КРИПТОГРАФІЯ

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 – криптологію.

Необхідно відзначити, що в російськомовних текстах по даному предмету зустрічаються різні використання основних термінів, таких як "криптографія", "тайнопис" і деяких інших. Більш того, і по класифікації криптоалгоритмів можна зустріти різні думки.

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 здійснюють над повідомленням перетворення, відомі тільки їм двом. Стороннім особам невідомий сам алгоритм шифрування. Деякі фахівці вважають, що тайнопис не є криптографією взагалі.
2. Криптографія з ключем. Алгоритм дії над даними, що передаються, відомий всім стороннім особам, але він залежить від деякого параметра – "ключа", яким володіють тільки відправник і одержувач.
    1. Симетричні криптоалгоритми. Для шифрування і розшифровки повідомлення використовується один і той же блок інформації (ключ).
    2. Асиметричні криптоалгоритми. Алгоритм такий, що для шифрування повідомлення використовується один ("відкритий") ключ, відомий всім, а для розшифровки – інший ("закритий"), існуючий тільки у одержувача.

Весь подальший матеріал буде присвячений криптографії з ключем, оскільки більшість фахівців саме по відношенню до цих криптоалгоритмам використовує термін криптографія, що цілком виправдане. Так, наприклад, будь-який криптоалгоритм з ключем можна перетворити на тайнопис, просто "зашивши" в початковому коді програми деякий фіксований ключ. Зворотне ж перетворення практично неможливе.

Залежно від характеру дій, вироблюваних над даними, алгоритми підрозділяються на:

1. Перестановочні. Блоки інформації (байти, біти, крупніші одиниці) не змінюються самі по собі, але змінюється їх порядок проходження, що робить інформацію недоступною спостерігачу.
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 подання. 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 не будуть стискатися більш ніж на 2-4%).

Залежно від розміру блоку інформації криптоалгоритми поділяються на:

1. Потокові шифри. Одиницею кодування є один біт. Результат кодування не залежить від того, що пройшло раніше вхідного потоку. 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 представниками потокових шифрів є скремблери.
2. Блокові шифри. Одиницею кодування є блок з декількох байтів. Результат кодування залежить від всіх початкових байтів цього блоку. Схема застосовується при пакетній передачі інформації і кодуванні файлів.

СИМЕТРИЧНІ КРИПТОАЛГОРИТМИ

Скремблери

Скремблерами називаються програмні або апаратні реалізації алгоритму, що дозволяє шифрувати побітно 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 «Виключаюче АБО» (XOR).

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

Суть скремблірування полягає в побітної зміні потоку даних, що проходять через систему. Практично єдиною операцією, що використовується в скремблерах є XOR – що "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 за XOR кодуючої послідовності на початкову.

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

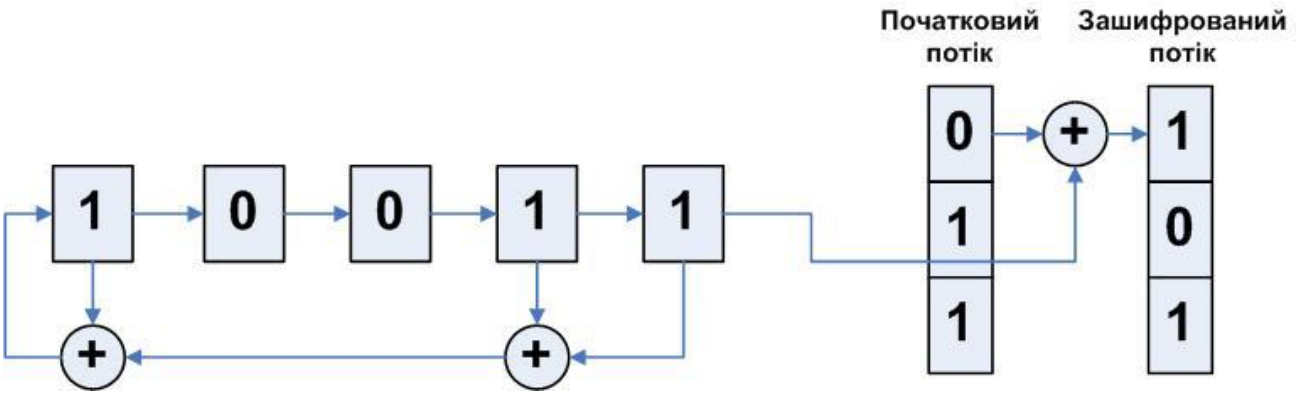


Рис.1. Структура п’ятирозрядного скремблера

З теорії передачі даних криптографія використовує для запису подібних схем двійкову систему запису. Т.ч., зображений на рис. 1 скремблер записується комбінацією "100112" – 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.

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

і т.д.

Таким чином, устрій скремблера гранично простий. Його реалізація можлива як на електронній, так і на електричній базі, що і забезпечило його широке застосування в польових умовах. Той факт, що кожен біт вихідної послідовності залежить лише від одного вхідного біта, ще більш зміцнило становище скремблерів в захисті потокової передачі даних. Це пов'язано з тими, що неминуче виникають в каналі передачі перешкодами, які можуть спотворити в цьому випадку лише ті біти, на які вони припадають, а не пов'язану з ними групу байт, як це має місце в блочних шифрах.

Декодування скрембльованих послідовностей відбувається по тій же самій схемі, що і кодування. Саме для цього в алгоритмах застосовується результуюче кодування по тому, що "виключає АБО" – схема, однозначно відновлена при раскодуванні без яких-небудь додаткових обчислювальних витрат.

Головна проблема шифрів на основі скремблерів - синхронізація передавального (що кодує) і приймаючого (що декодує) пристроїв. При пропуску або помилковому вставленні хоч би одного біта вся передавана інформація незворотно втрачається. Тому, в системах шифрування на основі скремблерів дуже велика увага приділяється методам синхронізації. На практиці для цих цілей звичайно застосовується комбінація двох методів:

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

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

Число біт, охоплених зворотним зв'язком, тобто розрядність при-строю пам'яті, що містить кодуючу послідовність біт, називається розрядністю скремблера. Зображений на рис. 1 скремблер має розрядність 5. Відносно параметрів криптостійкості дана величина повністю ідентична довжині ключа блочних шифрів. Чим більша розрядність скремблера, тим вище криптостійкість системи, заснованої на його використанні.

При достатньо довгій роботі скремблера неминуче виникає його зациклення. Після виконання певного числа тактів створиться комбінація біт, яка вже одного разу виявлялася, і з цієї миті кодуюча послідовність почне циклічно повторюватися з фіксованим періодом. Ця проблема неусувна за своєю природою, оскільки в N розрядах скремблера не може перебувати більш 2^N комбінацій біт, і, отже, максимум, через, 2^N-1 цикл повтор комбінації обов'язково відбудеться. Комбінація "всі нулі" відразу ж виключається з ланцюжка графа станів скремблера – вона призводить скремблер до такого ж стану "всі нулі". Це вказує ще і на те, що ключ "всі нулі" непридатний для скремблера. Кожен біт, що генерується при зсуві, залежить лише від декількох біт тієї комбінації, що зберігається в даний момент скремблером. Тому після повторення деякої ситуації, що одного разу вже зустрічалася в скремблері, всі наступні за нею будуть в точності повторювати ланцюжок, що вже пройшов раніше в скремблері.

Можливі різні типи графів стану скремблера. На рис. 2 приведені зразкові варіанти для 3-розрядного скремблера. На рис. 2а крім завжди присутнього циклу "000">>"000" показані ще два цикли – з 3-мя станами і 4-мя. На рис. 2б показаний ланцюжок, який сходиться до циклу з 3-х станів і вже ніколи звідти не виходить. На рис. 2в всі можливі стани крім нульового, об'єднані в один замкнутий цикл. Очевидно, що саме в цьому випадку, коли всі 2^N-1 стан системи утворюють цикл, період повторення вихідних комбінацій максимальний, а кореляція між довжиною циклу і початковим станом скремблера (ключем), яка привела 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.

