Міністерство освіти і науки України
Запорізький національний університет

Є. М. Кісельов
С. Л. Шмалій

ЗАСОБИ ЗАХИСТУ В МЕРЕЖАХ ЕЛЕКТРОННО – ОБЧИСЛЮВАЛЬНИХ МАШИН

конспект лекцій для здобувачів ступеня вищої освіти магістра спеціальності 171 «Електроніка» освітньої програми «Електроніка»

Затверджено
вченою радою ЗНУ
Протокол № 1 від 05.09.2024 р.

Запоріжжя
2024

УДК 621.38

Кісельов Є. М., Шмалій С. Л. Засоби захисту в мережах електронно – обчислювальних машин : конспект лекцій для здобувачів ступеня вищої освіти магістра спеціальності 171 «Електроніка» освітньої програми «Електроніка». Запоріжжя : ЗНУ, 2024. 118с.

Конспект лекцій розроблено для студентів денної та заочної форми навчання спеціальності 171 «Електроніка». Розглянуто основні методи обробки інформації, існуючі технології захисту інформації; відомості з створення, впровадження і супроводження систем кібербезпеки.

Рецензент
В.Л. Коваленко, доктор технічних наук, професор кафедри електричної інженерії та кіберфізичних систем

Відповідальний за випуск
Т.В. Критська, доктор технічних наук, професор кафедри електроніки, інформаційних систем та програмного забезпечення

ЗМІСТ

Вступ 5
1 Основні види і джерела атак на інформацію 6
1.1 Сучасна ситуація в області інформаційної безпеки 6
1.2. Категорії інформаційної безпеки 11
1.3. Термінали захищеної інформаційної системи 13
1.4 Етапи побудови комплексної системи захисту КСОІ 17
1.5. Соціальна психологія і інші способи отримання паролів 18
2 Криптографія 21
2.1 Класифікація криптоалгоритмів 21
2.2 Скремблери 22
2.3 Блокові шифри 25
2.3.1 Загальні відомості про блокові шифри 25
2.3.2 Мережі Фейштеля 29
2.3.3 Блоковий шифр TEA 33
2.3.4 Стандарт блокових шифрів AES 34
2.3.4.1 Загальні відомості про конкурс AES 34
2.3.4.2 Шифр MARS 35
2.3.4.3 Шифр RC6 37
2.3.4.4 Шифр Serpent 38
2.3.4.5 Шифр TwoFish 38
2.3.4.6 Шифр Rijndael 40
2.4 Симетричні криптосистеми 42
2.4.1 Функції криптосистем 42
2.4.2 Алгоритми створення ланцюжків 43
2.4.3 Методи рандомізації повідомлень 46
2.4.3.1 Огляд методів рандомізації повідомлень 46
2.4.3.2 Генератори випадкових і псевдовипадкових послідовностей 47
2.4.4 Архівація 48
2.4.4.1 Загальні принципи архівації 48
2.4.4.2 Алгоритм Хаффмана 48
2.4.4.3 Алгоритм Лемпеля - Зіва 50
2.4.5 Хешування паролів 50
2.4.6 Транспортне кодування 53
2.4.7 Загальна схема симетричної криптосистеми 55
2.5 Асиметричні криптоалгоритми 56
2.5.1 Загальні відомості про асиметричні криптоалгоритми 56
2.5.2 Алгоритм RSA 57
2.5.3 Технології цифрових підписів 59
2.5.4 Механізм розповсюдження відкритих ключів 60
2.5.5 Обмін ключами за алгоритмом Діффі - Хеллмана 61
2.5.6 Загальна схема асиметричної криптосистеми 62
3 Методи і пристрої зберігання інформації 64
3.1 Методи резервного копіювання 64
3.2 Пристрої резервного копіювання 65

3.2.1 Стрімери 66
3.2.2 Магнітооптичні накопичувачі 67
3.2.3 Накопичувачі типу Bernoulli 68
3.2.4 Накопичувачі Zip і Jaz 68
3.2.5 Портативні накопичувачі на жорстких дисках 68
3.2.6 CD та DVD приводи 70
3.2.7 Твердотільні накопичувачі SSD 70
3.2.8 Карти пам'яті 71
3.3 Дискові масиви з надмірністю даних 72
3.3.1 RAID 0 72
3.3.2 RAID 1 72
3.3.3 RAID 2 73
3.3.4 RAID 3 74
3.3.5 RAID 4 74
3.3.6 RAID 5 75
3.3.7 RAID 6 76
3.3.8 RAID 10 76
3.3.9 RAID 30 78
3.3.10 RAID 50 78
3.3.11 RAID 7 79
3.3.12 Варіанти реалізації RAID систем 80
4 Комп’ютерні віруси 83
4.1 Типи зловмисного програмного забезпечення 83
4.2 Класифікація вірусів 84
4.3 Типи антивірусних засобів 86
5 Захист інформації під час роботи в комп'ютерних мережах 88
5.1 Класифікація мережевих атак та засоби їх попередження 88
5.2 Міжмережеві екрани 93
5.3 Основні схеми захисту на базі міжмережевих екранів 98
6 Засоби перевірки цілісності інформації 102
6.1 Методи виявлення помилок 102
6.2 Метод обчислення циклічного надмірного коду 103
6.3 Приклад розрахунку CRC коду 109
6.4 Алгоритми програмної реалізації CRC 110
6.4.1 Пряма реалізація CRC 110
6.4.2 Реалізація табличного алгоритму 111
6.4.3 Перетворений табличний алгоритм 114
Перелік інформаційних джерел 117

ВСТУП

Інформація є одним з найбільш цінних ресурсів будь-якої компанії. Тому забезпечення захисту інформації є акутальним і важливійшим завданням.

Метою викладання дисципліни «Засоби захисту у мережах ЕОМ» є надання студентам чітких уявлень про принципи побудови систем захисту у інформаційних мережах.

Завданням дисципліни є ознайомлення студентів з основними методами обробки інформації, існуючими технологіями захисту інформації і практичними навичками з їх створення, впровадження і супроводження.

У результаті вивчення навчальної дисципліни студент повинен

знати:

- теорію побудови систем захисту інформації;
- криптографічні методи обробки інформації;
- технології зберігання інформації;
- методи захисту від комп'ютерних вірусів;
- основи технології контролю мережевого трафіку.

вміти:

- виконувати розрахунки контрольних сум, надмірних циклічних кодів;
- аналізувати захищеність комп'ютерних мереж від зовнішніх загроз, користуватися спеціальною та довідковою літературою;
- розробляти захищені вузли з обробки інформації

1 ОСНОВНІ ВИДИ І ДЖЕРЕЛА АТАК НА ІНФОРМАЦІЮ

1.1 Сучасна ситуація в області інформаційної безпеки

Інформація - це результат відбиття і обробки в людській свідомості різноманіття навколишнього світу. Відомості, якими обмінюється людина через ЕОМ з іншою людиною або з ЕОМ, і є предметом захисту. Проте захисту підлягає не всяка інформація, а тільки та, яка має ціну. Цінною стає та інформація, володіння якої дозволить її існуючому і потенційному власнику одержати який-небудь виграш: моральний, матеріальний, політичний і т. д. Оскільки в людському суспільстві завжди існують люди, охочі незаконним шляхом одержати цінну інформацію, у її власника виникає необхідність в її захисті.

У 60-ті роки минулого сторіччя, коли у світі з'явились перші хакери, серед них було багато аматорів, програмістів та студентів. Спочатку термін хакер описував людей з хорошими навичками програмування. Хакери використовували ці навички, щоб перевірити можливостей та обмежень існуючих тоді систем. Перші хакери також брали участь в розробці комп'ютерних ігор. У багатьох іграх того часу були чарівники та використовувалась магія.

З розвитком культури хакерства, до неї додавалась лексика цих ігор. Навіть зовнішній світ почав проектувати образи могутніх чарівників на незрозумілу хакерську культуру. Книги, такі як «Там, де чарівники лягають спати пізно: витоки Інтернету» (Where Wizards Stay up Late: The Origins of The Internet), яка опублікована у 1996 році, додали містики до хакерської культури. Ці образи та термінологія закріпились. Багато хакерських угрупувань у наш час використовують саме цей імідж. Одна з найбільш сумновідомих хакерських груп має назву Легіон долі (Legion of Doom). Важливо знати кіберкультуру, щоб зрозуміти злочинців кіберсвіту та їх мотивацію.

Китайський філософ і воїн Сунь-Цзи, що жив у шостому столітті до нашої ери. Написав книгу під назвою «Мистецтво війни» (The Art of War), що є класичною роботою про стратегії перемоги над ворогом. Ця книга століттями використовувалась як рекомендації з тактики та стратегії. Знати ворога - один з основних принципів Сунь-Цзи. Хоча він посилався саме на війну, більша частина його рекомендацій підходить і для інших аспектів життя, включаючи проблеми кібербезпеки.

Існує велика кількість груп даних, які утворюють різні домени «кіберсвіту». Якщо групи можуть збирати і обробляти велику кількість даних, вони починають накопичувати силу і вплив. Ці дані можуть бути у вигляді чисел, зображень, відео, аудіо або у вигляді будь-якого типу даних, який може бути поданий у цифровій формі. Групи можуть стати настільки потужними, щоб діяти як окремі сили, створюючи окремі області кібербезпеки.

Такі компанії, як Google, Facebook і LinkedIn, можуть вважатися доменами даних в нашому кіберсвіті. Розвиваючи цю аналогію далі - люди, які працюють в цих компаніях, можуть вважатися експертами з кібербезпеки.

Слово «domain» (домен) має велику кількість значень. Будь-яка область, де існує контроль, авторизація або захист, можливо вважати доменом. Уявіть собі, як дика тварина буде захищати свої володіння (домен в цьому розумінні). В цьому курсі домен розглядається як область, яка підлягає захисту. Вона може бути обмежена логічною або фізичною межею. Це буде залежати від розміру задіяної

системи. У багатьох аспектах, експерти в області кібербезпеки, повинні захищати свої домени відповідно до законів своєї країни.

Експерти Google створили один з перших і найпотужних доменів у відкритому кіберсвіті Інтернету. Мільярди людей щоденно використовують Google для пошуку в Інтернеті. Ймовірно Google, створив найбільшу в світі інфраструктуру збору даних. Google розробив Android - операційну систему, встановлену на більше, ніж 80% всіх мобільних пристроїв, підключених до Інтернету. Кожен пристрій вимагає від користувачів створення облікових записів Google, які можуть зберігати закладки та інформацію про обліковий запис, результати пошуку та навіть знаходити пристрій.

Facebook - ще один потужний домен у доступному Інтернеті. Експерти в Facebook виявили, що люди щодня створюють персональні облікові записи для спілкування з родиною та друзями. Роблячи це, вони добровільно надають значний обсяг особистих даних. Експерти Facebook створили величезний домен даних, який дозволяє людям спілкуватися і знайомитися використовуючи способи, які не можливо було уявити в минулому. Facebook щоденно впливає на мільйони людей і дозволяє компаніям та організаціям спілкуватися з людьми більш персоналізовано і цілеспрямовано.

LinkedIn - це ще один домен зберігання даних в Інтернеті. Експерти LinkedIn підтверджують, що учасники цієї мережі діляться інформацією про свої досягнення з метою створення мережі професійних контактів. Користувачі LinkedIn надають інформацію для створення онлайн-профілів і контактів з іншими учасниками цієї соціальної мережі. LinkedIn сприяє зв'язкам працівників з роботодавцями, а компаніям з іншими компаніями у всьому світі. LinkedIn і Facebook мають багато схожих рис.

Якщо заглянути у середину цих доменів, то можливо зрозуміти, як вони побудовані. На базовому рівні ці домени є потужними через здатність збирати дані, які надаються самими користувачами. Ці дані часто включають в себе біографічні дані користувачів, їх коментарі, симпатії, місця, що були відвідані, подорожі, інтереси, друзів і членів сім'ї, професію, хобі, а також робочий і особистий графік. Для організацій, що зацікавлені у використанні цих даних для кращого розуміння і спілкування з своїми клієнтами та співробітниками, така інформація має велику цінність.

Даних, які зібрані в Інтернеті, містять значно більше відомостей, ніж ті дані, які користувачі надають добровільно. Кібердомени продовжують рости відповідно до розвитку науки і техніки, дозволяючи експертам і їх роботодавцям (Google, Facebook, LinkedIn та ін.) збирати багато інших видів даних. В наш час у кіберфахівців є технології відстеження світових тенденцій погоди, моніторингу океанів, а також відслідковування переміщення і поведінки людей, тварин і об'єктів в режимі реального часу.

З'явилися нові технології, такі як геопросторові інформаційні системи (Geospatial Information Systems, GIS) та Інтернет речей (Internet of Things, IoT). Ці нові технології можуть відслідковувати стан дерев в районі. Вони можуть надавати актуальну інформацію про розташування транспортних засобів, пристроїв, людей і предметів. Цей тип інформації може заощадити енергію, підвищити ефективність і знизити ризики у сфері безпеки. Кожна з цих технологій також викличе різке збільшення обсягу зібраних, проаналізованих і

використаних даних, з метою розширення знань про оточуючий світ. Дані, що збираються GIS і IoT, будуть створювати величезну проблему для експертів з кібербезпеки в майбутньому. Потенційно типи даних, які генеруються цими пристроями, можуть дозволити кіберзлочинцям отримати доступ до дуже особистих аспектів повсякденного життя.

З появою Інтернету речей (Internet of Things, IoT) з'явилося значно більше даних, які потребують керування та захисту. IoT - це велика мережа фізичних об'єктів, таких як сенсори та пристрої, які виходять за межі традиційної комп'ютерної мережі. Всі ці зв'язки, а також зростання обсягів і послуг зберігання завдяки хмарним технологіям і віртуалізації призводять до експоненційного збільшення даних. Ці дані створили нову область технологій та бізнесу під назвою "Великі дані" (Big Data). Завдяки швидкості, обсягу та різноманітності даних, що генеруються IoT та щоденними бізнес-операціями, конфіденційність, цілісність і доступність цих даних є життєво важливими для організації.

Можливими наслідками атак на інформацію в першу чергу, є економічні втрати:

- розкриття комерційної інформації може привести до серйозних прямих збитків на ринку
- звістка про крадіжку великого обсягу інформації звичайно серйозно впливає на репутацію фірми, ведучи побічно до втрат в обсягах комерційних операцій
- фірми-конкуренти можуть скористатися крадіжкою інформації, якщо та залишилася непоміченою, для того, щоб повністю розорити фірму, нав'язуючи їй фіктивні або явно збиткові операції
- підміна інформації як на етапі передачі, так і на етапі зберігання у фірмі може привести до величезних збитків
- багаторазові успішні атаки на фірму, що надає який-небудь вид інформаційних послуг, знижують довіру до фірми у клієнтів, що позначається на розмірі прибутків.

Також, комп'ютерні атаки можуть принести і величезний моральний збиток. Жодному користувачу комп'ютерної мережі не хочеться, щоб його листи окрім адресата одержували ще 5-10 чоловік, або, наприклад, весь текст, що набирається на клавіатурі ЕОМ, копіювався в буфер, а потім при підключенні до Інтернету відправлявся на певний сервер, як це відбувається в тисячах і десятках тисяч випадків.

При зберіганні, підтримці і наданні доступу до будь-якого інформаційного об'єкту його власник, або уповноважена їм особа, накладає явно або самоочевидний набір вимог по роботі з нею. Навмисне їх порушення класифікується як атака на інформацію.

Профілі та мотиви кіберзлочинців змінювалися з роками. Хакінг почався в 60-ті роки з телефонного фрікінгу (phone freaking або phreaking), що полягав у використанні різних звукових частот для маніпулювання телефонними системами. У середині 80-х років злочинці використовували комп'ютерні модеми для комутованих ліній для підключення комп'ютерів до мереж і застосовували програми для зламу паролів, щоб отримати доступ до даних. В наш час злочинці виходять за рамки банальної крадіжки інформації. Сьогодні злочинці можуть використовувати шкідливе ПЗ і віруси як високотехнологічну зброю. Однак,

найбільша мотивація для більшості кіберзлочинців - фінансова. Кіберзлочинність стала більш прибутковою, ніж нелегальна торгівля наркотиками.

Профілі та мотиви хакерів сильно змінилися. На рис.1.1 показані сучасні хакерські терміни і короткий опис кожного з них.

Попит на фахівців в області кібербезпеки виріс більше, ніж попит на інші IT-спеціальності. Усі технології, які змінюють світ та покращують спосіб життя людей, також роблять їх більш вразливими для атак. Сама технологія не може запобігати, виявляти, реагувати і відновлюватися після кібератак. Слід прийняти до уваги наступні тенденції:



Рисунок 1.1 - Сучасні хакерські терміни

- рівень кваліфікації, необхідний для ефективної роботи фахівця з кібербезпеки, і нестача кваліфікованих фахівців у цій галузі веде до високого рівня зарплат;
- інформаційні технології постійно змінюються; це також справедливо і для кібербезпеки; через високу динамічність сфери кібербезпеки, вона може бути складною і захоплюючою;
- кар'єра фахівця з кібербезпеки також є дуже мобільною; робочі місця існують практично у кожній географічній точці;
- фахівці з кібербезпеки надають необхідні послуги організаціям, країнам та суспільству аналогічно, як співробітники правоохоронних органів або міністерства з надзвичайних ситуацій.

Актуальність і важливість проблеми забезпечення захисту інформації обумовлена наступними причинами:

а) різке збільшення обчислювальної потужності сучасних комп'ютерів, що веде до значного збільшення об'ємів оброблюваної інформації, зокрема такої, втрата якої може привести до серйозних проблем;

б) спрощення експлуатації сучасних комп'ютерів, що веде до значного розширення круга користувачів (не завжди володіючих достатнього для усвідомлення проблем захисту інформації кваліфікацією), що мають доступ до обчислювальних ресурсів, засобів розробки програм і даних;

в) зосередження в єдиних базах даних інформації різного призначення і різної приналежності (тобто що належить різним користувачам);

г) бурхливий розвиток програмних засобів, що не задовольняють навіть мінімальним вимогам безпеки і зростання кількості програмістів;

д) повсюдний розвиток мережних технологій, об'єднання локальних мереж і їх використовування для передачі даних, а також об'єднання мереж в глобальні комп'ютерні мережі, що надає потенційну можливість видаленого порушення безпеки комп'ютерних систем.

З масовим впровадженням комп'ютерів у всі сфери діяльності людини обсяг інформації, що зберігається в електронному вигляді виріс в тисячі раз. І тепер скопіювати за півхвилини дискету з файлом, що містить план випуску продукції, набагато простіше, ніж копіювати або переписувати купу паперів. З появою комп'ютерних мереж навіть відсутність фізичного доступу до комп'ютера перестала бути гарантією збереження інформації.

Згідно матеріалів статистичних досліджень, кожний десятий випадок пошкодження електронних даних пов'язаний з комп'ютерними атаками. Виконавцем цих дій були: в 81% випадків – поточний кадровий склад установ, тільки в 13% випадків – повністю сторонні люди, і в 6% випадків – колишні працівники цих же установ. Частка атак, виконуємих співробітниками фірм і підприємств, примушує застосовувати не тільки про технічних, але і про психологічні методи профілактики подібних дій. Діставшись до інформації: в 44% випадків атак були зроблені безпосередні крадіжки грошей з електронних рахунків, в 16% випадків виводилося з ладу програмне забезпечення, так же часто – в 16% випадків – здійснювалася крадіжка інформації з різними наслідками, в 12% випадків інформація була сфальсифікована, в 10% випадків зловмисники за допомогою комп'ютера скористалися або замовили послуги, до яких у принципі не повинні мати доступу.

Комп'ютерна система обробки інформації (КСОІ) - локалізований або розподілений апаратно-програмний комплекс і обслуговуючий його персонал, що здійснює генерацію, введення, зберігання, обробку, передачу, надання споживачам в різній формі і видалення інформації.

Основні компоненти КСОІ:

а) апаратні засоби - ЕОМ і їх компоненти, периферійні пристрої, лінії зв'язку і мережне устаткування;

б) програмне забезпечення: BIOS, операційні системи, прикладне програмне забезпечення, мови програмування;

в) дані (електронні і бережуться в твердій копії);

г) персонал.

Об'єктами КСОІ називають пасивні компоненти систем, що бережуть, приймають і передають інформацію (приклади об'єктів КСОІ - накопичувачі інформації, канали і пристрої зв'язку).

Суб'єктами КСОІ називають активні компоненти систем, які можуть стати причиною потоку інформації від об'єкту КСОІ до будь-якого суб'єкта КСОІ або причиною зміни стану КСОІ (приклади суб'єктів КСОІ - користувачі, програми, процеси).

Безпека КСОІ - її захищеність від випадкового або навмисного втручання в нормальний процес функціонування, а також від спроб розкрадання, зміни або руйнування її компонентів.

1.2. Категорії інформаційної безпеки

Інформація з погляду інформаційної безпеки володіє наступними категоріями:

- конфіденційність – гарантія того, що конкретна інформація доступна тільки, для кого вона призначена; порушення цієї категорії називається розкраданням або розкриттям інформації
- цілісність – гарантія того, що інформація зараз існує в її початковому вигляді, тобто при її зберіганні або передачі не було проведено несанкціонованих змін; порушення цієї категорії називається фальсифікацією повідомлення
- автентичність – гарантія того, що джерелом інформації є саме та особа, яка заявлена як її автор; порушення цієї категорії також називається фальсифікацією, але вже автора повідомлення
- апеллюємість – гарантія того, що при необхідності можливо буде довести, що автором повідомлення є саме заявлена людина, і не може бути ніхто інший; відмінність цієї категорії від попередньої в тому, що при підміні автора, хтось інший намагається заявити, що він автор повідомлення, а при порушенні апеллюємості – сам автор намагається "відхреститися" від своїх слів, підписаних їм одного разу.

Відносно інформаційних систем застосовуються інші категорії :

- надійність – гарантія того, що система поводиться в нормальному і позаштатному режимах так, як заплановано
- точність – гарантія точного і повного виконання всіх команд
- контроль доступу – гарантія того, що різні групи осіб мають різний доступ до інформаційних об'єктів, і ці обмеження доступу постійно виконуються
- контрольованість – гарантія того, що у будь-який момент може бути проведена повноцінна перевірка будь-якого компоненту програмного комплексу
- контроль ідентифікації – гарантія того, що клієнт, підключений в даний момент до системи, є саме тим, за кого себе видає
- стійкість до умисних збоїв – гарантія того, що при умисному внесенні помилок в межах заздалегідь обумовлених норм система поводитиметься так, як обумовлено заздалегідь.

Доступ до інформації - ознайомлення з інформацією і/або її обробка, зокрема, копіювання, модифікація або знищення. Типи доступу:

а) санкціонований - не порушує встановлені правила розмежування доступу, що є системою заходів (апаратних, програмних, адміністративних, законодавчих і

морально-етичних), що регламентують доступність об'єктів КСОІ і ступінь доступу (має рацію) суб'єктів КСОІ до цих об'єктів;

б) несанкціонований (НСД) - доступ, що порушує встановлені для КСОІ правила розмежування доступу.

Загроза безпеки КСОІ - потенційна можлива навмисна або випадкова подія, яка може надати небажану дію на КСОІ (табл. 1.1). Типи загроз безпеки КСОІ по дії на інформацію:

а) загрози порушення конфіденційності інформації (відбувається завжди при НСД);

б) загрози порушення цілісності інформації;

в) загрози порушення працездатності КСОІ (відмова в обслуговуванні -Deny Service - DOS).

Типи загроз безпеки КСОІ по характеру виникнення:

а) випадкові:

- відмови і збої апаратури;
- перешкоди на лініях зв'язку від дій зовнішнього середовища;
- помилки людини як ланки системи;
- схемні і системотехнічні помилки розробників;
- структурні, алгоритмічні і програмні помилки;
- аварійні ситуації і інші дії.

б) навмисні - пов'язані з цілеспрямованими діями порушника, що мотивуються, наприклад, прагненням одержати права на доступ до об'єктів КСОІ, не передбачені для нього правилами розмежування доступу КСОІ, отриманням матеріальних вигод від використовування інформації КСОІ, конкурентною боротьбою і шпигунством, прагненням до самоутвердження і ін.

Таблиця 1.1. - Можливі шляхи реалізації навмисних загроз безпеки

Компонент КСОІ	**Порушення конфіденційності інформації**	**Порушення цілісності інформації**	**Порушення працездатності системи**
Апаратні засоби	підключення і використання апаратних ресурсів, розкрадання носіїв інформації	зміна режимів роботи апаратури, що завдає зміну інформації	зміна режимів роботи апаратури, що завдає виведення апаратури з ладу
Програмне забезпечення	копіювання, розкрадання, підслуховування	зміна настройок ПЗ, комп'ютерні віруси	зміна настройок ПЗ, "мережні черв'яки"
Дані	копіювання, розкрадання, перехоплення, "троянські коні	спотворення, модифікація	спотворення, підміна, видалення даних
Персонал	Розголошування, передача, відомостей про захист, халатність	Вербування, підкуп персоналу	Відхід з робочого місця, фізичне усунення

1.3. Термінали захищеної інформаційної системи

Термінали – це точки входу користувача в інформаційну мережу. У тому випадку, коли до них мають доступ декількох чоловік або взагалі будь-хто, при їх проектуванні і експлуатації необхідне ретельне дотримання цілого комплексу заходів безпеки

Найпоширенішим способом входу в систему при атаках на інформацію залишається вхід через офіційний log-in запит системи. Обчислювальна техніка, яка дозволяє реалізувати вхід в систему, називається в теорії інформаційної безпеки терміналом. Якщо система складається всього з одного персонального комп'ютера, то він одночасно вважається і терміналом і сервером. Доступ до терміналу може бути фізичним, у тому випадку, коли термінал – це ЕОМ з клавіатурою і дисплеєм, або видаленим – найчастіше по телефонній лінії (в цьому випадку терміналом є модем, підключений або безпосередньо до системи, або до її фізичного терміналу).

При використанні терміналів з фізичним доступом необхідно дотримувати наступні вимоги:

1. Захищеність терміналу повинна відповідати захищеності приміщення: термінали без пароля може бути присутнім тільки в тих приміщеннях, куди мають доступ обличчя відповідного або вищого рівня доступу. Відсутність імені реєстрації можлива тільки в тому випадку, якщо до терміналу має доступ тільки одна людина, або якщо на групу осіб, що мають до нього доступ, розповсюджуються загальні заходи відповідальності. Термінали, встановлені в публічних місцях повинні завжди запрошувати ім'я реєстрації і пароль.
2. Системи контролю за доступом в приміщення зі встановленим терміналом повинні працювати повноцінно і відповідно до загальної схеми доступу до інформації.
3. У випадку установки терміналу в місцях з широким скупченням народу клавіатура, а якщо необхідно, то і дисплей повинні бути обладнані пристроями, що дозволяють бачити їх тільки працюючому в даний момент клієнту (непрозорі скляні або пластмасові огорожі, шторки, "втоплена" модель клавіатури).

При використанні видалених терміналів необхідно дотримувати наступні правила:

1. Будь-який видалений термінал повинен запрошувати ім'я реєстрації і пароль.
2. Своєчасне відключення всіх модемів, не потрібних в даний момент фірмі або не контрольованих в даний момент співробітниками.
3. По можливості рекомендується використовувати схему зворотного дзвінка від модему, оскільки вона гарантує з рівнем надійності АТС те, що видалений клієнт дістав доступ з певного телефонного номера.
4. З log-in запиту терміналу рекомендується прибрати всі безпосередню інформацію щодо імені фірми, її логотипу і т.п. – це не дозволить комп'ютерним злочинцям, що просто перебирають номери з модемами, дізнатися log-in екран якої фірми вони знайшли.
5. На вході в систему рекомендується виводити на екран попередження про те, що вхід в систему без повноважень на це переслідується згідно із законом. Це стане ще одним застереженням зловмисникам, а також, буде надійним

аргументом на користь атакованої фірми в судовому розгляді, якщо таке проводитиметься.

Безвідносно від фізичного або комутованого доступу до терміналу, лінія, що з'єднує термінал (комутований, або встановлений в публічному місці) із зоною ядра інформаційної системи повинна бути захищена від прослуховування, або ж весь обмін інформацією повинен здійснюватись за конфіденційною схемою ідентифікації і надійною схемою аутентифікації клієнта.

Подальші дії зловмисника, що дістав доступ до термінальної точки входу, можуть розвиватися по двох основних напрямах: спроби з'ясування пароля прямо або побічно; спроби входу в систему вчинено без знання пароля, ґрунтуючись на помилках в реалізації програмного або апаратного забезпечення.

Перебір паролів по словнику був якийсь час однією з найпоширенішої техніки підбору паролів. Сьогодні набули широке поширення мови складання паролів, що записують в абстрактній формі основні принципи складання паролів середньостатистичними користувачами ЕОМ.

Іншою модифікацією підбору паролів є перевірка паролів, встановлюваних в системах за умовчанням. В деяких випадках адміністратор програмного забезпечення, проїнсталювавши або отримавши новий продукт від розробника, не спромагається перевірити, з чого складається система безпеки. Як наслідок, пароль, встановлений у фірмі розробнику за умовчанням, залишається основним паролем в системі. В мережі Інтернет публікуються величезні списки паролів за умовчанням практично до всіх версій програмного забезпечення, якщо вони встановлюються на ньому виробником.

Основні вимоги до інформаційної безпеки, засновані на аналізі даного методу, наступні:

1. Вхід всіх користувачів в систему повинен підтверджуватися введенням унікального для клієнта пароля.
2. Пароль повинен ретельно підбиратися так, щоб його інформаційна місткість відповідала часу повного перебору пароля. Для цього необхідно детально інструктувати клієнтів про поняття "простий до підбору пароль", або передати операцію вибору пароля у ведення інженера по безпеці.
3. Паролі за умовчанням повинні бути змінені до офіційного запуску системи і навіть до скільки або публічних випробувань програмного комплексу. Особливо це відноситься до мережного програмного забезпечення.
4. Всі помилкові спроби увійти до системи повинні враховуватися, записуватися у файл журналу подій і аналізуватися через "розумний" проміжок часу. Якщо в системі передбачена можливість блокування клієнта або всієї системи після певної кількості невдалих спроб входу, цією можливістю необхідно скористатися. Зазвичай необхідно блокувати клієнта після 3-їй підряд неправильної спроби набору пароля, і, відповідно, блокувати систему після наступної кількості невдалих спроб входу за деякий період (годину, зміну, доба):

$$K=\max(\text{int}(N*0.1*3)+1, 3),$$

де N – середня кількість клієнтів, що підключаються за цей період до системи 0.1 – 10% межа "забудькуватості пароля", 3 – ті ж самі три спроби на згадку пароля.

Інформація про блокування клієнта або системи повинна автоматично поступати на пульт контролю за системою.

1. У момент відправки пакету підтвердження або відкидання пароля в системі повинна бути встановлена затримка (2 - 5 секунд). Це не дозволить зловмиснику, потрапивши на лінію з добрим зв'язком до об'єкту атаки перебирати по сотні тисяч паролів за секунду.

2. Всі дійсні в системі паролі бажано перевіряти сучасними програмами підбору паролів, або оцінювати особисто адміністратору системи.

3. Через певні проміжки часу необхідна примусова зміна пароля у клієнтів. Інтервалами зміни пароля, що найчастіше використовуються, є рік, місяць і тиждень (залежно від рівня конфіденційності інформації і частоти входу в систему).

4. Всі невживані протягом довгого часу імена реєстрації повинні переводитися в закритий (недосяжний для реєстрації) стан. Це відноситься до співробітників, що знаходяться у відпустці, на лікарняному, у відрядженні, а також до імен реєстрації, створених для тестів, випробувань системи і т.п.

5. Від співробітників і всіх операторів терміналу необхідно вимагати суворе нерозголошування паролів, відсутність яких-небудь взаємозв'язків пароля з широковідомими фактами і даними, і відсутність паперових записів пароля "через погану пам'ять".

Наступною по частоті використання є методика отримання паролів з самої системи. Основними двома можливостями з'ясування пароля є несанкціонований доступ до носія, що містить їх, або використання недокументованих можливостей і помилок в реалізації системи.

Перша група методів заснована на тому, що будь-якій системі доводиться де-небудь зберігати оригінали паролів всіх клієнтів для того, щоб звіряти їх у момент реєстрації. При цьому паролі можуть зберігатися як у відкритому текстовому вигляді, як це має місце в багатьох клонах UNIX, так і представлені у вигляді малозначних контрольних сум (хеш - значень), як це реалізовано в ОС Windows, Novell NetWare і багатьох інших. При цьому для зберігання паролів на носії не може бути використана основна методика захисту – шифрування. Якщо всі паролі зашифровані яким-небудь ключем, то цей ключ теж повинен зберігатися в самій системі для того, щоб вона працювала автоматично, не питаючи кожного разу у адміністратора дозвіл. Тому, діставши доступ до подібної інформації, зловмисник може або відновити пароль в читабельному вигляді, або відправити запити, підтверджені даним хеш - значенням, не відновляю його. Рекомендації ж по запобіганню розкраданням паролів полягають в перевірці чи не доступний файл з паролями, або таблиця в базі даних, що зберігає ці паролі, декому ще окрім адміністраторів системи, чи не створюється системою резервних файлів, в місцях доступних іншим користувачам і т.п. . Оскільки крадіжка паролів є найгрубішим вторгненням в систему, розробники надають їй досить пильну увагу, і дотримання всіх рекомендацій по використовуванню системи звичайно достатньо для запобігання подібним ситуаціям.

Отримання доступу до паролів завдяки недокументованим можливостям систем зустрічається украй рідко. Раніше ця методика використовувалася розробниками набагато частіше в основному в цілях відладки, або для екстреного

відновлення працездатності системи. Але поступово з розвитком як технологій зворотної компіляції, так і інформаційної зв'язаності миру вона поступово стала зникати. Будь-які недокументовані можливості рано чи пізно стають відомими, після чого новина про це облітає світ і розробникам доводиться розсилати всім користувачам скомпрометованої системи "програмні латочки" або нові версії програмного продукту. Єдиною мірою профілактики цього методу є постійний пошук на серверах, присвячених комп'ютерній безпеці, оголошень про всі неприємності з програмним забезпеченням, встановленим у КСОІ. Будь-яка подібна вбудована можливість може понизити на порядок загальну безпеку системи, як би добре вона не була завуальована в коді програмного продукту.

Наступною поширеною технологією отримання паролів є копіювання буфера клавіатури у момент набору пароля на терміналі. Цей метод використовується рідко, так для нього необхідний доступ до термінальної машини з можливістю запуску програм. Але якщо зловмисник все-таки дістає подібний доступ, дієвість даного методу дуже висока:

1. робота програми - перехоплювача паролів (так званого "троянського коня") на робочій станції непомітна.

2. подібна програма сама може відправляти результати роботи на наперед заданий сервер або анонімним користувачам, що різко спрощує саму процедуру отримання паролів хакером, і утрудняє пошук і доказ його провини.

Двома основними методами боротьби з копіюванням паролів є:

1. адекватний захист робочих станцій від запуску сторонніх програм:

   а) відключення змінних носіїв інформації,

   б) спеціальні драйвера, що блокують запуск здійснимих файлів без відома оператора, або адміністратора,

   в) монітори, що повідомляють про будь-які зміни системних настройок і списку програм, що автоматично запускаються,

2. система одноразових паролів (при кожній реєстрації в системі клієнтам з дуже високим рівнем відповідальності самою системою генерується новий пароль).

Сканування сучасними антивірусними програмами також може допомогти у виявленні "троянських" програм, але тільки тих з них, які набули широке поширення.

Наступний метод отримання паролів відноситься тільки до мережевого програмного забезпечення. В багатьох програмах не враховується можливість перехоплення будь-якої інформації, що йде по мережі – так званого мережевого трафіку. Більше половини протоколів мережі Інтернет передають паролі в нешифрованому вигляді – відкритим текстом. До них відносяться протоколи передачі електронної пошти SMTP і POP3, протокол передачі файлів FTP, одна з схем авторизації на WWW-серверах.

Сучасне апаратне і програмне забезпечення дозволяє одержувати всю інформацію, що проходить по сегменту мережі, до якого підключений конкретний комп'ютер, і аналізувати її в реальному масштабі часу. Можливі декілька варіантів прослуховування трафіку:

1. це може зробити службовець компанії з свого робочого комп'ютера,

2. зловмисник, що підключився до сегменту за допомогою портативної ЕОМ або мобільнішого пристрою,

3. трафік, що йде по мережі Інтернет, технічно може прослуховуватися з боку безпосереднього провайдера, з боку будь-якої організації, що надає транспортні послуги для мережі Інтернет (листування усередині країни в середньому йде через 3-4 компанії, за межі країни – через 5-8).

Для комплексного захисту від подібної можливості крадіжки паролів необхідно виконувати наступні заходи:

1. Фізичний доступ до мережних кабелів повинен відповідати рівню доступу до інформації.

2. При визначенні топології мережі слід уникати при будь-якій нагоді широкомовних топологій. Оптимальною одиницею сегментації є група операторів з рівними правами доступу, або якщо ця група складає більше 10 чоловік, то кімната або відділ усередині групи. У жодному випадку на одному кабелі не повинні знаходитися оператори з різними рівнями доступу, якщо тільки весь передаваний трафік не шифрується, а ідентифікація не виконується по прихованій схемі без відкритої передачі пароля.

3. До всіх інформаційних потоків, що виходять за межі закладу, повинні застосовуватися ті ж правила, що стосуються об'єднання різнорівневих терміналів.

1.4 Етапи побудови комплексної системи захисту КСОІ

Етапами побудови комплексної системи захисту КСОІ є наступне:

1. Аналіз можливих загроз КСОІ;

2. Планування системи захисту:

а) визначення законодавчих актів, що забезпечують захист КСОІ і передбачають відповідальність за її порушення і ознайомлення з ними користувачів КСОІ.

б) визначення морально-етичних заходів захисту КСОІ - створення здорового морально-етичного клімату в колективі користувачів КСОІ, матеріальні і моральні заохочення користувачів, що підвищують престиж організації, виконують всі вимоги безпеки КСОІ, не дозволяють невиправданого використовування ресурсів КСОІ і ін.

в) визначення адміністративних заходів захисту, що регламентують:

- загальну схему функціонування КСОІ - основні вирішувані задачі, функціональні блоки і їх взаємозв'язок;
- вибір персоналу КСОІ і визначення порядку взаємодії користувачів з системою;
- використовування ресурсів КСОІ і визначення прав доступу персоналу до них.

г) визначення фізичних заходів захисту:

- механічних (прихильність компонентів КСОІ і контроль доступу до них)
- забезпечення гарантованого електроживлення КСОІ; визначення засобів захисту від пожеж, повеней і інших стихійних лих;
- визначення засобів відео і аудіо нагляду, г) визначення апаратно-програмних засобів захисту:

- забезпечення ідентифікації (розпізнавання) і аутентифікації (перевірки достовірності) суб'єктів КСОІ; розмежування доступу до ресурсів КСОІ; контроль цілісності даних КСОІ; забезпечення конфіденційності даних КСОІ; аудит і аналіз подій КСОІ; резервування ресурсів і компонентів КСОІ.

3. Реалізація системи захисту інформації КСОІ.

4. Супровід системи захисту КСОІ, контроль і аналіз подій з метою виявлення пролому, корекції системи захисту і ін.

Основними принципами забезпечення безпеки інформації є:

- економічна ефективність засобів захисту - вартість захисту повинна бути менше вартості інформації, що захищається; проектування системи захисту з урахуванням принципу "ворожого оточення";
- виключення з кола контрольованих системою захисту користувачів її розробників;
- визначення особистої відповідальності осіб, що займаються забезпеченням безпеки КСОІ;
- забезпечення користувачам за умовчанням мінімальних прав і привілеїв, достатніх для їх нормальної роботи;
- розділення об'єктів і суб'єктів КСОІ на групи і управління захистом на рівні груп;
- забезпечення постійного аудиту використовування об'єктів і роботи суб'єктів КСОІ;
- забезпечення ухвалення рішення людиною, а не автоматизованого ухвалення рішення в найважливіших ситуаціях; система захисту інформації повинна відключатися тільки адміністратором і лише в екстрених випадках; приховування, по можливості, факту існування системи захисту.

1.5. Соціальна психологія і інші способи отримання паролів

Іноді зловмисники вступають і в прямий контакт з лицями, що володіють потрібною їм інформацією, розігруючи досить переконливі сцени. "Жертва" омани, що повірила в реальність розказаної їй по телефону або в електронному листі ситуації, сама повідомляє пароль зловмиснику.

Соціальна інженерія - область знань, що використовується хакерами і крекерами для забезпечення їх НСД до КСОІ інакше, ніж через злом ПО. Мета атакуючого - обхитрувати користувачів КСОІ з метою отримання їх паролів і ключів доступу до КСОІ. Класичне шахрайство включає дзвінки по телефону в організацію для визначення жертви, що атакується, і подальшого дзвінка адміністратору від її імені з проханням організувати негайний доступ до КСОІ (наприклад, змінивши пароль).

Класифікація атак класу соціальної інженерії:

1. По засобах вживання: телефон; електронна пошта; повідомлення Internet реального часу; звичайна пошта; особиста зустріч.

2. По рівню соціального відношення до об'єкту: офіційний стиль спілкування; товариський стиль спілкування; дружній стиль спілкування.

3. По ступеню доступу жертви до КСОІ: адміністратор КСОІ - високий ступінь доступу; керівник КСОІ (ланки КСОІ) (часто секретар керівника) -

середній ступінь доступу; користувач КСОІ - низький ступінь доступу; знайомі адміністратора, начальника, користувача.

Розглянемо декілька методів, що дуже часто зустрічаються.

Дзвінок адміністратору – зловмисник вибирає із списку співробітників того, хто не використовував пароль для входу протягом декількох днів (відпустка, відгули, відрядження) і кого адміністратор не знає по голосу. Потім слідує дзвінок з поясненням ситуації про забутий пароль, проханням зачитати пароль, або змінити його на новий.

