**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»**

*Спеціальність: 123 «Комп'ютерна інженерія»*

*Освітньо-професійна програма: «Безпека комп'ютерних систем і мереж»*

*Група: 4КБ-01*

# Дипломний проект

**здобувача освіти денної форми навчання**
**КБ.01.10.000.ДП**

## *КИСИЛИЦІ КИРИЛА ОЛЕКСІЙОВИЧА*

**м. Одеса**
**2024 р.**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: «**Безпека комп'ютерних систем і мереж»**

Група: **4КБ-01**

# ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту на тему:

**Програмна реалізація алгоритму шифрування Twofish для захисту користувацьких файлів**

Проектний матеріал складається з пояснювальної записки на 84 сторінках та графічного (презентаційного) матеріалу на 19 аркушах (слайдах)

Дипломник _____ ( Кисилиця К.О.)

Керівник _____ ( Кривченко А.А.)

**Консультанти:**

з економічного розділу _____ ( Іванченков В.С.)

з розділу охорони праці та техніки безпеки _____ ( Чорновол Н.І.)

з нормоконтролю _____ ( Петрашова В.І.)

старший консультант _____ ( Кривченко Ю.В.)

**До захисту допущений**

Голова циклової комісії _____ ( Кривченко Ю.В.)

Завідувач відділення _____ ( Скорнякова О.В.)

Захист «14» 06 2024 р. Протокол ЕК № 1

Оцінка ЕК 5 (відмінно) 95 б.

Секретар ЕК _____

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»**

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Безпека комп'ютерних систем і мереж»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"15" 01 2024 р.

# ЗАВДАННЯ

**на дипломний проект**

Здобувачеві освіти *Кисилиці Кирилові Олексійовичу*

(прізвище, ім'я, по батькові)

1. Тема проекту *Програмна реалізація алгоритму шифрування Twofish для захисту користувацьких файлів*

затверджена наказом по коледжу від "02" 11 2023 р. № 244-А2-ОД

2. Термін здачі закінченого проекту 10.06.2024

3. Вихідні данні до проекту *1. Реалізувати програмну модель криптостійкого кодеку на базі симетричного блокового шифру Twofish з перевіркою на цілісність даних. 2. Передбачити використання генератора випадкових чисел для генерування ключу або введення його уручну в 16-му вигляді; 3. Реалізувати візуальний інтерфейс користувача для створюваного додатку; 4. Використовувати мову C++ для реалізації криптоалгоритму; 5. Передбачити перевірку роботи кодеку за допомогою моделювання у Cryptool2*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

*Класифікація існуючих алгоритмів шифрування; Порівняльний огляд блокових алгоритмів шифрування; Використання криптографічного алгоритму Twofish для реалізації програмного застосунку; Моделювання роботи алгоритму Twofish; Створення програмного застосунку для захисту користувацьких файлів; Економічні розрахунки; Заходи охорони праці та ТБ*

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

*Узагальнена схема симетричного шифрування; Принципи блокового шифрування; Структура мережі Фейстеля при шифруванні; Структурна схема захисту інформації з застосуванням криптографічного алгоритму Twofish; Схема роботи алгоритму Twofish; Результат моделювання за алгоритмом Twofish; UML-діаграма варіантів використання; Deployment UML-діаграма для програмного застосунку; Концептуальна діаграма класів для програмного застосунку; Інтерфейс застосунку для захисту користувацьких файлів*

6. Консультанти по проекту, із зазначенням розділів проекту, що їх стосується

| Розділ | Консультант | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| Основний розділ | *Кривченко А.А.* | | |
| Економічний розділ | *Іванченков В.С.* | | |
| Розділ охорони праці | *Чорновол Н.І.* | | |
| Нормоконтроль | *Петрашова В.І.* | | |
| Старший консультант | *Кривченко Ю.В.* | | |

7. Дата видачі завдання 15.01.2024

Керівник *Кривченко А.А.* (підпис)

Завдання прийняв до виконання *Кисилиця К.О.* (підпис)

