгера.

Увімкнувши наше обладнання, ми почуємо клацання детектора. Усі сигнали однакові за інтенсивністю. Не буває слабших або сильніших сигналів. Сигнали можуть бути частішими або рідшими, але завжди однаковими за інтенсивністю (гучністю). Коли процес генерування електронів стабілізується, ми зможемо побачити, що кількість

zareєстрованих лічильником Гейгера електронів за однакові проміжки часу буд майже однаковою. Тому можна говорити про середню частоту клацань детектора, тобто певну кількість клацань, скажімо, на хвилину в середньому.

Якщо ми пересуваємо детектор по вісі x , частота клацань змінюється, але величина (гучність) кожного клацання лишається постійною. Якщо би ми розмістили перед поглинаючим екраном другий детектор (в іншому місці, звичайно), ми би почули, що клацає то один детектор, то другий, і ніколи обидва разом (хіба що іноді наше вухо не в змозі розрізнити два клацання підряд). Тому ми можемо зробити висновок про те, що електрони, як кулі в першому досліді, приходять “порціями”. Усі “порції” однакові за величиною, в детектор або в поглинаючий екран влучає лише одна “порція”. Тоді ми говоримо: “Електрони завжди приходять однаковими порціями”.

Як і в досліді з кулеметом, спробуємо знайти відповідь на питання: “Яка відносна ймовірність того, що електрон влучить у поглинаючий екран на різних віддальх від середини?”. Ми можемо отримати відповідь на це питання, підрахувавши частоту клацань лічильника під час стабільної роботи електронної гармати.

В результаті нашого уявного експерименту отримана дуже цікава крива, яка подана на рисунку 3.9в. Слід відзначити, що саме так і поведуть себе електрони в умовах реального експерименту.

Спробуємо пояснити таку дивну поведінку електронів. Оскільки ці електрони, як ми бачили, приходять “порціями”, тобто квантуються, то кожен з них проходить або крізь отвір 1, або крізь отвір 2. Тоді усі електрони, які влучили у поглинаючий екран, можна розділити на два класи: ті, що пройшли крізь перший отвір, і ті, що пройшли через другий. Це значить, що крива 3.9в — це сума ефектів від електронів, які пройшли крізь різні отвори. Тепер закриємо отвір 2 і будемо вивчати тільки електрони, що пройшли крізь отвір 1. Результат таких вимірів подано на рисунку 3.9б, крива P_1 . Там же подано результати для отвору 2, - крива P_2 . Вони виглядають звичним чином.

Крива P_{12} , яку отримано, коли обидва отвори відкрито, зовсім не узгоджується з сумою $P_1 + P_2$. Очевидно, що вона аналогічна до такої в досліді з хвилями. Отже, ми чомусь бачимо інтерференційну картину. Звідки вона взялася? Що з чим тут взаємодіє, щоби дати таку картину? Може, взаємодіють електрони, що пройшли крізь різні отвори? Але як це можливо, якщо вони “прибувають” до екрану дискретно і у різний час?

Може вони рухаються складніше, проходячи спочатку через один отвір, а потім — через другий? Але тоді, закривши один отвір, ми маємо зменшити кількість електронів, що досягли екрану. Тим не менше, ми бачимо на кривій 3.9б точки, де кількість електронів більша за таку, коли відкрито обидва отвори. Виходить, що закриття одного отвору збільшує кількість електронів, що проходить крізь другий. І навпаки, середній пік кривої P_{12} більш, ніж удвічі вища за суму значень кривих P_1 та P_2 . Очевидно, що цих два факти протирічать один одному! Пояснити обидва факти з точки зору складних траєкторій рух електронів неможливо. В усякому разі досі нікому це не вдалося зробити.

При всьому цьому, як не дивно, математика, що описує зв'язок P_{12} , P_1 та P_2 зовсім проста. Вона дуже схожа на ту, яка описувала дослід з хвилями. Такий результат не вдається отримати, якщо вважати, що електрони рухаються по складних траєкторіях.

Отже, у нас немає іншого виходу, як визнати: електрони перебувають “порціями”, тобто виявляють при цьому властивості частинок (корпускул), а ймовірність їх влучання в екран розподіляється так само, як інтенсивність хвиль. Саме в такому розумінні електрон поводить себе частково як частинка, частково як хвиля.

А як тепер бути з твердженням, що крива P_{12} , є сумарний ефект від електронів, які пройшли крізь отвір 1 і крізь отвір 2? Ми бачимо, що $P_{12} \neq P_1 + P_2$. Очевидно, твердження про те, що електрони проходять або крізь отвір 1, або крізь отвір 2, неправильне.

Так як же вони проходять?

Спробуємо модифікувати наш уявний експеримент. Відомо, що електрони добре взаємодіють зі світлом. Схема такого експерименту подана на рисунку 3.10. Якщо ми візьмемо деяке джерело світла і розмістимо його на шляху електронів, то за спалахами світла, наприклад, біля отвору 1 (чи 2) ми будемо знати, через який з них пройшов черговий електрон. А якщо ми побачимо спалах світла біля обох отворів, це буде свідчити, що ми невірно уявляємо собі реальність мікросвіту.

Запустивши устаткування, прослідкуємо, що відбувається з електронами. Щоразу, коли ми чуємо клацання детектора, ми бачимо спалах або біля першого, або біля другого отвору, але ніколи біля обох одразу. Так відбувається при будь-якому положенні детектора. Звідси можна зробити висновок про те, що коли ми спостерігаємо за електроном, він з необхідністю проходить лише через один отвір: або через перший, або через другий. Продовживши спостереження, побачимо, що цього разу

ми отримаємо ймовірність від двох щілин точно таку ж, як і в досліді з “кулеметом”, тобто $P'_{12} = P'_1 + P'_2$. Чому ж тоді в попередньому експерименті ми отримали $P_{12} \neq P_1 + P_2$? В чому різниця між експериментами? Єдина різниця полягає в тому, що в модифікованому експерименті ми спостерігали за електронами. Ну, що ж, цей результат можна зрозуміти: електрони настільки тендітна річ, що ми своїм джерелом світла збудуємо електрони, змінюючи їхню поведінку.

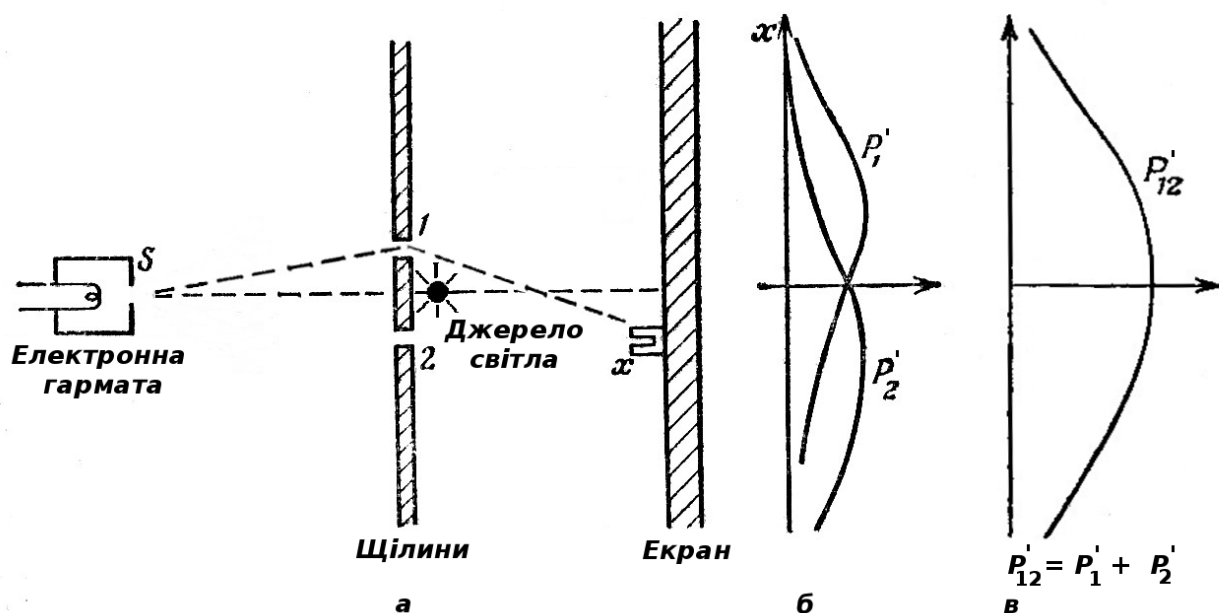


Рис. 6. Ілюстрація модифікованого досліді з електронами

Перевіримо це. Будемо зменшувати яскравість джерела світла, і ми не будемо надавати електронам такого збурення, щоби впливати на їхню поведінку. Перше, що ми відзначимо, це те, що яскравість спалаху при взаємодії світла з електроном не зменшується. Це й зрозуміло, оскільки світло — це потік фотонів. При зменшенні яскравості світла параметри самих фотонів не змінюється, а лише зменшується їх кількість у потоці. Зменшуючи яскравість, ми, нарешті, прийдемо до такої ситуації, що фотонів для взаємодії з електронами не буде ставати, і ми будемо пропускати електрони, тобто, лічильник клацнув, зафіксувавши електрон, але ми не побачили спалаху, бо на тому місці, де перебував електрон, в той час не було фотона. Таким чином, у нас буде вже три типи електронів: ті, що пройшли крізь перший отвір, ті, що крізь другий, і ті, яких ми не побачили. І саме тоді явище інтерференції знову з'являється!

Ну, що ж, це вже можна пояснити. Коли ми не побачили електрон, фотон не збував його; а якщо ми його помітили, — то фотон провзаємодіяв з електроном, змінивши його енергію якимось чином. Оскільки фотони усі однакові, значить ступінь збурення електрона завжди одна й та ж, достатня для того, щоби зруйнувати явище інтерференції.

Якщо ми міркуємо правильно, то існує ще один можливий спосіб “неруйнуючого” спостереження за електронами. Якщо взаємодія фотона з електроном, як це описується у фізиці, супроводжується передаванням електрону імпульсу фотона ($p = h / \lambda$), очевидно, що чим більша довжина хвилі, тим менший імпульс ми йому передамо. Для цього треба подіяти світлом червоного (може, навіть, інфрачервоного) діапазону.

Якщо ми будемо модифікувати дослід у такий спосіб, побачимо наступну картину. При збільшенні довжини хвилі спочатку нічого не зміниться. Однак, потім трапиться дуже неприємна річ: як тільки довжина світла стане порядку віддалі між щілинами, спалахи стануть такими розмитими, що ми не зможемо визначити, біля якої з них він відбувся. Бачимо, що електрон проскочив, а крізь яку щілину — не зрозуміло. І знову, саме в цей момент, коли почнуть з’являтися “невраховані” електрони, крива P_{12} продемонструє появу інтерференції! При подальшому збільшенні довжини хвилі інтерференція стане все помітнішою, і нарешті, коли довжина хвилі буде набагато більшою за відстань між щілинами, і вже зовсім незрозуміло де відбувся спалах, картина інтерференції стане подібною на таку, подану на рисунку 3.10в.

Як з’ясувалося в роки становлення квантової механіки, така поведінка мікрооб’єктів була помічена в багатьох експериментах. Вернер Гейзенберг припустив, що закони мікросвіту та макросвіту можна було би узгодити, якщо вважати, що існують деякі фундаментальні обмеження наших експериментальних можливостей, яких раніше не помічали. Він запропонував як загальний принцип так званий принцип невизначеності Гейзенберга. Цей принцип звучить так: *“Принципово неможливо одночасно виміряти з довільною точністю координати й імпульси квантового об’єкта”*. У термінах нашого експерименту він звучить трохи інакше: *“Неможливо створити устаткування для визначення того, через яку щілину пройшов електрон, не збурюючи його настільки, що інтерференційна картина зникає”*. Якщо устаткування здатне визначити, крізь яку щілину пройшов електрон, воно не здатне бути настільки делікатним, щоби не спотворити картину інтерференції. Жодній людині ніколи не вдавалося винайти устаткування, чи просто вказати спосіб, як

обійти принцип невизначеності. Тому ми зобов'язані припустити, що він відображає одну з фундаментальних властивостей природи.

Принцип невизначеності Гейзенберга надає нам єдиний спосіб, за допомогою якого ми повинні розмірковувати, щоби не зробити помилкових передбачень. Цей спосіб полягає в тому, що якщо у нас є прилад, здатний визначити, крізь яку щілину пройшов електрон, ми можемо говорити про те, пройшов він крізь першу (або другу). Але якщо ви не намагалися дізнатися де пройшов електрон, то ви не можете навіть думати про те, чи електрон пройшов крізь першу, чи крізь другу щілину. Якщо ж почати розмірковувати на цю тему, а потім робити з цього якісь припущення, можна прийти до протиріч та помилкових фізичних висновків. Отже ми повинні балансувати на цьому логічному лезі, щоби успішно описувати явища мікросвіту.

Якщо рух усієї речовини подібний до руху електронів в нашому уявному експерименті, і його можна описувати за допомогою хвильових понять, то як бути з кулями в нашому першому експерименті? Чому там ми не побачили інтерференційної картини? Спробуємо це пояснити.

Хвиля де Бройля визначається з виразу: $\lambda = h/mv = 6.62 \times 10^{-34} / mv$. Припустимо, що швидкість кулі та електрона однакова — 1000 м/с. Тоді отримаємо довжину хвилі де Бройля:

- для електрона: $\lambda = 6.62 \times 10^{-34} / 9.1 \times 10^{-31} \times 10^3 \approx 0.7 \text{ м}$;
- для «кулі»: $\lambda = 6.62 \times 10^{-34} / 10^{-2} \times 10^3 \approx 7 \times 10^{-35} \text{ м}$.

Чи існує такий прилад, який може виміряти сигнал з довжиною хвилі 10^{-34} м ? З такою довжиною хвилі інтерференційні піки будуть настільки тонкими, що жоден детектор їх просто не зможе розрізнити, а буде просто показувати середнє значення або огинаючи такого коливного руху.

Які ж висновки можна зробити з наших уявних експериментів?

1. Електрони, як й інші квантові об'єкти, поведуть себе частково як частинки (вони прибувають до екрану дискретними порціями), частково — як хвилі (дають на екрані інтерференційну картину).

2. Надзвичайно важливим є висновок про те, що вимірювання в мікросвіті впливають на стан квантової системи. Цей висновок накладає просто екстремальні обмеження на уся квантову інформатику, адже читання чи запис даних еквівалентні процесу вимірювання. Отже необхідно буде розробляти алгоритми, які працюють без проміжного читання/запису.

3. Цей висновок, до деякої міри, впливає з попереднього. Якщо неможливо чітко визначити траєкторію руху електрона без впливу на результат, то ми повинні говорити лише про ймовірність тієї чи іншої траєкторії. Це справедливо не тільки для руху електронів, але й для усіх квантових систем: ми можемо лише говорити про ймовірність того чи іншого значення в квантовій системі. Якщо ми захочемо дізнатися точно, яке значення міститься у квантовому ОЗП, ми повинні провести вимірювання, а значить, вплинемо на саме це значення. Чи отримаємо ми в результаті таких дій істинне значення змінної — велике питання. Воно буде виникати перед нами усюди в квантовій інформатиці.

Незвичність подій, які виникають при виконанні квантових обчислень, і не мають аналогів у класичних обчисленнях, ставить перед математиками досить незвичні завдання, які необхідно вирішити в процесі розробки квантових алгоритмів. Тим не менше, ці завдання успішно вирішуються, що говорить про те, що основні засади квантової інформатики розроблено, а на часі — технічна реалізація універсальних квантових комп'ютерів.

3.3. Квантові біти

Опис простої квантової системи

Як же ми можемо математично описати таку систему, про яку ми говорили вище? Для коректного описання придумано спеціальний формалізм.

Позначимо стан квантової системи, коли частинка пройшла крізь верхній отвір як $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. Кутові дужки вказують на те, що це не просто “0”, а стан квантової системи. Як бачимо, тут існує деякий зв'язок з реальністю, оскільки одиниця стоїть там, де пройшла частинка, а нуль — там, де її не було. Аналогічно можна записати для випадку, коли частинка пройшла крізь нижній отвір: $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Для цього випадку міркування абсолютно аналогічні. Описані стани називаються базисними, оскільки вони описують ситуацію, коли частинка точно пройшла крізь один отвір, і не проходила крізь другий. Однак, ми з'ясували, що існують такі ситуації, коли ми не можемо напевне сказати, крізь який отвір пройшла частинка. Ми знаємо

лише, що з деякою ймовірністю вона може проскочити крізь отвір 1, і з деякою — крізь отвір 2.

Такий стан квантової системи ми можемо описати суперпозицією станів $|0\rangle$ та $|1\rangle$.

$$|Q\rangle = a|0\rangle + b|1\rangle. \quad (3.1)$$

Коефіцієнти a та b описують ймовірності проходження частинки крізь відповідні отвори. Ці коефіцієнти повинні задовольняти деякі умови нормування. Вони описують той факт, що частинка точно пройшла або крізь перший, або крізь другий отвір, тобто сума ймовірностей проходження крізь кожен отвір повинна дорівнювати одиниці: $|a|^2 + |b|^2 = 1$.

Формула (3.1) є відображенням так званого *принципу суперпозиції*, одного з основних і найважливіших принципів квантової механіки. Цей принцип формулюється так:

якщо квантова система може перебувати в станах $|0\rangle$ та $|1\rangle$, то вона може перебувати й у будь-якій суперпозиції цих станів.

Розглянемо простий приклад опису квантової системи. Нехай система перебуває в стані, що описується виразом:

$$|Q\rangle = \frac{4}{5}|0\rangle + \frac{3}{5}|1\rangle.$$

Якою буде ймовірність того, що при вимірюванні ми отримаємо нуль?

$$P(0) = \left| \frac{4}{5} \right|^2 = \frac{16}{25} = 0.64.$$

Ймовірність отримати при вимірюванні одиницю:

$$P(1) = \left| \frac{3}{5} \right|^2 = \frac{9}{25} = 0.36.$$

Це означає, що в 64% випадків вимірювань ми отримаємо “0”, а в 36% - “1”.

Які ж реальні фізичні системи, крім проходження частинки крізь щілини, можуть бути такою квантовою системою?

Однією з найпростіших та найбільш популярних вважається спін мікрочастинки, наприклад, електрона.

Коротко та дуже умовно спін частинки можна описати таким чином. Якщо уявити собі заряджену частинку як маленьку кульку, то її обертання, як обертання будь-якого зарядженого тіла, створює магнітний момент, як це показано на рисунку 3.11. Цей магнітний момент і називається спіном (від англ. *to spin* - обертатись).

І хоча насправді це зовсім не так, але така аналогія прижилася, і ми будемо нею користуватися.

В залежності від напрямку обертання частинки напрям магнітного моменту може бути різним.

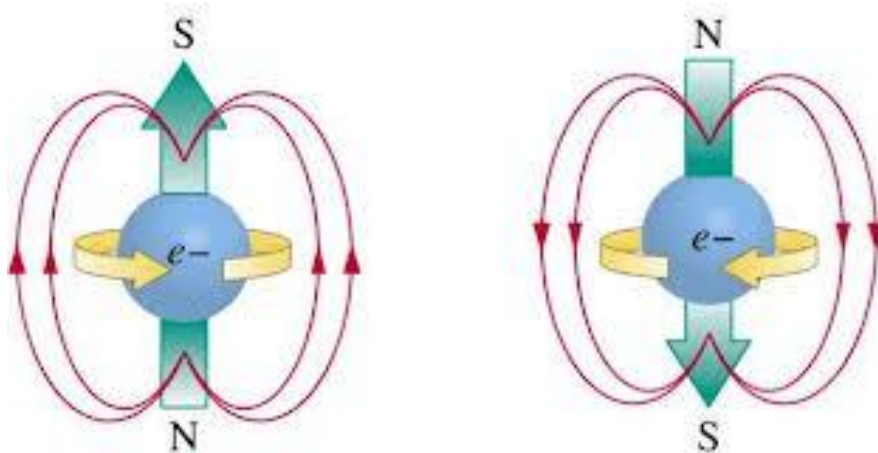


Рис. 3.11. Ілюстрація до поясненні спіну електрона

Якщо спін частинки напрямлений вгору (як це показано на рис. 3.11 ліворуч), будемо позначати такий стан як $|0\rangle$, а стан, коли спін напрямлено вниз, — через $|1\rangle$. Тоді знову стан, коли ми не знаємо куди напрямлений спін частинки, тобто стан суперпозиції, можна записати як

$$|Q\rangle = a|0\rangle + b|1\rangle.$$

І знов коефіцієнти при базисних станах вказують на ймовірність отримати при вимірюваннях нуль або одиницю. Звичайно, вони повинні задовольняти ту саму умову нормування:

$$|a|^2 + |b|^2 = 1.$$

Як буде реагувати така система на вимірювання? Припустимо, що ми при першому вимірюванні отримали нуль. В якому стані тоді перебуватиме система?

$$|Q\rangle = 1 \cdot |0\rangle + 0 \cdot |1\rangle = |0\rangle.$$

Як бачимо, система перейшла в стан $|0\rangle$ і при наступних вимірюваннях із 100% ймовірністю ми її знайдемо в такому стані. Що це означає? Ми бачимо, що до вимірювання система знаходилася в стані суперпозиції, а після вимірювання перейшла у строго визначений стан $|0\rangle$. Таким чином, ми можемо припустити (і це підтверджується численними експериментами), що *вимірювання впливають на стан квантової системи*. Більше того, скільки разів ми би не повторювали вимірювання, квантова система більше не змінить свій стан, і ми отримуємо *звичайну класичну систему з чітко визначеним станом*.

Оперативний запам'ятовувальний пристрій сучасного комп'ютера складається з комірок, двом станам яких (більший рівень напруги - менший рівень напруги, є електричний заряд - немає заряду, є електричний струм — немає струму тощо) привласнюються значення “1” та “0”. У наборі комірок (регістрі) записується та обробляється інформація у вигляді двійкових чисел. Один біт має два стани: “1” та “0”; регістр з N бітів має 2^N станів, тобто в такому регістрі можна закодувати двійкове число з N знаків.

Розглянемо систему з трьох бітів. В такому класичному регістрі ми можемо зберігати одне з чисел: 000; 001; 010; 011; 100; 101; 110; 111. Якщо ми хочемо зберігати усі ці числа *одночасно*, то для цього буде потрібно вісім трибітових регістрів, тобто 24 біти.

У квантовому комп'ютері елементарними комірками для запису та обробки інформації є *квантові біти* або *кубіти*. Кубіт — це квантова система, яка має два базисних стани, $|0\rangle$ та $|1\rangle$. Однак, на відміну від класичного біта, кубіт може знаходитися не тільки в одному визначеному стані, а й в суперпозиційному стані: $|Q\rangle = a|0\rangle + b|1\rangle$. Тут, з логічної точки зору, класична булева логіка, “1/0” – *True/False* не має місця. Суперпозиційний стан кубіта – частково “*true*” (з ймовірністю $|b|^2$), частково “*false*” $|a|^2$. Найважливішою відмінністю кубіта від класичного біта полягає в тому, що в системі кубітів можливий так званий “переплутаний” стан або стан суперпозиції. В такому стані кожен кубіт одночасно містить і “0” і “1”, а отже, квантова система з N кубітів одночасно містить усі можливі двійкові числа від 0 до 2^N , тобто 2^N бітів.

Дійсно, якщо один кубіт у стані суперпозиції містить одночасно і “1”, і “0”, тобто $2^1 = 2$ значення; два кубіти будуть містити суперпозицію $(a|0\rangle + b|1\rangle)(c|0\rangle + d|1\rangle) = (ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle)$, тобто $2^2 = 4$.

Відповідно, N кубітів у суперпозиційному стані будуть містити 2^N бітів.

Очевидно, що це приводить до значної економії використаної пам'яті комп'ютера. Більше того, деякі математичні операції над таким регістром можна виконувати усього один раз, оскільки вони виконуються над усіма значеннями одночасно. Такий спосіб обробки даних називають “квантовим прискоренням обчислень”. Його та алгоритми, які використовують таке прискорення ми розглянемо пізніше.

А зараз з'ясуємо, які найпростіші операції ми можемо виконувати над кубітами.

