Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Інфокомунікацій*
(повна назва)
Кафедра *Інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження криптографічних систем, симетричної та асиметричної архітектури
(тема)

Виконав:
студент _2_ курсу, групи *ІМІм-21-1*
Пермяков О.О.
(прізвище, ініціали)

Спеціальність *172. Телекомунікації та радіотехніка*
(код і повна назва спеціальності)
Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Інформаційно-мережна інженерія*
(повна назва освітньої програми)

Керівник *проф. Тихонов В.В.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ *Безрук В.М.*
(підпис) (прізвище, ініціали)

2024 р.

Не містить відомостей, заборонених
до відкритого публікування

Керівник _____ /*В.А.Тихонов*

Студент _____ / *О.О. Пермяков*

Харківський національний університет радіоелектроніки

Факультет *Інфокомунікацій*

Кафедра *Інформаційно-мережної інженерії*

Рівень вищої освіти другий (магістерський)

Спеціальність *172. Телекомунікації та радіотехніка*
(код і повна назва)

Тип програми *Освітньо-професійна*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Інформаційно-мережна інженерія*
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«_23_» *жовтня* 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Пермякову Олександру Олександровичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження криптографічних систем, симетричної та асиметричної архітектури*

затверджена наказом університету від 23 жовтня 2023 р. № 1233 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 24 січня 2024 р.

3. Вихідні дані до роботи *Дослідити існуючі шифросистеми симетричного та асиметричного типів. Визначити такі з них, архітектура яких є найбільш перспективною для створення власної шифросистеми. При цьому, такий шифр повинен забезпечувати можливість обробки мультимедійних даних у реальному часі. Визначити, яка з поширених асиметричних криптографічних схем може бути використаною для обміну ключами симетричного шифрування у ході обробки мультимедіа. Побудувати кодову реалізацію шифру на базі визначеної архітектури.*

4. Перелік питань, що потрібно опрацювати в роботі *Вступ*
1 . Загальні питання криптографічного захисту даних
2. Дослідження симетричної та асиметричної архітектур шифрування
3. Дослідження існуючих криптосистем
4. Синтез криптографічної шифросистеми на базі мережі Фейстеля
Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____
слайди презентації в форматі Power Point (назва та мета роботи, основні компоненти шифросистеми, симетричні та асиметричні шифросистеми, швидкість шифротворення поширених алгоритмів, вимоги до швидкості шифротворення, композиційні блокові шифри, мережа Фейстеля, алгоритм AES, алгоритм DES, побудова шифру на базы мережы Фейстеля, висновки)

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Вступ	24.10.23-26.10.23	виконано
2	Загальні питання криптографічного захисту даних	27.10.23-8.11.23	виконано
3	Дослідження симетричної та асиметричної архітектур шифрування	9.11.23-18.11.23	виконано
4	Дослідження існуючих криптосистем	19.11.23-27.11.23	виконано
5	Синтез криптографічної шифросистеми на базі мережі Фейстеля	28.11.23-16.12.23	виконано
6	Висновки	17.12.23-19.12.23	виконано
7	Оформлення пояснювальної записки	20.12.23-5.1.24	виконано

Дата видачі завдання *24 жовтня* 2023 р.

Студент _____
(підпис)

Керівник роботи _____ *проф. Тихонов В.А.*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 67 с., 18 рис., 21 джерело, 2 додатки

MIPS, AES, DES, RSA, ШИФР ЕЛЬ-ГАМАЛЯ, МЕРЕЖА ФЕЙСТЕЛЯ, РАУНДОВИЙ КЛЮЧ, ЛАВИННИЙ ЕФЕКТ, ДИФУЗІЯ

Об'єкт дослідження – принципи побудови шифрів симетричного типу.

Мета роботи – дослідження можливості побудови криптосистеми з високою швидкістю шифротворення на базі існуючої архітектури шифру симетричного типу.

Розглядаються засади побудови асиметричних та симетричних шифрів. Доводиться доцільність використання симетричних шифрів для обробки мультимедійних даних реального часу. Виконується аналіз архітектурних особливостей шифрів на основі мережі Фейстеля. На базі цього створюється програмна реалізація шифру.

THE ABSTRACT

Explanatory note: 67 p., 18 fig., 21 sources, 2 apps.

MIPS, AES, DES, RSA, EL-GAMAL CIPHER, FEISTEL NETWORK, ROUND KEY, AVALANCHE EFFECT, DIFFUSION

The object of research is the principles of constructing symmetric ciphers. The purpose of the work is to study the possibility of building a cryptosystem with high encryption speed based on the existing symmetric cipher architecture.

The basics of constructing asymmetric and symmetric ciphers are considered. The expediency of using symmetric ciphers for real-time multimedia data processing is proved. An analysis of the architectural features of ciphers based on the Feistel network is performed. Based on this, a software implementation of the cipher is created.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ

UTF-8 – (Unicode Transformation Format, 8-bit) – стандарт кодування символів Юнікоду;

ASCII – (American Standard Code For Information Interchange) – стандарт кодування символів;

IPS – (instructions per second) – кількість інструкцій за секунду;

CUDA – (Compute Unified Device Architecture) – програмно-апаратна архітектура паралельних обчислень;

VoIP – (Voice Over IP) – група сервісів, що надають послуги голосового зв'язку у IP-мережах;

VoD – (Video On Demand) – група сервісів, що надають послуги доставки потокового відео на запит користувача;

AoD – (Audio On Demand) — група сервісів, що надають послуги доставки потокового аудіо на запит користувача;

QoS – технологія контролю якості послуг, які надає інформаційно-комунікаційна мережа;

RSA – (Rivest, Shamir, Adleman) – асиметричний алгоритм шифрування;

AES – (Advanced Encrypting System) – симетричний алгоритм блочного шифрування;

DES – (Data Encrypting System) – симетричний алгоритм блочного шифрування;

XSL – (eXtended Sparse Linearization) – алгебраїчна атака, метод криптографічного аналізу.

ВСТУП

Серед пріоритетних напрямків розвитку інформаційно-комунікаційних систем та їх окремих компонент одним з головних є забезпечення захищеності даних у ході їх зберігання та/або під час сеансів обміну.

При цьому, під захищеністю даних у загальному випадку слід розуміти мінімізацію, або узагалі нівелювання ймовірності несанкціонованого доступу до тієї чи іншої інформації. В більш уточненому трактуванні данні можуть вважатися захищеними, якщо для них забезпечується виконання ряду вимог, серед яких:

- цілісність;
- доступність;
- конфіденційність.

Тут конфіденційність вказує на те, що доступ до інформації має тільки певне встановлене коло осіб.

У свою чергу, цілісність гарантує, що лише перший перелік осіб може вносити зміни у дані.

Доступність у даному разі слід сприймати як властивість, у разі забезпечення якої авторизовані особи гарантовано зможуть мати доступ до інформації.

Разом з тим, говорячи про існуючі шляхи досягнення захищеності інформації, зазначимо, що сьогодні, у загальному випадку, застосовуються:

- шифрування та маскування даних на рівні джерела;
- шифрування даних на рівні мережевих протоколів (наприклад, IPSec, Kerberos тощо);
- тунелювання трафіку;
- логічне розмежування даних у загальному середовищі (наприклад, створення віртуальних мереж);
- фізичне розмежування каналів обміну даних з обмеженим доступом та загальнодоступних каналів.

При цьому, традиційно ті чи завдання забезпечення захищеності даних вирішуються у залежності від:

- типу даних, які потребують захисту;
- того, чи є необхідним зберігати дані у сховищі, чи такі дані беруть участь у транзакціях;

- інформаційної інтенсивності даних;
- особливостей базового протоколу, що відповідає за доставку даних мережею.

Означені чинники, у свою чергу, здійснюють суттєвий вплив на вибір тих чи інших засобів – так, недоцільно створювати фізично відокремлений канал між філіями підприємства, територіально рознесеними на значні відстані. Також немає сенсу використовувати асиметричну криптосистему для закриття потокового трафіку реального часу.

Зазначимо, що серед перелічених напрямків реалізації безпеки даних одними з найбільш ефективних з позиції підсумкового рівня захищеності інформації можна вважати фізичне розмежування каналів та шифрування/маскування на рівні джерела.

Це пояснюється тим, що:

1. У першому випадку зловмисник повинен отримати фізичний доступ до середовища розповсюдження даних, що не є простим завданням. Утім, якщо таким середовищем є, наприклад радіоефір, або коаксіальний кабель чи кабель типу вита пара, існує можливість пасивного зчитування даних без порушення фізичної цілісності носія. Виходячи з цього, навіть у даному форматі реалізації захисту даних необхідно застосовувати додаткові засоби.
2. У другому випадку є неефективними методи захоплення трафіку, оскільки одержані таким чином пакети даних являтимуть собою шифрограми, які потребують попереднього дешифрування.

Таким чином, потенційно найбільш перспективним напрямком захисту даних можна вважати застосування методів шифрування на рівні джерела. Утім, результативність таких методів залежить від ряду факторів, як то - типу даних, вимог до якості сервісу, інтенсивності інформаційного обміну тощо.

Отже, дослідження методів шифрування, особливостей їх реалізації та умов застосування на сьогодні є актуальним завданням.

1. ЗАГАЛЬНІ ПИТАННЯ КРИПТОГРАФІЧНОГО ЗАХИСТУ ДАНИХ

1.1 Класи даних, які потребують захисту

З позиції вимог щодо захищеності, сьогодні може бути виокремлено такі дані, як:

- конфіденційні дані;
- дані, що є об’єктом комерційної таємниці;
- грифовані дані (для службового користування, таємна та цілком таємна інформація).

Ураховуючи те, що грифовані дані афілюються з профільними відомствами, які беруть участь у формуванні та підтримці інформаційної безпеки суспільства та державних утворень у цілому, відповідно, до зазначених даних висуваються найбільш жорсткі вимоги щодо захищеності. І навпаки – до даних, що являють собою комерційну таємницю та конфіденційних даних вимоги з безпеки є менш жорсткими. Це пов’язано з тим, що негативні наслідки у разі їх втрати чи модифікації матимуть суттєво менший масштаб, ніж для грифованих даних.

При цьому, виходячи з того, що кількість мережевих сервісів та послуг, що надаються на їх базі, постійно збільшується, відтак розширюється перелік класів даних, відносно яких необхідно забезпечити виконання вимог щодо захищеності.

Так, якщо початково захисту потребувала, за великим рахунком, текстова інформація (облікові записи, хеші паролів, інформація банківських транзакцій, безпосередньо конфіденційні дані того чи іншого рівня доступу), надалі виникла необхідність криптографічного захисту даних таких класів, як:

- аудіо (у форматі потоку та/або окремих файлів);
- графічних даних;
- відео (у форматі потоку та/або окремих файлів);
- мультимедійних потоків, що являють собою поєднання усіх зазначений класів, разом з текстовою складовою.

Умовно усю інформацію, яка потребує захисту, можна розподілити на службові дані та, безпосередньо, інформативну складову.

При цьому, між інформативною та службовою складовими існує суттєва різниця у об’ємі, що буде найменшою для інформації текстового класу і максимальною – для т.з. «важких» типів трафіку – відео, аудіо і т.д.

На додачу до усього раніше зазначеного, різний характер даних, що належать різним класам, та різні вимоги для них за показниками затримки, джитеру та допустимого відсотку втрат, визначають різні допустимі умови для шифрування кожного з класів.

Виконаємо дослідження таких умов, а спочатку - розглянемо ключові архітектури шифросистем та їхні характеристики.

1.2 Загальні засади побудови шифросистем

Шифросистема, або криптографічна система – модель обробки даних, у рамках якої відносно інформації виконуються такі операції, як (рис.1.1):

- попереднє форматування;
- шифрування;
- передавання (зберігання);
- дешифрування;
- представлення у вигляді, прийнятному для сприйняття відповідним додатком та/або безпосередньо користувачем (зворотне форматування).

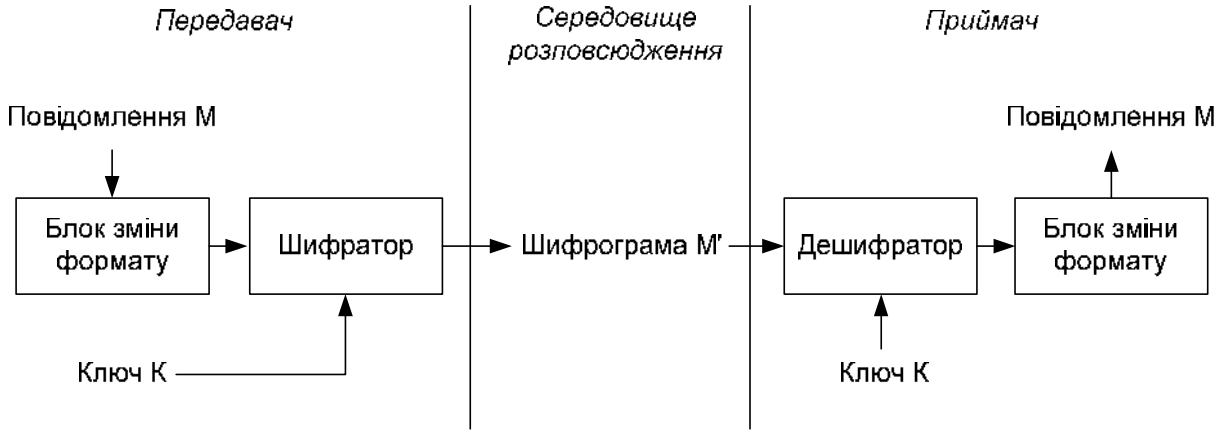

Рисунок 1.1 – Типова схема та основні компоненти шифросистеми

Під попереднім форматуванням тут мається на увазі приведення даних до вигляду, який є необхідним для функціонування безпосередньо алгоритму шифротворення. Це може бути, наприклад:

- конвертація тексту UTF-8 до вигляду бінарних сегментів;
- переведення двійкового масиву до HEX-представлення;
- доповнення масиву символів до значення, кратного певній величині (визначається типом шифру);
- зміна бінарного формату опису на символьний і т.д.

При цьому, операція попереднього форматування, як, відповідно, і фінальна зміна формату після дешифрації, є опціональними. Також за умови їх наявності нерідко дані технологічні етапи не обов'язково входять до складу шифросистеми та реалізуються зовнішніми засобами.

Після приведення повідомлення М до вигляду, прийнятного для шифрування, у блоці шифратора за участю ключа К та на базі того чи іншого алгоритму ε_c шифрування формується шифрограма М′. У загальному випадку дана процедура здійснюється згідно з виразом:

$$M' = \varepsilon_c(M; K). \tag{1.1}$$

Відповідно, після проходження шифрограми M' середовищем передачі у точці прийому виконується її дешифрування.

При цьому, якщо дешифрування здійснюється відповідно до наступного принципу:

$$M = \varepsilon_c(M'; K), \tag{1.2}$$

відтак мова йде про симетричну шифросистему.

У цілому, ознаки симетричної системи може бути сформовано наступною системою виразів:

$$\begin{cases} \varepsilon_c = \varepsilon_d = \varepsilon; \\ K_c = K_d = K, \end{cases} \tag{1.3}$$

де ε_c та ε_d - алгоритми шифрування та дешифрування відповідно, які є аналогічними між собою;

$К_c$ та $К_d$ - ключі шифрування та дешифрування відповідно, які є однаковими для операції як шифрування, так і дешифрування.

У свою чергу, в умовах, коли справедливою є наступна система виразів:

$$\begin{cases} \varepsilon_c \neq \varepsilon_d; \\ К_c \neq К_d, \end{cases} \tag{1.4}$$

розглядається криптографічна система асиметричного типу.