КАЛЕНДАРНИЙ ПЛАН

| № з/р | Назва етапів дипломного проекту | Термін виконання етапів дипломного проекту (роботи) | Відмітка про виконання |
|---|---|---|---|
| 1 | *Вступ. Постановка задачі проектування* | 01.05.24 | виконано |
| 2 | *Аналіз технічного завдання та пошук літератури* | 15.05.24 | виконано |
| 3 | *Аналіз властивостей блокових алгоритмів* | 20.05.24 | виконано |
| 4 | *Порівняльний огляд сучасних алгоритмів блокових* | 25.05.24 | виконано |
| | *шифрування. Вивчення алгоритму Twofish* | 28.05.24 | виконано |
| 5 | *Огляд існуючих продуктів для захисту* | | |
| | *користувацьких файлів засобами криптографії* | 29.05.24 | виконано |
| 6 | *Використання криптографічного алгоритму Twofish* | | |
| | *для реалізації програмного застосунку* | 30.05.24 | виконано |
| 7 | *Моделювання роботи алгоритму Twofish* | 01.06.24 | виконано |
| 8 | *Вибір програмних засобів розробки* | 02.06.24 | виконано |
| 9 | *Створення програмного застосунку для захисту* | | |
| | *користувацьких файлів* | 04.06.24 | виконано |
| 10 | *Випробування додатку та аналіз результатів* | 05.06.24 | виконано |
| 11 | *Виконання економічних розрахунків* | 06.06.24 | виконано |
| 12 | *Розробка питань з охорони праці та техніки безпеки* | 07.06.24 | виконано |
| 13 | *Підготовка мультимедійної презентації проекту* | 09.06.24 | виконано |

Дипломник (підпис)

Керівник (підпис)

| Формат | Зона | Поз. | Позначення | Назва | Кіл. | Примітка |
|---|---|---|---|---|---|---|
| | | | | Документація | | |
| | | | КБ 01. 10 000. 00 ДП | Дипломний проект | | |
| А4 | | | КБ 01. 10 000. 00 ДП ПЗ | Пояснювальна записка | 1 | |

| Зм. | Арк. | № докум. | Підпис | Дата | КБ 01. 10 000. 00 ДП | | |
|---|---|---|---|---|---|---|---|
| Розробив | | Кисилиця К.О. | | 8.06 | Програмна реалізація алгоритму шифрування Twofish для захисту користувацьких файлів | Літ. | Аркуш 4 / Аркушів 84 |
| Перевірив | | Кривченко А.А. | | 10.06 | | Н Д П | |
| Н. контр. | | Петрашова В.І. | | 10.06 | | ВСП "ОТФК ОНТУ" гр.4КБ-01 | |
| Затверд. | | Кривченко Ю.В. | | 10.06 | | | |

# ЗМІСТ

# ВСТУП

Широке впровадження мережевих технологій та інтегрованих систем обробки даних стало основним стимулом розвитку систем захищення даних і комп'ютерної безпечності, яка, як частина комп'ютерної інженерії інтенсивно розвивається у останні роки.

Активний вплив на динаміку розвитку засобів захищення даних містить вдосконалення методів та засобів порушення безпечності. Основним елементом більшості сучасних систем захищення даних є її криптографічне перетворення. Швидке зростання продуктивності сучасних засобів обчислювальної техніки та можливості організації паралельного вирішення завдань криптоаналізу на комп'ютерах, об'єднаних у мережі, різко знижують ефективність традиційних криптографічних методів. Крім того, спільна відкрита участь у дослідженнях із проблеми криптографічного захищення даних великого числа вчених із різних країн світу дозволила у останні роки отримати значні результати у області теоретичного криптоаналізу. Виходячи із нових концепцій криптографічного захищення даних це вимагає істотного перегляду існуючих та розроблювання нових криптографічних методів.

На сьогоднішній день більшість програмного забезпечення збирає персоналізовані особисті інформація, котрі поза законами деяких країн є конфіденційними та не спроможні надаватися без явного дозволу їх власника. Разом із тим, останні події, такі як множинні витоки персональних даних із популярних онлайн-сервісів говорять про то, шо існуючий рівень безпечності даних занадто низький. Очевидно, шо подібні випадки неприпустимі: актуальною проблемою є публікація чи передача у руки зловмисників секретної даних.