3.4. Квантові операції

Прості операції над кубітами

Операції над кубітами часто називають *гейтами* або *вентиллями*. Для визначеності будемо вважати, що ми маємо справу зі спіном електрона, хоча яка саме фізична система розглядається неважливо. Важливим є те, що вона виявляє квантові властивості.

Отже, нехай ми маємо систему, базисні стани якої складаються з $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ та $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Найпростішими гейтами, які працюють з одним кубітом будемо вважати: *тотожну операцію (I)*, *заперечення (NOT)*, *гейт Адамара (H)*.

Тотожна операція (I)

За своїм змістом дія тотожної операції не змінює операнд. Це означає, що $I|0\rangle = |0\rangle$ та $I|1\rangle = |1\rangle$. Визначимо тотожну операцію так:

$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. Перевіримо її дію:

$$I|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \cdot 1 + 0 \cdot 0 \\ 0 \cdot 1 + 1 \cdot 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle,$$

$$I|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \cdot 0 + 0 \cdot 1 \\ 0 \cdot 0 + 1 \cdot 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle.$$

Як бачимо, дія такої матриці на стани $|0\rangle$ та $|1\rangle$ дійсно не призводить до зміни операндів. Це значить, що тотожна операція визначена нами вірно.

Заперечення (NOT)

Стандартна унарна операція заперечення (NOT) повинна перетворювати стани $|0\rangle$ в $|1\rangle$ та навпаки.

Визначимо операцію заперечення таким чином: $\text{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.

Перевірка дасть такі результати:

$$\text{NOT}|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \cdot 1 + 1 \cdot 0 \\ 1 \cdot 1 + 0 \cdot 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle;$$

$$\text{NOT}|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \cdot 0 + 1 \cdot 1 \\ 1 \cdot 0 + 0 \cdot 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle.$$

Бачимо, що уведений оператор діє саме так, як нам потрібно.

Гейт Адамара (Hadamard, H)

Перетворення Адамара є одним з найважливіших ґейтів квантових обчислень. Чому це так, зараз ми зрозуміємо.

Уведемо це перетворення наступним чином: $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.

Застосуємо його до нашої квантової системи. Будемо мати:

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \cdot 1 + 1 \cdot 0 \\ 1 \cdot 1 - 1 \cdot 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle);$$

$$H|1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \cdot 0 + 1 \cdot 1 \\ 1 \cdot 0 - 1 \cdot 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle).$$

Що ж ми отримали в результаті? Як бачимо, ґейт Адамара, діючи на строго визначений стан переводить його у стан суперпозиції. Ця дія надзвичайно важлива і використовується практично в усіх квантових алгоритмах, наприклад, для ініціалізації квантових регістрів, де вимагається переплутаний стан.

Зрозуміло також, що подвійне застосування ґейта Адамара є тотожною операцією. Доведемо це:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle);$$

$$H\left[\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right] = \frac{1}{2}[|0\rangle + |1\rangle + |0\rangle - |1\rangle] = |0\rangle$$

Аналогічне доведення і для стану $|1\rangle$.

Ґейт Адамара має просту фізичну інтерпретацію (рис. 3.12). Це може бути простий оптичний дільник, який ділить падаючий промінь у співвідношенні 50:50, тобто напівпрозорий фільтр.

Той факт, що промінь світла знаходиться над пластинкою, позначимо як стан системи $|0\rangle$ незалежно від того, чи він тільки падає на пластинку, чи вже відбивається від неї.

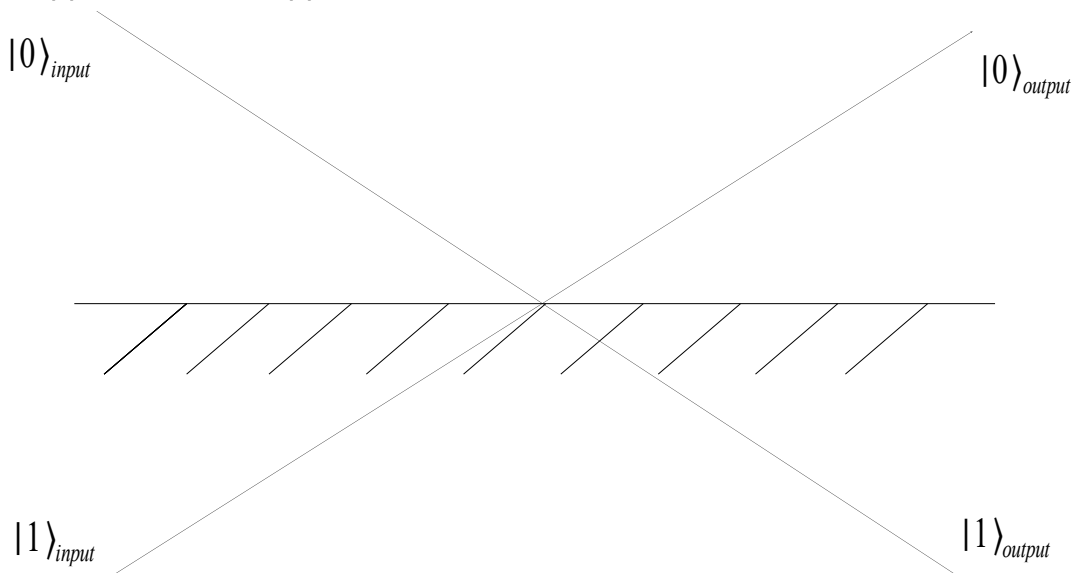
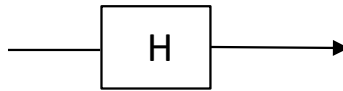


Рис. 3.12. Фізична інтерпретація ґейта Адамара

Аналогічно, якщо промінь знаходиться під пластинкою, позначимо через $|1\rangle$. Оскільки пластинка напівпрозора, то ймовірність відбиття падаючого променя та ймовірність того, що він пройде крізь пластинку, рівні. Отже, направляючи промінь на пластинку, ми не знаємо точно де знайдемо оброблений промінь: вище, чи нижче пластинки. Якщо вважати промінь квантовою системою, то вхідний стан системи буде чітко визначеним, а вихідний – переплутаним. Так само себе поводить ґейт Адамара.

На квантових схемах ґейт Адамара, як би він не був реалізований, позначається так:



Багатокубітні операції

Розглянуті ґейти працюють з одним кубітом. Існує багато ґейтів, які працюють з кількома кубітами. Ми розглянемо лиш два з них: контрольоване заперечення та ґейт Тоффолі.

Контрольоване заперечення (Controlled NOT, CNOT)

ґейт CNOT використовує два кубіти, один з яких контрольний, а другий – контрольований. Логіка роботи цього ґейта така: якщо контрольний кубіт дорівнює одиниці, то контрольований кубіт інвертується:

$$|00\rangle \rightarrow |00\rangle$$

$$|01\rangle \rightarrow |01\rangle$$

$$|10\rangle \rightarrow |11\rangle$$

$$|11\rangle \rightarrow |10\rangle.$$

Тут контрольним вважається перший кубіт, а контрольованим – другий. Дуже схоже на таблиці істинності логічних операцій. Цим ми пізніше скористаємося.

ґейт Тоффолі (T)

ґейт Тоффолі або подвійний CNOT (CCNOT), використовує три кубіта: два контрольних та один контрольований. Працює він так: якщо обидва контрольних кубіти дорівнюють одиниці, контрольований кубіт інвертується. Дію ґейта Тоффолі можна зобразити таким чином:

$$|000\rangle \rightarrow |000\rangle$$

$$|001\rangle \rightarrow |001\rangle$$

$$|010\rangle \rightarrow |010\rangle$$

$$|011\rangle \rightarrow |011\rangle$$

$$|100\rangle \rightarrow |100\rangle$$

$$|101\rangle \rightarrow |101\rangle$$

$$|110\rangle \rightarrow |111\rangle$$

$$|111\rangle \rightarrow |110\rangle.$$

Тут перші два біти вважаються контрольними, а останній – контрольованим.

В принципі, маючи такий набір ґейтів, можна спроектувати довільний пристрій для обробки інформації. Розглянемо, наприклад, суматор за модулем два на основі контрольованого заперечення.

Елемент складається з двох кубітів: верхній – контролюючий, нижній – контрольований. Якщо верхній кубіт несе одиницю, то нижній кубіт інвертується. Розглянемо відповідність такого елемента додаванню за модулем два (таблиця 3.2).

Таблиця 3.2

Суматор за модулем два на основі контрольованого заперечення

Значення a	Значення b	Результат CNOT	Результат XOR
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

Як видно з таблиці, результат дії CNOT повністю відповідає додаванню за модулем два. Це яскравий приклад того, як за допомогою квантових ґейтів можна побудувати оператори булевої алгебри, а відповідно, й інші пристрої обчислювальної техніки.

Часткові вимірювання

Раніше ми обговорювали той факт, що вимірювання впливають на стан кубітів, перетворюючи його з суперпозиційного у класичний, цілком визначений стан. При подальших вимірах цей стан вже не змінюється, залишаючись весь час однаковим.

Тим не менше, ми можемо виконувати так звані *часткові вимірювання*, вимірюючи окремі біти квантової системи.

Розібратися в тому, як це відбувається і до чого приводить, можна на простому прикладі.

Розглянемо систему з двох квантових бітів у стані:

$$|Q\rangle = \frac{3}{5}|00\rangle + \frac{2}{5}|01\rangle + \frac{2}{5}|10\rangle + \frac{2\sqrt{2}}{5}|11\rangle.$$

Спробуємо виміряти другий кубіт. З якою ймовірністю ми отримаємо одиницю?

Другий кубіт не дорівнює нулю у другому та четвертому доданках. Тоді ймовірність того, що ми при вимірюванні отримаємо одиницю можна записати так:

$$P(1) = \left(\frac{2}{5}\right)^2 + \left(\frac{2\sqrt{2}}{5}\right)^2 = \frac{4}{25} + \frac{8}{25} = \frac{12}{25}.$$

В якому стані опиниться система, якщо ми отримаємо одиницю в другому біті? Зрозуміло, що після вимірювань другий біт буде завжди дорівнювати одиниці, отже перший та третій доданки зникнуть, оскільки там другий біт дорівнює нулю. Але просто так забрати ці доданки не можна, оскільки необхідно задовольнити умову нормування. Отже треба обчислити нові коефіцієнти біля доданків з таких умов:

$$\begin{aligned} |a|^2 + |b|^2 &= 1; \\ \frac{a}{b} &= \frac{2/5}{2\sqrt{2}/5} = \frac{\sqrt{2}}{2}. \end{aligned}$$

Отже для обчислення нових коефіцієнтів необхідно розв'язати систему:

$$\begin{cases} \frac{a}{b} = \frac{\sqrt{2}}{2} \\ |a|^2 + |b|^2 = 1. \end{cases}$$

Обчисливши коефіцієнти, отримуємо такий стан системи:

$$|Q\rangle = \frac{1}{\sqrt{3}}|01\rangle + \sqrt{\frac{2}{3}}|11\rangle.$$

Як бачимо, часткові вимірювання хоч і не руйнують систему, однак також змінюють її квантовий стан.

Переплутані стани

Розглянемо систему з двох кубітів. Простір станів такої системи буде описуватися виразом:

$$\{|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle\};$$

або

$$\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$$

де $\otimes$ - тензорний добуток, який визначається наступним чином:

$$\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac & ad \\ bc & bd \end{pmatrix}.$$

Як бачимо, тензорний добуток збільшує розмірність вектора до матриці. Тензорний добуток підкоряється стандартним для математики властивостям лінійності, асоціативності та комутативності, що дозволяє в деяких випадках працювати з ним аналогічно до звичайного добутку.

У загальному випадку, як це було показано раніше, система з n кубітів має 2^n базисних станів, тобто станів, тензорний добуток яких дає повний простір квантових станів системи.

Однак, не завжди усі стани квантової системи можна зобразити станами окремих кубітів.

Наприклад, стан $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ не можна зобразити через стани окремих кубітів. Тобто не існує таких коефіцієнтів a_1, a_2, b_1, b_2 , щоби:

$$(a_1|0\rangle + b_1|1\rangle)(a_2|0\rangle + b_2|1\rangle) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \quad (3.2)$$

Дійсно:

$$(a_1|0\rangle + b_1|1\rangle)(a_2|0\rangle + b_2|1\rangle) = a_1a_2|00\rangle + a_1b_2|01\rangle + b_1a_2|10\rangle + b_1b_2|11\rangle.$$

Для того, щоби виконувалася рівність (3.2), необхідно, щоби $a_1b_2=0$ та $b_1a_2=0$. Це означає, що $a_1a_2=0$ та $b_1b_2=0$. Отже стан $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ не може бути суперпозицією станів $(a_1|0\rangle + b_1|1\rangle)$ та $(a_2|0\rangle + b_2|1\rangle)$.

Таким чином, стан $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ є так званим *переплутаним* станом по відношенню до базису $(a_1|0\rangle + b_1|1\rangle)$ та $(a_2|0\rangle + b_2|1\rangle)$.

Іншим способом визначення *переплутаного* стану є таке. Ми знаємо, що тензорний добуток має властивості асоціативності та комутативності. Отже, ми можемо виносити за знак тензорного добутку спільні множники. Наприклад, стан

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$$

можна зобразити як

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle \otimes (|0\rangle + |1\rangle)$$

або $|\psi\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |11\rangle)$ - у вигляді $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |1\rangle$. Такі стани будемо вважати не переплутаними. А от квантовий стан $|Q\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ не можна подати у вигляді тензорного добутку двох однокубітових станів. Отже “винести спільний множник” з такої конструкції не можна. Такий стан будемо вважати переплутаним. З такої точки зору квантовий стан $|\phi\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$ також вважається переплутаним.

Можна подивитися на переплутані стани ще з іншого боку, з точки зору вимірювань. Частинки не будуть переплутаними, якщо вимірювання однієї з них не вплинуть на стан іншої. Наприклад, стан $|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |01\rangle) = \frac{1}{\sqrt{2}}|0\rangle \otimes (|0\rangle + |1\rangle)$ не вважається переплутаним, оскільки при будь-якому вимірюванні першого біта ми отримуємо нуль незалежно від того, вимірювався другий біт, чи ні. Аналогічно й вимірювання другого біта може дати нуль або одиницю з такою ж ймовірністю незалежно від того, чи вимірювався перший біт.

Цього не можна сказати про стан $|Q\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. У цьому випадку отримані значення будь-яких кубітів істотно залежать від того, чи проводилося вимірювання іншого біта. Наприклад, якщо при вимірюваннях першого біта ми отримали нуль, то значення другого біта може бути тільки нуль, і ніяк не одиниця. Якщо ж при вимірюванні першого біта отримано одиницю, то другий автоматично також дорівнює одиниці. Аналогічні міркування справедливі й для другого біта. Таким чином, цей стан буде переплутаним. Переплутаними будуть також стани $|\phi\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$ або $|\theta\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle)$, який, по суті, є тим самим станом $|\phi\rangle$.

Підсумуємо викладену інформацію. Квантовий стан, наприклад, двокубітний, будемо вважати переплутаним, якщо:

– його не можна зобразити у вигляді суперпозиції двох однокубітних станів;

- з виразу, який характеризує стан квантової системи, не можна винести “спільний множник”;
- значення, отримані при вимірюванні одного з кубітів істотно залежать від результатів вимірювання іншого.

Переплутані стани грають визначальну роль у квантових алгоритмах, безпосередньо відповідаючи за квантове прискорення обчислень.

Квантовий паралелізм

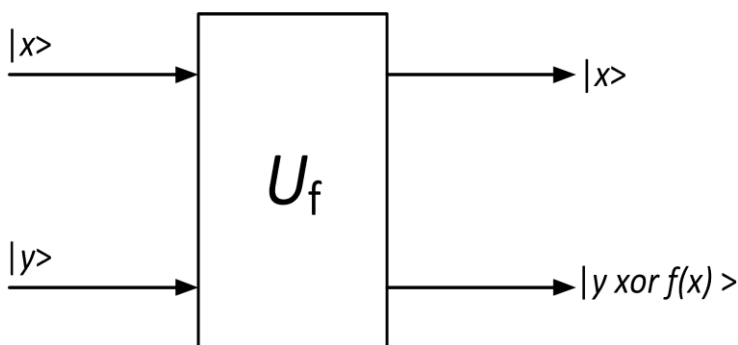
Для початку уведемо поняття квантового оракула. Припустимо, що ми маємо деяку квантову схему (гейт), яка реалізує на практиці обчислення якоїсь функції $f(x)$. Нехай це буде унарна операція такого типу:

$$U_f : |x, y\rangle \rightarrow |x, y \oplus f(x)\rangle.$$

Ми не знаємо ані структури цієї схеми, ні як вона реалізує обчислення функції $f(x)$. Для нас головне, що вона це робить. У цьому випадку така схема виступає як “чорна скринька”. На вхід цієї “скриньки” подається значення аргументів, на виході ми маємо результат обчислення функції. Ми будемо називати таку “чорну скриньку” *квантовим оракулом* або просто *оракулом*.

Для того, щоби обчислити $f(x)$, нам треба подати на вхід оракула стан $|x, 0\rangle$ тоді на виході отримаємо:

$$U_f : |x, 0\rangle \rightarrow |x, 0 \oplus f(x)\rangle \rightarrow |x, f(x)\rangle.$$



Що відбудеться, якщо ми подамо на оракул переплутаний стан? Відповідь проста, але досить дивна. Оскільки U_f - лінійне унарне перетворення, то застосування його до суперпозиційного стану

приводить до суперпозиції результатів. Отже таким способом можна обчислити $f(x)$ для усіх значень аргументу одразу, при одноразовому застосуванні оракула. Такий ефект називають *квантовим паралелізмом*.

На практиці це реалізується в такий спосіб. Регістр з n -кубітів заповнюють нулями. Далі до цього регістра застосовують гейт Адамара для отримання суперпозиційного стану:

$$|Q\rangle = \frac{1}{\sqrt{2^n}}(|000\dots 0\rangle + |000\dots 1\rangle + \dots + |111\dots 1\rangle) = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle.$$

До цього стану додається регістр нулів і все подається на вхід квантового оракула. Тоді на виході отримаємо:

$$U_f : \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x, 0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} U_f |x, 0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x, f(x)\rangle.$$

Необхідно відзначити, що оскільки n -кубітів дозволяють одночасно працювати з 2^n станами, то квантовий паралелізм може надати користувачеві експоненційне зростання обчислювального простору при лінійному зростанні об'єму фізичного простору.

В якості прикладу розглянемо дію оператора Тоффолі для обчислення кон'юнкції двох величин.

На вхід системи подамо суперпозицію усіх можливих значень x та y крім $x=1, y=1$.

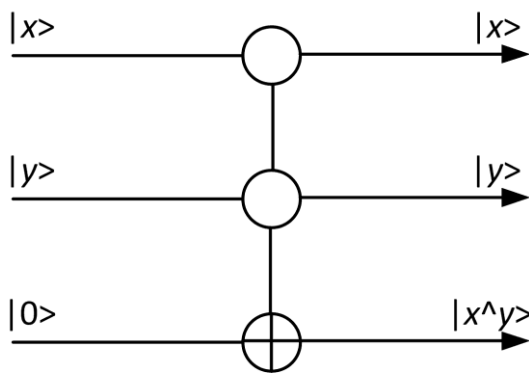
$$H|0\rangle \otimes H|0\rangle \otimes |0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle = \frac{1}{2}(|000\rangle + |010\rangle + |100\rangle + |110\rangle).$$

На результат подіємо гейтом Тоффолі:

$$T : \left\{ \frac{1}{2}(|000\rangle + |010\rangle + |100\rangle + |110\rangle) \right\} = \frac{1}{2}(|000\rangle + |011\rangle + |101\rangle + |111\rangle).$$

Цю результуючу суперпозицію можна розглядати як таблицю істинності деякої логічної функції. Тут перші два біти розглядаються як вхідні значення, а третій — як значення функції. Видно, що ці результати повністю відповідають таблиці істинності кон'юнкції двох змінних: x та y .

Схематично це можна зобразити таким чином, як подано на рисунку.



На перший погляд тут немає жодної переваги над класичними обчисленнями, адже якщо ми проведемо вимірювання такої суперпозиції, ми отримаємо лише один результат, до того ж невідомо який.

Насправді ж важливість квантового паралелізму важко переоцінити. Існує цілий клас квантових перетворень, які дають змогу значно збільшити ймовірність отримання потрібного результату. Керування квантовим паралелізмом такого роду зовсім не має класичних аналогів і часто є основою тих чи інших квантових обчислень. Однак, розв'язання таких задач вимагає від дослідника використання спеціальних математичних методів та нестандартних засобів програмування.

3.5. Парадокс Ейнштейна-Подольські-Розена

Використовуючи поняття переплутаного стану, Альберт Ейнштейн, Борис Подольські та Натан Розен, запропонували уявний експеримент, який, здається порушує основні принципи теорії відносності. Цей експеримент було описано у їх статті “Чи можна вважати квантово-механічне описання фізичної реальності повним?”, яка вийшла у друці 1935 року [20]. Цей експеримент пізніше було названо парадоксом Ейнштейна-Подольські-Розена (ЕПР-парадоксом).

Суть цього парадоксу полягає в наступному.

Припустимо, у нас є джерело, яке генерує максимально переплутані частинки, наприклад, в стані $|Q\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Ці частинки називаються ЕПР-парою. Припустимо, що учасники експерименту отримали по одній частинці з цієї пари і роз'їхалися на максимально можливу відстань.

Нехай власник першої частинки виконав її вимірювання і з'ясовується, що його частинка знаходиться в стані $|0\rangle$. Це означає, що загальний стан системи повинен бути $|00\rangle$. Отже, коли власник другої частинки виконає вимірювання, він також отримає $|0\rangle$. Аналогічно, якщо перша частинка при вимірюваннях продемонструє стан $|1\rangle$, то друга неодмінно буде в такому ж стані $|1\rangle$. Зміна стану відбувається миттєво, не

дивлячись на те, що частинки знаходяться на довільній відстані. Виходить, що учасники експерименту можуть миттєво обмінюватися інформацією. Однак, це протирічить спеціальній теорії відносності, яка доводить, що найбільша швидкість передавання інформації дорівнює швидкості світла у вакуумі.

Насправді суть парадоксу значно глибше. В загальному вона полягає в тому, чи може квантова механіка адекватно описувати фізичну реальність, тобто чи це повна наука, і вона не протирічить іншим, відомим та доведеним теоріям, чи квантова механіка є наукою, яка вимагає значного вдосконалення й узгодження з цими теоріями.

Сьогодні існує кілька інтерпретацій цього парадоксу.

1. Теорія прихованих параметрів Девіда Бома.
2. Теорема Белла та її експериментальні перевірки.
3. Багатосвітова інтерпретація.
4. Квантовомеханічна інтерпретація (інтерпретація Н.Бора).

Розглянемо ці тлумачення детальніше.

Теорія прихованих параметрів Девіда Бома

У 1952 році Девід Бом у своїй книзі “Квантова теорія” запропонував реальний фізичний експеримент, який на той час здійснити було нереально, однак, він дозволяв безпосередньо перевірити правильність різних тлумачень ЕПР-парадоксу [21]. Суть експерименту полягає в тому, що деяке джерело випромінює два фотони у переплутаному стані в різні боки. Кожен фотон має якісь свої конкретні значення компонентів спіну в напрямках x , y , z . Ми можемо вимірювати ці значення, але тільки один компонент за раз. Нехай ми виміряли x -компонент спіну першого фотона і отримали якусь величину x_0 . Оскільки фотони знаходяться у переплутаному стані, то яке б не було значення y -компоненти першого фотона, y -компонент спіну другого фотона обов'язково повинен бути протилежним за напрямом. Таким чином, вимірюючи y -компонент першого фотона, ми неявним чином вимірюємо такий самий компонент другого, як це й було описано в ЕПР-парадоксі. Якщо би така картина була справедливою для усіх вимірів та усіх напрямів випромінювання фотонів, то це протирічило би твердженням нерівності Гейзенберга, яка стверджує, що не можна достовірно виміряти дві величини однієї частинки.

На основі таких міркувань, Д. Бом робить висновок про те, що квантова механіка як наука неповна, таким чином припускаючи існування прихованих параметрів, що можуть розв'язати цю проблему:

“... сучасна форма квантової теорії вказує на те, що Всесвіт не можна однозначно відобразити ніякою сукупністю строго визначених математичних величин, і що повна теорія завжди вимагатиме більш загальних понять, ніж розкладання на точно визначені компоненти ...”