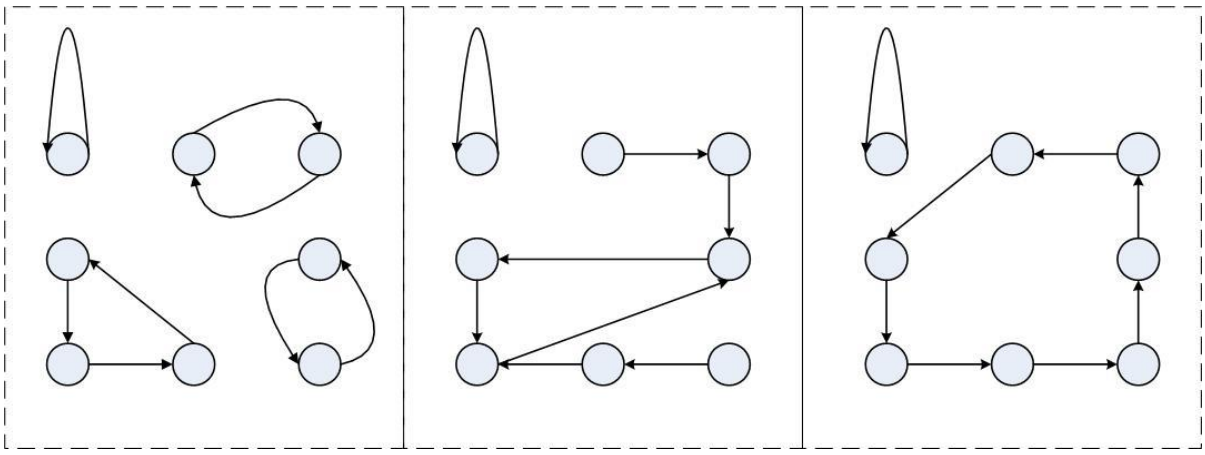


Рисунок 2 – Графи, що характеризують стан скремблера

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, можна завжди знайти в спеціалізованих математичних довідниках.

Істотним недоліком скремблерів є їх нестійкість до фальсифікації.

Блокові шифри

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 те, що в ході своєї роботи вони здійснюють перетворення блоку вхідної інформації фіксованої довжини і одержують результуючий блок того ж обсягу, але недосяжний для прочитання стороннім особам, що не володіють ключем. Таким чином, схему роботи блокового шифру можна 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 реалізовано 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, що є більш повільними. Відсутність статистичної кореляції між бітами вихідного потоку блокового шифру використовується для обчислення контрольних сум пакетів даних і в хешуванні паролів.

Наступні розробки всесвітньо визнані стійкими алгоритмами :

Назва алгоритму	**Автор**	**Розмір блоку**	**Довжина ключа**
IDEA	Xuejia Lia and James Massey	64 біти	128 біт
CAST128		64 біти	128 біт
BlowFish	Bruce Schneier	64 біти	128 – 448 біт
ГОСТ	НДІ	64 біти	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 теорією 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 експоненційно із її збільшенням. 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 менше 10^{12} років. Але це відноситься тільки до ідеально стійких шифрів, якими, наприклад, з великим ступенем упевненості є алгоритми, що наведені в таблиці.

Окрім цієї умови до ідеально стійких криптоалгоритмів застосовується ще одна дуже важлива вимога, якій вони повинні обов'язково відповідати. При відомих початковому і зашифрованому значеннях блоку, ключ, яким було здійснено це перетворення, можна дізнатися також тільки повним перебором. Ситуації, в яких сторонньому спостерігачу відома частина початкового тексту зустрічаються повсюдно. Це можуть бути стандартні надписи в електронних бланках, фіксовані заголовки форматів файлів, що досить часто зустрічаються в тексті, довгі слова або послідовності байт.

Таким чином, на функцію стійкого блокового шифру Z = EnCrypt(X, Key) накладаються наступні умови:

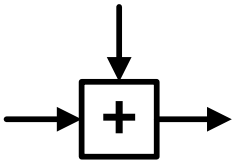
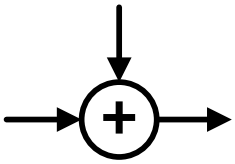
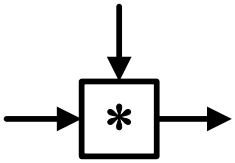
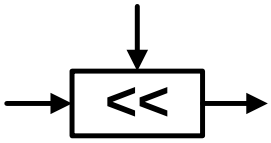
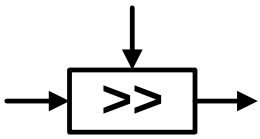
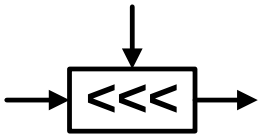
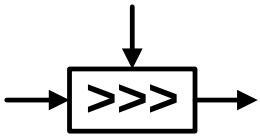
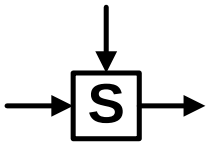
1. Функція EnCrypt повинна бути оборотною.
2. Не повинне існувати інших методів прочитання повідомлення X за відомим блоком Z, окрім як повним перебором ключів Key.
3. Не повинне існувати інших методів визначення яким ключем Key було здійснено перетворення відомого повідомлення X в повідомлення Z, окрім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

Розглянемо методи, за допомогою яких розробники блокових криптоалгоритмів досягають одночасного виконання цих трьох умов з дуже великим ступенем достовірності.

Всі дії, що виконуються над даними блоковим криптоалгоритмом, засновані на тому факті, що перетворюваний блок може бути наведено у вигляді цілого ненегативного числа з діапазону, відповідного його розрядності. Так, наприклад, 32-бітовий блок даних можна інтерпретувати як число з діапазону 0…4 294 967 295. Крім того, блок, розрядність якого звичайно є "ступенем двійки", можна трактувати як декілька незалежних ненегативних чисел з меншого діапазону (розглянутий вище 32-бітовий блок можна також навести у вигляді двох незалежних чисел з діапазону 0…65535 або у вигляді чотирьох незалежних чисел з діапазону 0…255).

Над цими числами блоковим криптоалгоритмом виконуються наступні дії за певною схемою (зліва наведені умовні позначення цих операцій на графічних схемах алгоритмів):

Таблиця 1 - Бієктівні математичні функції

	Додавання	X' = X + V
	Виключаюче АБО	X' = X XOR V
	Множення за модулем 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]

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

1. фіксоване число (наприклад, X' = X + 125)
2. число, що отримується з ключа (наприклад, X' = X + F(Key))
3. число, що отримується з незалежної частини блоку (наприклад, X2' = X2 + F(X1)).

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

Послідовність операцій, що виконуються над блоком, є комбінаціями зазначених вище варіантів V, а власне функції F 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) ганебно йде в історію криптографії.

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 власне значення ключа або його частини, а деяка, іноді необоротна (небієктівна) функція від матеріалу ключа. Більш того, в таких перетвореннях один і той же блок або елемент ключа використовується багато разів. Це дозволяє при виконанні умови оборотності функції щодо величини X зробити функцію необоротною 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 Vi(Key) залишається незмінним, то іноді стає доцільно наперед одноразово обчислити такі значення і зберігати їх в оперативній пам'яті спільно з ключем. Оскільки ці значення залежать тільки від ключа, то в криптографії вони називаються матеріалом ключа. Ця операція жодним чином 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 носять назву розширення ключа (key scheduling).

Мережа Фейштеля

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

Класична мережа Фейштеля має структуру, що наведено на рис. 3.