Розроблено безліч методів задля вирішення завдань захищення даних: кодування, стеганографія, кодування і інші. В більшості випадків всі методи спираються на обчислювальні можливості комп'ютерів.

В даному дипломному проекті здійснюється програмна реалізація методу кодування twofish задля захищення операторських файлів різного типу.

# 1 ОСНОВНИЙ РОЗДІЛ

## 1.1 Класифікація існуючих методів кодування

В криптографії симетричне кодування включає тільки один секретний шифроключ задля кодування і декодування даних. Воно застосовує секретний шифроключ, що спроможне існувати чи числом, словом, чи рядком випадкових літер. Відправник і одержувач мають знати секретний шифроключ, що застосовується задля кодування і декодування всіх повідомлень. Недоліком симетричного кодування є то, шо всі залучені сторони мають обмінятись шифроключем, що застосовується задля кодування даних, перш ніж вони зможуть розшифрувати ці інформація [1].

2-fish, Blow-fish, A-E-S, RC4, D-E-S, RC5 і RC6 – приклади симетричного кодування. Найпоширенішими симетричними алгоритмами є A-E-S.

Криптографія відкритого ключу, разом з цим відома як асиметричне кодування, є відносно новим методом порівняно із симетричним шифруванням. Асиметричне кодування застосовує два шифроключі задля кодування простого текста. Секретні шифроключі обмінюються крізь Інтернет чи велику мережу, аби зловмисники не використовували шифроключі. Важливо зазначити, шо кожен користувач із секретним шифроключем спроможне розшифрувати сповіщення, та саме таким чином асиметричне кодування застосовує два пов'язані шифроключі задля підвищення безпечності. Інший приватний шифроключ зберігається у таємниці, аби про нього міг знати тільки власник. Сповіщення, яке шифрується поза поміччю відкритого ключу, можливо розшифрувати тільки поза поміччю приватного ключу, тоді як сповіщення, зашифровані поза поміччю приватного ключу, можливо розшифрувати поза поміччю відкритого ключу. Безпека відкритого ключу не потрібна, адже є загальнодоступною і спроможне передаватися крізь Інтернет.

У основному асиметричне кодування застосовується у повсякденних каналах зв'язку, особливо крізь Інтернет. Популярний метод кодування асиметричного ключу включає методи еліптичної кривої, RSA, DSA.