Що означає поняття “прихованих параметрів”? Розглянемо два таких висловлювання:

1) В момент часу $t=1$ с частинка буде знаходитися в точці з координатою $x=1$ м. Це – приклад класичного, детерміністичного твердження.

2) В момент часу $t=1$ с частинка буде знаходитися між координатами $x=1$ м та $x=1,1$ м з ймовірністю $P=0,8$. Це – приклад того, що можуть дати квантовомеханічні обчислення.

Як бачимо, квантовомеханічні обчислення передбачають наявність деякого елемента випадковості у фізичних процесах. У зв'язку з цим, фізики й розділилися на два (як мінімум!) табори. Одні вважали, що випадковість притаманна самій фізичній природі. Інші були впевнені, що ніякої випадковості в природі не існує, а ймовірності, які виникають у квантовомеханічних розрахунках насправді є наслідком недосконалості самої теорії та нашого неповного знання про природу речей. Загалом ці два підходи можна охарактеризувати так:

Класичний підхід: поточний стан фізичної системи повністю та однозначно визначає її майбутній стан. Наприклад, якщо зараз система знаходиться в стані “А”, то через секунду вона буде знаходитися в стані “В”. І якщо квантова механіка стверджує, що через секунду квантова система буде знаходитися або в стані “В”, або в стані “С”, то це лиш значить, що вона при розрахунках не враховує якісь невідомі (приховані) параметри системи, врахування яких може однозначно визначати її майбутній стан.

Квантовий підхід: поточний стан системи несе в собі кілька (загалом — безмежну кількість) можливих варіантів її майбутнього стану. Який саме з цих можливих варіантів буде реалізовано — принципово неможливо передбачити, оскільки вибір тут відбувається абсолютно випадково, причому немає ніяких прихованих параметрів, які б могли вплинути на цей вибір. Можна говорити лише про ймовірність того чи іншого вибору.

Прибічники класичного та квантового підходу сперечаються вже багато років але жодна сторона не може надати вирішальних доказів на користь тієї чи іншої моделі. Тим не менше, часто результати експериментів, наприклад, таких, як запропонував Д.Бом, можна пояснити

за допомогою понять квантової механіки досить просто, не залучаючи до розгляду теорії прихованих параметрів.

Дійсно, нехай у нас є поляризатор, який випромінює фотони. Поляризатор має два канали “+” та “-”, напрямлені в різні боки. Ймовірність виявити фотон в будь-якому каналі — випадкова величина, причому рівноймовірна. Отже, $P_+(a)=P_-(a) = 0,5$. Аналогічно й для іншого фотона: $P_+(b)=P_-(b) = 0,5$. Саме цей результат говорить про те, що ми не можемо сказати точно яка поляризація у кожного фотона переплутаної пари, оскільки це процес ймовірнісний, і кожне вимірювання її дасть випадковий результат з ймовірністю 0,5. Тоді можна підрахувати ймовірності того, що фотони будуть однакової або різної поляризації :

$$\begin{aligned}P_{++}(a,b) &= P_{--}(a,b) = 1/2 \cos^2(a,b); \\ P_{+-}(a,b) &= P_{-+}(a,b) = 1/2 \sin^2(a,b).\end{aligned}$$

Якщо фотони летять строго в різні боки (кут між a та b дорівнює 180°), то

$$\begin{aligned}P_{++}(a,b) &= P_{--}(a,b) = 1/2; \\ P_{+-}(a,b) &= P_{-+}(a,b) = 0.\end{aligned}$$

Звідси слідує, що якщо ми виявляємо один фотон в каналі “+”, то інший не може бути в каналі “-” і навпаки. Отже тут спостерігається кореляція між двома випадковими подіями. Якщо ввести коефіцієнт кореляції, то у розглянутому випадку вона буде повною.

Теорема Белла та її експериментальні перевірки

У 1964 році з'явилася теорема, яка може примирити прибічників класичного та квантового підходів. Запропонував її Джон Стюарт Белл [22]. Вона стверджує, що незалежно від того, містить квантова теорія деякі приховані параметри чи ні, можна провести серійний експеримент, статистичні результати якого підтвердять або спростують наявність таких параметрів. Коротше кажучи, Дж. Белл запропонував спосіб, яким можна було перевести перевірку існування прихованих параметрів з філософської площини в експериментальну. Белл вивів деяку формулу (нерівність) для математичної обробки запропонованого їм експерименту та довів таке:

– якщо нерівність виконується, то праві прибічники класичного підходу, тобто квантова механіка не враховує деякі приховані параметри;

– якщо нерівність не виконується, то квантова механіка — повна наука, не треба шукати прихованих параметрів, ймовірнісність притаманна фізичній реальності і є її властивістю.

Суть експерименту Белла полягає в такому.

Нехай у нас є джерело переплутаних частинок, які випромінюються ним у протилежні боки. Частинки проходять через поляризатори (якщо це фотони) або через прилади з магнітним полем (прилади Штерна-Герлаха, якщо це частинки з напівцілим спіном). Проходячи через поляризатори (прилади Штерна-Герлаха), частинки набувають певної поляризації (напрямку спіну), а потім реєструються детекторами поляризації (спіну). Якщо вектор поляризації (спіну) частинки напрямлений вгору, таку частинку будемо позначати “+”; якщо вниз – “-”. Таким чином, позначивши частинки, що випромінюються в протилежні боки, як “a” та “b”, можна записати таке: $N_{++}(a,b)$ – кількість частинок, які мали поляризацію (спін) вгору в обох детекторах, тобто кількість частинок, поляризація (спін) яких були: тип a – вгору; тип b – вниз. Аналогічно можна позначити $N_{--}(a,b)$ – кількість частинок з поляризацією “-” “-” в обох напрямках; $N_{+-}(a,b)$, $N_{-+}(a,b)$ – аналогічно.

Тоді коефіцієнт кореляції можна записати таким чином:

$$E(a,b) = 1/N [N_{++}(a,b) + N_{--}(a,b) - N_{+-}(a,b) - N_{-+}(a,b)],$$

де N – повна кількість частинок, що їх випромінило джерело. Виконавши такі вимірювання з різною орієнтацією (a,b); (a,b'); (a',b) та (a',b'), можна обчислити величину:

$$S(a,a',b,b') = E(a,b) + E(a',b') + E(a',b) - E(a,b')$$

та скласти нерівність Белла:

$$-2 \leq S(a,a',b,b') \leq 2.$$

З 1972 року нерівність Белла перевірялася неодноразово. Результати подано в таблиці 3.3.

Результати перевірки нерівності Белла

Автори експериментів	Отримані значення S
Фрідман і Клаузер (1972 рік)	2.28 ± 0.04
А.Аспе зі співр. (1982 рік)	2.23 ± 0.05
Дж. Вайс зі співр. (1998 рік)	2.23 ± 0.09
Шайдл зі співр. (2010 рік)	2.37 ± 0.02

Як бачимо з таблиці, результати усіх експериментів доводять, що нерівність Белла не виконується. Це означає, що правильною виявляється квантова механіка, отже класичний підхід до парадоксу ЕПР застосовувати не можна. Тим не менше, прибічники теорій прихованих параметрів весь час знаходять в експериментах слабкі місця, що дає їм право сумніватися в наведених результатах.

Багатосвітова інтерпретація

Іншим способом пояснення парадоксу ЕПР була й залишається багатосвітова інтерпретація. Суть її полягає в наступному.

Стан частинок переплутаної пари являє собою квантову суперпозицію усіх можливих станів. Згідно з цими теоріями це можна інтерпретувати як суперпозицію станів однакових паралельних всесвітів, що не взаємодіють між собою. Кожен всесвіт містить альтернативну історію частинок пари і характеризується окремими значеннями фізичних величин, тобто містить один визначений стан. Поки не проведено вимірювань, неможливо встановити в якому саме всесвіті відбувається експеримент. В момент вимірювання обирається один з паралельних всесвітів, відбувається незворотне їх “розщеплення” й історія частинок стає визначеною. Ось чому подальші вимірювання не впливають на частинки і протиріччя з принципом причинності відсутнє.

Квантовомеханічна інтерпретація

Цю інтерпретацію було сформульовано засновниками квантової механіки Нільсом Бором та Вернером Гейзенбергом приблизно у 1927 році під час спільної роботи у Копенгагені. Тому часто її називають копенгагенською інтерпретацією. Вона стверджує, що ймовірнісного характеру передбачень квантової механіки принципово не можна позбавитися, причому він зовсім не свідчить про те, що наші знання

обмежені, квантова наука не повна, що вона містить якісь приховані параметри. Це означає, що в квантовій механіці результат вимірювань принципово недетермінований.

Н.Бор та В.Гейзенберг відзначають, що фізика взагалі та квантова механіка зокрема, є наукою вимірювальних процесів. Тому питання типу *“... де була частинка до того, як я зареєстрував її положення ...”* або *“... якими були параметри частинки до вимірювання ...”* беззмістовні. Процес вимірювання приводить до випадкового вибору в точності одного з можливих варіантів значень параметрів, що вимірюються, які “містилися” у переплутаному стані.

Автори цієї інтерпретації відзначають також, що тут немає ніякого протиріччя з теорією відносності, оскільки переплутаний стан квантової системи з кількох частинок описується однією математичною функцією, яка називається хвильовою, причому її не можна розділити на окремі функції, які б описували стан кожної частинки системи окремо. Оскільки це так, то говорити про передавання якоїсь інформації між частинами хвильової функції всередині її також беззмістовно. Звичайно, така інтерпретація також не позбавлена недоліків.

Наприклад, процесу вимірювання тут надається якійсь дуже особливий статус, але пояснення, що ж він таке, які його визначальні риси, як його відділити від інших, невимірювальних процесів, немає. Довгі роки копенгагенська інтерпретація була найпопулярнішою серед спеціалістів з квантової механіки. Зараз картина змінюється.

У 1997 році на симпозіумі в Мерилендському університеті було проведено опитування про підтримку тієї чи іншої інтерпретації квантової механіки. Результати подано в таблиці 3.4

Таблиця 3.4.

Результати опитування про підтримку інтерпретації квантової механіки

Інтерпретація	Віддано голосів
Копенгагенська	13
Багатосвітова	8
Теорія прихованих параметрів	4
Критерій непротиричливості	4
Модифікована динаміка	1
Жодна з наведених	18

Як бачимо, популярність копенгагенської інтерпретації, яка раніше була домінуючою, поступово спадає.

Сьогодні багато фізиків схиляються до так званої “ніякої” інтерпретації квантової механіки, а значить, і ЕПР-парадоксу, яку вдало висловив Девід Мермін: “*Shut up and calculate!*”, що в м’якому перекладі з англійської можна подати як: “*Мовчи і працюй!*”

Однак це питання, можливо не остаточно, але було вирішено у 2022 році, коли Міжнародний Нобелівський комітет присудив Нобелівську премію з фізики трьом науковцям: Алену Аспе, Антону Цайлінгеру та Джону Клаузеру. У формулюванні підстав для нагородження було сказано, що премію присуджено “...за експерименти із заплутаними фотонами, встановлення порушення нерівностей Белла та новаторство у квантовій інформаційній науці”. Усі згадані науковці якраз і займалися експериментами з нерівністю Белла, і як ми знаємо, встановили факт його порушення. Це означає, що світова наука нарешті вирішила, яка з усіх інтерпретацій ЕПР-парадоксу вважається правильною. Як бачимо, питання було вирішено на користь копенгагенської інтерпретації. Таким чином, і ми будемо вважати, що фізичний світ визнав справедливність трактування ЕПР-парадоксу від Н.Бора та В.Гейзенберна, і це питання можна вважати закритим.

3.6. Квантові алгоритми

Квантова телепортація

Однією з найбільш популярних в комп’ютерній техніці операцій є копіювання інформації. У класичних комп’ютерах все відбувається звичним для нас чином: зчитуються дані з однієї комірки пам’яті та записуються в іншу. У квантовому випадку читання інформації з кубіта може прирівнюватися до вимірювання його стану. Таким чином, існує певна підозра, що операцію копіювання стану кубіта в цьому випадку взагалі виконати неможливо.

Дійсно, припустимо, що нам треба скопіювати невідомий квантовий однокубітний стан. Припустимо, що ми маємо деякий унарний оператор, який виконує цю операцію. Тоді ми зможемо записати:

$$U: |a0\rangle = |aa\rangle. \quad (3.3)$$

Формула (3.3) справедлива для усіх станів $|a\rangle$. Нехай $|a\rangle$ та $|b\rangle$ - деякі ортогональні стани. Тоді справедливо: $U:|a0\rangle=|aa\rangle$. та $U:|b0\rangle=|bb\rangle$. Припустимо також, що ми маємо квантовий стан $|c\rangle=\frac{1}{\sqrt{2}}(|a\rangle+|b\rangle)$. Тоді можна записати:

$$U:|c0\rangle=\frac{1}{\sqrt{2}}(U|a0\rangle+U|b0\rangle)=\frac{1}{\sqrt{2}}(|aa\rangle+|bb\rangle). \quad (3.4)$$

З другого боку:

$$U|c0\rangle=|cc\rangle, \quad (3.5)$$

оскільки U — клонуюче перетворення. Підставляючи в (3.5) значення $|c\rangle$, отримаємо:

$$|cc\rangle=\frac{1}{\sqrt{2}}(|a\rangle+|b\rangle)\otimes\frac{1}{\sqrt{2}}(|a\rangle+|b\rangle)=\frac{1}{2}(|aa\rangle+|ab\rangle+|ba\rangle+|bb\rangle)\neq\frac{1}{\sqrt{2}}(|aa\rangle+|bb\rangle). \quad (3.6)$$

Звідси явно видно, що унарної операції, яка може копіювати (клонувати) невідомий стан, не існує. Цей факт отримав назву *теорему про заборону клонування*. Очевидно, що клонувати стан за допомогою вимірювання також неможливо, оскільки виміри руйнують квантовий стан. Відзначимо, що відомий квантовий стан легко копіюється або клонується. Проблема стосується лише невідомого стану.

Як же можна передати квантовий стан системи? Без вирішення цієї проблеми будь-які обчислення неможливі.

Для розв'язання цієї задачі розроблено схему *квантової телепортації*. Цей термін з'явився після публікації в 1993 році статті Чарлза Беннетта та Жіля Brassара [14]. Там пояснюється протокол квантової телепортації та її відмінності від того, який зміст вкладається у це поняття звичайними людьми. Квантова телепортація не передає ні енергії, ні речовини на відстані. Це гарантує, що під час квантової телепортації не порушуються закони фізики. Більше того, квантовий стан кубіта, який телепортується, після закінчення операції руйнується.

Припустимо, що сторона A хоче передати стороні B невідомий квантовий стан $|Q\rangle=a|0\rangle+b|1\rangle$. Сторони вже мають по квантовому біту з

переплутаної пари $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Ідея полягає в тому, що сторона А підмішує невідомий кубіт до свого “переплутаного” біта, проводить вимірювання і відправляє результати стороні В. Та, у свою чергу, виконує допоміжні перетворення і отримує $|Q\rangle$.

Розглянемо протокол квантової телепортації детальніше.

Стартовий стан. Сторона А має один кубіт з “переплутаної” пари і кубіт з невідомим станом $|Q\rangle$. Сторона В має лише один “переплутаний” кубіт.

Сторона А підмішує невідомий кубіт до пари, ставлячи його ліворуч:

$$|\psi\rangle = \frac{a}{\sqrt{2}}|000\rangle + \frac{a}{\sqrt{2}}|011\rangle + \frac{b}{\sqrt{2}}|100\rangle + \frac{b}{\sqrt{2}}|111\rangle. \quad (3.7)$$

Сторона А застосовує CNOT до перших двох кубітів стану (3.7):

$$|\psi\rangle = \frac{a}{\sqrt{2}}|000\rangle + \frac{a}{\sqrt{2}}|011\rangle + \frac{b}{\sqrt{2}}|110\rangle + \frac{b}{\sqrt{2}}|101\rangle. \quad (3.8)$$

Якщо тепер сторона А виміряє середній кубіт, то вона отримає стан системи або $\frac{1}{\sqrt{2}}(a|00\rangle + b|11\rangle)$, або $\frac{1}{\sqrt{2}}(a|01\rangle + b|10\rangle)$.

Результати вимірювань відправляються стороні В. В разі, якщо цей результат дорівнює $|1\rangle$, сторона В інвертує свій кубіт. Таким чином, стан системи завжди буде однаковим і дорівнюватиме $|\psi\rangle = \frac{1}{\sqrt{2}}(a|00\rangle + b|11\rangle)$.

Тепер сторона А застосовує гейт Адамара до першого кубіта (це той самий невідомий біт, стан якого треба передати). В результаті отримаємо:

$$|\psi\rangle = \frac{a}{2}|00\rangle + \frac{a}{2}|10\rangle - \frac{b}{2}|11\rangle + \frac{b}{2}|01\rangle. \quad (3.9)$$

Тепер сторона А вимірює перший кубіт. Якщо результат дорівнює $|0\rangle$, то стан системи буде $|\psi\rangle = \frac{1}{2}(a|0\rangle + b|1\rangle)$. Якщо ж він дорівнює $|1\rangle$, тоді маємо $|\psi\rangle = \frac{1}{2}(a|0\rangle - b|1\rangle)$. В останньому випадку сторона В змінює знак біля свого кубіта, і тоді ми завжди отримуємо стан системи $|\psi\rangle = \frac{1}{2}(a|0\rangle + b|1\rangle)$. Таким чином, ми телепортували стан квантового біта від сторони А до

сторони В. Однак для того, щоби здійснити таку телепортацію, необхідно передати два класичних біти, - результати вимірів сторони А. Підсумовуючи, складемо таблицю 3.5, яка продемонструє вищенаведене.

Таблиця 3.5.

Операції над кубітами

Результати вимірювань сторони А	Стан кубіта сторони В	Операції сторони В над кубітом	Результуючий стан
0, 0	$(a 0\rangle + b 1\rangle)$	Тотожна операція, Тотожна операція	$(a 0\rangle + b 1\rangle)$
0, 1	$(a 0\rangle - b 1\rangle)$	Тотожна операція, Інверсія знаку кубіта	$(a 0\rangle + b 1\rangle)$
1, 0	$(a 1\rangle + b 0\rangle)$	Інверсія кубіта, Тотожна операція	$(a 0\rangle + b 1\rangle)$
1, 1	$(a 1\rangle - b 0\rangle)$	Інверсія кубіта, Інверсія знаку кубіта	$(a 0\rangle + b 1\rangle)$

Слід відзначити, що описаний протокол не порушує жодних принципів сучасної фізики, як це може здаватися, зокрема, принципів спеціальної теорії відносності. Дійсно, стан кубіта сторони В не несе ніякої інформації про стан кубіта сторони А, тому не можна стверджувати, що під час квантової телепортації інформація передається зі швидкістю, що перевищує швидкість світла. Більше того, для перенесення стану необхідно передати два класичних біти про результати вимірювань. Це здійснюється стандартними класичними каналами зв'язку, які також не порушують жодних фізичних законів.

Протокол квантової телепортації не порушує також теорему про заборону клонування, оскільки результуючий кубіт знаходиться в стані $|Q\rangle = a|0\rangle + b|1\rangle$, а вихідний кубіт – в одному з базисних станів: $|0\rangle$ або $|1\rangle$.

У практичному розумінні телепортація свідчить, що EPR-пара спільно з двома класичними бітами інформації утворюють ресурс для передавання одного кубіта інформації.

Експериментальну реалізацію квантової телепортації поляризаційного стану фотона було здійснено у 1997 році майже

одночасно групами фізиків під керівництвом А. Цайлінгера з університету Інсбрука Experimental quantum teleportation та Ф. Де Мартіні (університет Риму) [24]. У 2004 році було оголошено про успішну телепортацію квантового стану атомів кальцію (група М. Рібе) та атома берилію (група М. Барретта). У 2006 році науковою групою з Інституту Нільса Бора (Копенгаген) вперше було здійснено телепортацію між об'єктами різної природи: квантами лазерного випромінювання та атомами цезію [25].

У січні 2009 року вченим вперше вдалося телепортувати квантовий стан іона на один метр, а у травні 2010 року дослідниками з університету Сінхуа виконано передавання квантового стану на 16 кілометрів [26]. На сьогодні, однак, рекорд телепортації належить вченим Китайського університету науки і технологій, які у 2017 році здійснили передавання квантового стану на відстані 1200 кілометрів [27].

3.6.1. Протокол квантового щільного кодування

Протокол квантового щільного кодування використовує один кубіт разом з EPR-парою для кодування та передавання такої кількості інформації, яку можна передати лише двома класичними бітами [11]. Оскільки бітами EPR-пари можна обмінятися завчасно, виходить, що у випадку маніпулювання квантовими бітами треба фізично передати лише один біт там, де в класичному випадку були б потрібні два. Цей результат до деякої міри дивний, оскільки ми звикли до того, що один біт містить лише одну одиницю інформації.

Щоб зрозуміти в чому тут суть, розглянемо протокол квантового щільного кодування детальніше [11].

Припустимо, що сторони інформаційного обміну А та В мають по одному кубіту з переплутаної пари:

$$|Q\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \quad (3.10)$$

У такому разі вони можуть виконувати операції лише над своїми кубітами.

Припустимо, що сторона А хоче передати стороні В якусь інформацію, наприклад, здійснити вибір одного з чотирьох можливих варіантів або закодованих повідомлень. Для цього сторона А виконує деякі операції над своїм кубітом і відправляє його стороні В. Отримавши

кубіт, сторона В виконує деякі перетворення над парою і дізнається, яку операцію виконала сторона А.

Отже, первісний стан описується співвідношенням (3.10).

Що може зробити сторона А зі своїм кубітом?

1. Може нічого не робити. Тоді стан системи не зміниться:

$$|Q_1\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

2. Може змінити знак біля $|1\rangle$. Отже, будемо мати: $|Q_2\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$.

3. Може переставити місцями $|0\rangle$ та $|1\rangle$. Тоді (3.10) матиме вигляд:

$$|Q_3\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle).$$

4. Може переставити місцями $|0\rangle$ і $|1\rangle$ та змінити знак біля $|1\rangle$. В цьому випадку (3.10) перетвориться у: $|Q_4\rangle = \frac{1}{\sqrt{2}}(|10\rangle - |01\rangle)$.

Оскільки пара кубітів переплутана, то відповідних змін зазнає й кубіт, який знаходиться у сторони В.

Після виконання однієї з чотирьох дій сторона А відправляє свій кубіт стороні В, яка тепер матиме обидва кубіти переплутаної пари. Виконавши відповідні операції над нею та провівши вимірювання, сторона В може сказати, яку операцію було виконано над кубітом сторони А.

Які ж це операції?

1. Сторона В застосовує до пари кубітів CNOT:

а) Для випадку першої операції (коли сторона А нічого не зробила зі своїм кубітом):

$$\text{CNOT} : |Q_1\rangle = \text{CNOT} : \left\{ \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \right\} = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle \equiv |\psi_1\rangle.$$

Як бачимо, перший кубіт пари має переплутаний стан, а другий – дорівнює $|0\rangle$.

б) Для випадку другої операції будемо мати:

$$\text{CNOT} : |Q_2\rangle = \text{CNOT} : \left\{ \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \right\} = \frac{1}{\sqrt{2}}(|00\rangle - |10\rangle) = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)|0\rangle \equiv |\psi_2\rangle.$$

Як бачимо, другий кубіт знову дорівнює $|0\rangle$.

в) У випадку, коли сторона А застосувала третю функцію, отримаємо:

$$\text{CNOT} : |Q_3\rangle = \text{CNOT} : \left\{ \frac{1}{\sqrt{2}} (|10\rangle + |01\rangle) \right\} = \frac{1}{\sqrt{2}} (|11\rangle + |01\rangle) = \frac{1}{\sqrt{2}} (|1\rangle + |0\rangle) |1\rangle \equiv |\psi_3\rangle.$$

В цьому випадку вже другий кубіт дорівнює $|1\rangle$.