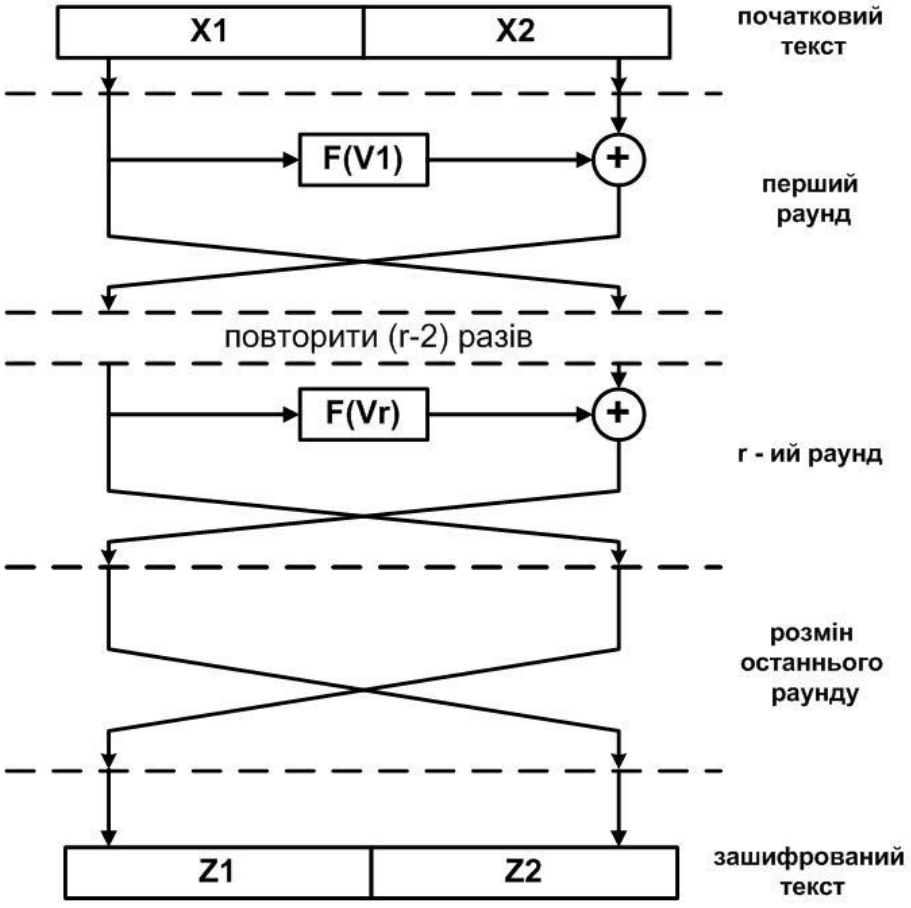


Рис.3 – Структура мережі Фейштеля

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 однократного 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 криптостійкість 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 розумні 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 відновлений. 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 парною або непарною. 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 можна зробити і абсолютно симетричною, тобто використовуючи утворюючі функції шифрування і дешифрування з одним і тим же набором операцій. Математичною мовою це записується як "Функція EnCrypt тотожно дорівнює функції DeCrypt". Розглянемо граф станів криптоалгоритма (рис. 4), на якому вершинами позначено блоки вхідної і вихідної інформації.

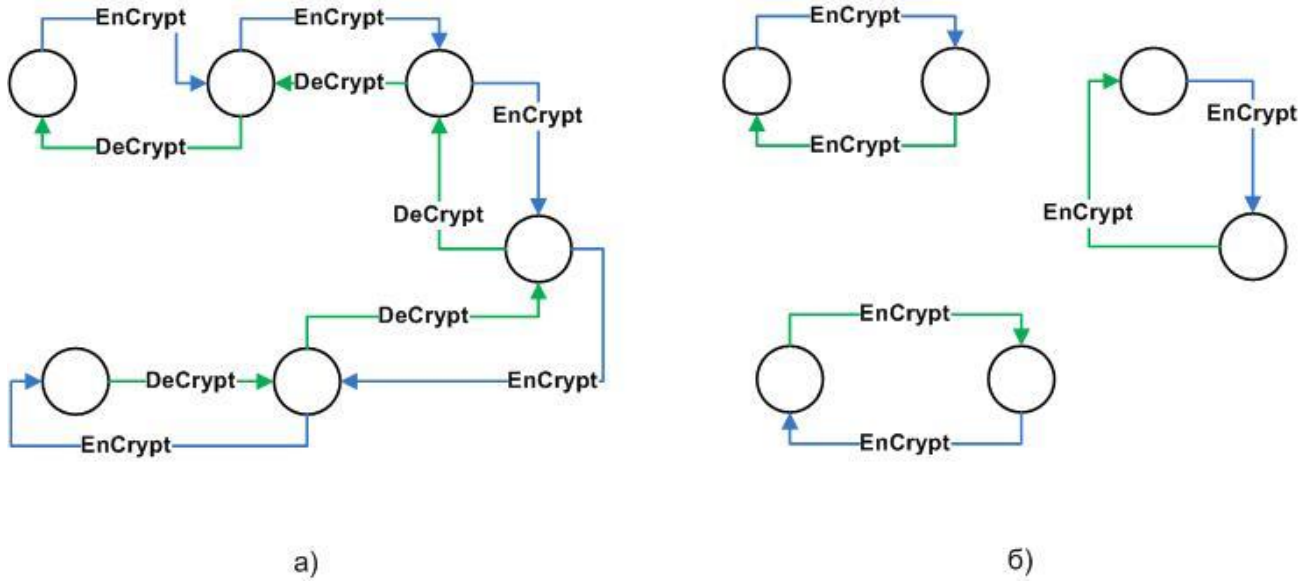


Рис. 4 – Графи станів мережі Фейштеля

При деякому фіксованому ключі для класичної мережі Фейштеля граф відповідає рис. 4а, а в іншому випадку кожна пара вершин має унікальний зв'язок, як зображено на рис. 4б. Модифікація мережі Фейштеля, що володіє подібними властивостями приведена на рис. 5. Основна її відмінність полягає у повторному використанні даних ключа в зворотному порядку в другій половині циклу. Саме через цю недостатню дослідженість специфіки схеми (тобто потенційної можливості послаблення зашифрованого тексту зворотними перетвореннями) її обмежено використовують в криптоалгоритмах.