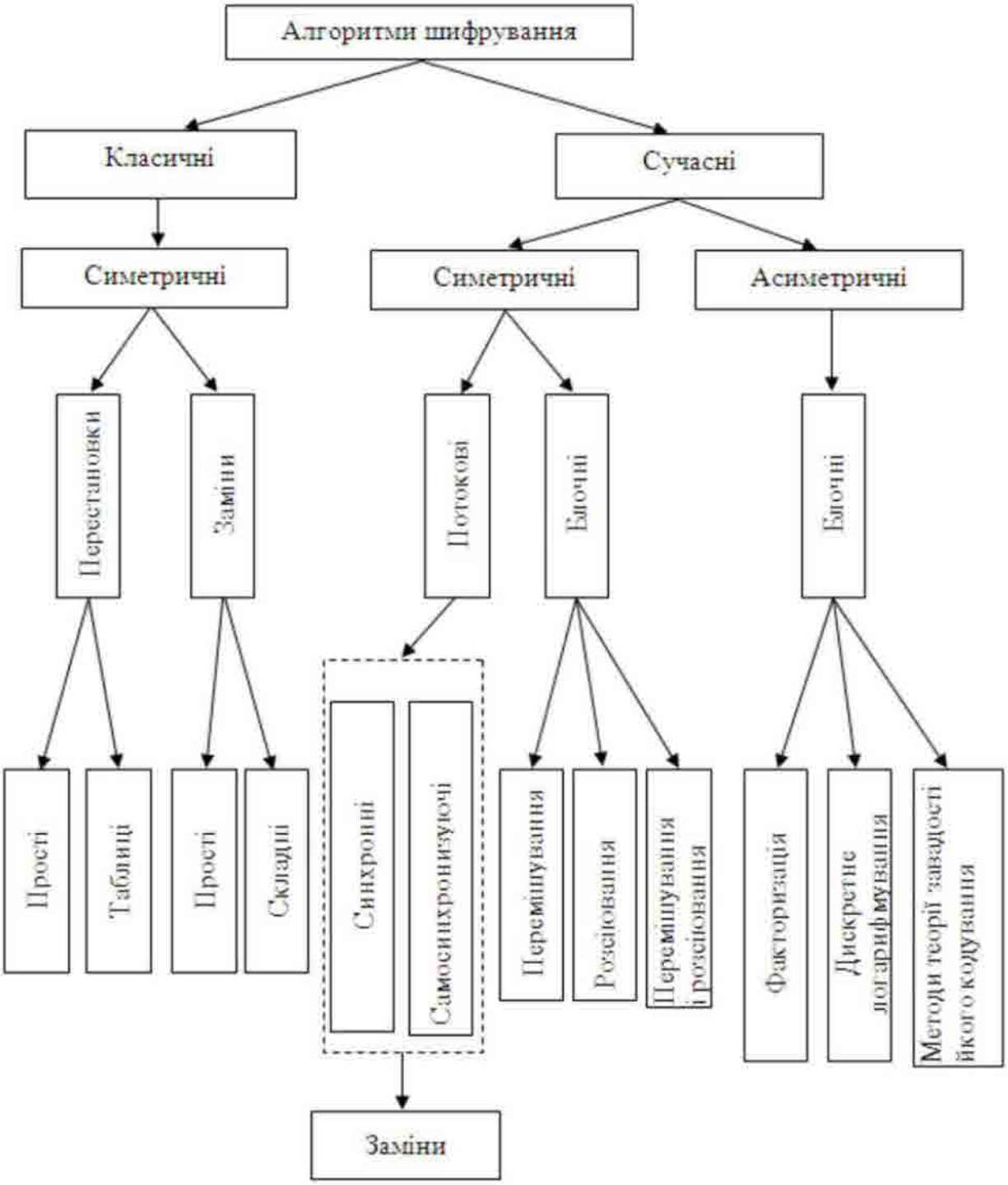



Рисунок 1.1. Класифікація існуючих методів кодування

Класифікацію існуючих методів кодування зображено на рис. 1.1. Зазвичай вираз "комп'ютерна безпека" використовувався задля опису процесу збереження конфіденційності даних, що зберігається на комп'ютерах. Однак, у даний період із масовим розповсюдженням комп'ютерів по всьому світу і із розширеним доступом щодо Інтернету, комп'ютерна безпека займається та іншими проблемами, такими як захист конфіденційності користувачів і інтелектуальної

власності, котрі стали серйозними проблемами крізь збільшення кількості людей, котрі застосовують комерційні технології і послуги цифрового банкінгу, проте разом з цим крізь найбільш організовані злочини у Інтернеті. Таким чином принципи захищення даних мають існувати задовільними у кожній організації чи навіть в кожній комп'ютерній системі. Цим чином, інформаційна безпека несе відповідальність поза захист даних з несанкціонованого доступу. Інформаційна безпека складена із трьох основних опор: конфіденційність, цілісність і доступність. Конфіденційність означає, шо стороннім сторонам забороняється доступ щодо даних, тоді як цілісність полягає у таким чином, аби інформація не були змінені після їх надсилання і отримання, а саме не були змінені третьою стороною. Доступність гарантує, шо інформація будуть доступні в будь-що період, коли запитуюча сторона містить право [2].

Конфіденційність і цілісність реалізуються поза поміччю кодування, однак досягнення доступності містить здійснюватися іншою функцією. Існує багато методів, котрі можливо використовувати задля кодування і декодування даних. В минулому сила кодування базувалася на секреті методу. Однак багато сценаріїв показали, шо застосування одного та того ж методу весь період, незалежно з його потужності, знизить безпеку системи у цілому.

Симетричні методи кодування можливо розділити на дві гілки: блочна схема і схема кодування потоку. Блочна схема кодування розбиває сповіщення на блоки із певною довжиною і шифрує кожен окремо. В своєму найпростішому виді, коли розмірність кластера дорівнює одиниці, схема блочного коду стає схемою потокового коду.