Майже така ж схема, але у зворотний бік може бути розіграна зловмисником на адресу співробітника – дзвінок від адміністратора. В цьому випадку він представляється вже співробітником служби інформаційної безпеки і просить назвати пароль або через збій, що відбувся, в базі даних, або нібито для підтвердження особи самого співробітника з якої-небудь причини, або з приводу останнього підключення співробітника до якого-небудь інформаційного серверу усередині закладу. Крім того, така схема може бути проведена і без телефонного дзвінка – по електронній пошті, що неодноразово і виконувалося нібито від імені поштових і Web - серверів в мережі Інтернет. Атакуючий, одержавши ім'я користувача КСОІ і пароль (наприклад, одним з вищеописаних способів), дзвонить адміністратору КСОІ працюючої, наприклад, під ОС UNIX, і повідомляє, що він не може чомусь виконати звичайно дозволених для нього дій, наприклад, проглянути файли в своєму домашньому каталозі (це виконується командою ls). Адміністратор, зареєстрований в системі як root (з найвищими привілеями), входить в каталог користувача і виконує успішно програму ls. Якщо в змінній PATH адміністратора включений поточний каталог (крапка), то від імені адміністратора буде виконана програма, яку створив атакуючий, пойменував її ls і розмістив в домашньому каталозі користувача-жертви. Така програма може порушувати конфіденційність і цілісність інформації в КСОІ, і надавати НСД до її об'єктів.

Обидва цих методи відносяться до групи "атака з соціальній психології" і можуть приймати самі різні форми. Їх профілактикою може бути тільки ретельне роз'яснення всім співробітникам, в особливо важливих випадках введення адміністративних заходів і особливого регламенту запиту і зміни пароля.

Необхідно ретельно інструктувати співробітників про небезпеку залишення робочих станцій, не закритих паролем. В першу чергу це, звичайно, відноситься до терміналів, що працюють в публічних місцях і офісах з нижчим рівнем доступу до інформації, проте, і при роботі в приміщеннях з рівним рівнем доступу не рекомендується давати можливість співробітникам працювати за іншими ЕОМ тим більше у відсутність власника. Як програмні профілактичні заходи використовуються екранні заставки з паролем, що з'являються через 5-10 хвилин відсутності робочої активності, автоматичне відключення сервером клієнта через такий же проміжок часу.

Велика увага слідує уділяти будь-яким носіям інформації, що покидають межі закладу. Найчастішими причинами цього бувають ремонт апаратури і списання технологічно застарілої техніки. На робочих поверхнях носіїв навіть у видалених областях знаходиться інформація, яка може представляти або безпосередній інтерес, або побічно послужити причиною вторгнення в систему. При використанні віртуальної пам'яті частина вмісту ОЗП записується на

жорсткий диск, що теоретично може привезти навіть до збереження пароля на постійному носії. Ремонт за місцем роботи, повинен проводиться під контролем інженера із служби інформаційної безпеки. Всі носії інформації, що покидають заклад повинні надійно чиститися або знищуватися механічно (залежно від подальших цілей їх використовування).

На сьогоднішній день не існує розумних за критерієм "ціна/надійність" носіїв інформації, не доступних до втручання. Будова файлів, їх заголовки і розташування в будь-якій операційній системі може бути прочитане при використанні відповідного програмного забезпечення. Практично повністю захищеним може бути тільки незалежний носій, що автоматично руйнує інформацію при спробі несанкціонованого підключення до будь-яких точок, окрім дозволених роз'ємів, що бажано саморуйнується при розгерметизації, що має всередині мікропроцесор, що аналізує пароль по схемі без відкритої передачі.

Для приватного листування ця проблема розв'язується набагато простіше і дешевше – за допомогою криптографії. Будь-який об'єм інформації будучи зашифрований за допомогою більш менш стійкої криптосистеми, недосяжний для прочитання без знання ключа. Цілком не важливо, зберігається він на жорсткому диску, на дискеті або компакт-диску, не важливо під управлінням якої операційної системи.

2 КРИПТОГРАФІЯ

2.1 Класифікація криптоалгоритмів

Сама криптографія не є вищим ступенем класифікації суміжних з нею дисциплін. Навпаки, криптографія спільно з криптоаналізом (метою якого є протистояння методам криптографії) складають комплексну науку – кріптологію.

Необхідно відзначити, що в російськомовних текстах по даному предмету зустрічаються різні вживання основних термінів, таких як "криптографія", "тайнопис" і деяких інших. Більш того, і по класифікації криптоалгоритмів можливо зустріти різні думки. У зв'язку з цим автор не претендує на те, що його варіант використовування подібних термінів є єдино вірним.

Відносно криптоалгоритмів існує декілька схем класифікації, кожна з яких заснована на групі характерних ознак. Таким чином, один і той же алгоритм "проходить" відразу за декількома схемами, опиняючись в кожній з них в якій-небудь з підгруп.

Основною схемою класифікації всіх криптоалгоритмів є наступна:

1. Тайнопис. Відправник і одержувач проводять над повідомленням перетворення, відомі тільки їм двом. Сторонним особам невідомий сам алгоритм шифрування. Деякі фахівці вважають, що тайнопис не є криптографією взагалі, і автор знаходить це абсолютно справедливим.

2. Криптографія з ключем. Алгоритм дії на дані відомий всім стороннім особам, але він залежить від деякого параметра – "ключа", яким володіють тільки відправник і одержувач.

- Симетричні криптоалгоритми. Для зашифровуваної і розшифровки повідомлення використовується один і той же блок інформації (ключ).
- Асиметричні криптоалгоритми. Алгоритм такий, що для зашифровуваної повідомлення використовується один ("відкритий") ключ, відомий всім охочим, а для розшифровки – інший ("закритий"), існуючий тільки у одержувача.

Далі будемо розглядати криптографію з ключем, оскільки більшість фахівців саме по відношенню до цих криптоалгоритмів використовує термін криптографія. Так, наприклад, будь-який криптоалгоритм з ключем можливо перетворити на тайнопис, просто «зашивши» в початковому коді програми деякий фіксований ключ. Зворотне ж перетворення практично неможливе.

Залежно від характеру дій над даними, алгоритми підрозділяються на:

1. Перестановочні. Блоки інформації (байти, біті, крупніші одиниці) не змінюються самі по собі, але змінюється їх порядок проходження, що робить інформацію недоступної стороннім спостерігачам.

2. Підстановлювальні. Самі блоки інформації змінюються за законами криптоалгоритму. Переважна більшість сучасних алгоритмів належить цій групі.

Будь-які криптографічні перетворення не збільшують об'єм інформації, а лише змінюють її представлення. Тому, якщо програма шифрування значно (більш, ніж на довжину заголовка) збільшує об'єм вихідного файлу, то в її основі лежить неоптимальний, а можливо і взагалі некоректний криптоалгоритм. Зменшення об'єму закодованого файлу можливе тільки за наявності вбудованого алгоритму архівації в криптосистемі і за умови стисливості інформації (так,

наприклад, архіви, музичні файли формату MP3, відеозображення формату JPEG стискатися більш ніж на 2-4% не будуть).

Залежно від розміру блоку інформації криптоалгоритми діляться на:

1. Потокові шифри. Одиницею кодування є один біт. Результат кодування не залежить від того, що пройшло раніше у вхідному потоці. Схема застосовується в системах передачі потоків інформації, тобто в тих випадках, коли передача інформації починається і закінчується в довільні моменти часу і може випадково уриватися. Найпоширенішими прикладами потокових шифрів є скремблери.

2. Блокові шифри. Одиницею кодування є блок з декількох байтів (сьогодні 8-256). Результат кодування залежить від всіх початкових байтів цього блоку. Схема застосовується при пакетній передачі інформації і кодуванні файлів.

2.2 Скремблери

Скремблерами називаються програмні або апаратні реалізації алгоритму, що дозволяє шифрувати побітово безперервні потоки інформації. Сам скремблер представляє з себе набір біт, що змінюються на кожному кроці за певним алгоритмом. Після виконання кожного чергового кроку на його виході з'являється шифруючий біт – або 0, або 1, який накладається на поточний біт інформаційного потоку операцією XOR (виключаюче АБО).

Останнім часом сфера вживання скремблюючих алгоритмів значно скоротилася. Це пояснюється в першу чергу зниженням об'ємів побітної послідовної передачі інформації, для захисту якої були розроблені ці алгоритми. Практично повсюдно в сучасних системах застосовуються мережі з комутацією пакетів, для підтримки конфіденційності якої використовуються блокові шифри. А їх криптостійкість перевершує, і деколи досить значно, криптостійкість скремблерів.

Суть скремблювання полягає в побітній зміні потоку даних, що проходить через систему. Практично єдиною операцією, що використовується в скремблерах є XOR – «побітне виключаюче АБО». Паралельно проходженню інформаційного потоку в скремблері за певним правилом генерується потік біт – кодуючий потік. Як пряме, так і зворотне шифрування здійснюється накладенням по XOR кодуючої послідовності на початкову.

Генерація кодуючої послідовності біт проводиться циклічно з невеликого початкового об'єму інформації – ключа за наступним алгоритмом. З поточного набору біт обираються значення певних розрядів і складаються по XOR між собою. Всі розряди зсовуються на 1 біт, а тільки що набуте значення ("0" або "1") поміщується в наймолодший розряд, що звільнився. Значення, що знаходилося в найстаршому розряді до зсуву, додається до кодуючої послідовності, стаючи черговим її бітом (див. рис. 2.1).

З теорії передачі даних криптографія запозичувала для запису подібних схем двійкову систему запису. По ній зображений на рисунку скремблер записується комбінацією "10011_2" – одиниці відповідають розрядам, з яких знімаються біти для формування зворотного зв'язку.

Розглянемо приклад кодування інформаційної послідовності 010111_2 скремблером 101_2 з початковим ключем 110_2.

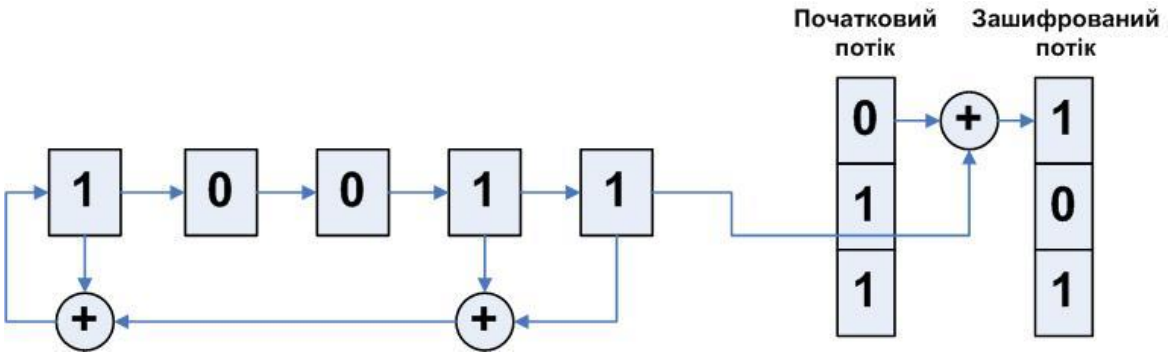


Рисунок 2.1 - Структура п’ятирозрядного скремблера

Стан ключа	Кодуючий біт		Інформаційний біт		Результат
110	0	XOR	0	=	0
111	1	XOR	1	=	0
011	1	XOR	0	=	1
101					

і т.д.

Т.ч., устрій скремблера гранично простий. Його реалізація можлива як на електронній, так і на електричній базі, що і забезпечило його широке вживання в польових умовах. Більш того, той факт, що кожний біт вихідної послідовності залежить тільки від одного вхідного біта, ще більш зміцнило положення скремблерів в захисті потокової передачі даних. Це пов'язано з неминуче виникаючими в каналі передачі перешкодами, які можуть спотворити в цьому випадку тільки ті біти, на які вони доводяться, а не пов'язану з ними групу, байт, як це має місце в блокових шифрах.

Декодування заскремблірованних послідовностей відбувається по тій же самій схемі, що і кодування. Саме для цього в алгоритмах застосовується результуюче кодування по тому, “Виключаюче АБО» – схема, однозначно відновлена при розкодуванні без яких-небудь додаткових обчислювальних витрат. Проведемо декодування одержаного фрагмента.

Головна проблема шифрів на основі скремблерів - синхронізація передаючого (кодуючого) і приймаючого (декодуючого) пристроїв. При пропуску або помилковому вставлянні хоча б одного біта вся передавана інформація необоротно втрачається. Тому, в системах шифрування на основі скремблерів дуже велика увага надається методам синхронізації. На практиці для цих цілей звичайно застосовується комбінація двох методів:

а) додавання в потік інформації синхронізуючих бітів, наперед відомих приймальній стороні, що дозволяє їй при незнаходженні такого біта активно почати пошук синхронізації з відправником,

б) використовування високоточних генераторів тимчасових імпульсів, що дозволяє в моменти втрати синхронізації проводити декодування бітів інформації, що приймаються, "по пам'яті" без синхронізації.

Кількість біт, охоплених зворотним зв'язком, тобто розрядність пристрою пам'яті для біт, що породжують кодуючу послідовність, називається розрядністю скремблера. Зображений вище скремблер має розрядність 5. Відносно параметрів криптостійкості ця величина повністю ідентична довжині ключа блокових шифрів, який буде проаналізований далі. На даному ж етапі важливо відзначити, що чим більше розрядність скремблера, тим вище криптостійкість системи, заснованої на його використовуванні.

При достатньо довгій роботі скремблера неминуче виникає його зациклення. Після виконання певного числа тактів в осередках скремблера створиться комбінація біт, яка в ньому вже одного разу виявлялася, і з цієї миті кодуюча послідовність почне циклічно повторюватися з фіксованим періодом. Така проблема невирішується за своєю природою, оскільки в N розрядах скремблера не може перебувати більше 2^N комбінацій біт, і, отже, максимум, через, 2^N-1 циклів повтор комбінації обов'язково відбудеться. Комбінація "всі нулі" зразу ж виключається з ланцюжка графа станів скремблера – вона приводить скремблер до такого ж положення "всі нулі". Це указує ще і на те, що ключ "всі нулі" непридатний для скремблера. Кожний біт, що генерується при зсуві, залежить тільки від декількох біт береженої в даний момент скремблером комбінації. Тому після повторення деякої ситуації, що одного разу вже зустрічалася в скремблері, всі наступні за нею повторюватимуть в точності ланцюжок, що вже пройшов раніше в скремблері.

Можливі різні типи графів стану скремблера. На рис. 2.2 наведені зразкові варіанти для 3-розрядного скремблера. У випадку "А" окрім завжди присутнього циклу "000">>"000" ми бачимо ще два цикли – з 3-ма станами і 4-ма. У випадку "Б" ми бачимо ланцюжок, який сходиться до циклу з 3-х станів і вже ніколи звідти не виходить. І нарешті, у випадку "В" всі можливі стани окрім нульового, об'єднані в один замкнутий цикл. Очевидно, що саме в цьому випадку, коли всі 2^N-1 станів системи утворюють цикл, період повторення вихідних комбінацій максимальний, а кореляція між довжиною циклу і початковим станом скремблера (ключем), яка призвела б до появи слабкіших ключів, відсутня.

Доведено, що для скремблера будь-якої розрядності N завжди існує такий вибір охоплюваних зворотним зв'язком розрядів, що послідовність біт, що генерується ними, матиме період, рівний 2^N -1 бітів. Так, наприклад, в 8-бітовому скремблері, при обхваті 0- го, 1-го, 6-го і 7-го розрядів дійсно за час генерації 255 біт послідовно минають всі числа від 1 до 255, не повторюючись жодного разу.

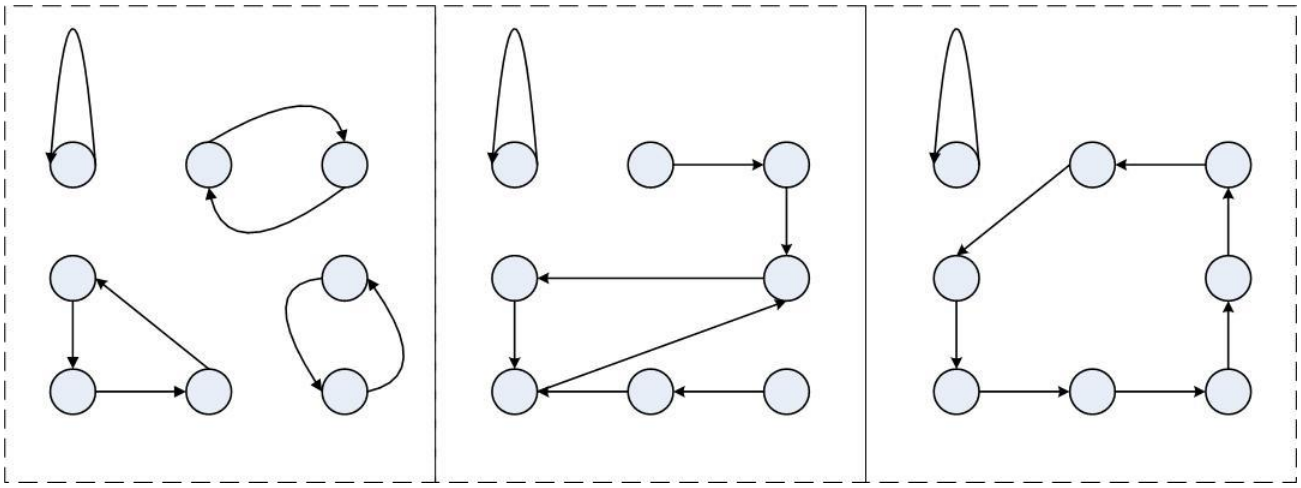


Рисунок 2.2 - Графи, що характеризують стан скремблера

Схеми з вибраними за цим законом зворотними зв'язками називаються генераторами послідовностей найбільшої довжини (ПНД), і саме вони використовуються в скремблюючої апаратурі. З множини генераторів ПНД заданої розрядності у часи, коли вони реалізовувалися на електричній або мінімальній електронній базі вибиралися ті, у яких число розрядів, що беруть участь в створенні чергового біта, було мінімальним. Звичайно генератора ПНД вдавалося досягти за 3 або 4 зв'язки. Сама ж розрядність скремблерів перевищувала 30 біт, що давало можливість передавати до 240 біт = 100 МБайт інформації без побоювання повторення кодуючої послідовності. ПНД нерозривно пов'язані з математичною теорією поліномів, що неприводяться. Достатньо щоб поліном ступеня N не був представимо по модулю 2 у вигляді добутку ніяких інших поліномів, для того, щоб скремблер, побудований на його основі, створював ПНД. Наприклад, єдиним поліномом ступеня 3, що неприводиться, є x^3+x+1, у двійковому вигляді він записується як 1011_2 (одиниці відповідають присутнім розрядам). Скремблери на основі поліномів, що не приводяться утворюються відкиданням самого старшого розряду (він завжди присутній, а, отже, несе інформацію тільки про ступінь полінома), так на основі вказаного полінома, ми можемо створити скремблер 011_2 з періодом зациклення 7 ($=2^3 - 1$). На практиці застосовуються поліноми значно вищих порядків. А таблиці поліномів будь-яких ступенів, що неприводяться, можливо завжди знайти в спеціалізованих математичних довідниках. Істотним недоліком скремблерів є їх нестійкість до фальсифікації.

2.3 Блокові шифри

2.3.1 Загальні відомості про блокові шифри

Блокові шифри шифрують цілі блоки інформації (від 4 до 32 байт) як єдине ціле – це значно збільшує стійкість перетворень до атаки повним перебором і дозволяє використовувати різні математичні і алгоритмічні перетворення.

Характерною особливістю блокових криптоалгоритмів є той факт, що в ході своєї роботи вони проводять перетворення блоку вхідної інформації фіксованої довжини і одержують результуючий блок того ж об'єму, але недосяжний для

прочитання стороннім особам, що не володіють ключем. Таким чином, схему роботи блокового шифру можливо описати функціями

$$Z = EnCrypt(X, Key)$$

і

$$X = DeCrypt(Z, Key).$$

Ключ Key є параметром блокового криптоалгоритму і є деяким блоком двійкової інформації фіксованого розміру. Початковий (X) і зашифрований (Z) блоки даних також мають фіксовану розрядність, рівну між собою, але необов'язково рівну довжині ключа.

Блокові шифри є основою, на якій реалізовані практично всі криптосистеми. Методика створення ланцюжків із зашифрованих блоковими алгоритмами байт дозволяє шифрувати ними пакети інформації необмеженої довжини. Така властивість блокових шифрів, як швидкість роботи, використовується асиметричними криптоалгоритмами, повільними за своєю природою. Відсутність статистичної кореляції між бітами вихідного потоку блокового шифру використовується для обчислення контрольних сум пакетів даних і в хешуванні паролів.

У табл. 2.1 наведено розробки всесвітньо визнані стійкими алгоритмами.

Таблиця 2.1 – Відомості з відомих стійких криптоалгоритмів

Назва алгоритму	**Автор**	**Розмір блоку**	**Довжина ключа**
IDEA	Xuejia Lia and James Massey	64 біт	128 біт
CAST128		64 біт	128 біт
BlowFish	Bruce Schneier	64 біт	128 – 448 біт
ГОСТ	НДІ	64 біт	256 біт
TwoFish	Bruce Schneier	128 біт	128 – 256 біт
MARS	Корпорація IBM	128 біт	128 – 1048 біт

Криптоалгоритм є ідеально стійким, якщо прочитати зашифрований блок даних можливо тільки перебравши всі можливі ключі, до тих пір, поки повідомлення не виявиться осмисленим. Оскільки за теорії ймовірності відшукуваний ключ буде знайдений з ймовірністю 1/2 після перебору половини всіх ключів, то на злом ідеально стійкого криптоалгоритму з ключем довжини N потрібно в середньому 2^N-1 перевірок. Таким чином, в загальному випадку стійкість блокового шифру залежить тільки від довжини ключа і зростає експоненціально з її зростанням. Навіть припустивши, що перебір ключів проводиться на спеціально створеній багатопроцесорній системі, в якій завдяки діагональному паралелізму на перевірку 1 ключа йде тільки 1 такт машинного часу, то на злом 128 бітового ключа сучасній техніці потрібно не менше10^{21} років. Але це відноситься тільки до ідеально стійких шифрів, якими, наприклад, з великим ступенем упевненості є алгоритми, що наведені в табл. 2.1.

Окрім цієї умови до ідеально стійких криптоалгоритмам застосовується ще одна дуже важлива вимога, якій вони повинні обов'язково відповідати. При відомих початковому і зашифрованому значеннях блоку ключ, яким здійснено це перетворення, можливо взнати також тільки повним перебором. Ситуації, в яких сторонньому спостерігачу відома частина початкового тексту зустрічаються скрізь. Це можуть бути стандартні надписи в електронних бланках, фіксовані заголовки форматів файлів, досить що часто зустрічаються в тексті довгі слова або послідовності байт. Тому така вимога не є надмірною і також критично виконується стійкими криптоалгоритмами, як і перша вимога.

Т. ч., на функцію стійкого блокового шифру Z = EnCrypt(X, Key) накладаються наступні умови:

1. Функція EnCrypt повинна бути зворотною.
2. Не повинне існувати інших методів прочитання повідомлення X за відомим блоком Z, окрім як повним перебором ключів Key.
3. Не повинне існувати інших методів визначення, яким ключем Key було проведено перетворення відомого повідомлення X в повідомлення Z, окрім як повним перебором ключів.

Розглянемо методи, за допомогою яких розробники блокових криптоалгоритмів добиваються одночасного виконання цих трьох умов з дуже великою часткою достовірності.

Всі дії, що виконуються над даними блоковим криптоалгоритмом, засновані на тому факті, що перетворюваний блок може бути представлений у вигляді цілого позитивного числа з діапазону, відповідного його розрядності. Так, наприклад, 32-бітовий блок даних можливо інтерпретувати як число з діапазону 0…4 294 967 295. Крім того, блок, розрядність якого звичайно є "ступенем двійки", можливо трактувати як декілька незалежних позитивних числа з меншого діапазону (розглянутий 32-бітовий блок можливо також представити у вигляді двох незалежних чисел з діапазону 0…65 535 або у вигляді чотирьох незалежних чисел з діапазону 0…255).

Над цими числами блоковим криптоалгоритмом і виконуються за певною схемою наступні дії, що наведені у табл. 2.2 (зліва дані умовні позначення цих операцій на графічних схемах алгоритмів) :

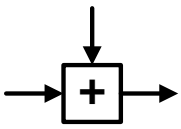
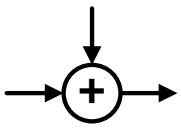
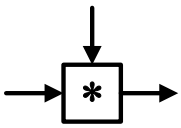
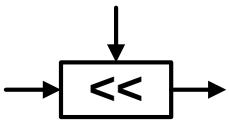
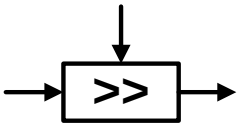
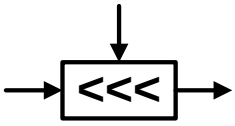
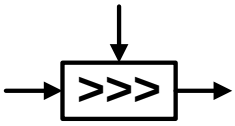
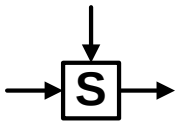
У якості параметру V для будь-якого з цих перетворень може використовуватися:

- фіксоване число (напр., X' = X + 125);
- число, що отримується з ключа (напр., X' = X + F(Key));
- число, що отримується з незалежної частини блоку (напр., X2' = X2 + F(X1)).

Останній варіант використовується в схемі, що має назву - мережа Фейштеля.

Послідовність виконуваних над блоком операцій, комбінації перерахованих вище варіантів V і самі функції F і є "ноу-хау" кожного конкретного блокового криптоалгоритму. Щорічно світові дослідницькі центри публікують черговий блоковий шифр, який під аналізом криптоаналітиків або набуває на декілька років статус стійкого криптоалгоритму, або (що відбувається частіше) безславно йде в історію криптографії.

Таблиця 2.2 - Бієктивні математичні функції

	Додавання	X' = X + V
	Виключаюче АБО	X' = X XOR V
	Множення по модулю 2^N	X' = (X * V) mod (2^N)
	Арифметичний зсув вліво	X' = X SHL V
	Арифметичний зсув вправо	X' = X SHR V
	Циклічний зсув вліво	X' = X ROL V
	Циклічний зсув вправо	X' = X ROR V
	Табличні підстановки S-box	X' = Table[X, V]

Характерною ознакою блокових алгоритмів є багатократне і непряме використання матеріалу ключа. Це зумовлюється в першу чергу вимогою неможливості зворотного декодування відносно ключа при відомих початковому і зашифрованому текстах. Для вирішення цієї задачі в наведених вище перетвореннях найчастіше використовується не саме значення ключа або його частини, а деяка, іноді незворотна (небієктівная) функція від матеріалу ключа. Крім того, в подібних перетвореннях один і той же блок або елемент ключа використовується багато разів. Це дозволяє при виконанні умови зворотності функції щодо величини X зробити функцію зворотною щодо ключа Key.

Оскільки операція шифрування або розшифровки окремого блоку в процесі кодування пакету інформації виконується багато разів (іноді до сотень тисяч раз), а значення ключа і, отже, функцій V_i(Key) залишається незмінним, то іноді стає доцільним наперед одноразово обчислити такі значення і зберігати їх в оперативній пам'яті спільно з ключем. Оскільки ці значення залежать тільки від ключа, то вони в криптографії називаються матеріалом ключа. Така операція не змінює ні довжину ключа, ні криптостійкість алгоритму в цілому. При цьому відбувається лише оптимізація швидкості обчислень шляхом кешування

проміжних результатів. Такі дії використовуються в багатьох блокових криптоалгоритмах і мають назву розширення ключа (key scheduling).

2.3.2 Мережі Фейштеля

Мережа Фейштеля є подальшою модифікацією методу змішування поточної частини шифрованого блоку з результатом деякої функції, що обчислюється від іншої незалежної частини того ж блоку. Ця методика набула широке поширення, оскільки забезпечує виконання вимоги про багаторазове використання ключа і матеріалу початкового блоку інформації.

Класична мережа Фейштеля має структуру, що наведено на рис. 2.3.

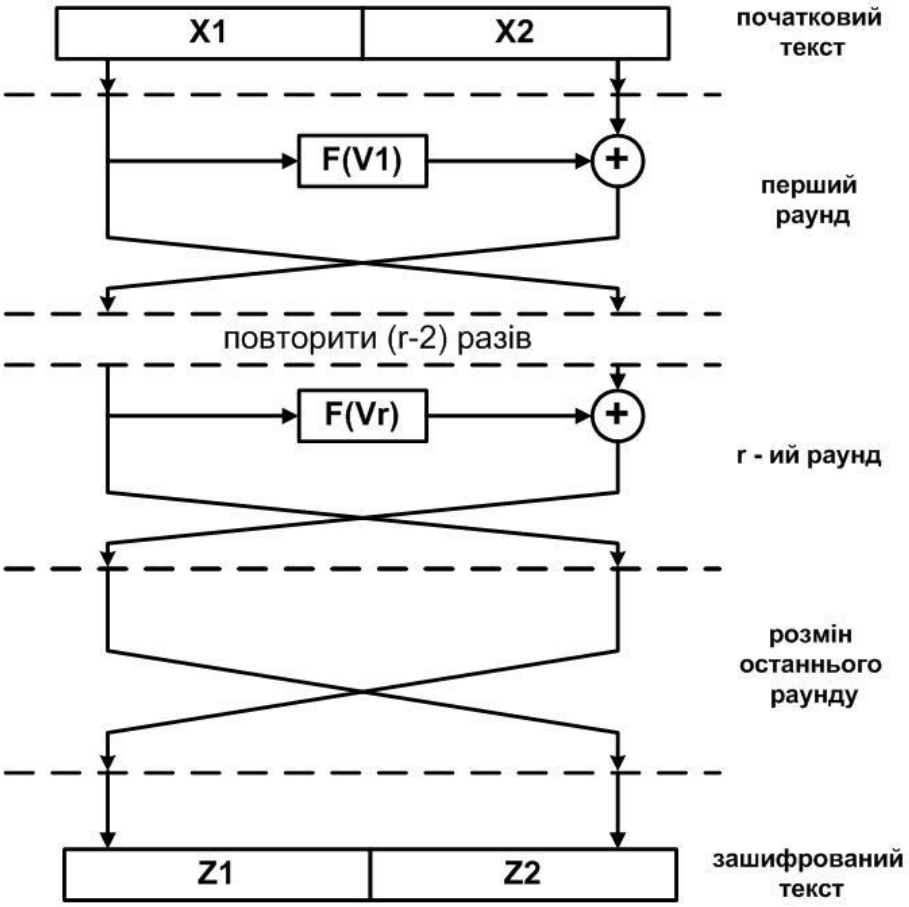


Рисунок 2.3 – Структура класичної мережі Фейштеля

Незалежні потоки інформації, що утворюються з початкового блоку, називаються гілками мережі. В класичній схемі їх дві. Величини Vi є параметрами мережі - зазвичай це функції від матеріалу ключа. Функція F називається утворюючою. Дія, що складається з одноразового обчислення утворюючою функції і подальшого накладення її результату на іншу гілку з обміном їх місцями, називається циклом або раундом мережі Фейштеля. Оптимальна кількість раундів – від 8 до 32. При цьому важливо те, що збільшення кількості раундів значно збільшує кріптостійкість будь-якого блокового шифру відносно криптоаналізу. Ця особливість вплинула на активне розповсюдження мережі Фейштеля – адже при виявленні, якого-небудь слабкого місця в алгоритмі, майже

завжди достатньо збільшити кількість раундів на 4 - 8, не перетворюючі власне сам алгоритм. Часто кількість раундів не фіксується розробниками алгоритму, а лише вказуються оптимальні межі (обов'язково мінімальне значення, і не завжди – максимальне) цього параметра.

Мережа Фейштеля володіє такою властивістю, що навіть якщо у якості утворюючою функції F буде застосовано незворотне перетворення, то і в цьому випадку весь ланцюжок буде відновлено. Це відбувається унаслідок того, що для зворотного перетворення мережі Фейштеля не потрібно обчислювати функцію F^{-1}.

Більш того, мережа Фейштеля є симетричною. Використання операції XOR, зворотної власним же повтором, і інверсія останнього обміну гілок надають можливість декодування блоку тією ж мережею Фейштеля, але з інверсним порядком параметрів V_i. Для зворотності мережі Фейштеля не має значення чи є кількість раундів парним або непарним числом. В більшості реалізацій схеми, де обидві зазначені умови (операція XOR і інверсія останнього обміну) збережені, пряме і зворотне перетворення виконуються однією і тією ж процедурою, до якої передається вектор величин V_i або в початковому, або в інверсному порядку.

З незначними доробками мережу Фейштеля можливо зробити і абсолютно симетричною, тобто використовуючи утворюючі функції шифрування і дешифрування з одним і тим же набором операцій. Математичною це записується як «Функція EnCrypt тотожно дорівнює функції DeCrypt». Розглянемо граф станів криптоалгоритму (рис. 2.4), на якому вершинами позначено блоки вхідної і вихідної інформації. При деякому фіксованому ключі для класичної мережі Фейштеля граф відповідає рис. 2.4 а, а в іншому випадку кожна пара вершин має унікальний зв'язок, як зображено на рис. 2.4 б.

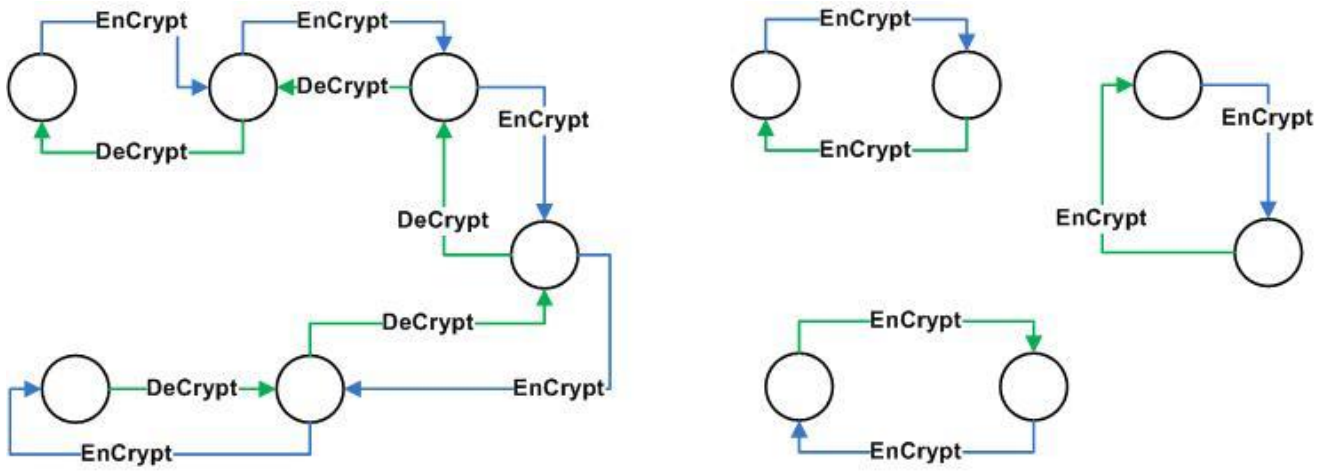


Рисунок 2.4 – Графи станів мережі Фейштеля

Модифікація мережі Фейштеля, що має вказані властивості, наведено на рис. 2.5. Основна її відмінність полягає у повторному використанні даних ключа в зворотному порядку в другій половині циклу. Саме через цю недостатню дослідженої специфіки такої схеми (тобто потенційної можливості послаблення зашифрованого тексту зворотними перетвореннями) її обмежено використовують в криптоалгоритмах.

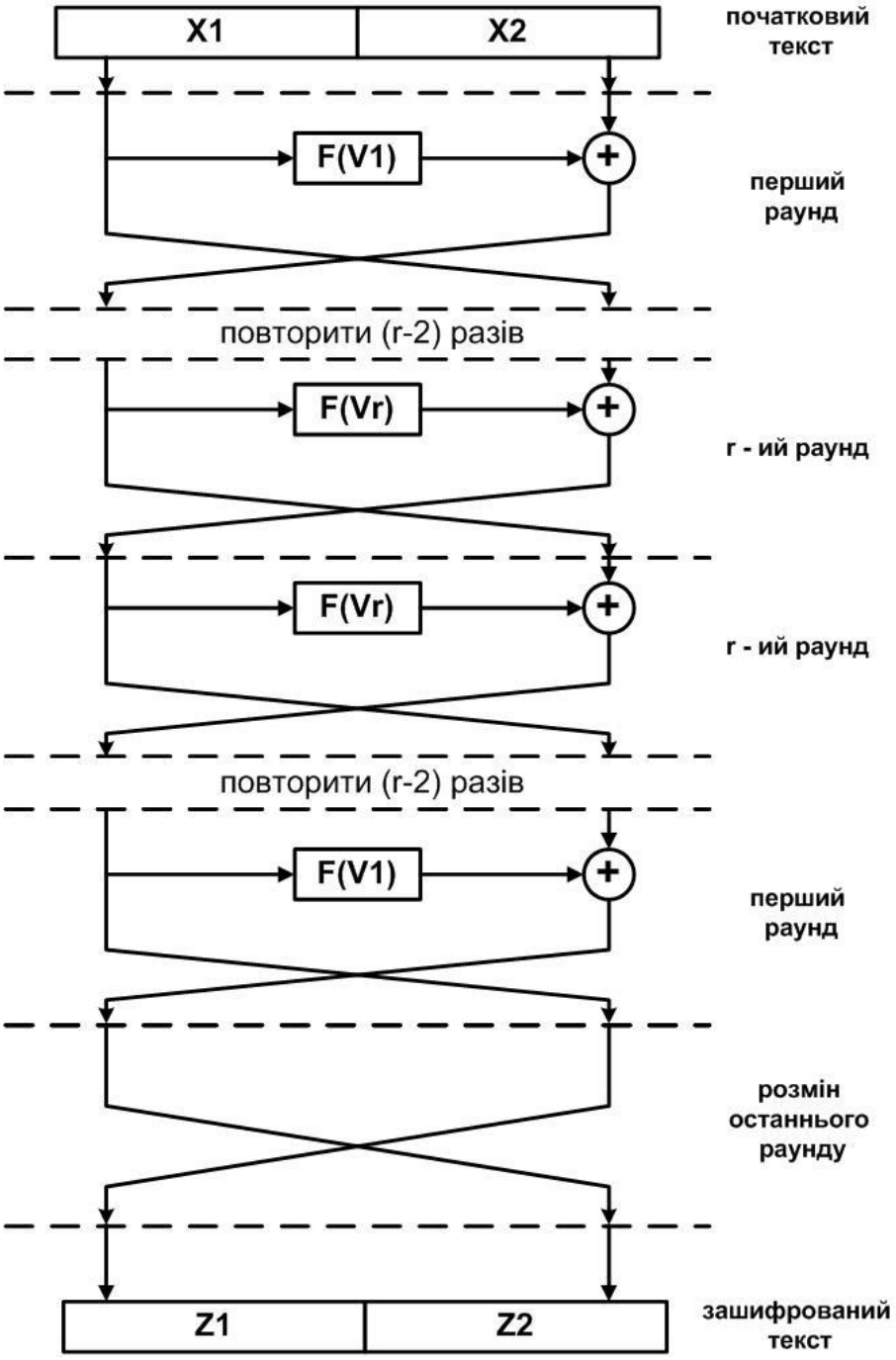


Рисунок 2.5 – Симетрична мережа Фейштеля

Модифікації мережі Фейштеля для більшої кількості гілок застосовують значно частіше. Це пов'язано з тим, що при великих розмірах кодованих блоків (256 і більш біт) стає незручно використовувати математичні функції за модулем 128 і вище. Сьогодні основними одиницями інформації, що оброблюються процесорами – це подвійне машинне слово 64 біти. Тому все частіше і частіше в блокових криптоалгоритмах зустрічаються мережі Фейштеля з чотирма гілками. Найпростіший принцип її модифікації зображений на рис. 2.6 а. Для швидшого перемішування інформації між гілками (це основна проблема мережі Фейштеля з великою кількістю гілок) застосовуються дві модифіковані схеми, звані «type-2» і «type-3». Вони зображені на рис. 2.6 б і рис. 2.6 в.