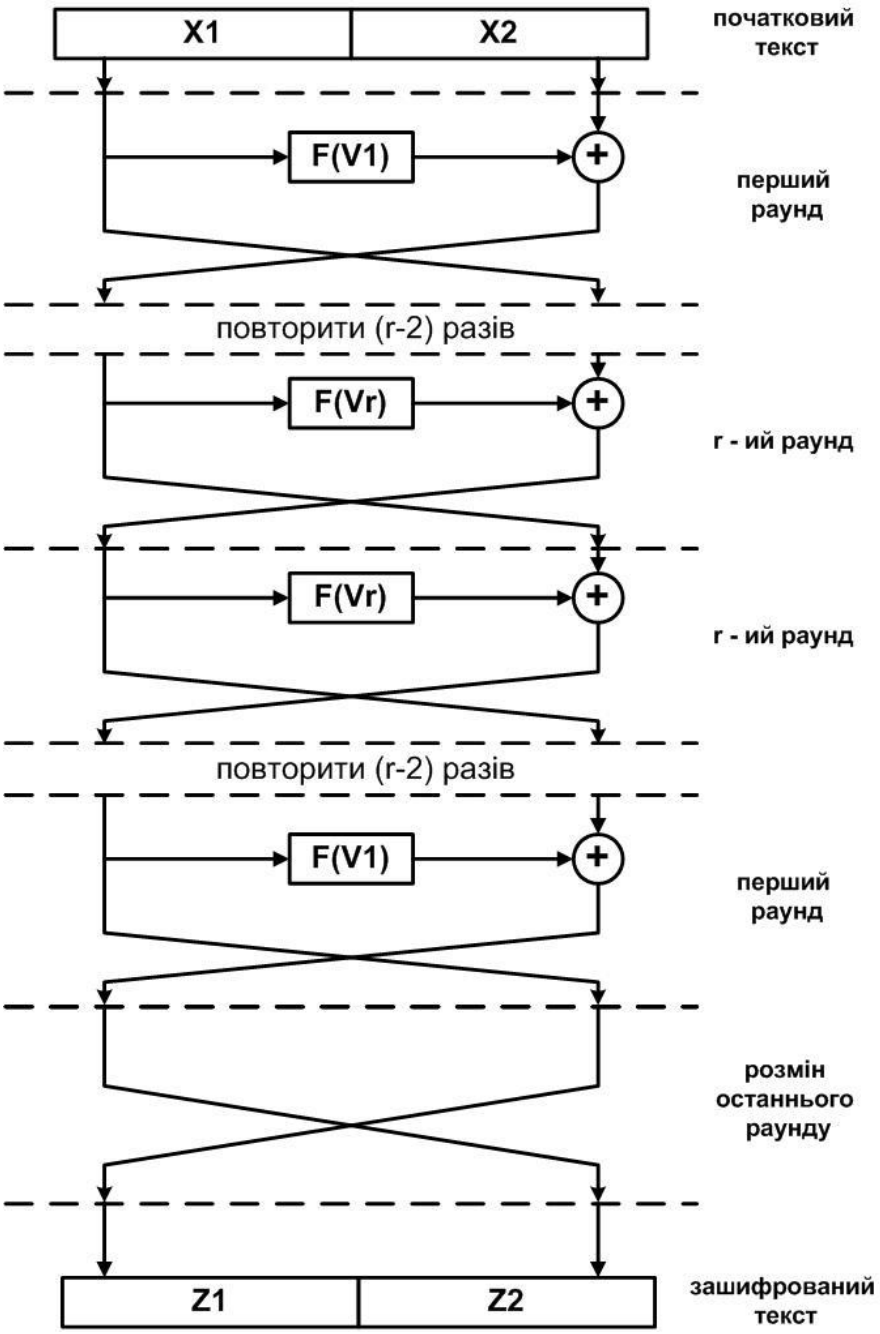


Рис.5 – Симетрична мережа Фейштеля

Модифікації мережі Фейштеля для більшої кількості гілок застосовують набагато частіше. Це пов'язано з тим, що при великих розмірах кодованих блоків (128 і більш біт) стає незручно використовувати математичні функції за модулем 128 і вище. Тому все частіше і частіше в блокових криптоалгоритмах зустрічаються мережі Фейштеля з чотирма гілками. Найпростіший принцип її модифікації зображений на рис. 6а. Для швидшого перемішування інформації між гілками (це основна проблема мережі Фейштеля з великою кількістю гілок) застосовують дві модифіковані схеми, звані "type-2" і "type-3" (рис. 6б і 6в відповідно).

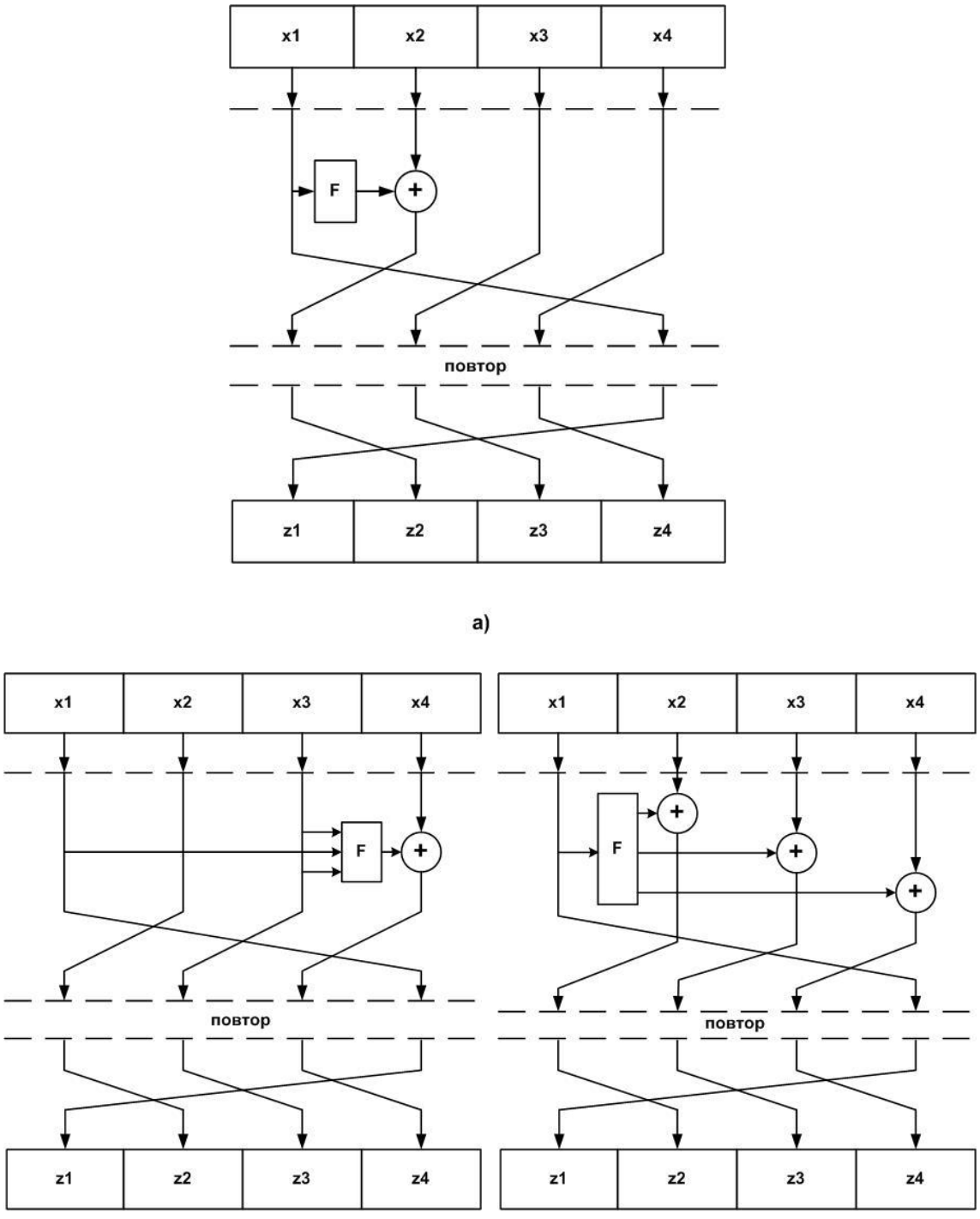


Рис. 6 - Модифікації мережі Фейштеля з чотирма гілками

Мережа Фейштеля надійно зарекомендувала себе як криптостійка схема здійснення перетворень, і її можна 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 складанням, призначені для підвищення початкової рандомізації вхідного тексту. Таким чином, криптостойкість блокового шифру, що використовує мережу Фейштеля, визначається на 95% функцією F і правилом обчислення V_i з ключа. Ці функції і є об'єктом всі нових і нових досліджень фахівців в області криптографії.

Блоковий шифр TEA

Розглянемо один з найпростіших у 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 складання. Схема роботи алгоритму приведено на рис. 7.

Нижче приведений код криптоалгоритма на мові програмування PASCAL.