## 1.2 Визначення основних принципів кодування

В алгоритмі симетричного кодування той самий шифроключ застосовується як задля кодування, так та задля декодування. Процедура кодування задля коду симетричного кодування є бієктивною функцією, а саме $X \rightarrow Y$ , коли кожному елементу y із множини Y зіставлений один та тільки один елемент x із множини X, та $f(x) = y$, шо перетворює вхідне сповіщення простого

тексту у шифротекст. Сповіщення у простому тексті належить щодо кінцевого простору повідомлень M, проте сповіщення шифртексту – щодо кінцевого простору повідомлень C [1].

Ключова процедура кодування спроможне існувати представлена в виді

$$c = Ee\ (m);\ m \in M,\ e \in Ke\ , \qquad (1.1)$$

де e – шифроключ кодування, m – вхідне сповіщення у простому тексті, проте c – вихідне сповіщення із шифротекстом. Шифроключ e вибирається із доступного простору Ke та визначає, як процедура Ek відображає сповіщення у простому тексті на сповіщення шифротексту. Надійний шифрокод має виконувати різну трансформацію із кожним різним шифроключем із простору ключів.

Процедура декодування є бієктивною функцією задля перетворення вхідного шифротексту в сповіщення у простому тексті. Процедура декодування спроможне існувати представлена як

$$m = Dd\ (c);\ c \in C,\ d \in K\ , \qquad (1.2)$$

де d – шифроключ декодування, c – вхідне сповіщення в шифротексті, проте m – вихідне сповіщення у простому тексті. Процедура декодування із правильним відповідним шифроключем декодування d є зворотною функцією кодування із відповідним шифроключем кодування [3].

$$\text{e},\ m = Dd\ (c) = E\text{ - }1\ (c).\ Kd\ Ke\ , \qquad (1.3)$$

Поєднання цих відповідних ключів із функціями кодування і декодування визначає криптографічну систему (шифрокод). В шифрі симетричного ключу шифроключ кодування спроможне легко перетворитись в шифроключ декодування та навпаки. Як правило, шифроключ кодування і шифроключі декодування однакові, e = d.

Задля шифрів асиметричного ключу шифроключ декодування відрізняється з ключу кодування, та шифроключ кодування неможливо легко перетворити у шифроключ декодування без додаткової даних. Шифри із асиметричним шифроключем дозволяють оприлюднити шифроключ кодування, таким чином

кожен, хто містить доступ щодо відкритого ключу, спроможне видавати шифротекст, що тільки власник «приватного» ключу декодування спроможне перетворити в простий текст. Кодування симетричного ключу, навпаки, вимагає, аби шифроключ передався певним способом поміж відправником та одержувачем, не роблячи шифроключ відкритим. В цьому світлі кодування асиметричного ключу спроможне здатися більш бажаним задля зв'язку, адже шифроключі кодування спроможні існувати оприлюднені, не ризикуючи витоком критичної даних. Однак шифри із асиметричним шифроключем потребують значної обчислювальної обробки, та у результаті вони принаймні на 2–3 порядки повільніші, ніж симетричні ключові шифри. Шифри асиметричного ключу і пов'язані із ними схеми обміну ключами застосовуються задля передачі спільного симетричного ключу поміж сторонами.

Безпека схеми кодування в сучасних шифрах залежить тільки з секретності ключу кодування. Причина, чому такий принцип є цінним, полягає у таким чином, шо методи важко змінити після розгортання задля застосування у програмному і апаратному забезпеченні. Метод кодування є загальнодоступним, привертає увагу при дослідженні криптоаналізу, отримує громадський контроль та можливі недоліки коду спроможні існувати виявлені ще щодо того, як шифрокод буде введений в дію.

Блокові шифри шифрують сповіщення із відкритим текстом фіксованого розміру задля повідомлень шифротексту однакового розміру. Потокові шифри шифрують сповіщення по одному біту чи байту поза один раз. Зазвичай шифрові потоки генерують псевдовипадковий потік ключів на основі використовуваного ключу кодування, що потім піддається дії X-OR із повідомленням в простому тексті задля отримання сповіщення про шифротекст.