г) Для четвертої функції:

$$\text{CNOT} : |Q_4\rangle = \text{CNOT} : \left\{ \frac{1}{\sqrt{2}} (|10\rangle - |01\rangle) \right\} = \frac{1}{\sqrt{2}} (|11\rangle - |01\rangle) = \frac{1}{\sqrt{2}} (|1\rangle - |0\rangle) |1\rangle \equiv |\psi_4\rangle.$$

І в цьому випадку другий кубіт дорівнює $|1\rangle$.

2. Сторона В вимірює значення другого кубіта. Якщо воно дорівнює нулю, то сторона А застосувала або першу, або другу функції, якщо ж отримано одиницю – третю або четверту. Далі необхідно з'ясувати, яку саме функцію було застосовано.

3. Застосуємо гейт Адамара до першого кубіта:

$$H : |\psi_1\rangle = \frac{1}{\sqrt{2}} (H|0\rangle + H|1\rangle) |0\rangle = \frac{1}{\sqrt{2}} \left[\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) + \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \right] = \frac{1}{2} 2|0\rangle = |0\rangle. \quad (3.11)$$

Аналогічно отримаємо вирази для другої, третьої та четвертої функцій:

$$H : |\psi_2\rangle = \frac{1}{\sqrt{2}} (H|0\rangle - H|1\rangle) |0\rangle = \frac{1}{\sqrt{2}} \left[\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) - \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \right] = \frac{1}{2} 2|1\rangle = |1\rangle. \quad (3.12)$$

$$H : |\psi_3\rangle = \frac{1}{\sqrt{2}} (H|0\rangle + H|1\rangle) |1\rangle = \frac{1}{\sqrt{2}} \left[\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) + \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \right] = \frac{1}{2} 2|0\rangle = |0\rangle. \quad (3.13)$$

$$H : |\psi_4\rangle = \frac{1}{\sqrt{2}} (H|0\rangle - H|1\rangle) |1\rangle = \frac{1}{\sqrt{2}} \left[\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) - \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \right] = \frac{1}{2} 2|1\rangle = |1\rangle. \quad (3.14)$$

З формул (3.11) – (3.14) видно, що дія гейтом Адамара на перший кубіт дозволяє остаточно визначити, яку ж саме функцію було застосовано (таблиця 3.6).

Таблиця 3.6.

Операції над кубітами

Значення другого кубіта	Значення першого кубіта	Використана функція
$ 0\rangle$	$ 0\rangle$	$ Q_1\rangle = \frac{1}{\sqrt{2}}(00\rangle + 11\rangle)$. – перша функція
	$ 1\rangle$	$ Q_2\rangle = \frac{1}{\sqrt{2}}(00\rangle - 11\rangle)$. – друга функція
$ 1\rangle$	$ 0\rangle$	$ Q_3\rangle = \frac{1}{\sqrt{2}}(10\rangle + 01\rangle)$. – третя функція
	$ 1\rangle$	$ Q_4\rangle = \frac{1}{\sqrt{2}}(10\rangle - 01\rangle)$. – четверта функція

Таким чином, використовуючи квантову переплутану пару, ми можемо вибрати одну з чотирьох можливих альтернатив за допомогою передавання каналами зв'язку лише одного біту інформації. Такий факт називається.

Очевидно, що для вирішення такої задачі класичними методами необхідно було би передати по каналах зв'язку два класичних біти (таблиця 3.7).

Таблиця 3.7.

Квантове надщільне кодування

Значення бітів	Використана функція
00	Перша функція
01	Друга функція
10	Третя функція
11	Четверта функція

Описаний протокол був запропонований Чарлзом Беннеттом та Стівеном Візнером у 1992 році. Це один з перших прикладів квантового прискорення обчислень та ущільнення передавання інформації.

3.6.2. Алгоритм Дойча

Алгоритм Дойча (часто відомий як алгоритм Дойча-Джоза) був запропонований Девідом Дойчем та Річардом Джозом у 1992 році і став одним з перших алгоритмів, призначених для квантових комп'ютерів [13]. Ці алгоритми використовують суто квантові явища, такі як квантова суперпозиція та квантова заплутаність, та демонструють значний приріст швидкості виконання порівняно з аналогічними класичними алгоритмами.

Суть задачі Дойча-Джоза полягає у визначенні, чи є вибрана функція двійкової змінної постійною (тобто такою, що дорівнює 0 або 1 незалежно від значення аргументу) або збалансованою (тобто дорівнює 0 для половини області визначення, а для іншої половини – 1). При цьому апіорі відомо, що функція може бути або константою, або збалансованою.

Для розв'язку цієї задачі класичному алгоритму необхідно виконати $2^{n+1}+1$ обчислень функції $f(n)$. Квантовий алгоритм Дойча-Джоза здатний дати правильну відповідь, один раз виконавши фазовий запит, що відповідає функції $f(n)$.

Алгоритм Дойча-Джоза ґрунтується на алгоритмі, розробленому Девідом Дойчем у 1985 році. Він є частковим випадком першого, і розглядає функцію однієї змінної. Ми розглянемо цей алгоритм якраз у варіанті функції однієї змінної [13].

Нехай у нас є деякий квантовий оракул, який реалізує деяку функцію $f(x)$.

$$U_f = (|x\rangle|0\rangle) \rightarrow |x\rangle|y\rangle \oplus f(x). \quad (3.15)$$

Ця функція отримує на вході один кубіт та повертає також один кубіт. Які тут можуть бути варіанти? Наведена нижче таблиця 3.8 дає відповідь на це запитання.

Таблиця 3.8.

Результати реалізації функцію $f(x)$

	$x=0$	$x=1$	$f(x)$	Тип
f_0	0	0	I_0	Постійна
f_1	0	1	$CNOT_{iO}$	Збалансована
f_2	1	0	$NOT_O CNOT_{iO}$	Збалансована
f_3	1	1	NOT_O	Постійна

Як видно з таблиці, функції f_0 та f_3 не змінюють своїх значень для усього вхідного діапазону, тобто вони є постійними, а f_1 та f_2 – збалансовані. Функцію f_0 можна вважати як тотожну операцію за виходом I_0 , f_1 – $CNOT$ за входом та виходом (iO); f_2 – можна зобразити як $NOT_O CNOT_{iO}$, а f_3 – як заперечення за виходом.

Суть алгоритму Дойча зводиться до наступного.

Нехай ми маємо два кубіти в нульовому стані: $|0\rangle|0\rangle$. Подіємо на них гейтом NOT , щоби перевести у стан $|1\rangle|1\rangle$. Можна було би, звичайно, приготувати кубіти в одиничному стані, але простіше зробити це для нульового, а потім отримати одиничний за допомогою заперечення.

Подіємо гейтом Адамара на кожен кубіт пари:

$$(H \otimes H)|1\rangle|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{2}(|00\rangle - |10\rangle - |01\rangle + |11\rangle) \equiv |Q\rangle. \quad (3.16)$$

Ми сформуваємо вихідний стан і можемо діяти на нього квантовим оракулом.

$$U_f : |Q\rangle = \frac{1}{2} \{ U_f |00\rangle - U_f |10\rangle - U_f |01\rangle + U_f |11\rangle \} = \frac{1}{2} \{ |0\rangle |f(0)\rangle - |1\rangle |f(1)\rangle - |0\rangle |1 \oplus f(0)\rangle + |1\rangle |1 \oplus f(1)\rangle \}.$$

Згрупуємо доданки біля однакових кубітів. Будемо мати:

$$|Q\rangle = \frac{1}{2} \{ |0\rangle (|f(0)\rangle - |1 \oplus f(0)\rangle) - |1\rangle (|f(1)\rangle - |1 \oplus f(1)\rangle) \}. \quad (3.17)$$

Формула (3.17) буде робочою для наступного матеріалу.

Розглянемо випадок постійної функції. В такому разі можна записати: $f(0)=f(1)$. Тоді формула (3.17) буде мати вигляд:

$$|Q\rangle = \frac{1}{2}\{|0\rangle(|f(0)\rangle - |1\oplus f(0)\rangle) - |1\rangle(|f(0)\rangle - |1\oplus f(0)\rangle)\} = \frac{1}{2}\{(|0\rangle - |1\rangle)(|f(0)\rangle - |1\oplus f(0)\rangle)\}. \quad (3.18)$$

У випадку збалансованої функції ми можемо записати такі співвідношення:

а) якщо $f(0)=0$:

$$1\oplus f(0) = 1\oplus 0 = 1 = \overline{f(0)} = f(1);$$

б) якщо $f(0)=1$:

$$1\oplus f(0) = 1\oplus 1 = 0 = \overline{f(0)} = f(1);$$

в) якщо $f(1)=0$:

$$1\oplus f(1) = 1\oplus 0 = 1 = \overline{f(1)} = f(0);$$

г) якщо $f(1)=1$:

$$1\oplus f(1) = 1\oplus 1 = 0 = \overline{f(1)} = f(0).$$

З урахуванням цього формула (3.17) для збалансованої функції буде мати вигляд:

$$|Q\rangle = \frac{1}{2}\{|0\rangle(|f(0)\rangle - |1\oplus f(0)\rangle) - |1\rangle(|f(1)\rangle - |1\oplus f(1)\rangle)\} = \frac{1}{2}\{|0\rangle(|f(0)\rangle - |f(1)\rangle) + |1\rangle(|f(0)\rangle - |f(1)\rangle)\}.$$

Винесемо спільний множник:

$$|Q\rangle = \frac{1}{2}\{(|0\rangle + |1\rangle)(|f(0)\rangle - |f(1)\rangle)\}. \quad (3.19)$$

Тепер подіємо на отримані результати гейтом Адамара.

Для постійної функції будемо мати:

$$\begin{aligned} H|Q\rangle &= \frac{1}{2}H(|0\rangle - |1\rangle)(|f(0)\rangle - |1\oplus f(0)\rangle) = \frac{1}{2}\frac{1}{\sqrt{2}}\{(|0\rangle + |1\rangle) - (|0\rangle - |1\rangle)\}\{|f(0)\rangle - |1\oplus f(0)\rangle\} = \\ &= \frac{1}{2}\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle - |0\rangle + |1\rangle)\{|f(0)\rangle - |1\oplus f(0)\rangle\} = \frac{1}{\sqrt{2}}|1\rangle\{|f(0)\rangle - |1\oplus f(0)\rangle\}. \end{aligned}$$

Для збалансованої функції:

$$H|Q\rangle = \frac{1}{2}H(|0\rangle + |1\rangle)(|f(0)\rangle - |f(1)\rangle) = \frac{1}{\sqrt{2}}|0\rangle(|f(0)\rangle - |f(1)\rangle).$$

Відзначимо, що ми застосували квантовий оракул лише один раз.

Тепер нам потрібно виміряти перший кубіт. Якщо ми отримаємо в результаті вимірювання одиницю, то використовували постійну функцію, якщо ж нуль, – збалансовану.

Якщо би ми використовували класичні обчислення, то нам необхідно було б використати оракул двічі: для $x=0$ та $x=1$. Однак, ми тоді могли би не тільки сказати яку функцію використали, а й обчислити значення цієї функції. У квантовому випадку ми не можемо обчислити значення функції, однак можемо сказати тип використаної функції лише за одне її застосування. В цьому і є суть квантового прискорення обчислень.

3.7. Квантова криптографія

3.7.1. Алгоритм факторизації цілих чисел (алгоритм Шора)

Алгоритм Пітера Шора – це квантовий алгоритм, призначений для швидкого розкладання цілих чисел на прості множники. Він дозволяє розкласти число N за час $O(\lg^3 N)$, використовуючи $O(\lg N)$ кубітів. Алгоритм було розроблено Пітером Шором 1994 року [15]. Лише через сім років, 2001 року, змогли продемонструвати його працездатність: група спеціалістів з компанії IBM розклали число 15 на множники 3 та 5, використавши лабораторний зразок квантового комп'ютера з сімома кубітами.

Як бачимо, алгоритм Шора не тільки здатен розкласти числа на множники, але може це зробити, як довів сам Пітер Шор, за поліноміальний час. Вважається, що швидкодія алгоритму дозволить розкласти числа майже так само швидко, як і виконувати їх множення.

Що це означає?

В курсі криптографії вивчають, що стійкість асиметричних криптоалгоритмів ґрунтується на так званих односторонніх задачах, тобто на таких математичних задачах, коли пряма функція знаходиться значно швидше за обернену. Найвідомішим прикладом такої задачі є факторизація великих цілих чисел, тобто розкладання їх на прості множники. З математичної практики відомо, що задача розкладання великих цілих чисел на прості множники є проблемою надзвичайної обчислювальної складності. На цій задачі побудований криптоалгоритм RSA. Модуль криптоалгоритму утворюється множенням двох великих простих чисел. Цю операцію виконати не складно. А от обчислити приватний ключ, знаючи публічний, можна лише розклавши модуль на

множники. При спробі виконати такі обчислення ми нашоїхуємося на задачу надзвичайної обчислювальної складності.

Якщо квантовий алгоритм дозволить значно прискорити ці обчислення, асиметрична криптографія може припинити своє існування. Більше того, йдеться не тільки про RSA, а й про інші асиметричні алгоритми, які квантовий комп'ютер здатен зламати аналогічним чином.

Відмінністю алгоритму Шора порівняно з класичними алгоритмами факторизації є його ймовірнісний характер, оскільки задача факторизації зводиться до задачі знаходження періоду функції. Необхідно також спостерігати за квантовою пам'яттю, що теж призводить до випадкових результатів.

Припустимо, що нам треба розкласти ціле число N на прості множники. Це означає, що необхідно знайти таке просте число $a=(1 \div N-1)$, на яке N ділиться без залишку.

Для цього виберемо якесь ціле число a та за допомогою алгоритму Евкліда обчислимо найбільший спільний дільник двох чисел НСД (a, N). Якщо він більший за одиницю, то задачу розв'язано, число N розкладено на множники. Якщо ж числа a та N взаємно прості, то за теоремою Ейлера відомо, що існує такий степінь a , який ділиться на N із залишком 1.

Нехай r – найменший такий степінь. Тоді

$$a^r \equiv 1 \pmod{N}. \quad (3.20)$$

Нагадаємо, що r називають порядком a за $\pmod{N}$. Покажемо, що інформація про N може нам надати інформацію про дільник N .

Розглянемо послідовність степенів двійки за модулем 15 тобто $2^x \pmod{15}$. Послідовність буде така:

$$2, 4, 8, 1, 2, 4, 8, 1, \dots$$

Як бачимо, степені двійки за модулем 15 являють собою періодичну послідовність з періодом 4.

Інший приклад: $2^x \pmod{21}$.

$$2, 4, 8, 16, 11, 1, 2, 4, 8, 16, 11, 1, 2, 4, \dots$$

Бачимо, що ми також маємо періодичну послідовність з періодом 6.

Згідно теореми Ейлера, якщо число N є добутком двох простих чисел p та q , тоді якщо a в послідовності (3.20) не ділиться націло ні на p , ні на q , то послідовність буде мати період, який є дільником числа $(p-1)(q-1)$.

Наприклад, якщо $N=15$, то числа $p=3$, $q=5$, $(p-1)(q-1)=2 \times 4=8$. І дійсно, період такої послідовності дорівнює 4, що є дільником 8.

Аналогічно, якщо $N=21$, $p=3$, $q=7$, $(p-1)(q-1) = 2 \times 6 = 12$. В цьому випадку період послідовності дорівнює 6, що також є дільником 12.

Отже, ми бачимо, що знання періоду дійсно надасть нам корисні відомості про $(p-1)(q-1)$.

Тепер з'ясуємо, як ми можемо використати отриману інформацію.

Припустимо, що ми знайшли період r . Нехай він виявився парним. Тоді (3.20) можна записати таким чином:

$$a^x - 1 \equiv 0 \pmod{N}. \quad (3.21)$$

Якщо період парний, то ліву частину (3.21) можна розкласти на множники як різницю квадратів:

$$(a^{r/2} - 1)(a^{r/2} + 1) \equiv 0 \pmod{N}. \quad (3.22)$$

Нехай $A = (a^{r/2} - 1)$ а $B = (a^{r/2} + 1)$. Тоді добуток AB ділиться на N . Якщо ні A , ні B окремо не діляться на N , тоді кожен з них обов'язково має з ним спільний множник. В такому разі знайшовши $\text{НСД}(A, N)$ та $\text{НСД}(B, N)$, за допомогою того ж алгоритму Евкліда, ми знайдемо нетривіальний дільник числа N .

Продемонструємо це на простому прикладі. Розглянемо $N=15$. Оберемо взаємно просте з N число $a=7$.

Обчислимо:

$$7^1 \pmod{15} \equiv 7;$$

$$7^2 \pmod{15} \equiv 14;$$

$$7^3 \pmod{15} \equiv 343 \pmod{15} \equiv 13;$$

$$7^4 \pmod{15} \equiv 2401 \pmod{15} \equiv 1.$$

Отже, ми маємо порядок 7 за $\pmod{15} = 4$. Ми знайшли r . Воно парне. Використаємо формулу (3.22):

$$(7^2 - 1)(7^2 + 1) = 48 \times 50 = 2400.$$

Це число має ділитися на 15 без залишку: $2400 : 15 = 160$. Оскільки ж ні 48, ні 50 окремо не діляться без залишку на 15, то вони повинні мати з ним спільні множники. Очевидно, що $\text{НСД}(15, 48) = 3$, а $\text{НСД}(15, 50) = 5$.

Таким чином, ми розклали число N на прості множники.

Що трапиться, якщо знайдене r буде непарним або взаємно простим з N ? Тоді вибирається інше число a й алгоритм повторюється спочатку.

Тепер нам залишилося з'ясувати, як можна обчислити період послідовності a . Для цього використовують квантове перетворення Фур'є [28].

Перетворення Фур'є у загальному випадку переносить дані з часової області аргументів у частотну. Наприклад, якщо подіяти перетворенням Фур'є на функцію з періодом r , то її буде перетворено у функцію з ненульовими коефіцієнтами там, де значення частоти кратні $2\pi/r$.

Наприклад, нехай ми маємо періодичний процес типу $\exp(2i\pi\omega x)$. Припустимо, що частота цього процесу невідома. Як її визначити?

Уведемо перетворення Фур'є як:

$$F \rightarrow \Phi(\lambda) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-i\lambda x} f(x) dx.$$

Обернене перетворення Фур'є:

$$F^{-1} \rightarrow f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{i\lambda x} \Phi(\lambda) d\lambda.$$

Подамо на вхід цього перетворення наш коливний процес:

$$F : e^{2i\pi\omega x} \rightarrow \Phi(\lambda) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{i(2\pi\omega - \lambda)x} dx.$$

Якщо аргумент λ близький до $2\pi\omega$, то значення інтегралу будуть ненульовими. І навпаки, якщо $2\pi\omega$ і λ далекі, то інтеграл прямує до нуля. Таким чином, після перетворень Фур'є ми чітко можемо побачити, з яких періодичних процесів складається наш коливний процес, які частоти цих коливань та іншу інформацію про нього. Цей факт робить перетворення Фур'є незамінним інструментом для дослідження коливних процесів. Варто сказати, що на основі Фур'є-аналізу працює уся радіотехніка та системи розпізнавання.

Дискретне перетворення Фур'є діє на N рівновіддалених вибірок у напівінтервалі $(0, 2\pi]$ для деякого N , і видає на виході функцію, чия область визначення – цілі числа від 0 до $N-1$. Дискретне перетворення Фур'є функції з періодом r – це функція, зосереджена біля значень N/r . Отже, якщо період r ділить N без залишку, то результатом перетворення

буде функція, ненульові значення якої будуть лише в точках, кратних N/r . В іншому разі результат буде наближеним, і відмінні від нуля числа з'являться в точках, близьких до кратних N/r .

Швидке перетворення Фур'є – різновид дискретного перетворення Фур'є, де N – степінь двійки, тобто $N=2^m$.

Квантове перетворення Фур'є є варіантом дискретного перетворення Фур'є, де також використовуються степені двійки.

Квантове перетворення Фур'є діє не на функцію, а на квантові стани так, що

$$\sum_x g(x)|x\rangle \rightarrow \sum_c G(c)|c\rangle,$$

де $G(c)$ – перетворення функції $g(x)$, x , c – цілі числа від 0 до $N-1$ у двійковому представленні. Якби ми виміряли значення квантового регістра, то ймовірність того, що ми б отримали в результаті c буде $|G(c)|^2$.

Застосовуючи квантове перетворення Фур'є до періодичної функції $g(x)$ з періодом r , ми отримаємо $\sum_c G(c)|c\rangle$ з відмінним від нуля $G(c)$ лише там, де значення c кратні N/r . Отже, коли стан виміряно, то результатом будуть лише значення jN/r . Квантове перетворення Фур'є дає лише наближений результат для періодів, які не є степенями двійки, однак чим більший степінь двійки використаний як база, тим точнішим буде результат.

Квантове перетворення Фур'є можна увести в такий спосіб:

$$\begin{aligned} \text{QFT} : |a\rangle &\rightarrow \frac{1}{\sqrt{N}} \sum_{b=0}^{N-1} e^{\frac{-2\pi iab}{N}} |b\rangle; \\ \text{QFT}^{-1} : |b\rangle &\rightarrow \frac{1}{\sqrt{N}} \sum_{a=0}^{N-1} e^{\frac{2\pi iab}{N}} |a\rangle. \end{aligned}$$

Отже, якщо в квантовому регістрі буде двійкове представлення якоїсь функції з періодом, кратним N , то після перетворення в ньому будуть відмінні від нуля значення лише в точках, кратних N .

Квантове перетворення Фур'є з базою $N=2^m$ визначимо так:

$$\text{QFT} : |x\rangle \rightarrow \frac{1}{\sqrt{2^m}} \sum_{c=0}^{2^m-1} e^{\frac{-2\pi icx}{2^m}} |c\rangle.$$

Тепер розглянемо як використовується квантовий алгоритм Фур'є для знаходження періоду функції $a^x \bmod N$ в алгоритмі Шора.

Для його реалізації потрібно мати два квантових реєстри, X та Y в нульовому стані.

Крок 1. На першому кроці за допомогою оператора Адамара реєстр X переводиться до стану суперпозиції. Квантова система тоді буде знаходитися в стані:

$$|Q\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x, 0\rangle.$$

Крок 2. Нехай ми маємо квантовий оракул, який діє звичним нам чином:

$$U_f : |x, 0\rangle \rightarrow |x, a^x \bmod N\rangle.$$

Подіявши на стан $|Q\rangle$ будемо мати:

$$U_f : |Q\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x, a^x \bmod N\rangle.$$

Це значить, що в другому реєстрі ми отримали значення функції, розраховані для усіх x одночасно. Оскільки функція в другому реєстрі періодична, нам треба залучити квантове перетворення Фур'є для знаходження періоду.

Крок 3. Для цього подіємо на результуючий стан системи оракулом, який реалізує квантове перетворення Фур'є:

$$\text{QFT} : |Q\rangle \rightarrow \frac{1}{N} \sum_{x=0}^{N-1} \sum_{k=0}^{N-1} e^{-2\pi i \frac{kx}{N}} |k, a^x \bmod N\rangle.$$

Крок 4. Зі стандартного Фур'є-аналізу ми знаємо, що експонента буде відмінною від нуля лише там, де k кратне N , тобто в точках jN/k . Вимірювання першого реєстру нам дає стан $|k, a^k \bmod N\rangle$ з ймовірністю

$$\left(\frac{1}{N} \sum_{x: a^x \bmod N = a^k \bmod N} e^{-2\pi i \frac{kx}{N}} \right)^2.$$

Отже, ми знайшли значення k . Решту часу працює класичний комп'ютер.

Крок 5. Нехай при цьому вимірюванні ми отримали в другому регістрі значення v . Якщо період дорівнює $r=2^k$, то ми можемо записати:

$$v = j \frac{2^m}{r}$$

для деякого j . Тоді $\frac{v}{2^m} = \frac{j}{r}$, звідки знаходиться r .

Якщо період не дорівнює степені двійки або він непарний, або алгоритм видає саме число N як його дільник, чи період r та множник j мають спільний множник, то операція знаходження періоду повторюється $O(\lg \lg N)$ разів з тим же самим a . Якщо й цього разу період не знайдений, змінюють a і алгоритм повторюється спочатку.

Пітер Шор довів, що невелика кількість повторень алгоритму знаходить множник числа N з великою ймовірністю.