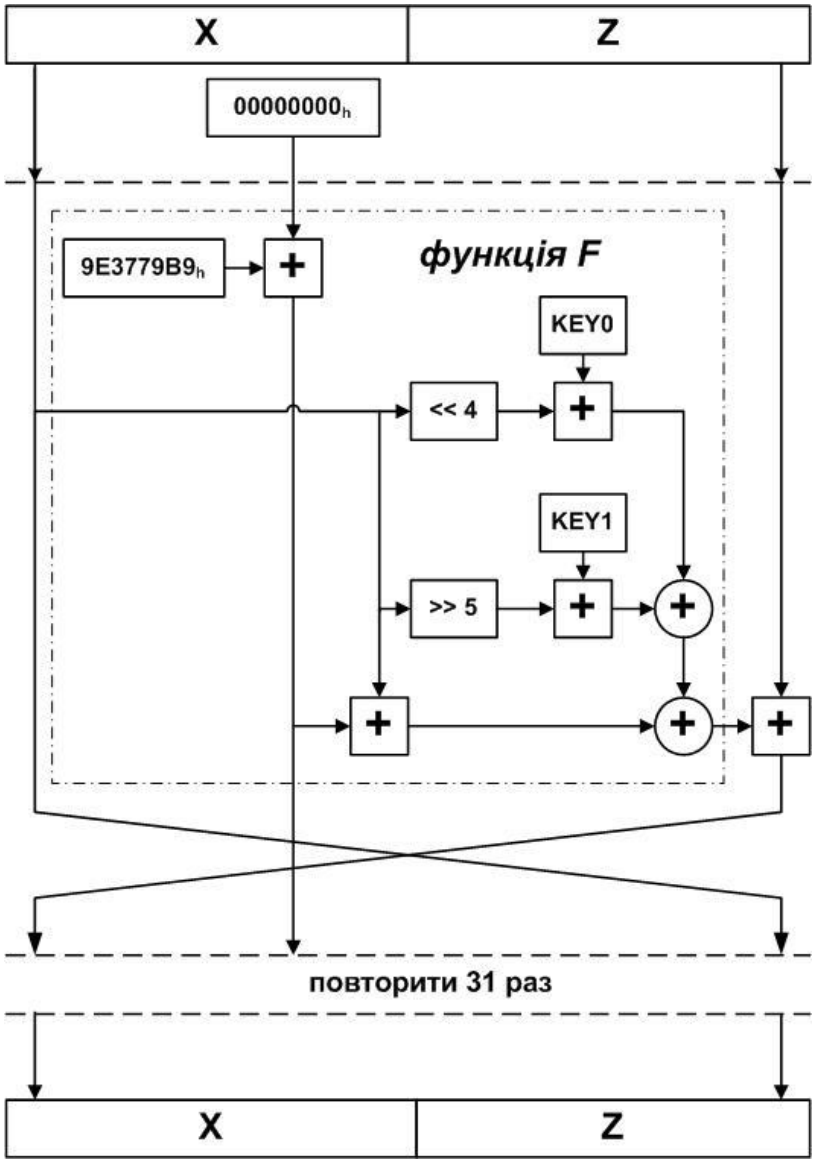


Рис. 7 – Схема алгоритму TEA

```
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 гранично малому обсязі коду. Недоліком алгоритму є деяка повільність, викликана необхідністю повторювати цикл Фейштеля 32 разів (це необхідно для ретельного "перемішування даних" через відсутність табличних підстановок).

AES : стандарт блокових шифрів США

Загальні відомості з конкурсу AES

У 80-х роках минулого сторіччя в США був прийнятий стандарт симетричного криптоалгоритма для внутрішнього вживання 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 мала на сучасному етапі розвитку ЕОМ,

2) другорядна – при розробці алгоритм був орієнтований на апаратну реалізацію, тобто містив операції, виконувані на мікропроцесорах за великий час (наприклад, 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 найкрупніші світові центри з кріптології. Тим самим, переможець цього конкурсу, названого AES – Advanced Encryption Standard, став де-факто світовим криптостандартом.

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

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

Додатково кандидатам рекомендувалося:

1. використовувати операції, що легко реалізовати як апаратно (в мікрочіпах), так і програмно (на персональних комп'ютерах і серверах),
2. орієнтуватися на 32-розрядні процесори,
3. не ускладнювати без необхідності структуру шифру для того, щоб всі зацікавлені сторони були в змозі самостійно здійснити незалежний криптоаналіз алгоритму і переконатися, що в ньому не створено яких-небудь недокументованих можливостей.