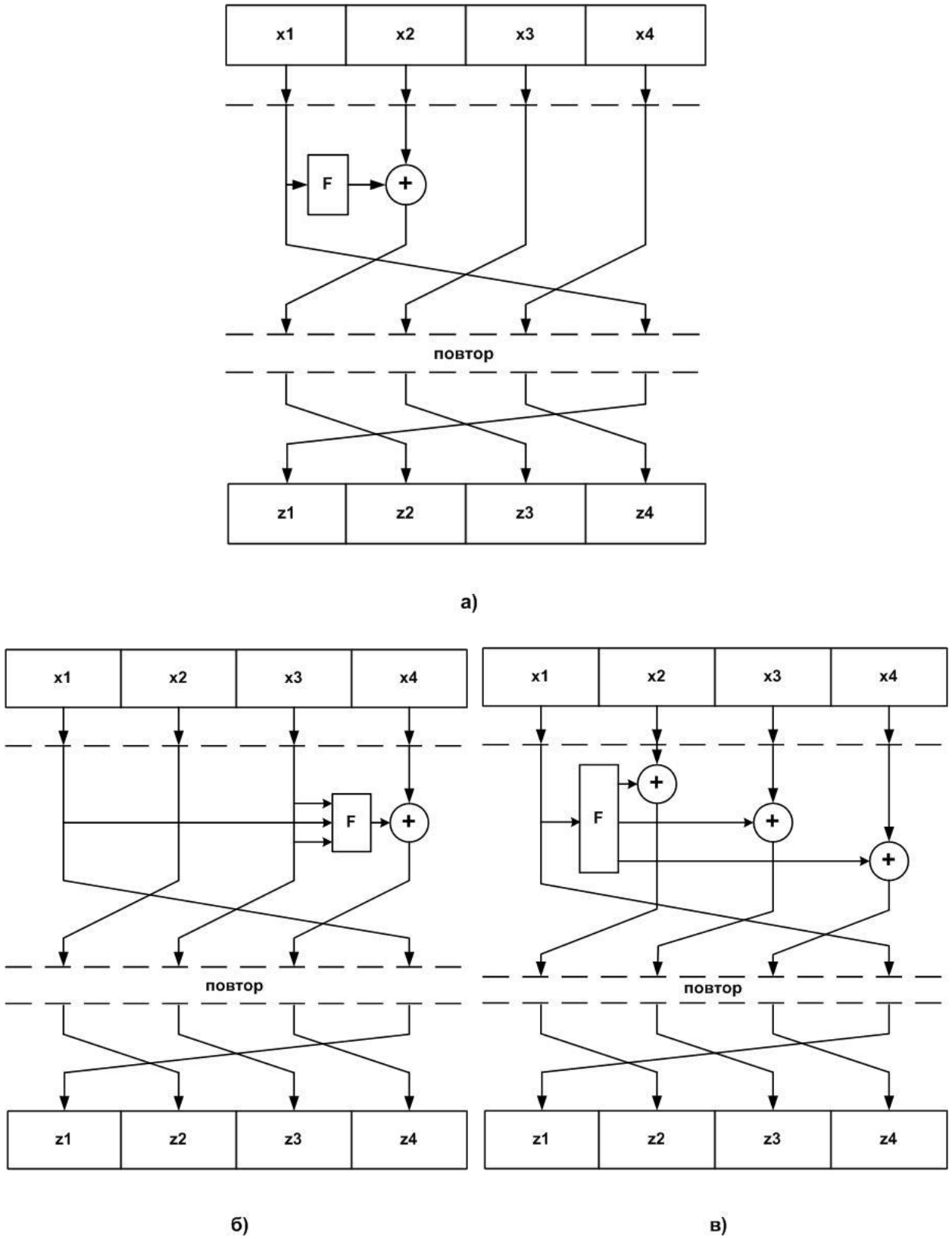


Рисунок 2.6 - Модифікації мережі Фейштеля з чотирма гілками

Мережа Фейштеля надійно зарекомендувала себе як криптостійка схема здійснення перетворень, і її можливо знайти практично в будь-якому сучасному блоковому шифрі. Незначні модифікації стосуються звичайно додаткових початкових і кінцевих перетворень (whitening) над шифрованим блоком. Подібні перетворення, що виконуються за допомогою операції "Виключаюче АБО" чи

додаванням, призначені для підвищення початкової рандомізації вхідного тексту. Таким чином, криптостійкість блокового шифру, що використовує мережу Фейштеля, визначається на 95% функцією F_i правилом обчислення V_i з ключа. Ці функції і є об'єктом досліджень фахівців в області криптографії.

2.3.3 Блоковий шифр TEA

Розглянемо один з найпростіших в реалізації, але визнаний стійкім криптоалгоритм – TEA (Tiny Encryption Algorithm), що має розмір блоку – 64 біт, а довжину ключа – 128 біт.

В алгоритмі використана мережа Фейштеля з двома гілками в 32 біти кожна. Утворююча функція F є зворотною. Мережа Фейштеля несиметрична через використання для накладення не операції "Виключаюче АБО", а арифметичного додавання. Схема роботи алгоритму приведено на рис. 2.7.

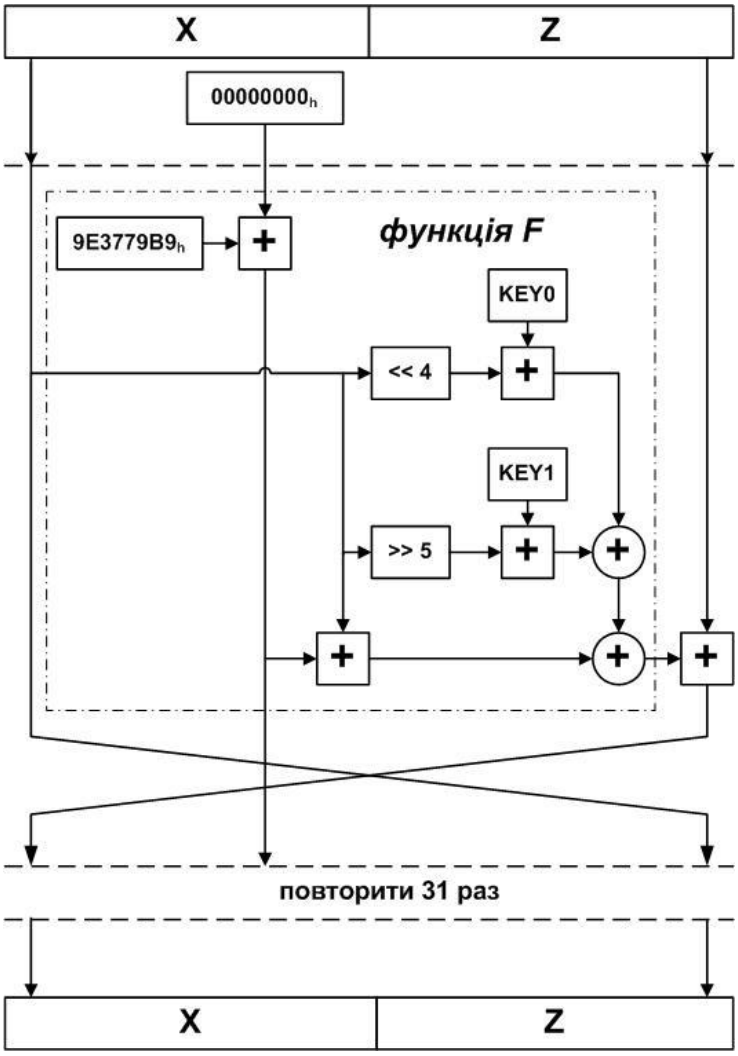


Рисунок 2.7 – Схема алгоритму TEA

Нижче приведений код криптоалгоритма на мові програмування PASCAL.

```
type TLong2=array[0.. 1] longint;
  TLong2x2=array[0.. 1] TLong2;
```

```
const Delta=$9E3779B9;
var   key:TLong2x2;
procedure EnCryptRouting(var data);
var y,z,sum:longint; a:byte;
begin
y:=TLong2(data)[0];z:=TLong2(data)[1];sum:=0;
for a:=0 to 31 do
  begin
  inc(sum,Delta);
  inc(y,((z shl 4)+key[0,0]) xor (z+sum) xor ((z shr 5)+key[0,1]));
  inc(z,((y shl 4)+key[1,0]) xor (y+sum) xor ((y shr 5)+key[1,1]));
  end;
TLong2(data)[0]:=y;TLong2(data)[1]:=z
end;
```

Відмінною рисою криптоалгоритму TEA є його розмір. Простота операцій, відсутність табличних підстановок і оптимізація під архітектуру процесорів дозволяє реалізувати його на мові ASSEMBLER в мінімальному обсязі коду. Недоліком алгоритму є деяка повільність, викликана необхідністю повторювати цикл Фейштеля 32 разів (це необхідно для ретельного "перемішування даних" через відсутність табличних підстановок).

2.3.4 Стандарт блокових шифрів AES

2.3.4.1 Загальні відомості про конкурс AES

У 80-х роках минулого сторіччя в США було прийнято стандарт симетричного криптоалгоритму для внутрішнього використання - DES (Data Encryption Standard), який набув достатньо широке поширення свого часу. Згодом цей стандарт став повністю неприйнятний для використання з двох причин:

1) основна – довжина його ключа складає 56 біт, що надзвичайно мало на сучасному етапі розвитку ЕОМ;

2) другорядна – при розробці алгоритм був орієнтований на апаратну реалізацію, тобто містив операції, виконувані на мікропроцесорах за великий час (напр., такі як перестановка біт усередині машинного слова за певною схемою).

Тому Американський інститут стандартизації NIST – National Institute of Standards & Technology відкрив у 1997 році конкурс на новий стандарт симетричного криптоалгоритму. Для цього було враховано основні недоліки шифру-попередника, а до розробки були підключені найбільші центри з кріптології. Тим самим, переможець цього конкурсу, що отримав назву AES – Advanced Encryption Standard, став би де-факто світовим криптостандартом.

Вимоги, висунуті до кандидатів на AES в 1998 році, були гранично прості:

- алгоритм повинен бути симетричним,
- алгоритм повинен бути блоковим шифром,
- алгоритм повинен мати довжину блоку 128 біт, і підтримувати три довжини ключа: 128, 192 і 256 біт.

Додатково кандидатам рекомендувалося:

- використовувати операції, що легко реалізувати як апаратно (у мікрочіпах), так і програмно (на персональних комп'ютерах і серверах),
- орієнтуватися на 32-розрядні процесори,
- не ускладнювати без необхідності структуру шифру для того, щоб всі зацікавлені сторони були в змозі самостійно провести незалежний криптоаналіз алгоритму і переконатися, що в ньому не створено будь - які недокументовані можливості.

На першому етапі до оргкомітету конкурсу надійшло 15 заявок з абсолютно різних частин світу. Протягом двох років фахівці комітету, досліджуючи самостійно, і вивчаючи публікації інших дослідників, обрали п’ять кращих представників (табл. 2.3), що пройшли до "фіналу" конкурсу.

Таблиця 2.3 – Кращі алгоритми конкурсу AES

Алгоритм	**Розробник**	**Країна**	**Швидкодія (asm, 200МГц)**
MARS	IBM	US	8 МБ/с
RC6	R. Rivest, Co	US	12 МБ /с
Rijndael	V. Rijmen, J. Daemen	BE	7 МБ /с
Serpent	Universities	IS, UK, NO	2 МБ /с
TwoFish	B. Schneier, Co	US	11 МБ /с

Всі ці алгоритми були визнані достатньо стійкими і успішно запобігали всім широко відомим методам криптоаналізу.

У 2000 році NIST оголосив про свій вибір – переможцем конкурсу став бельгійський алгоритм Rijndael. З цього часу стосовно алгоритму - переможця були зняті всі патентні обмеження – його можливо було використовувати в будь-якій криптопрограмі без відрахування жодний коштів розробникам.

Розглянемо основні частини алгоритмів - переможців першого етапу. Повний опис всіх 15 алгоритмів претендентів на AES, включаючи дослідження з їх криптостійкості можливо знайти на сторінці інституту NIST.

2.3.4.2 Шифр MARS

Шифр складається з трьох видів операцій, які повторюються спочатку в прямому, а потім в інверсному порядку. На першому кроці йде класичне вхідне забілення: до всіх байтів початкового тексту додаються байти з матеріалу ключа.

Другий етап: пряме перемішування - однотипна операція, що має структуру мережі Фейштеля повторюється 8 разів. Проте, на цьому етапі не виконується додавання матеріалу ключа. Мета цього перетворення – ретельна рандомізація даних і підвищення стійкості шифру до деяких видів атак (рис. 2.8).

Третій етап: власне шифрування. В ньому використовується мережа Фейштеля третього типу з чотирма гілками, тобто значення трьох функцій, обчислених від однієї гілки накладаються відповідно на три інших, потім йде перестановка машинних слів. Ця операція також повторюється вісім разів (рис.

2.8). Саме на цьому етапі відбувається змішування тексту з основною (більшої) частиною матеріалу ключа.

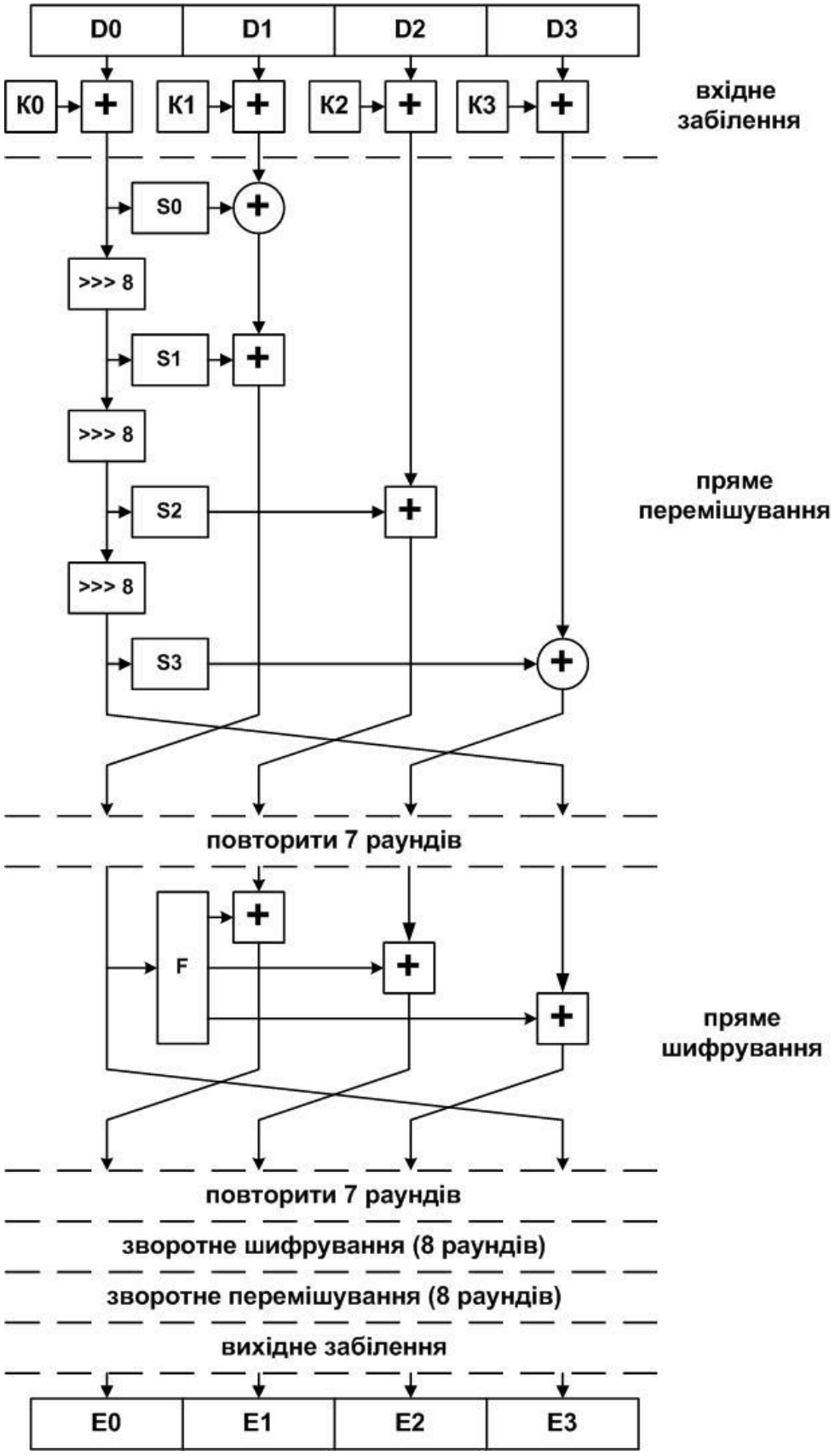


Рисунок 2.8 – Схема алгоритму MARS

Самі функції, що накладаються на гілці, зображені на рис. 2.9. Т. ч., в алгоритмі MARS використані практично всі види операцій, що використовуються в криптографічних перетвореннях: додавання, "Виключаюче АБО", зсув на фіксовану кількість біт, зсув на змінну кількість біт, твір і табличні підстановки.

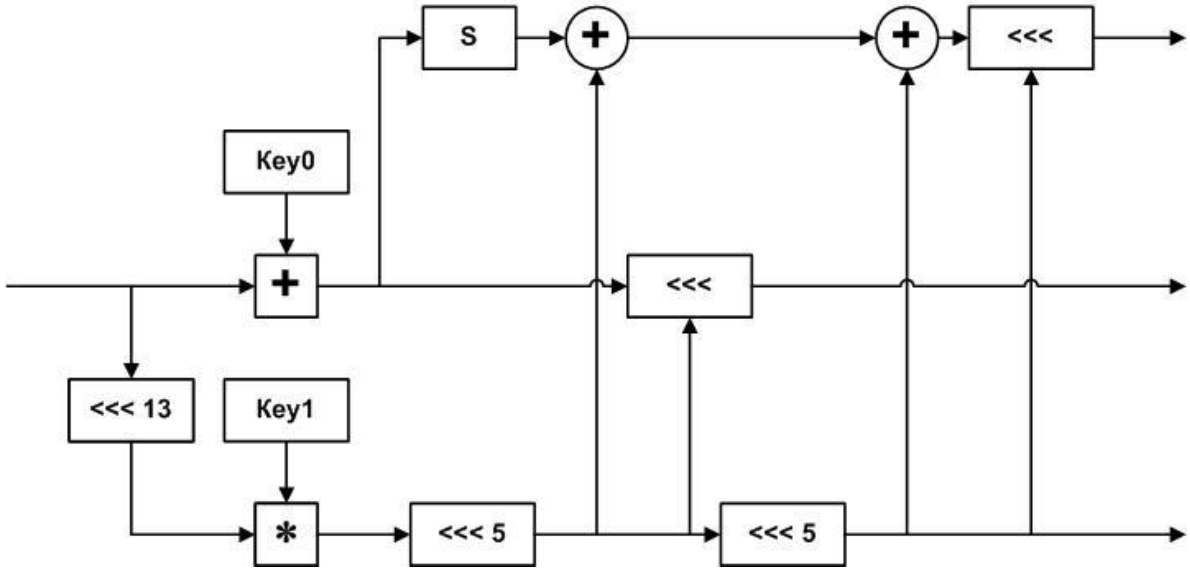


Рисунок 2.9 – Схема утворюючої функції алгоритму MARS

У другій частині операції шифрування повторюються ті ж операції, але в зворотному порядку: спочатку шифрування, потім перемішування, і, нарешті, забілення. При цьому в другі варіанти всіх операцій внесені деякі зміни так, щоб криптоалгоритм в цілому став абсолютно симетричним. Тобто, в алгоритмі MARS для будь-якого X виконується функція EnCrypt(EnCrypt(X)) = X.

2.3.4.3 Шифр RC6

Алгоритм є продовженням криптоалгоритму RC5, розробленого Р. Рівестом – відомим науковцем у галузі криптографії. RC5 був дещо змінений для того, щоб відповідати вимогам AES стосовно довжині ключа і розміру блоку. При цьому алгоритм став ще швидшим, а його ядро, що було використано в RC5, мав великий резерв досліджень, проведених задовго до оголошення конкурсу AES.

Алгоритм є мережею Фейштеля з чотирма гілками змішаного типу: в ньому два парні блоки використовуються для одночасної зміни вмісту двох непарних блоків. Потім здійснюється звичний для мережі Фейштеля зсув на одне машинне слово, що змінює парні і непарні блоки місцями. Алгоритм є простим і його структуру наведено на рис. 2.10. Розробники рекомендують використовувати при шифруванні 20 раундів мережі, хоча у принципі їх кількість не регламентується. При 20 повторах операції шифрування алгоритм має найвищу швидкість серед 5 фіналістів AES.

Перетворення T(X) є дуже простим:

$$T(X)=(X \times (X + 1)) \bmod 2^N.$$

Воно використовується як нелінійне перетворення з високими показниками перемішування бітового значення вхідної величини X.

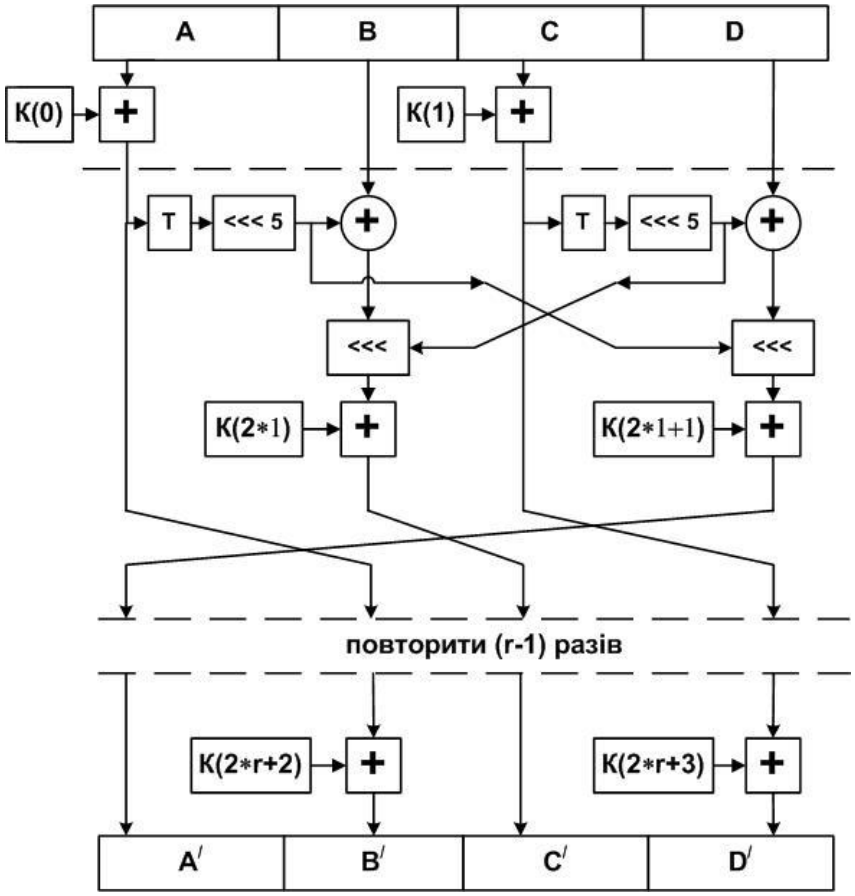


Рисунок 2.10 – Схема алгоритму RC6

2.3.4.4 Шифр Serpent

Алгоритм розроблений групою фахівців з декількох дослідницьких центрів світу. Алгоритм є мережою Фейштеля для чотирьох гілок змішаного типу (рис. 2.11): дві парні гілки змінюють сумісно значення непарних, потім змінюються позиціями. У якості криптоперетворювань використовуються операції "Виключаюче АБО", табличні підстановки і бітові зсуви. Алгоритм складається з 32-х раундів. Власно раунди складені таким чином, що додавання до гілок матеріалу ключа на першому і останньому раундах утворює вхідне і вихідне забілення.

2.3.4.5 Шифр TwoFish

Цей алгоритм було розроблено у Counterpain Security Systems на голові з Б. Шнайером. Попередня програмна розробка цієї фірми - BlowFish на той час була визнаним криптостійким алгоритмом.

У алгоритмі TwoFish розробники залишили деякі вдалі рішення з проекту-попередника. Крім цього було виконано ретельні дослідження з перемішування даних в мережі Фейштеля. Алгоритм є мережею Фейштеля змішаного типу: перша і друга гілка на непарних раундах здійснюють модифікацію третьої і

четвертої гілок, а на парних раундах виконуються протилежні зміни (рис. 2.12). В алгоритмі також використовується криптоперетворювання Адамара (PHT - Pseudo-Hadamar Transform) – зворотне арифметичне складання першого потоку з другим, а потім другого з першим.

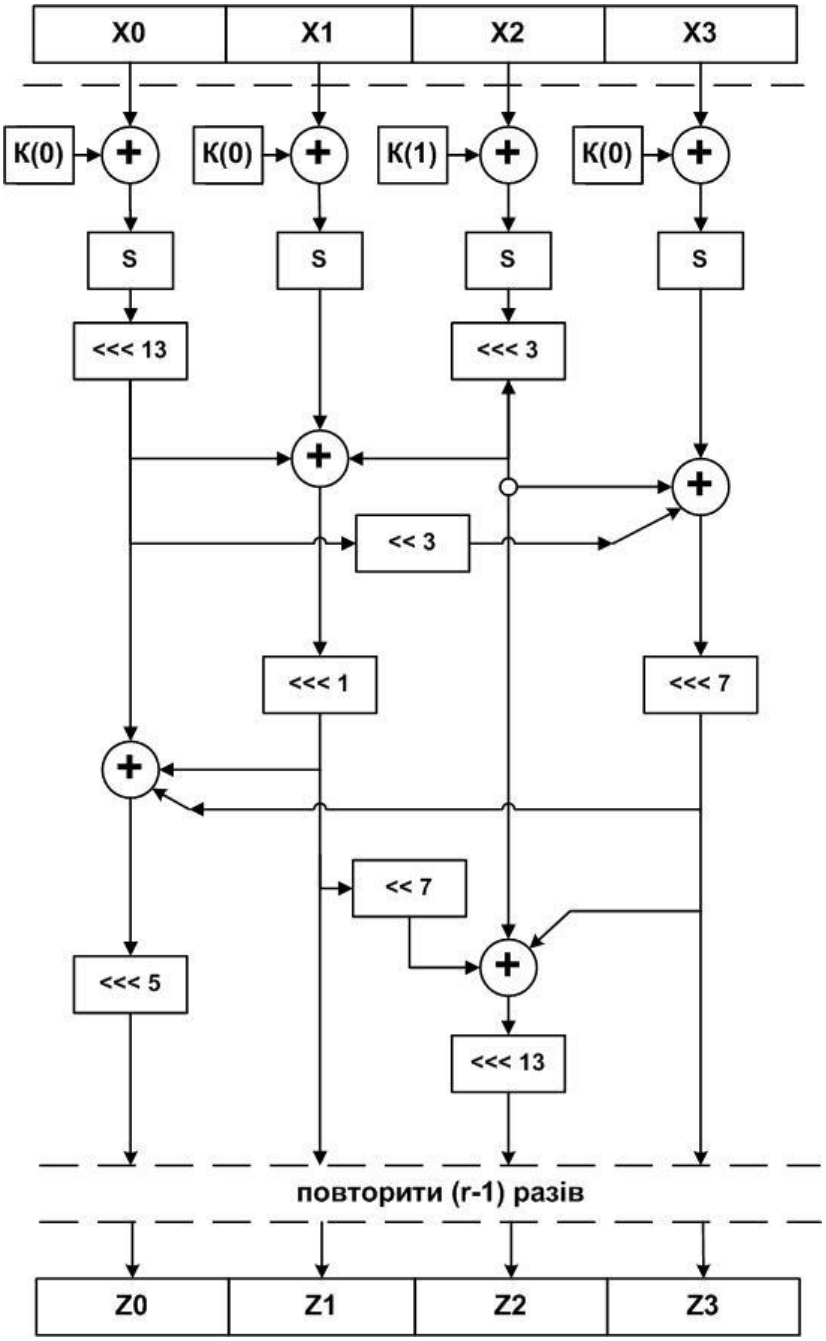


Рисунок 2.11 – Схема алгоритму Serpent

Єдиним зауваженням щодо алгоритму TwoFish від незалежних дослідників, було те, що при розширенні матеріалу ключа в алгоритмі використовується саме же алгоритм. Подвійне використання блокового шифру значно ускладнює його аналіз щодо наявності слабких ключів або недокументованих замаскованих зв'язків між вхідними і вихідними даними.

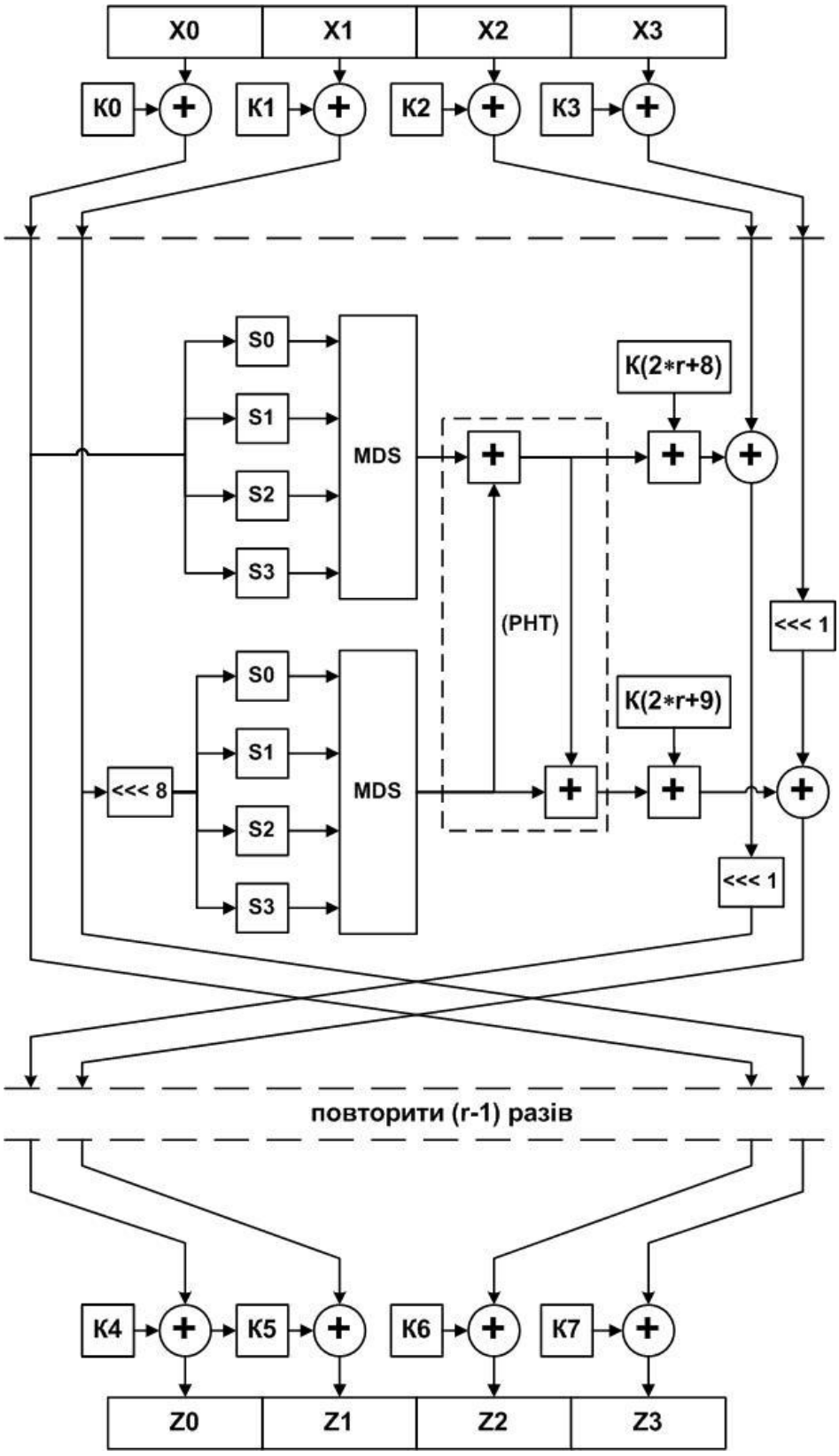


Рисунок 2.12 – Схема алгоритму TwoFish

2.3.4.6 Шифр Rijndael

Цей алгоритм було розроблено двома бельгійськими фахівцями з криптографії. Він є нетрадиційним блоковим шифром, оскільки не використовує мережу Фейштеля для криптоперетворювань. Алгоритм подає кожний блок кодованих даних у вигляді двовимірного масиву розміром 4x4, 4x6 або 4x8 байт

залежно від встановленої довжини блоку. Далі на відповідних етапах перетворення здійснюються або над незалежними стовпцями, або над незалежними рядками, або взагалі над окремими байтами в таблиці.

Всі перетворення в шифрі мають математичне обґрунтовування. Структура і послідовність операцій дозволяють виконувати такий алгоритм ефективно на процесорах різної розрядності. В структурі алгоритму закладена можливість паралельного виконання деяких операцій, що на багатопроцесорних робочих станціях може підняти швидкість шифрування ще в 4 рази.

Алгоритм складається з деякої кількості раундів (від 10 до 14 – це залежить від розміру блоку і довжини ключа), в яких послідовно виконуються наступні операції:

- ByteSub – таблична підстановка 8x8 біт (рис. 2.13);

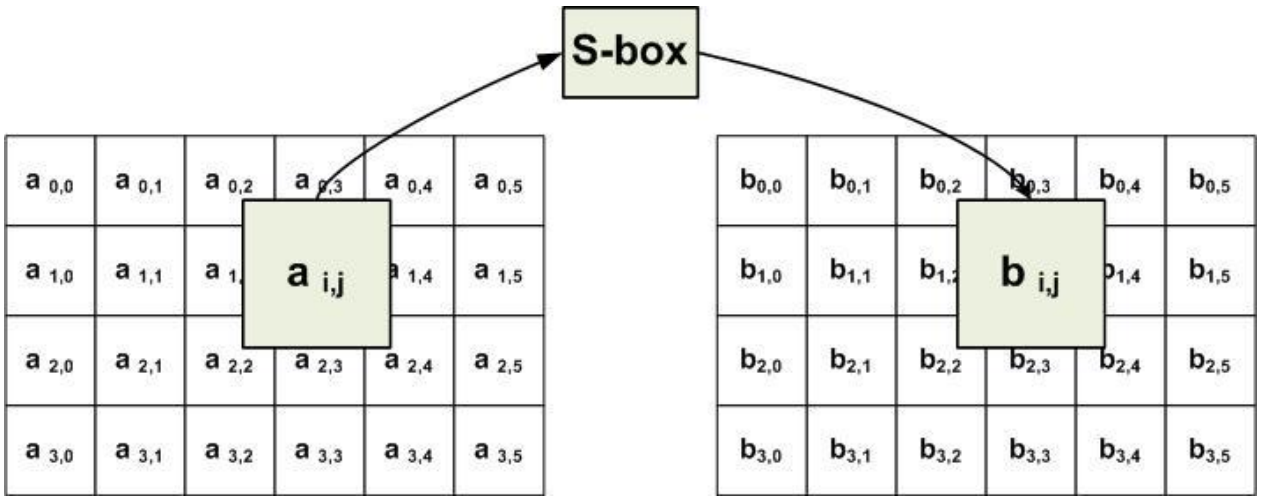


Рисунок 2.13 – Схема табличної підстановки алгоритму Rijndael

- ShiftRow – зсув рядків в двовимірному масиві на різну кількість елементів (рис. 2.14);

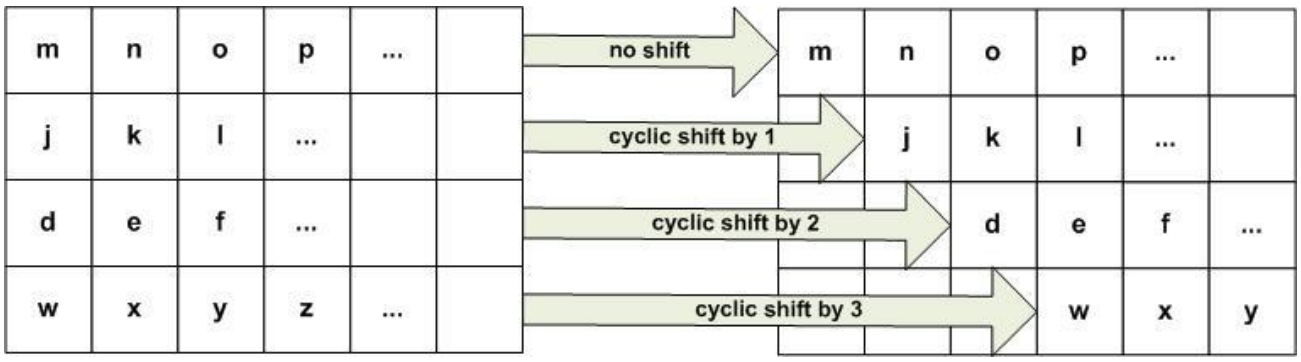


Рисунок 2.14 – Схема зсувів у алгоритмі Rijndael

- MixColumn – математичне перетворення, що перемішує дані усередині стовпця (рис. 2.15);

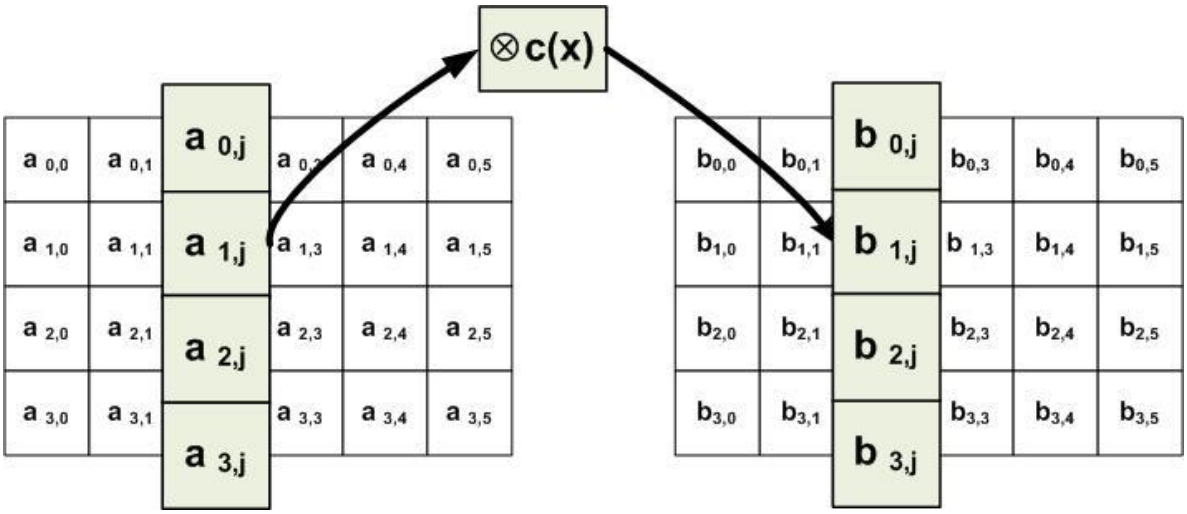


Рисунок 2.15 – Схема математичних перетворювань алгоритму Rijndael

- AddRoundKey – додавання матеріалу ключа операцією XOR (рис. 2.16).

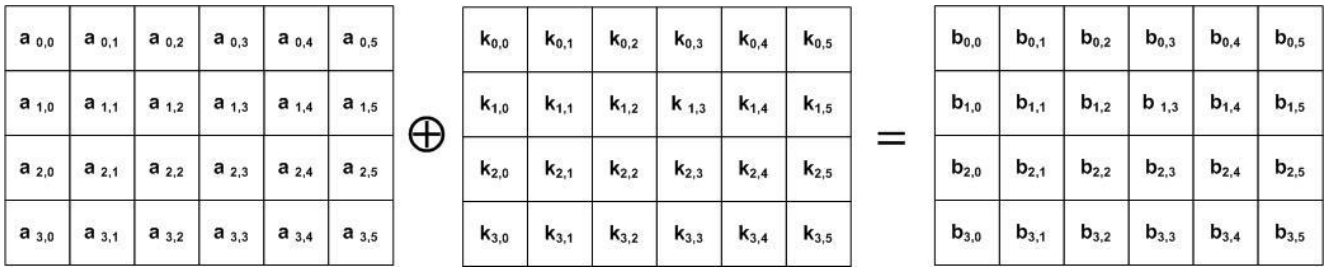


Рисунок 2.16 – Додавання матеріалу ключа у алгоритмі Rijndael

У останньому раунді операція перемішування стовпців відсутня, що надає симетричності всієї послідовності операцій.

2.4 Симетричні криптосистеми

2.4.1 Функції криптосистем

Криптоалгоритми є методами перетворення невеликого блоку даних (від 4 до 64 байт) в закодований вигляд залежно від заданого двійкового ключа, що є основою криптографічних систем, але їх безпосереднє використання без яких-небудь модифікацій для кодування великих об'ємів даних не є оптимальним.

Всі недоліки безпосереднього використання криптоалгоритмів усуваються в криптосистемах. Криптосистема – це завершена комплексна модель, здатна здійснювати двосторонні криптоперетворення над даними довільного об'єму і підтверджувати час відправки повідомлення, яка володіє механізмом перетворення паролів і ключів, а також системою транспортного кодування. Таким чином, криптосистема виконує три основні функції:

- посилення захищеності даних;
- полегшення роботи з криптоалгоритмом з боку користувача;

- забезпечення сумісності потоку даних з іншим програмним забезпеченням.

Конкретна програмна реалізація криптосистеми називається криптопакетом.

2.4.2 Алгоритми створення ланцюжків

Перша задача, яка вирішується при шифруванні даних криптоалгоритмом – це дані з довжиною, що не дорівнюють довжині одного блоку криптоалгоритму. Така ситуація відбувається практично завжди.

Перше проблема: що можливо зробити, якщо необхідно зашифрувати 24 байт тексту, при тому що використовується криптоалгоритм з довжиною блоку 8 байт ? Можливо було б послідовно зашифрувати три рази по 8 байт і скласти їх у вихідний файл так, як вони містились в початковому файлі. Але якщо даних багато і деякі блоки по 8 байт повторюються, це значить, що у вихідному файлі ці ж блоки будуть зашифровані однаково - це призводить до зниження криптостійкості.

Друга проблема: якщо у файлі даних не 24, а 21 байт. Можливо було б не шифрувати останні 5 байт або чимось заповнювати ще 3 байти, а потім при дешифруванні їх відкидати. Перший варіант взагалі є неприємним, а другий застосовується, але виникає питання чим заповнювати останні байти?

Для вирішення цих проблем і були введені в криптосистеми алгоритми створення ланцюжків (chaining modes). Найпростіший метод - метод ECB (Electronic Code Book) , де шифрований файл тимчасово поділяється на блоки, рівні блокам алгоритму, кожний з яких шифрується незалежно, а потім із зашифрованих пакетів даних компонується в тієї ж послідовності файл, який буде надійно захищений криптоалгоритмом. Назву алгоритм одержав через те, що він широко застосовувався в портативних пристроях для шифрування – електронних шифрокніжках. Схема такого методу приведена на рис. 2.17.