На сьогоднішній час широко використовуються шифросистеми обох типів. При цьому, специфіка та особливості їх застосування визначаються притаманними системам кожного з зазначених типів перевагами та недоліками.

Так, на базі симетричних систем може бути оброблено значний об’єм даних за одиницю часу, тоді як асиметричні системи у цьому суттєво поступаються.

Водночас, оскільки симетрична система, згідно з виразом 1.3, для шифрування та дешифрування використовує єдиний ключ К, у випадку його втрати (або перехоплення зловмисником) на боці прийому шифрограму неможливо буде відкрити, або її буде скомпрометовано.

З іншого боку, асиметрична система, що використовує ключі $К_c$ та $К_d$, не має зазначеного недоліку. У рамках асиметричної системи передбачено такий сценарій роботи з ключами:

- ініціатор обміну даними (приймач) формує пару ключів – відкритий $К_c$, що використовується для шифрування, та секретний $К_d$, на базі якого здійснюється відкриття шифру;
- ключ $К_c$ надсилається передавачеві;
- передавач, отримавши ключ $К_c$, формує шифрограму $М'$, та надсилає її ініціатору обміну:

$$М' = \varepsilon_c(М; К_c); \tag{1.5}$$

- приймач, скориставшись ключем $К_d$, відновлює повідомлення $М$ з шифрограми $М'$:

$$M = \varepsilon_d(M'; K_d). \quad (1.6)$$

Перевага такого способу шифротворення полягає у тому, що ключ K_c може надсилатися відкритим каналом.

Тоді навіть у випадку, коли зловмисник отримає до нього доступ, на його базі він не зможе розшифрувати повідомлення M', оскільки ключ K_d, необхідний для цього, не публікується.

Також, беручи до уваги нижчу швидкість роботи асиметричних систем, для створення захищеного каналу обміну обидва типи систем можуть використовуватися спільно, як показано схемою на рис. 1.2.

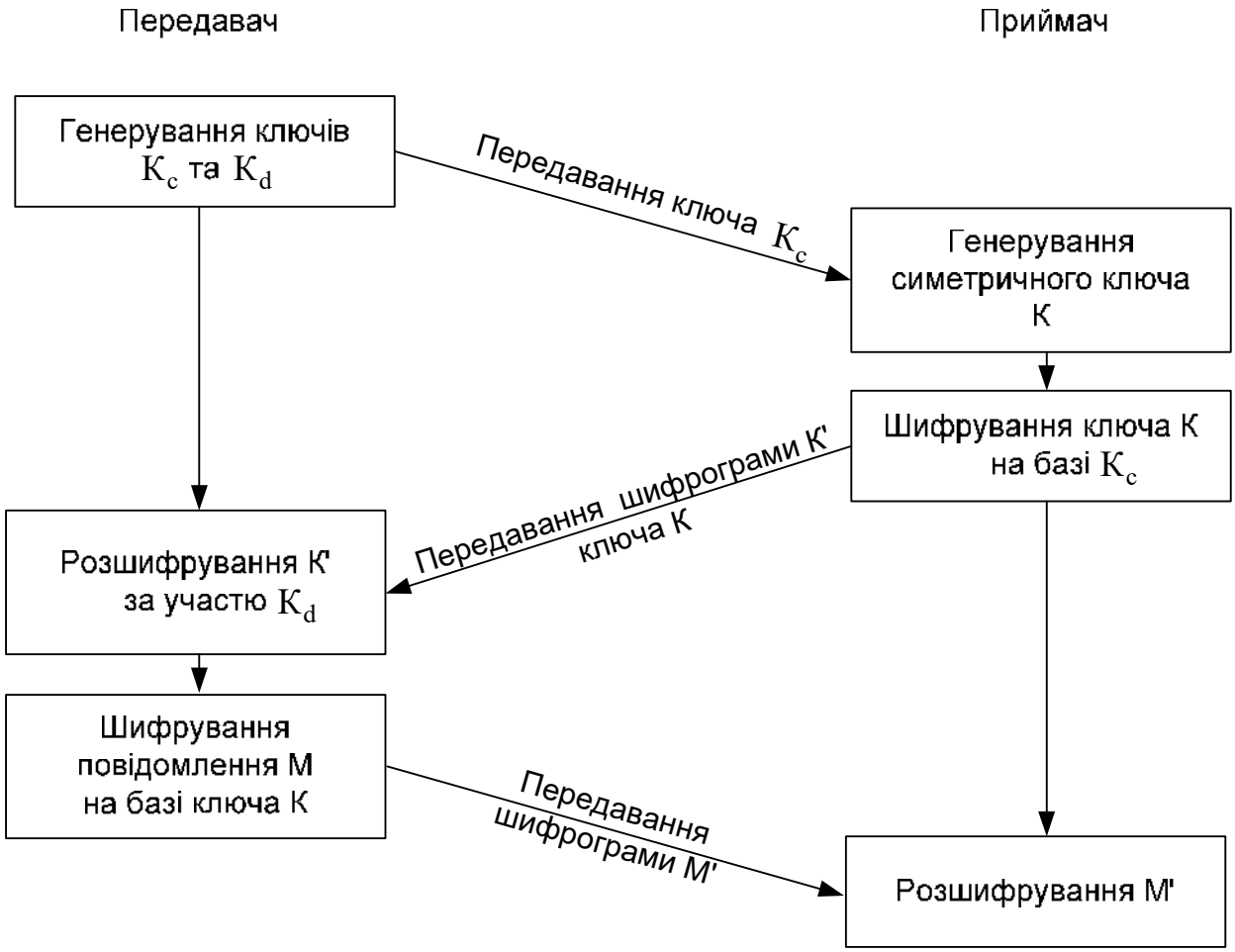

Рисунок 1.2 – Схема спільного використання шифросистем симетричного та асиметричного типів у ході побудови захищеного каналу обміну даними

У зазначений спосіб досягається певний баланс між захищеністю та швидкістю функціонування захищеного каналу.

Далі розглянемо найважливіші характеристики шифросистем, незалежно від способу їх побудови.

1.3 Ключові характеристики шифросистем

Незалежно від того, на обробку яких саме даних орієнтована шифросистема, особливостей архітектури та базового алгоритму, її продуктивність може бути оцінено за рядом базових показників, серед яких наступні:

- криптографічна стійкість R;
- швидкодія t (час шифротворення);
- рівень обчислювального навантаження Ξ.

1.3.1 Криптографічна стійкість

Поняття криптографічної стійкості R має ряд трактувань, а саме:

- може розглядатися як час τ_{dc}, необхідний зловмиснику для того, щоб реконструювати повідомлення M з шифрограми G;

- може вимірюватися показником IPS (instructions per second) – кількість інструкцій за секунду.

Власне, показник IPS тут слід розглядати, як обсяг обчислень, що необхідно виконати зловмиснику для гарантованого дешифрування криптограми. Приклади показників криптографічної стійкості для ряду поширених сьогодні алгоритмів показано табл. 1.1.

У таблиці 1.1. рівень R стійкості подано у MIPS, при цьому $1\,\text{MIPS} = 10^6$ IPS.

Для того, щоб проаналізувати потенційні можливості зламу зазначених шифросистем, розглянемо показники IPS ряду сучасних процесорів, як показано табл. 1.2.

Таблиця 1.1 – Рівні криптографічної стійкості ряду поширених алгоритмів шифрування

Алгоритм	Довжина ключа, біт	Криптографічна стійкість
RSA	1300	$2{,}7\times10^{28}$ MIPS
Ель-Гамаль		$2{,}7\times10^{28}$ MIPS
DES	56 біт	7×10^{16} MIPS

Таблиця 1.2 – Показники швидкодії деяких поширених процесорів

Тип процесору	MIPS/МГц	D-операцій за такт	Рік релізу
ARM Cortex-M0	45 MIPS at 50 MHz	0,9	2009
ARM Cortex-R4	450 MIPS at 270 MHz	1,66	2006
ARM Cortex A5	1 256 MIPS at 800 MHz	1,57	2011
ARM Cortex A7	2 850 MIPS at 1.5 GHz	1,9	2011
Intel Atom N270 (Single core)	3 846 MIPS at 1,6 GHz	2,4	2008
Raspberry Pi 2	4 744 MIPS at 1 GHz	4,744	2014
ARM Cortex-A9 (Dual core)	7 500 MIPS at 1,5 GHz	5,0	2009
AMD E-350 (Dual core)	10 000 MIPS at 1,6 GHz	6,25	2011
PS3 Cell BE (PPE only)	10 240 MIPS at 3,2 GHz	3,2	2006

Як показує спільний аналіз таблиць 1.1 та 1.2, приймаючи до уваги рівні стійкості алгоритмів шифрування та наявні показники швидкодії процесорів, злам зазначених шифросистем у неспеціалізованому технологічному базисі на сьогодні являє собою майже не вирішуване завдання.

Водночас, зловмисник може суттєвим чином підвищити обчислювальний потенціал, зокрема, скориставшись для цього:

- хмарними обчисленнями;
- побудовою brutforce-засобів на базі паралельних обчислень;
- використавши можливості технології CUDA.

Тобто, попри теперішній рівень стійкості, ряд шифросистем у майбутньому можуть бути зламані, а відтак – потенційно потребують модифікації.

1.3.2 Швидкодія алгоритму

Показник швидкодії t алгоритму визначає можливість обробки даних у реальному часі – VoIP, VoD, AoD, відеоконференцій тощо.

У загальному випадку, якщо говорити про дані, які підлягають шифруванню без урахування їхньої типоналежності, за показником швидкодії шифросистема має відповідати наступній вимозі:

$$t \to \min \tag{1.7}$$

При цьому, якщо мова йде про обробку даних саме реального часу – вираз (1.7) буде доповнено до наступного вигляду:

$$\begin{cases} t_{enc}, t_{dec} \to \min; \\ t_{enc} \approx t_{dec}, \end{cases} \tag{1.8}$$

де t_{enc} та t_{dec} - час шифрування та дешифрування відповідно.

Як видно з виразу (1.8), час шифрування та дешифрування для випадків обробки даних реального часу не повинен суттєво різнитися.

Показники швидкодії деяких широко застосовуваних алгоритмів шифрування показано у табл. 1.3.

Таблиця 1.3 – Показники швидкодії ряду поширених алгоритмів шифрування

Алгоритм	Швидкодія, Кбіт/с
RSA	1
Ель-Гамаль	≈ 1
DES	≤ 800

При цьому, для асиметричних алгоритмів, до яких, зокрема, належать RSA та шифр Ель-Гамаля, вимога щодо наближеної рівності часу шифрування t_{enc} та дешифрування t_{dec} не виконується.

Так, для алгоритму RSA швидкість шифротворення за умови, що на опис ключа витрачається 8 біт та за умови шифрування 1024-бітного відрізка, складає 0,08 с, тоді як дешифрування потребує 0.93 с за тієї ж обчислювальної потужності.

Для шифру Ель-Гамаля часові показники є приблизно такими ж.

З іншого боку, швидкість шифрування/дешифрування DES приблизно на 2 порядки перевищує швидкість шифротворення як Ель-Гамаля, так і RSA. Зазначимо при цьому, що для DES, як симетричної системи, вимога $t_{enc} \approx t_{dec}$ виконується повністю.

1.3.1 Обчислювальне навантаження

Рівень обчислювального навантаження Ξ, що створює шифросистема на апаратну платформу, визначає можливість обробки:

- втликих масивів даних з мінімальною затримкою;
- даних реального часу.

У цілому, показник Ξ може вважатися величиною, аналогічною до часу t шифротворення, за виключенням того, що:

- значення Ξ залежить від кількості елементарних операцій, необхідних для шифрування умовно 1 біту вихідних даних;
- показник обчислювального навантаження визначає можливість реалізації алгоритму у межах апаратних платформ з обмеженими обчислювальними можливостями.

При цьому, очевидно, що серед розглянутих шифросистем найбільшою швидкодією характеризується DES.

Дане твердження потребує додаткових пояснень.

Зокрема, на формування шифру DES необхідно задіяти таку кількість операцій:

$$N_{round} = 16\varepsilon + 2g + 32\rho, \qquad (1.9)$$

де g - кількість операцій XOR;

ε - кількість операцій табличної підстановки (S-блоки);

ρ - кількість операцій зсуву.

Для повного шифру, відповідно, маємо:

$$N_{des} = 2 \times 64 \times \rho + 16 \times (16 \times \varepsilon + 2 \times g + 32 \times \rho) = 640 \times \rho + 256 \times \varepsilon + 32 \times g \qquad (1.10)$$

Як видно з виразів (1.9) та (1.10), кількість арифметичних операцій для побудови шифру DES визначаться його архітектурою та обсягом даних, що підлягають шифруванню, та може бути попередньо розраховано для резервування ресурсів.

Таку ж кількість операцій буде витрачено і на процедуру розшифрування.

На відміну від DES, для розшифрування RSA та/або шифру Ель-Гамаля може бути витрачено на 2-3 порядки більше обчислювальних операцій, ніж для зашифрування.

1.4 Огляд умов шифротворення

Усі дані, які потребують захисту з використанням шифрування, можемо віднести до:

- даних, що необхідно шифрувати у локальному сховищі;
- даних, які потребують шифрування одночасно з надсиланням їх до мережі.

У першому випадку тип шифросистеми, що може бути використано для закриття даних, та її характеристики, такі, як – швидкість шифротворення та обчислювальне навантаження, не є критичними.

Водночас для даних, які потребують шифрування одночасно з надсиланням їх до мережі, має забезпечуватися мінімізація часу шифрування, як показано виразом (1.7).

При цьому, якщо розглядається випадок обробки та передавання трафіку інтерактивного типу чи трафіку, що належить тому чи іншому потоковому сервісу, вимоги, подані системою (1.8), буде уточнено.

Так, для випадку трафіку інтерактивного типу (наприклад, відеоконференція) час шифротворення формально має наближатися до 0. Разом з тим, очевидно, що виконання даної вимоги в існуючій технологічній парадигмі не може бути забезпечено. Тому реальні вимоги до часу шифротворення можуть бути подані у наступному вигляді:

$$\begin{cases} t_{enc}, t_{dec} \to \Delta\tau_{int}; \\ t_{enc} \approx t_{dec}, \end{cases} \tag{1.11}$$

де $\Delta\tau_{int}$ - деяка обмежено мала величина, значення якої регламентується вимогами QoS, до затримки передавання для конкретного типу інтерактивного трафіку у відповідності з наступним принципом:

$$\Delta\tau \geq \Delta\tau_{int} + \Delta\tau_{hnd} + \Delta\tau_{tr}, \tag{1.12}$$

де $\Delta\tau_{hnd}$ - час обробки пакету на кінцевих вузлах;

$\Delta\tau_{tr}$ - час розповсюдження пакету мережею.

У свою чергу, для трафіку потокових сервісів вираз (1.8), аналогічно до випадку трафіку інтерактивного типу, може бути переписано наступним чином:

$$\begin{cases} t_{enc}, t_{dec} \to \Delta\tau_{str}; \\ t_{enc} \approx t_{dec}, \end{cases} \tag{1.13}$$

де $\Delta\tau_{str}$ - деяка обмежено мала величина, значення якої регламентується вимогами QoS, до затримки передавання для конкретного типу трафіку. Величина $\Delta\tau_{str}$, у свою чергу, обмежується за принципом аналогічним поданому виразом (1.12).