Значення алгоритму Шора полягає в тому, що з його допомогою, використовуючи квантовий комп'ютер з кількома сотнями логічних кубітів, стає можливим злам криптосистем з публічним ключем. Однією з найпопулярніших двоключових криптосистем є RSA. Відомо (теорема Рабіна [29]), що криптостійкість такої системи дорівнює складності розкладання великого цілого числа (модуль криптосистеми) на прості множники. За допомогою такої факторизації можна, знаючи публічний ключ криптосистеми, обчислити приватний. Найкращий з класичних алгоритмів факторизації потребує часу порядку $N^{1/3}$. Алгоритм Шора, використовуючи можливості квантових комп'ютерів, здатен виконати факторизацію числа не просто за поліноміальний час, а й ненабагато повільніше за множення цілих чисел. Таким чином, реалізація масштабованого квантового комп'ютера зробила би абсолютно безперспективною усю асиметричну криптографію, оскільки він здатен зламати аналогічним чином й інші схожі схеми.

3.7.2. Алгоритм пошуку у невідсортованому масиві (алгоритм Гровера)

Алгоритм був розроблений американським математиком Ловом Гровером у 1996 році [17].

Припустимо, що ми маємо базу даних з N невідсортованих записів, один з яких задовольняє певну умову. Ми хочемо знайти цей особливий запис. Будь-якому класичному алгоритму для знаходження такого запису

потрібно не менше за $O(N)$ кроків. З елементарних міркувань зрозуміло, що якщо ми перевіримо k записів, то знайдемо потрібний з ймовірністю k/N . Ця ймовірність прямує до нуля зі зростанням N , якщо k не того ж порядку, що й N .

Квантовий алгоритм Гровера розв'язує цю задачу з прискоренням у $\sqrt{N}$ разів, тобто за $O(\sqrt{N})$ кроків. Неструктурованість даних дуже важлива, оскільки на сортованому масиві класичний алгоритм дихотомії гарантує швидкість $O(\lg N)$ кроків [17].

Переформулюємо задачу таким чином. Нам дано “чорну скриньку”, яка обчислює потрібну нам функцію від N вхідних даних зі значеннями на виході, що дорівнюють нулю або одиниці. Крім того, відомо, що $f(x)=1$ лише в одному випадку, для деякого вхідного значення x_0 . Наша задача полягає в тому, щоби знайти x_0 .

Нехай N таке, що $2^n \geq N$, U_P – квантовий оракул, який обчислює класичну функцію $P(x)$, що перевіряє істинність твердження (істині відповідає одиниця):

$$U_P : |x\rangle \rightarrow |x, P(x)\rangle.$$

Для того, щоби реалізувати в таких умовах алгоритм Гровера, необхідно виконати такі кроки.

Крок 1 – стандартний для квантових обчислень. Необхідно заповнити регістр нулями, тобто отримати стан $|0\rangle$. Після цього ми діємо на цей квантовий регістр гейтом Адамара:

$$H : |0\rangle \rightarrow \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle.$$

Таким чином ми отримаємо суперпозицію усіх можливих станів x в нашому регістрі.

Крок 2. Діємо квантовим оракулом на отриманий стан квантової системи:

$$U_P : |Q\rangle \rightarrow \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x, P(x)\rangle.$$

Тепер задача полягає в тому, щоби отримати з цієї суперпозиції корисний результат. Очевидно, що тут буде присутній й потрібний нам результат: $|x_0, 1\rangle$. Оскільки амплітуда такого стану буде $2^{-n/2}$, то ніякої

користі поки що немає. Значить, нам треба буде змінити квантовий стан таким чином, щоби амплітуда при потрібному нам результаті була значно більшою. Якщо нам вдасться це зробити, тоді при вимірюванні з великою ймовірністю ми зможемо отримати $|x_0, 1\rangle$.

Крок 3. Для збільшення амплітуди біля одиничних членів алгоритм пропонує такий спосіб. Перше – це зміна знаку біля одиничних членів, друге – інверсія відносно середнього значення.

Розглянемо простий приклад того, як можна змінити знак лише біля одиничних значень функції.

Нехай у нас є квантовий оракул, який виконує перетворення:

$$U_P : |x, b\rangle \rightarrow |x, b \wedge P(x)\rangle.$$

Застосувавши такий оракул до суперпозиції $|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{n-1} |x\rangle$ з $b = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ отримаємо стан, де знак усіх x , при яких $P(x) = 1$ буде змінено, а b залишиться незмінним.

Дійсно,

$$\begin{aligned} |\psi, b\rangle &\rightarrow \frac{1}{\sqrt{2^n}} \sum_{x=0}^{n-1} |x, b \wedge P(x)\rangle = \frac{1}{\sqrt{2^n}} \left(\sum_{x=X_0} |x, 0\rangle + \sum_{x=X_1} |x, 1\rangle \right) \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) = \\ &= \frac{1}{\sqrt{2^{n+1}}} \left(\sum_{x=X_0} |x, 0 \wedge 0\rangle + \sum_{x=X_1} |x, 1 \wedge 0\rangle - \sum_{x=X_0} |x, 0 \wedge 1\rangle - \sum_{x=X_1} |x, 1 \wedge 1\rangle \right) = \\ &= \frac{1}{\sqrt{2^{n+1}}} \left(\sum_{x=X_1} |x, 0\rangle - \sum_{x=X_1} |x, 1\rangle \right). \end{aligned}$$

Тут X_0 – ті стани, де $P(x) = 0$, а X_1 – де $P(x)=1$. Ми бачимо, що знак біля одинички змінився.

Інверсія відносно середнього. Щоби виконати інверсію відносно середнього значення, необхідно побудувати таке перетворення:

$$\sum_{i=0}^{n-1} a_i |x_i\rangle \rightarrow \sum_{i=0}^{n-1} (2A - a_i) |x_i\rangle.$$

Тут A – середнє значення, a_i – коефіцієнти біля x_i , які визначають ймовірність того чи іншого стану системи.

Почергове застосування цих операцій приведе до того, що ймовірність біля одиничних станів зросте порівняно з нульовими, як це показано на рисунку 3.13.

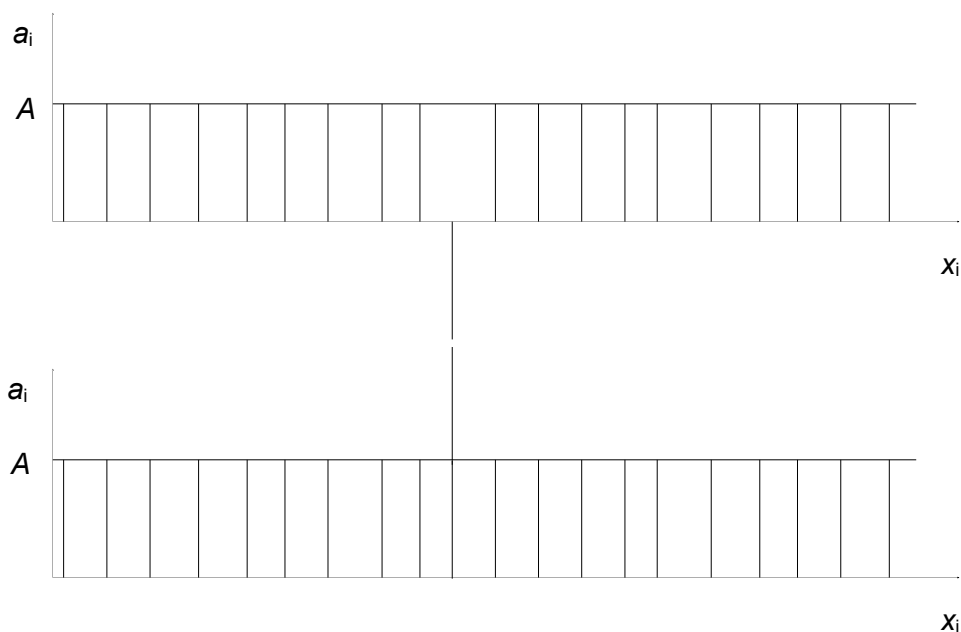


Рис. 3.13. Інверсія відносно середнього

Верхній рисунок демонструє зміну знаку у елементів, де значення функції дорівнює одиниці, а нижній – інверсію відносно середнього значення A .

Лов Гровер довів, що повторення цих кроків $\frac{\pi}{4}\sqrt{2^n}$ разів дозволяє максимізувати ймовірність отримання при вимірюванні одиничного значення.

Цікавим є той факт, що подальше застосування вказаних операцій не тільки не приводить до збільшення ймовірності, але й зменшує її. Цей факт стає зрозумілим, якщо взяти до уваги, що процедури, які виконуються над регістром, є обертанням комплексного простору на певний кут. Отже, спочатку застосування квантового перетворення повертає стан до бажаного, а потім, очевидно, подальші повороти будуть віддаляти вектор від потрібного кута. Таким чином, щоби отримати задовільні результати від застосування цього перетворення, необхідно чітко знати можливу кількість повторень. Для цього існують спеціальні алгоритми, які ми розглядати не будемо.

3.7.3. Квантові протоколи узгодження ключів

Якщо дослідникам та конструкторам вдасться реалізувати універсальний квантовий комп'ютер, який зможе ефективно реалізувати алгоритми Шора та Гровера, то цілі розділи класичної криптографії опиняться під загрозою зникнення. Дійсно, швидкий алгоритм факторизації зробить неефективною асиметричну криптографію, причому це стосується не лише RSA, оскільки з не меншою ефективністю, напевне, можна буде розв'язати й задачі дискретного логарифмування та знаходження точок кратності еліптичних кривих. В той же час, швидкий пошук у невпорядкованому масиві робить тривіальною атаку “грубою силою” на криптографічні ключі симетричних криптосистем.

Таким чином, необхідно шукати альтернативу класичним криптоалгоритмам. Сьогодні одним з новітніх підходів до рішення цієї проблеми є квантова криптографія.

Квантова криптографія — це розділ квантової інформатики, який вивчає методи захисту інформації з використанням квантових носіїв. Можливість такого захисту визначається, перш за все, теоремою про неможливість клонування невідомого квантового стану. Існування цієї теореми привело до розробки протоколу квантової телепортації, вона ж забезпечує захист інформації від прослуховування в квантових каналах зв'язку.

Здається, тоді достатньо встановити такий канал зв'язку між учасниками інформаційного обміну, й можна обмінюватися відкритими повідомленнями без використання криптографічного захисту. Однак, сьогодні це занадто дорого й повільно.

З другого боку, квантовий канал зв'язку можна використати для узгодження криптографічних ключів симетричних алгоритмів, оскільки популярні сьогодні протоколи узгодження ключа на основі асиметричних алгоритмів знаходяться під найбільшою загрозою з боку алгоритму Шора. Це також можливо, але існує алгоритм Гровера, який дозволить досить швидко підібрати ключ до перехопленого, зашифрованого симетричним алгоритмом, повідомлення. Отже симетричні алгоритми також використовувати небажано.

Єдиний варіант, який залишається нам – це потокові шифри, наприклад, шифр Вернама, ключ до якого узгоджується за допомогою квантових протоколів.

Саме так сьогодні й виглядає найпопулярніша схема квантової криптографії [30].

3.7.4. Квантовий протокол узгодження криптографічного ключа BB84

Цей протокол був розроблений Чарлзом Беннеттом та Жілем Brassarом у 1984 році, звідки й походить його назва. Потім він зазнав кількох модифікацій, однак ми розглянемо його першу інтерпретацію, - на чотирьох квантових станах [10].

Нехай ми маємо канал зв'язку, чистий настільки, що він може пропускати окремі фотони без значних втрат. Нехай у нас є також обладнання, яке вміє випромінювати фотони поодиночі. Ці фотони мають довільну, але строго визначену поляризацію.

Назвемо поляризацією електромагнітних хвиль напрямлені коливання векторів електричного та магнітного полів. Будемо вважати, що такою ж поляризацією володіють й окремі фотони, хоча насправді це не зовсім так.

Припустимо, що фотони можуть мати або вертикально-горизонтальну поляризацію, яку будемо називати ортогональною, або діагональну, як це показано на рисунку 3.14.

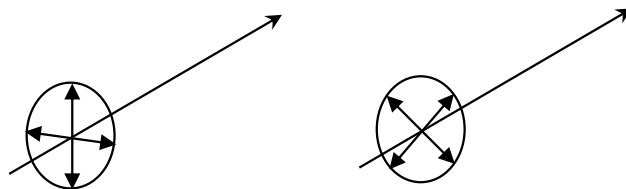


Рис. 3.14. Типи поляризації фотонів: ліворуч – ортогональна поляризація; праворуч – діагональна

Як цього можна досягти, адже лазер випромінює фотони довільної поляризації? Справа в тому, що на виході нашого устаткування стоїть поляризаційний фільтр, який може пропускати фотони лише певної поляризації. Наприклад, якщо фільтр може пропускати лише фотони вертикальної поляризації і затримувати усі горизонтальної, то після нього всі фотони будуть мати тільки вертикальну поляризацію (рис. 3.15).

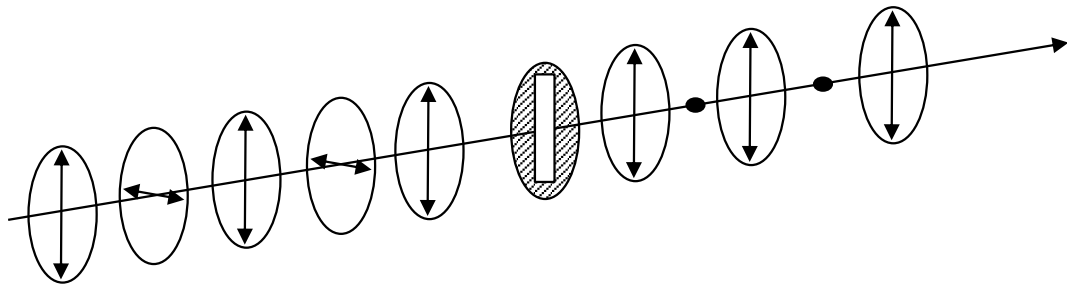


Рис. 3.15. Схема фільтрування потоку фотонів поляризаційним фільтром

Це приведе до зменшення кількості фотонів після фільтру, тобто до зменшення інтенсивності потоку. Так ми платимо за те, що потік фотонів буде чітко поляризований, наприклад, вертикально.

Якщо ж лазер випромінює фотони діагональної поляризації, частину їх також буде затримано, однак тут ми можемо лише говорити про те, з якою ймовірністю той чи інший фотон буде затриманий. Отже, у цьому випадку працює звична для нас квантова схема. Можливі, наприклад, такі варіанти, які показано на рис. 3.16.

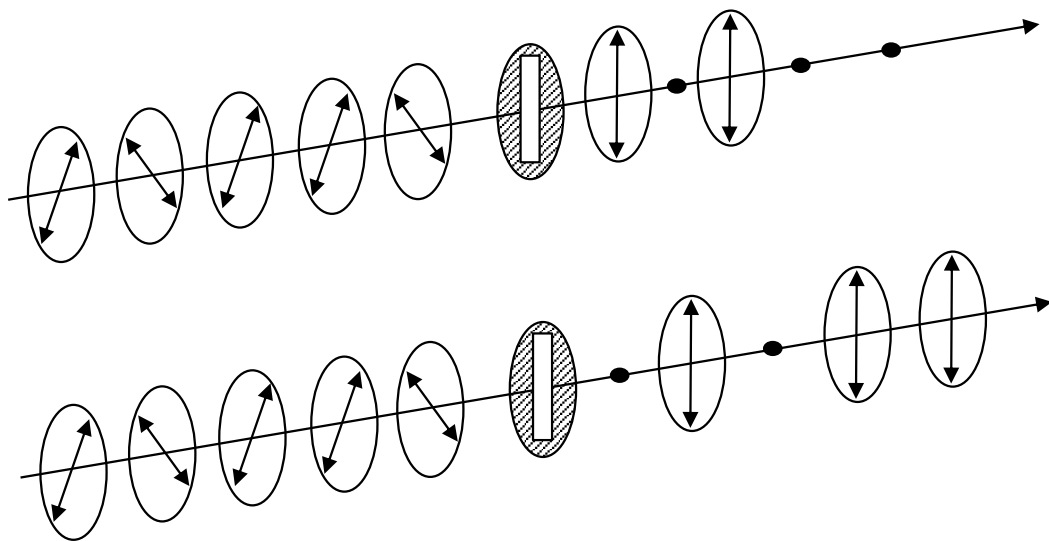


Рис. 3.16. Різні варіанти пропускання фільтром фотонів діагональної поляризації

Одним словом, ми не знаємо, яким чином піде процес, а лише можемо говорити про певну ймовірність того чи іншого стану системи.

Яким же чином цю ситуацію можна використати для узгодження криптографічних ключів?

Припустимо, що абонент А має таку систему, вдосконалену тим, що фільтр може довільним чином змінювати напрям поляризації, вибираючи з чотирьох позицій, які показано на рис. 3.17.

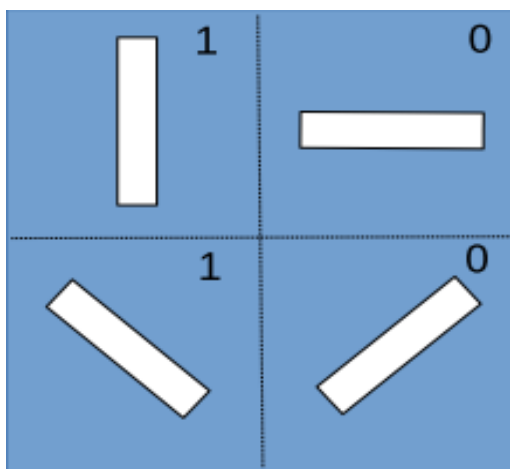


Рис. 3.17. Схема поляризаційного фільтру, використаного у протоколі узгодження ключа

Вибір позиції щоразу випадковий. Будемо приймати за “1” вертикальну поляризацію фотона, а за “0” – горизонтальну. Одиниця та нуль діагональної поляризації показані на рисунку.

Нехай абонент А передає потік одиничних фотонів, весь час змінюючи положення поляризаційного фільтру випадковим чином. На виході ми отримуємо потік фотонів кожної з чотирьох поляризацій. На приймальному боці у абонента Б є приймальна апаратура, яка використовує такий самий фільтр, також орієнтуючи його випадковим чином. Отже, якщо фільтр зорієнтовано горизонтально, а на нього потрапляє вертикально зорієнтований фотон, то його буде затримано, на виході приймальної апаратури не буде нічого. Якщо ж фільтр зорієнтовано діагонально, а фотон поляризований вертикально, то він з певною ймовірністю може бути пропущений. І ми не можемо стверджувати, що станеться напевне.

Розглянемо, як ми можемо узгодити криптографічний ключ за допомогою такої схеми.

Припустимо, що лазер абонента А згенерував послідовність фотонів з такою поляризацією, які показано на рис. 3.18.



Рис. 3.18. Послідовність фотонів з різною поляризацією

Тоді ми отримали такий бітовий рядок: 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1.

Ці та подальші дії, а також отримані результати подано в таблиці 3.9.

Таблиця 3.9.

Результати реалізації схеми поляризаційного фільтру

1	0	1	1	0	1	1	0	0	1	0	1	1	0	0	1
2	X	+	X	+	+	+	+	+	X	X	+	X	X	X	+
3	0	-	1	-	1	1	0	0	-	0	1	1	-	0	1
4	+	X	X	+	+	X	X	+	X	+	X	X	X	X	+
5	1		1		1	0	0	0		1	1	1		0	1
6	+		X		+	X	X	+		+	X	X		X	+
7			√		√			√				√		√	√
8			1		1			0				1		0	1
9					1									0	
10					√									√	
11			1					0				1			1

Записані в таблиці дії слід читати в такий спосіб.

Перші п'ять дій виконуються квантовим каналом зв'язку.

1. В цьому рядочку поданий бітова послідовність, яку згенерував лазер абонента А.

2. Орієнтація (базиси) поляризаційного фільтра, які були обрані абонентом А: X – діагональний; + – ортогональний.

3. Фотони (бітовий рядок), відправлений абонентом А в канал зв'язку.

4. Випадкові базиси, обрані абонентом Б під час приймання сигналу.

5. Результуючий бітовий рядок, отриманий абонентом Б.

Решта дій виконуються відкритим каналом зв'язку.

6. Абонент Б повідомляє А обрані ним базиси.

7. А відзначає, які базиси узгоджуються.

8. Узгоджені біти ключа, які можна використовувати, якщо не було порушень під час узгодження ключа.

Якщо є підозра, що канал прослуховується, виконуються наступні операції:

9. Абонент Б вказує деякі, вибрані випадково, біти ключа.
10. Абонент А підтверджує ці біти.

Результат

11. Секретні біти, які можна використовувати як узгоджений криптографічний ключ.

Коли цей одноразовий ключ буде повністю використано, протокол узгодження нової порції ключа повторюється спочатку.

Цей протокол досить малоефективний, оскільки, як ми бачимо, з переданих 15 бітів в цьому прикладі узгодили лише 4.

3.7.5. Протокол узгодження ключа B92

Протокол BB84 використовує чотири напрями поляризації фотонів. У 1992 році Чарлз Беннетт показав, що аналогічний протокол можна побудувати за допомогою лише двох напрямів поляризації [12]. Протокол було названо B92 за аналогією з BB84.

Він використовує фотони, поляризовані у двох різних напрямках: за "1" приймається поляризація під кутом $+45^\circ$, а за нуль — під кутом 0° .

Процедуру узгодження ключа за цим протоколом схематично подано на рисунку 3.19.

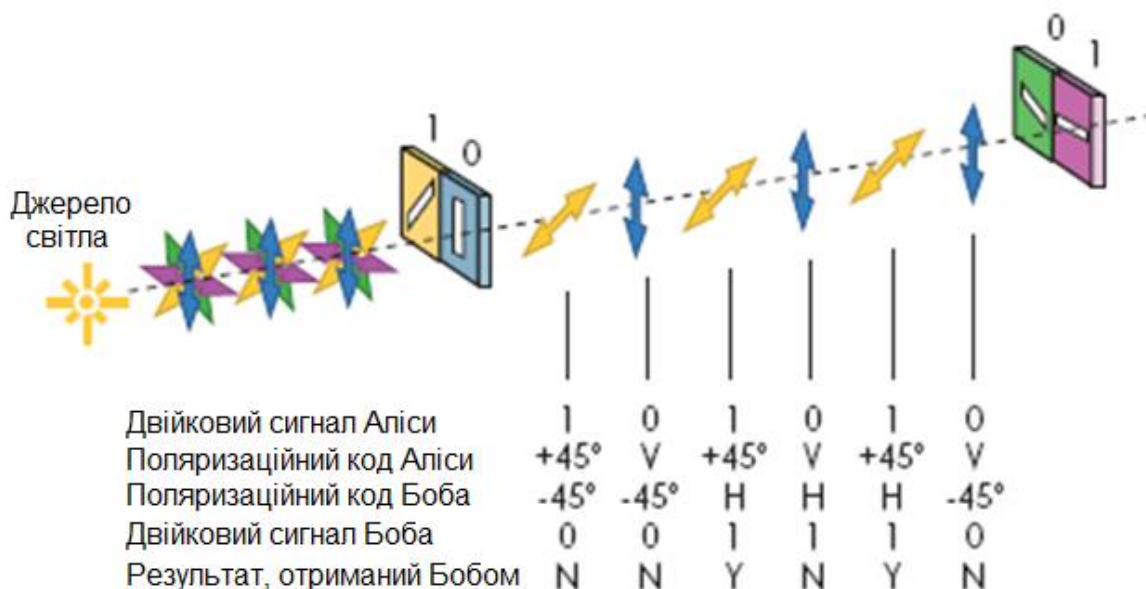


Рис. 3.19. Схема узгодження ключа за протоколом B92

Апаратура абонента А генерує фотони, поляризовані в напрямках 0 та $+45$ градусів. Відповідно, вони представляють нулі та одиниці, причому

поляризація кожного фотона випадкова. Детектувальна апаратура на приймальному боці приймає фотони крізь фільтри, орієнтовані під кутами 90° та -45° градусів. При цьому, якщо фотон аналізується крізь фільтр, орієнтований під кутом 90° відносно його поляризації, то фотон буде затримано. Якщо ж цей кут становить 45° , то фільтр пропустить фотон з ймовірністю 50%.

Розглянемо приклад узгодження бітів ключа за цим протоколом.