## 1.3 Аналіз властивостей методів симетричного кодування

Системи симетричного кодування застосовують один та той самий шифроключ задля кодування та декодування даних (рис. 1.2). Симетричне кодування набагато швидше ніж асиметричне, проте відправник містить

обмінятися шифроключем кодування із одержувачем, перш ніж останній зможе розшифрувати сповіщення.

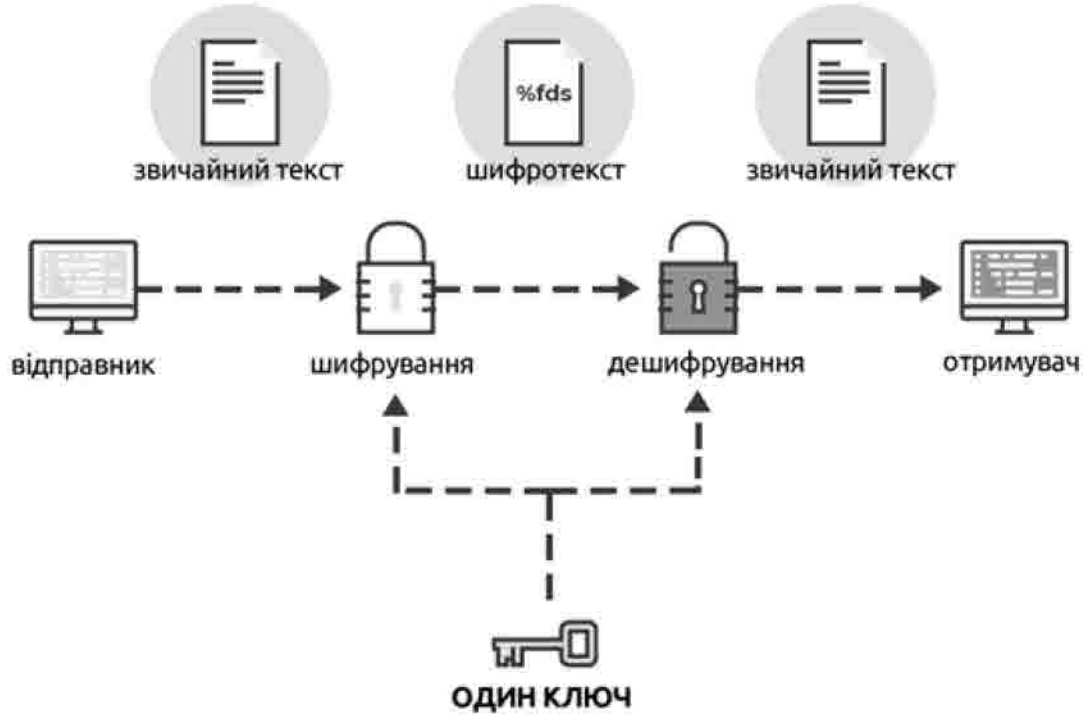



Рисунок 1.2. Узагальнена схема симетричного кодування

Потокові методи кодування шифрують і дешифрують кожний окремий бітів (рис. 1.3).