Зрозуміло також, що справедливим є наступне співвідношення:

$$\Delta\tau_{str} > \Delta\tau_{int}. \tag{1.14}$$

1.5 Попередні висновки

Підводячи підсумки огляду загальних умов шифротворення, можемо зазначити наступне:

- для даних, які не підлягають передаванню мережею, час шифрування та обчислювальне навантаження, яке чинить шифросистема, можуть бути несуттєвими;

- вимоги щодо допустимого часу шифрування даних, що передаються, визначаються для кожного типу трафіку окремо, виходячи з базових вимог QoS до затримки;

- очевидно, що найбільший час шифрування відповідає трафіку «важких» типів;

- шифрування великих обсягів даних, у т.ч. «важкого» трафіку, до якого належать також мультимедійні дані, (відео та аудіо) на базі асиметричних алгоритмів недоцільне з огляду недостатньої швидкодії;

- на випадок необхідності шифрування мультимедійного контенту асиметричні шифросистеми може бути використано виключно для створення захищеного каналу обміну ключами симетричного шифрування.

Отже, приймаючи до уваги усе вищезазначене, а також вимоги технічного завдання, далі пропонується:

- дослідити ряд алгоритмів шифрування симетричного типу для виявлення найбільш перспективного з точки зору можливості його модифікації;
- проаналізувати деякі асиметричні алгоритми для визначення більш стійкого з них для використання у якості засобу шифрування симетричних ключів.

2. ДОСЛІДЖЕННЯ СИМЕТРИЧНОЇ ТА АСИМЕТРИЧНОЇ АРХІТЕКТУР ШИФРУВАННЯ

Приймаючи до уваги необхідність прив’язки до існуючих криптографічних систем, розглянемо закономірності побудови асиметричних та симетричних шифрів. Відтак, має сенс дослідити архітектуру блокових шифрів, як таких, що сьогодні мають широке застосування, та є характерними представниками шифрів симетричного типу.

При цьому, слід брати до уваги те, що асиметричний шифр не є одностайно потоковим, як антагоніст блоковому, так як за замовчуванням асинхронний шифр обробляє послідовність символів фіксованого розміру, яка, частіше за все, дорівнює довжині ключа. І лише у випадку, коли розмір послідовності символів, які обробляються у ході однієї операції асиметричного шифрування, може сягати довжини вихідного повідомлення, такий шифр можна вважати потоковим. У зв’язку з цим, виконаємо дослідження архітектури симетричного типу на прикладі блокового шифрування.

2.1 Блокові шифри, як частковий випадок шифрів симетричної архітектури

Ключовою ознакою симетричного шифру, як зазначалося у розділі 1 системою виразів (1.3), є рівність ключів шифрування та дешифрування.

Свого часу композиційні блокові шифри, як високоефективний та перспективний засіб забезпечення конфіденційності повідомлень у системах секретного зв'язку, було запропоновано Клодом Шенноном.

Для того, щоб мати можливість дослідити принцип функціонування криптографічних засобів на їх базі, спершу розглянемо закономырносты побудови блокових шифрів.

Блоковий шифр у загальному вигляді являє собою особливу криптосистему, у рамках якої відносно вихідного повідомлення передбачено виконання таких дій, як:

- поділ відкритого повідомлення на окремі блоки (зазвичай - однакового розміру);

- незалежна обробка кожного з блоків у базисі того чи іншого алгоритму шифрування, у наслідок чого отримується послідовність блоків шифрованого повідомлення.

У найпростішому вигляді криптосистема блокового типу може бути представлена наступною схемою (рис.2.1).

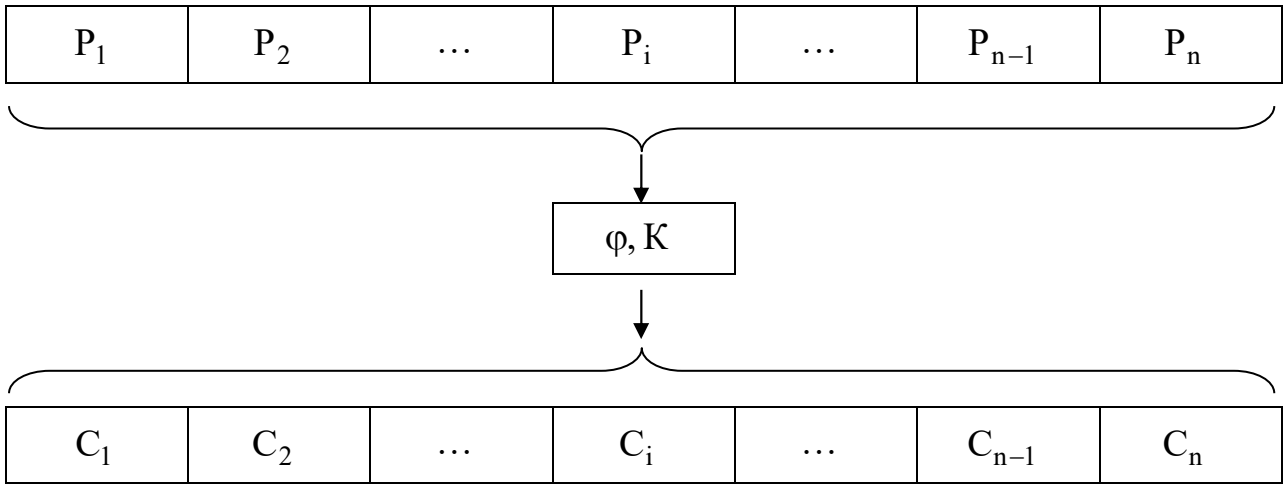

Рисунок 2.1 – Схема найпростішої блочної криптосистеми

На рис.2.1 вихідне повідомлення M розглядається у вигляді конкатенації n блоків, кожен з яких містить послідовність з P біт, тобто:

$$M = P_1 \,\&\, P_2 \,\&\, ... \,\&\, P_i \,\&\, ... \,\&\, (P_n + p'), \qquad (2.1)$$

де p' - біти розширення, які додаються до P_n-го блоку; у свою чергу, $p' = |P_n - P_i|$. Тобто, очевидно, що $|P_n - P_i| = 0$ не задіюються. У підсумку цього, створюються умови для виконання наступної вимоги:

$$P_1 = P_2 = ... = P_i = ... = P_n. \qquad (2.2)$$

При цьому, шифроване повідомлення G, за аналогією з виразом (1), являє собою конкатенацію окремих шифрованих блоків C_i, а саме:

$$G = C_1 \,\&\, C_2 \,\&\, ... \,\&\, C_i \,\&\, ... \,\&\, C_n. \qquad (2.3)$$

Таким чином, виходячи з усього зазначеного, деякий детермінований блоковий шифр G може бути описано у наступний спосіб:

$$G = \varphi(P; C; K), \tag{2.4}$$

де P – множина вхідних значень;

C – множина вихідних значень;

K – простір ключів;

φ – функція шифрування.

Тобто, з виразу (2.4) функцію шифрування можемо представити, як:

$$\varphi: P \times K \to C. \tag{2.5}$$

Тепер розглянемо випадок композиційного шифру. У цьому разі блоковий шифр визначається сімейством перетворень G, які, у свою чергу, мають спільні множини вхідних та вихідних значень, тобто, $P_i = C_i = \Theta$, при цьому, результат виконання функції шифрування φ_i залежить лише від значення ключового елемента $k_i \in K$.

У базисі даного сімейства перетворень, використовуючи операцію композиції, може бути створено шифр, який описується наступним відображенням:

$$\varphi: \Theta \times (K_1 \times K_2 \times ... \times K_r) \to \Theta, \tag{2.6}$$

де:

$$\varphi := \varphi_r \times \varphi_{r-1} \times ... \times \varphi_2 \times \varphi_1, \tag{2.7}$$

а ключем є вектор наступного вигляду:

$$(k_1, k_2, \ ...k_r) \in K_1 \times K_2 \times ... \times K_r. \tag{2.8}$$

У цьому випадку перетворення φ_i є і-м раундом шифрування. При цьому, ключ k_i – і-м раундовим ключем.

Ідея, що лежить в основі складових (або композиційних) блокових шифрів, полягає у побудові криптостійкої системи за допомогою багаторазового застосування відносно простих криптографічних перетворень.

На сьогодні такими перетвореннями є:

- підстановки (substitution, або S-операції);
- перестановки (permutation, або P-операції).

Відтак криптографічні схеми, що базуються на зазначених перетвореннях, називаються SP-мережами.

Зазначимо, що у деяких випадках раундові ключі формуються на базі єдиного секретного ключа за допомогою алгоритму вироблення раундових ключів (при цьому розмір ключа системи істотно менший від сумарного розміру всіх раундових ключів).

Якщо ключові простори K та перетворення φ_i для всіх раундів збігаються, то такий складовий шифр називається *ітераційним*, що є композицією однієї і тієї ж криптографічної функції, що використовується з різними ключами (рис. 2.1).

У свою чергу, сенс багаторазового виконання S- та P-перетворень полягає у створенні умов для надання підсумковій шифрограмі двох властивостей, притаманних стійким шифрам, а саме:

- властивості дифузії;
- конфузії.

При цьому, дифузія (рис.2.2) передбачає поширення впливу одного знака відкритого тексту на значну кількість знаків шифротексту.

Така властивість шифрограми надає ряд істотних переваг, серед яких найбільш істотними є наступні:

- дає можливість усунення статистичної залежність між символами вихідного тексту, іншими словами, дозволяє перерозподілити надлишковість вихідних даних у вигляді поширення її на усю множину символів тексту;
- нівелює ймовірність відновлення невідомого ключа частинами.

стійкій криптосистемі повинні бути притаманними обидві властивості.

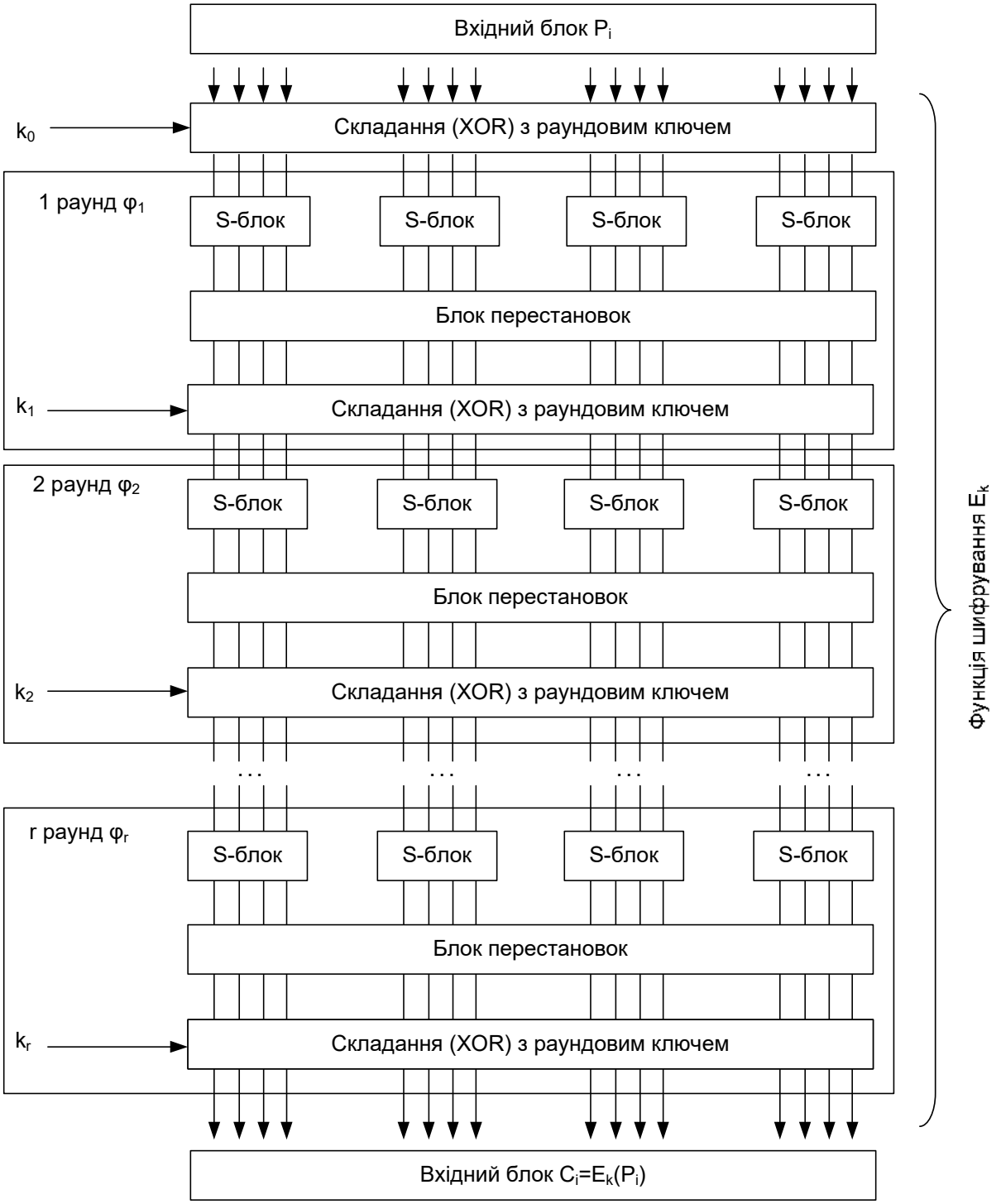

Рисунок 2.1 – Схема найпростішого ітераційного шифру

Водночас, конфузія дозволяє досягти якомога складнішої залежності між ключем і підсумковою шифрограмою. Тобто, при цьому використання статистичного аналізу перемішаного тексту не повинно забезпечувати отримання скільки-небудь значної кількості інформації щодо використаного ключа.

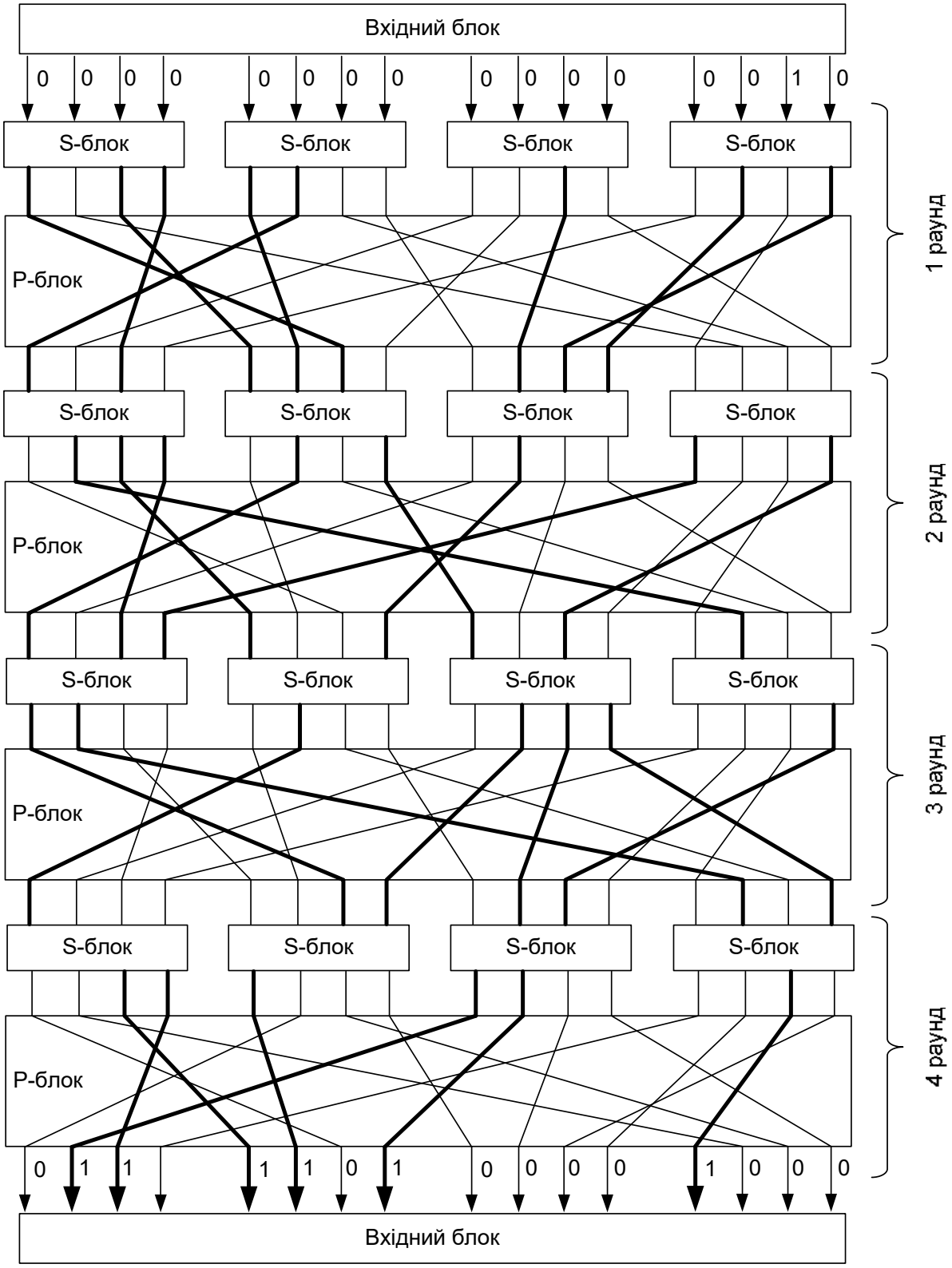

Рисунок 2.2 – Дифузія та конфузія у SP-мережі

Слід також брати до уваги, що у випадку забезпечення для шифру лише однієї з властивостей - дифузії або конфузії - не гарантується досягнення необхідного рівня стійкості.