На першому етапі в оргкомітет змагання надійшло 15 заявок з абсолютно різних кутів світу. Протягом двох років фахівці комітету, досліджуючи самостійно, і вивчаючи публікації інших дослідників, обрали п’ять кращих представників, що пройшли до "фіналу" конкурсу.

Назва алгоритму	Розробник	Країна	Швидкодія (asm, 200МГц)
MARS	IBM	US	8 Мбайт/с
RC6	R. Rivest & Co	US	12 Мбайт/с
Rijndael	V. Rijmen & J. Daemen	BE	7 Мбайт/с
Serpent	Universities	IS, UK, NO	2 Мбайт/с
TwoFish	B. Schneier & Co	US	11 Мбайт/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 алгоритма - переможця були зняті всі патентні обмеження – його можна буде 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.

Шифр MARS

Цей 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 проводиться додавання

матеріалу ключа. Мета цього перетворення – ретельна рандомізація даних і підвищення стійкості шифру до деяких видів атак (рис. 8).

Третій етап: власне шифрування. В ньому використовується мережа Фейштеля третього типу з 4 гілками, тобто значення трьох функцій, обчислених від однієї гілки накладаються відповідно на три інших, потім йде перестановка машинних слів. Ця операція також повторюється 8 разів (рис. 8). Саме на цьому етапі відбувається змішування тексту з основною (більшою) частиною матеріалу ключа. Самі функції, що накладаються на гілки, зображені на рис. 9. Т.ч., в алгоритмі MARS використані практично всі види операцій, що використовуються в криптографічних перетвореннях: складання, "Виключаюче АБО", зсув на фіксовану кількість біт, зсув на змінну кількість біт, множення і табличні підстановки.

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

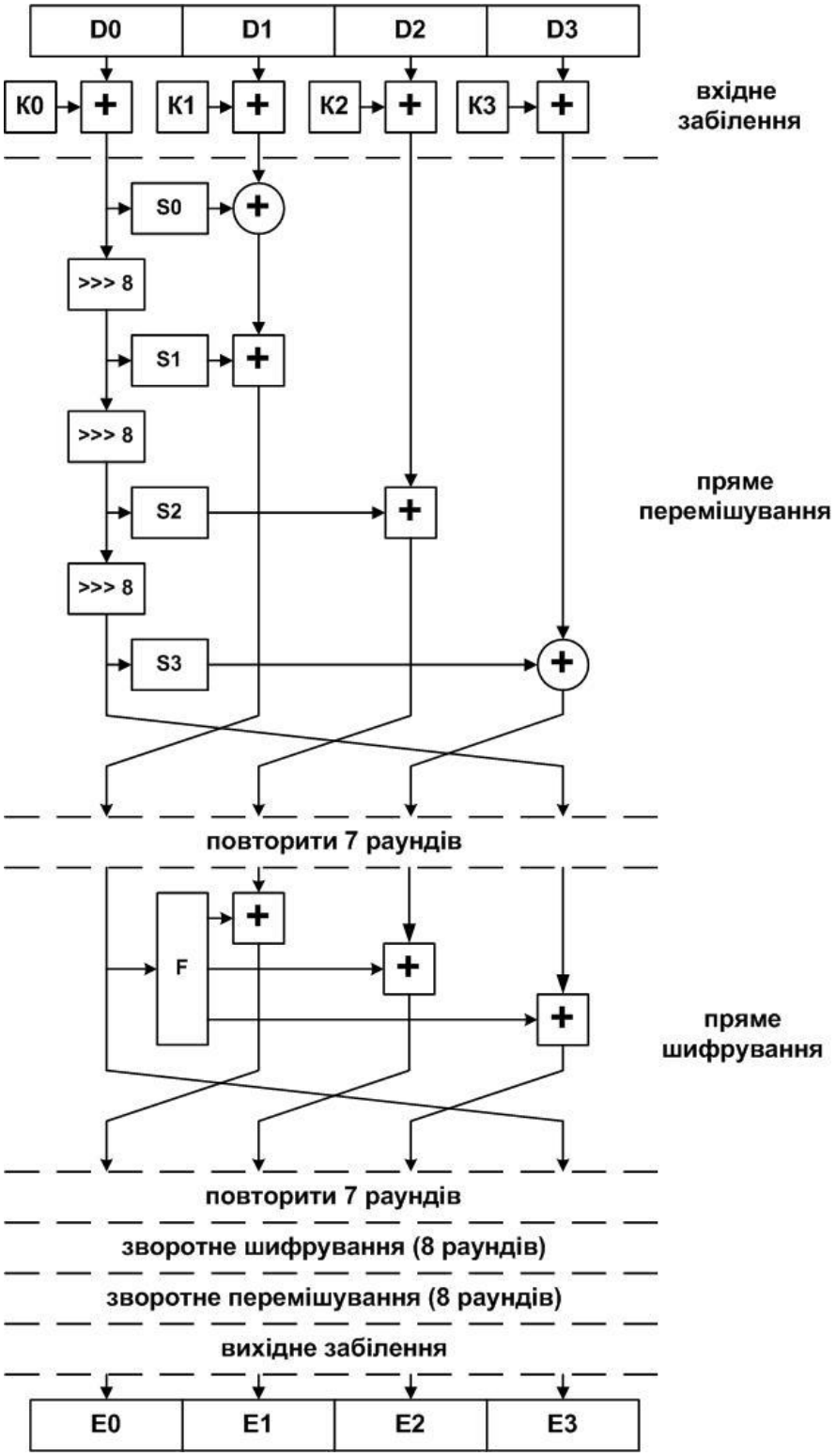


Рис. 8 – Схема алгоритму MARS

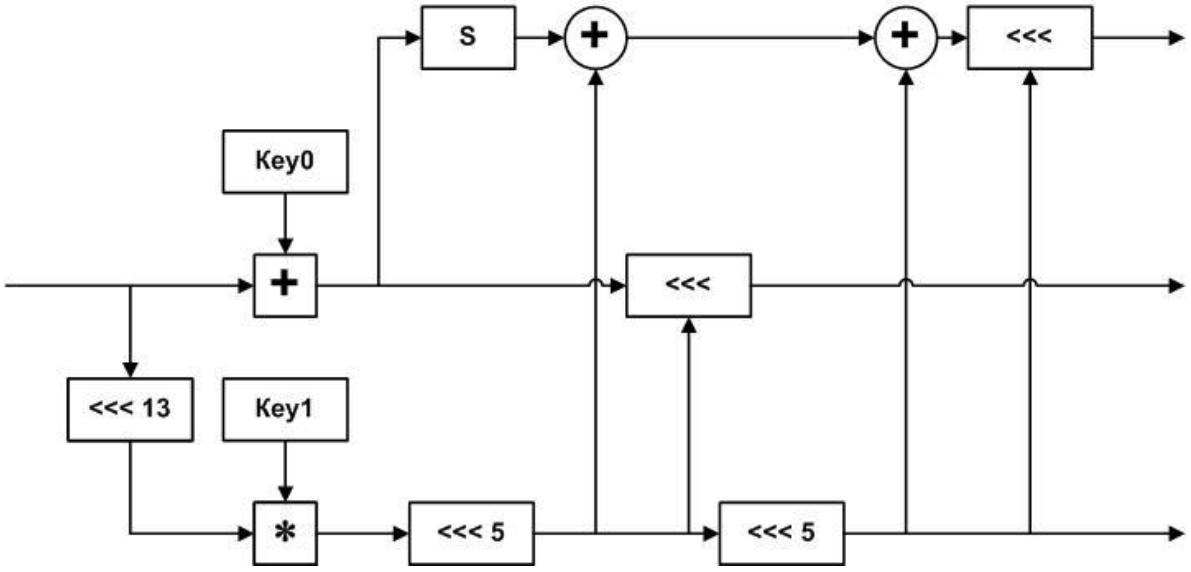