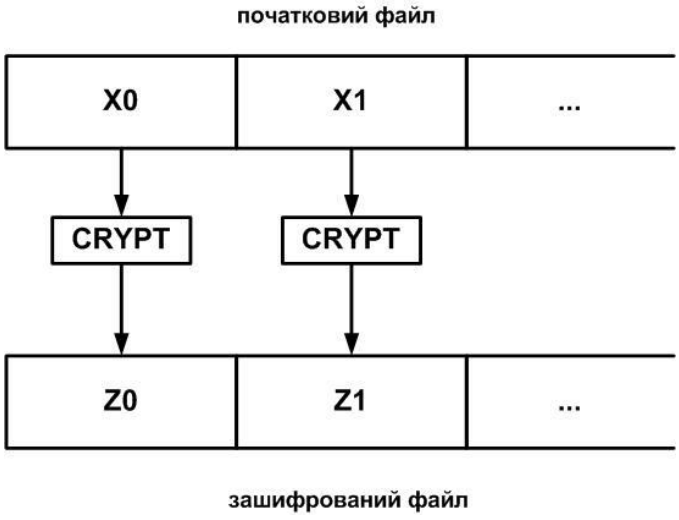


Рисунок 2.17 – Схема методу ECB

У тому випадку, коли довжина пакету інформації, що пересилається, не кратна довжині блоку криптоалгоритму можливо розширення останнього

(неповного) блоку байт до необхідної довжини або ж за допомогою генератора псевдовипадкових чисел, що не завжди безпечне відносно крипостійкості, або ж за допомогою хеш-суми інформації, що шифрується. Другий варіант більш переважним, оскільки хеш-сума володіє кращими статистичними показниками, а її апріорна відомість сторонній особі рівносильна знанню нею всієї інформації, що шифрується.

Вказаним вище недоліком цієї схеми є те, що при повторі в початковому тексті однакових символів протягом більш, ніж 2N байтів (де N – розмір блоку криптоалгоритму), у вихідному файлі будуть присутні однакові зашифровані блоки. Тому, для підсилення захисту великих пакетів інформації за допомогою блокових шифрів застосовуються декілька зворотних схем «створення ланцюжків». Всі вони майже рівнозначні за крипостійкістю, але кожна має деякі переваги і недоліки, залежні від виду початкового тексту.

Всі схеми створення ланцюжків засновані на принципах залежності результуючого зашифровуваного блоку або від попередніх блоків, або від позиції блоку в початковому файлі. Це досягається за допомогою блоку «пам'яті» – пакету інформації з довжиною, що дорівнює довжині блоку алгоритму. Блок пам'яті (до нього застосовують термін IV – Initial Vector) обчислюється за певним принципом зі всіх попередніх зашифрованих блоків, а потім накладається за допомогою якої-небудь зворотної функції (звичайно - XOR) на оброблюваний текст на одній із стадій шифрування. В процесі розкодування на приймальному боці операція створення IV повторюється на основі прийнятого і розшифрованого тексту, унаслідок чого алгоритми створення ланцюжків повністю зворотні.

Два найпоширеніших алгоритми створення ланцюжків – CBC і CFB, їх структура наведена на рис. 2.18 і рис. 2.19 відповідно. Метод CBC одержав назву від абревіатури Cipher Block Chaining – об'єднання в ланцюжок блоків шифру, а метод CFB – від абревіатури Cipher FeedBack – зворотний зв'язок за шифроблоком.

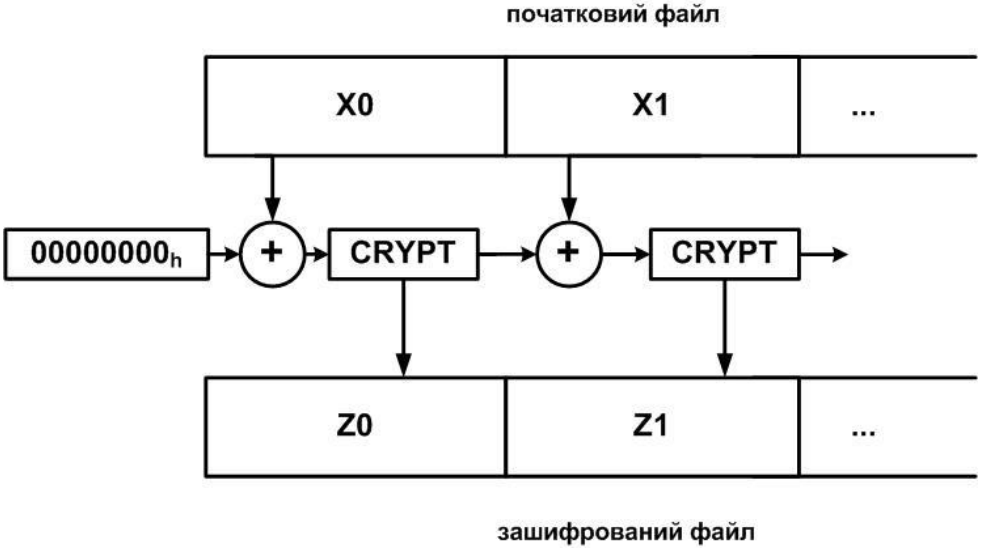


Рисунок 2.18 – Схема методу CBC

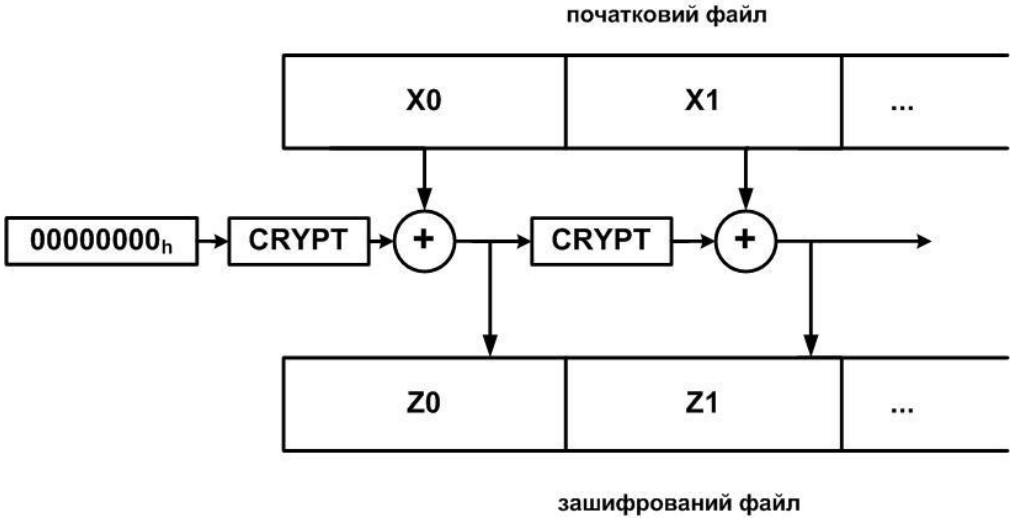


Рисунок 2.19 – Схема методу CFB

Ще один метод створення ланцюжків отримав назву OFB (Output FeedBack – зворотний зв'язок за виходом) має дещо іншу структуру (рис. 2.20): в ньому значення, що накладається на шифрований блок, не залежить від попередніх блоків, а залежить лише від позиції шифрованого блоку (в цьому значенні він повністю відповідає скремблерам), і через це він не поширює помилки на подальші блоки.

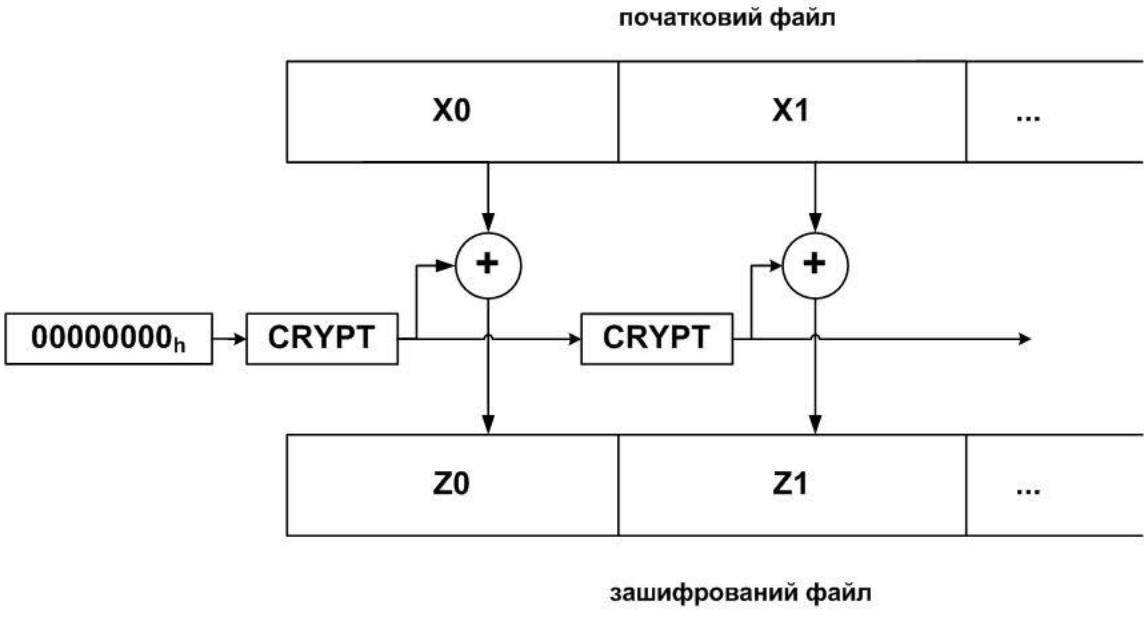


Рисунок 2.20 – Схема методу OFB

Т.ч., всі алгоритми створення ланцюжків є однозначно відновлюваними. Порівняльна характеристика методів створення ланцюжків наведена у табл. 2.4.

Таблиця 2.4 – Порівняльна характеристика методів створення ланцюжків

Метод	Шифрування блоку залежить від …	Спотворення одного біту при передаванні …	Чи кодується некратне блоку число байт без доповнення	На вихід криптосистеми надходить …
ECB	поточного блоку	псує весь поточний блок	ні	вихід криптоалгоритму
CBC	всіх попередніх блоків	псує весь поточний і всі подальші блоки	ні	вихід криптоалгоритму
CFB	всіх попередніх блоків	псує один біт поточного блоку і всі подальші блоки	так	XOR маска з початковим текстом
OFB	позиції блоку у файлі	псує тільки один біт поточного блоку	так	XOR маска з початковим текстом

2.4.3 Методи рандомізації повідомлень

2.4.3.1 Огляд методів рандомізації повідомлень

Ще одним вдосконаленням, спрямованим на підвищення стійкості всієї криптосистеми в цілому є створення ключів сеансу. Ця операція необхідна в тих випадках, коли здійснюється багаторазове шифрування схожих блоків даних одним і тим же ключем. Наприклад, це має місце при передаванні інформації або команд в автоматизованих системах управління, в банківських операціях, кінцевих пристроях Інтернету речей і багатьох інших випадках передачі інформації, що має певний, наперед відомий формат.

У таких випадках для підвищення криптостійкості необхідне введення деякої випадкової величини в процес шифрування. Це можливо зробити декількома способами:

- записом на початок файлу даних псевдовипадкової послідовності байт наперед обумовленої довжини з відкиданням її при дешифруванні – цей метод є працездатним тільки при використанні алгоритмів створення ланцюжків з пам'яттю (CBC, CFB, OFB);
- використанням модифікованих алгоритмів створення ланцюжків, які при шифруванні кожного блоку змішують з ним або фіксовану випадкову величину, прикріплену до початку зашифрованого файлу, або значення, що обчислюється за допомогою того ж шифру і ключа від наперед обумовленої величини;
- створенням спеціально для кожного файлу абсолютно випадкового ключа, так званого ключа сеансу, яким і шифрується весь файл (сам же ключ сеансу шифрується первинним ключем, так званим майстер-ключем і розміщується на початку зашифрованого файлу).

Всі ці способи принципово не мають очевидних недоліків, але через більшу опрацьованість останнього методу частіше застосовується саме він.

2.4.3.2 Генератори випадкових і псевдовипадкових послідовностей

Найбільша проблема всіх методів рандомізації повідомлень – це породження дійсно випадкової послідовності біт. Проблемою при цьому є те, що генератори випадкових послідовностей, що використовуються для загальних цілей, наприклад, в мовах програмування, у дійсності є псевдовипадковими генераторами. Це пояснюється тим, що у принципі існує кінцева, а не нескінченна множина станів ЕОМ, і, як би складно не формувалося в алгоритмі число, воно все одне має відносно небагато біт інформаційної насиченості.

Найчастіше в прикладних задачах випадковий (псевдовипадковий) результат формують з лічильника тиків – системного годинника. В цьому випадку дані про поточну годину несуть приблизно 32 біт інформації, значення лічильника тиків – ще 32 біт. Це дає можливість отримати 64 біт інформації. На сьогоднішній день межею стійкої криптографії є значення в 40 біт, при реальних довжинах ключів в 128 біт. Т. ч., подібний метод є неоптимальним. До 64 біт можливо додати ще 32 біт з надшвидкого таймеру і цього ще недостатньо. Крім того, навіть якщо і можливо набрати довжину ключа в 128 біт (що дуже сумнівне), він матиме псевдовипадковий характер, оскільки буде засновано на стані тільки такої ЕОМ на момент початку шифрування. Джерелами по-справжньому випадкових величин можуть бути тільки зовнішні об'єкти, наприклад, людина.

Два найчастіше поширених методів створення випадкових послідовностей за допомогою людини засновані на введенні з клавіатури. В обох випадках користувачу пропонується, не замислюючись, натиснути на клавіатурі безглузді поєднання літер.

За першим методом над власне введеними значеннями виконуються дії, що підвищують випадковість вихідного потоку. Так, наприклад, обов'язково видаляються перші 3 біт введеного ASCII символу, часто видаляються ще один перший і ще один останній біти. Потім, обсяг одержаної послідовності зменшується ще в три рази накладенням першого і другого біта, на третій, операцією XOR. Це генерує достатньо випадкову послідовність біт.

За другим методом введені символи алгоритм не обробляє, проте запам’ятовує інтервали часу, через які відбулися натиснення клавіш. Запис моментів проводиться за відліками швидкого системного таймера або внутрішньому лічильника процесора, що працює на частоті ядра. Оскільки перші і молодші біти мають певну кореляцію між символами (перші через фізичні характеристики людини, а молодші через особливості операційної системи), то вони відкидаються (звичайно видаляються 0 - 8 старших біта і 4 - 10 молодших).

Також як варіанти методів, що менш розповсюджені, можливо виділити комбінацію обох клавіатурних методів і метод, заснований на маніпуляторі "миша", що отримує випадкову інформацію із зсувів користувачем курсору миші.

У потужних криптосистемах військового використання застосовуються дійсно випадкові генератори чисел, засновані на фізичних процесах. Вони є платою, або зовнішніми пристроями, що підключаються до ЕОМ через порт введення - виведення. При цьому двома основними джерелами білого Гауссовського шуму є високоточне вимірювання теплових флуктуацій і запис радіоефіру на частоті, вільній від радіомовлення.

2.4.4 Архівація

2.4.4.1 Загальні принципи архівації

Більшість сучасних форматів запису даних містять їх у вигляді, зручному для швидкого маніпулювання, для зручного прочитання користувачами. При цьому дані займають об'єм більший, ніж це дійсно потрібно для їх зберігання. Алгоритми, які усувають надмірність запису даних, називаються алгоритмами стиснення даних, або алгоритмами архівації. Сьогодні існує багато програм для стиснення даних, заснованих на декількох основних способах.

В сучасному криптоаналізі, тобто науці про протистояння криптографії, з доведено, що ймовірність злому криптосхеми за наявності кореляції між блоками вхідної інформації значно вище, ніж за відсутності такої. Тому алгоритми стиснення даних мають своєю основною задачею усунення надмірності, тобто кореляцій між даними у вхідному тексті.

Всі алгоритми стиснення даних якісно поділяються на:

- алгоритми стиснення без втрат, при використанні яких дані відновлюються без щонайменших змін;
- алгоритми стиснення з втратами, які видаляють з потоку даних інформацію, що незначно впливає на зміст даних, або взагалі на інформацію, що не сприймається людиною (такі алгоритми розроблені тільки для аудіо- і відеозображень).

В криптосистемах використовується тільки перша група алгоритмів.

Існує два основні методи архівації без втрат:

- алгоритм Хаффмана (Huffman), орієнтований на стиснення послідовностей байт, не зв'язаних між собою,
- алгоритм Лемпеля - Зіва (Lempel, Ziv), орієнтований на стиснення будь-яких видів текстів, що використовує факт неодноразового повторення «слів» – послідовностей байт.

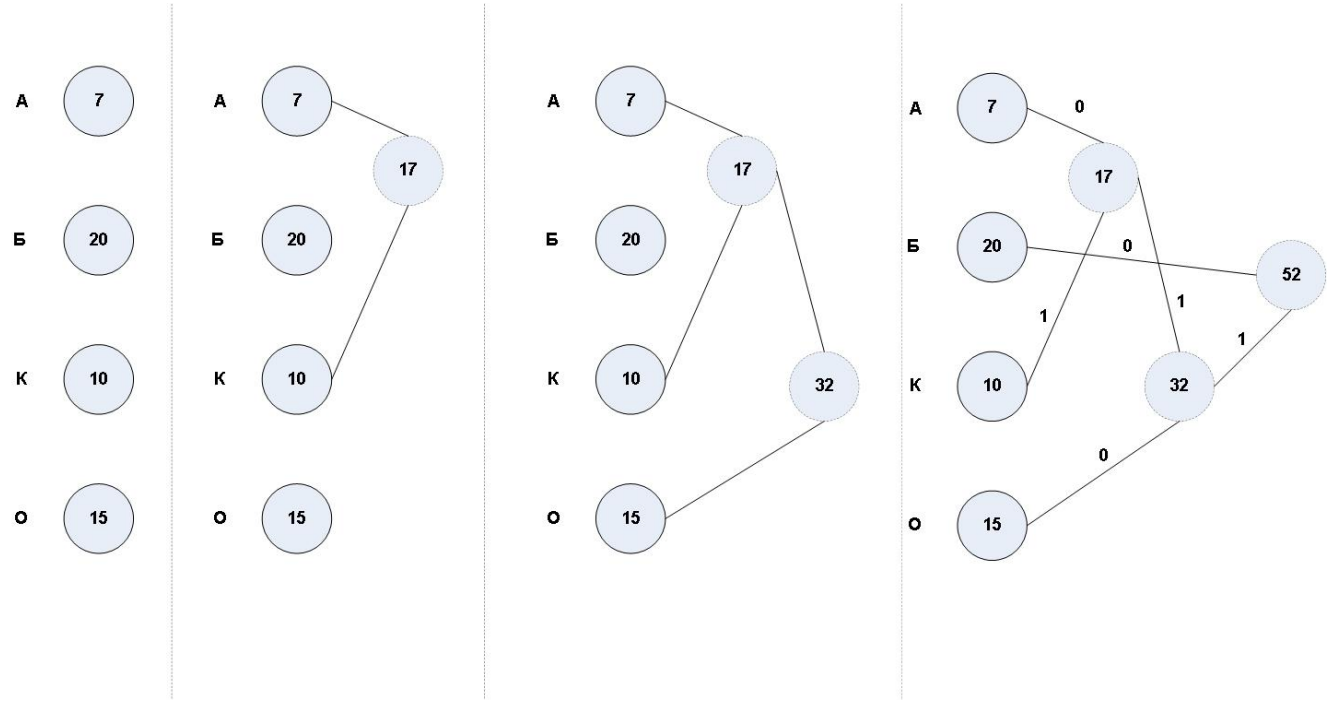
Практично всі відомі програми архівації без втрат (ARJ, RAR, ZIP і т. і.) використовують поєднання цих двох методів – алгоритм LZH.

2.4.4.2 Алгоритм Хаффмана

Алгоритм заснований на тому факті, що деякі символи із стандартного 256-символьного набору в довільному тексті можуть зустрічатися частіше за середній період повторювання, а інші, відповідно, – рідше. Отже, якщо для запису поширених символів використовувати короткі послідовності біт, завдовжки менше 8, а для запису рідких символів – довгі, то сумарний об'єм файлу зменшиться.

Хаффманом було запропоновано простий алгоритм визначення того, який символ необхідно визначати яким кодом для отримання файлу з довжиною, дуже близькою до його ентропії (тобто інформаційної насиченості). Наприклад, є перелік всіх символів, що зустрічаються в початковому тексті, причому відома частота появи кожного символу в ньому. Перелічимо їх вертикально в стовпчик у вигляді осередків майбутнього графа за правою межею аркушу (рис. 2.21 а). Оберемо два символи з якнайменшою кількістю повторень в тексті (якщо три або

більша кількість символів мають однакові значення, обираємо будь-які два з них). Проведемо від них лінії вліво до нової вершини графа і позначимо в неї значення, що дорівнює сумі частот повторення кожного з об'єднуваних символів (рис. 2.21 б). Далі не братимемо до уваги при пошуку якнайменших частот повторення двох об'єднаних вузлів (для цього видалімо числа в цих двох вершинах), але розглядатимемо нову вершину як повноцінний осередок з частотою появи, рівній сумі частот появи двох вершин, які було поєднано. Операція поєднання вершин здійснюється до тих пір, поки не залишиться одна вершина (рис. 2.21 в, г), де буде записано довжину кодованого файлу.



а б в г

Рисунок 2.21 – Приклад отримання коду Хаффмана

Далі, кожне з двох ребер, що йдуть з певної вершини графа, позначають «1» або «1»: ребро, яке йде до вершини з більшою частотою ніж інша вершина, позначається через «1», інше ребро – відповідно через «1». У випадку коли вершина поєднує дві вершини з однаковою частотою порядок кодування дуг цієї вершини не має значення.

Для визначення коду кожного певного символу необхідно записати шлях від вершини дерева до вершини символу, виписуючи нулі і одиниці за маршрутом проходження. На рис. 2.21 символ «А» отримує код «000», символ «Б» – код «01», символ «К» – код «001», а символ «О» – код «1».

У теорії кодування інформації доведено, що код Хаффмана є префіксним, тобто код жодного символу не є початком коду будь-якого іншого символу. З цього слід, що код Хаффмана є однозначно відновлюваним одержувачем, навіть якщо не зазначається довжина коду кожного символу, який передається. Одержувачу пересилають тільки дерево Хаффмана в компактному вигляді, а потім вхідна послідовність кодів символів декодується їм самостійно будь-якої додаткової інформації. Наприклад, при прийманні «0100010100001» спочатку

відокремлюється перший символ «Б» : «01-00010100001», потім знову починаючи з вершини дерева – «А» «01-000-10100001», потім аналогічно декодується весь запис «01-000-1-01-000-01» «БАОБАБ».

2.4.4.3 Алгоритм Лемпеля - Зіва

Класичний алгоритм Лемпеля - Зіва – LZ77, названий так за роком своєї публікації, є простим. Він формулюється наступним чином: «якщо в вихідному потоці даних, що пройшов раніше, вже зустрічалася подібна послідовність байт, причому запис про її довжину і зсув від поточної позиції коротший ніж сама ця послідовність, то у вихідний файл записується посилання (зсув, довжина), а не сама послідовність». Так послідовність «ВОРОНА_ПРОВОРОНИЛА_ВОРОНЕНЯ» закодується як «ВОРОНА_ПРО(10,5)ИЛА_(9,5)ЕНЯ».

Також використовується подібний до цього метод стиснення RLE (Run Length Encoding), який полягає в записі замість послідовності однакових символів одного символу і їх кількості. Наприклад, послідовність «АААААААА» за допомогою алгоритму RLE буде закодовано як «(А,7)», в той же час її можливо стиснути і за допомогою алгоритму LZ77: «А(1,6)». При цьому ступінь стиснення саме такої послідовності алгоритмом LZ77 буде меншою (приблизно на 30-40%), але метод Лемпеля - Зіва є більш універсальним, і може набагато краще обробляти послідовності, які взагалі нестискаються методом RLE.

2.4.5 Хешування паролів

Хешування паролів – дозволяє користувачам запам'ятовувати не 128 байт, тобто 256 шістнадцяткових цифр ключа, а деякий осмислений вираз, слово або послідовність символів, що називається паролем. При розробці будь-якого криптоалгоритму слід враховувати те, що часто кінцевим користувачем системи є людина, а не автоматична система. Це ставить питання про те чи, зручно, і чи взагалі реально людині запам'ятати 128-бітний ключ (32 шістнадцяткові цифри). Межа запам’ятовування складає не більш 8-12 подібних символів, а, отже, якщо користувач буде оперувати саме ключем, то він буде примушуватись до запису ключа на якому-небудь аркуші паперу або електронному носії, наприклад, в текстовому файлі. Це, звісно, різко знижує захищеність системи.

Для вирішення цієї проблеми були розроблені методи, що перетворюють осмислений рядок довільної довжини – пароль, у вказаний ключ наперед заданої довжини. В переважній більшості випадків для цієї операції використовуються так звані хеш-функції (hashing – дрібна нарізка і перемішування). Хеш-функцією називається таке математичне або алгоритмічне перетворення заданого блоку даних, яке володіє наступними властивостями:

1. хеш-функція має нескінченну область визначення;
2. хеш-функція має кінцеву область значень;
3. вона незворотна;
4. зміна вхідного потоку інформації на один біт змінює близько половини всіх біт вихідного потоку, тобто результату хеш-функції.

Ці властивості дозволяють подавати на вхід хеш-функції паролі, тобто текстові рядки довільної довжини на будь-якій національній мові і, обмеживши

область значень функції діапазоном $0…2^N-1$, де N – довжина ключа в бітах, отримати на виході достатньо рівномірно розподілені за областю значення блоки інформації – ключі. Властивості, подібні 3 і 4 пунктам вимог до хеш-функції, здійснюють блокові шифри. Це передбачає один з можливих шляхів реалізації стійких хеш-функцій – проведення блокових криптоперетворень над матеріалом рядка-пароля. Такий метод і використовується в різних варіаціях практично у всіх сучасних криптосистемах.

Матеріал рядка-пароля багато разів послідовно використовується як ключ для шифрування деякого наперед відомого блоку даних – на виході виходить зашифрований блок інформації, однозначно залежний тільки від пароля і при цьому він має достатньо добрі статистичні характеристики. Такий блок або декілька таких блоків і використовуються як ключ для подальших криптоперетворень. Характер використання блокового шифру для хешування визначається відношенням розміру блоку криптоалгоритма і розрядності необхідного хеш-результату, що застосовується. Якщо вказані вище величини співпадають, то використовується схема одноланцюгового блокового шифрування. Первинне значення хеш-результату H_0 встановлюється рівним 0, весь рядок-пароль поділяється на блоки байт, рівні за довжиною ключу блокового шифру, що використовується для хешування, потім виконується перетворення за рекурентною формулою:

$$H_j=H_{j-1} \text{ XOR } EnCrypt(H_{j-1},PSW_j),$$

де EnCrypt(X,Key) – блоковий шифр, що використовується (рис. 2.22).

Останнє значення H_k використовується як результат хешування.

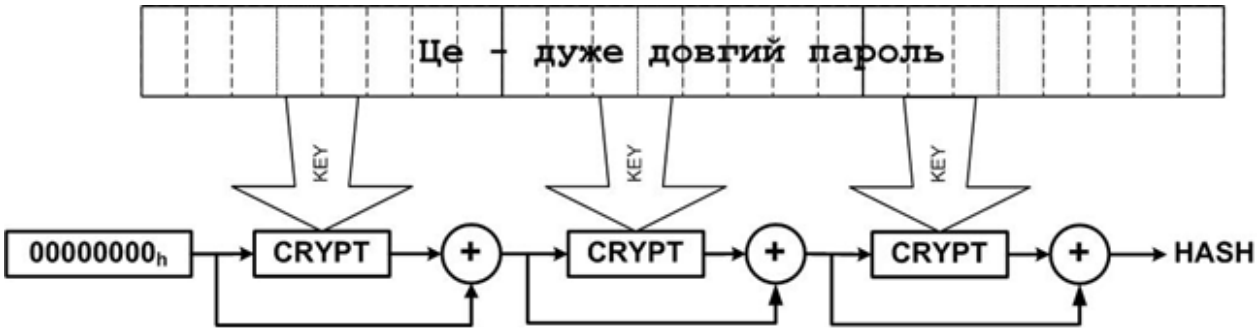


Рисунок 2.22 – Схема хешування на основі одноланцюгового блокового шифрування

У тому випадку, коли довжина ключа в два рази перевищує довжину блоку, а подібна залежність досить часто зустрічається в блокових шифрах, використовується схема, що нагадує мережу Фейштеля. Характерним недоліком і наведеної вище формули, і хеш-функції, заснованої на мережі Фейштеля, є велика ресурсомісткість відносно паролю. Для здійснення лише одного перетворення, наприклад, блоковим шифром з ключем завдовжки 128 біт використовується 16 байт рядка-пароля, а власне довжина пароля зрідка перевищує 32 символи. Отже, при обчисленні хеш-функції над паролем будуть проведено максимум 2 «повноцінних» криптоперетворення.

Рішенням цієї проблеми є :

1) використання заздалегідь помноженого рядку-паролю, наприклад, записавши його багато разів послідовно до досягнення довжини у 256 символів;

2) модифікувати схему застосовування криптоалгоритму так, щоб матеріал рядка-пароля "повільніше" витрачався при обчисленні ключа.

За другим методом Девіс і Майер запропонували на основі блокового шифру алгоритм, що використовує матеріал рядка-пароля багатократно і невеликими порціями. Йому притаманні елементи обох наведених вище схем, але крипостійкість цього алгоритму підтверджена численними реалізаціями в різних криптосистемах. Алгоритм одержав назву "Tandem DM" (рис. 2.23):

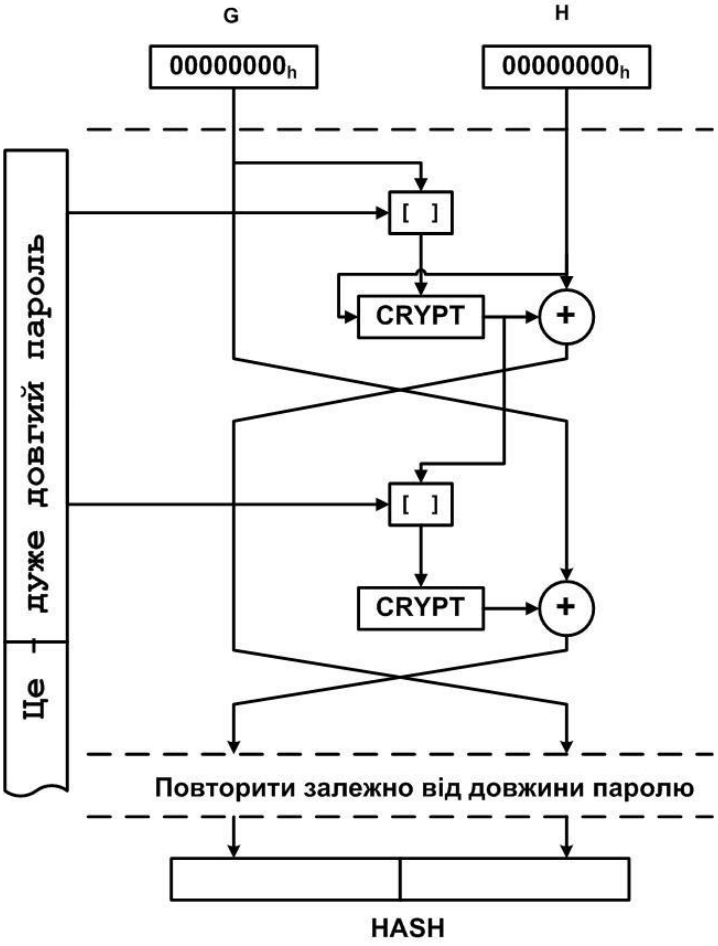


Рисунок 2.23 – Схема алгоритму хешування Tandem DM

G0=0; H0=0 ;
FOR J = 1 TO N DO
BEGIN
TMP=EnCrypt(H,[G,PSWj]); H'=H XOR TMP;
TMP=EnCrypt(G,[PSWj,TMP]); G'=G XOR TMP;
END;
Key=[Gk,Hk]

Квадратними дужками (X16=[A8,B8]) тут позначено просте поєднання (склеювання) двох блоків інформації рівної величини в один – подвоєної розрядності. А у якості процедури EnCrypt(X,Key) може бути обрано будь-який

стійкий блоковий шифр. Як слід з формул, такий алгоритм орієнтовано на те, що довжина ключа двократно перевищує розмір блоку криптоалгоритму. Характерною особливістю схеми є те, що рядок пароля зчитується блоками по половині довжини ключа, і кожний блок використовується в створенні хеш-результату двічі. Таким чином, при довжині пароля в 20 символів і необхідність створення 128 бітового ключа внутрішній цикл хеш-функції повториться 3 рази.

2.4.6 Транспортне кодування

Оскільки системи шифрування даних часто використовуються для кодування текстової інформації : листування, рахунків, платежів електронної комерції, і при цьому криптосистема повинна бути абсолютно прозорою для користувача, то над вихідним потоком криптосистеми часто здійснюється транспортне кодування, тобто додаткове кодування (а не шифрування) інформації винятково для забезпечення сумісності з протоколами передачі даних.

На виході криптосистеми байт може приймати всі 256 можливих значень, незалежно від того чи був вхідний потік текстовою інформацією чи ні. А при передачі поштових повідомлень багато систем орієнтовано на те, що припустимі значення байтів тексту змінюються у вужчому діапазоні: всі цифри, розділові знаки, алфавіт латиниці плюс, можливо, національної мови. Перші 32 символи набору ASCII призначаються для спеціальних цілей. Для того, щоб вони і деякі інші службові символи ніколи не з'явилися у вихідному потоці даних використовується транспортне кодування.

Найпростіший метод полягає в записі кожного байту двома шістнадцятковими цифрами-символами. Так байт 252 буде записаний двома символами «FC»; байт з кодом 26, якому відповідає спеціальний символ CTRL-Z, буде записаний двома припустимими символами «1A». Така схема є дуже надмірною, так як у одному байті передається тільки 4 біти інформації.

Практично в будь-якій системі комунікації без проблем можливо передавати приблизно 68 символів (латинський алфавіт рядковий і прописний, цифри і розділові знаки). З цього слід, що цілком реально створити систему з передачею 6 біт в одному байті (26<68), тобто кодувати 3 байти довільного змісту чотирма байтами з виключно дозволених (так званих друкарських) символів. Подібна система була розроблена і стандартизована на рівні протоколів мережі Інтернет – це система Base64 (стандарт RFC1251), яку наведено на рис. 2.24.

Процес кодування перетворює 4 вхідні символи у вигляді 24-бітової групи, обробляючи їх зліва направо. Ці групи потім розглядаються як 4 сполучені шостибітові групи, кожна з яких транслюється в одиночну цифру алфавіту Base64. При кодуванні Base64 вхідний потік байтів повинен бути впорядкований старшими бітами спочатку. Кожна шостибітова група використовується як індекс для масиву 64-х друкарських символів. Символ, на який вказує значення індексу, поміщається у вихідний рядок. Ці символи вибрані так, щоб бути універсально уявними і виключають символи, що мають спеціальне значення (".", CR, LF).

Вихідний потік (закодовані байти) повинен мати довжину рядків не більше 76 символів. Всі ознаки перекладу рядка і інші символи, відсутні на рис. 8, повинні бути проігноровані декодером base64. Серед даних в Base64 символи, не перераховані на рис. 2.24, закінчення рядку і т. і., повинні говорити про помилку

передачі даних, і, відповідно, програма-декодер повинна сповістити користувача про неї.

Код	Символ	Код	Символ	Код	Символ	Код	Символ
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Рисунок 2.24 – Алфавіт Base64

Якщо у хвості потоку кодованих даних залишилося менше, ніж 24 біти, справа додаються нульові біти до утворення цілої кількості шостибітових груп. У випадках коли до кінця 24-бітової групи може залишатися тільки від 0 до 3-х не дістаючих шостибітових груп, замість кожної з яких ставиться символ-заповнювач «=». Оскільки весь вхідний потік є цілим числом восьмибітових груп (тобто, просто байтних значень), то можливі лише наступні випадки:

1. вхідний потік закінчується рівно 24-бітовою групою (довжина файлу кратна 3); у такому разі вихідний потік закінчуватиметься чотирма символами Base64 без будь-яких додаткових символів;

2. «хвіст» вхідного потоку має довжину 8 біт; тоді в кінці вихідного коду містяться два символи Base64, з додаванням двох символів «=»;

3. «хвіст» вхідного потоку має довжину 16 біт; тоді в кінці вихідного коду містяться три символи Base64 і один символ «=».

Оскільки символ «=» є прикінцевим заповнювачем, його поява в змісті повідомлення може означати тільки те, що кінець даних досягнутий. Але спиратися на пошук символу «=» для виявлення кінця файлу невірно, оскільки,

якщо кількість переданих бітів кратне 24, то у вихідному файлі не з'явиться жодного символу «=».

2.4.7 Загальна схема симетричної криптосистеми

Узагальнена схема симетричної криптосистеми з урахуванням всіх розглянутих функцій зображена на рис. 2.25.

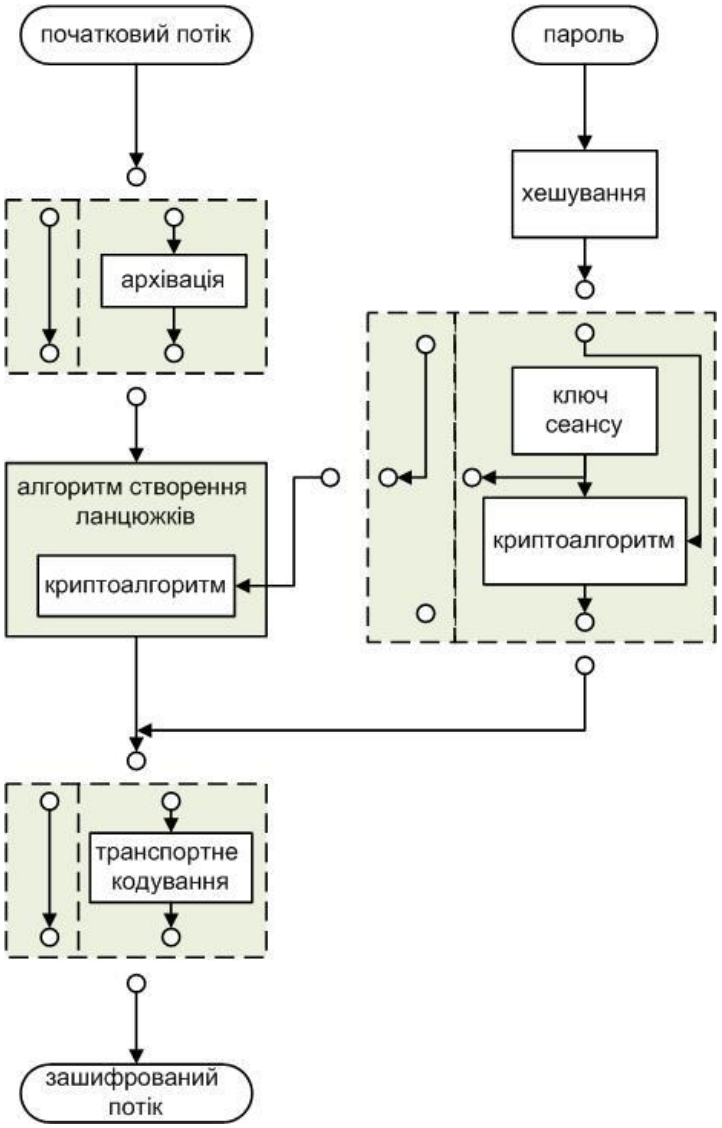


Рисунок 2.25 – Узагальнена схема симетричної криптосистеми

Найважливішими параметрами всіх систем симетричного шифрування є:

- стійкість;
- довжина ключа;
- кількість раундів;
- довжина блоку, якій оброблюється;
- складність апаратно/програмної реалізації;
- складність перетворень.

До переваг симетричних систем можливо віднести:

- порівняно високу швидкість (приблизно на 3 порядки вищу ніж у асиметричних систем);
- простота реалізації (за рахунок більш простих операцій);
- менша необхідна довжина ключа для відповідної стійкості.

Але є також суттєві недоліки, які практично призводять до того, що такі системи майже не використовується:

- складність керування ключами у великій мережі: це означає квадратичне збільшення кількості ключів, які необхідно генерувати, зберігати, передавати та знищувати у мережі (для мережі з 10 абонентів потрібно 45 ключів, для 100 – вже 4950, для 1000 – 499500);
- складність обміну ключами: для застосування симетричної системи необхідно вирішити проблему надійної передачі ключів до кожного абонента, тому що необхідний секретний канал для передачі кожного ключа обом сторонам.

2.5 Асиметричні криптоалгоритми

2.5.1 Загальні відомості про асиметричні криптоалгоритми

Симетричні криптосистеми, що мають множину переваг, володіють одним суттєвим недоліком, що виникає, коли спілкування між собою проводять не три-чотири людини, а сотні і тисячі користувачів. В цьому випадку для кожної пари, що переписується між собою, необхідно створювати свій секретний симетричний ключ. Це у результаті призводить до існування в системі з N користувачів $N^2/2$ ключів, що є дуже великою кількістю. Крім того, при порушенні конфіденційності будь-якої робочої станції зловмисник отримує доступ до всіх ключів цього користувача і може відправляти, нібито від його імені, повідомлення всім абонентам, з якими «жертва» вела листування.

Своєрідним рішенням цієї проблеми є використання асиметричної криптографії. Ця область криптографії дуже молода в порівнянні з іншими представниками. Перша така схема, що мала прикладну цінність, була запропонована всього близько 40 років тому. За цей час асиметрична криптографія перетворилася на один з основних напрямів криптології, і використовується в сучасному світі також часто, як і симетричні схеми.

Асиметрична криптографія спочатку розроблялась як засіб передачі повідомлень від одного об'єкту до іншого (а не для конфіденційного зберігання інформації, яке забезпечують тільки симетричні алгоритми). Тому наступний зміст використовує терміни "відправник" – особа, шифрує, а потім відправляє інформацію за незахищеним каналом і "одержувач" – особа, що приймає і відновлює інформацію в її початковому вигляді. Основна ідея асиметричних криптоалгоритмів полягає в тому, що для шифрування повідомлення використовується один ключ, а при дешифруванні – інший.