Разом з тим, якщо для тієї чи іншої шифросистеми забезпечується наявність властивостей дифузії та конфузії, це створює умови для виникнення т.з. *лавинного ефекту* (avalanche).

Даний ефект полягає у тому, що зміна лише одного біта вихідного повідомлення M чи ключа K веде до зміни значної кількості символів підсумкової шифрограми G.

Відповідно, чим більш вираженим є лавинний ефект, тим вища надійність шифру.

Як свідчить результат аналізу рис. 2.1 та рис.2.2, на рівень Q(G) підсумкової стійкості утвореної шифрограми можуть здійснювати вплив такі фактори, як:

- алгоритми, утілені у межах S- та P-блоків;
- кількість раундів шифрування;
- матеріал раундового ключа k_i.

Аналітично таку залежність між показником Q(G) стійкості та факторами впливу можна подати у наступному вигляді:

$$Q(G) \propto abs(k_i) \,\&\, d(k_i) \,\&\, r \,\&\, d(\phi(S); \phi(P)), \qquad (2.9)$$

де $abs(k_i)$ - абсолютна довжина раундового ключа;

$d(k_i)$ - рівень складності раундового ключа; отже, тут має суттєве значення функція f формування раундового ключа k_i з секретного ключа K, що формально описується виразом:

$$k_i = f(K), \qquad (2.10)$$

r - кількість раундів шифрування;

$d(\phi(S); \phi(P))$ - складність алгоритмів підстановки та перестановки відповідно.

Відтак, маніпулюючи означеними параметрами, може бути забезпечено той чи інший рівень захищеності результуючого шифру.

На сьогодні одним з найбільш вдалих утілень SP-мережі, що забезпечує високий рівень захищеності шифрограми, є мережа Фейстеля.

2.1.1 Мережа Фейстеля

При цьому, мережею Фейстеля формально є специфічний метод перетворень вихідного масиву P символів повідомлення, у якому значення, обчислене з однієї частини масиву, накладається інші частини. Зазвичай мережа Фейстеля має структуру сформовану таким чином, що виконання операцій шифрування та дешифрування застосовується один і той же алгоритм. При цьому, відмінність полягає виключно у порядку застосування даних ключа K.

Розглянемо класичну схему мережі Фейстеля, як показано рис.2.3.

У рамках структури Фейстеля незалежні потоки даних, породжені вихідним блоком P_i, називаються *гілками мережі*.

У свою чергу, величини V_i являють собою *параметри мережі*, зазвичай це функції від матеріалу ключа (похідна від ключа).

Залежно від особливостей реалізації процедури творення раундових ключів, у ході опису функціонування алгоритму можуть застосовуватися як величини V_i, так і, безпосередньо, самі раундові ключі k_i. У загальному випадку, коли відсутня детальна специфікація алгоритму, зазначені величини можуть використовуватися рівнозначно.

При цьому, функція Ф зветься у*творюючою*.

Водночас, операція, що складається з одноразового обчислення утворюючої функції та подальшого накладення її результату на іншу гілку з обміном їх місцями, являє собою *цикл* або *раунд* мережі Фейстеля.

У класичній реалізації мережі Фейстеля на вхід алгоритму шифрування надходить блок відкритого повідомлення, що має довжину 64 біта і ключ K.

Далі виконується поділ блоку P_i відкритого повідомлення на дві рівні частини P_i і P_i, які послідовно проходять через 16 раундів обробки, після чого знову об'єднуються для отримання блоку шифрованого тексту відповідної довжини.

Для кожного окремого раунду, а також для усього алгоритму у цілому, початковими даними є:

- суб-блоки $P_i^{(1)}$ і $P_i^{(2)}$, які було отримано на виході попереднього раунду;

- величина V_i , що обчислюється за загальним секретним ключем К; зазвичай, для збільшення стійкості шифрограми має виконуватися наступна умова:

$$V_1 \neq V_2 \neq ... \neq V_i \neq ... \neq V_r, \qquad (2.11)$$

тобто, усі раундові ключі повинні відрізняються як від загального ключа К, так і один від одного.

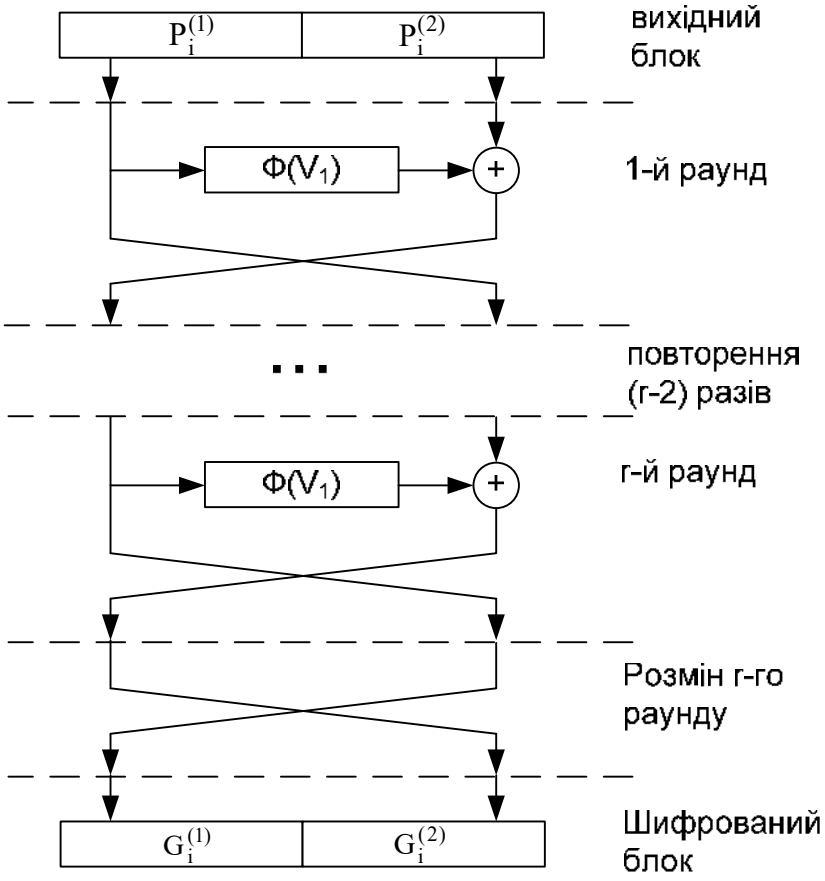

Рисунок 2.3 – Структура класичної мережі Фейстеля

Виконання усіх раундів шифрування здійснюється за тією ж схемою. Спочатку для правої половини блоку даних, тобто, $P_i^{(2)}$, виконується операція підстановки. Дана операція полягає у застосуванні до лівої половини блоку даних ($P_i^{(1)}$) деякої функції раунду r та подальшому складання отриманого результату з правої половиною блоку даних за допомогою операції XOR.

Для всіх раундів функція раунду має одну й ту саму структуру, але залежить від параметра V_i (раундового ключа). Після підстановки виконується перестановка, яка є обміном місцями двох половин блоку даних.

Власне, процес дешифрування криптограми, отриманої на базі шифру, що наслідує структуру Фейстеля, принципово не відрізняється від процесу шифрування. У цьому випадку використовується той же самий алгоритм, але на вхід подаються шифровані дані, а параметри V_i (раундові ключі) використовуються у зворотній послідовності.

Так, для першого раунду застосовується ключ V_r, для другого - V_{r-1} і так далі доти, поки не буде задіяно ключ V_r для останнього раунду. Така властивість даної схеми шифротворення є досить зручною, оскільки для дешифрування криптограм немає потреби використання іншого алгоритму, що є відмінним від алгоритму шифрування.

2.2 Асиметричні шифросистеми

Порівняно з симетричними шифрами, асиметричні характеризуються структурно є суттєво простішими, так як головний інструментарій закриття вихідної інформації з використанням секретного ключа базується на складності виконання тих чи інших математичних обчислень.

Загальна структура асиметричної криптографічної системи (рис.2.4) у цілому наслідує схему 1.1 з деякими доповненнями.

У системах такого типу функціонал шифрування/дешифрування базується, у сутності, на певній математичній операції, що являє собою одностороннюю функцію – тобто, функцію вигляду $M' = f(M)$, яку може бути легко обчислено для будь-яких M. При цьому, зворотнє завдання – а саме, реконструкція M за M', потребує значних часових та обчислювальних ресурсів.

Прикладом такої односторонньої функції f може слугувати функція двох аргументів $f(x;y) = xy$, яка є добутком пари простых чисел x та y. Навіть за умови великих значень x та y, її обчислення не викликає труднощів. Попри це, операція факторизації – тобто, підбору вихідної пари простих чисел з їх добутку, є досить складним завданням.

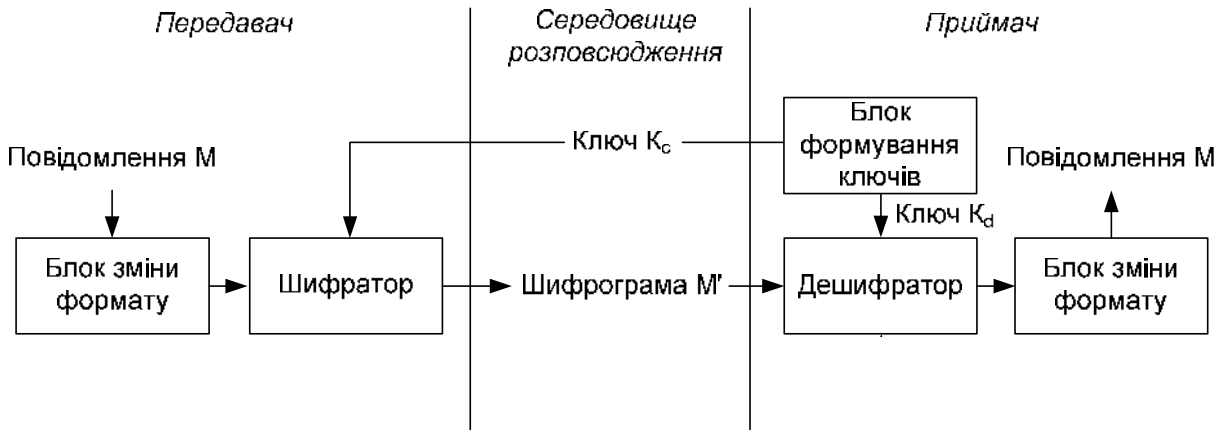

Рисунок 2.4 – Спрощена структура асиметричної шифросистеми

Для систем даного типу підсумкова стійкість $Q(G)$ отриманого шифру, судячи зі схеми, може залежати від односторонньої функції f, яка реалізує сам алгоритм, та особливостей ключа K_c шифрування.

Разом з тим, сам алгоритм f перетворення, по-перше, вважається достатньо надійним, по-друге, не передбачає наявності опцій, що теоретично можуть впливати на зміну рівня $Q(G)$. Тобто, можливість маніпулювання підсумковим значенням $Q(G)$ досягається виключно за рахунок зміни довжини $abc(K_c)$ відкритого ключа та самого матерыалу ключа. Формально таку залежність можна подати у вигляді:

$$Q(G) \propto abs(K_c) \,\&\, d(K_c), \qquad (2.12)$$

де $abs(K_c)$ - абсолютна довжина ключа шифрування;

$d(K_c)$ - рівень складності ключа шифрування.

При цьому, ключ K_c (та далі - відповідний йому K_d) генерується за певними алгоритмами, ураховуючи ряд обмежень, що знижують потенційну стійкість $Q(G)$. У сутності, тут користувач може встановлювати єдину опцію – а саме, $abs(K_c)$. Зрозуміло, що такий підхід до шифротворення, коли ключ генерується системою, у свою чергу, додатково вимагає часових витрат, які збільшуються відповідно до збільшення довжини $abs(K_c)$ ключа. Ця обставина обмежує можливості застосування асиметричних схем для шифрування у реальному часі, а також для обробки великих масивів даних.

2.3 Висновки за розділом

Досліджено принципи побудови та загальну архітектуру симетричних та асиметричних шифросистем. Зокрема:

- розглянуто загальні засади та специфіку побудови блокових композиційних шифрів;
- виконано огляд відмінностей між асиметричними та симетричними шифросистемами;
- виявлено показники ефективності шифросистеми, а саме – те, що алгоритмам шифрування мають бути притаманні властивості дифузії та конфузії.

За результатами виконаних досліджень визначено, що:

- асиметричні шифросистеми, попри характерні для них переваги, поступаються симетричним швидкістю шифротворення а відтак – не можуть застосовуватися для захисту великих об’ємів даних (якщо шифрування і передача даних є операціями, поєднаними у єдиному технологічному циклу) і трафіку потокового та інтерактивного типів;
- в основі значної кількості симетричних блокових шифросистем лежить структура Фейстеля – зокрема, це справедливо для криптографічної системи DES;
- використання структура Фейстеля, як основу побудови криптосистем, є перспективним базисом для створення алгоритмів шифрування.

Далі виконаємо аналіз поширених шифрів архітектур обох типів для того, щоб визначити найбільш перспективні з них з позиції доцільності використання для випадку обробки мультимедійних даних.

3. ДОСЛІДЖЕННЯ ІСНУЮЧИХ КРИПТОСИСТЕМ

3.1 Поширені симетричні криптосистеми

3.1.1 Криптографічна система AES

Шифросистема AES являє собою приклад симетричного криптографічного алгоритму, який на сьогоднішній час є одним з поширених.

При цьому, нерідко AES використовується як самодостатній засіб, та паралельно виступає у якості однієї з компонент комплексної системи криптографічного захисту даних.