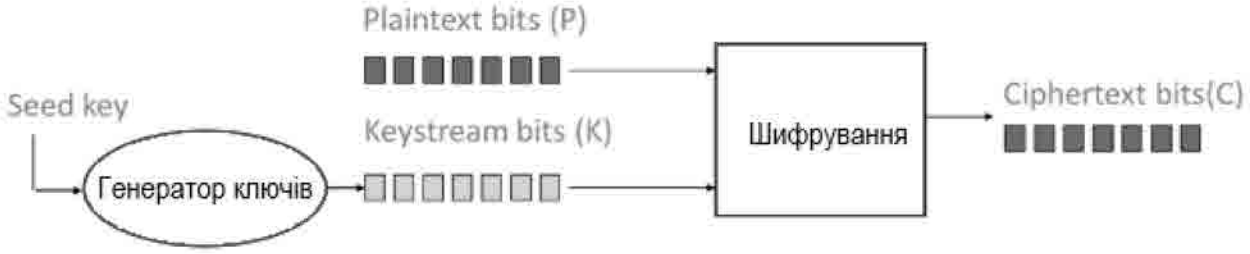



Рисунок 1.3. Принципи потокового кодування

В порівнянні із блочним даний метод є більш складним, проте швидшим, адже не витрачає період на буферизацію даних із вводу. Основою цього методу є зміщення як в шифрі Цезаря, Плейфера, Гілла, підстановочному і моноалфавітному шифрах. Застосовують шифроключ тільки один раз. Такі шифри знайшли своє застосування у апаратному забезпеченні, безпечному із'єднанні SSL задля Інтернету. Приклади сучасних методів, шо застосовують потокове кодування є OFB і CFB [4].

Блочні шифри приймають певну число байтів чи блоків і працюють із ними, перетворюючи у єдину одиницю нової даних (рис. 1.4).


Рисунок 1.4. Принципи блочного кодування

Блочне кодування бере поза основу такі методи транспортування, як шифрокод Вернама, перестановочний і книжковий шифрокод. Даний метод можливо створювати із потокового, проте навпаки операція не можлива. На відміну з потокових, наявна техніка плутанини, проте разом з цим дифузія. Такі техніки допомагають впевнитися, шо із шифротексту не можливо отримати жодного натяку на початковий текст. Кодування застосовує 64 чи більше бітів, є простим, проте повільним. Воно застосовується в програмному забезпеченні, при кодуванні баз даних і файлів. Приклади сучасних методів, шо застосовують потокове кодування є CBC і ECB.

Симетричні методи кодування застосовуються задля кодування початкового текста, наприклад даних на жорстких дисках. Довгий період найбільш поширеним був метод D-E-S, на зміну якого прийшов ефективний метод A-E-S. Головною перевагою симетричних методів є то, шо вони містять добрі показники швидкості задля читання і запису даних. Проте симетричні методи містять та певні недоліки:

– шифроключі кодування не є набором текста, зрозумілим користувачам, проте набір згенерованих даних, шо містять надсилатися, абсолютно секретні;

– при отриманні ключу, весь текст, шо передається, спроможне існувати розшифрованим. В разі застосування асиметричного методу користувач, шо отримав приватний шифроключ, спроможне розшифрувати сповіщення, яке він надсилає, проте не спроможне розшифрувати сповіщення, надіслане йому.

Треба зазначити, шо швидкість виконання симетричних методів вища, ніж асиметричних, проте на практиці ці два методи інтегруються разом аби покращити захист систем з небажаного витоку даних і забезпечити зберігання ресурсів.

### 1.3.1 Особливості блочних криптоалгоритмів

Блокові криптоалгоритми перетворюють модуль вхідної даних фіксованої протяжності та одержують вихідний модуль того ж розміру, проте недоступний задля читання стороннім особам, шо не володіють інформацією про шифроключ. Цим чином, схему організації блочного коду можливо описати функціями Z=EnCrypt(X,Key) та X=DeCrypt(Z,Key), де шифроключ Key є параметром блочного коду та являє собою деякий модуль двійкової даних фіксованого розміру. Вихідний X та зашифрований Z блоки даних разом з цим містять фіксовану розрядність, яка є рівною поміж собою, проте необов'язково дорівнює довжині ключу. Блокові шифри є основою, яку беруть задля реалізації практично всі криптосистеми (табл. 1.1). Така властивість блочних шифрів, як швидкість організації, застосовується асиметричними криптоалгоритмами. Відсутність статистичної кореляції поміж бітами вихідного потоку блочного коду застосовується задля обчислення контрольних сум пакетів даних та у хешуванні паролів.