Рис. 9 – Схема утворюючої функції алгоритму MARS

Шифр RC6

Алгоритм є продовженням криптоалгоритма RC5, розробленого Р. Рівестом – дуже відомим науковцем у галузі криптографії. RC5 був трохи 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 у RC5, має солідний обсяг досліджень, проведених задовго до оголошення конкурсу AES.

Алгоритм є мережею Фейштеля з 4 гілками змішаного типу: в ньому два парні блоки використовуються для одночасної зміни вмісту двох непарних блоків (рис. 10). Потім здійснюється звичайний для мережі Фейштеля зсув на одне машинне слово, що змінює парні і непарні блоки місцями. Сам алгоритм гранично простий і

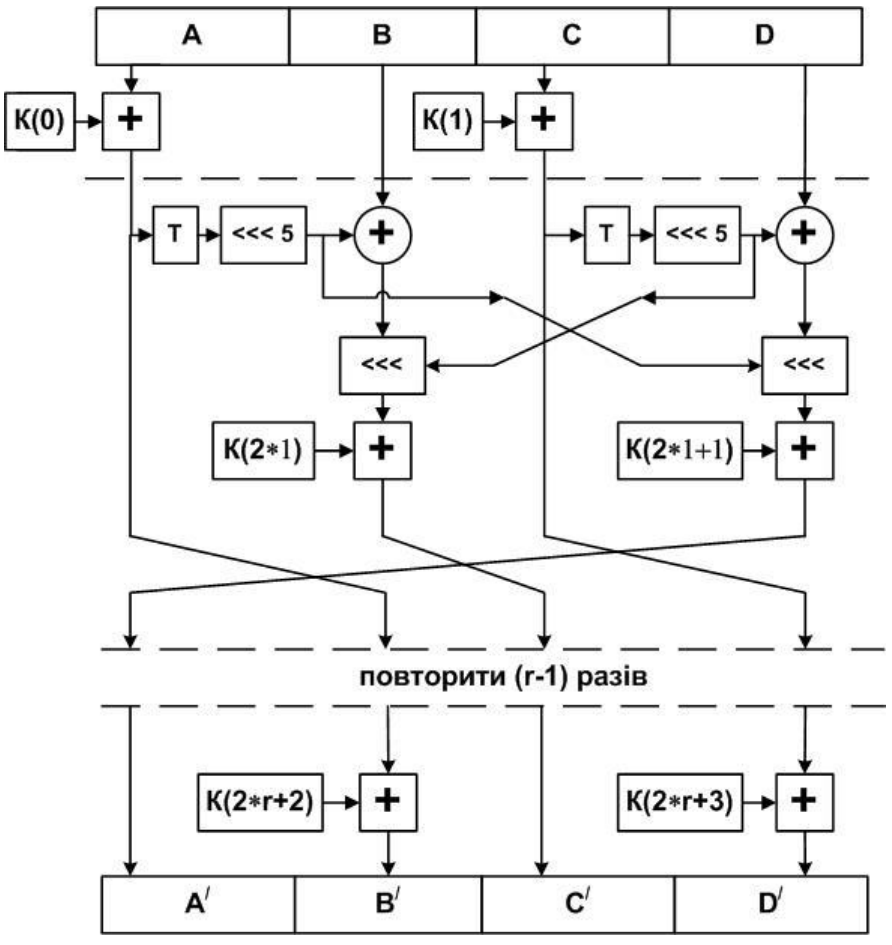


Рис. 10 – Схема алгоритму MARS

зображено на рис. 10. Розробники рекомендують використовувати при шифруванні 20 раундів мережі, хоча їх кількість не регламентується. При 20 повторах операції шифрування алгоритм має найвищу швидкість серед п’ятьох фіналістів AES.

Перетворення T(X) є дуже простим :

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

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

Шифр Serpent

Алгоритм розроблений групою фахівців з декількох дослідницьких центрів світу. Алгоритм є мережою Фейштеля з чотирма гілками змішаного типу : 2 парні гілки змінюють сумісно значення непарних, потім міняються місцями (рис. 11). У якості криптоперетворювань використовуються операції "Виключаюче АБО", табличні підстановки і бітові зсуви. Алгоритм складається з 32 раундів. Власно раунди складені таким чином, що додавання до гілок матеріалу ключа на першому і останньому раундах утворює вхідне і вихідне забілення.

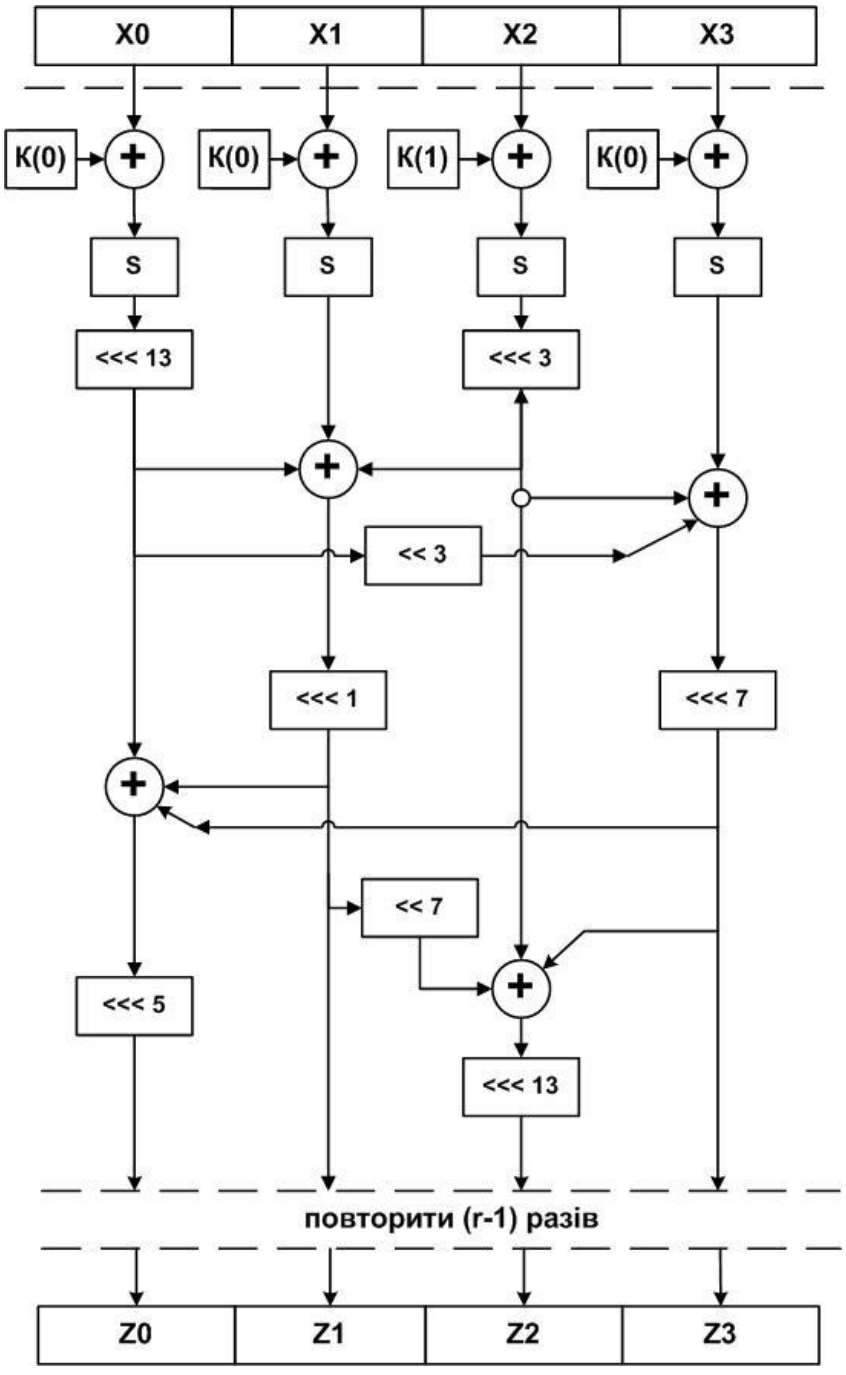