Припустимо, що абонент А згенерував таку послідовність фотонів (рис. 3.20).



Рис. 3.20. Послідовність фотонів

Це означає бітовий рядок 1, 0, 1, 0, 1, 0.

На приймальному боці абонент Б застосував до цих фотонів такі фільтри (рис. 3.21).

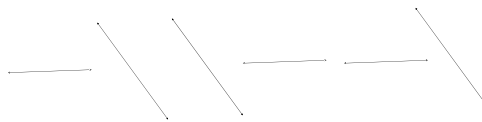


Рис. 3.21. Результат застосування фільтрів

Ми бачимо, що в такому разі тільки для 3-го та 4-го фотонів напрям приймальних фільтрів був перпендикулярний до напрямку поляризації фотонів. Відповідно, ці фотони точно були затримані апаратурою. Отже, ми отримаємо таку послідовність з прийнятих та затриманих фотонів: +, +, -, -, +, +, де “+” – це прийнятий фотон, а “-” – затриманий. Однак, тут не враховано той факт, що в разі кута між поляризацією фільтра і фотона в 45° градусів, крізь фільтр проходять лише 50% фотонів. Нехай і в нашому випадку з чотирьох можливих система пропустила лише два. Тоді послідовність пропущених та затриманих фотонів буде виглядати приблизно так: -, +, -, -, +, -. Залишилося з’ясувати, які саме двійкові значення отримав абонент Б. Напрямок поляризації 2-го фотона на вході був 0° (це “0” за нашими домовленостями). Якщо він пройшов крізь приймальний фільтр, то його поляризація стала -45° , що також відповідає

нулю, як це показано на рисунку 5. Отже значення, яке “приніс” другий фотон – нуль. Аналогічно, передостанній фотон мав напрям поляризації $+45^\circ$, що відповідає одиниці. Якщо ж його пропустила приймальна апаратура, то напрям поляризації став 90° , що відповідає одиниці. Таким чином, ми узгодили два біти ключа: 0, 1.

Для цього протоколу використовується така ж процедура просіювання ключа, як і для BB84.

Протокол B92 ще більш витратний, ніж BB84. Дійсно, ймовірність того, що абонент Б застосує для чергового фотона фільтр з поляризацією в 45° в середньому буде дорівнювати 50%. В свою чергу, ймовірність того, що цей фільтр пропустить фотон також дорівнює 50%. Отже, на приймальному боці зможуть виявити лише 25% відправлених абонентом А фотонів. Тим не менше, в цьому є й позитивний момент: дії зломисника, який буде прослуховувати канал, буде легше виявити, оскільки спотворення інформації будуть більшими.

3.7.6. Атаки на протоколи квантового узгодження ключа

Розглянемо, які дії повинен виконати зломисник для того, щоби перехопити цей ключ, і чи зможе він це зробити.

Можливі варіанти такі:

1. Скопіювати ключ під час передавання неможливо, оскільки ми знаємо, що існує теорема про неможливість клонування невідомого квантового стану. Це саме такий випадок, оскільки зломисник не знає, який стан того чи іншого фотона. Саме це він і прагне з'ясувати. Копіювання змінить стан фотонів, і перевірка абонентів А та Б це виявить. Отже, цей варіант не проходить.

2. Скористатися протоколом квантової телепортації зломисник також не може, оскільки при цьому руйнується стан оригінального фотона.

Тим не менше, припустимо, що зломисникові якимось невідомим для нас чином вдалося виміряти поляризацію кожного фотона, який мандрував каналом зв'язку. Хай навіть так! Однак, він не в змозі дізнатися, яка випадкова послідовність положень фільтра була на приймальному боці у абонента Б. Отже, зломисник спробує вгадувати ці стани, тобто застосує свою послідовність. Звичайна теорія ймовірностей говорить про те, що в такому разі він може вгадати не більше 50% положень фільтра, що значно менше, ніж потрібно. Однак, це ще не всі його проблеми. Якщо він при цьому сформує свій фотон для того, щоби відправити його

абоненту Б, то його поляризація буде неправильною внаслідок редукції хвильової функції. Вимірюючи цей “неправильний” фотон в своєму базисі, абонент Б неодмінно помітить помилку і зрозуміє, що канал прослуховують. Значення похибки на приймальному боці можна оцінити з таких міркувань. Припустимо, зломисник атакував не кожний фотон, а якусь їх частину з ймовірністю p . Тоді 1-р сигналів приходить абоненту Б без змін. Для сигналів, що були атаковані, існує два варіанти розвитку подій: 1) зломисник правильно вгадав базис, а отже правильно передав сигнал абоненту Б, не вносячи ніяких спотворень; 2) базис був вгаданий неправильно, ймовірність чого дорівнює 0,5. Отже з цією ймовірністю результат вимірювань був неправильним, і абсолютно точно він передав “неправильний” фотон абоненту Б. Це призводить до помилки на його боці з такою самою ймовірністю 0,5.

Ймовірність кожного сценарію дорівнює 0,5, а отже легко бачити, що на приймальному боці накопичується 25% неправильних результатів, що занадто багато для якісного каналу. Більше того, теоретичні оцінки для каналів зв'язку, згідно з критерієм Шеннона, зменшують цю ймовірність до 11%. Таким чином, прийнято вважати, що 11% помилок - це межа до якої можливе секретне узгодження ключа.

Отже, будь-які дії зломисника призведуть до того, що абонент Б, проходячи по отриманій послідовності, обов'язково помітить, що у деяких ситуаціях він застосував правильну схему, а чомусь результату (ні “0”, ні “1”) не отримав. Це буде свідчити, що канал зв'язку знаходиться під контролем сторонньої особи. Цей факт буде сигналом для переходу на інший канал і повторення усієї процедури узгодження ключа з самого початку.

Сьогодні на ринку немає однофотонних випромінювачів, тому в апаратурі для квантового узгодження ключа використовують слабкі когерентні імпульси, які випромінюються високоякісними лазерними діодами. Це дає змогу реалізувати так звану атаку розділення фотонів. Якщо зломисник помічає в імпульсі більше одного фотона, він відділяє один з них, дозволяючи решті безперешкодно відправитися до абонента Б. Далі зломисник підмішує цей фотон до свого, утворюючи переплутану пару, і чекає оголошення базисів. Виконуючи потім часткові вимірювання свого фотона в парі, він може точно визначити стан підмішаного фотона, не вносячи жодних змін в ключ.

Якщо ж імпульс складається з одного фотона, то стратегія зломисника може бути різною. Наприклад, він може просто пропускати

одиночні фотони, не втручаючись в хід подій, або заставити їх провзаємодіяти з пробним фотоном і відправити абоненту Б. Проби так само зберігаються до оголошення результатів просіювання ключа.

Таким чином, ми бачимо, що існують деякі типи атак на протокол квантового узгодження ключів, які, скоріше, можна назвати теоретичними. Однак, сьогодні існують й атаки на реалізацію. Одну з таких атак продемонстрував Ларс Лідерсен зі співробітниками [31].

Атака виконується на особливості апаратної частини устаткування.

Оскільки імпульси, якими обмінюються сторони, дуже слабкі, на приймальному боці використовують фотоелектричні помножувачі – лавинні фотодіоди. Ці фотодіоди можуть працювати в двох режимах. Коли на вхід поступає слабкий сигнал, фотодіод підсилює його, відправляючи на вихід електричний сигнал. Коли ж інтенсивність падаючого світла зростає, прилад перестає реагувати на окремі фотони, і працює лише з потужними імпульсами.

Цю особливість помножувачів використала для зламу протоколу й група Л.Лідерсена. Вони повністю перехоплюють потік фотонів від абонента А і генерують потужні імпульси тієї самої поляризації, що й перехоплені фотони, які відправляють на попередньо “засліплений” фотодіод, який вже не здатен реагувати на окремі фотони.

Таку атаку було реалізовано на діючій системі квантового узгодження ключа в Сингапурському університеті, коли за 5 хвилин було перехоплено більше 8 млн. фотонів, залишившись повністю непомітними для учасників інформаційного обміну.

Тим не менше, від такої атаки існує дуже простий захист. Він полягає в тому, що перед приймачем розміщують додаткове джерело одиничних фотонів, яке відправляє їх у випадкові моменти часу. Якщо фотодіод на них не реагує, значить канал прослуховується.

3.7.7. Реалізації протоколу квантового узгодження ключа

Практичні роботи у галузі квантової криптографії виконують такі компанії як IBM, Mitsubishi, Toshiba, лабораторія GAP-Optique, Національна лабораторія в Лос-Аламосі, Каліфорнійський технологічний інститут, MagiQ, холдинг QinetiQ.

Досягнуті успіхи:

– Ніколя Гісен (GAP-Optique) – реалізовано протокол BB84 на відстані 67 км;

- компанія Mitsubishi – 87 км зі швидкістю 7,2 біти за секунду;
- Toshiba Research Europe – 100 км.

Створено також комерційну квантову криптосистему id 3000 Clavis Quantum Key Distribution System, яка підтримує:

- безпечне узгодження ключа на відстані до 100 км;
- підтримку протоколу BB84;
- вбудований протокол просіювання ключа;
- протокол шифрування та передавання файлів;
- бітрейт – 1500 біт за секунду.

Розглянуті протоколи квантового узгодження криптографічного ключа вважаються основними. Існують, тим не менше, багато різних модифікацій, які не принципово відрізняються від оригіналів.

3.7.8. Протокол узгодження ключа E91

Тим не менше, існує ще один дуже цікавий протокол узгодження криптографічного ключа, який ґрунтується на зовсім іншій ідеї. Цей протокол був запропонований Артуром Екертом у 1991 році і називається E91 [32]. Інша назва цього протоколу – ЕПР-протокол, оскільки він використовує ЕПР-пари фотонів. Такі пари фотонів породжуються симетричним атомом. Поляризація фотонів невизначена, однак, оскільки вони переплутані, то їх напрями поляризації строго протилежні.

$$|S_0\rangle = 1/\sqrt{2} (|0\rangle|3\pi/6\rangle + |3\pi/6\rangle|0\rangle);$$

$$|S_1\rangle = 1/\sqrt{2} (|\pi/6\rangle|4\pi/6\rangle + |4\pi/6\rangle|\pi/6\rangle);$$

$$|S_2\rangle = 1/\sqrt{2} (|2\pi/6\rangle|5\pi/6\rangle + |5\pi/6\rangle|2\pi/6\rangle).$$

Якщо у абонента А є подібне обладнання, він зможе реалізувати такий протокол узгодження ключа.

Один фотон зі згенерованої пари абонент залишає у себе, а другий – відправляє на приймальний бік абоненту В. Обидва абоненти не знають, яка поляризація їхніх фотонів. Більше того, вони зможуть це з'ясувати лише після вимірювання.

Отже, згенерувавши певну кількість переплутаних фотонів, учасники інформаційного обміну можуть починати вимірювання. Для отримання ідентичних послідовностей абонент А записує істинні значення, а абонент В – їх доповнення до одиниці.

Така ідея цього протоколу. Звичайно, тут також існують процедури просіювання ключа.

Реалізація такого протоколу значно складніша за розглянуті раніше, оскільки сьогодні не існує джерел переплутаних пар фотонів з високим ступенем кореляції та великим часом життя.

3.7.9. Квантові гроші Стівена Візнера

У кінці 60-х рр. XX сторіччя аспірант Колумбійського університету Стівен Візнер (син директора MIT, радника багатьох президентів США з науки Джерома Візнера) запропонував своєму керівникові ідею квантового захисту грошей. Науковий керівник не звернув на цю ідею жодної уваги. С.Візнер відправив статтю в журнал.

Редакція відхилила статтю, визнавши її антинауковою.

Лише 1983 року стаття «Conjugate Coding» (журнал «ACM SIGACT News») вийшла в друці та була визнана однією з основних у галузі квантового захисту інформації [6].

Основна перевага ідеї полягає в тому, що захист побудовано не на складності підробки, а на принциповій неможливості цього.

Ідея полягала в наступному. Припустимо, ми маємо такі квантові пастки, в яких можемо утримувати окремі фотони необмежений час.

Тоді ми можемо реалізувати такий.

На кожній купюрі розміщено деяку кількість таких пасток, які заповнюються поляризованими фотонами під час випуску купюр.

Банк зберігає дані про поляризацію кожного фотона на кожній купюрі у своїй базі даних. Схематично С.Візнер уявляв це собі у вигляді, що представлений на рисунку 3.4.

Детальніше підхід, запропонований С.Візнером, пояснено на рисунку 3.22.

Система захисту банкноти складається зі 100 пасток, де знаходяться квантові частинки, кожна з яких має багато квантових атрибутів, які можна вимірювати у трьох напрямках.

Серійний номер банкноти пов'язаний з квантовою конфігурацією цих частинок, і зберігається у базі даних емітента валюти. Для перевірки істинності банкноти банк шукає в базі даних номер банкноти і отримує квантову конфігурацію кожної зі ста частинок, що знаходяться у пастках цієї банкноти.

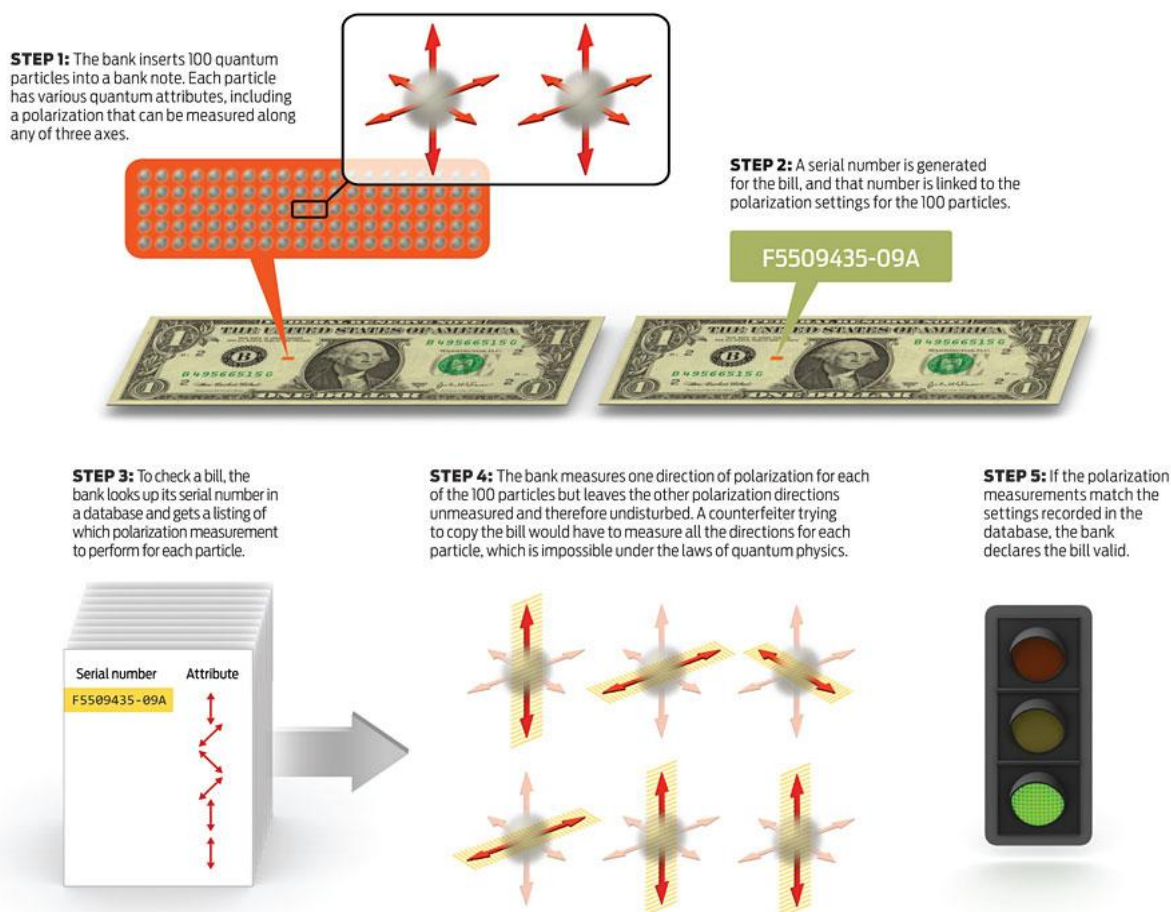


Рис. 3.22. Захист грошей, запропонований С.Віснером

Після цього банк випадковим чином вибирає один з напрямів, в якому виконує часткові вимірювання квантових характеристик кожної частинки, не зачіпаючи решти напрямів. Якщо результати вимірів збігаються з даними, записаними в базі даних банку, він вважає, що банкнота справжня. Зловмиснику для копіювання банкноти необхідно виконати вимірювання в усіх трьох напрямках, що заборонено принципами квантової механіки.

Звичайно, недоліком такої схеми є ніким не контрольовані права банку на необмежений випуск банкнот.

Співробітники Массачусетського технологічного інституту запропонували модифікацію ідеї С.Візнера: банк генерує кілька станів квантових частинок та змішує їх при заповненні пасток. У цьому випадку банк не знає, який отримується результат. При цьому банк публікує “публічний ключ” – алгоритм, за допомогою якого кожне відділення може перевірити справжність купюри, але не зможе відтворити цей стан квантових частинок.

Стверджується, що такий підхід позбавляє банк-емітент монополізму на випуск купюр.

Довгий час ідея С.Візнера була лише теоретичною іграшкою, оскільки не існувало квантових пасток, які могли би утримувати частинки хоча би макроскопічний час, не кажучи вже про необмежений.

Однак, у 2012 році Серж Арош (Франція) та Девід Вайнленд (США) отримали Нобелівську премію з фізики за розроблені методики маніпулювання квантовими частинками, які могли утримуватися в пастках протягом однієї секунди, а це вже макроскопічний час The Nobel Prize in Physics 2012. Проте, це відбувається за дуже низьких температур.

Інший варіант пасток пропонується науковцями Центру квантової фізики Гарвардського університету, які можуть утримати квантовий стан атому азоту при кімнатній температурі в матриці надчистих кристалів кремнію.

Таким чином, ідеї Стівена Візнера сформували фундамент нового напрямку науки – *квантової інформаційної безпеки*.

Суть цього напрямку – використання фундаментальних квантових властивостей мікросвіту для захисту інформації.

Основна перевага – теоретична стійкість квантових протоколів та способів захисту.

Основний недолік – складність практичної реалізації.

3.8. Архітектура та основні вимоги до квантових комп'ютерів

Будь-який комп'ютер повинен мати як мінімум два стани: основний та неосновний.

Для класичного ОЗП цими станами можуть бути:

- Відсутність/наявність струму (або малий його рівень);
- Відсутність/наявність напруги (або малий її рівень);
- Відсутність/наявність заряду тощо.

Для квантового комп'ютера основним станом є $|0\rangle$, а неосновним – $|1\rangle$. Окрім регістру пам'яті повинен бути пристрій обробки інформації, яка в ньому зберігається.

У випадку квантових комп'ютерів повинні бути:

- гейти для виконання однокубітних операцій (I, NOT, гейт Адамара);

- гейти для виконання багатокубітних операцій (CNOT, гейт Тоффолі тощо);
- пристрої для виконання інших квантових операцій (квантові оракули).

Необхідно передбачити також систему вимірювання результатів обчислень. Також необхідні системи зв'язку та передавання даних (шини, канали тощо).

Виходячи з наведеного, можна зобразити архітектуру квантового комп'ютера таким чином (рис. 3.23).



Рис. 3.23. Архітектура квантового комп'ютера

Квантова система не може безмежно перебувати у неосновному стані. Під дією зовнішніх факторів:

- температурного фону;
- електромагнітних полів (у тому числі електромагнітного поля Землі);
- взаємодії з іншими кубітами;
- самовільно (спонтанно)

вони повертаються до основного стану. Стан суперпозиції зникає. Таке явище отримало назву *декогерентизації*. Отже, необхідно регулярно виконувати операцію регенерування інформації у квантовому реєстрі пам'яті.

Сукупність усіх можливих операцій, що формують вихідний стан, виконують обчислення, усувають декогерентизацію, виправляють помилки, виконують у квантовому комп'ютері роль програмного забезпечення (*software*).

Обираючи ту чи іншу схему квантового комп'ютера, необхідно розв'язати інше завдання:

- яку фізичну систему обрати для створення квантового регістру пам'яті;
- який фізичний механізм буде відповідати взаємодію між кубітами, що необхідно для реалізації багатокубітних операцій;
- визначити спосіб селективного керування кубітами та вимірювання їх стану на виході.

Усе це утворює аналог апаратного забезпечення квантового комп'ютера (*hardware*).

Сьогодні вважається, що для реалізації повноцінного квантового комп'ютера необхідно забезпечити виконання таких вимог:

- фізична система квантового регістру повинна складатися з великої кількості кубітів (1024 та більше);
- необхідно забезпечити умови для ініціалізації, тобто для організації основного стану $|0\rangle$.
- необхідно забезпечити максимально можливе усунення декогерентизації. Час декогерентизації повинно як мінімум у 10^4 разів перевищувати такт. Для цього система кубітів повинна бути ретельно ізольованою від оточуючого середовища.

Необхідно також забезпечити виконання за час такту необхідної сукупності унарних операцій, яка може складатися з багатьох одно- та багатокубітних операцій та забезпечити найвищу надійність вимірювань, оскільки “другої спроби” виконати їх вже не буде. Це є однією з головних проблем квантового комп'ютингу.

Отже, необхідно забезпечити ефективне керування квантовим обчислювачем з боку класичного керуючого комп'ютера.

Основні фізичні технології

Твердотільні квантові точки у напівпровідниках. У якості кубітів використовують або зарядові стани (наявність/відсутність електрона у квантовій ямі) або напрям його спіну. Керування здійснюється або зовнішнім потенціалом, або лазерним імпульсом.

Надпровідникові елементи (NEC Research Laboratories) – джозефсонівські переходи, сквіди тощо. Як кубіт використовують

наявність/відсутність куперівської пари у певній точці простору. Керують зовнішнім потенціалом або магнітним полем.

Переваги цього способу: високий ступінь інтеграції; значно вищі температури (високотемпературні надпровідники); порівняно легке керування кубітами.

Недоліки - недостатня стабільність станів; високий ступінь декогерентизації.

Іони у вакуумних пастках Пауля або оптичних пастках (NIST, Los Alamos National Laboratories):

- кубіти – основний та збудований стан зовнішнього електрона цього іона;

- керування – лазерними імпульсами.

Переваги – просте керування кубітами; недоліки – наднизькі температури; жорсткі обмеження на кількість іонів у пастці.

Використання методів ядерного магнітного резонансу – ЯМР (MIT, LANL, Оксфорд).

Кубіти – атоми рідин з непарними ядерними спінами; взаємодія – скалярна взаємодія між атомами, керування – методами ЯМР.

Переваги цього способу – просте, добре відлагоджене керування методами ЯМР, якими користуються вже багато років у магніто-резонансному устаткуванні, кімнатні температури та великий час декогерентизації (секунди та більше). Недоліки – складність ініціалізації, обмежена кількість кубітів, невелика швидкодія.

Атоми азоту у ґратці алмазу (Гарвардський університет). Тут як кубіти використовують магнітні моменти атомів азоту в ґратці суперчистого кремнію. Керування здійснюють електромагнітними імпульсами.

Переваги такого методу – високий ступінь інтеграції, оскільки кремнієві технології добре налагоджені, сьогодні досягнуто великого часу декогерентизації (секунди). Та основною перевагою є кімнатні температури. Ця технологія не позбавлена й недоліків. До них можна віднести невідпрацьовані технології та поки що невелика кількість кубітів у реєстрі.

Розглянемо детальніше перелічені варіанти фізичної реалізації квантових комп'ютерів у наступних розділах.

3.9. Огляд фізичних реалізацій квантових комп'ютерів

Цей розділ присвячено деталізації тих ідей, про які було згадано у попередньому параграфі, з аналізом переваг та недоліків кожної технології. Матеріал викладається за книгами [34] та [35].

Раніше ми обговорювали вимоги до квантових комп'ютерів та фізичні принципи, якими повинні керуватися розробники для їх фізичної реалізації. Ми повинні не лише знайти вдалу фізичну реалізацію для кубітів, а й знайти керовану систему кубітів. Звісно, необхідно обрати метод організації первісного стану та вимірювальну підсистему.