AES відноситься до шифросистем блочного типу. Це означає, що обробка даних за його участю здійснюється на рівні блоків деякої фіксованої розмірності, а саме - 128-бітих блоків.

До того ж, кожен з блоків, що підлягають обробці на базі системи AES, розглядаються у вигляді сукупності чарунок даних з розмірністю 8 біт (1 байт).

Також слід зазначити, що AES використовує ключі фіксованої розмірності. Так, залежно від режиму шифрування передбачається можливість ключів різних довжин, а саме:

- 128 біт - для шифрування у базовому режимі (даний режим умовно відповідає випадку обробки даних з грифом «для службового користування»);
- 192 та 256 біт – для побудови шифрограм повідомлень, що відносяться до категорій «таємно» та «цілком таємно».

У будь якому випадку, незалежно від даних, що оброблюються, та обраних ключів, процесу шифротворення є характерним наступне:

- побудова шифрограми реалізується на рівні байт кожного з блоків, що шифруються;
- обробка виконується на рівні як рядків, так і стовпців кожного блоку.

Для AES також характерною є залежність між довжиною ключа та кількістю η_r раундів шифрування.

Таку залежність відображено у табл. 3.1.

Таблиця 3.1 – Залежність підсумкової кількості раундів шифрування алгоритму AES виходячи з довжини секретного ключа

Розмірність ключа, біт	η_r
128	10
192	12
256	14

При цьому, не зважаючи на те, ключ якої довжини було застосовано для шифротворення, до складу кожного раунду шифрування входить аналогічний набір технологічних етапів обробки вихідних даних, як зображено рис. 3.1.

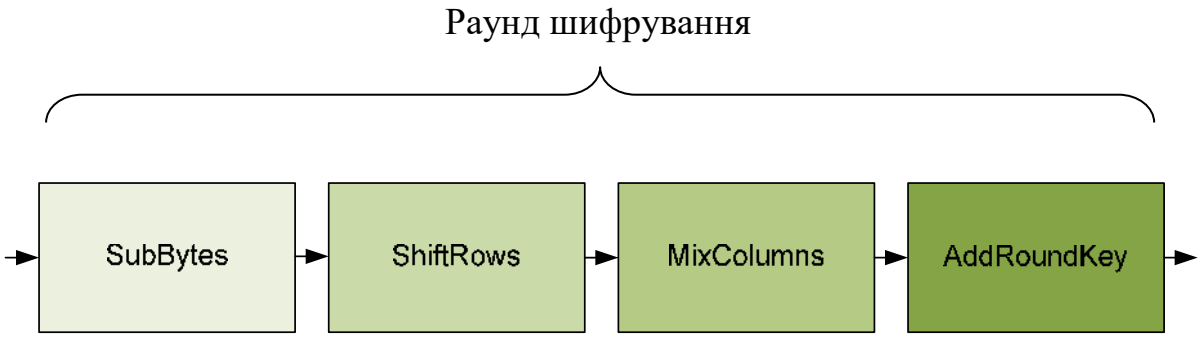

Рисунок 3.1 – Технологічні кроки обробки вихідних даних, які реалізуються у ході одного раунду шифрування на базі AES

Керуючись рисунком 3.1, виконаємо стислий огляд технологічних етапів, які входять до одного раунду шифрування AES.

1. SubBytes.

Протягом даного етапу реалізується таблична заміна кожного з 16 байт, що разом утворюють вихідний блок даних. Процедура заміни тут виконується поелементно.

2. ShiftRows.

Під час виконання зазначеного етапу обробки даних алгоритм здійснює обробку рядків блоку після процедури SubBytes. Дані рядки, при цьому, містять у собі проміжні дані криптоутворення, та являють собою двовимірні масиви, елементами яких є байти.

3. Етап MixColumns.

У ході даного етапу обробки кожен стовпець масиву, що утворилися за результатом виконання етапу ShiftRows, підлягають процедурі перемноження на поліном $a(x)$. Зазначений поліном, у свою чергу, може бути описаний наступним виразом:

$$a(x) = 3x^3 + x^2 + x + 2. \qquad (3.1)$$

Важливо зазначити, що у цьому випадку, множення виконується за модулем $(x^4 + 1)$.

4. AddRoundKey.

Даний етап обробки є фінальним на рівні раунду шифрування. Під час його реалізації з масивом байт, що були утворені у передуючому етапі, а саме - MixColumns, складається секретний ключ W_{4r+i}. Такий ключ, у сутності, є послідовністю довільного змісту що може мати довжини, попередньо представлені у табл.2.1. У цьому разі ідентифікатор r являє собою лічильник поточного раунду обробки (шифротворення).

На сьогодні шифросистема AES розглядається як достатньо стійка і як наслідок – на її базі зараз виконується шифрування значної частини урядових документів США. Разом з тим, AES не забезпечує абсолютну стійкість. Причиною цьому є присутність ряду потенційних уразливостей критичного характеру. Такі потенційні вразливості було виявлено протягом останнього десятиліття. Зокрема, до даних вразливостей доцільно віднести те, що:

- шифросистеми, що мають в основі архітектури схему AES, є, за великим рахунком, потенційно нестійкими до атак XSL-типів;
- доведено, що підходи до зламу AES, що базуються на використанні атак за другорядними параметрами (наприклад – атак за часом), є перспективними.

3.1.2 Шифросистема DES

Алгоритм DES (Data Encryption Standard) було розроблено у 1977 р. Розробником шифросистеми є Національним бюро Стандартів США (NBS). При цьому, алгоритм DES було рекомендовано для використання у державних та урядових закладах США для побудови захисту від

неавторизованого доступу до інформації, що має гриф не вище, ніж для службового користування.

Пізніше, у 1980 році стандарт було схвалено ANSI, що перетворювало DES не лише за назвою, але і фактично.

На сьогоднішній час DES є найбільш розповсюдженою криптосистемою симетричного типу, що знайшла застосування у системах захисту даних комерційного спрямування. Більш того - використання алгоритму DES для захисту даних зараз може вважатися ознакою «гарного тону».

Так, наприклад, програмний модуль Diskreet, що входить до пакету Norton Utilities, та застосовується для створення шифрованих розділів на жорсткому диску, базується саме на використанні алгоритму DES. При цьому, так званий "власний алгоритм шифрування" має відмінності з DES лише стосовно кількості раундів шифрування.

У цілому, така поширеність алгоритму DES пояснюється тим, що:

- алгоритм характеризується достатньо високим рівнем стійкості;
- забезпечується висока швидкість шифротворення за рахунок відносної простоти алгоритму;
- використовується єдиний ключ, що має довжину у 56 біт;
- для повідомлення, зашифрованого з використанням одного ПЗ, може використовуватися будь-яке інше ПЗ (що підтримує DES) для розшифрування.

Загальна схема шифрування на базі алгоритму показана на рис. 3.2.

Процедура дешифрування у DES зводиться до повторення операцій циклу шифрування у зворотній послідовності.

У свою чергу, цикл шифрування включає у себе виконання:

- початкової перестановки біт 64-бітового блоку;
- 16 раундів шифрования;
- зворотної перестановки біт.

Таблиці перестановок, які використовуються, є стандартними та незмінними.

Усі перестановки та операції у таблицях розробниками підобрано таким чином, щоб процес розшифрування, що реалізується на базі механізмів підбору ключа, було максимально утруднено.

Структурна схема алгоритму ілюструється рис.3.3.

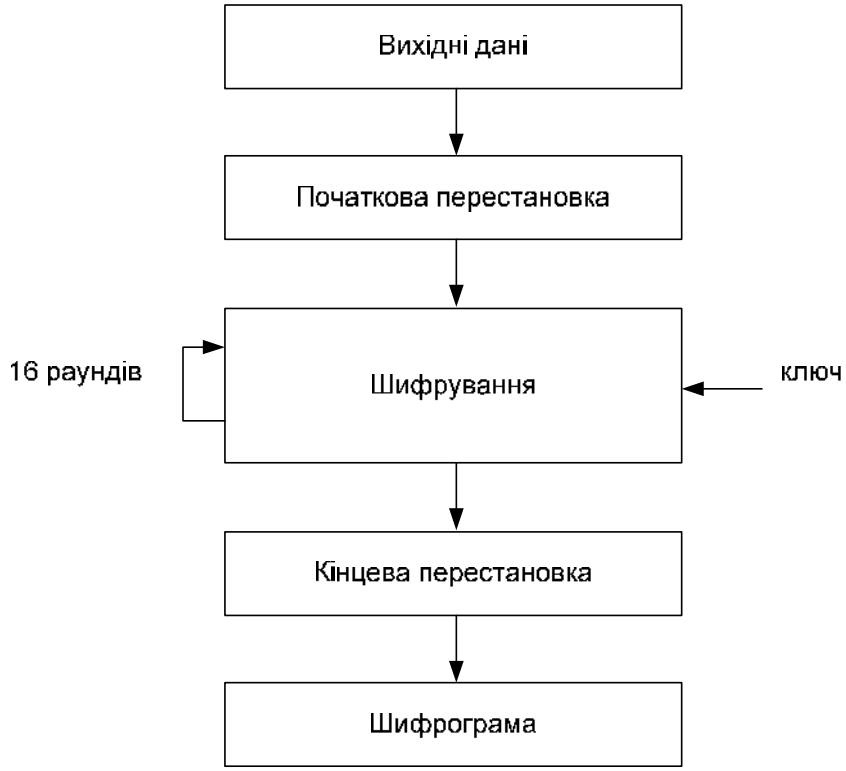

Рисунок 3.2 – Загальна схема шифрування на базі алгоритму DES

Отже, вихідне повідомлення M розглядається як конкатенація з N 8-байтових блоків T, тобто:

$$M = T_1 \,\&\, T_2 \,\&\, \ldots \,\&\, T_i \,\&\, \ldots \,\&\, T_N . \qquad (3.2)$$

Зчитування блоків T при цьому здійснюється послідовно.

Кожен черговий блок на першому етапі обробки зазнає перетворення на базі матриці початкової перестановки IP, як показано таблицею 3.2.

Дане перетворення реалізується наступним чином: так, біт 58 з блоку T перетворюється на біт 1.

У свою чергу, біт 50 стає бітом 2 і т.д. У результаті отримується перетворення $T(0) = IP(T)$.

Далі, одержана таким чином послідовність біт T(0) підлягає розподілу на дві окремі суб-послідовності, кожна довжиною по 32 біта, а саме - L(0) – послідовність лівих, або старших біт, та R(0) – послідовність правих, або молодших біт.

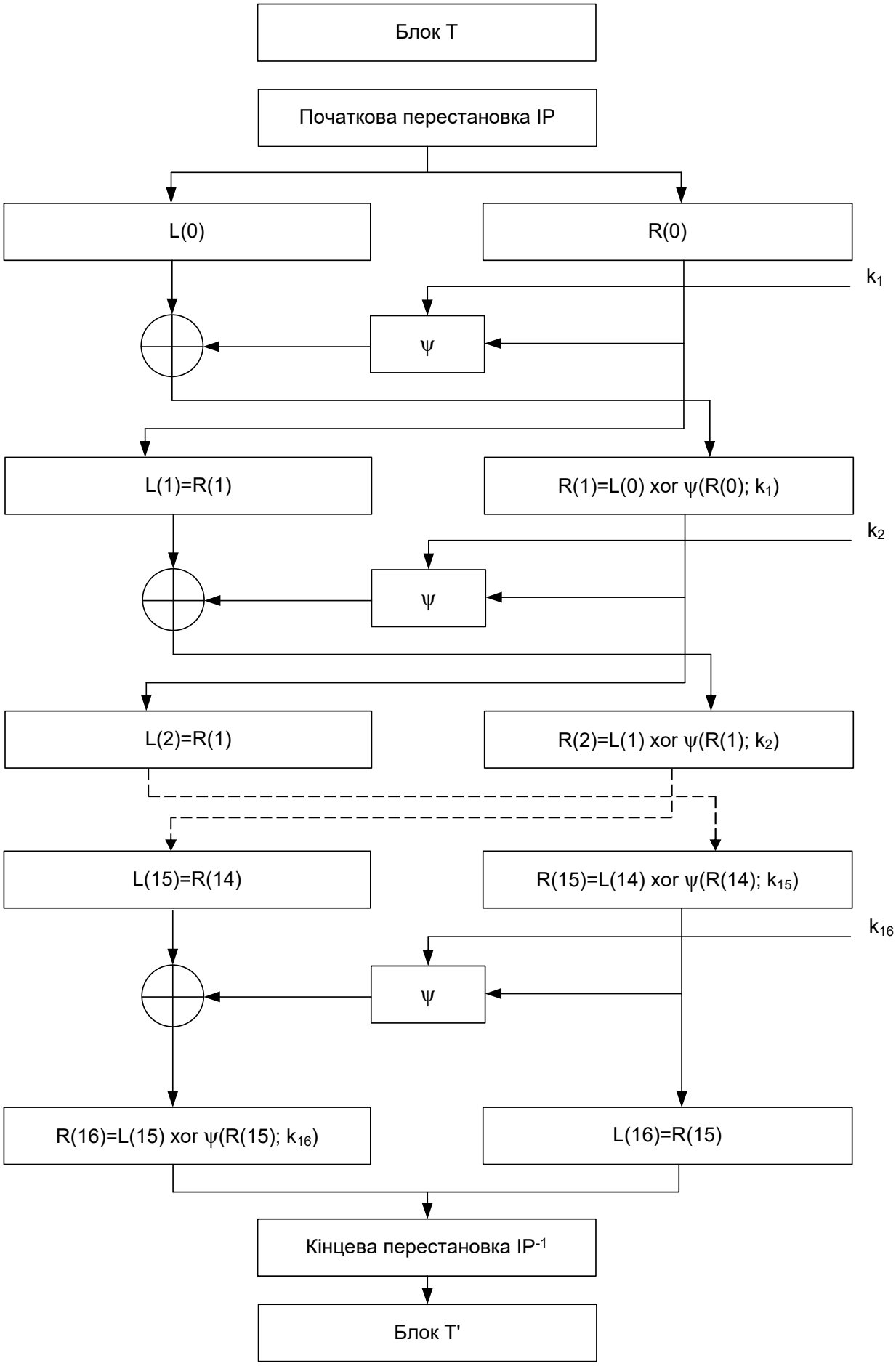

Рисунок 3.3 - Структурна схема алгоритму DES

Таблиця 3.2 - Матриця початкової перестановки IP

58	50	42	34	26	18	10	02
60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06
64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01
59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05
63	55	47	39	31	23	15	07

На наступному кроці обробки виконується процедура шифрування, що уміщує у себе 16 ітерацій (раундів).

У свою чергу, результат виконання і-го раунду для суб-послідовності L(i) може бути описано наступним виразом:

$$L(i) = R(i-1), \tag{3.3}$$

де R(i-1) – 32-бітна послідовність, яку було одержано за результатом виконання (і-1)-ї ітерації.

Відповідно, вираз для опису результату виконання і-го раунду для суб-послідовності R(i) буде таким:

$$R(i) = L(i-1) \oplus \psi(R(i-1); k_i), \tag{3.4}$$

де ψ – функція шифрування;

k_i - ключ шифрування довжиною 48 біт, який було отримано з 64-бітового ключа K.

Подібним чином виконується обробка усіх суб-послідовностей R(i) та L(i) по 15 раунд включно.

Врешті решт, у результаті виконання 16-го раунду формуються суб-послідовності R(16) та L(16) (без перестановки), які далі конкатенують у 64-бітову послідовність R(16)L(16).

Далі позиції біт одержаної послідовності R(16)L(16) підлягають перестановці відповідно до матриці IP^{-1}, як показує таблиця 3.3.