Рис. 11 – Схема алгоритму Serpent

Шифр TwoFish

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

У алгоритмі TwoFish розробники залишили деякі

вдалі рішення з проекту-попередника, окрім 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, а на парних раундах ситуація міняється на протилежну (рис. 12). В алгоритмі використовується криптоперетворювання Адамара (Pseudo-Hadamar Transform) – зворотне арифметичне складання першого потоку з другим, а потім другого з першим.

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

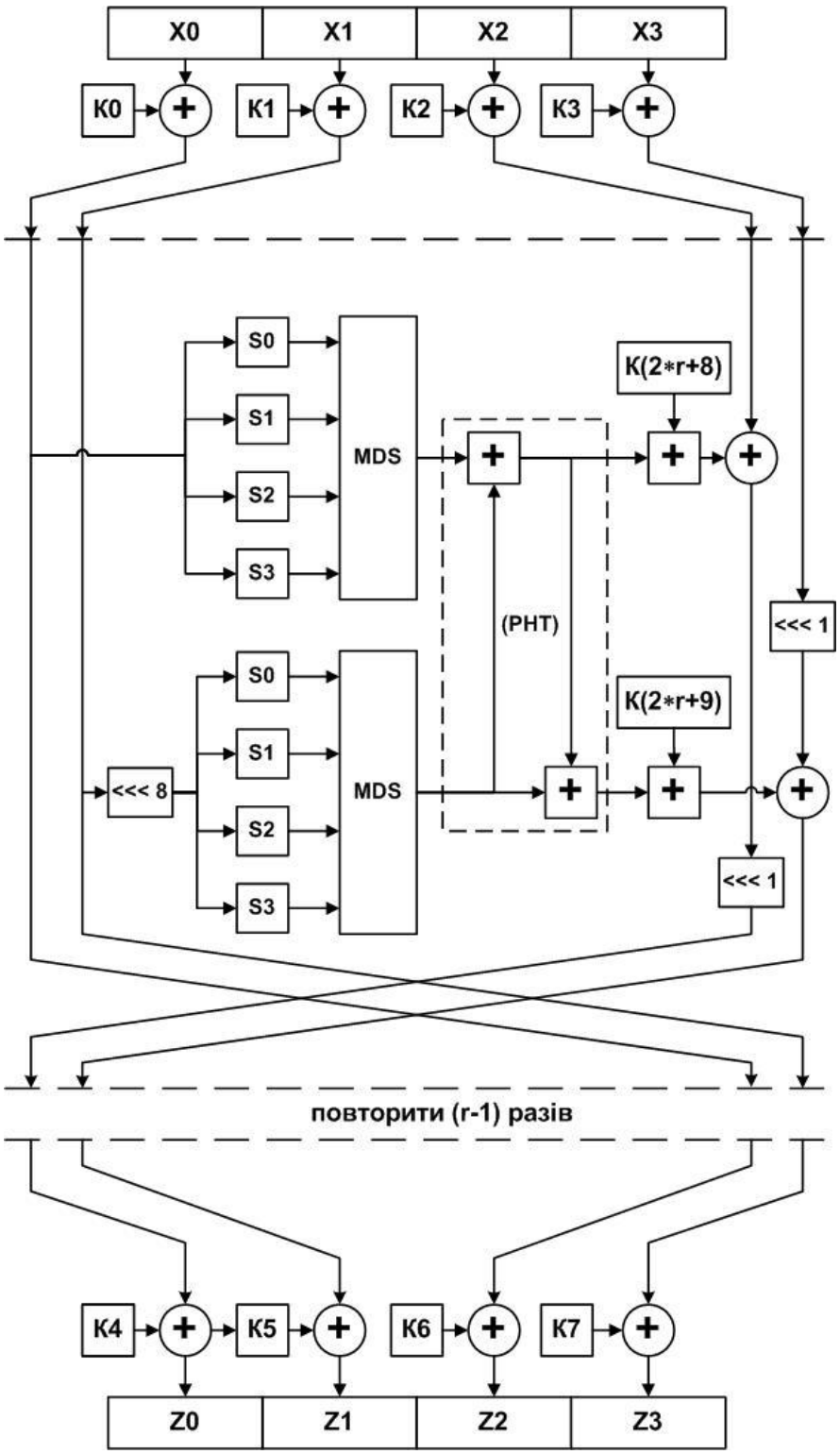


Рис. 12 – Схема алгоритму TwoFish

Шифр Rijndael

Цей алгоритм було розроблено двома фахівцями з криптографії з Бельгії. Він є нетрадиційним блоковим шифром, оскільки не використовує мережу Фейштеля для криптоперетворювань. Алгоритм подає кожний блок кодованих даних у вигляді двовимірного масиву байт розміром 4x4, 4x6 або 4x8 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.13),

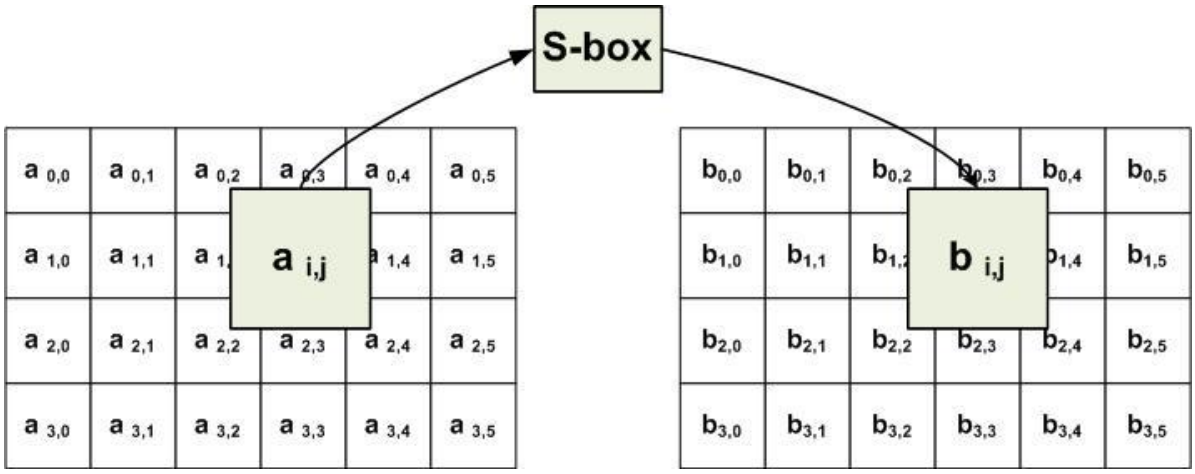


Рис. 13 – Схема табличної підстановки алгоритму Rijndael

- ShiftRow – зсув рядків в двовимірному масиві на різну кількість елементів (рис. 14),

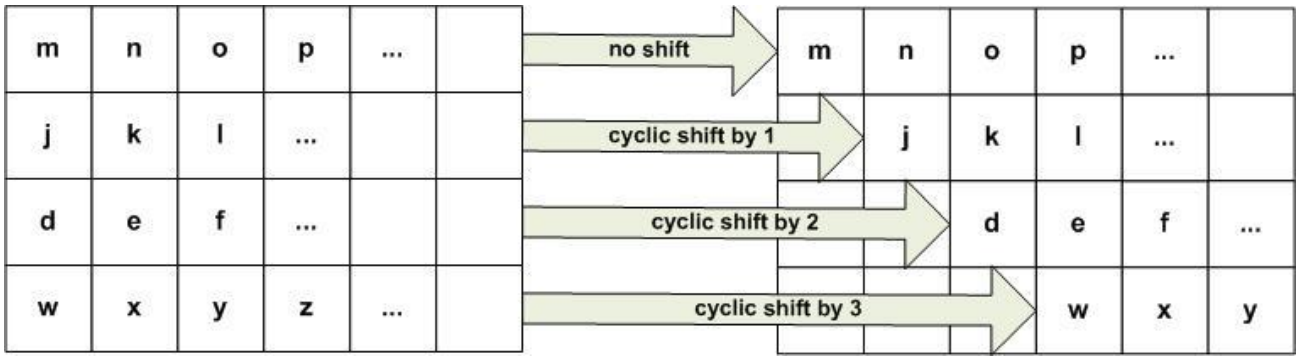


Рис. 14 - Схема зсувів у алгоритмі Rijndael

- MixColumn – математичне перетворення, що перемішує дані усередині стовпця (рис. 15),

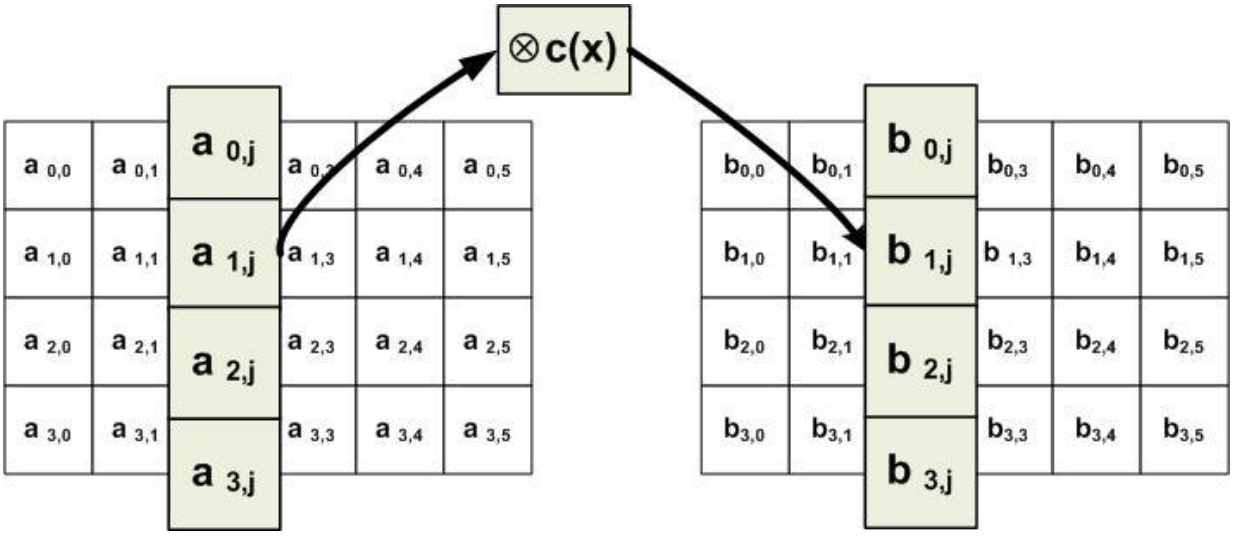


Рис. 15 - Схема математичних перетворювань алгоритму Rijndael

- AddRoundKey – додавання матеріалу ключа операцією XOR (рис. 16).

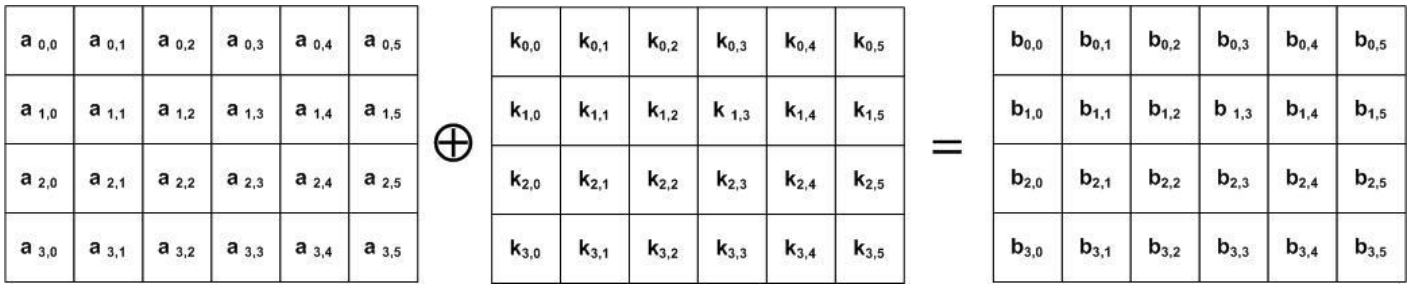


Рис. 16 – Додавання матеріалу ключа у алгоритмі Rijndael

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