Крім того, процедура шифрування обрана так, що вона є незворотною навіть за відомим ключом шифрування – це друга необхідна умова асиметричної криптографії. Тобто, знаючи ключ шифрування і зашифрований текст, неможливо відновити початкове повідомлення – прочитати його можливо тільки за допомогою другого ключа – ключа дешифрування. Т. ч., ключ шифрування для

відправки листів будь-якій особі можливо взагалі не приховувати – знаючи його все одно неможливо прочитати зашифроване повідомлення. Тому, ключ шифрування називають в асиметричних системах «відкритим ключем», а ключ дешифрування одержувачу повідомлень необхідно тримати в секреті – він називається «закритим ключем». На питання: «Чому, знаючи відкритий ключ, не можливо обчислити закритий ключ?» висуває третю необхідну умову щодо асиметричної криптографії – алгоритми шифрування і дешифрування створюються так, щоб знаючи відкритий ключ, неможливо обчислити закритий ключ.

У цілому система листування при використанні асиметричного шифрування виглядає наступним чином. Для кожного з N абонентів, що ведуть листування, обрана своя пара ключів: «відкритий» E_j і «закритий» D_j, де j – номер абонента. Всі відкриті ключі відомі всім користувачам мережі, кожний закритий ключ, навпаки, зберігається тільки у того абонента, якому він належить. Якщо абонент, наприклад під номером 7, збирається передати інформацію абоненту під номером 9, він шифрує дані ключем шифрування E_9 і відправляє її абоненту 9. Не зважаючи на те, що всі користувачі мережі знають ключ E_9 і, можливо, мають доступ до каналу, за яким передається зашифроване повідомлення, вони не можуть прочитати початковий текст, оскільки процедура шифрування незворотна за відкритим ключем. Тільки абонент №9, одержавши повідомлення, проводить над ним перетворення за допомогою відомого тільки йому ключа D_9 і відновлює початковий текст. Якщо повідомлення потрібно відправити в протилежному напрямі (від абонента 9 до абонента 7), то потрібно буде використовувати вже іншу пару ключів (для шифрування ключ E_7, а для дешифрування – ключ D_7).

Т. ч., по-перше, в асиметричних системах кількість існуючих ключів пов'язана з кількістю абонентів лінійно (в системі з N користувачів використовуються 2N ключів), а не квадратична, як в симетричних системах. По-друге, при порушенні конфіденційності k-ої робочої станції зловмисник довідається тільки ключ D_k: це дозволяє йому читати всі повідомлення, що надходять до абоненту k, але не дозволяє видавати себе за нього при відправці листів. Окрім цього, асиметричні криптосистеми володіють ще і іншими корисними можливостями.

2.5.2 Алгоритм RSA

Алгоритм RSA було запропоновано трьома математиками; Рональдом Рівестом (R. Rivest), Аді Шаміром (A. Shamir) і Леонардом Адльманом (L. Adleman) у 1977-78 роках.

Першим етапом асиметричного алгоритму є створення пари ключів: відкритого і закритого і розповсюдження відкритого ключа всім бажаючим. Для алгоритму RSA етап створення ключів складається з наступних операцій:

1. обираються два прості (!) числа p і q;
2. обчислюється їх твір n = p q;
3. обирається довільне число e (e < n), таке, що НОД(e, (p - 1)(q - 1)) = 1, тобто e повинне бути взаємно простим з числом (p - 1)(q - 1);
4. методом Евкліду розв'язується в цілих числах (!) рівняння

$$e\,d + (p - 1)(q - 1)\,y = 1,$$

де невідомими є змінні d і y – метод Евкліда якраз і знаходить множину пар (d, y), кожна з яких є рішенням рівняння в цілих числах;

5. два числа (e, n) – публікуються як відкритий ключ;

6. число d зберігається в таємниці – це і є закритий ключ, який дозволить читати всі повідомлення, що зашифровані за допомогою пари чисел (e, n).

Для шифрування виконуються наступні дії:

1. відправник розбиває своє повідомлення на блоки, що дорівнюють

$$k = [\log_2(n)] \text{ біт},$$

де квадратні дужки позначають отримання цілої частини від дробового числа.

2. такий блок може бути інтерпретовано як число з діапазону $(0; 2^k - 1)$. Для кожного такого числа (m_i) обчислюється вираз $c_i = ((m_i)\ e\ ) \bmod n$. Блоки c_i і є зашифроване повідомлення, їх можливо передавати за відкритими каналами, т. я. операція зведення в ступінь за модулем простого числа, є незворотною математичною задачею. Зворотна їй задача носить назву "логарифмування в кінцевому полі" і є на декілька порядків складнішою задачею. Тобто навіть якщо зловмисник знає числа e і n, то за c_i прочитати початкові повідомлення mi він не може ніяк, окрім як повним перебором значень m_i.

На приймальному боці процес дешифрування виконується за допомогою числа d, що зберігається в таємниці. За доказом теореми Ейлера слід, що якщо число n надано у вигляді двох простих чисел p і q, то для будь-якого x має місце рівність:

$$(x^{(p - 1)(q - 1)}) \bmod n = 1.$$

Така формула і застосовується з метою дешифрування RSA - повідомлень. Для цього підносяться обидві її частини до ступеня (-y) :

$$(x^{(-y)(p - 1)(q - 1)}) \bmod n = 1^{(-y)} = 1.$$

Далі помножуються обидві її частини на x :

$$(x^{(-y)(p - 1)(q - 1) + 1}) \bmod n = 1x = x.$$

При створенні відкритого і закритого ключів, за допомогою алгоритму Евкліда, обирається таке d, що $ed + (p - 1)(q - 1)y = 1$, тобто $ed = (-y)(p - 1)(q - 1) + 1$. А отже в виразі $ci = ((mi)^{\ e}\ ) \bmod n$ можливо замінити показник ступеня на число (ed). Звідси одержується $(x^{ed}) \bmod n = x$. Тобто для того, щоб прочитати повідомлення $ci = ((mi)^{\ e}\ ) \bmod n$ достатньо піднести його за ступенем d по модулю m :

$$((ci)^{\ d}\ ) \bmod n = ((mi)^{\ ed}) \bmod n = mi.$$

Але операції зведення в ступінь великих чисел достатньо трудомісткі, навіть якщо вони проводяться по оптимізованих за часом алгоритмах. Тому весь текст повідомлення кодується блоковим шифром (що є набагато швидшим), але з використанням ключа сеансу, а ключ сеансу шифрується асиметричним алгоритмом за допомогою відкритого ключа одержувача і заноситься на початок файлу.

2.5.3 Технології цифрових підписів

Теорія асиметричного шифрування дозволяє вирішувати ще одну проблему інформаційної безпеки – перевірку достовірності автора повідомлення. Для вирішення цієї проблеми за допомогою симетричної криптографії була розроблена дуже трудомістка і складна схема. В той же час за допомогою, наприклад, того ж алгоритму RSA, створити алгоритм перевірки достовірності автора і незмінності повідомлення надзвичайно просто.

Припустимо, що потрібно передати будь-який текст, не обов'язково секретний, але важливо те, щоб в нього при передачі за незахищеним каналом не були внесені зміни. До таких текстів звичайно відносяться різні розпорядження, довідки, і тому подібна документація, що не представляє таємниці. Обчислимо від цього тексту будь-яку хеш - функцію – це буде число, яке більш менш унікально характеризує такий текст.

Хоча можливо знайти і інший текст, який дає те ж саме значення хеш - функції, але змінити в початковому тексті десять - двадцять байт так, щоб текст залишився повністю осмисленим, та ще і змінився у вигідний зловмиснику бік (наприклад, зменшив суму оплати у два рази) – надзвичайно складно. Саме для усунення цієї можливості хеш - функції створюють такими ж складними як і криптоалгоритми – якщо текст з таким же значенням хеш - функції можливо буде підібрати тільки методом повного перебору, а множина значень складатиме як і для блокових шифрів 2^{32} – 2^{128} можливих варіантів, то для пошуку подібного тексту зловмиснику «потрібно» також мільйони років.

Таким чином, якщо можливо передати одержувачу захищеним від зміни методом хеш - суму від тексту, що пересилається, то у нього завжди буде можливість самостійно обчислити хеш - функцію від тексту вже на приймальній стороні і звірити її з надісланим значенням. Якщо хоча б один біт в обчисленій їм самостійно контрольній сумі тексту не співпадає з відповідним бітом в одержаному хеш - значенні, то, текст при передаванні зазнав несанкціоновані зміни.

Наведемо призначену до передачі хеш - суму у вигляді декількох k - бітових блоків h_i, де k – це розмір повідомлень за алгоритмом RSA. Обчислимо над кожним блоком значення:

$$s_i = ((h_i)^{d}) \bmod n ,$$

де d – закритий ключ відправника. Тепер повідомлення, що складається з блоків si можливо передавати по мережі. Ніякій небезпеці за відомими h_i і s_i знайти секретний ключ немає – це настільки ж складна задача, як і задача

«логарифмування в кінцевому полі». А будь-який одержувач повідомлення може легко прочитати початкове значення h_i, виконавши операцію:

$$((s_i)^{e}) \bmod n = ((h_i)^{de}) \bmod n = h_i .$$

Відкритий ключ (e, n) є у всіх, а те, що зведення будь-якого числа в ступінь (ed) за модулем n дає початкове число, було доведено вище. При цьому ніхто інший, окрім відправника, не знаючи його закритого ключа d не може, змінивши текст, а отже, і хеш - суму, обчислити такі s'_i, щоб при їх зведенні в ступінь е вийшла хеш-сума h'_i, яка буде дорівнювати хеш - сумі фальсифікованого тексту.

Таким чином, маніпуляції з хеш - сумами тексту представляють з себе «асиметричне шифрування навпаки»: при відправці використовується закритий ключ відправника, а для перевірки повідомлення – відкритий ключ відправника. Подібна технологія одержала назву «електронний підпис». Інформацією, яка унікально ідентифікує відправника (його віртуальним підписом), є закритий ключ d. Жодна людина, що не володіє цією інформацією, не може створити таку пару (текст, s_i), щоб алгоритм перевірки дав би позитивний результат.

Подібний обмін місцями відкритого і закритого ключів для створення з процедури асиметричного шифрування алгоритму електронного підпису можливий тільки в тих системах, де виконується властивість комутативності ключів. Для інших асиметричних систем алгоритм електронного підпису або значно відрізняється від базового, або ж взагалі не може бути реалізовано.

2.5.4 Механізм розповсюдження відкритих ключів

Хоча асиметричні криптосистеми позбавлені одного з найголовніших недоліків симетричних алгоритмів – необхідності попереднього обміну сторонами секретним ключем за захищеною схемою (наприклад, особисто або за допомогою повіреного кур'єра), недостатнім вільне розповсюдження відкритих ключів. Але припустимо є відправник повідомлення і отримувач. Для того, щоб відправити зашифроване повідомлення, відправник повинен дізнатися відкритого ключа отримувача. Якщо отримувач не передав цей ключ відправнику особисто на змінному носії, то відправник отримав ключ з інформаційної мережі. При цьому не існує гарантії, що цей набір байт є саме відкритим ключем отримувача. Адже зловмисник може згенерувати довільну пару (закритий ключ, відкритий ключ), потім активно поширювати або пасивно підміняти при запиті відправника відкритий ключ отримувача на створеним їм. В цьому випадку при відправці повідомлення 1) відправник шифрує його тим ключем, який він вважає, що належить отримувачу; 2) зловмисник, перехопивши повідомлення дешифрує його парним закритим ключем, прочитає і більш того 3) може переслати далі, зашифрувавши дійсно вже відкритим ключем отримувача. Так само, але за інверсною схемою, він може підмінити і електронний підпис відправника під його листом.

Таким чином, якщо між відправником і одержувачем немає конфіденційної схеми передачі асиметричних ключів, то виникає серйозна небезпека появи зловмисника-посередника. Але асиметрична криптографія знайшла спосіб дуже значного зниження ризику подібної атаки. Теза, що між відправником і

одержувачем немає гарантованої лінії зв'язку, не є повністю вірною. Поза сумнівом у будь-кого знайдеться троє-четверо надійних знайомих в столиці або закордоном, у них в свою чергу також знайдеться безліч знайомих в багатьох точках країни і світу. Врешті-решт, кожна особа користуєтеся програмним забезпеченням фірм, представництва яких знаходяться в тій країні або в тому місті, куди необхідно відправити листа. Проблема тільки в тому, що починаючи, з другої від відправника ланки людина особисто не знаєте його і ймовірність того, що вона, або більш того, велика компанія, будуть якось мотивовані, дуже мала.

Але, якщо безліч однодумців об'єднаються з метою створити надійну мережу розповсюдження ключів, то це буде ймовірно. А сама асиметрична криптографія допоможе їм в цьому таким чином: в дійсності нікуди ходити зі змінним носієм не потрібно, для передання відкритого ключа відправника одержувачу. Адже особа спілкуєтеся з власним знайомим, тому у відправника є його відкритий ключ, одержаний будь-яким надійним способом. А отже, він може передати відправнику цей відкритий ключ одержувача, підписавши повідомлення власним електронним підписом. А від відправника вимагається лише надіслати цей ключ далі по ланцюжку у напрямі отримувача, підписавши вже власним електронним підписом. Таким чином, минувши дещо перепідписань, відкритий ключ дійде від місця відправлення до місця вимоги за надійним шляхом. У принципі від відправника навіть може не вимагатися ніяких дій – йому потрібно лише встановити на власній ЕОМ спеціальний сервер розповсюдження ключів, і він всі зазначені дії виконуватиме автоматично.

На сьогоднішній день не існує єдиної мережі розповсюдження відкритих ключів, що пов’язано з «війною» стандартів. Тепер розвиваються декілька незалежних систем, але жодна з них не одержала переваги над іншими.

Необхідно відзначити, що ланцюжок розповсюдження ключів в реальних випадках не дуже великий. Звичайно він складається з двох-чотирьох ланок. Із залученням до процесу розповсюдження ключів крупних фірм-виробників програмних продуктів він стає ще коротшим. Якщо на носії з придбаним програмним забезпеченням вже знаходиться відкритий ключ цієї фірми, а сама вона має крупний ринок збуту, то ланцюжок складатиметься або з однієї ланки (якщо ПЗ цієї ж фірми встановлено і у потенційних отримувачів), або з двох (другим стане будь-який інший концерн, ПЗ якого встановлено у отримувача, між собою всі великі компанії обмінялися ключами електронних підписів достатньо давно). Відкритий ключ, підписаний будь-якою третьою стороною, називається завіреним за допомогою сертифікату. Сертифікатом називається інформаційний пакет, що містить будь-який об'єкт (зазвичай ключ) і електронний підпис, підтверджуючий цей об'єкт від імені чиєї-небудь особи.

2.5.5 Обмін ключами за алгоритмом Діффі - Хеллмана

Алгоритм названо за прізвищами його розробників Діффі (Diffie) і Хеллмана (Hellman). Припустимо, що двом абонентам необхідно здійснити конфіденційне листування, а в їх розпорядженні немає спочатку обумовленого конфіденційного ключа. Проте, між ними існує канал, захищений від модифікації, тобто дані, що надходять за ним, можуть прослуховуватись, але не можуть бути

змінені. В цьому випадку дві сторони можуть створити однаковий таємний ключ, жодного разу не передавши його по мережі, за наступним алгоритмом.

Припустимо, що обом абонентам відомо деякі два числа v і n. Вони, втім, відомі і всій решті зацікавлених осіб. Наприклад, вони можуть бути просто фіксовано «зашиті» до програмного забезпечення. Для того, щоб створити секретний ключ, обидва абоненти генерують випадкові або псевдовипадкові прості числа: перший абонент – число x, другий абонент – число y. Потім перший абонент обчислює значення (v^x) mod n і надсилає його другому, а другий обчислює значення (v^y) mod n і надсилає його першому. Якщо зловмисник отримає обидва ці значення, то модифікувати їх (втрутитися в процес передачі) не він не зможе. Далі перший абонент на основі того, що у нього є x і одержане по мережі значення (v^y) mod n обчислює значення $(((v^y) \bmod n)^x) \bmod n$, а другий абонент на основі власного значення y і одержаного по мережі (v^x) mod n обчислює значення $(((v^x) \bmod n)^y) \bmod n$. Така операція зведення в ступінь може здійснюватися через операцію узяття модуля за простим числом (комутативна у кінцевому полі), тобто у обох абонентів розраховано однакове число: $((v^{xy}) \bmod n$. Його вони і можуть використовувати як секретний ключ, оскільки тут зловмисник знову зустрінеться з проблемою RSA при спробі з'ясувати за перехопленими (v^x) mod n і (v^y) mod n власно числа x і y – це дуже ресурсоємна операція, якщо числа v, n, x, y обрано достатньо великими.

Алгоритм Діффі - Хеллмана працює тільки за лініями зв'язку, що надійно захищені від модифікації. Якщо його застосовувати на будь-яких відкритих каналах, то це вирішило б проблему розповсюдження ключів і, можливо, замінило б собою всю асиметричну криптографію. Проте, в тих випадках, коли в каналі можлива модифікація даних, з'являється очевидна можливість втручання до процесу генерації ключів «зловмисника – посередника» за тією ж самою схемою, що і для асиметричної криптографії.

2.5.6 Загальна схема асиметричної криптосистеми

Загальна схема асиметричної криптосистеми зображена на рис. 2.26. За структурою вона практично ідентична симетричній криптосистемі (рис. 2.25) з ключем сеансу.

До переваг асиметричних криптосистем можливо віднести:

- не потрібно передавати закритий ключ будь-якими каналами зв’язку;
- на противагу симетричній криптосистемі, секретний ключ зберігається тільки у одній стороні;
- у симетричній криптосистемі варто змінювати ключ після кожного сеансу передачі даних, у асиметричній ключі можливо тримати незміненими достатньо довгий час;
- у більшості мереж кількість ключів при асиметричному шифруванні набагато менша, ніж при симетричному.

Але асиметричні криптосистеми мають наступні недоліки:

- хоча повідомлення шифруються надійно, але самі сторони «засвічуються» самим фактом передачі (на чому може бути побудована атака);

- асиметричні алгоритми використовують набагато довші ключі ніж симетричні;
- у чистому вигляді асиметричні системи вимагають значних обчислювальних ресурсів, тому на практиці вони частіше використовуються разом з іншими алгоритмами.

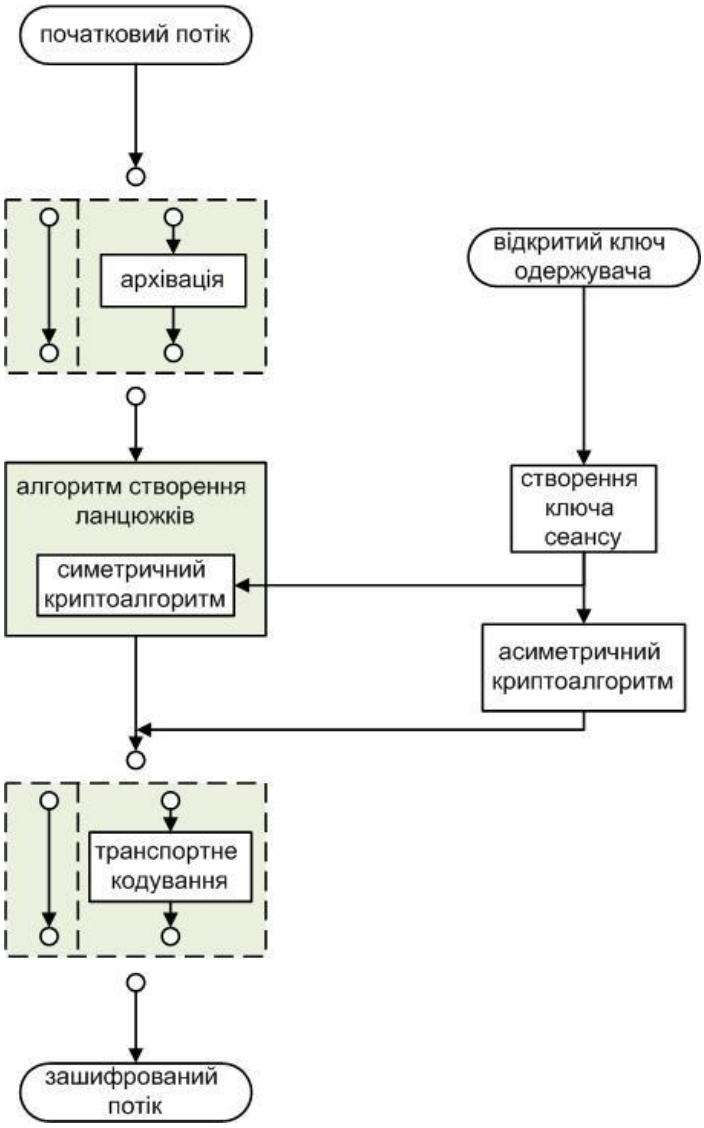


Рисунок 2.26 – Узагальнена схема асиметричної криптосистеми

Також асиметрична система з відкритим ключем використовується у системах цифрових підписів. Цей процес ґрунтується на властивості комутативності операцій шифрування та розшифрування.

3 МЕТОДИ І ПРИСТРОЇ ЗБЕРІГАННЯ ІНФОРМАЦІЇ

3.1 Методи резервного копіювання

Резервне копіювання - це процес виготовлення копій файлів, що знаходяться на жорсткому диску комп'ютера, з метою забезпечення можливості відновлення початкових даних у випадку, якщо оригінал буде стертий, перезаписаний, пошкоджений, або знищений. Архівація - це процес відносно більш довготривалого (або потрібного згідно із законом) зберігання важливої інформації, звичайно у вигляді додаткової копії, часто в стислому вигляді. Різниця між резервним копіюванням і архівацією даних досить принципова - резервне копіювання застосовують для зберігання інформації на зовнішніх носіях із тим, щоб її можливо було відновити (англ., еквівалент - restore або recovery) при аваріях або збоях, а також в деяких випадках для перенесення системних даних з машини на машину. При резервному копіюванні метою є збереження саме поточного стану системи. Попередні стани при цьому не завжди потрібно зберігати, хоча у ряді випадків це все-таки робиться. Архівація (archive) призначена для забезпечення довгострокового збереження напрацьованої інформації, при-чому так, щоб дані можливо було відновити, навіть якщо вони створені декілька місяців або років тому. Наприклад, архівації, як правило, піддаються дані бухгалтерських і складських програм, різні фінансові документи, вже закінчені проекти. Завдання резервного копіювання і архівації в цілому аналогічні і на практиці часто ці поняття вважають синонімами. У сучасній технології роботи з копіями даних з'явилася нова технологія - клонування - створення файлу-образу жорсткого диска (або його розділу). Клонування є окремим випадком резервного копіювання.

Резервне копіювання і архівація здійснюються відповідно до трьох основних програмних методів запису на зовнішні носії: повним, інкрементальним і диференціальним.

При повному (звичайному) методі кожного разу проводиться копіювання всього обсягу вказаної інформації, наприклад, переноситься цілком файлова система, файли бази даних або окремий каталог на диску. Кожний з файлів позначається як архівний. Даний метод займає багато часу при записі і веде до великої витрати інформаційних носіїв. Однак в цьому випадку відновлення інформації здійснюється швидше, ніж при будь-якому іншому методі (для цього буде потрібно лише один образ (один набір носіїв)). Повне копіювання є найбільш простим рішенням при резервному копіюванні системної інформації і служить відправною крапкою для інших методик.

Інкрементальний (додатковий) метод є поетапним способом запису інформації. При такому методі перший запис на зовнішній носій є повною копією. При кожній подальшому запису на нього поміщаються лише модифіковані в порівнянні з попереднім копіюванням файли (тобто ті, у яких змінилися властивості або права доступу). Після закінчення заданого адміністратором часу цикл повторюється, а значить, знову робиться повна копія, а потім додаються інкрементальні. При цьому файли спеціальним атрибутом позначаються як архівні. З погляду швидкості і економії носіїв, даний метод є найшвидшим і веде до мінімальної витрати його простору. Проте відновлення інформації займає

багато часу, оскільки спочатку інформацію потрібно відновити з повної копії, а потім по порядку зі всіх наступних.

При диференціальному (різницевому) методі перший запис на цифровий носій також є повною копією. На наступних етапах копіюються лише змінені з часу проведення першого (повного) копіювання файли. При цьому файли не позначаються як архівні, що вказує, що файли не заархівували. Після закінчення циклу вся процедура знову починається з повної копії. Диференціальний метод резервування потребує більше часу, ніж інкрементальний, та зате при відновленні інформації вимагає тільки дві копії: повну і останню диференціальну.

Програма архівації даних операційної системи Windows окрім цих трьох методів пропонує також:

- метод копій - використовується, коли потрібно виконати архівацію окремих файлів в проміжку між сеансами повної (звичайної) і інкрементальної (додаткової) архівації, оскільки при цьому не скасовуються результати попередніх завдань архівації; при цьому файли не позначаються як архівні.
- щоденний метод - використовується для архівації файлів, які були змінені в день виконання архівації по інших методиках.

Головною проблемою інкрементального і диференціального копіювання є проблема вибору надійного критерію для встановлення факту модифікації файлу. Звичайно у якості цього застосовується атрибут Archive (для систем DOS/Windows), час створення/зміни файлів, його розмір або контрольна сума вмісту файлу. Але всі вони мають ті або інші недоліки, пов'язані з особливостями обробки атрибутів і прав доступу окремими прикладними про-грамами. Тому найнадійнішим (і при цьому найекономічнішим по витраті носі-їв) є метод повного резервного копіювання.

Рекомендовані схеми ротації носіїв надають можливість використання де-кількох комплектів носіїв і забезпечують «глибину» версій файлів, дозволяючи відновлювати файл виходячи з його стану в певний момент часу. Однією з таких популярних схем є Grandfather-Father-Son (Дід – батько - син) - GFS. «Син» - це інкрементальне або диференціальне щоденне резервування, «Батько» - повне щотижневе резервування, а «Дід» - повне щомісячне резервування. Для цієї базової схеми ротації необхідно 12 комплектів носіїв (за умови 5 робочих днів в тиждень): чотири щоденних (понеділок-четвер); п'ять щотижневих, що виконуються в кожну п'ятницю 1-5; і три щомісячних, місяці 1-3). Носії використовуються в день, тиждень або місяць, відповідні їх етикетці. Така схема забезпечує глибину історії до 4-х місяців.

3.2 Пристрої резервного копіювання

Залежно від змістовного наповнення, частоти звернення та строку зберігання, виділяються три ключові задачі збереження електронних даних: оперативний доступ до інформації, резервне копіювання та архівне (довготривале) зберігання. Вирішення кожної з завдань потребує вибору та застосування різноманітного обладнання (носіїв) та програмного забезпечення відповідно до специфічних вимог щодо збереження та доступу.

Основні вимоги до оперативного збереження - безперервність доступу та висока швидкість роботи. Типовим прикладом може бути файловий сервер, головне завдання якого - негайне надання необхідних даних великому кількості користувачів корпоративної мережі. Варіантом вирішення завдання оперативного збереження є магнітні пристрої - HD-диски, RAID-масиви, останнім часом - флеш-пам'ять. На рівні оперативного зберігання інформації SSD-пристрої мають певні переваги перед HDD-накопичувачами: продуктивність (швидкість читання та запису до 270 Мбіт/с); низька латентність у режимі читання 65 мкс, у режимі запису 85 мкс (34-нм технологічний процес); низьке споживання енергії; відсутність рухомих частин; висока механічна стійкість; широкий діапазон робочих температур. При організації RAID 5 (7+1) на основі 2 Гбайт FC HDD та SSD (ZeusIOPS, 3,5 дюйми,

Архівне збереження передбачає збереження важливої інформації протягом тривалої години при забезпеченні швидкого доступу до неї, що диктує цілком визначені вимоги до технологій зберігання та обладнання, зокрема, тривале зберігання великих обсягів інформації в незмінному вигляді. Таким умовам відповідають роботизовані бібліотеки оптичних дисків (можуть використовуватись носії DVD/BD/UDO та магнітооптика). На рівні довгострокового зберігання інформації, MO-накопичувачі мають значно меншу годину пошуку інформації, ніж пристрої на магнітній стрічці. Крім того, MO диски більш довговічні, ніж стрімерні касети. Відповідно до даних кампанії Plasmon, загальна вартість володіння для систем UDO значно менша порівняно з магнітооптикою.

Резервне копіювання передбачає високу потокову швидкість запису та читання та велику ємність носія. Оптимальним вибором будуть системи на основі стрічкових накопичувачів. Довговічність зберігання не критична, оскільки резервне копіювання проводиться регулярно.

3.2.1 Стрімери

Стрімер - пристрій, що підключається до комп'ютера, для запису та відтворення цифрової інформації на касету з магнітною плівкою. На сьогоднішній день у деяких областях досі стрічкові накопичувачі находять широке застосування. Основне призначення – це запис та читання даних, а також їх копіювання та архівування. Запис відбувається за допомогою запам'ятовування інформації, фіксуючи її на накопичувачі паралельно за всіма доріжками. При роботі магнітно-стрічкового накопичувача сама стрічка може здійснювати рухи в обох напрямках. Робочий процес при цьому нагадує використання магнітофона, оскільки в кінці запису, коли закінчується стрічка, апарат рухає її в зворотному напрямку. Переваги - великий обсяг даних (до 80Гбайт на касету), недолік - послідовний доступом до даних і, отже, низька швидкість доступу.

Нинішні стрімери мають можливість підключатися за допомогою високопродуктивного інтерфейсу SAS, який здатний забезпечити швидкість передачі даних від трьох до шести гігабіт на секунду. Більш старі представники моделей IBM підключалися через роз'єм FICON.

Сучасні представники магнітних накопичувачів дотримуються стандартів, іменованих LTO. Компанія IBM представила стрічковий накопичувач (стрімер)

LTO-5 TS2350, який має два роз'єми підключення SAS і один порт для виходу в інтернет (рис. 3.1).



Рисунок 3.1 – Стрімер LTO-5 TS2350

3.2.2 Магнітооптичні накопичувачі

Магнітооптичні накопичувачі - запис здійснюється зміною намагніченості підкладки з феромагнітного матеріалу шляхом нагрівання за допомогою лазерного променю її невеликої ділянки у зовнішньому магнітному полі. Для зчитування даних використовується ефект Керра, який полягає у зміні напряму поляризації променю, відбитого від намагніченої поверхні. При нагріванні вище за температуру Кюрі (приблизно 300°C) відповідну ділянку активного шару можливо перемагнітити, тобто. стерти інформацію. Сьогодні випускаються магнітооптичні накопичувачі, призначені для роботи з носіями діаметром 3.5 та 5.25 дюйми. Диски розміщені в нерозбірні картриджі, що нагадують за конструкцією 3.5-дюймові дискети; таким чином вони надійно захищені від випадкового пошкодження (рис. 3.2).



Рисунок 3.2 – Зовнішній магнітооптичний накопичувач і магнітооптичний диск

Використовуючи магнітооптичні диски, можливо досягти надзвичайно надійного зберігання інформації, оскільки час збереження даних визначається практично стійкістю використаної підкладки (скло чи полікарбонат). Що стосується циклів запису, то у випробуванні на 100 мільйонів циклів не було

помічено жодних незворотних змін властивостей магнітного шару та підкладки. Завдяки тому, що головки читання/запису в них ніколи не торкаються диска, забезпечується висока стійкість до вібрацій та ударних навантажень. У магнітооптичних дисках, на відміну магнітних, немає мимовільного спотворення інформації, що робить ці пристрої придатними для довгострокового архівування даних. Вони не зазнають впливу підвищених та знижених температур, електромагнітних випромінювань та забруднень.

3.2.3 Накопичувачі типу Bernoulli

Головка читання/запису у таких накопичувачах, спроектована з урахуванням вимог аеродинаміки, плаває над поверхнею гнучкого диска Бернуллі. Повітряні потоки, що виникають внаслідок обертання диска з високою швидкістю, викликає вигін частини поверхні диска, що знаходиться під головкою читання/запису, у напрямку до останньої. Однак диск не стикається з головкою, між ними залишається невеликий досить стабільний проміжок, що забезпечується потоками повітря. Будь-яка зміна нормальних умов роботи накопичувача Бернуллі (наприклад, через удар або появу плями забруднення на поверхні диска) викликається порушення ефекту Бернуллі і призводить до того, що диск відходить від головки, замість того, щоб зіткнутися з нею (як це б сталося на звичайному вінчестері). Завдяки цьому виключається можливість відмов накопичувача, оскільки диск, що обертається, практично не може зіткнутися з головкою. Тому диски Бернуллі найстійкіші. Сам накопичувач Бернуллі, хоча він є гнучким і на вигляд схожий на звичайну дискету, насправді може експлуатуватися до п'яти років у режимі зчитування/запису - тобто. характеризується у 20 разів більшою довговічністю, ніж дискета, - згідно з даними постачальника. Носій з барієво-феритовим покриттям не тільки дозволяє записувати дані з втричі вищою щільністю ніж носій із звичайних HDD накопичувачів або НГМД, але й відрізняється значно більшою стійкістю до зношування, ніж у звичайних дискет. Накопичувачі Бернуллі за швидкістю доступу не поступаються ряду накопичувачів, що широко використовуються, на жорстких дисках із середньою швидкодією. Так наприклад, накопичувач , Bernoulli230 має ємність однієї касети 230 Мб, вбудований кеш 256 Кб, інтерфейс SCSI-2 або IDE и час доступу 12 мс.

3.2.4 Накопичувачі Zip і Jaz

Накопичувачі Zip є недорогими засібами для переміщення даних і збільшення обсягу інформації, що зберігається (рис. 3.3). На змінних дисках міститься до 750 Мб інформації.

Накопичувачі типу Jaz дозволяють зберігати на одному диску до 1 Гб даних (рис. 3.4).

3.2.5 Портативні накопичувачі на жорстких дисках

Поява на ринку портативних накопичувачів на жорстких дисках (Portable HDD), що працюють за швидкісним інтерфейсом USB (рис. 3.5), дозволило

використовувати їх не тільки для перенесення даних, але і як зручний пристрій резервного копіювання. Простота підключення до комп'ютера та прийнятна швидкість роблять ці пристрої незамінними для резервного копіювання домашніх комп'ютерів та офісних робочих станцій.



Рисунок 3.3 – Накопичувач Iomega ZIP



Рисунок 3.4 – Накопичувач Iomega Jaz



Рисунок 3.5 – Портативні накопичувачі на жорстких дисках

3.2.6 CD та DVD приводи

CD та DVD приводи з можливістю запису. Ємність CD-дисків до 800МБайт, DVD-дисків до 8,6 ГБайт (двошаровий двосторонній). Максимальна швидкість запису 56х (х = 150кБайт / с ≈ 8МБайт / с) для CD і 4х (х = 1,32МБайт / с ≈ 5МБайт / с) для DVD. Першим серед форматів із одноразовим записом на світ з'явився DVD-R. Такі диски здатні читати практично всі існуючі автономні DVD-програвачі і DVD-ROM-дисководи, і, звичайно ж, всі приводи, що пишуть. На сьогоднішній день максимальна швидкість запису на DVD-R складає 4-х, а ціна носіїв найнижча серед усіх доступних модифікацій. DVD-RW — це формат багаторазового запису, що базується на технології DVD-R. Такі диски сумісні з більшістю доступних пристроїв для читання, хоча в деяких випадках для забезпечення сумісності може знадобитися оновлення мікропрограми.

У 2001 р. було розроблено конкуруючий стандарт DVD+RW, заснований на технології CD-RW. За сумісністю з існуючими плеєрами та дисководами він схожий з DVD-RW, проте через менший вік проблеми вірогідніші. У той же час, на відміну від DVD-RW, DVD+RW дозволяють виконувати форматування носія на ходу, заощаджуючи користувачеві до декількох хвилин, залежно від приводу.

Дисководи цього формату, що пишуть, на сьогоднішній день представлені двома поколіннями - перше здатне досягати швидкості запису 2,4х, а друге - 4х. "Плюсова" версія стандарту з одноразовим записом з'явилася зовсім недавно, але за сумісністю близька до DVD-R. Доступні DVD+R пристрої з двократною та чотирикратною швидкістю запису, однак для використання 4-швидкісних носіїв у 2-швидкісних приводах потрібне оновлення мікропрограми останніх. Дещо окремо від DVD±RW коштує формат DVD-RAM. Він мало поширений у Європі та не сумісний з більшістю пристроїв читання та запису, проте найбільш ефективний як DVD-формат із довільним доступом та активною модифікацією. DVD-RAM Type 2 у режимі запису можуть використовуватися лише усередині спеціальних картриджів. Для зручності ми звели дані про сумісність приводів та носіїв у табл. 3.1

3.2.7 Твердотільні накопичувачі SSD

SSD-диск є немеханічним запам’ятовуючим пристроєм, здатним зберігати інформацію за умови відсутності живлення. За допомогою флеш-накопичувача можливо суттєво збільшити продуктивність ПК.

Такі накопичувачі є альтернативою HDD і використовуються як у мобільних, так і у стаціонарних комп'ютерах. Відмінною особливістю є наявність мікросхеми пам'яті та контролера, що керує, замість магнітної стрічки та металевих обертових дисків.

У порівнянні з жорсткими дисками вони мають такі переваги:

- менша вага та габарити;
- беззвучність роботи;
- стійкість до механічних впливів;
- висока швидкість довільних операцій.

З характеристик SSD-накопичувачів варто виділити:

- обсяг пам'яті – 60 ГБ – 4 та більше ТБ;

- швидкість запису даних - 450-530 Мб/сек.;
- швидкість читання – 520-560 Мб/с.

Таблиця 3.1 - Сумісність дисководів та дисків різних типів

Тип диска	Тип дисководу					
	DVD-ROM	DVD-R(G)	DVD-R(A)	DVD-RW	DVD-RAM	DVD+RW
DVD-ROM	Читає	Читає	Читає	Читає	Читає	Читає
DVD-R(G)	Зазвичай зчитує	Читає, пише	Читає	Читає, пише	Читає	Читає
DVD-R(A)	Зазвичай зчитує	Читає	Читає, пише	Читає	Читає	Читає
DVD-RW	Зазвичай зчитує	Читає	Читає	Читає, пише	Зазвичай читає	Зазвичай читає
DVD-RAM	Зрідка зчитує	Не зчитує	Не зчитує	Не зчитує	Читає, пише	Не зчитує
DVD+RW	Зазвичай зчитує	Зазвичай читає	Зазвичай читає	Зазвичай читає	Зазвичай читає	Читає, пише
DVD+R	Часто зчитує	Зазвичай читає	Зазвичай читає	Зазвичай читає	Читає	Читає, пише
CD-ROM	Читає	Читає	Читає	Читає	Читає	Читає
CD-R	Читає	Читає, пише	Читає	Читає, пише	Читає	Читає, пише
CD-RW	Читає	Читає, пише	Читає	Читає, пише	Читає	Читає, пише

Особливість SSD – обмежене допустиме кількість циклів запису. Як і HDD, флеш-пам'ять буває зовнішньою та внутрішньою. Зовнішня використовується як гігантська «флешка».

3.2.8 Карти пам'яті

Flash-карта є енергонезалежною карткою пам'яті з можливістю перезаписування інформації та збереження її навіть при вимкненому живленні. Маючи вигляд чіпів, на яких біти зберігаються в осередках пам'яті, вона працює за принципом оперативної пам'яті. Однак назвати її повноцінною заміною не можливо через обмежену тривалість експлуатації.

Читання з карти не призводить до зношування мікросхеми, на відміну від перезаписування, коли йде стирання попереднього запису. У сучасних флеш-картах заміна вмісту осередків може здійснюватися від 10 000 до 1 млн. разів.

Карти пам'яті також не можливо порівнювати з жорсткими дисками через обмеження місткості інформації (максимально 1 ТБ, але доступними залишаються флешки до 256 ГБ).

3.3 Дискові масиви з надмірністю даних

Дискові масиви з надмірністю даних, які прийнято називати RAID (Redundant Arrays of Inexpensive Disks – надлишковий масив дешевих дисків) були представлені дослідниками (Петтерсон, Гібсон та Катц) з Каліфорнійського університету в Берклі у 1987 р. З абревіатурою RAID свого часу трапився казус. Справа в тому, що дешевими дисками під час винаходу RAID називалися диски, які використовувалися у ПК, на противагу недешевим дискам для мейнфреймів. Однак для використання в масивах RAID довелося використовувати досить недешеву апаратуру в порівнянні з іншими комплектуючими ПК, тому RAID почали розшифровувати як Redundant Array of Independent Disks - надмірний масив незалежних дисків.

3.3.1 RAID 0.

RAID 0 - дисковий масив без стійкості до відмов (Striped Disk Array without Fault Tolerance) - є дисковим масивом, в якому дані розбиваються на блоки, і кожен блок записуються (або ж зчитується) на окремий диск (рис. 3.6). Таким чином, можливо здійснювати кілька операцій введення-виведення одночасно.

Переваги:

- найвища продуктивність для додатків, які потребують інтенсивної обробки запитів вводу/виводу та даних великого обсягу;
- простота реалізації;
- низька вартість на одиницю обсягу.

Недоліки:

- невідмовостійке рішення;
- відмова одного диска спричиняє втрату всіх даних масиву.

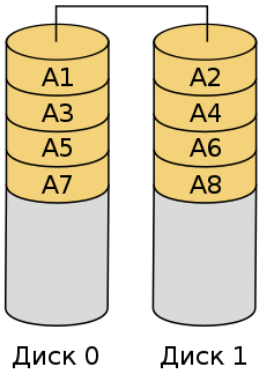


Рисунок 3.6 – RAID 0 - дисковий масив без відмовостійкості

3.3.2 RAID 1

RAID 1 - це дисковий масив із дублюванням або дзеркалюванням (mirroring). Дзеркалювання - традиційний спосіб підвищення надійності дискового масиву невеликого об'єму. У найпростішому варіанті використовується два диски, на які записується однакова інформація, і у разі відмови одного з них залишається дубль, який продовжує працювати в минулому режимі (рис. 3.7).

Переваги RAID 1:

- простота реалізації;
- простота відновлення масиву у разі відмови (копіювання);

- досить висока швидкодія для додатків з великою інтенсивністю запитів.