Таблиця 3.3 - Матриця зворотної перестановки IP^{-1}

40	08	48	16	56	24	64	32
39	07	47	15	55	23	63	31
38	06	46	14	54	22	62	30
37	05	45	13	53	21	61	29
36	04	44	12	52	20	60	28
35	03	43	11	51	19	59	27
34	02	42	10	50	18	58	26
33	01	41	09	49	17	57	25

Важливим тут є те, що між матрицями IP^{-1} та IP існує чіткий взаємозв'язок, який наближено може бути представлений наступним чином: значення першого елементу матриці IP^{-1} є рівним 40, тоді як значення 40-го елементу початкової матриці IP рівне 1. У свою чергу, значення 2-го елементу, що належить зворотній матриці (IP^{-1}) дорівнює 8, тоді як значення 8-го елементу з початкової матриці IP дорівнює 2 і т.д.

Для випадку DES, як типової криптосистеми симетричного типу, процес дешифрування попередньо сформованої шифрограми є оберненим (інверсним) відносно до процедури шифротворення. У даному разі це означає, що уся послідовність дій повинна виконуватися у зворотній послідовності.

Інакше кажучи, у процесі дешифрування існуюча шифрограма спочатку підлягає процедурі перестановки відповідно до порядку, який задається матрицею IP^{-1}. Після цього відносно послідовності біт R(16)L(16) виконується той же каскад перетворень, що і під час шифротворення, але у зворотньому порядку, тобто – 16 раундів дешифрування з використанням ключів - з k_{16} по k_1 а далі – перестановка за матрицею IP.

Власне, більш детально процедура ітеративного дешифрування може бути проілюстровано наступними виразами:

$$R(i-1) = L(i), i = \overline{1;16}, \qquad (3.5)$$

$$L(i) = R(i) \oplus \psi(L(i); k_i),\ i = \overline{1;16}. \quad (3.6)$$

У підсумку, після виконання дій 16-го раунду дешифрування, отримуються суб-послідовності L(0) та R(0), які далі конкатенують у 64-бітну послідовність L(0)R(0).

На фінальному кроці дешифрування виконується зміна позицій біт отриманої послідовності L(0)R(0), для чого використовується принцип, заданий матрицею IP. У результаті цього формується вихідна послідовність довжиною 64 біта, тобто, початковий блок T.

Далі розглянемо, що саме собою являє функція $\psi(R(i-1), k_i)$ шифрування. Її схематичну інтерпретацію зображено на рис. 3.4.

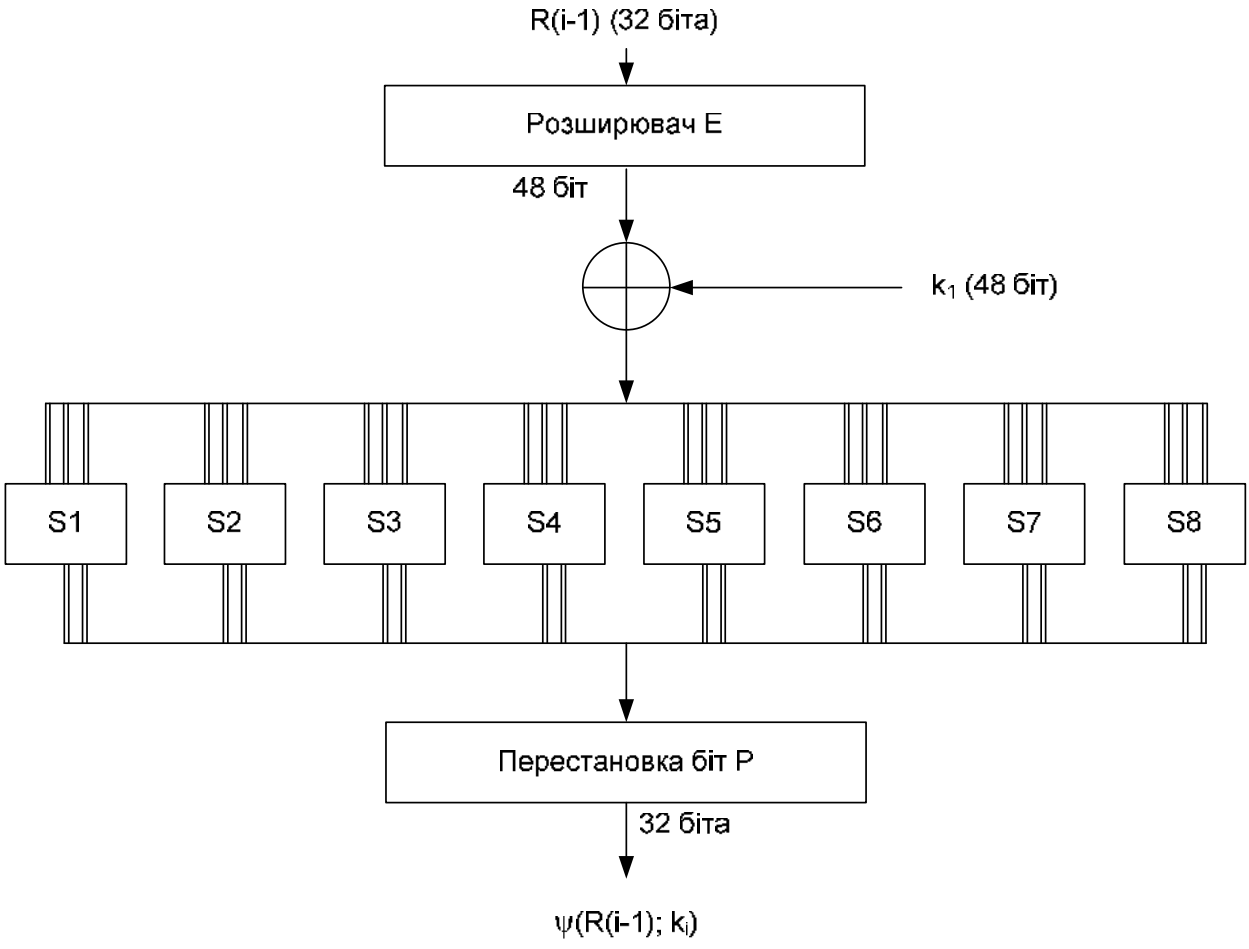

Рисунок 3.4 – Обчислення функції шифрування

При цьому, для обчислення значення функції $\psi(R(i-1), k_i)$ застосовуються такі функції-матриці, як:

- E, або розширення 32-бітової послідовності до 48-бітової;
- S1- S8 - пперетворення 6-бітового блоку на 4-бітовий;
- P - перестановка біт у 32-бітовій послідовності.

У свою чергу, функція розширення (розширювач) E визначається згідно з таблицею 3.4.

Так, відповідно до зазначеної таблиці перші 3 біта $E(R(i-1))$ - це біти 32, 1 а також 2, а, відповідно, останні 3 біта – це біти 31, 32 та 1.

Таблиця 3.4 - Функція розширення E

32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	01

При цьому, результатом виконання функції $E(R(i-1))$ є послідовність з 48 біт. Далі утворена послідовність складається за модулем 2 з 48-бітовим ключем k_i. У результаті цього формується 48-бітова послідовність, яка далі підлягає процедурі розбиття на вісім 6-бітових блоків – з B(1) по B(8). Дане перетворення формально може бути представлено наступним виразом:

$$E(R(i-1)) \oplus k_i = B(1)B(2)...B(8) \qquad (3.7)$$

Водночас, визначення функцій S1 - S8 здійснюється за таблицею 3.5.

Дана таблиця має формат, відмінний від попередньо розглянутих, а відтак - потребує додаткових пояснень.

Отже, припустимо, що на вхід функції-матриці S_j надходить 6-бітовый блок $B(j) = \{b_1; b_2; b_3; b_4; b_5; b_6\}$. У цьому випадку двобітове число $(b_1; b_6)$ вказуватиме на номер рядку матриці, а, у свою чергу, $\{b_2; b_3; b_4; b_5\}$ - на номер стовпця. З урахуванням цього, результатом Sj(B(j)) будет 4-бітовий

елемент, який буде позиціонованим на перетині зазначених таким чином рядку та стовпця.

Таблиця 3.5 – Функції перетворення S1 - S8

		Номер стовпця 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	
Номер рядку	0 1 2 3	14 4 13 1 2 15 11 8 3 10 6 12 5 9 0 7 0 15 7 4 14 2 13 1 10 6 12 11 9 5 3 8 4 1 14 8 13 6 2 11 15 12 9 7 3 10 5 0 15 12 8 2 4 9 1 7 5 11 3 14 10 0 6 13	S1
	0 1 2 3	15 1 8 14 6 11 3 4 9 7 2 13 12 0 5 10 3 13 4 7 15 2 8 14 12 0 1 10 6 9 11 5 0 14 7 11 10 4 13 1 5 8 12 6 9 3 2 15 13 8 10 1 3 15 4 2 11 6 7 12 0 5 14 9	S2
	0 1 2 3	10 0 9 14 6 3 15 5 1 13 12 7 11 4 2 8 13 7 0 9 3 4 6 10 2 8 5 14 12 11 15 1 13 6 4 9 8 15 3 0 11 1 2 12 5 10 14 7 1 10 13 0 6 9 8 7 4 15 14 3 11 5 2 12	S3
	0 1 2 3	7 13 14 3 0 6 9 10 1 2 8 5 11 12 4 15 13 8 11 5 6 15 0 3 4 7 2 12 1 10 14 9 10 6 9 0 12 11 7 13 15 1 3 14 5 2 8 4 3 15 0 6 10 1 13 8 9 4 5 11 12 7 2 14	S4
	0 1 2 3	2 12 4 1 7 10 11 6 8 5 3 15 13 0 14 9 14 11 2 12 4 7 13 1 5 0 15 10 3 9 8 6 4 2 1 11 10 13 7 8 15 9 12 5 6 3 0 14 11 8 12 7 1 14 2 13 6 15 0 9 10 4 5 3	S5
	0 1 2 3	12 1 10 15 9 2 6 8 0 13 3 4 14 7 5 11 10 15 4 2 7 12 9 5 6 1 13 14 0 11 3 8 9 14 15 5 2 8 12 3 7 0 4 10 1 13 11 6 4 3 2 12 9 5 15 10 11 14 1 7 6 0 8 13	S6
	0 1 2 3	4 11 2 14 15 0 8 13 3 12 9 7 5 10 6 1 13 0 11 7 4 9 1 10 14 3 5 12 2 15 8 6 1 4 11 13 12 3 7 14 10 15 6 8 0 5 9 2 6 11 13 8 1 4 10 7 9 5 0 15 14 2 3 12	S7
	0 1 2 3	13 2 8 4 6 15 11 1 10 9 3 14 5 0 12 7 1 15 13 8 10 3 7 4 12 5 6 11 0 14 9 2 7 11 4 1 9 12 14 2 0 6 10 13 15 3 5 8 2 1 14 7 4 10 8 13 15 12 9 0 3 5 6 11	S8

Розглянемо приклад, коли B(1)=011011.

У зазначеному випадку S1(B(1)) знаходиться на перетині першого рядку та 13-го стовпця.

У стовпці 13 рядку 1 задано значення 5. Звідси виходить, що S1(011011)=0101.

Далі, застосувавши процедуру вибору відносно кожного з 6-бітових блоків B(1)-B(8), у підсумку отримуємо 32-бітову послідовність S1(B(1))S2(B(2))S3(B(3))...S8(B(8)).

В решті решт, для того, щоб одержати результат функції шифрування, необхідно виконати перестановку біт утвореної послідовності.

Для цього буде застосовано функцію перестановки P, що задається таблицею 2.6.

При цьому, перестановка біт початкової послідовності здійснюється таким чином, щоб біт 16 став бітом 1, біт 7 - бітом 2 і т.д.

Таблиця 3.6 - Функція перестановки P

16	07	20	21
29	12	28	17
01	15	23	26
05	18	31	10
02	08	24	14
32	27	03	09
19	13	30	06
22	11	04	25

Отже, функцію шифрування далі може бути описано наступним чином:

$$\psi(R(i-1); k_i) = P(S1(B(1)),...S8(B(8)) \qquad (3.8)$$

Далі розглянемо, яким чином функціонує алгоритм отримання 48-бітових ключів k_i, $i = \overline{1,16}$.

У цьому разі у ході кожного з раундів використовується нове значення ключа k_i, що розраховується на базі загального секретного ключа K.

Як було зазначено з самого початку, у рамках DES секретний ключ K являє собою 64-бітовий блок.

При цьому, 8 біт такого блоку – біти контролю парності, вони мають позиційні номери 8, 16, 24, 32, 40, 48, 56 та 64.

Процедуру видалення контрольних біт, а також перестановку інших біт послідовності, реалізує функція G попередньої підготовки ключа, що задається таблицею 3.7.

Таблиця 3.7 - Матриця G попередньої підготовки ключа

57	49	41	33	25	17	09
01	58	50	42	34	26	18
10	02	59	51	43	35	27
19	11	03	60	52	44	36
63	55	47	39	31	23	15
07	62	54	46	38	30	22
14	06	61	53	45	37	29
21	13	05	28	20	12	04

У свою чергу, далі отриманий результат перетворення G(K) зазнає розділення на 2 блоки довжиною 28 біт кожен - C(0) та D(0).

Тут блок C(0) формують біти 57 - 36 секретного ключа K, тоді як блоку D(0) належатимуть біти з 63 по 4 вихідного ключа K.

Наступним кроком, після побудови C(0) та D(0), є рекурсивне визначення C(i) та D(i) при $i = \overline{1;16}$.

На даному технологічному етапі застосовується процедура циклічного зсуву вліво на один або два біта, залежно від номеру раунду, як зазначається таблицею 3.8.

Таблиця 3.8 – Величина зсуву для обчислення раундового ключа

Раунд	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Зсув, біт	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Після виконання означеної процедури, одержані таким чином дані далі підлягають процедурі перемішування.

Закономірність реалізації даної процедури визначає матриця H, наведена у табл. 2.9.

Таблиця 2.9 - Матриця H фінальної обробки ключа

14	17	11	24	01	05
03	28	15	06	21	10
23	19	12	04	26	08
16	07	27	20	13	02
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Тут ключ k_i складатиметься з біт 14 - 32 послідовності C(i)D(i), інакше кажучи:

$$k_i = H(C(i)D(i)). \quad (3.9)$$

Блок-схема алгоритму розрахунку раундових ключів представлена на рис. 3.5

Отже, виконавши детальне дослідження принципів формування шифрограм на базі технологій DES та AES, можемо зазначити наступне:

- як DES, так і AES, незважаючи на значну передбачену кількість раундів шифрування (10-14 для AES та 16 для DES) характеризуються високою швидкістю шифротворення;
- обидві розглянуті шифросистеми відзначаються простотою з алгоритмічної точки зору: найбільш складною операцією для DES є складання за модулем 2, для AES – поліноміальні операції у полі Галуа;
- криптографічна система AES є жорстко специфікованим алгоритмом, що накладає обмеження на внесення змін у її наявну архітектуру;
- стандарт DES, з точки зору можливості модифікації, за винятком таблиць перестановок технічно є більш демократичнішою шифросистемою, на засадах якої може бути побудовано власний алгоритм криптографічного захисту даних.

Таким чином, у якості базису для розробки власної криптосистеми доцільно розглядати технологію DES.