Технічно задовольнити такі вимоги надзвичайно складно. Часто квантові стани в системі добре розрізняються але існують вони дуже короткий час внаслідок впливу зовнішнього середовища, і необхідно витратити значні ресурси для їх підтримки. Іноді навпаки: квантовий стан може існувати макроскопічний час, але виміряти його дуже складно. Водночас кубіти мають бути легко доступними для того, щоби виконувати квантові обчислення та вимірювати результати. Такі, часто протилежні вимоги, спонукають змінити питання з “чи можна побудувати квантовий комп'ютер” на “наскільки якісним його можна побудувати”.

Для того, щоби оцінити перспективність тієї чи іншої фізичної “платформи” для побудови квантового комп'ютера, необхідно $T_{оп}$.

Таблиця 3.9.

Результати порівняння часу існування когерентного стану з часом, за який виконується квантова операція

Система	T_d , сек.	$T_{оп}$, сек.	$n_{оп}=T_d/T_{оп}$
Спін ядра	$10^{-2} - 10^8$	$10^{-3} - 10^{-6}$	$10^5 — 10^{14}$
Спін електрона	10^{-3}	10^{-7}	10^4
Йонна пастка	10^{-1}	10^{-14}	10^{13}
Електрон — Au	10^{-8}	10^{-14}	10^6
Електрон - GaAs	10^{-10}	10^{-13}	10^3
Квантові точки	10^{-6}	10^{-9}	10^3
Оптичний резонатор	10^{-5}	10^{-14}	10^9
НВЧ-резонатор	10^0	10^{-4}	10^4

З таблиці ми бачимо, що усі зазначені технології перспективні з точки зору організації квантових обчислень, оскільки час декогерентизації значно більший за час виконання квантової операції, а отже кількість операцій, $n_{\text{оп}} = T_{\text{д}} / T_{\text{оп}}$, що їх можна виконати за час існування когерентного стану, досить велика.

Значну проблему також являє можливість приготування первісного (вхідного) стану квантового регістра пам'яті. Навіть якщо система виконує обчислення абсолютно точно, це буде малоефективним в разі неможливості контролювати вхідний стан. Ми повинні надійно вміти приготувати хоча би один початковий стан, оскільки інші можна отримати з нього за допомогою унарних операцій.

Для більшості фізичних систем приготування початкового стану часто є складною проблемою. Наприклад, йони у пастці могли би приводитися у початковий стан охолодженням, але реалізувати це непросто.

Для фізичних систем, що реалізують ансамблі квантових комп'ютерів, виникають додаткові складнощі. Наприклад, у випадку ЯМР-реалізації кожна молекула повинна розглядатися як окремий квантовий комп'ютер. Щоби отримати сигнал, придатний для вимірювання, необхідно багато молекул, причому в одному й тому ж квантовому стані. Це непросто, оскільки енергія між $|0\rangle$ та $|1\rangle$ набагато менша за kT .

З другого боку, з початкового стану, у більшості випадків, необхідно отримати стан квантової суперпозиції, інакше зміст стартового стану втрачається, оскільки він є незмінним.

Іншою проблемою є вимірювання результатів квантових обчислень. Зручно розглядати вимірювання як взаємодію квантової системи з деякою класичною. Ця взаємодія відбувається протягом деякого часу, після чого стан класичної системи вказує нам на результати вимірювання. Наприклад, вимірювання стану кубіта $a|0\rangle + b|1\rangle$, який подано основним та збудженим станами дворівневої атомної системи, може бути подано спостереженням флуоресценції. Тоді ми будемо впевнені, що кубіт був у стані $|1\rangle$. Якщо ж флуоресценції не спостерігалось, – в стані $|0\rangle$. Однак, тут злий жарт з дослідниками можуть зіграти шуми вимірювань та недосконалість апаратної частини, в результаті чого ми можемо неправильно оцінити стан кубіта. Вимірювання також не повинні виконуватися, коли їх не потрібно; це призведе до втрати когерентності системи.

Вимірювання можуть бути “сильними”, коли взаємодія з вимірювальною системою сильно впливає на квантову систему, і “слабкими”, коли як квантова система використовуються ансамблі частинок. Тоді сигнал від такої системи є макроскопічною величиною і несе інформацію про її квантовий стан.

Якість вимірювання можна оцінити за відношенням сигналу до шуму. Воно дає інформацію про силу сигналу та взаємодію вимірювального пристрою із квантовою системою.

3.9.1. Гармонічний осцилятор як модель квантового комп'ютера

Спочатку давайте розглянемо дуже просту квантову систему: гармонічний осцилятор та з'ясуємо, чому вона не може бути вдалою моделлю квантового комп'ютера. Підходи цього прикладу будемо вважати основою для наступних фізичних систем.

Гамільтоніан системи

Прикладом простого гармонічного осцилятора є частинка у параболічній потенціальній ямі. У класичній механіці це може бути вантаж на пружині; в електриці – коливальний контур з ємністю та індуктивністю.

У квантовій механіці, яка починає працювати, коли зв'язок системи з оточуванням середовищем дуже малий, повна енергія системи може набувати дискретних значень, на відміну від класичних випадків.

Для простого осцилятора власні стани з певною енергією позначаються як $|n\rangle$, де $n=0,1,2, \dots$. Час життя кубітів визначається фізичними параметрами системи. Для того, щоби застосувати унарний оператор, необхідно надати системі можливість еволюціонувати протягом певного часу при певних умовах. Така схема, однак, має цілий ряд недоліків, які ми обговоримо пізніше. Почнемо з опису гамільтоніана системи та перейдемо до простих квантових логічних операцій.

Гамільтоніан частинки у одновимірному параболічному потенціалі має вигляд:

$$H = \frac{p^2}{2m} + \frac{1}{2}m\omega^2x^2,$$

де p – імпульс частинки; m – маса; x – координата та ω – параметр потенціалу. Графічно це представлено на рисунку 3.24.

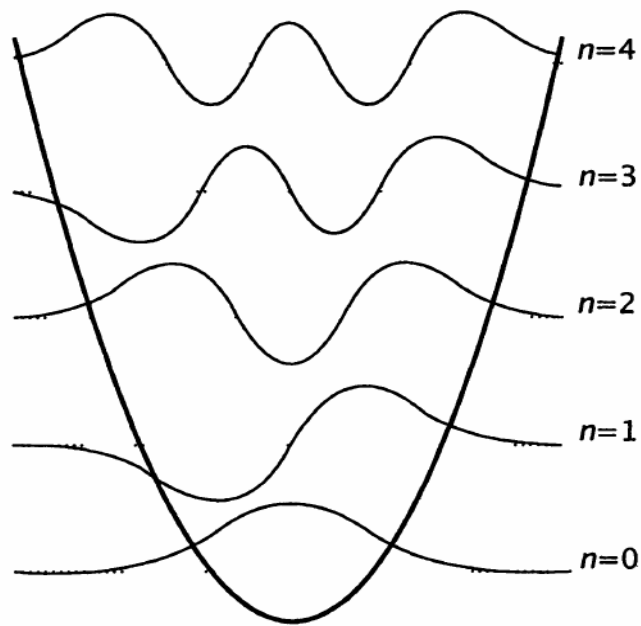


Рис. 3.24. Гамільтоніан частинки у одновимірному параболічному потенціалі

Зображена функція описує амплітуду ймовірності того, що частинку буде виявлено у тій чи іншій точці в області дії потенціалу.

Стани системи еволюціонують з часом згідно рівняння Шредінгера. Ми також будемо вважати, що можемо точно приготувати довільний початковий стан осцилятора, виконувати проєктивні виміри, але взаємодія осцилятора з оточуваним середовищем відсутня.

Квантові обчислення

Спробуємо використати простий гармонічний осцилятор для квантових обчислень. Кубіти в такому разі подаються власними станами енергії $|n\rangle$. Як тоді реалізувати оператор CNOT? Нагадаємо, що дія цієї операції на двокубітові базисні стани має вигляд:

$$\begin{aligned} |00\rangle_L &\rightarrow |00\rangle_L \\ |01\rangle_L &\rightarrow |01\rangle_L \\ |10\rangle_L &\rightarrow |11\rangle_L \\ |11\rangle_L &\rightarrow |10\rangle_L \end{aligned}$$

Індекс L говорить про те, що це є логічними станами. Ми можемо закодувати ці два кубіти наступним чином:

$$\begin{aligned} |00\rangle_L &= |0\rangle \\ |01\rangle_L &= |2\rangle \\ |10\rangle_L &= (|4\rangle + |1\rangle) / \sqrt{2} \\ |11\rangle_L &= (|4\rangle - |1\rangle) / \sqrt{2}. \end{aligned}$$

Нехай в момент часу $t=0$ стан системи є лінійною комбінацією векторів $|0\rangle$, $|1\rangle$, $|2\rangle$, $|4\rangle$, а система еволюціонує на проміжку часу $t=\pi/\hbar\omega$. При цьому власні стани системи перетворюються за законом: $|n\rangle = (-1)^n |n\rangle$. Тоді $|0\rangle = (-1)^0 |0\rangle = |0\rangle$; $|1\rangle = (-1)^1 |1\rangle = -|1\rangle$; $|2\rangle = (-1)^2 |2\rangle = |2\rangle$; $|4\rangle = (-1)^4 |4\rangle = |4\rangle$, звідки видно, що стани $|0\rangle$, $|2\rangle$, $|4\rangle$ лишаються незмінними, а стан $|1\rangle$ – змінює знак. У результаті для логічних станів отримаємо потрібне нам перетворення CNOT.

Отже, бачимо, що у такий спосіб можна реалізувати практично довільний набір квантових операцій з довільним числом кубітів. Може здатися, звичайно, що за допомогою такої квантової системи можна реалізувати квантовий комп'ютер. Однак, не все так просто!

Недоліки

Очевидними недоліками запропонованої схеми є те, що кубіти подаються як рівні енергії багаторівневої квантової системи. Це означає, що: а) це аналоговий спосіб подання інформації, тоді як усі квантові алгоритми оперують із електроним поданням; б) таке подання інформації (енергетичними рівнями та їх комбінаціями) не може бути застосоване для довільного квантового оператора, оскільки часто їх спектр невідомий.

3.9.2. Квантовий комп'ютер на фотонах

Однією з найцікавіших фізичних систем для представлення квантового біта є оптичний фотон. Фотони є нейтральними частинками, які досить слабо взаємодіють один з одним та оточуваним середовищем. Їх можна практично без втрат переносити на великі відстані за допомогою оптичного кабелю, змінювати фазу за допомогою оптичних пристроїв та створювати стан суперпозиції за допомогою розділювачів. Фотони демонструють яскраві квантові ефекти, наприклад, інтерференцію на двох щілинах. Крім того, у нелінійних середовищах може спостерігатися фотон-фотонна взаємодія, яке виникає внаслідок нелінійності взаємодії фотонів з речовиною. Проаналізуємо фотонну модель квантового комп'ютера, її архітектуру та деталі.

Фізична апаратура

Одним з експериментальних способів генерування одиничних фотонів є зменшення інтенсивності лазерного випромінювання. Лазер випромінює так зване когерентне випромінювання, імпульс якого містить n фотонів. У результаті зменшення інтенсивності пакет зменшує середню енергію, що дає змогу з великою ймовірністю отримати однофотонний стан.

Існує також достатньо багато способів детектування одиничних фотонів у широкому спектральному діапазоні з великою квантовою ефективністю, наприклад, за допомогою фотопомножувачів. На практиці на ці процеси сильно впливає недосконалість детектора: час відгуку, смуга пропускання, шумові та темнові характеристики.

Експериментальна техніка для керування станами фотонів складається з трьох важливих компонентів: дзеркал, фазообертача та розділювача світла. На сьогодні дзеркала, у яких втрачається лише 0,01% фотонів є цілком доступними. Фазообертач являє собою просто прозору пластинку з показником заломлення, відмінним від одиниці. Наприклад, звичайне боро-силікатне скло має показник заломлення приблизно $n=1,5$. Зрозуміло, що при проходженні фотона крізь пластинку товщини L , його фаза буде змінюватися на $\exp(ikL)$, де $k=n\omega/c_0$, c_0 – швидкість світла у вакуумі.

Розділювач світла являє собою скло, вкрите тонким шаром срібла, з коефіцієнтом відбивання R та коефіцієнтом пропускання $1-R$.

Квантові обчислення

Зручно описувати розділювач світла за допомогою кута θ , уведеного як $\cos(\theta)=R$. У такому разі цей кут характеризує ступінь відбиття та, у загальному випадку, не має жодного стосунку з геометрією розділювача.

Якщо ми маємо два входи та два виходи розділювача, інтенсивність на них пов'язана такими співвідношеннями:

$$\begin{aligned}a_{\text{вих}} &= a_{\text{вх}} \cos \theta + b_{\text{вх}} \sin \theta, \\ b_{\text{вих}} &= -a_{\text{вх}} \sin \theta + b_{\text{вх}} \cos \theta,\end{aligned}$$

де a та b – класичні амплітуди електромагнітного поля двох променів. Якщо ж нам треба отримати розділювач 50/50, маємо $\theta=45^\circ$.

Кубіти найчастіше представляються одиничними фотонами, які можуть перебувати на двох модах: $|01\rangle$ та $|10\rangle$. Зміну квантового стану реалізують за допомогою ефекту Керра [36], коли у речовині під дією зовнішнього постійного або змінного електричного поля спостерігається

подвійне променезаломлення внаслідок зміни показника заломлення. Ефект Керра зумовлений, головним чином, переполяризацією середовища внаслідок деформації електронних орбіталей атомів та молекул, або їхньої переорієнтації. Ефект Керра дуже швидкий, оскільки у твердих тілах може відбутися лише деформація електронної хмари атомів.

Зрозуміло, що й тут є свої недоліки, головний з яких — поглинання та розсіювання світла у різних оптичних структурах, внаслідок чого ми не можемо застосувати оптичні канали будь-якої довжини.

Основні недоліки

Представлення кубіти за допомогою одиничних фотонів дуже привабливе, оскільки їх порівняно легко генерувати та детектувати. У двомодовому представленні можна реалізувати довільний однокубітовий квантовий оператор. На жаль, забезпечити взаємодію між фотонами значно складніше: взаємодія, навіть у найкращих керрівських середовищах, значно слабша, щоби забезпечити фазову модуляцію порядку π між однофотонними станами. Крім того, оскільки нелінійність, як правило, відбувається поблизу оптичного резонансу, ефект Керра практично завжди буде супроводжуватися поглинанням світла. Теоретичні оцінки вказують на те, що на один фотон, який пройшов крізь середовище, припадає приблизно 50 поглинутих фотонів. Це означає, що у рамках сучасної нелінійної оптики побудова квантового комп'ютера має дуже мало шансів на успіх.

Тим не менше, обговорення квантового однофотонного комп'ютера надало нам знання про те, як би міг виглядати реальний лабораторний квантовий комп'ютер.

Взагалі кажучи, оптичні комп'ютери давно розглядалися як заміна електронним класичним комп'ютерам. Розроблялися експериментальні зразки оптичних процесорів та комп'ютерів у цілому [37]. Однак, пов'язані з ними надії, врешті рещт, не справдилися, оскільки поки що не вдається отримати матеріали з високим ступенем нелінійності, а також тому, що складнощі з налаштуванням та енерговитрати не компенсують переваги у швидкості та можливості паралельних обчислень.

Не дивлячись на вказані недоліки оптичної реалізації квантового комп'ютера, формальна теорія, що його описує, є фундаментом для решти реалізацій. Далі ми розглянемо дещо інший варіант оптичного квантового комп'ютера, де використовується інше нелінійне середовище.

3.9.3. Квантовий комп'ютер на оптичних резонаторах

Тут розглядається задача про взаємодію окремих атомів з невеликою кількістю оптичних фотонів. Така взаємодія може бути реалізована в оптичних резонаторах з великою добротністю. Якщо в резонаторі існує всього одна-дві оптичних моди, а напруженість електричного поля достатньо велика, то взаємодія цих оптичних мод з дипольним моментом атома стає істотним. Відповідно, можливі процеси, коли фотон встигає багато разів вступити у взаємодію з атомом раніше, чим він вийде з резонатора. Така експериментальна техніка знаменита тим, що дозволяє вивчати окремі квантові системи та маніпулювати ними.

Зокрема, використання такої техніки дозволяє подолати основну перешкоду до реалізації оптичного квантового комп'ютера, тобто слабку взаємодію фотонів між собою.

Фізична апаратура

Реальна апаратура для такого роду процесів складається з двох компонентів: електромагнітний резонатор та атом. Процеси відбуваються у сильному електричному полі, тому зручно використовувати резонатор Фабрі-Перо [38]. Основною частиною резонатора Фабрі-Перо є напівпрозоре дзеркало, вкрите шаром срібла, на якому відбуваються багаторазові процеси відбивання та проходження променів світла. В результаті цих процесів промінь набуває додаткової фази, яка залежить від довжини шляху, що його він пройшов. У резонаторі фотони (одиночні фотони) взаємодіють з одиничними (спеціально підібраними) атомами речовини, і змінюють при цьому свою фазу. Оскільки електричне поле досить сильне, то тут можна досягти значної взаємодії між однофотонними станами. Це робить описану техніку досить перспективною для створення оптичного квантового комп'ютера.

Подання квантової інформації

Тут можливі дві схеми подання квантової інформації: різними станами фотонів, а самий резонатор з атомами використовується для створення фотон-фотонної взаємодії; різними станами атомів, а власне фотони служать для передавання інформації між атомами. У першому випадку, так само, як і у попередній схемі однофотонного оптичного комп'ютера, як кубіт використовується один фотон, який може перебувати на двох модах $|01\rangle$, $|10\rangle$ або з різними поляризаціями.

Довільний квантовий оператор можна реалізувати за допомогою фазообертачів, світло розділювачів та резонатора Фабрі-Перо, всередині

якого утримуються кілька атомів, що взаємодіють з фотонними модами резонатора.

Початковий стан системи можна забезпечити, як і в попередньому випадку, генеруванням однофотонних станів шляхом, наприклад, зменшення інтенсивності лазерного променя.

Вимірювання остаточного результату досягається застосуванням фотопомножувачів для реєстрації одиничних фотонів.

Недоліки

Оскільки фотон-фотонна взаємодія зумовлена проміжною взаємодією з атомом, бажано збільшити ступінь зв'язку атома з електричним полем. Однак, в такому разі, зменшується час існування стану когерентності атомів, що, у свою чергу, обмежує час виконання квантових обчислень.

3.9.4. Йони у пастках

У цьому випадку до інших способів представлення кубіта, - атомних чи ядерних станів, наприклад, електронного та ядерного спінів. І хоча спіні — це досить дивне (хоча й реальне!) поняття, причому різниця енергій різних спінових станів, як правило, значно менша за решти характерних енергій (наприклад, кінетичної енергії атома при кімнатній температурі), однак, існує спеціальна експериментальна техніка, що дозволяє контролювати спінові стани. Для цього невелика кількість ізольованих йонізованих атомів утримується в електромагнітній пастці, після чого вони охолоджуються до такої температури, коли кінетична енергія стає меншою за спінові енергії. Вмикаючи зовнішнє резонансне електромагнітне поле, можна селективно впливати на кожну пару станів. Це й становить основу квантових обчислень методом йонів у пастці.

Фізична апаратура

Фізичне приладдя для такого роду експериментів складається з двох частин: електромагнітної пастки, обладнану лазерами та власне йонів. Основна частина, електромагнітна пастка, складається з чотирьох циліндричних електродів, як це подано на рисунку 3.25.

Ці електроди утворюють в центрі пастки електричний потенціал, що утримує чотири йони, зображені чорними кульками. Пастку, як правило, розміщують у глибокому вакуумі (порядку 10^{-8} Па), а йони постачаються з розміщеного поруч джерела.

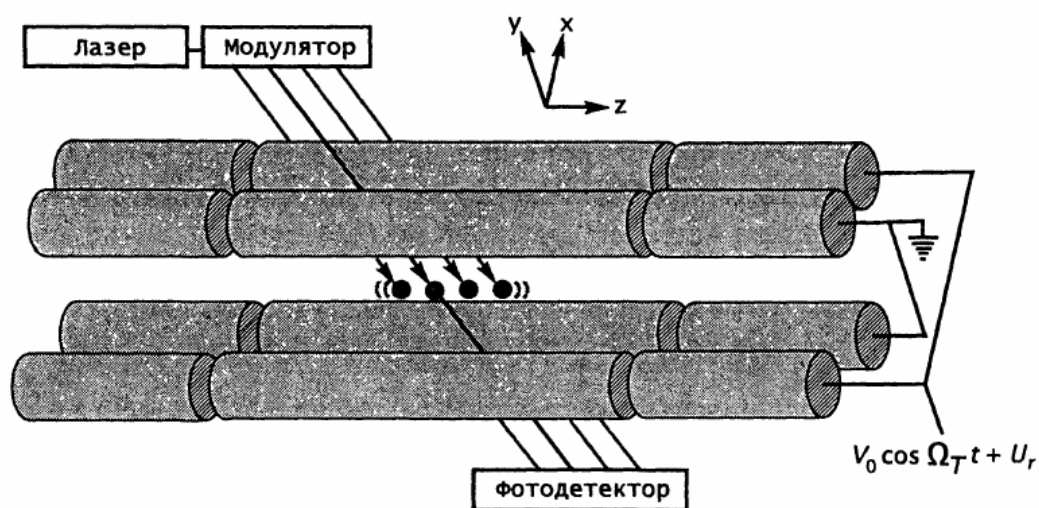


Рис. 3.25. Електромагнітна пастка

Модульоване лазерне випромінювання потрапляє на йони крізь вікно вакуумної камери. Впливаючи ним на атомні стани, ми можемо виконувати квантові обчислення та читати результати.

Зрозуміло, що тут розглянуто дуже простий випадок усього чотирьох йонів у пастці. Зі збільшенням кількості йонів геометрія стає складнішою, утворюючи різні структури. Якщо така система добре ізольована й теплові флуктуації не руйнують стан системи, рух йонів починає квантуватися. Крім того, флуктуації електричного та магнітного полів, можуть викликати хаотичні рухи йонів, які ми назвемо шумом. Такий шум практично неминучий у цих системах, а його збільшення також призводить до руйнування квантового стану. Тим не менше, більшість завад може бути зменшено до такої величини, коли їх впливом за час експерименту можна знехтувати.

Для зменшення амплітуди коливань атомів використовують спеціальну техніку зниження їхньої температури, яка використовує ефект Доплера. Як ми знаємо, фотон володіє енергією та імпульсом. Подібно тому, як автомобільний сигнал від автомобіля, що наближається до спостерігача, має вищу частоту, ніж у автомобіля, що віддаляється від нього, атом, що рухається “до лазера”, має частоту переходу трохи вищу, ніж той, що рухається “від лазера”. Якщо підібрати частоту лазера так, щоби випромінювання поглиналося лише атомами, що наближаються, то атоми почнуть сповільнюватися, оскільки імпульс фотонів, що поглинаються, спрямований назустріч руху атома. Цей метод отримав назву доплерівського охолодження. Він дозволяє охолодити атоми до дуже низьких температур.

Подання квантових станів

Внутрішній стан йонів характеризується електронним спіном та ядерним, які в сумі утворюють повний спін атома. Внутрішній стан атома зручно описувати базовими квантовими станами повного кутового моменту атома.

Як приклад, розглянемо два спіни $1/2$. У цьому випадку як базові стани можна взяти $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$. Але, як правило, використовують інші базові стани, оскільки пов'язують їх з повним спіном атома:

$$\begin{aligned}|0,0\rangle_J &= \sqrt{2}(|01\rangle - |10\rangle), \\ |1,0\rangle_J &= \sqrt{2}(|01\rangle + |10\rangle), \\ |1,1\rangle_J &= |11\rangle, \\ |1,-1\rangle_J &= |00\rangle.\end{aligned}$$

Як довго може існувати суперпозиція різних спінових станів? Її час життя обмежується процесом спонтанного випромінювання, коли атом переходить зі збудженого стану до основного, випромінюючи фотон. Акт випромінювання відбувається у випадковий момент часу. На перший погляд може здатися дивним, що атом, який у вільному просторі без зовнішнього впливу може випромінити фотон. Насправді атом не перебуває у вільному просторі, оскільки на нього діє зовнішнє електричне поле. І саме взаємодія з ним спонукає атом до спонтанного випромінювання фотона.