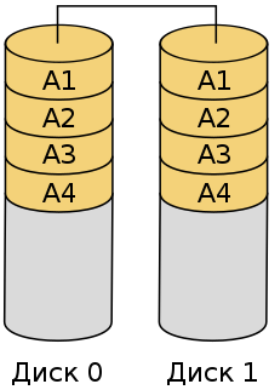


Рисунок 3.7 – RAID 1 - дисковий масив із дублюванням або дзеркалюванням

Недоліки RAID 1:
- висока вартість на одиницю обсягу - 100% надмірність;
- невисока швидкість передачі.

3.3.3 RAID 2

RAID 2 - це відмовостійкий дисковий масив із використанням коду Хеммінга (Hamming Code ECC). Код Хеммінга дозволяє виправляти поодинокі та виявляти подвійні несправності. Сьогодні активно використовується в технології кодування даних в оперативній пам'яті типу ECC та кодуванні даних на магнітних дисках. На рис. 3.8 показано приклад з фіксованою кількістю дисків у зв'язку з громіздкістю опису (слово даних складається з 4-біт, відповідно ECC код з 3-х).

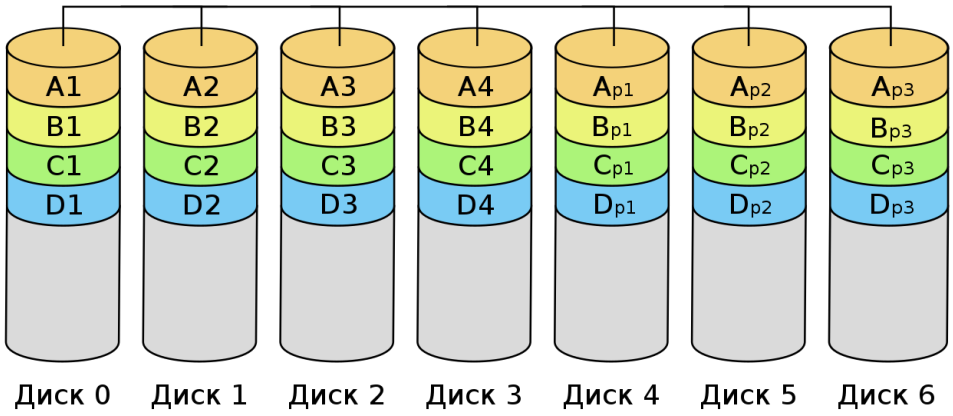


Рисунок 3.8 – RAID 2 - відмовостійкий дисковий масив із використанням коду Хеммінга

Переваги RAID 2:
- швидка корекція помилок ("на льоту");
- дуже висока швидкість передачі великих обсягів;
- зі збільшенням кількості дисків, накладні витрати зменшуються;
- досить проста реалізація.

Недоліки RAID 2:

- висока вартість при малій кількості дисків;
- низька швидкість обробки запитів (не підходить для систем орієнтованих на обробку транзакцій).

3.3.4 RAID 3

RAID 3 - це відмовостійкий масив з паралельною передачею даних та парністю (Parallel Transfer Disks with Parity). Дані у такому масиві розбиваються на підблоки на рівні байт і записуються одночасно на всі диски масиву, крім одного, який використовується для парності (рис. 3.9). Використання RAID 3 вирішує проблему великої надмірності в RAID 2. Більшість контрольних дисків, які у RAID рівня 2, потрібні визначення положення несправного розряду. Але цього немає потреби, оскільки більшість контролерів може визначити, коли диск відмовив за допомогою спеціальних сигналів, або додаткового кодування інформації, записаної на диск і використовуваної для виправлення випадкових збоїв.

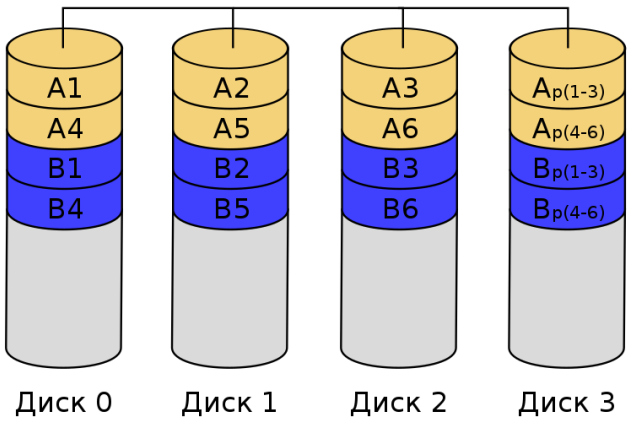


Рисунок 3.9 – RAID 3 - відмовостійкий масив з паралельною передачею даних та парністю

Переваги RAID 3:
- дуже висока швидкість передачі;
- відмова диска мало впливає швидкість роботи масиву;
- малі накладні витрати для реалізації надмірності.

Недоліки RAID 3:
- складна реалізація;
- низька продуктивність при великій інтенсивності запитів даних невеликого обсягу.

3.3.5 RAID 4

RAID 4 - це відмовостійкий масив незалежних дисків з диском парності (Independent Data disks with shared Parity disk). Дані у ньому розбиваються на блочному рівні. Кожен блок даних записується окремий диск і може бути прочитаний окремо (рис. 3.10). Парність для групи блоків генерується під час запису і перевіряється під час читання. RAID рівня 4 підвищує продуктивність

передачі невеликих обсягів даних за рахунок паралелізму, даючи можливість виконувати більше одного звернення щодо введення/виводу одночасно. Головна відмінність між RAID 3 і 4 полягає в тому, що в останньому розшарування даних виконується на рівні секторів, а не на рівні бітів або байтів.

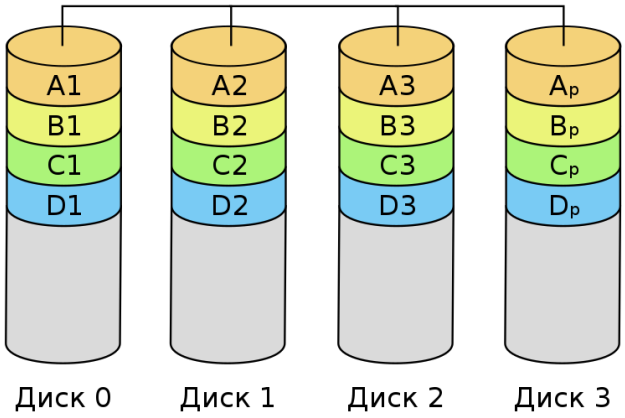


Рисунок 3.10 – RAID 4 - Відмовостійкий масив незалежних дисків з диском парності

Переваги RAID 4:

- висока швидкість читання даних великих обсягів;
- висока продуктивність при великій інтенсивності запитів читання даних;
- малі накладні витрати для реалізації надмірності.

Недоліки RAID 4:

- досить складна реалізація;
- дуже низька продуктивність під час запису даних;
- складне відновлення даних;
- низька швидкість читання даних малого обсягу при поодиноких запитах;
- асиметричність швидкодії щодо читання та запису.

3.3.6 RAID 5

RAID 5 - це відмовостійкий масив незалежних дисків з розподіленою парністю (Independent Data disks with distributed parity blocks). Цей рівень схожий на RAID 4, але, на відміну від попереднього, парність розподіляється циклічно по всіх дисках масиву (рис. 3.11). Ця зміна дозволяє збільшити продуктивність запису невеликих обсягів даних багатозадачних системах. Якщо операції запису спланувати належним чином, то можливо паралельно обробляти до N/2 блоків, де N - число дисків у групі.

Переваги RAID 5:

- висока швидкість запису даних;
- досить висока швидкість читання даних;
- висока продуктивність при великій інтенсивності запитів читання/запису даних;
- малі накладні витрати для реалізації надмірності.

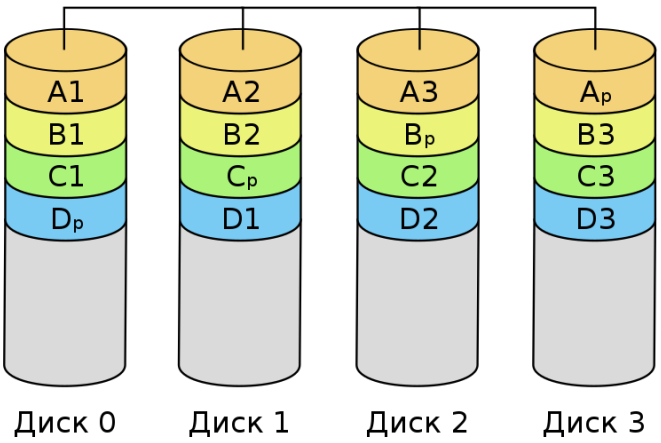


Рисунок 3.11 – RAID 5 - відмовостійкий масив незалежних дисків з розподіленою парністю

Недоліки RAID 5:

- швидкість читання даних нижче, ніж у RAID 4;
- низька швидкість читання/запису даних малого обсягу при поодиноких запитах;
- досить складна реалізація;
- складне відновлення даних.

3.3.7 RAID 6

RAID 6 - це відмовостійкий масив незалежних дисків із двома незалежними розподіленими схемами парності (Independent Data disks with two independent distributed parity schemes). Дані розбиваються на блочному рівні, аналогічно RAID 5, але на додаток до попередньої архітектури використовується друга схема для підвищення стійкості до відмов (рис. 3.12). Ця архітектура є стійкою до подвійних відмов. Однак при виконанні логічного запису реально відбувається шість звернень до диска, що збільшує час обробки одного запиту.

Переваги RAID 6:

- висока відмовостійкість;
- досить висока швидкість обробки запитів;
- досить малі накладні витрати для реалізації надмірності.

Недоліки RAID 6:

- дуже складна реалізація;
- складне відновлення даних;
- дуже низька швидкість запису даних.

3.3.8 RAID 10

Сучасні RAID контролери дозволяють комбінувати різні рівні RAID. Таким чином, можливо реалізувати системи, які поєднують у собі переваги різних рівнів, а також системи з великою кількістю дисків. Зазвичай це комбінація нульового рівня (stripping) та будь-якого відмовостійкого рівня.

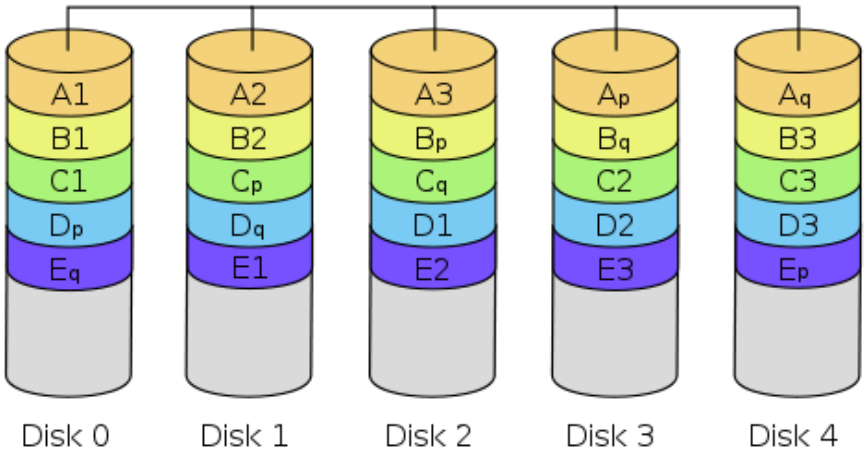


Рисунок 3.12 – RAID 6 - відмовостійкий масив незалежних дисків з двома незалежними розподіленими схемами парності

RAID 10 - це відмовостійкий масив з дублюванням та паралельною обробкою. Така архітектура є масивом типу RAID 0, сегментами якого є масиви RAID 1 (рис. 3.13). Він поєднує в собі дуже високу відмовостійкість і продуктивність.

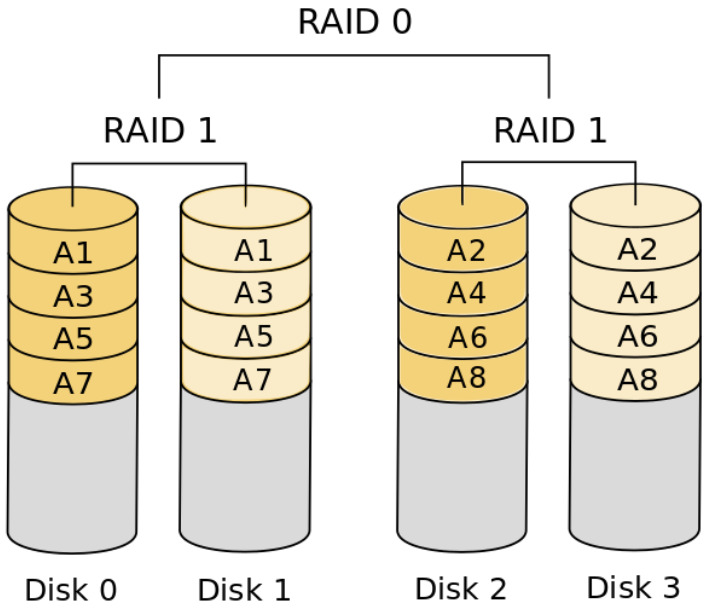


Рисунок 3.13 – RAID 10 - відмовостійкий масив з дублюванням та паралельною обробкою

Переваги RAID 10:
- висока відмовостійкість;
- висока продуктивність.

Недоліки RAID 10:
- дуже висока вартість;
- обмежене масштабування.

3.3.9 RAID 30.

RAID 30 - це відмовостійкий масив з паралельною передачею даних та підвищеною продуктивністю. Він є масивом типу RAID 0, сегментами якого є масиви RAID 3 (рис. 3.14) та поєднує в собі відмовостійкість і високу продуктивність. Зазвичай використовується додатків потребують послідовної передачі великих обсягів.

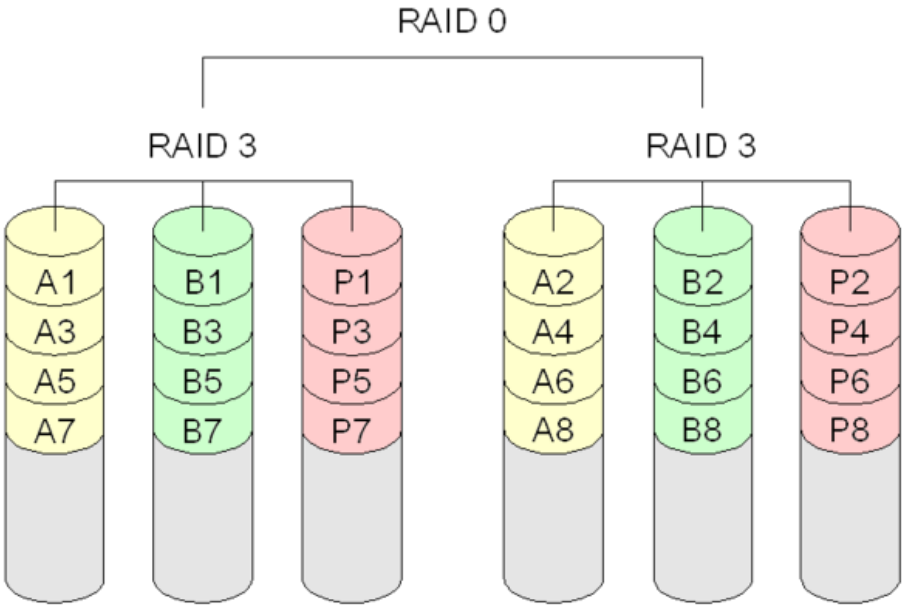


Рисунок 3.14 – RAID 30 - відмовостійкий масив з паралельною передачею даних та підвищеною продуктивністю

Переваги RAID 30:
- висока відмовостійкість;
- висока продуктивність.

Недоліки RAID 30:
- висока вартість;
- обмежене масштабування.

3.3.10 RAID 50

RAID 50 - це відмовостійкий масив з розподіленою парністю та підвищеною продуктивністю і є масивом типу RAID 0, сегментами якого є масиви RAID 5 (рис. 3.15). Він поєднує в собі відмовостійкість і високу продуктивність для додатків з великою інтенсивністю запитів і високу швидкість передачі даних.

Переваги RAID 50:
- висока відмовостійкість;
- висока швидкість передачі даних;
- висока швидкість обробки запитів.

Недоліки RAID 50:
- висока вартість;
- обмежене масштабування.

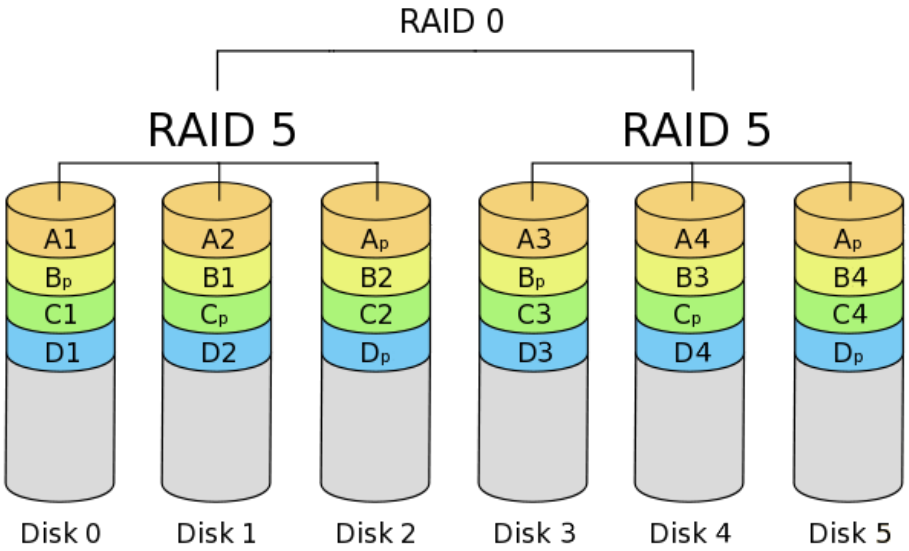


Рисунок 3.15 – Відмовостійкий масив з розподіленою парністю та підвищеною продуктивністю

3.3.11 RAID 7

RAID 7 - це відмовостійкий масив, оптимізований для підвищення продуктивності (рис. 3.16). (Optimized Asynchrony для High I/O Rates як добре як High Data Transfer Rates).

RAID 7® є зареєстрованою торговою маркою Storage Computer Corporation (SCC). Для розуміння архітектури RAID 7 розглянемо її особливості:

- всі запити на передачу даних обробляються асинхронно та незалежно;
- всі операції читання/запису кешуються через високошвидкісну шину x-bus;
- диск парності може бути розміщений будь-якому каналі;
- у мікропроцесорі контролера масиву використовується операційна система реального часу, орієнтована на обробку процесів;
- система має хорошу масштабованість: до 12-ти host-інтерфейсів, і до 48 дисків.
- операційна система контролює комунікаційні канали;
- використовуються стандартні SCSI диски, шини, материнські плати та модулі пам'яті;
- використовується високошвидкісна шина X-bus для роботи із внутрішньою кеш пам'яттю;
- процедура створення парності інтегрована в кеш;
- диски, приєднані до системи, можуть бути задекларовані як окремі;
- для керування та моніторингу системи можливо використовувати SNMP агент.

Переваги RAID 7:

- висока швидкість передачі даних і висока швидкість обробки запитів (1.5 - 6 разів вище за інші стандартні рівні RAID);
- висока масштабованість хост інтерфейсів;
- швидкість запису даних збільшується зі збільшенням кількості дисків у масиві;
- для обчислення парності немає необхідності додаткової передачі даних.

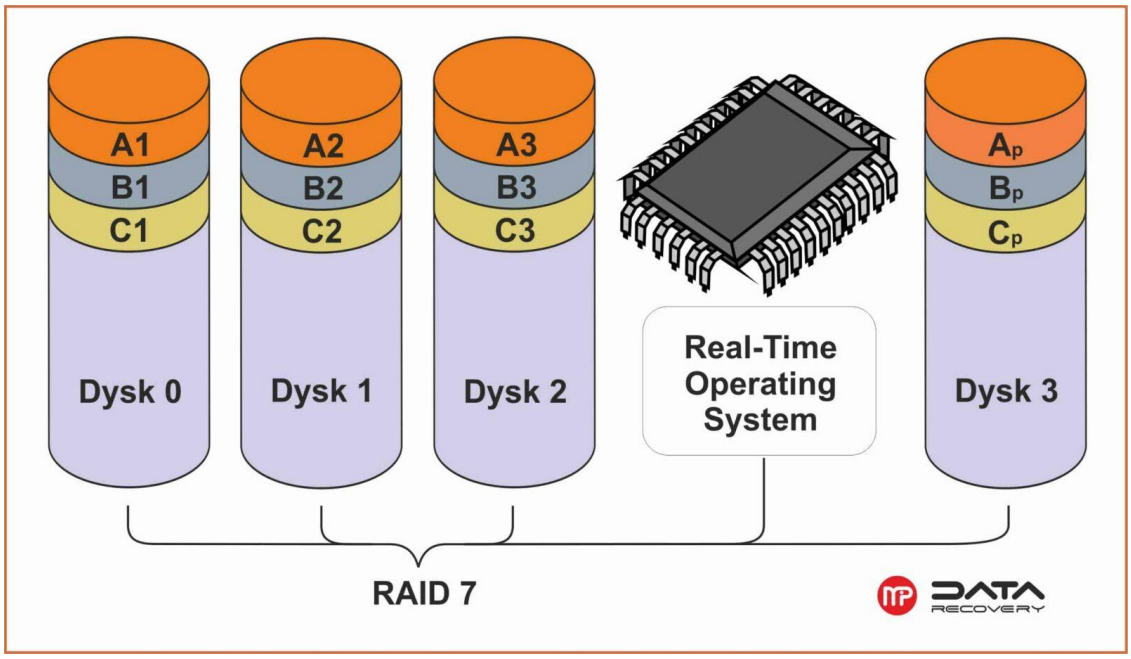


Рисунок 3.16 – Відмовостійкий масив, оптимізований для підвищення продуктивності

Недоліки RAID 7:

- власність одного виробника;
- дуже висока вартість одиницю обсягу;
- короткий гарантійний термін;
- не може обслуговуватись користувачем;
- потрібно використовувати блок безперебійного живлення для запобігання втрати даних із кеш пам'яті.

3.3.12 Варіанти реалізації RAID систем

Порівняльний аналіз рівнів RAID наведено у табл. 3.2.

Існує три основні варіанти реалізації RAID систем:

- програмна (software-based);
- апаратна – шинно-орієнтована (bus-based);
- апаратна - автономна підсистема (subsystem-based).

Головна перевага програмної реалізації – низька вартість. Але при цьому вона має багато недоліків: низьку продуктивність, завантаження додатковою роботою центрального процесора, збільшення шинного трафіку.

Програмно зазвичай реалізують прості рівні RAID – 0 та 1, оскільки вони не вимагають значних обчислень. З огляду на ці особливості RAID системи з програмною реалізацією використовуються в серверах початкового рівня.

Апаратні реалізації RAID відповідно коштують більше ніж програмні, оскільки використовують додаткову апаратуру для виконання операцій виводу. При цьому вони розвантажують або звільняють центральний процесор та системну шину і, відповідно, дозволяють збільшити швидкодію.

Таблиця 3.2 - Порівняння характеристик рівнів RAID

RAID	Мінімум дисків	Потреба дисків	Відмовості йкість	Швидкість передачі даних	Інтенсивність обробки запитів	Практичне використання
0	2	N	< 1 диск	< RAID 3	дуже висока до Nx1 диск	графіка, відео
1	2	2N*	< RAID 6	R>1 диск W=1 диск	до 2x1 диск W=1 диск	малі файл-сервери
2	7	2N<X<N +1	< RAID 1	~ RAID 3	низька	мейнфрейми
3	3	N+1	< RAID 1	< RAID 7	низька	графіка, відео
4	3	N+1	< RAID 1	R<RAID3 W<RAID5	R=RAID0 W << 1 диск	файл-сервери
5	3	N+1	< RAID 1	R<RAID4 W<RAID3	R=RAID0 W<1 диск	сервери баз даних
6	4	N+2	сама висока	низька	R>1 диск W<RAID4	використовується вкрай зрідка
7	12	N+1	< RAID 1	якнайвища	якнайвища	різні типи додатків

Шинно-орієнтовані реалізації є RAID контролерами, які використовують швидкісну шину комп'ютера, в який вони встановлюються (останнім часом зазвичай використовується шина PCI). У свою чергу шинно-орієнтовані реалізації можливо поділити на низькорівневі та високорівневі. Перші зазвичай не мають SCSI чіпів і використовують так званий RAID порт на материнській платі з вбудованим контролером SCSI. При цьому функції обробки коду RAID та операцій введення/виводу розподіляються між процесором на RAID контролері та чіпами SCSI на материнській платі. Таким чином, центральний процесор звільняється від обробки додаткового коду і зменшується трафік шини в порівнянні з програмним варіантом. Вартість таких плат зазвичай невелика, особливо якщо вони орієнтовані на системи RAID - 0 або 1 (є також реалізації RAID 3,5,10,30, 50, але вони дорожчі), завдяки чому вони потроху витісняють програмні продажі з ринку серверів початкового рівня. Високорівневі контролери з шинною реалізацією мають дещо іншу структуру, ніж молодші брати. Вони беруть він всі функції, пов'язані з вводом/виводом і виконанням RAID коду. Крім того, вони не такі залежні від реалізації материнської плати і, як правило, мають більше можливостей (наприклад, можливість підключення модуля для зберігання інформації в кеш у разі відмови материнської плати або зникнення живлення). Такі контролери зазвичай коштують дорожче за низькорівневі і використовуються в серверах середнього і високого рівня. Вони зазвичай реалізують RAID рівнів 0,1,3,5,10,30,50. Враховуючи те, що шинно-орієнтовані реалізації підключаються прямо до внутрішньої PCI шини комп'ютера.

Разом із перерахованими перевагами шинно-орієнтована архітектура має такі недоліки:

- залежність від операційної системи та платформи;
- обмежена масштабованість;
- обмежені можливості щодо організації відмовостійких систем.

Усіх цих недоліків можливо уникнути, використовуючи автономні підсистеми. Ці системи мають повністю автономну зовнішню організацію й у принципі є окремий комп'ютер, який використовується організації систем зберігання інформації. Крім того, у разі розвитку технології оптоволоконних каналів швидкодія автономних систем ні в чому не поступатиметься шинно-орієнтованим системам. Зазвичай зовнішній контролер ставиться в окрему стійку і, на відміну від систем з шинною організацією, може мати велику кількість каналів вводу/виводу, у тому числі й хост-каналів, що дає можливість підключати до системи кілька хост-комп'ютерів та організовувати кластерні системи. У системах з автономним контролером можливо реалізувати гаряче резервування контролерів. Одним із недоліків автономних систем залишається їхня велика вартість.

4 КОМП’ЮТЕРНІ ВІРУСИ

4.1 Типи зловмисного програмного забезпечення

Зловмисне програмне забезпечення (Malicious Software) - це будь-який код, який може використовуватися для викрадання даних, обходу системи керування доступом, пошкодження або компрометації системи. Нижче наведено кілька загальних типів зловмисного програмного забезпечення (ПЗ):

Шпигунські програми (Spyware) – це зловмисне програмне забезпечення, призначене для стеження та шпигування за користувачем. Шпигунські програми часто містять засоби відстеження активності, зчитування з клавіатури та перехоплення даних. Намагаючись подолати заходи безпеки шпигунські програми часто змінюють налаштування безпеки. Шпигунські програми часто прив'язуються до легального програмного забезпечення або до троянських коней.

Рекламне ПЗ (Adware) – призначене для автоматичного поширення реклами. Рекламне ПЗ часто встановлюється разом з деякими версіями програмного забезпечення. Іноді рекламне ПЗ призначене лише для поширення реклами, але досить часто з ним поширюється шпигунське ПЗ.

Бот (Bot) – походить від слова robot, бот – це шкідлива програма, призначена для автоматичного виконання дій, зазвичай в Інтернеті. Хоча більшість ботів безпечні, зростання кількості зловмисних ботів призводить до створення бот-мереж (botnet). Певна кількість комп'ютерів інфікується ботами, які запрограмовані спокійно очікувати команди від зловмисника.

Програми-вимагачі (Ransomware) – це шкідливе ПЗ призначене для блокування комп'ютерної системи або розміщених на ній даних до моменту здійснення викупу. Такі програми зазвичай шифрують дані на комп'ютері за допомогою невідомого користувачу ключа. Деякі інші версії програм-вимагачів можуть для блокування системи використати її певні вразливості. Ransomware розповсюджується через завантажений файл або певну вразливість програмного забезпечення.

Псевдоантивірус (Scareware) – це тип шкідливого ПЗ, що переконує користувача виконати конкретну дію, використовуючи його страх. Scareware створює спливаючі вікна, схожі на діалогові вікна операційної системи. Ці вікна відображають підроблені повідомлення про те, що система знаходиться під загрозою або необхідне виконання відповідної програми для повернення до нормальної роботи. Насправді жодних проблем на комп'ютері немає, і якщо користувач погоджується та дозволяє запустити зазначену програму, його система буде заражена шкідливим ПЗ.

Руткіт (Rootkit) – це зловмисне програмне забезпечення, призначене для змін в операційній системі з метою створення чорного ходу (backdoor). Після чого нападники використовують цей чорний хід для віддаленого доступу до комп'ютера. Більшість руткітів використовують вразливості ПЗ для підвищення привілеїв та модифікації системних файлів. Руткітам також притаманні зміни в системних інструментах експертизи та моніторингу, що дуже ускладнює виявлення цього зловмисного ПЗ. Часто для знищення руткіту необхідноповністю очистити ПК та перевстановити операційну систему.

Вірус (Virus) - це шкідливий виконуваний код, який прикріплюється до інших виконуваних файлів, часто легітимних програм.

Вважається, що термін "комп'ютерний вірус" вперше застосував співробітник Лехаїського університету (США) Ф. Коен на конференції з безпеки інформації в 1984 р. Комп'ютерні віруси також володіють здібністю до самовідтворення з наступним впровадженням копій вірусу у файли, системні області комп'ютера або навіть на інший комп'ютер по мережі. При цьому дублікати також зберігають здібність до подальшого розповсюдження. Як правило, віруси володіють якою-небудь деструктивною дією, хоча це і не є обов'язковою умовою. Здібність до самовідтворення треба розуміти дуже широко. Різні примірники одного вірусу не тільки не зобов'язані повністю збігатися, але можуть, навіть не мати жодного загального байта (йдеться про так звані складнополіморфні віруси).

Більшість вірусів вимагає активації з боку кінцевих користувачів і можуть справцьовувати у певний день або час. Віруси можуть бути нешкідливими і просто відображати рисунок або можуть мати руйнівні наслідки, такі, як зміна або видалення даних. Віруси також можуть бути запрограмовані на мутацію для запобігання виявленню. На сьогодні більшість вірусів поширюється через USB-накопичувачі, оптичні диски, мережні ресурси або електронною поштою.

Троянський кінь (Trojan horse) - це зловмисне ПЗ, яке здійснює шкідливі дії під виглядом бажаної операції. Цей шкідливий код використовує привілеї користувача, який його запускає. Часто трояни містяться в файлах зображень, аудіофайлах або іграх. Троянський кінь відрізняється від вірусу тим, що він приєднується до невиконуваних файлів.

Черв'яки або хробаки (Worms) – це шкідливий код, який клонує себе, самостійно використовуючи вразливості в мережах. Хробаки зазвичай уповільнюють роботу мереж. Якщо для активації вірусу необхідно запустити на виконання програму-носій, то хробаки можуть працювати самостійно. Участь користувача потрібна тільки під час первинного зараження. Після того, як хост заражений, хробак може швидко поширюватися мережею. Хробаки мають схожі шаблони поведінки. А саме, використовують вразливість системи, мають здатність до самостійного розповсюдження і виконання дій на користь зловмисника.

Хробаки є причиною деяких найбільш руйнівних атак в Інтернеті. Як показано на рисунку 1, у 2001 році хробак Code Red інфікував 658 серверів. На рисунку 2 зображено як за 19 годин він інфікував понад 300 000 серверів.

4.2 Класифікація вірусів

Класифікація вірусів за метою дії:

- Завантажувальні (бутові) віруси - віруси, що заражають завантажувальні сектори дискет і жорстких дисків. Існують бутові віруси, завантажувальні сектори дискет, інші заражають - ще і завантажувальні сектори вінчестерів, деякі - заражають головний завантажувальний сектор вінчестера (MBR). Часто віруси "всеїдні" і заражають і те, і інше.
- Файлові віруси - заражають виконувані файли. Крім того, до файлових відносяться так звані macro-віруси - файлові віруси, що заражають файли деяких

систем документообігу, наприклад, MS Office, AmiPro ін. Такі системи мають вбудовані макро - мови (VBA, VB). Ці мови володіють достатніми можливостями, щоб проводити практично всі операції, необхідні вірусу. Для розмноження в середовищі MS Office такі віруси використовують файл шаблону за умовчанням normal.dot.

- Віруси, що вражають код BIOS комп'ютера.
- Скрипт - віруси створюються на мові сценаріїв VBS, розробленій Microsoft. Така мова виконується інтерпретатором Windows, крім того, веб - браузером MS Internet Explorer. Подібні віруси можуть видаляти файли на жорсткому диску, змінювати настройки системи, запускати троянських коней і ін. Якщо у веб - браузері неправильно настроєний розділ «Безопасность», то загроза інфікування такими вірусами існує при відвідуваннях веб - сторінки, що містять шкідливий VBS - скрипт.
- Мережеві черв'яки - це шкідницькі комп'ютерні програми, які можуть створювати свої копії на одному і тому ж комп'ютері чи ж копіювати себе на інші комп'ютери мережі, використовуючи найчастіше систему електронної пошти. Розсилка черв'яків за адресами адресної книги поштових клієнтів призводить до вибухоподібного зростання повідомлень, що пересилаються, що може привести до перевантаження каналів і відмови в обслуговуванні.
- Троянські коні - це програми, які маскуються під які-небудь корисні застосування (наприклад, утиліти або навіть антивірусні програми), але при цьому завдають різні руйнівні дії. Трояни не вбудовуються в інші файли і не володіють здібністю до власнодублювання. В порівнянні з іншими типами вірусів «троянські коні» мало поширені, оскільки після запуску вони або знищують себе разом з рештою даних на диску, або знищуються самим постраждалим користувачем.
- Backdoor-програми - утиліти прихованого адміністрування аналогічні троянським коням, але на відміну від останніх, самі не виконують активних дій, а тільки відчиняють доступ на комп'ютер зловмиснику.

Класифікація за властивостями вірусів:

- Віруси, що маскуються (Stealth). Видів маскування безліч, але всі вони засновані на перехопленні вірусами переривань BIOS і операційної системи. Перехопивши переривання, віруси контролюють доступ до заражених об'єктів. Наприклад, при прогляданні зараженого об'єкту, вони можуть "підсунути" замість нього здоровий. Крім того, віруси спотворюють інформацію BIOS (наприклад, повертають невірне значення довжини файлу, приховуючи свою присутність в ньому). Для більшості антивірусних програм віруси, що використовують стелс - технологію, є серйозною проблемою. Винятком є програми-ревізори дисків (наприклад, AdInf).
- Поліморфні віруси. Аналіз вірусів полягає у виділенні в них сигнатур (послідовності байт, специфічної (у ідеалі - унікальної) для конкретної програми) і подальшому пошуку в потенційних об'єктах вірусної атаки. Поліморфні віруси шифрують свій код. Мета такого шифрування достатньо очевидна: маючи заражений і оригінальний файли, і, отже, маючи можливість виділити вірус "в чистому вигляді", не представляється можливим проаналізувати його код за допомогою звичайного дизасемблювання. Розшифровка проводиться самим вірусом вже безпосередньо під час виконання. При цьому можливі самі різні варіанти: вірус може розшифрувати себе всього відразу, а може виконувати таку розшифровку по

частинах, може знов шифрувати вже відпрацьовані ділянки і т.д. (причому іноді не тим способом, яким вони були зашифровані раніше). Проте зашифровані віруси обов'язково містять деякий незашифрований фрагмент - розшифровщик (або його частка). По ньому можливо побудувати сигнатуру такого вірусу і далі вже боротися з ним звичайними способами. Ситуація змінилася, коли були розроблені алгоритми, що дозволяють не тільки шифрувати код вірусу, але і змінювати розшифровщики - кожна нова копія вірусу містить новий розшифровщик, який може в кожному біті відрізнятися від розшифровщика, що створив похідні копії. Навіть з простих комбінаторних міркувань ясно, що розшифровщиків, довжина яких звичайно обмежена, не може бути нескінченно багато. Але якщо їх всього лише два-три трильйона, то ясно, що задача перебору всіх можливих не стоїть. Віруси, що використовують описану технологію, одержали назву поліморфних.

- Резидентні віруси відрізняються від нерезидентних тим, що після запуску інфікованої програми вони залишаються в оперативній пам'яті комп'ютера.

Незалежно від того, яким типом зловмисного ПЗ інфікована система, можливо виділити найпоширеніші симптоми того, що пристрій заражено:

- збільшується навантаження на центральний процесор;
- знижується швидкодія комп'ютера;
- комп'ютер часто зависає або дає збої;
- знижується швидкість перегляду веб-сторінок;
- з'являються незрозумілі проблеми з мережними з'єднаннями;
- модифікуються файли;
- видаляються файли;
- з'являються невідомі файли, програми або значки на робочому столі;
- запущено невідомі процеси;
- програми самостійно припиняють виконання або змінюють свої налаштування;
- електронна пошта надсилається без відома та згоди користувача.

4.3 Типи антивірусних засобів

Можливо здійснити наступну класифікацію антивірусних засобів:

- Сканери - поліфаги. Пошук вірусів в простому випадку зводиться до пошуку їх сигнатур. Після виявлення вірусу в тілі програми поліфаг знешкоджує його. Для цього розробники антивірусних засобів ретельно вивчають роботу кожного конкретного вірусу: що він псує, як він псує, де він ховає те, що зіпсує і т. д. В більшості випадків поліфаг здатний благополучно видалити вірус і відновити працездатність зіпсованих програм. Але необхідно добре розуміти, що це можливо не завжди. Сучасні антивірусні програми містять засоби евристичного аналізу, що дозволяють розпізнавати деструктивний код в нових, ще не відомих поліфагу вірусах.

- Сторожа - невеликі резидентні програми, перевіряючі завантажувані в пам'ять файли на предмет знаходження в них сигнатур вірусів. Перевага - швидке виявлення вірусів. Недолік - сильне завантаження систем і зниження їх продуктивності.

- Ревізори - програми, у функції яких входить антивірусна перевірка шляхом контролю властивостей файлів (розміру, дати і часу створення / зміни,

контрольної суми і ін.). При цьому виконується порівняння властивостей файлів з їх первинними значеннями (дуже важливо, щоб вони не належали вже зараженим файлам). Внаслідок перевірки користувачу виводиться список файлів із зміненими властивостями, і користувач вирішує, які з файлів змінив він сам, а які можливо заражені. Часто ревізори працюють в тандемі зі сканерами - поліфагами і автоматично передають їм на перевірку файли із зміненими властивостями.

- Антивірусні вакцини, які використовувалися для обробки файлів і завантажувальних секторів. Вакцини бувають пасивними (приклад| такої вакцини описаний нижче) і активними. Активна вакцина, "заражаючи" файл, подібно до вірусу, оберігає його від будь-якої зміни і у ряді випадків здатна не тільки виявити сам факт зараження, але і вилікувати файл. Пасивні вакцини використовувалися (тепер це вже велика рідкість) для запобігання зараження файлів деякими вірусами, що використовують прості ознаки їх зараженості - "дивні" час або дата створення, певні символьні рядки і ін. Нині вакцинація широко не застосовується.

Крім того, існують спеціальні додаткові пристрої (звичайно електронна плата із спеціалізованим процесором), що забезпечують досить надійний захист. Перевагою плати є незалежність від операційної системи і BIOS (які можуть бути заражені), а також практично відсутність використання ресурсів комп'ютера. Прикладом апаратних засобів антивірусного захисту може служити плата Sheriff.

Існує таке поняття як вірусна містифікація (Virus hoax) - помилкове попередження про нібито грізному вірусі, який насправді не існує. Наприклад, поступило по e-mail попередження про загрозу небезпечного вірусу, який записує своє тіло в один з файлів в каталозі Windows. В результаті багато користувачів помилково видаляли цей необхідний для функціонування операційної системи файл.

5 ЗАХИСТ ІНФОРМАЦІЇ ПІД ЧАС РОБОТИ В КОМП'ЮТЕРНИХ МЕРЕЖАХ

Більшість реалізацій стека TCP/IP успадковують функції стека, розробленого більше 40 років тому, коли вимоги безпеки практично були відсутні і їм спочатку притаманна вразливість. Стек TCP/IP - базовий стек для Інтернет, як і більшість програмних засобів (включно з операційними системами FreeBSD, Linux) є відкритими і існує можливість використання хакерами "дір", які з'являються в тому чи іншому продукті і висвітлюються в Інтернет.

5.1 Класифікація мережевих атак та засоби їх попередження

Мережева розвідка - збір інформації про мережу за допомогою загальнодоступних даних та програм, наприклад, використання ping для виявлення адрес мережі, сканування портів, запити до DNS для дозволу адрес в імена хостів і навпаки.

Сканування портів - це процес зондування комп'ютера, сервера або іншого мережного пристрою для виявлення відкритих портів. В мережах кожному застосунку, запущеному на пристрої, призначається ідентифікатор, який називається номером порту. Саме завдяки використанню цього номера на обох кінцях сеансу зв'язку необхідні дані передаються до потрібного застосунку. Сканування портів може використовуватися зі зловмисною метою як розвідувальний інструмент для ідентифікації операційної системи та служб, що працюють на комп'ютері або іншому пристрої. А також цей підхід застосовується мережним адміністратором для перевірки політик безпеки в мережі.

Для захисту від атак цього типу можливо рекомендувати:

- використання систем виявлення атак – Intrusion Detection System (IDS);
- заборона проходження пакетів ICMP.