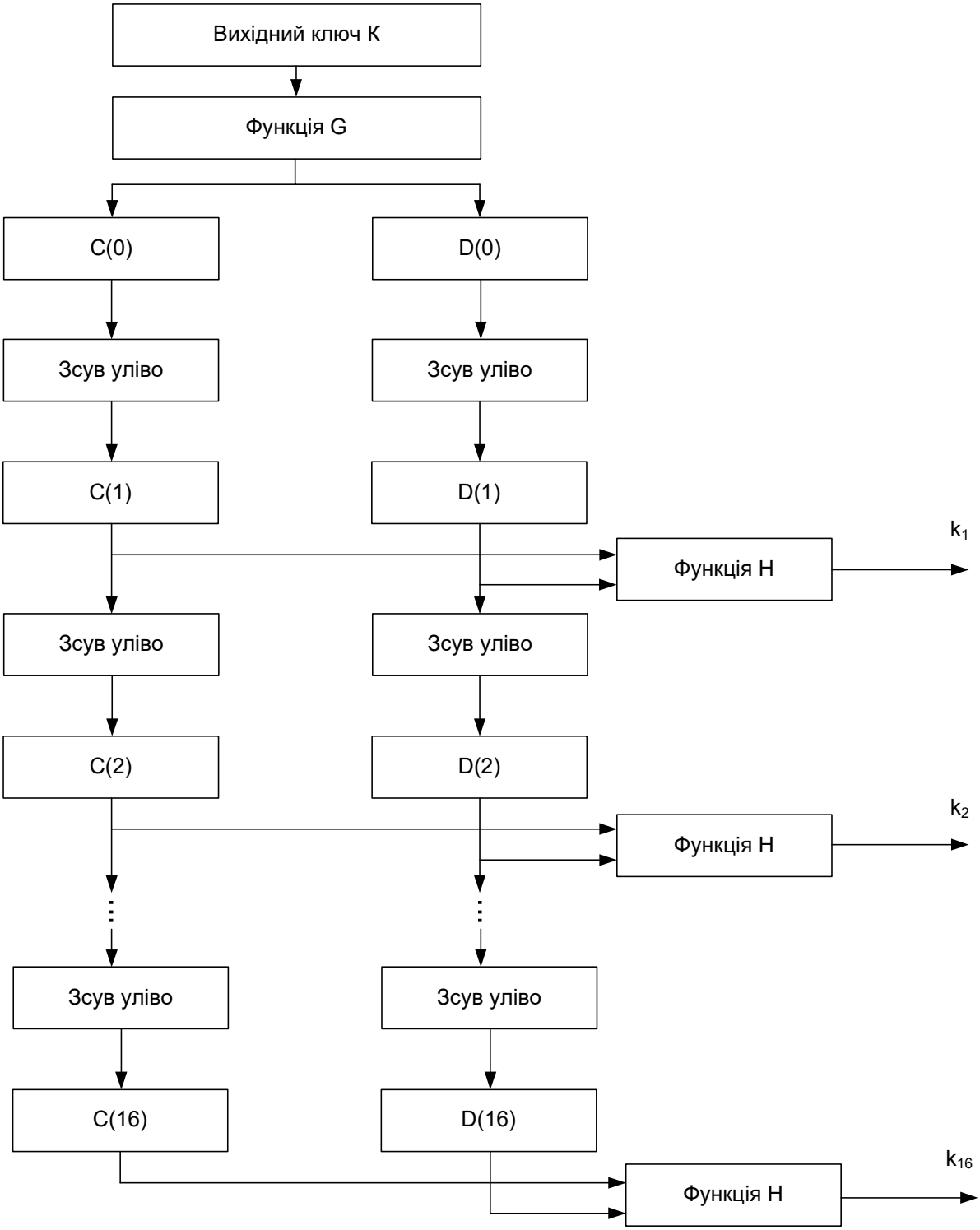

Рисунок 3.5 – Блок-схема обчислення раундових ключів DES

3.2 Поширені криптосистеми асиметричної архітектури

3.2.1 Криптосистема RSA

Алгоритм RSA, як, типова асиметрична шифросистема, для реалізації процедур шифрування/дешифрування, використовує публічний, або

відкритий ключ (public key), а також ключ закритого типу або приватний (private key).

Для формування ключів кожного з типів, при цьому, використовується пара цілочисельних величин. При цьому, ініціатор обміну даними (приймач) формує як публічний, так і приватний ключ.

У рамках ідеології RSA пара відкритого та закритого ключів являє собою т. з. *узгоджену пару*.

У цьому випадку узгодженій парі (O,C) ключа O відкритого типу та приватного C ключа як шифруючі $S_o(x)$, так і обернені – тобто, дешифруючі $D_c(x)$ перетворення. У свою чергу, це, для будь-якого довільного повідомлення m з простору M допустимих повідомлень, здатне гарантувати виконання наступного виразу:

$$m = D_c(S_o(m)) = S_o(D_c(m)). \qquad (3.10)$$

Відтак, побудова будь-якої шифрограми на базі RSA починається з розрахунку ключів відкритого та закритого типів.

Технологічний процес формування ключів RSA є стандартизованим, та уміщу у себе ряд кроків, таких, як:

1. Вибір пари простих чисел a та b, на базі яких далі формуються ключі розміром Θ, який встановлюється спочатку; сьогодні більшість додатків, що тим чи іншим чином реалізують алгоритм RSA, найчастіше використовують ключ довжиною $\Theta = 2048,$, або $\Theta = 4096$ біт.

2. Виконання розрахунку величини модулю ϕ алгоритму; зазначена процедура являє собою обчислення добутку попередньо обраної пари чисел a та b:

$$\phi = a \times b. \qquad (3.11)$$

3. Розрахунок значення величини функції Ейлера, у ході чого також беруться до уваги величини пари чисел a та b, як показує наступний вираз:

$$E = (a-1) \times (b-1). \qquad (3.12)$$

4. Формування відкритої експоненти e. Тут відкрита експонента - це цілочисельна величина, яка є взаємно простою зі значенням функції $Q(\phi)$. Тобто, для відкритої експоненти e, також, як і для $Q(\phi)$, має існувати один і лише один спільний дільник, що дорівнює 1. Разом з тим, значення e має задовольняти наступну вимогу:

$$1 < e < Q. \tag{3.13}$$

Додатковою вимогою до значення e відкритої експоненти тут може слугувати забезпечення мінімізації кількість одиничних елементів у її двійковому форматі опису. Тому, виходячи з даної вимоги, значення величини e доцільно обирати з простору чисел Ферма. При цьому, обчислення величини e виконується на базі наступного виразу:

$$e = 2^{2^n} + 1,\ n \geq 0. \tag{3.14}$$

З точки зору досягнення відносно високої стійкості утвореного шифру при порівняно невисоких часових витратах на реалізацію даного процесу, нерідко величину e приймається з діапазоні значень n такого, що $n = \overline{2;4}$.

А саме, за умови, що n=2 (відповідає $e = 17$), чи n=3 (відповідає $e = 257$ за виразом (2.21)), чи при n=4 (відповідає $e = 65537$).

З іншого боку, очевидно, що значення e є замалим. У цьому випадку, наприклад, коли $e = 1$ (n=0), або $e = 3$ (при n=1), загальний рівень захищеності отриманої шифрограми на базі RSA суттєво зменшуватиметься.

5. Обчислення значення δ секретної експоненти. Розрахунок величини δ виконується, беручи до уваги тотожність:

$$(\delta \times e) \equiv 1 \mod Q(\phi). \tag{3.15}$$

Відтак, значення величини δ повинно мати властивість мультиплікативності по відношенню до величини e. Отже, остача від ділення добутку $(\delta \times e)$ за модулем $Q(\phi)$ тут повинен дорівнювати 1.

Зазвичай, обчислення величини δ реалізується з використаннямрозширеного алгоритму Евкліда.

6. Використання пари значень $(e;\phi)$ як складових компонент публічного ключа.

7. Використання пари значень $(\delta;\phi)$ як складових компонент приватного ключа.

2.2.1 Шифросистема Ель-Гамаля

Дана шифросистема, як і RSA, також належить до систем криптографічного захисту асиметричного типу.

Її стійкість, у свою чергу, базується на складності виконання обчислень дискретних логарифмів у скінченому полі. Шифросистема Ель-Гамаля поєднує у собі механізм шифротворення та механізм формування цифрового підпису.

Формально алгоритм відзначається простотою та у режимі шифрування умовно поділяється на такі етапи, як:

- формування ключів;
- побудова шифру.

На етапі формування ключів виконуються такі операції:

1. Синтез випадкової простої величини p.
2. Вибір першообразного кореню p - числа g.
3. Вибір цілої випадкової величини x, що задовольняє вимогу:

$$1 < x < (p-1). \tag{3.16}$$

4. Обчислення величини y за формулою:

$$y = g^x \bmod p \tag{3.17}$$

У цьому випадку (y, g, p) - є ключем відкритого типу, x - ключ приватного типу.

У свою чергу, далі, у режимі шифрування, виконується:

1. Перевірка умови:

$$M < p, \tag{3.18}$$

де M - довжина повідомлення.

2. Вибір сесійного ключа, у ролі якого виступає випадкова величина k, що є взаємно простою зі значенням $(p-1)$, та відповідає наступній вимозі:

$$1 < k < (p-1) \tag{3.19}$$

3. Обчислення величин a та b згідно з виразами:

$$a = g^k \bmod p \tag{3.20}$$

$$b = y^k M \bmod p \tag{3.21}$$

У цьому випадку пара (a,b) являє собою шифрограму.

Для того, щоб виконати дешифрування, слід скористатися наступним виразом:

$$M = b(a^x)^{-1} \bmod p \tag{3.22}$$

Таким чином, з урахуванням аналізу специфіки алгоритмів Ель-Гамаля та RSA, зазначимо наступне;

- розглянуті алгоритми асиметричного шифрування характеризуються схожою архітектурою;
- криптографічна стійкість шифрограм, утворених на базі алгоритмів Ель-Гамаля та RSA, є аналогічною, як показано у розділі 1;
- розглянуті алгоритми характеризуються у цілому однаковим рівнем обчислювального навантаження.

Разом з тим, алгоритм RSA має ряд потенційних недоліків, як то:

- потенційна можливість зламу шифру з використанням атаки Вінера;
- потенційна можливість реконструкції приватного ключа (а саме – секретної експоненти), керуючись відомостями про публічний ключ та деякими закономірностями з теорії чисел, які є справедливі для RSA.

Існування зазначених недоліків, у свою чергу, не гарантує зловмисникові можливість зламу RSA-шифру. Разом з тим, дані недоліки

дозволяють суттєвим чином скоротити простір перебору у випадку використання brootforce-атак.

Відтак, беручи до уваги усе зазначене, більш перспективним алгоритмом, на базі якого може виконуватися обмін ключами симетричного шифрування, можемо вважати алгоритм Ель-Гамаля.

4. СИНТЕЗ КРИПТОГРАФІЧНОЇ ШИФРОСИСТЕМИ НА БАЗІ МЕРЕЖІ ФЕЙСТЕЛЯ

4.1 Технологічна база для побудови шифросистеми

Побудова шифру на засадах структури Фейстеля виконується у наступних умовах:

1. Апаратне забезпечення:
- Pentium Dual-Core CPU T4500 @ 2,3 GHz;
- 8 Gb RAM;
- архітектура x64.

2. Програмне забезпечення:
- мова Python 3.10;
- ОС Windows 10 Pro збірка 18363.657.

Так як технічне завдання жорстко не регламентує параметрів шифру, створюється шифр, що включає у себе 4 раунди, та наслідує архітектуру мережі Фейстеля.

Програмна реалізація шифру передбачається у вигляді консольного додатку з можливістю керування «гарячими» клавішами.

4.2 Опис функцій у складі шифру

Реалізовані програмні функції відповідають ключовим етапам перетворення мережі Фейстеля у загальних рисах подібні до функцій, реалізованих у DES.

4.2.1 Формування раундових ключів

Так, для генерування раундового k_i ключа з загального секретного ключа К виконується зсув початкового ключа на номер раунду і далі конкатенація утвореної послідовності з її вихідною версією, тобто:

$$\begin{cases} K'(i) := sh(K; i) \\ k_i = K'(i) \,\&\, K \end{cases}, \tag{4.1}$$

де $sh(K;i)$ - оператор зсуву послідовності, символів ключа К на і позицій.

За рахунок цього гарантується, що для кожного раунду застосовується окремий ключ , отриманий з базового.

Дану операцію реалізовано функцією *generate_subkey(key, round_num)* програмного коду.

4.2.2 Об’єднання зміненої та вихідної половини блоку

Зазначену процедуру реалізує функція *xor(a, b)*.

На вхід даної функції надходять два рядки однакової довжини, які далі посимвольно складаються за модулем 2.

Для можливості виконання цієї операції, кожен символ попередньо перетворюється на код ASCII, тобто, операція XOR виконується над кодами. Далі результат перетворюється на символ.

4.2.3 Накладення раундового ключа

Операцію застосування раундового ключа, яка виконується на кожному з раундів, виконує функція *feistel_function(right, key)*.

Ця функція отримує на вхід праву половину вхідних даних, а також раундовий ключ, повертаючи на виході результат їх складання за модулем 2.

4.2.4 Розширення ключа

Виходячи з того, що блоковий шифр обробляє блоки даних фіксованого розміру, це накладає обмеження на довжину секретного ключа.

У випадку, коли довжина $abs(K)$ ключа є меншою, ніж довжина $abs(M)$ блоку повідомлення, ключ має бути розширено для забезпечення виконання умов $abs(K) = abs(M)$.

У нашому випадку передбачається, що шифр може працювати з блоками довільної довжини, відтак актуальною тут є, скоріше, процедура не доповнення ключа, а узгодження довжин ключа та повідомлення.

Процедуру розширення ключа реалізує функція *pad_key(key, length, pad_char=’ ‘).*

Вхідними даними для неї є ключ К, необхідна довжина та символ підстановки.

Функція доповнює ключ масивом з вказаних символів до отримання сумарної його довжини, рівної довжині блоку.

Для вибору конкретного символу підстановки необхідно змінювати його у специфікації функції.

4.2.5 Шифрування

Дана функція є об’єднуючою для ряду часткових функцій. Сама функція *feistel_network(text, key)* приймає на вхід повідомлення M та секретний ключ K, та повертає шифроване повідомлення M’.

Для цього спочатку заповнюється ключ, виконується розбиття повідомлення на фрагменти такої ж довжини, яку має ключ, після цього виконується 4 раунди шифрування для кожного з фрагментів повідомлення. У результаті цього отримується рядок шифрованих фрагментів, поєднаних разом.

4.2.6 Дешифрування

Розшифрування повідомлення виконує функція *feistel_network_decrypt(text, key)*.

Вхідними даними для неї є зашифроване повідомлення та секретний ключ K. На виході отримується початкове повідомлення, як результат зворотної процедури шифрування.

4.2.7 Керуюча змінна

Для забезпечення можливості управління режимом роботи програмного засобу, використовуються т.з. «гарячі клавіші».

Тут режиму шифрування відповідає символ «1» а дешифрування, відповідно, символ 2.

Щоб мати можливість використання зазначеного механізму, до коду введено змінну *inp.*

Відтак, при *inp=1* ініціалізується функція *feistel_network(text, key)* і навпаки.

На той випадок, коли довжина повідомлення є непарним числом (тобто, поділ на 2 частини неможливий), попередньо виконується його доповнення символом пробілу.

Вихідний код алгоритму наводиться у Додатку Б.