Квантові обчислення

Зміну стану такої квантової системи можна ініціювати лазерним випромінюванням певної частоти. На рисунку 3.26 видно, що перехід атома зі стану $|10\rangle$ до стану $|11\rangle$ відбувається лазерним імпульсом з частотою ω_0 , а зі стану $|00\rangle$ до $|01\rangle$ – з частотою ω_z тощо. Про обернені переходи можна судити за частотою фотона, що випромінюється атомом.

Взаємодія між кубітами відбувається засобами спільного фононного стану. Вимірювання кінцевого результату виконується вимірюванням заселеності квантових станів такої структури, що виконується за допомогою індукованої флуоресценції.

Недоліки

Основним недоліком такої системи є малий час існування збуджених станів атому і, відповідно, важким є приготування основного коливного стану системи.

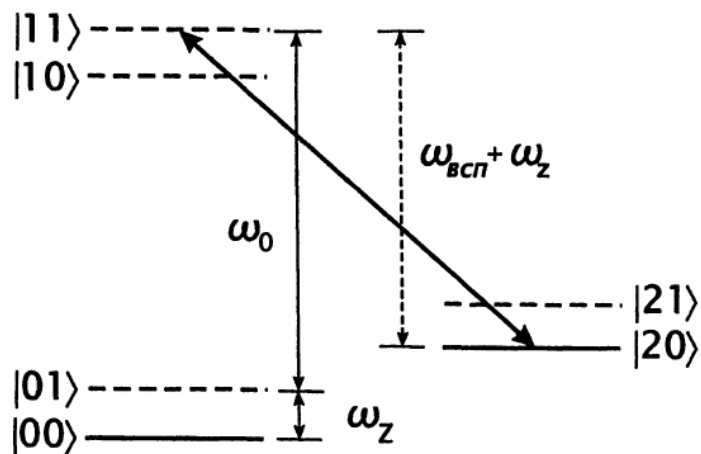


Рис. 3.26. Стани квантової системи

3.9.5. Ядерний магнітний резонанс

Ми бачили, ядерні спіни були б ідеальною системою для реалізації квантових обчислень за умови, що спін-спінова взаємодія була би досить сильною та контрольованою. Принциповий недолік використання йонів у пастках був якраз у слабкості спін-спінової взаємодії, а також у швидкій втраті когерентності. Можна було би розташувати у пастці не атом а молекулу, де ця взаємодія значно сильніша, однак внаслідок наявності у спектрі молекули коливних мод, утримувати її у пастці та охолодити значно складніше.

З іншого боку, існує добре відпрацьований метод, що дозволяє маніпулювати ядерними спінами та вимірювати їх за допомогою радіочастотних імпульсів. Це – ядерний магнітний резонанс (ЯМР). Цей метод широко застосовується у різних галузях науки і техніки, і дозволяє виконувати експерименти з контролю десятків, сотень і навіть тисяч ядер.

Однак, при використанні ЯМР для квантових обчислень виникають дві проблеми. По-перше, внаслідок малості величини ядерного магнітного моменту, необхідно використовувати зразки з великою кількістю (порядку 10^8 й більше) молекул. Це, у свою чергу, ставить питання, чи можемо ми використовувати результат усереднення по квантових системах? По-друге, ЯМР використовують при кімнатних температурах. Це значить, що початковий стан системи близький до неупорядкованого. Постає питання: чи можна виконувати квантові обчислення над системою у змішаному початковому стані та має велику ентропію?

Фізична апаратура

Імпульсне ЯМР-приладдя для рідких зразків складається зі спектрометру та власне зразку. Молекули, у яких спостерігається ЯМР, як правило, містять деяку кількість протонів зі спіном $1/2$ (наприклад, ^{13}C , ^{19}F , ^{15}N , ^{31}P). У магнітному полі порядку 12 Тл протони мають частоту резонансу приблизно 500 МГц. Резонансні частоти для різних ядер певної молекули можуть відрізнятися на величину до кількох сотень кілогерц. Молекули ці знаходяться у розчині, так що їх концентрація дуже мала, тож їхньою взаємодією можна знехтувати. Тому систему таких молекул можна описати як ансамбль багатокубітових квантових обчислювачів.

ЯМР-спектрометр принципово складається з радіочастотних котушок та великого надпровідного магніту, де як сердечник розміщено зразок у скляній пробірці, як це показано на рисунку. Постійне магнітне поле B_0 , напрямлене перпендикулярно до зразку, налаштовується так, що у межах зразку його неоднорідність не перевищує 10^{-9} . Розміщені перпендикулярно до магніту котушки Гельмгольца формують слабе змінне магнітне поле у площині зразка. Це поле можна швидко вмикати та вимикати, що дозволяє маніпулювати ядерними спінами. Ті ж котушки використовують також для приймання радіочастотного сигналу від прецесії ядерних спінів подібно до того, як магніт, що обертається, створює змінний індукційний струм у котушці (рис. 3.27).

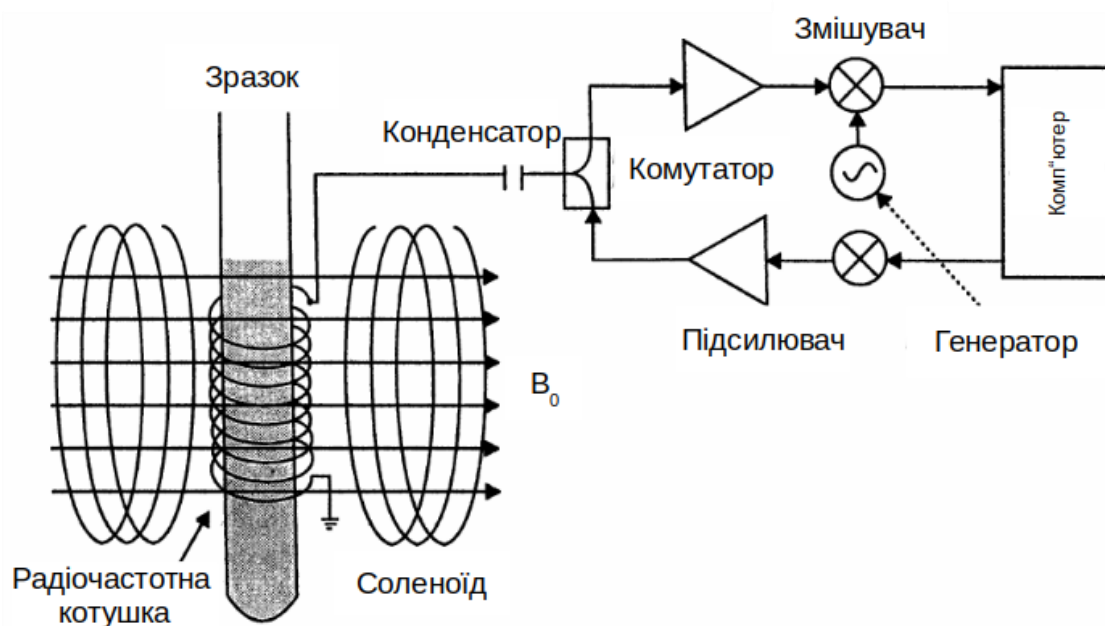


Рис. 3.27. ЯМР-спектрометр

Типовий експеримент починається з етапу очікування для того, щоби ядра досягли стану термодинамічної рівноваги. Цей час може складати кілька хвилин. Після чого класичний комп'ютер створює послідовність радіочастотних імпульсів, які виконують потрібні перетворення над станами ядер. Далі потужні підсилювачі, які формували імпульси, вимикаються і вимірюється кінцевий стан спінів за допомогою високочутливого підсилювача. До отриманого сигналу застосовується перетворення Фур'є, за результатами якого, обчислюючи площу піків у спектрі частот, можна визначити поточний стан ядерних спінів.

Як приклад, на рисунку 3.28 подано вуглецевий спектр ЯМР трихлоретилену (^{13}C).

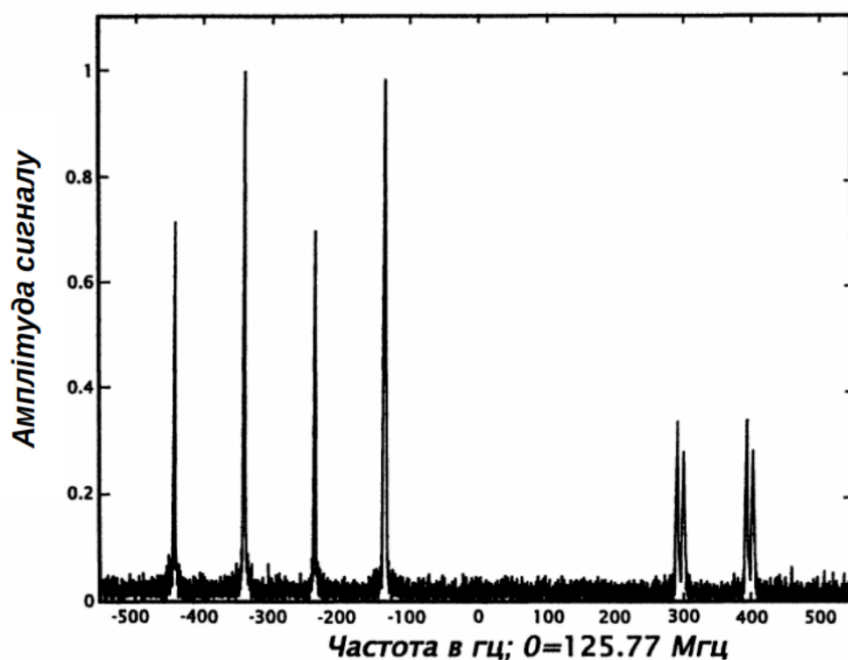


Рис. 3.28. Вуглецевий спектр ЯМР трихлоретилену

Чотири лінії ліворуч належать ядру вуглецю, яке є сусідом протона. Їх чотири внаслідок спарювання з протоном та другим ядром вуглецю. Чотири лінії праворуч – сигнал від другого ядра вуглецю. Воно знаходиться далі від протону, ніж перше, тому спарювання слабше.

Квантові обчислення

Основною проблемою виконання квантових обчислень за допомогою ЯМР є те, що ми тут працюємо з ансамблем ядер. Тому результатом квантових обчислень буде якесь середнє значення, тоді як нам треба отримати результат строго одного квантового обчислювача. І тут ми стикаємося з наступною проблемою, яку простіше усього

продемонструвати на прикладі алгоритму Шора. У цьому випадку квантовий алгоритм повинен дати число c/r , де c – невідоме ціле число, а r – цілий показник ступеня, який нас і цікавить. Далі, використовуючи класичний алгоритм розкладання у ланцюговий дріб, можна з великою ймовірністю визначити число r , після чого можна визначити, чи є r дільником, а якщо ні, - виконуємо весь алгоритм спочатку. На жаль, у випадку вимірювань на ансамблі, ми можемо отримати лиш середнє значення $\langle c/r \rangle$, з якого жодної інформації про r отримати неможливо.

Виходом з такої ситуації є наступне. Необхідно включити усі класичні обчислення, які завершують роботу алгоритму, до його квантової частини. Нехай кожен квантовий обчислювач, що входить до ансамблю (тобто кожна молекула), виконує розкладання до ланцюгового дробу, знаходить r та перевіряє, чи є він дільником. Далі, можна модифікувати алгоритм так, що внесок до підсумкового сигналу нададуть лише ті молекули, де обчислення дільника пройшло успішно. Тепер ми зможемо взяти середнє за ансамблем як остаточний результат, вимірюючи сигнал індукції, який виникає при прецесії магнітного моменту ядер.

Недоліки

Основним недоліком квантових систем на основі ЯМР у випадку, коли поляризація спінів у початковому стані не дуже висока, є те, що вихідний сигнал зменшується експоненційно зі збільшенням кількості кубітів.

Як видно з наведених принципів реалізувати квантовий комп'ютер – надзвичайно нетривіальне завдання, оскільки необхідно врахувати багато майже протилежних вимог. Усі запропоновані варіанти реалізації є незадовільними в тому розумінні, що на їх основі у найближчому майбутньому не можна побудувати багатокубітовий квантовий комп'ютер. Тим не менше, це ще не означає що дана задача не має вирішення, оскільки існує багато інших варіантів реалізації.

З таблиці на початку розділу зрозуміло, що практично будь-яка фізична величина, що квантується, може представляти кубіти. В той же час, фундаментальні фізичні квантові об'єкти, такі як фотон та спін, є найбільш привабливими представленнями кубіта.

Існує ще одна фундаментальна величина, придатна для ролі кубіта. Це електричний заряд. Сучасна електроніка надає нам можливість створювати, контролювати та вимірювати заряд навіть в одноелектронному режимі. Наприклад, у квантових точках, виготовлених з напівпровідників, металів, чи навіть, з невеликих молекул, можуть

локалізуватися заряди у тривимірних потенціальних ямах. На відміну від фотонів, заряди не можуть народжуватися та знищуватися, вони можуть лише мігрувати по системі. Таким чином, у представленні кубіта зарядовим станом ми маємо використовувати, наприклад, локалізації в одній з двох квантових точок чи двом станам електрона в одній точці.

Однокубітові операції можна реалізувати за допомогою електростатичні затвори, спеціальні двоканальні розсіювачі, тунельні контакти між квантовими точками. Використовуючи кулонівською взаємодією, можна реалізувати операції над кубітами. Як вимірювачі, можуть використовуватися, наприклад, одноелектронні транзистори. На жаль, неконтрольований рух віддалених зарядів призводить, внаслідок їхньої взаємодії з системою, до втрати когерентності. Разом з електрон-фононою взаємодією це зменшує час когерентності зарядових станів відносно малим: від сотень пікосекунд до сотень фемтосекунд.

Для представлення кубітів також пропонувалося використовувати носії заряду у надпровідниках, які об'єднуються у так звані куперівські пари. Їх також можна локалізувати у квантових точках. Однокубітові операції реалізуються за допомогою електростатичних зарядів, що впливають на пару, та джозефсонівських контактів, що можуть з'єднувати локалізації різних пар. Їх також можна використовувати для реалізації взаємодії між кубітами, причому силу цієї взаємодії можна змінювати зовнішнім магнітним полем за допомогою надпровідних квантових інтерферометрів. Вимірювання кубітів в такому разі – це просто вимірювання електричного заряду. Кубіти, що представлені куперівськими парами, відносно стійкі; оцінки показують, що час когерентності може перевищувати одну мікросекунду. На жаль, зовнішній електростатичний та електромагнітний фон обмежують час існування когерентності і у цьому випадку.

Короткодіюча магніто-дипольна взаємодія спонукає нас звернути увагу на представлення кубітів спінами у напівпровідникових системах. Наприклад, квантова точка, яка містить багато електронів, може поводити себе як спін $1/2$, якщо кількість електронів непарна. Відповідний стан можна приготувати, якщо розмістити квантову точку у сильному магнітному полі при низьких температурах. Маніпулювати спінами можна імпульсними магнітними полями. Вимірювати спінові стани можна, наприклад, надаючи електронам можливості тунелювати у сусідню “вимірювальну” квантову точку. Однак, сучасні технології поки що не дозволяють вимірювати спін у напівпровідниках з потрібною точністю.

Аналогічні “конструкції” можна запропонувати для ядерних спінів, а не лише для електронних.

Отже, найбільш перспективними з технологічної точки зору, вважаються сьогодні схеми, що ґрунтуються на твердотільних системах.

Можливо, будуть запропоновані й зовсім інакші, за принципом дії, платформи для реалізації квантового комп’ютера, однак це – напевне, справа майбутнього.

Бажаючи дізнатися більше про квантові комп’ютери та закони квантових обчислень, можуть додатково скористатися книжками [39, 40].

Контрольні запитання до теми 2

1. Опишіть уявні досліди, які призвели до неможливості пояснення поведінки мікрочастинок з точки зору класичних уявлень. Які висновки з дослідів можна зробити?
2. Поняття про кубіти. Опишіть просту квантову систему.
3. Опишіть основні прості операції над кубітами.
4. Опишіть відомі Вам багатокубітні операції. Як можна їх реалізувати для побудови логічних операцій?
5. Поняття про переплутані стани. Квантовий паралелізм та його переваги для квантової інформатики.
6. Вплив вимірювань на квантову систему. Що Ви розумієте під частковими вимірюваннями? Як вони впливають на стан квантової системи?
7. Теорема про неможливість копіювання невідомого квантового стану.
8. Опишіть протокол квантової телепортації та обґрунтуйте його необхідність.
9. Опишіть протокол квантового щільного кодування.
10. Парадокс Ейнштейна-Подольські-Розена та способи його інтерпретації. Що можна з'ясувати стосовно ЕПР-парадоксу з Нобелівської премії з фізики 2022 року?
11. Опишіть алгоритм Дойча-Джози. До яких висновків призвів цей алгоритм?
12. Опишіть алгоритм Шора. До яких висновків стосовно класичних асиметричних криптосистем призвела його розробка?
13. Опишіть алгоритм Гровера. Які висновки зробив Національний Інститут стандартів і технологій США стосовно постквантової криптографії з урахуванням цього алгоритму?
14. Опишіть квантовий протокол узгодження ключа BB84.
15. Опишіть квантові протоколи узгодження ключа B92 та E91. Чим вони

відрізняються від BB84?

16. Які Ви знаєте атаки на квантові протоколи узгодження ключа.

Опишіть їхню

класифікацію.

17. Що Ви знаєте про квантові гроші С.Віснера? Коротко опишіть, яким чином можна

організувати захист цінних паперів на основі алгоритму С.Віснера?

18. Опишіть архітектуру та вимоги до квантових комп'ютерів.

Використана література до теми 3

1. Закон Мура. Вікіпедія. Вільна енциклопедія. - https://en.wikipedia.org/wiki/Moore%27s_law (дата звернення: 21.11.2022).
2. Закон Амдала. Вікіпедія. Вільна енциклопедія. - https://en.wikipedia.org/wiki/Amdahl%27s_law (дата звернення: 21.11.2022).
3. Planck, M. // *Verhandl. Dtsch. phys. Ges.*, 1900. -2. - 202-210.
4. L. de Broglie. *Ondes et quanta* // *Comptes rendus de l'Académie des sciences*, 1923. -177. - P.507-510.
5. Simon B., *Functional Integration and Quantum Physics* // Academic Press, 1979. - 546.
6. Wiesner S. *Conjugate coding* // *SIGACT News*, 1983. -Vol. 15, No. 1. - P.78-88.
7. Feynman, R.P. *There's plenty of room at the bottom (data storage)* // *Journal of Microelectromechanical Systems*, 1992. -vol. 1, no. 1. - P. 60—66.
8. Toffoli T. *Reversible computing* // *International Colloquium on Automata, Languages, and Programming*, 1980. - P.632-644.
9. Wootters W., Zurek W. H. *A Single Quantum Cannot be Cloned* // *Nature*, 1982. -299. - P.802-803.
10. Bennett C. H., Brassard G. *Quantum Cryptography: Public Key Distribution and Coin Tossing* // *Proceedings of International Conference on computers, Systems & Signal Processing*, Dec. 9-12, 1984, Bangalore, India, 1984. - P.175.
11. Bennett C. and Wiesner S. *Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states* // *Phys. Rev. Lett*, 1992. -69. - P.2881.
12. Bennett C. *Quantum Cryptography Using Any Two Nonorthogonal States* // *Physical Review Letters*, 1992. -25.05. - P.3122.
13. Wolf R. de, *Quantum Computing: Lecture notes* // *arXiv:1907.09415*, 2019. - 176 P.
14. Bennett C. H., Brassard G., Crépeau C., Jozsa R., Peres A., Wootters W. *Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels* // *Phys. Rev. Lett*, 1993. -V.70, N.13. - P.1895-1899.
15. Shor P. W. *Algorithms for quantum computation: discrete logarithms and factoring* // *Foundations of Computer Science: Conference Publications*, 1994. - P.124-134.

16. Report on Post-Quantum Cryptography. NIST 9105 Draft Report. - https://csrc.nist.gov/csrc/media/publications/nistir/8105/final/documents/nistir_8105_draft.pdf (дата звернення: 21.11.2022).
17. Grover L.K. A fast quantum mechanical algorithm for database search // Proceedings, 28th Annual ACM Symposium on the Theory of Computing, 1996. - P.212.
18. Quantum Computing: How D-Wave System Works. D-Wave: The Quantum Computing Company. - <https://www.dwavesys.com/quantum-computing> (дата звернення: 21.11.2022).
19. Фейнман Р., Лейтон Р., Сэндс М., Фейнмановские лекции по физике. Т.1-9. // М.: "Мир", 1976.
20. Einstein A., Podolsky B., Rosen N. Einstein A., Podolsky B., Rosen N. Can Quantum-Mechanical Description of Physical Reality Be Considered Complete? // Phys. Rev., 1935. - V.47, N.10. - P.777-780.
21. Bohm D., Quantum Theory // New York: Prentice Hall, 1951. - 700 P.
22. Bell J.S. On the Einstein Podolsky Rosen Paradox // Phys. Phys. Fiz., 1964. - V.1, N.3. - P. 195-200.
23. Bouwmeester D., Pan J-W., Mattle K., Eibl M., Weinfurter H. & Zeilinger A. Experimental quantum teleportation // Nature, 1997. - 390. - P. 575-579.
24. D. Boschi, S. Branca, F. De Martini, L. Hardy, and S. Popescu. Experimental Realization of Teleporting an Unknown Pure Quantum State via Dual Classical and Einstein-Podolsky-Rosen Channels // Phys. Rev. Lett., 1998. - 80. - 1121.
25. Jacob F. Sherson, Hanna Krauter, Rasmus K. Olsson, Brian Julsgaard, Klemens Hammerer, Ignacio Cirac and Eugene S. Polzik. Quantum teleportation between light and matter // Nature, 2006. - 443. - P. 557-560.
26. Xian Min-Jin et al. Experimental free-space quantum teleportation // Nature Photonics, 2010. - V. 4. - PP. 376-381.
27. Satellite-based photon entanglement distributed over 1,200 kilometers. EurekAlert!. - https://www.eurekalert.org/pub_releases/2017-06/uosa-spe061217.php (дата звернення: 21.11.2022).
28. The classical and quantum Fourier transform. Домашня сторінка Р. де Вольфа. - <https://homepages.cwi.nl/~rdewolf/qfourierintro.pdf> (дата звернення: 21.11.2022).
29. Rabin M.O. Probabilistic Algorithm for Testing Primality // JOURNAL OF NUMBER THEORY, 1980. - 12. - PP.128-138.

30. Gisin N., Ribordy G., Tittel W., Zbinden N. Quantum Cryptography // Rev. Mod. Phys., 2002. -V.74, N.1. - PP. 145-191.
31. Lydersen L. et al. Hacking commercial quantum cryptography systems by tailored bright illumination // Nature Photonics, 2010. -4. - PP.686-689.
32. Ekert, A. K. Quantum Cryptography Based on Bell's Theorem // Physical Review Letters, 1991. -67. - PP.661-663.
33. The Nobel Prize in Physics 2012. Веб-сайт Нобелівського комітету. - <https://www.nobelprize.org/prizes/physics/2012/summary/> (дата звернення: 21.11.2022).
34. Нильсен М., Чанг И., Квантовые вычисления и квантовая информация//М.:Мир, 2006. - 824 С.
35. Баумейстер Д., Экрет А., Цайлингер А., Физика квантовой информации//М.: Постмаркет, 2002. - 376 С.
36. Эффект Керра. Вікіпедія. Відкрита енциклопедія. - https://en.wikipedia.org/wiki/Kerr_effect (дата звернення: 21.11.2022).
37. Оптичний комп'ютер. Вікіпедія. Вільна енциклопедія. - https://en.wikipedia.org/wiki/Optical_computing (дата звернення: 21.11.2022).
38. Fabry–Pérot interferometer. Вікіпедія: вільна енциклопедія. - https://en.wikipedia.org/wiki/Fabry%E2%80%93P%C3%A9rot_interferometer (дата звернення: 21.11.2022).
39. Химено-Сеговіа М., Херриган Н., Джонстон Э., Программирование квантовых компьютеров // O'Reilly, 2021. - 336 с.
40. Крохмальський Т. Вступ до квантових обчислень // Львів: ЛНУ, 2018. - 204 с.