Система виявлення вторгнень (Intrusion Detection System, IDS) може бути окремим мережним пристроєм або одним з кількох інструментів на сервері чи міжмережному екрані, який сканує трафік відповідно до бази даних правил або сигнатур у пошуках шкідливих атак. При виявленні відповідності правилам або сигнатурам IDS реєструє виявлення атаки та створює попередження для мережного адміністратора. Система виявлення вторгнень не запобігає атакам, оскільки не вживає ніяких дій при виявлення інциденту. Завдання IDS - виключно виявлення, реєстрація та створення попередження.

Сканування, яке виконує IDS, уповільнює роботу мережі (відоме як латентність). Щоб запобігти затримці у мережі , IDS зазвичай розміщують офлайн (в автономному режимі), окремо від звичайного мережного трафіку. Дані копіюються або віддзеркалюються комутатором, а потім пересилаються до IDS для проведення аналізу в режимі офлайн. Існують також інструменти IDS, які можливо встановлювати поверх операційної системи комп'ютера, наприклад Linux або Windows.

Система запобігання вторгненням (Intrusion Prevention System, IPS) має можливість блокувати або відхиляти трафік на основі спрацювання правила або відповідної сигнатури. Однією з найбільш відомих систем IPS/IDS є Snort. Комерційна версія Snort - це Cisco’s Sourcefire. Sourcefire має можливість у

реальному часі виконувати аналіз трафіку і портів, ведення журналів, пошук і зіставлення контенту, а також виявлення спроб зондування, атак і сканування портів. Це рішення також інтегрується з іншими сторонніми інструментами для створення звітності, аналізу лог-файлів та продуктивності.

Використання вразливостей - це ще один розповсюджений метод проникнення. Нападники сканують комп'ютери, щоб отримати інформацію про них. Нижче наведено загальний метод використання вразливостей:

- Крок 1. Збір інформації про систему жертви. Це можливо зробити багатьма різними способами, такими як сканування портів або застосування методів соціальної інженерії. Мета полягає в тому, щоб дізнатись якомога більше про комп'ютер жертви.
- Крок 2. На кроці 1 зловмисник може дізнатися таку цінну інформацію, як встановлена операційна система, її версія та перелік сервісів, які працюють на ній.
- Крок 3. Коли операційна система та її версія відомі, атакуючий шукає будь-які специфічні вразливості, виявлені для цієї версії ОС або інших сервісів.
- Крок 4. Коли виявлена вразливість, атакуючий шукає раніше написаний для неї експлойт. Якщо таких експлойтів не було створено, зловмисник може розглянути можливість його написання.

Парольні атаки – виконуються з метою виявлення паролів користувачів на доступ до системи. Найпростіший варіант - перехоплення мережевих пакетів з даними реєстрації telnet-сервера, FTP-сервера або POP3-сервера, складніші - копіювання файлів, що зберігають бази з ідентифікаторами користувачів та паролями (Sam для Windows, masterpasswd для UNIX-систем) та їх зламування методами перебору за словником або повним перебором (brute-force).

Загроза парольних атак може бути знижена такими способами:

- використанням складних паролів, що включають малі та великі символи, цифри, службові символи та мають достатню довжину (не менше 8 символів).
- використанням "сильної" аутентифікації - наприклад, з використанням біометричних засобів або одноразових паролів. Застосування методів автентифікації, які не надсилають відкритий пароль по мережі.
- розробкою та впровадженням політики безпеки - "формальних правил, яким повинні підкорятися люди, які отримують доступ до корпоративної технології та інформації" - (RFC 2196).

Сніфери пакетів - програми, що використовують мережну карту, яка працює в "нерозбірливому режимі" (promiscuous mode), і що копіюють у свій буфер всі пакети сегмента мережі (домена колізій - для Ethernet). Ці програми призначені для діагностики несправностей та аналізу трафіку і навіть включаються до операційних систем (Мережевий Монітор Windows, tcpdump UNIX). У той же час ці засоби можуть бути використані для перехоплення конфіденційних даних (насамперед мережевих ідентифікаторів та паролів користувачів), які в класичних реалізаціях мережевих сервісів (Telnet, FTP, SMTP, POP3) передаються у пакетах у відкритому вигляді.

Пом'якшити загрозу сніфінгу пакетів можливо за допомогою:

- використання комутованої інфраструктури - при використанні комутаторів сніффер може захопити трафік тільки порту комутатора, до якого він підключений.
- шифрування сеансу передачі - наприклад з використанням VPN і протоколів IPSec, SSH, SSL та інших.
- антисніферів - апаратних або програмних засобів, що вимірюють час реагування хостів на тестовий пакет і визначають хости, що обробляють більше, ніж зазвичай, кількість пакетів в одиницю часу (наприклад, AntiSniff) .
- "сильна" аутентифікація - наприклад, з використанням біометричних засобів або одноразових паролів.

Спуффінг мережевих адрес - підстановка в пакетах хакером мережного адреси комп'ютера користувача мережі (можливо змінювати як адресу мережного, і адресу канального рівня) замість свого. Часто спуфінгом вважають також вставку хибної інформації або шкідливих команд у звичайний потік даних, наприклад, широкомовне розсилання хибного DHCP сервера, що автоматично призначає мережеві адреси, необхідні зловмиснику (далі комп'ютер зловмисника, наприклад, може імітувати мережеві сервіси та отримувати ідентифікацію їх реєстрації на "хибних серверах").

Пом'якшити загрозу спуфінгу пакетів можливо за допомогою:

- фільтрів мережевих екранів, що відсікають будь-який трафік, що надходить із зовнішньої мережі з вихідною адресою, що має розташовуватися всередині мережі;
- фільтрації, що рекомендується RFC 2827 - відкидає вихідний трафік з IP адрес, що не належать організації. використанням діапазону адрес для локальних мереж і технології трансляції мережевих адрес, коли шлюз замінює мережеву адресу відправника в пакетах, що відправляються з локальної мережі, на адресу свого вихідного інтерфейсу.

Відмова в обслуговуванні (Denial of Service - DoS) - один із найпоширеніших типів атак, відносно простий і в той же час завдає величезної шкоди мережам і серверам. Атака DoS робить мережу або мережеві послуги недоступними для звичайного використання за рахунок перевищення допустимих меж навантаження на мережу, операційну систему або додаток серверів.

Розрізняють два основні типи DoS-атак:

- надмірна кількість даних (Overwhelming Quantity of Traffic) - це коли до мережі, хоста або програми надсилається величезна кількість даних з такою швидкістю, що вони не спроможні їх обробити. Це спричиняє затримку передачі чи відгуку, або відмову пристрою, аварійне завершення роботи сервісу.
- пакети зловмисного формату (Maliciously Formatted Packets) - це коли пакет зловмисного формату надсилається до хоста або програми і одержувач не здатен його обробити. Наприклад, зловмисник пересилає пакети, що містять помилки, які не можуть бути ідентифіковані програмою, або передає неправильно відформатовані пакети. Це може викликати сповільнення роботи або відмову пристрою-отримувача.

DoS атаки вважаються серйозною загрозою, оскільки вони можуть легко переривати зв'язок та викликати значну втрату часу та грошей. Ці атаки відносно прості для виконання навіть некваліфікованим нападником.

Розподілена DoS атака (Distributed DoS Attack, DDoS) подібна до атаки DoS, але вона походить від декількох скоординованих джерел. Наприклад, атака DDoS може відбуватися наступним чином:

1. Нападник створює мережу заражених хостів, яка називається ботнетом. Заражені хости називаються "зомбі". Їх контролюють системи керування.

2. Зомбі-комп'ютери постійно сканують і заражають нові хости, збільшуючи кількість "зомбі". У призначений час, хакер наказує центрам керування розпочати DDoS-атаку за участі зомбі-ботнету.

Прикладами DoS-атак можливо назвати TCP SYN Flooding, Ping of Death, використання мережевих черв'яків і троянських коней для організації з допомогою Distributed DoS - DDoS та інших.

Загроза атак DoS може бути знижена такими способами:

- використанням функцій анти-DoS на маршрутизаторах та мережевих екранах, що обмежують кількість напіввідкритих каналів в одиницю часу;
- обмеженням обсягу трафіку в одиницю часу (traffic rate limiting), що насамперед використовується для атак - ICMP, UDP, TCP з прапором PSH, на маршрутизаторах;
- регулярною перевіркою вірусів, оновленням антивірусних баз та системного та прикладного програмного забезпечення;
- використанням програмного забезпечення, що виявляє сканування портів.

Атаки типу Man-in-the-Middle - використання даних, що передаються по мережах. За допомогою такого рівня доступу зловмисник може перехоплювати, збирати та підмінювати інформацію користувача перш ніж передати її одержувачу. Атаки MitM широко використовуються для викрадення фінансової інформації. Існує безліч зловмисних програм та методик, які наділяють атакуючих можливостями MitM.

Посередник в мобільному телефоні (Man-In-The-Mobile або MitMo) – варіант атаки "Людина посередині", який використовується для отримання контролю над мобільним пристроєм. Заражений мобільний пристрій може отримати наказ зібрати конфіденційну інформацію користувача і відправити її нападникам. ZeuS, як приклад експлойта з можливостями MitMo, дає змогу атакуючим непомітно перехоплювати SMS-повідомлення, які надсилаються користувачам під час проведення 2-етапної перевірки.

Для боротьби з атаками типу Man-in-the-Middle слід використовувати:

- шифрування сеансів передачі.
- регулярне встановлення виправлень операційних систем та додатків, що випускаються їх виробникам та виконання рекомендацій виробників щодо забезпечення безпеки програмного забезпечення.

Атаки на рівні програм - використання проломів у захисті серверного програмного забезпечення (HTTP, FTP, sendmail та ін.) з метою отримати доступ до системи з правами інших користувачів (наприклад, адміністратора). У цьому випадку використовуються порти та адреси, які не можуть бути закриті. Тому для захисту від атак цього типу можливо рекомендувати:

- регулярний перегляд лог-файлів та їх аналіз.

- регулярне встановлення виправлень операційних систем та додатків, що їх випускають їх виробникам та виконання рекомендацій виробників щодо забезпечення безпеки програмного забезпечення.
- використання систем виявлення атак - Intrusion Detection System (IDS), які порівнюють проходять через хост (або домен) пакети з сигнатурами відомих атак і в разі збігу подають сигнал адміністратору.

Фішинг, це коли зловмисник надсилає шахрайського електронного листа, який виглядає як повідомлення від легального надійного джерела. Мета цього повідомлення - змусити одержувача встановити зловмисне ПЗ на своєму пристрої або розкрити особисту чи фінансову інформацію. Прикладом фішингу є підробка листів електронної пошти, які виглядають як повідомлення, надіслані роздрібним магазином з проханням до користувача натиснути на посиланні для участі у розігруванні призів. Посилання може переадресовувати на підроблений сайт, який запитує особисту інформацію, або може встановити вірус.

Спрямований фішинг - це цілеспрямована фішингова атака. Щоб дістатися жертв фішинг і спрямований фішинг використовують електронні листи, але спрямований фішинг націлений на конкретну особу. Переш ніж надіслати електронне повідомлення нападник вивчає інтереси жертви. Наприклад, зловмисник дізнався, що жертва цікавиться автомобілями і хоче купити конкретну модель. Зловмисник приєднується до того ж дискусійного автомобільного форуму, що і жертва, готує пропозицію продажу автомобіля і надсилає її жертві електронною поштою. Електронний лист містить посилання на фотографію автомобіля. Коли жертва натискає на посилання, шкідливе ПЗ встановлюється на її комп'ютер.

Ще один зі способів проникнення - це вдосконалені постійні загрози (advanced persistent threats - APTs). Вони складаються з багатофазних, довготривалих, непомітних та складних дій, спрямованих на конкретну жертву. APT-атаки є складними і для їх виконання необхідні досвід та високий рівень кваліфікації нападника, тому вони зазвичай добре фінансуються. APT спрямовані на організації або країни з комерційних або політичних причин.

Метою APT є розгортання спеціального шкідливого програмного забезпечення на одній або кількох системах жертви. Зазвичай атаки пов'язані зі шпигунством в мережах та залишаються невиявленими. Через багатоетапне виконання та використання декількох типів налаштовуваних шкідливих програм, які впливають на різні пристрої та виконують визначені функції, окремий нападник часто не має навичок, ресурсів або наполегливості для виконання APT.

Пошукові системи, такі як Google, працюють шляхом ранжування сторінок та надання релевантних результатів на основі пошукових запитів користувачів. Залежно від значимості вмісту веб-сайту, він може відображатися вище або нижче в переліку результатів пошуку. Пошукова оптимізація (Search Engine Optimization, SEO) - це набір методів, що використовуються для покращення рейтингу веб-сайту в пошуковій системі. В той час як багато легальних компаній спеціалізуються на оптимізації веб-сайтів з метою їх кращого позиціонування, зловмисник може використовувати SEO для того, щоб його шкідливий веб-сайт відображався вище у результатах пошуку. Ця техніка називається "отруєння SEO".

Найпоширенішою метою отруєння SEO є збільшення трафіку на шкідливі веб-сайти, які можуть містити зловмисні програми або застосовувати прийоми соціальної інженерії. Щоб розташувати шкідливий сайт вище у результатах пошуку, зловмисники використовують популярні пошукові терміни.

Змішані атаки (Blended attacks) - це атаки, які використовують кілька методів для компрометації жертви. Використовуючи одразу декілька різних методів атаки, зловмисники застосовують шкідливі програми, які є гібридом хробаків, троянських коней, шпигунських програм, клавіатурних шпигунів, спаму та фішингових схем. Тенденція змішаних атак схильна до використання більш складного зловмисного програмного забезпечення та створює великі ризики для користувачів.

Найпоширеніший тип змішаної атаки використовує спам-листи, миттєві повідомлення або легальні веб-сайти для поширення посилань через які шкідливе або шпигунське програмне забезпечення непомітно завантажується на комп'ютер. Ще однією поширеною змішаною атакою є використання DDoS у поєднанні з фішинговими електронними листами. Спочатку DDoS-атака використовується для блокування веб-сайту популярного банку та надсилання його клієнтам електронних листів з вибаченнями за незручності. Ці електронні листи також скеровують користувачів до підробленого "аварійного сайту", де їх інформація авторизації може бути викрадена.

Більшість з найнебезпечніших комп'ютерних черв'яків, таких як Nimbda, CodeRed, BugBear, Klez і Slammer краще класифікувати як змішані атаки:

1. Деякі різновиди Nimbda розповсюджувалися через вкладення в електронні листи; завантаження файлів зі скомпрометованого веб-сервера та обмін файлами Microsoft (наприклад, через анонимні ресурси).
2. Інші різновиди Nimbda могли змінювати привілеї гостьових облікових записів системи, щоб надати зловмиснику або шкідливому коду адміністративні права.
3. До змішаних атак також належать нещодавно створені черв'яки Conficker і ZeuS/LICAT. Conficker використовував всі традиційні методи розповсюдження.

5.2 Міжмережеві екрани

Міжмережевий екран (firewall, брендмауер) - система міжмережевого захисту, що дозволяє розділити загальну мережу на дві або більш частин і реалізувати набір правил, визначальних умови проходження пакетів з даними через кордон з однієї частини загальної мережі в іншу. Як правило, ця межа проводиться між корпоративною (локальної) мережею і глобальною мережею Internet, хоча її можливо організувати і усередині корпоративної мережі підприємства. Міжмережевий екран пропускає через себе весь трафік, ухвалюючи для кожного пакету рішення - пропускати його або відкинути - на основі визначених для нього набору правил фільтрації (рис. 5.1).

Існують чотири основні засоби, за допомогою яких міжмережеві екрани здійснюють контроль доступу і забезпечують реалізацію політик захисту:

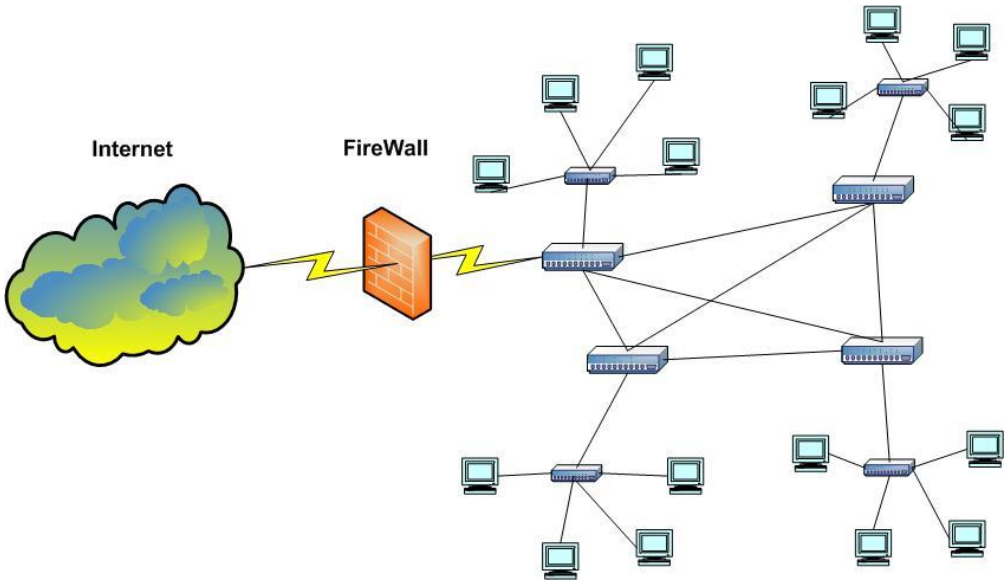


Рисунок 5.1 – Схема встановлення міжмережевого екрану

- управління сервісами - визначення типів служб Internet, до яких можливо дістати доступ з внутрішньої мережі назовні і із зовнішнього оточення усередину; міжмережевий екран може фільтрувати потік даних на основі IP-адрес і номерів портів TCP;
- управління напрямом руху - визначення напряму, в якому можуть ініціюватися і проходити через міжмережевий екран запити до тих або інших служб;
- управління користувачами - надання доступу до служб залежно від прав доступу користувачів, що звертаються до цих служб; ця функція звичайно застосовується до локальних користувачів; однак вона може застосовуватися і до потоку даних, що поступає від зовнішніх користувачів, що вимагає реалізації в якійсь формі технології аутентифікації, наприклад, забезпечуваної протоколом IPSec;
- управління поведінкою – контроль за використанням окремих служб; так, міжмережевий екран може фільтрувати електронну пошту, відсіваючи спам чи ж вирішувати доступ ззовні певної частини інформації, що знаходиться на локальному Web-сервері.

Можливо виділити наступні типи міжмережевих екранів:

- фільтруючі маршрутизатори (packet-filtering router);
- шлюзи мережевого рівня;
- шлюзи прикладного рівня.

Звичайно міжмережеві екрани включають всі або більшість з цих компонент.

Фільтруючий маршрутизатор є маршрутизатором або працюючою на сервері програмою, що фільтрує пакети, які входять в корпоративну мережу і виходять з неї. Фільтрація пакетів звичайно здійснюється на основі інформації, що міститься в IP і TCP заголовках пакетів, найчастіше для цього використовуються:

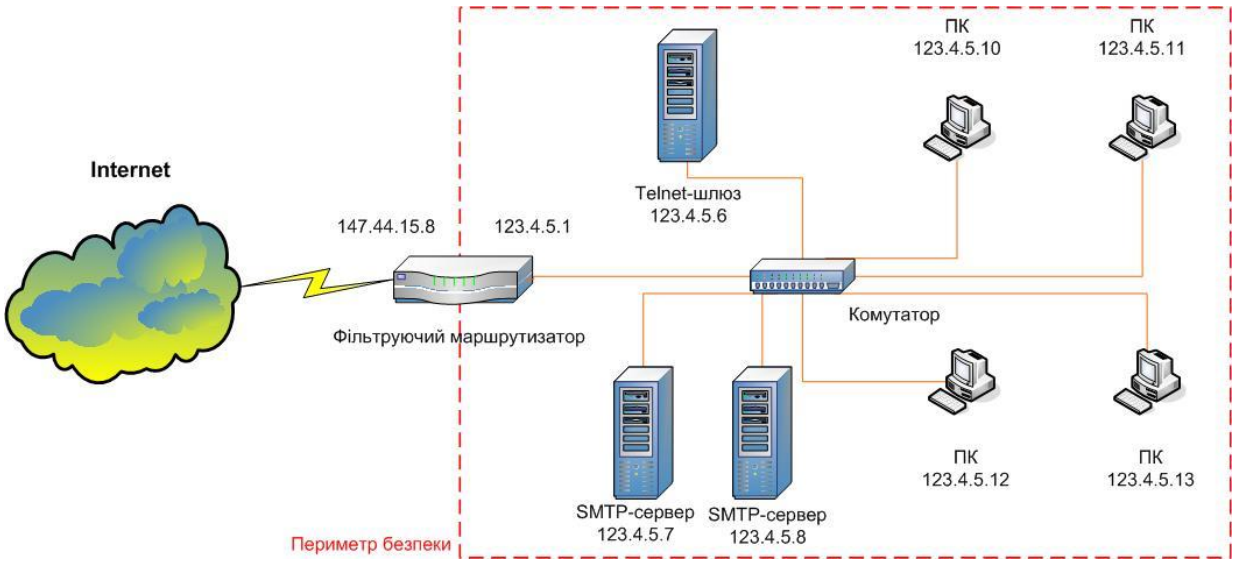
- IP адреса відправника пакету;
- IP адреса одержувача пакету;

- порт системи відправника;
- порт системи одержувача.

Фільтр пакетів звичайно представляється у вигляді списку правил, що використовують значення полів заголовків IP і TCP. Якщо виявляється відповідність одному з правил, то на підставі цього правила ухвалюється рішення про можливість передачі пакету далі. При невідповідності всім правилам виконується операція, передбачена для використання за умовчанням. Для неї існують наступні варіанти локальних політик захисту:

- Default = discard. Все, що не дозволено, заборонено.
- Default = forward. Все, що незаборонено, дозволено.

Як приклад роботи фільтруючого маршрутизатора розглянемо реалізацію політики безпеки, що допускає певні з'єднання з локальною корпоративною мережею 123.4/16 (рис. 5.2).



Протокол	Адреса відправника	Адреса одержувача	Порт відправника	Порт одержувача	Дія
TCP		123.4.5.6	>1023	23	allow
TCP		123.4.5.7	>1023	25	allow
TCP		123.4.5.8	>1023	25	allow
TCP	1239.6.48.254	123.4.5.9	>1023	119	allow
UDP		123.4/16	>1023	123	allow
*	*	*	*		deny

Рисунок 5.2 – Приклад набору правил фільтрації

При цьому з'єднання по протоколу telnet дозволені тільки з хостом 123.4.5.6, що виконує роль прикладного telnet-шлюзу, а з'єднання по протоколу SMTP - тільки з двома хостами 123.4.5.7 і 123.4.5.8, що виконують роль серверів електронної пошти. Обмін по протоколу NNTP дозволений тільки від зовнішнього сервера новин з адресою 129.6.48.254 і лише з NNTP сервером

локальної мережі 123.4.5.9. Використання мережевої служби синхронізації часу NTP дозволене для всіх комп'ютерів локальної мережі.

Позитивні якості фільтруючих маршрутизаторів:

- порівняно невисока вартість;
- гнучкість у визначенні правил фільтрації;
- невелика затримка при проходженні пакетів.

Недоліки фільтруючих маршрутизаторів:

- внутрішня мережа маршрутизується з Internet;
- правила фільтрації важкі в описі і відсутні засоби їх тестування;
- відсутня аутентифікація на призначеному для користувача рівні.

Шлюзи мережевого рівня інколи називають системами трансляції мережевих адрес (Network Address Translation - NAT), оскільки вони виключають пряму взаємодію між авторизованим клієнтом і зовнішнім хост - комп'ютером. Шлюз мережевого рівня приймає запит довіреного клієнта на конкретні послуги і після перевірки чи задовольняє клієнт базовим критеріям фільтрації (наприклад, чи може DNS - сервер визначити IP-адреса клієнта і асоційоване з ним ім'я) встановлює з'єднання із зовнішнім хостом. Шлюз мережевого рівня виконує процедуру трансляції адрес, при якій відбувається перетворення внутрішніх IP адрес локальної мережі на одну "надійну" IP адресу інтерфейсу міжмережевого екрану, через який передаються всі витікаючі з локальної мережі пакети.

Кожен раз, коли витікаючий пакет передається через NAT - маршрутизатор локальна IP адреса відправника замінюється на "чесну" IP адресу (рис. 5.3).

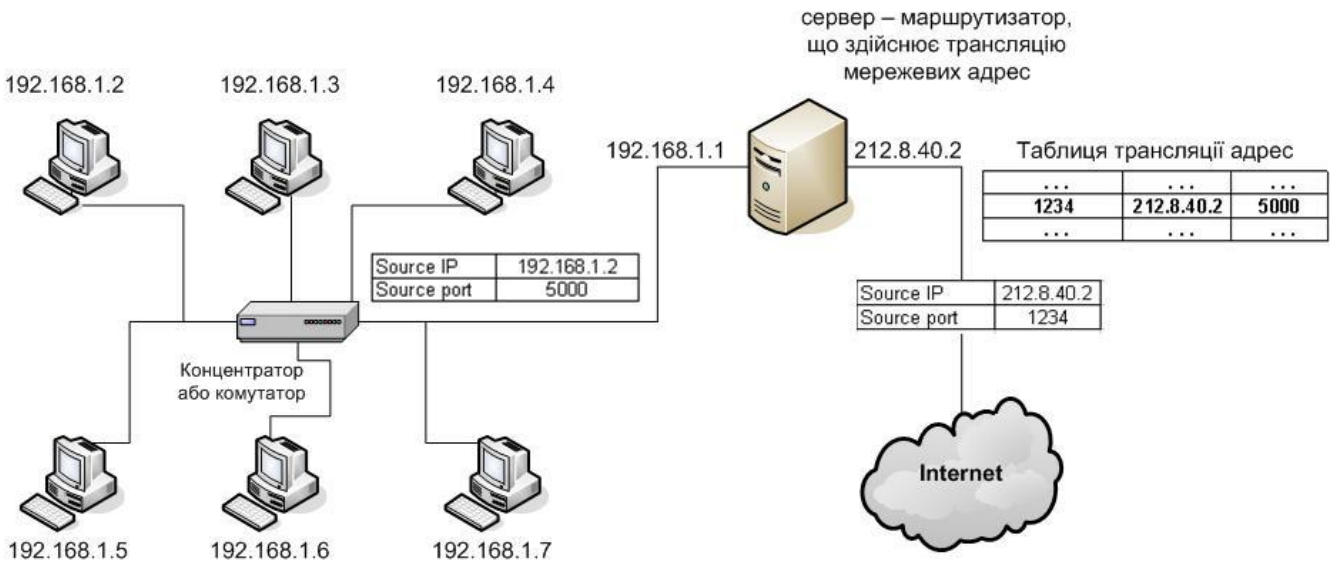


Рисунок 5.3 – Трансляція мережевих адрес шлюзом мережевого рівня

Окрім цього, в полі порту відправника заголовка TCP заноситься 16-бітовий індекс таблиці трансляції адрес (з максимум 65536 рядками). Кожен рядок таблиці містить цей індекс, первинну IP адресу відправника і первинний порт відправника. Після описаної заміни прораховуються контрольні суми заголовків IP і TCP і заносяться у відповідні поля пакету. Коли NAT- маршрутизатор отримує у відповідь пакети для даного процесу, значення порту одержувача в заголовку TCP використовується для пошуку індексу в таблиці трансляції адрес. З

визначеного по індексу рядка витягується локальна IP адреса відправника і первинний TCP порт відправника, які заносяться у відповідні поля пакету. Після цього знову перераховуються контрольні суми в полях IP і TCP заголовків, заносяться у відповідні поля пакету, після чого пакет прямує на робочу станцію локальної мережі.

Шлюз мережевого рівня відстежує процедуру квитування зв'язку по протоколу TCP через порядкові номери сегментів даних. Коли сеанс завершується, шлюз видаляє відповідний елемент з таблиці і розриває з'єднання. Недоліком шлюзів мережевого рівня є те, що після встановлення зв'язку вони не можуть перевіряти вміст пакетів на прикладному рівні. Також для них відсутня призначена для користувача аутентифікація.

Шлюзи прикладного рівня також виключають пряму взаємодію між авторизованим клієнтом і зовнішнім хостом, проте при цьому вони фільтрують всі вхідні і витікаючі пакети на прикладному рівні. Це досягається установкою і настройкою програмного забезпечення повноважних серверів – проксі - серверів (proxy servers), що перенаправляють через шлюз інформацію прикладного рівня, що генерується хостами (рис. 5.4). Користувач зв'язується з цим шлюзом за допомогою такого побудованого на основі TCP/IP застосування, як HTTP-браузер, Telnet-термінал або FTP- клієнт. Ці застосування передають шлюзу ім'я віддаленого вузла, до якого необхідно дістати доступ. Шлюз зв'язується з відповідним застосуванням віддаленого вузла і ретранслює сегменти TCP, що містять дані застосування локального користувача. Шлюз можливо настроїти так, щоб він підтримував тільки певні можливості застосування, які адміністратор мережі вважає допустимими з погляду безпеки.

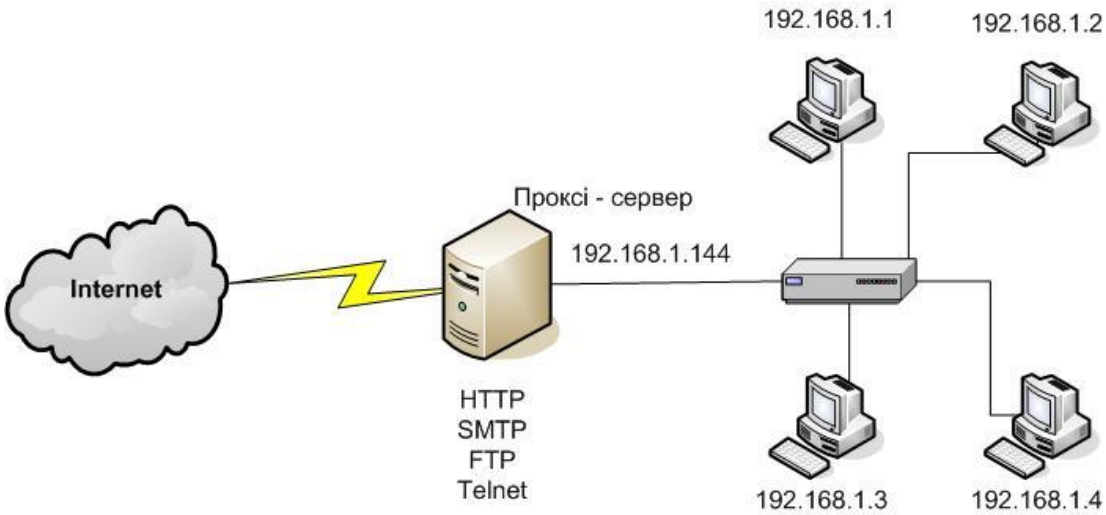


Рисунок 5.4 – Шлюз прикладного рівня – проксі – сервер

У загальному випадку шлюзи прикладного рівня забезпечують надійніший захист, ніж фільтри пакетів. Замість того щоб перевіряти численні можливі комбінації дозволів і заборон на рівні TCP і IP, шлюзу прикладного рівня доводиться розглядати лише обмежене число застосувань. Крім того, при цьому легко протоколювати і контролювати весь вхідний потік даних прикладного рівня. Проте, для досягнення вищого рівня безпеки і гнучкості шлюзи

прикладного рівня і фільтруючі маршрутизатори можуть об'єднуватися в одному міжмережевому екрані. Переваги використання проксі-серверів:

- пропускають пакети лише тих служб, які їм доручено обслуговувати (частіше за все HTTP, HTTPS, FTP, Gopher);
- приховують структуру мережі, що захищається;
- дозволяють виконувати кешування запрошуваної Інтернет інформації і будувати ієрархію кешів;
- дозволяють виконувати аутентифікацію користувачів;
- дозволяють вести статистику відвідувань за витікаючими IP адресами.

Основним недоліком шлюзів даного типа є додаткове навантаження на процесор, що створюється кожним з'єднанням. Насправді між двома кінцевими користувачами встановлюється два зв'язані з'єднання, загальною точкою яких є шлюз, і шлюзу доводиться перевіряти весь потік даних в обох напрямах, щоб переправити його далі.

Міжмережеві екрани можуть бути інтегровані в маршрутизатори або виконані як окремі пристрої.

З роками, комп'ютерні та мережні атаки стали більш витонченими, тому були розроблені нові типи міжмережних екранів для задоволення різних потреб у процесі захисту мережі. Нижче наведено перелік найбільш поширених типів міжмережних екранів:

- Фаєрвол мережного рівня (Network Layer Firewall) – фільтрування на основі IP-адрес джерела та отримувача.
- Міжмережний екран транспортного рівня (Transport Layer Firewall) – фільтрування на основі портів відправника й отримувача, та фільтрування на основі станів з’єднання.
- Міжмережний екран прикладного рівня (Application Layer Firewall) – фільтрування на основі типу застосунку, програми або сервісу.
- Міжмережний екран на основі контексту (Context Aware Application Firewall) – фільтрування на основі користувача, пристрою, ролі, типу застосунку та профілю загроз.
- Проксі-сервер (Proxy Server) – фільтрування вмісту web-запитів, таких як URL-адреса, домен, соцмережі тощо.
- Зворотній проксі-сервер (Reverse Proxy Server) – розміщується перед веб-серверами і захищає, приховує, розвантажує та розподіляє доступ до веб-серверів.
- NAT-міжмережний екран (Network Address Translation (NAT) Firewall) – приховує або маскує приватні адреси хостів у мережі.
- Фаєрвол на базі хоста (Host-based Firewall) – фільтрація портів та викликів системних служб на одному комп'ютері.

5.3 Основні схеми захисту на базі міжмережевих екранів

Існують наступні основні схеми захисту на базі міжмережевих екранів:

- міжмережевий екран - фільтруючий маршрутизатор;
- міжмережевий екран на основі двопортового шлюзу;
- міжмережевий екран на основі екранованого шлюзу;

- міжмережевий екран - екранована підмережа.

Міжмережевий екран - фільтруючий маршрутизатор - найбільш простий в реалізації - виконує фільтрацію вхідних і витікаючих пакетів на основі аналізу їх адрес і портів (рис. 5.5).

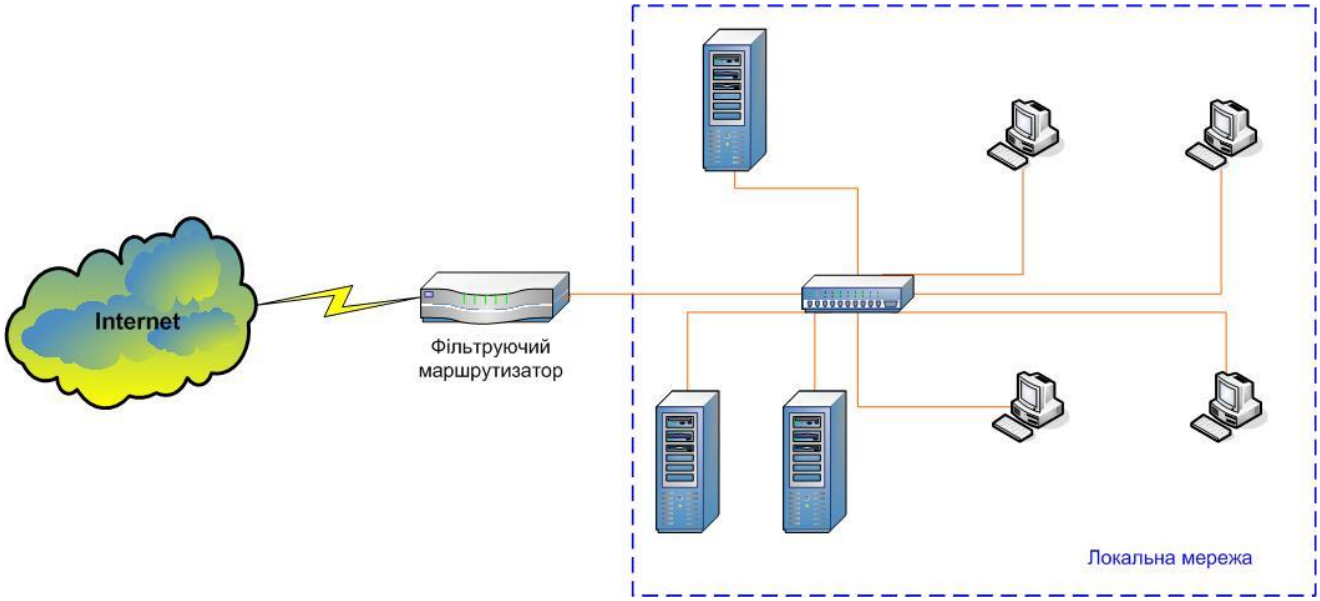


Рисунок 5.5 – Міжмережевий екран на основі фільтруючого маршрутизатора

Міжмережевий екран на основі двопортового шлюзу прикладного рівня часто є комп'ютером з двома мережевими інтерфейсами, при передачі даних між якими здійснюється фільтрація (рис. 5.6). Для забезпечення додаткового захисту між шлюзом прикладного рівня і Internet звичайно розміщують фільтруючий маршрутизатор. Між прикладним шлюзом і фільтруючим маршрутизатором утворюється внутрішня екранована підмережа - демілітаризована зона, в якій можливо розташовувати доступні ззовні інформаційні сервери.

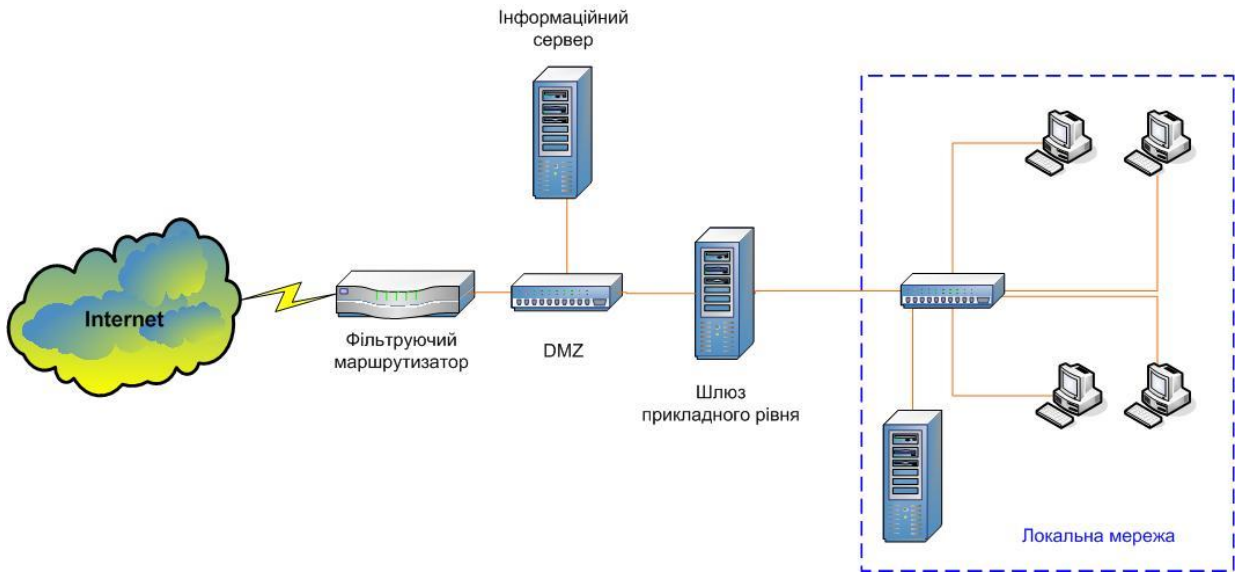


Рисунок 5.6 – Міжмережевий екран з шлюзом прикладного рівня і фільтруючим маршрутизатором

Міжмережевий екран на основі екранованого шлюзу прикладного рівня, реалізованого на комп'ютері лише з одним мережевим інтерфейсом, є гнучкіший варіант реалізації міжмережевого екрану, оскільки є можливість реалізувати з'єднання з Internet для деяких служб через проксі - сервер, а для деяких через фільтруючий маршрутизатор (рис. 5.7).

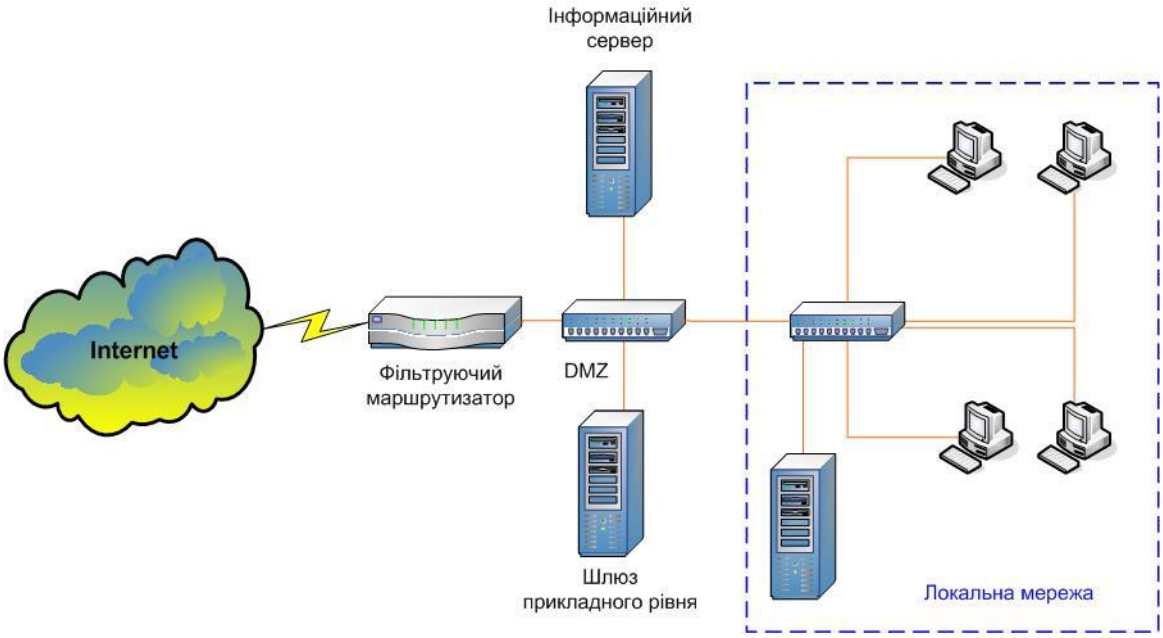


Рисунок 5.7 – Міжмережевий екран з екранованим шлюзом прикладного рівня і фільтруючим маршрутизатором

Міжмережевий екран, що складається з екранованою зовнішнім і внутрішнім фільтруючими маршрутизаторами підмережею показаний на рис. 5.8.