Структурна схема алгоритму шифрування у прив’язці до кодової реалізації ілюструється рисунком 4.1

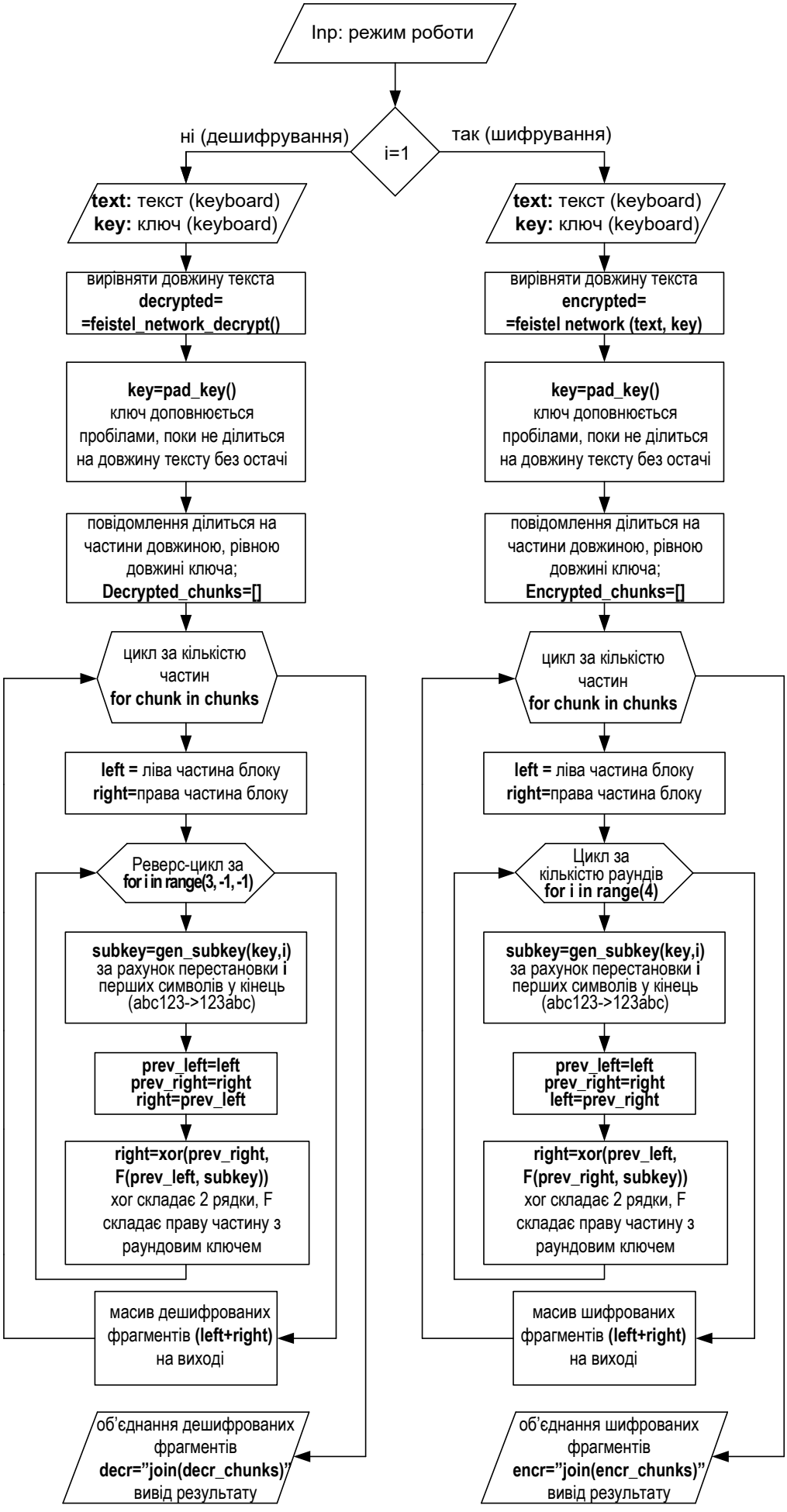

Рисунок 4.1 – Структурна схема алгоритму

4.4 Методика тестування кодової реалізації алгоритму

Процес тестування функціональності реалізованої шифросистеми включає у себе режими:

- тестування процедури обробки повідомлення парної довжини;
- тестування процедури обробки повідомлення непарної довжини.

Кожен з режимів, у свою чергу, містить у собі:

- перевірку шифрування повідомлення
- перевірку процесу дешифрування повідомлення за відомим ключем та порівняння результату з вихідним повідомленням

Критерієм вірності роботи створеної шифросистеми є рівність шифрованого та дешифрованого повідомлень для випадків парних та непарних довжин.

При цьому, у рамках тестування системи для найгіршого випадку, пропонується розглянути приклад шифрування, коли ключ являє собою масив однакових символів.

4.4.1 Результати тестування шифросистеми

Спочатку розглянемо приклад шифрування, коли повідомленням є рядок символів кирилиці (парний) а ключ – цифрова послідовність однакових чисел (рис.4.2).

```
PS C:\Users\3v3rs> & H:/pythonn/python.exe c:/Users/3v3rs/feistel.py
Press 1 for encrypt or 2 for decrypt: 1
Enter a message to encrypt: эщкере
Enter a key: 99999
epeA0/
PS C:\Users\3v3rs> & H:/pythonn/python.exe c:/Users/3v3rs/feistel.py
Press 1 for encrypt or 2 for decrypt: 2
Enter a message to decrypt: epeA0/
Enter a key: 99999
эщкере
```

Рисунок 4.2 – Результат шифрування тексту кирилиці з ключем у вигляді послідовності однакових чисел

Утворена таким чином шифрограма, як видно з рис.4.2, містить фрагмент вихідного повідомлення.

Далі розглянемо випадок, коли шифрується повідомлення парної довжини з парним та більш ефективним ключем (рис.4.3).

```
PS C:\Users\3v3rs> & H:/pythonn/python.exe c:/Users/3v3rs/feistel.py
Press 1 for encrypt or 2 for decrypt: 1
Enter a message to encrypt: random text of even len1
Enter a key: anykey09
xt.y?lw len1ng+#uh~m)=:1
PS C:\Users\3v3rs> & H:/pythonn/python.exe c:/Users/3v3rs/feistel.py
Press 1 for encrypt or 2 for decrypt: 2
Enter a message to decrypt: xt.y?lw len1ng+#uh~m)=:1
Enter a key: anykey09
random text of even len1
```

Рисунок 4.3 – Результат шифрування повідомлення парної довжини з парним ключем довільного змісту

У цьому випадку, як можна пересвідчитись, шифрограма не має недоліків, властивих умовам використання низькоефективного ключа.

Далі, для випадку повідомлення парної довжини та непарного ключа отримуємо результат, як зображено на рис. 4.4.

```
PS C:\Users\3v3rs> & H:/pythonn/python.exe c:/Users/3v3rs/feistel.py
Press 1 for encrypt or 2 for decrypt: 1
Enter a message to encrypt: random text of even len1
Enter a key: keyanyy
s~/rv<n len1nm `.q7t)=:1
PS C:\Users\3v3rs> & H:/pythonn/python.exe c:/Users/3v3rs/feistel.py
Press 1 for encrypt or 2 for decrypt: 2
Enter a message to decrypt: s~/rv<n len1nm `.q7t)=:1
Enter a key: keyanyy
random text of even len1
```

Рисунок 4.4 – Результат шифрування повідомлення парної довжини та непарним ключем

Тут, як показує аналіз рис. 4.4, у структурі шифрограми візуалізується фрагмент вихідного повідомлення.

Тепер виконаємо аналіз випадку, коли ключ має парну довжину а повідомлення - непарну (рис.4.5).

```
PS C:\Users\3v3rs> & H:/pythonn/python.exe c:/Users/3v3rs/feistel.py
Press 1 for encrypt or 2 for decrypt: 1
Enter a message to encrypt: random text of odd len1
Enter a key: anykey09
xt.s-m9len1 ng+)gi0! 6e
PS C:\Users\3v3rs> ^C
PS C:\Users\3v3rs> & H:/pythonn/python.exe c:/Users/3v3rs/feistel.py
Press 1 for encrypt or 2 for decrypt: 2
Enter a message to decrypt: xt.s-m9len1 ng+)gi0! 6e
Enter a key: anykey09
random text of odd len1
```

Рисунок 4.5 - Результат шифрування повідомлення непарної довжини парним ключем

З рисунку 4.5 бачимо, що шифрограма для даного випадку має недолік, характерний попередній ситуації – шифруванню повідомлення парної довжини непарним ключем. Нарешті, розглянемо випадок, коли і ключ, і повідомлення мають непарну довжину (рис.4.6).

```
PS C:\Users\3v3rs> & H:/pythonn/python.exe c:/Users/3v3rs/feistel.py
Press 1 for encrypt or 2 for decrypt: 1
Enter a message to encrypt: rendom text of odd len1
Enter a key: 1010102
ng!nfv len1 -2~9(928 6e
PS C:\Users\3v3rs> & H:/pythonn/python.exe c:/Users/3v3rs/feistel.py
Press 1 for encrypt or 2 for decrypt: 2
Enter a message to decrypt: ng!nfv len1 -2~9(928 6e
Enter a key: 1010102
rendom text of odd len1
```

Рисунок 4.6 - Результат шифрування повідомлення непарної довжини непарним ключем

На цей випадок, знову ж таки, маємо відкриту ділянку вихідного повідомлення у сформованій шифрограмі.

4.5 Висновки за результатами тестування

Як свідчать результати тестування, найбільшу ефективність шифротворення для розробленого шифру забезпечується у випадку, коли

виконується обробка парної послідовності символів повідомлення за участю ключа парної довжини. Це дає можливість стверджувати, що шифр може розглядатися, як потенційно ефективний засіб захисту даних.

Беручи до уваги те, що створена шифросистема не виконує формування ключів самостійно, та не містить у собі складних операцій математичних обчислень, її швидкодія, принаймні, не поступається швидкості шифротворення DES, тобто, даний алгоритм може розглядатися, як засіб обробки мультимедіа.

В інших досліджених випадках (примітивний ключ, різні комбінації парності та непарності) шифрограма містила відкритий фрагмент вихідного повідомлення.

ВИСНОВКИ

Відповідно до умов технічного завдання, у роботі виконувалося:

1. Дослідження архітектур та способів реалізації поширених сучасних шифросистем симетричного та асиметричного типу;

2. Пошук криптографічних алгоритмів, які:

- є перспективними для швидкої обробки великих масивів даних з мінімальною затримкою;

- мають архітектуру, що дозволяє на її базі побудувати власну шифросистему, яка дозволяє обробляти великі масиви даних в одиницю часу, тобто, здатна забезпечити шифрування даних мультимедіа.

3. Дослідження асиметричних алгоритмів, що можуть бути використані для обміну ключами симетричного шифрування та виявлення найбільш ефективного з них.

На базі цього було необхідно:

- дослідити принципи побудови шифру за попередньо виявленою архітектурою;

- отримати програмну реалізацію власної шифросистеми керуючись даною архітектурою.

У підсумку виконаного дослідження асиметричним алгоритмом, який є найбільш ефективним за рядом показників для забезпечення обміну симетричними ключами, визнано шифр Ель-Гамаля.

У свою чергу, симетричним алгоритмом, що забезпечує відносно високу швидкодію, а також дає змогу на базі його архітектури побудувати власний шифр, за результатами дослідження було визнано алгоритм DES.

На базі архітектури DES, яка у цілому відповідає структурі Фейстеля, використовуючи мову Python, було побудовано власну шифросистему симетричного типу.

Створена шифросистема характеризується такими особливостями, як:

- власний алгоритм формування раундових ключів з загального секретного ключа;

- власна архітектура раундової обробки;

- реалізація шифрограми за 4 раунди криптографічних перетворень;

- відносно висока швидкодія алгоритму, що дозволяє його використовувати для обробки мультимедійної інформації;
- можливість шифрування парних та непарних повідомлень, парними та непарними ключами.

Разом з тим, як показали дослідження, стійка шифрограма генерується лише у випадку, коли обробляється повідомлення парної довжини з використанням парного ключа. Це, у свою чергу, пояснюється недосконалістю алгоритму розширення ключа.

Відтак, можливим вектором розвитку шифросистеми є розробка ефективних алгоритмів розширення ключа/повідомлення.

Отже, усі пункти технічного завдання виконано у повному обсязі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Microsoft security report [Електронний ресурс] – Режим доступу: https://microsoft.com/securityinsights.

2. Antivirus and Cybersecurity Statistics & Facts 2021 [Електронний ресурс] – Режим доступу: https://wethegeek.com/antivirus-statistics-facts/

3. Microsoft unsecurity solutions [Електронний ресурс] Режим доступу: http://www.seattlepi.com/business/275780_msftsuit29.html

4. Шаньгін В.Ф. Захист інформації в розподілених корпоративних мережах і системах [Текст]: підручник / В.Ф. Шаньгин, А.В. Соколов. – М.: ДМК Пресс, 2002. – 656 с.

5. Качество обслуживания в операторских сетях [Електронний ресурс] – Режим доступу: https://www.opennet.ru/docs/RUS/qos_oper/

6. CUDA Tutorial [Електронний ресурс] – Режим доступу: https://cuda-tutorial.readthedocs.io/en/latest/

7. Рябко Б.Я. Основы современной криптографии и стеганографии [Электронный ресурс] : [монография] / А.Н. Фионов, Б.Я. Рябко. — М. : Горячая линия – Телеком, 2010 .— 233 с.

8. Шнайер Б.. Прикладная криптография. Протоколы, алгоритмы и исходный код на С. – М.: Вильямс, 2016. – 1024 с.

9. Мао В. Современная криптография: Теория и практика / пер. Д. А. Клюшина — М.: Вильямс, 2005. — 768 с.

10.Шенон К. Теория связи в секретных системах / пер. с англ. В. Ф. Писаренко // Работы по теории информации и кибернетике / Под редакцией Р. Л. Добрушина и О. Б. Лупанова. — М. : Издательство иностранной литературы, 1963. — 829 с.

11. Cryptology ePrint Archive [Електронний ресурс] – Режим доступу: https://eprint.iacr.org/

12. Block Ciphers — Introduction and overview [Електронний ресурс] – Режим доступу: http://cacr.uwaterloo.ca/hac/about/chap7.pdf

13. Rivest R., Shamir A., Adleman L. A method for obtaining digital signatures and public-key cryptosystems // Commun. ACM - [New York]: Association for Computing Machinery, 1978. - Vol. 21, Iss. 2. - P. 120-126.

14. Авдошин С., Набебин А. Дискретная математика. Модулярная алгебра, криптография, кодирование. – М.: ДМК Пресс, 2016. – 352 с.

15. Крэндалл Р., Померанс К. Простые числа. Криптографические и вычислительные аспекты. – М.: Либроком, 2011. – 664 с.

16. ROCA: Vulnerable RSA generation (CVE-2017-15361) | Crocs Wiki [Електронний ресурс] – Режим доступу: https://crocs.fi.muni.cz/public/papers/rsa_ccs17

17. Maitra S. Revisiting Wiener's Attack - New Weak Keys in RSA. — Indian Statistical Institute. — 2008.

18. Ferguson N., Schroeppel R. and Whiting D. A simple algebraic representation of Rijndael. Selected Areas in Cryptography, Proc. SAC 2001, Lecture Notes in Computer Science #2259. — Springer Verlag, 2001. — P. 103—111.

19. Rijndael block cipher [Електронний ресурс] – Режим доступу: https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf

20. Schindler W., Koeune F., Quisquater J-J. Improving Divide and Conquer Attacks against Cryptosystems by Better Error Detection/Correction Strategies. Proc. of 8th IMA International Conference on Cryptography and Coding :— 2001. — P. 245—267. — doi:10.1.1.13.5175

21. Python Tutorial [Електронний ресурс] – Режим доступу: https://www.pythontutorial.net/