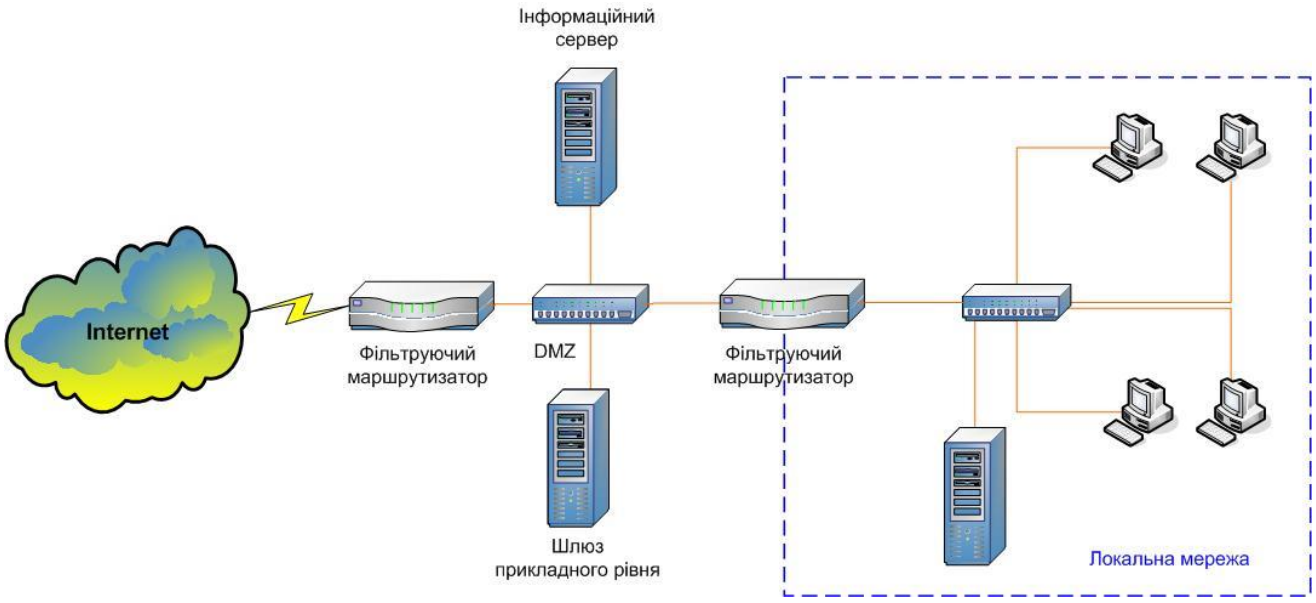


Рисунок 5.8 – Міжмережевий екран з екранованою фільтруючими маршрутизаторами підмережею

У цій підмережі розташовується шлюз прикладного рівня і сервери. Зовнішній маршрутизатор дозволяє трафік зовнішніх користувачів лише до проксі - серверу і серверам в демілітаризованій зоні. Внутрішній маршрутизатор захищає внутрішню мережу як від Internet, так і від екранованої підмережі (у разі її компрометації).

6 ЗАСОБИ ПЕРЕВІРКИ ЦІЛІСНОСТІ ІНФОРМАЦІЇ

6.1 Методи виявлення помилок

При передачі даних по каналах зв'язку існує ймовірність появи помилок. Це зв'язано, перш за все, з тим, що канал передачі інформації звичайно піддається дії різних спотворень, викликаних шумом, який може мати як природну дію, так і створюватися використовуваним обладнанням. Не можливо також виключати умисне спотворення даних при несанкціонованому доступі до ліній передачі і передавального обладнання. Процес захисту від помилок включає виявлення і виправлення помилок.

Існує два основні підходи до виправлення помилок:

- перший базується на використанні різних методів локалізації і виправлення помилок в приймаючому пристрої (реалізується достатньо рідко);
- другий використовує методи визначення лише факту появи помилок, "виправлення" яких здійснюється за рахунок повторної передачі повідомлення (реалізується часто).

Існують різні методи виявлення помилок, більшість з яких перетворить повідомлення, доповнюючи його надмірною інформацією (методи контрольних сум). Методи CRC (Cyclic Redundancy Codes) відносяться до методів, які залишають без зміни початковий текст повідомлення, додаючи до повідомлення контрольну суму.

Метод контролю парності полягає в тому, що для щодо невеликого блоку біт даних обчислюється контрольна сума, яка передається і зберігається разом з даними у вигляді біта парності. При прочитуванні даних контрольна сума обчислюється ще раз і порівнюється з бітом парності. Якщо вони співпадають, то дані вважаються істинними, інакше генерується повідомлення про помилку парності, яке приводить до зупинки комп'ютера.

Використовують два основні підходи для реалізації цього методу:

- коли встановлений контроль по парності (Even), то біт парності приймає таке значення, щоб разом з ним кількість двійкових одиниць «1» (Mark) в блоці даних, що захищається, була парною.
- коли встановлений контроль по непарності (Odd), то біт парності приймає таке значення, щоб разом з ним кількість двійкових одиниць «1» (Mark) в блоці даних, що захищається, була непарною.

Цей метод використовується тільки для щодо невеликих блоків інформації, наприклад, байтів, що зберігаються в оперативній пам'яті комп'ютерів, оскільки має наступний недолік, ймовірність якого зростає із зростанням розміру блоку інформації, що захищається: метод ніяк не застрахований від подвійних помилок (одночасна зміна значень двох біт не впливає на біт парності). Також до втрати інформації може привести помилка в біті парності.

Найбільш простим способом перевірки цілісності даних, передаваних в цифровому представленні, є метод контрольних сум. Під контрольною сумою розуміється деяке значення, розраховане шляхом складання всіх чисел з вхідних даних. Оскільки часто сума всіх чисел перевищує максимально допустиме значення, наперед задане для цієї величини (розміром поля контрольної суми), то контрольну суму звичайно розраховують як залишок від ділення підсумкової

суми на максимально можливе значення контрольної суми, збільшене на одиницю (такий алгоритм часто називають доповненням контрольної суми до одиниці):

$$Checksum = Total /(MaxVal + 1),$$

де Total - підсумкова сума, розрахована за вхідними даними; MaxVal - максимально допустиме значення контрольної суми, задане наперед.

Наприклад, нехай блок інформації, що захищається, є наступною послідовністю величин завдовжки 10 байт: 36 211 163 4 109 192 58 247 47 92 і розмір поля, відведеного для контрольної суми, складає один байт (тобто максимальне число, яке можливо в нього занести, рівне 255). Сума даних блоку рівна 1159, а залишок від ділення суми на 255+1=256 (контрольна сума) дорівнює 135. Якщо контрольна сума, розрахована відправником інформації, дорівнювала, скажімо, 246, а після отримання вона має значення 135, означає, інформація піддалася зміні.

Недолік методу контрольних сум полягає в тому, що неспівпадіння значень цих сум служить вірним доказом, що такий документ піддався зміні, рівність порівнюваних значень ще не дає гарантії, що інформація залишилася незмінною. Можливо довільним чином змінити порядок чергування чисел в документі, а контрольна сума при цьому буде зберігати вірне значення. Крім того, можливо змінити окремі числа в документі і підігнати інші так, щоб забезпечити вірне значення контрольної суми. При використанні для контрольних сум восьмирозрядної змінної ймовірність того, що контрольні суми двох абсолютно випадково обраних послідовностей даних будуть однакові, дорівнює 1/256. При збільшенні довжини змінної під контрольну суму до 16 або 32 розрядів, ймовірність збігів зменшується, проте цей механізм все одно дуже чутливий до можливих помилок, щоб забезпечити високий ступінь довіри до представлених даних.

6.2 Метод обчислення циклічного надмірного коду

Більш досконалий спосіб цифрової ідентифікації блоків даних - це обчислення її циклічного надмірного коду (Cyclic Redundancy Code - CRC). Цей метод широко використовується при захисті інформації в комп'ютерних мережах. Наприклад, останнє поле в кадрі Ethernet - це чотирьохбайтове поле контрольної послідовності кадру (Frame Check Sequence, FCS). Значення цього поля обчислюється на основі вмісту заголовка і даних (разом із заповнювачем, але без урахування преамбули і обмежувача) за допомогою 32-розрядного циклічного надмірного коду (CRC-32) за наступною формулою (у двійковій системі числення):

$$контрольна\ послідовність = MOD(дані/поліном),$$

де MOD - операція розрахунку залишку від ділення; дані - бітова послідовність блоку даних, що захищається, поліном - так званий утворюючий поліном, відомий і відправнику, і одержувачу.

Утворюючим поліномом називається мінімальний поліном з сімейства поліноміальний простих поліномів (тобто що не діляться поліноміальний без залишку на будь-яке ціле число, окрім 2 і самого себе). Такий поліном має властивості лінійності (сума по модулю 2 (XOR) будь-яких двох поліномів сімейства призводить до отримання полінома з цього ж сімейства) і циклічності (циклічний зсув будь-якого полінома сімейства на будь-яку кількість розрядів призводить до здобуття полінома цього ж сімейства). Утворюючий поліном володіє властивістю «розмазувати» зміну в деякому масиву біт даних на можливо більшу кількість біт контрольної суми і тим самим підвищувати ймовірність детектування помилки. Зворотність операції XOR дозволяє використовувати один і той же алгоритм при обчисленні суми і відправником, і одержувачем, крім того, відносно просто алгоритм реалізовується електронними схемами.

Арифметичне ділення дуже схоже на схему розрахунку контрольної суми, званої CRC, сам же алгоритм CRC декілька складніший. Всі CRC алгоритми засновані на поліноміальних обчисленнях, і для будь-якого алгоритму CRC можливо вказати, який поліном він використовує. Замість представлення дільника, ділимого (повідомлення), приватного і залишку у вигляді позитивних цілих чисел, можливо представити їх у вигляді поліномів з двійковими коефіцієнтами або у вигляді рядка біт, кожний з яких є коефіцієнтом полінома. Наприклад, десяткове число 23 в шістнадцятиричній системі числення має вигляд 17, а в двійковому – 10111, що співпадає з поліномом:

$$\mathbf{1x^4 + 0x^3 + 1x^2 + 1x^1 + 1x^0}$$

або, спрощено:

$$\mathbf{x^4 + x^2 + x^1 + x^0}$$

І повідомлення, і дільник можуть бути представлені у вигляді поліномів, з якими можливо виконувати будь-які арифметичні дії. Припустимо, що необхідно перемножити, наприклад, 1101 і 1011. Це можливо виконати, як множення поліномів:

$$\mathbf{(x^3 + x^2 + x^0)(x^3 + x^1 + x^0) = (x^6 + x^4 + x^3 + x^5 + x^3 + x^2 + x^3 + x^1 + x^0) =}$$

$$\mathbf{= x^6 + x^5 + x^4 + 3x3 + x^2 + x^1 + x^0}$$

Тепер для отримання правильної відповіді необхідно вказати, що X рівний 2, і виконати перенесення біта від члена $3x^3$. В результаті одержимо:

$$\mathbf{x^7 + x^3 + x^2 + x^1 + x^0}$$

Т. ч., якщо вважати, що «X» не відомо, то неможливо виконати перенесення, оскільки не відомо, що $3x^3$ – це те ж саме, що і $x^4 + x^3$, оскільки не

відомо, що X = 2. У поліноміальній арифметиці зв’язки між коефіцієнтами невизначені, і, тому, коефіцієнти при кожному члені полінома стають такими, що строго типізуються — коефіцієнт при x^2 має інший тип, ніж при x^3. Якщо коефіцієнти кожного члена полінома цілком ізольовані один від одного, то можливо використовувати будь-які види поліноміальної арифметики, міняючи правила, за якими коефіцієнти використовуються. Одна з таких схем застосується в теорії CRC полягає у наступному: коефіцієнти складаються за модулем 2 без перенесення – тобто коефіцієнти можуть мати значення лише 0 або 1, перенесення не враховується. Це називається «поліноміальна арифметика за модулем 2». Щодо попереднього прикладу:

$$(x^3 + x^2 + x^0)(x^3 + x^1 + x^0) = (x^6 + x^4 + x^3 + x^5 + x^3 + x^2 + x^3 + x^1 + x^0) =$$

$$= x^6 + x^5 + x^4 + 3x3 + x^2 + x^1 + x^0$$

За правилами звичайної арифметики, коефіцієнт члена $3x^3$ розподіляється по інших членах полінома, використовуючи механізм перенесення і припускаючи, що X = 2. У «поліноміальній арифметиці за модулем 2» не відомо, чому дорівнює «X», перенесень при цьому не існує, а всі коефіцієнти розраховуються за модулем 2. В результаті одержуємо:

$$= x^6 + x^5 + x^4 + x^3 + x^2 + x^1 + x^0$$

Існує схожість між поліноміальною арифметикою і арифметикою високої точності (multiple precision arithmetic), коли основа b замінюється на x. Основна відмінність полягає в тому, що коефіцієнт u_k члена x_k в поліноміальній арифметиці вважається малою величиною, або величиною, не пов'язаною з найближчими коефіцієнтами $x_{k\ 1}$ (і x_{k+1}), тому перенесення від одного члена до іншого в ній відсутній. Фактично, поліноміальна арифметика за модулем b багато в чому аналогічна арифметиці високої точності по основі b за винятком виключення всіх перенесень. Т. ч., поліноміальна арифметика за модулем 2 – це фактично двійкова арифметика за модулем 2 без урахування перенесень. Хоча поліноми мають зручні математичні засоби для аналізу CRC алгоритмів і алгоритмів корекції помилок, для спрощення вони будуть замінені на безпосередні арифметичні дії в системі, з якою вони ізоморфні, а саме в системі двійкової арифметики без перенесень.

Дії, що виконуються під час обчислення CRC, є арифметичними операціями без урахування перенесень. Це називається поліноміальною арифметикою. Складання двох чисел в CRC арифметиці повністю аналогічно звичайній арифметичній дії за винятком відсутності перенесень з розряду в розряд. Це означає, що кожна пара бітів визначає результат свого розряду незалежно від результатів інших пар. Наприклад:

```
 10011011
+11001010
 --------
 01010001
```

Для кожної пари бітів можливі 4 варіанти:

```
0+0=0
0+1=1
1+0=1
1+1=0
```

Те ж саме справедливо і для віднімання:

```
 10011011
-11001010
 --------
 01010001
```

коли є також 4 можливих комбінації:

```
0-0=0
0-1=1
1-0=1
1-1=0
```

Фактично, як операція складання, так і операція віднімання в CRC арифметиці ідентичні операції «Виключаюче АБО» (eXclusive OR – XOR), що дозволяє замінити 2 операції першого рівня (складання і віднімання) однією дією, яка, одночасно, виявляється інверсною самому собі. Згрупувавши складання і віднімання в одну єдину дію, CRC арифметика виключає з поля своєї уваги всі величини, що знаходяться за межами самого старшого свого біта. Хоча ясно, що значення 1010 більше, ніж 10, це виявляється не так, коли говорять, що 1010 повинне бути більше, ніж 1001. Спробуємо одержати з 1010 значення 1001, віднявши або додавши до нього одну і ту ж величину:

```
1001 = 1010 + 0011
1001 = 1010 - 0011
```

Ця властивість виключає уявлення про порядок.

Множення, як і в звичайній арифметиці, вважається сумою значень першого співмножника, зсунутих відповідно до значення другого співмножника.

```
   1101
 x 1011
   ----
   1101
  1101.
 0000..
1101...
-------
1111111
```

Ділення дещо складніше, так як потрібно знати, коли «одне число перетворюється на інше». Щоб зробити це, потрібно буде ввести слабке поняття розмірності: число X більше або рівне Y, якщо позиція самого старшого одиничного біта числа X більш значуща (або відповідає) позиції самого старшого одиничного біта числа Y. Нижче приведений приклад ділення :

```
                1100001010
              ______________
10011 )  11010110110000
         10011,,.,,,,....
         -----,,.,,,,....
          10011,.,,,....
          10011,.,,,....
          -----,.,,....
           00001.,,....
           00000.,,....
           -----.,,....
            00010,,....
            00000,,....
            -----,,....
             00101,....
             00000,....
             -----,....
              01011....
              00000....
              -----....
               10110...
               10011...
               -----...
                01010..
                00000..
                -----..
                 10100.
                 10011.
                 -----.
                  01110
                  00000
                  -----
                   1110
```

Якщо дії складання і віднімання в CRC - арифметиці – це одне і те ж, то у такому разі A+0=A і A-0=A. Якщо число A одержане множенням числа B, то в CRC арифметиці це означає, що існує можливість сконструювати число A з нуля, застосовуючи операцію XOR до B, зсунутому на різну кількість позицій. Наприклад, якщо A рівне 0111010110, а B – 11, то ми може сконструювати A з B наступним способом:

```
   0111010110
= .......11.
+ ....11....
+ ...11.....
  .11.......
```

Проте, якби A було б рівне 0111010111, то не вдалося б скласти його за допомогою різних зсувів числа 11. Тому в CRC арифметиці воно не ділиться на B. Таким чином, CRC арифметика зводиться головним чином до операції що «Виключаюче АБО» деякого значення при різних величинах зсуву.

Передане повідомлення T є добутком полінома. Необхідно враховувати, що:

- останні W біт повідомлення – це залишок від ділення доповненого нулями початкового повідомлення на обраний поліном;
- складання рівносильне відніманню, тому збільшення залишку доповнює значення повідомлення до наступного повного добутку.

Якщо повідомлення при передачі було пошкоджено, то одержують повідомлення $T \oplus E$, $T \oplus E$, де E – це вектор помилки, а $\oplus$ – це CRC складання (або операція XOR). Одержавши повідомлення, приймач ділить $T \oplus E$ $T \oplus E$ на G. Оскільки T mod G = 0, (TE) mod G = E mod G. Отже, якість полінома визначатиметься набором добутків G, оскільки у разі, коли E також є твором G, така помилка виявлена не буде. Отже, завдання полягає в тому, щоб знайти такі класи G, добутки яких будуть якомога менше схожі на шуми в каналі передачі (які і викликають пошкодження повідомлення). Розглянемо, які типи шумів в каналі передачі можуть діяти у каналах передавання даних.

Однобітові помилки - помилки такого роду означають, що E=1000...0000. Можливо гарантувати, що помилки цього класу завжди буде розпізнані за умови, що в G принаймні 2 біти встановлені в «1». Будь-який добуток G може бути сконструйований операціями зсуву і складання, і, в теж час, неможливо набути значення з 1 одиничним бітом, зсовуючи і складаючи величину, що має більше 1 одиничного біта, оскільки в результаті завжди присутні, принаймні, 2 біта.

Двобітові помилки - для виявлення будь-яких помилок вигляду 100...000100...000 (тобто коли E містить принаймні 2 одиничних біта) необхідно обрати таке G, яке б не мало множників 11, 101, 1001, 10001, і так далі.

Помилки з непарною кількістю біт - можливо визначити будь-які пошкодження, коли E має непарну кількість біт, обравши поліном G таким, щоб він мав парну кількість біт. Це визначається наступними властивостями:

- CRC множення є простою операцією XOR постійного регістрового значення з різними зсувами;
- XOR - це операція перемикання бітів;
- якщо застосовується в регістрі операцію XOR до величини з парною кількістю біт, парність кількості одиничних біт в регістрі залишиться незмінною.

Наприклад, почнемо з E=111 і спробуємо скинути всі 3 біта в «0» послідовним виконанням операції XOR з величиною 11 і одним з 2 варіантів зсувів (тобто, «E=E XOR 011» і «E=E XOR 110»). Це аналогічно завданню про перевертання стаканів, коли за одну дію можливо перевернути одночасно будь-які два стакани. Більшість розповсюджених CRC поліномів містять парну кількість одиничних біт.

Пакетна помилка виглядає як E=000...000111...11110000...00, тобто E складається з нулів за винятком групи одиниць десь у середині. Цю величину можливо перетворити в E=(10000...00)(1111111...111), де є z - нулів в лівій частині і n - одиниць в правій. Для виявлення цих помилок необхідно встановити молодший біт G в 1. При цьому необхідно, щоб ліва частина не була множником

G. Але завжди, поки G ширше за праву частину, помилка завжди буде розпізнана. Ймовірність пакетної помилки з шириною, більшою, ніж W, дорівнює (0.5) W.

У Ethernet як утворюючий поліномом (CRC-32) використовується багаточлен:

$$x^{32}+x^{26}+x^{23}+x^{22}+x^{l6}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+l,$$

що визначає бітову послідовність 10000010011000001000111010110111. Цей код дозволяє виявити ~ 99,9999% всіх помилок в повідомленнях завдовжки до 64 байт. Таким чином, ймовірність того, що приймаюча станція сприйме зіпсований кадр як цілий, практично дорівнює нулю. Алгоритм контролю CRC вже протягом тривалого часу широко використовується в системах мережевих адаптерів, контролерів жорстких дисків і інших пристроїв для перевірки ідентичності вхідної і вихідної інформації. Також цей механізм застосовується в багатьох комунікаційних програмах і архіваторах.

Так само, як і для контрольних сум, CRC коду не потрібно багато місця (звичайно їх довжина складає 16 або 32 розрядів); проте в порівнянні з ними, надійність виявлення невеликих змін вхідної інформації буде значно вище. Якщо в деякому величезному блоці даних лише один розряд став іншим, то і контрольний параметр CRC з 100-процентною ймовірністю також матиме інше значення. Якщо ж зміняться два розряди, то ймовірність виявлення помилки при довжині параметра CRC в 16-розрядів, складає більше 99,99%. На відміну від контрольних сум метод CRC може розпізнати перестановку двох байт, додавання 1 до одного байт і відніманню 1 з іншого і інші методи фальсифікації.

Таким чином, послідовність кодування CRC є наступною:

1. Початкова бітова послідовність даних зсовується вліво на m розрядів, де m - найбільший ступінь утворюючого полінома (для CRC-32 рівна 32 біта).
2. виконується побітове ділення (шляхом застосування операції XOR) одержаної зсунутої інформаційної послідовності на утворюючий поліном g(x) до неподільного залишку (довжина залишку не повинна перевищувати m біт);
3. зсунута послідовність об'єднується із залишком, в результаті чого утворюється інформаційна послідовність з CRC кодом;
4. перевірка цієї послідовності одержувачем здійснюється діленням інформаційної послідовності з CRC кодом на той же утворюючий поліном; у випадку, якщо залишок від ділення дорівнює 0, вважається, що помилок в одержаних послідовності і CRC коді немає, якщо ж залишок не дорівнює нулю, отже, спотворені або інформаційна послідовність, або CRC код і повинна бути виконана повторна передача.

6.3 Приклад розрахунку CRC коду

1. Нехай початкова послідовність 10110101, а утворюючий поліном оберемо рівним 1011 (x^3+x+1).
2. Виконуємо зсув послідовності на максимальний ступінь утворюючого полінома - 3 біта: 10110101000.
3. Виконуємо ділення (побітове XOR) на утворюючий поліном:

```
10110101000 | 1011
1011        | 100001
000001010
    1011
    000100
```

4. Залишком (кодом CRC-3) є 100. Об'єднуємо початкову послідовність і CRC-3 код. Одержуємо інформаційну послідовність з кодом, готову до передачі: 10110101100.

5. Одержувач, одержавши послідовність, виконує її ділення (побітове XOR) на той же утворюючий поліном:

```
10110101100 | 1011
1011        | 100001
000001011
    1011
    000000
```

Оскільки залишок від ділення дорівнює нулю, можливо зробити висновок про те, що при передачі інформація не піддалася зміні.

6.4 Алгоритми програмної реалізації CRC

6.4.1 Пряма реалізація CRC

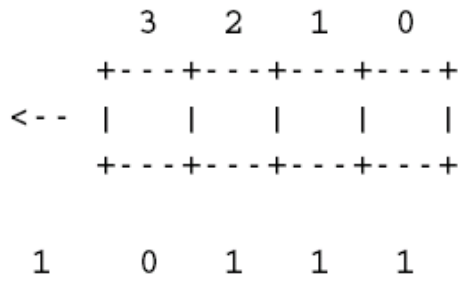
Розглянемо абсолютно найпростішу і малоефективну реалізацію алгоритму розрахунку CRC. Потім поступово перетворимо програму до тих пір, поки не одержимо компактний табличний код.

Спочатку необхідно створити програму, що виконує CRC ділення. При цьому існують 2 причини, за якими неможливо використовувати звичайні машинні інструкції ділення:

1) необхідно виконувати операцію за правилами CRC арифметики;
2) ділиме може мати розмір в десятки мегабайтів, а сучасні процесори не мають таких регістрів.

Т. ч., при виконанні операції CRC ділення необхідне повідомлення по частинах записувати до регістру. На цьому етапі розробки потрібно поводитися з даними абсолютно точно. У всіх подальших прикладах у якості повідомлення використовується потік байт по 8 біт кожен, з яких сьомий біт вважатиметься самим значущим бітом. Потік біт, одержаний з цих байт, передаватиметься, починаючи з 7 біта, і закінчуючи 0 бітом першого байту, потім другого і так далі.

Для прикладу розглянемо поліном 4-го ступеню (W=4) - 10111. Для виконання ділення буде потрібний 4-х бітовий регістр



Ділення виконується наступним способом:

Завантажимо регістр нульовими бітами

Доповнимо хвостову частину повідомлення W нульовими бітами

While (поки що є необроблені біти)

Begin

Зсунемо регістр на 1 біт вліво і помістимо черговий ще не оброблений біт з повідомлення в 0 позицію регістра.

If (з регістра був висунутий біт із значенням "1")

Регістр = Регістр XOR Поліном

End

Тепер в регістрі міститься залишок

На практиці умова *IF* може бути протестоване за змістом старшого біта регістру до виконання операції зсуву.)

Фактично такий алгоритм можливо представити операціями віднімання полінома в різних ступенях (що досягалося операціями зсуву) з початкового повідомлення до тих пір, поки в результаті не залишилося нічого, окрім залишку.

6.4.2 Реалізація табличного алгоритму

Розглянутий алгоритм є безпосередньо пов'язаним з теорією і тому він дійсно простий. Проте, унаслідок того, що він працює на рівні біт, його достатньо важко закодувати, а ефективність його роботи низька, весь цикл виконується за кожним бітом. Для збільшення швидкості роботи необхідно щоб алгоритм обробляв блоки, більші ніж 1 біт. Такими блоками можуть бути півбайти (4 біт), байти, (8 біт), слова (16 біт) і довгі слова (32 біт), і навіть довші фрагменти. Півбайти не розглядаються, оскільки вони не вирівняні за границею байта. Для збільшення швидкості роботи програми необхідно, щоб інформаційні блоки були вирівняні за границею байта, і, фактично, більшість табличних алгоритмів оперують за один цикл саме з байтом. Для простоти будемо використовувати

замість 4-х бітового полінома 32-х бітовий. Регістр виглядатиме точно також, як і раніше, з тією лише різницею, що осередки тепер представляють не біти, а байти, а поліном складатиметься з 33-х біт (один окремий біт і 32 «активних» біта) .

Спочатку використовуємо «простий» алгоритм. Припустимо, що алгоритм повністю перероблено на роботу з байтами, а старші 8 біт – 32-х бітові регістри (3 байт) мають значення:

$$t7\ \ t6\ \ t5\ \ t4\ \ t3\ \ t2\ \ t1\ \ t0$$

Під час чергового циклу «простого» алгоритму значення «t7» визначає, чи буде виконана операція «Виключаюче АБО» (XOR) полінома зі всім вмістом регістра. Якщо t7 = 1, то операція виконується, інакше вона буде пропущена. Припустимо також, що старші 8 біт полінома мають значення g7 g6.g0, тоді до наступного циклу старший байт матиме наступний вміст:

$$\begin{array}{r} t6\ \ t5\ \ t4\ \ t3\ \ t2\ \ t1\ \ t0\ \ ?? \\ \oplus \\ t7 * (g7\ \ g6\ \ g5\ \ g4\ \ g3\ \ g2\ \ g1\ \ g0) \end{array}$$

«Новий» старший біт (який визначатиме, що відбудеться в наступному циклі) тепер має значення $t6 \oplus t7 \bullet g7$. З інформаційної точки зору все, що потрібно для обчислення значення «нового» старшого біта, міститься в двох старших бітах початкового старшого байта. Точно також, «наступний» старший біт може бути наперед обчислено, виходячи їз трьох старших бітів t7, t6 і t5. У загальному випадку величину старшого біта регістра до циклу можливо обчислити з старших бітів регістра. Припустимо, що старші 8 біт регістра використовуються для розрахунку значення старшого біта регістра на восьмому циклі. Припустимо, що наступні вісім циклів виконуються, використовуючи передобчислені значення (які можливо записати до однобайтового регістру і висувати з нього послідовно по одному біту). Тоді можливо відмітити, що:

1) величина старшого байта регістра тепер не матиме ніякого значення - не матиме значення, скільки разів і з яким зсувом полінома буде виконана операція XOR, оскільки всі біти так чи інакше будуть висунуті з правої частини регістра за подальші вісім циклів;

2) біти регістру, що залишилися в ньому, опиняться зрушені вліво на одну позицію, а найправіший байт регістру буде зсунутий до позиції наступного найлівійшого байту;

3) т. ч., регістр проходить серію операцій XOR відповідно до значень бітів наперед обчисленого керуючого байту.

Розглянемо вплив на регістр операцій XOR з постійною величиною при різних зсувах. Наприклад:

```
0100010
...0110
..0110.
0110...
-------
0011000
```

.

З цього прикладу можливо зробити наступний висновок: при виконанні операції XOR регістра з постійною величиною при різних її зсувах завжди існуватиме деяке значення, яке при застосуванні операції “Виключаюче АБО» з початковою величиною регістра дасть той же самий результат.

Т. ч., одержуємо:

While (повідомлення повністю не оброблене)

Begin

Перевіримо старший байт регістра

Розрахуємо на його основі контрольний байт

Ґрунтуючись на значенні контрольного байту, виконаємо операцію XOR полінома з різними зсувами

Зсунемо регістр на один байт вліво, додавши справа новий байт з повідомлення

Виконаємо операцію XOR вмісту регістра з підсумовуваним поліномом

End

Отриманий результат не буде кращим, ніж для «простого» алгоритма. Але у ньому більшу частину обчислень можливо виконати наперед, а результати зібрати в таблицю. Тоді одержимо:

While (повідомлення повністю не оброблене)

Begin

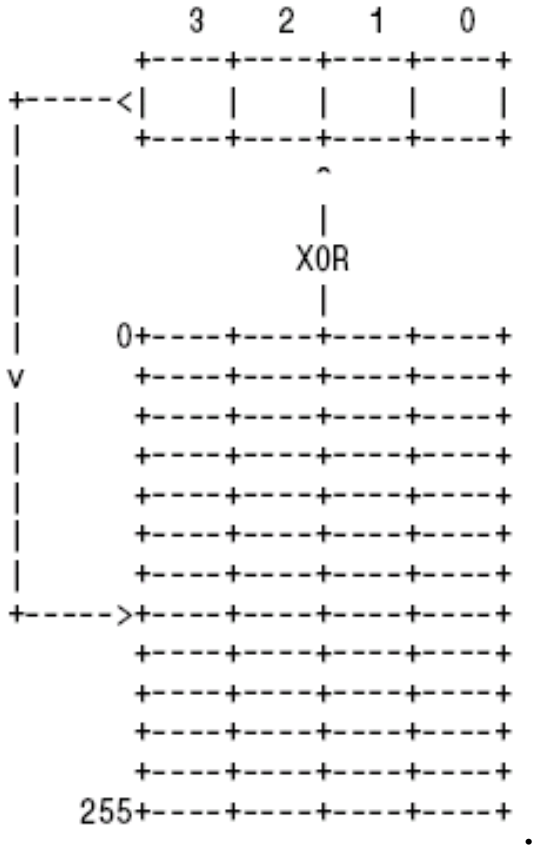
Top = Старший_байт(Register);

Register = (Register << 24) | Новий_байт_ повідомлення;

Register = Register XOR Розрахована_таблиця[Top];

End

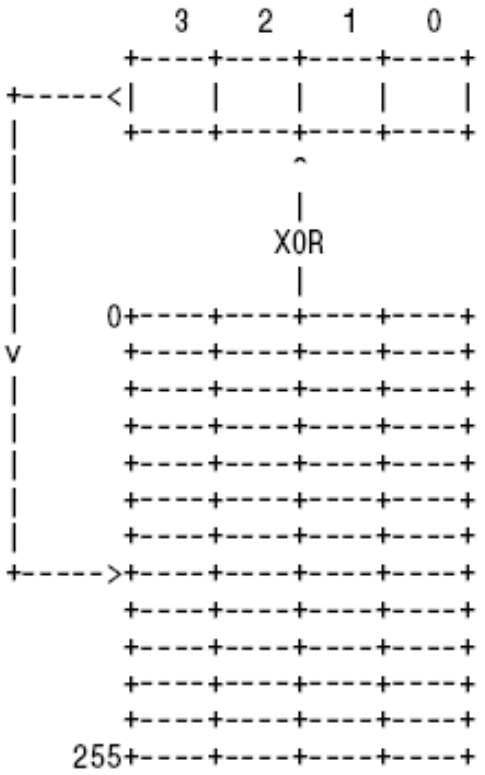
Це дуже ефективний алгоритм, що вимагає тільки операцій зсуву, OR, XOR, а також операції пошуку в таблиці. Графічно він виглядає як:



Також він характеризується чітким і ефективним циклом.

6.4.3 Перетворений табличний алгоритм

Розглянутий вище цикл працює з вирівняним повідомленням, тобто для його використання необхідно доповнити повідомлення W/8 нульовими байтами перш, ніж передавати алгоритму. У різних випадках це може і не скласти великої проблеми, проте, якщо оброблюваний блок даних передається іншою програмою, таке доповнення може виявитися проблематичним. Одним з варіантів є просте додавання нульових байтів після основного циклу коду. Тепер можливо перетворити все так, щоб уникнути і необхідності доповнення повідомлення нульовими байтами, і обробки кожного додаткового нульового байта окремо. Для пояснення цієї можливості повернемося до вже розглянутої вище діаграми:



У повідомленні, що оброблюється хвіст складається з W/8 додаткових нульових байт, які з'являються на останньому етапі роботи алгоритму, щоб бути вставленими до регістру при завершенні обробки. Проте, вони (нулі) не надають будь - якого впливу на регістр, оскільки :

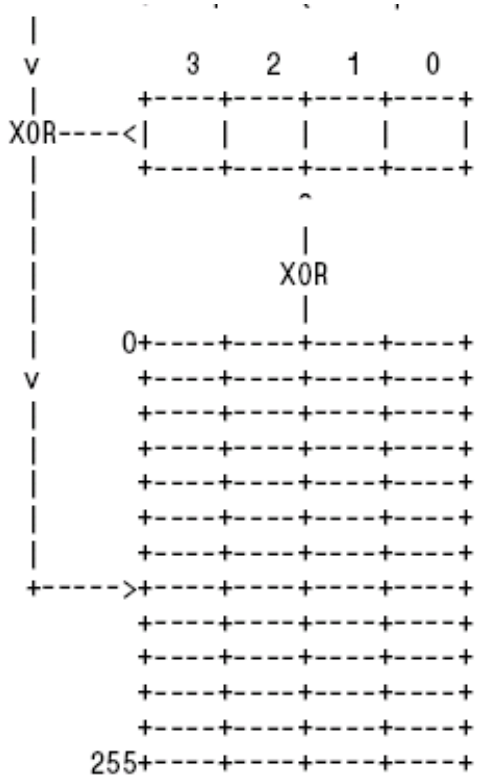
1) операція XOR з нульовим вмістом не змінює значення другого операнда;

2) ці чотири нульових байти ніколи не висуваються з лівої частини регістра, де їх вміст міг би завдати будь - якого впливу.

Отже, єдине призначення цих нульових байтів полягає в тому, щоб продовжити роботу алгоритму W/8 циклів і дозволити кінцевим байтам оброблюваного повідомлення пройти через весь регістр.

Якщо початковий вміст регістра дорівнює нулю, то початкові чотири цикли всього лише завантажать перші чотири байти повідомлення до регістру, зсуваючи їх справа наліво. Це відбувається тому, що перші 32 контрольних біта будуть дорівнювати нулю, отже в результаті операції XOR вміст регістра змінюватись не буде. У разі ж, коли в регістрі спочатку міститься ненульове значення, в процесі перших 4 циклів також відбуватиметься заповнення регістра 4 байтами повідомлення з подальшою операцією «Виключаюче АБО» з деяким постійним значенням, яке цілком залежить від початкового вмісту регістра.

Це разом з асоціативною властивістю операції «Виключаюче АБО» (A xor B) xor C = A xor (B xor C) означає, що байтам оброблюваного повідомлення немає ніякої необхідності проходити через W/8 байтів регістра. Натомість вони можуть бути безпосередньо скомбіновані операцією XOR із старшим байтом регістра з використанням результату як табличний індекс. Т. ч., одержимо наступну модифікацію алгоритму:



Початкове значення регістра в цьому випадку повинне бути тим же самим, що і початкове значення регістра попереднього алгоритму, що пройшло 4 рази через таблицю. Таблиця ж така, що якщо попередній алгоритм використовував «0», то і новий алгоритм також використовуватиме це ж значення.

Ці алгоритми ідентичні, і вони дають ідентичні результати.

ПЕРЕЛІК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Гапак О. М., Балога С. І. Захист інформації в комп’ютерних системах : підручник. Ужгород : «АУТДОР-ШАРК», 2021. 184 с.
2. Бурячок В. Л. та інш. Технології забезпечення безпеки мережевої інфраструктури : підручник. К. : КУБГ, 2019. 225 с.
3. Гулак Г. М. Методологія захисту інформації. Аспекти кібербезпеки : підручник. К. : Видавництво НА СБ України, 2020. – 256 с.
4. Даник Ю. Г., Воробієнко П. П., Чернега В.М. Основи кібербезпеки та кібероборони : підручник. Одеса : ОНАЗ ім. О.С. Попова, 2019. 320 с.
5. Кісельов Є. М. Засоби захисту інформації у мережах ЕОМ : методичні вказівки до лабораторних робіт. Запоріжжя : ЗДІА, 2008. 95 с.
6. Швець Є. Я., Кісельов Є. М. Засоби захисту інформації у мережах ЕОМ : методичні вказівки до курсового проектування. Запоріжжя : ЗДІА, 2005. 31 с.
7. Грайворонський М. В., Новіков О. М. Безпека інформаційно-комунікаційних систем : підручник для внз. Київ : BHV, 2009. 607 с.
8. Манжай О. В., Манжай І. А. Правові засади захисту інформації : підручник. Харків : Панов, 2020. 162 с.
9. Бурячок В. Л. та інш. Інформаційна та кібербезпека : соціотехнічний аспект : підручник. К. : ДУТ, 2015. 288 с.
10. Тарнавський Ю. А. Технології захисту інформації : підручник. Київ : КПІ ім. Ігоря Сікорського, 2018. 162 с.
11. Жураковський Б. Ю., Зенів І. О. Компютерні мережі : частина 1 : навч. посібник. Київ : КПІ ім. Ігоря Сікорського, 2020. 336 с
12. Комп’ютерні мережі. Частина 1. Моделювання комп’ютерних мереж : лабораторний практикум. / Укладачі: О. С. Яценко, О. І. Яценко. Житомир : Вид-во ЖДУ ім. І. Франка, 2022. 76 с.
13. Єсін В. І., Рассомахін С. Г. Стислий словник основних термінів з безпеки інформаційних систем, технологій, кібербезпеки. Харків : ХНУ ім. В. Н. Каразіна, 2018. 63с.
14. Nadalin A. WASEC : Web Application Security for the everyday software engineer : Everything a web developer should know about application security : concise, condensed and made to last. Leanpub, 2020. 161 p.
15. Musa S. M. Network Security and Cryptography. Mercury Learning and Information, 2022. 832 p.
16. Stallings W. Cryptography and Network Security : Principles and Practice. London : Pearson Education Limited, 2019. 832 p.
17. Doherty J. Wireless and Mobile Device Security + Cloud Labs. Burlington : Jones & Bartlett Learning, 2021. 368 p.
18. Learn Tutorials on Network Security for Beginners in PDF : веб-сайт. URL: https://www.computer-pdf.com/tutorials-network-security.
19. Get Started Today In Introduction to Cybersecurity | Networking Academy : веб-сайт. URL: https://www.netacad.com/courses/cybersecurity/introduction-cybersecurity.
20. Learn To Protect Networks With Cybersecurity Essentials | Networking Academy : веб-сайт. URL: https://www.netacad.com/courses/cybersecurity/cybersecurity-essentials.

21. Cloud Security | Networking Academy : веб-сайт. URL: https://www.netacad.com/courses/cybersecurity/cloud-security.

22. Gain Connectivity Skills With The Networking Essentials Course | Networking Academy : веб-сайт. URL: https://www.netacad.com/courses/networking/networking-essentials.

23. Hands-on Labs for Security Education : веб-сайт. URL: https://seedsecuritylabs.org/.

24. Lecture Videos | Computer Systems Security | Electrical Engineering and Computer Science | MIT OpenCourseWare : веб-сайт. URL: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-858-computer-systems-security-fall-2014/video-lectures/.

25. Network and Computer Security | Electrical Engineering and Computer Science | MIT OpenCourseWare : веб-сайт. URL: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-857-network-and-computer-security-spring-2014/.