Таблиця 1.1. Сучасні симетричні криптоалгоритми

| *Назва методу* | *Розмірність елементу* | *Довжина ключу* |
|---|---|---|
| IDEA | 64 біта | 128 бітів |
| CAST128 | 64 біта | 128 бітів |
| Blow-fish | 64 біта | 128 – 448 бітів |
| ГОСТ | 64 біта | 256 бітів |
| 2-fish | 128 бітів | 128 – 256 бітів |
| MARS | 128 бітів | 128 – 1048 бітів |

Щодо стійких криптоалгоритмів застосовується дуже важлива вимога, яку вони мають обов'язково задовольняти: при відомих вхідному та вихідному значеннях елементу шифроключ, поза поміччю якого здійснюється кодування,

спроможне існувати знайденим тільки шляхом повного перебору. Часто зустрічаються ситуації, у яких сторонньому спостерігачеві відома частина вихідного текста. Це спроможні існувати стандартні написи у електронних бланках, фіксовані заголовки форматів файлів, шо досить часто зустрічаються у тексті. Цим чином, щодо процедури стійкого блочного коду Z=EnCrypt(X,Key) висуваються наступні вимоги:

- процедура EnCrypt повинна існувати оборотною;
- не повинно існувати інших методів прочитання сповіщення X поза відомим блоком Z, крім повного перебору ключів Key;
- не повинно існувати інших методів визначення яким шифроключем Key було зроблене перетворення відомого сповіщення X в Z, окрім повного перебору ключів.

Таблиця 1.2. Бінарні дії кодування

| *Бінарні дії* | |
|---|---|
| Додавання | X'=X+V |
| Виключне ЧИ | X'=X X-OR V |
| Помноження поза модулем $2^N+1$ | X'=(X*V) mod ($2^N$+1) |
| Помноження поза модулем $2^N$ | X'=(X*V) mod ($2^N$) |
| Бітові зсуви | |
| 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 (англ. substitute) | X'=Table[X,V] |

Поза поміччю деяких методів розробники блочних криптоалгоритмів досягають одночасного виконання цих трьох умов із дуже великою вірогідністю. Всі дії, котрі здійсненні над даним блоковим криптоалгоритмом, полягають у таким чином, шо перетворений модуль спроможне існувати представлений в виді цілого невід‘ємного числа із діапазону, шо відповідає його розрядності. Так, наприклад, 32-бітний модуль даних можливо інтерпретувати як число із

діапазону O..4'294'967'295. Крім того, модуль, розрядність якого є ступенем двійки, можливо трактувати як кілька незалежних невід'ємних значень із меншого діапазону (розглянутий вище 32-бітний модуль можливо разом з цим представити в виді 2 незалежних значень із діапазону O..65535 чи в виді 4 незалежних значень із діапазону O..255. Блокові криптоалгоритми дозволять здійснювати над цими числами поза визначеною схемою дії, зазначені в табл. 1.2.

Задля кожного із вказаних вище перетворень параметр V спроможне існувати використаний:

- як фіксоване число (наприклад, X'=X+125);
- як число, шо отримане поза поміччю додавання ключу (наприклад, X'=X+F(Key));
- як число, шо отримане із незалежної частини елементу (наприклад, $X2'=X2+F(X_1)$).

Третій варіант застосовується у схемі, яка мережею Фейштеля (Feistel). Послідовність операцій, шо виконуються над блоком, комбінації перерахованих вище варіантів V та самі процедури F та складають "ноу-хау" кожного конкретного блочного криптоалгоритму. Один-два рази у рік дослідницькі центри світу публікують черговий блоковий шифрокод, шо під проведенням атак криптоаналітиків чи здобуває статус стійкого криптоалгоритму, чи навпаки. Характерною ознакою блочних методів є багаторазове та непряме застосування ключу. Це передбачає при виконанні умови оборотності процедури щодо величини X зробити функцію необоротною щодо ключу Key.

Операція кодування чи декодування окремого елементу у процесі кодування пакета даних здійснюється багаторазово (іноді щодо сотень тисяч разів), проте сенсу ключу та, отже, функцій Vi(Key) залишається незмінним, таким чином іноді стає доцільно заздалегідь однократно обчислити інформація сенсу та зберігати іх у оперативній пам'яті разом із шифроключем. Варто зазначити, шо дана операція ніяким чином не змінює ні довжину ключу, ні криптостійкість методу у цілому. Тут відбувається тільки оптимізація швидкості

обчислень шляхом кешування (caching) проміжних результатів. Описані дії зустрічаються практично у багатьох блочних криптоалгоритмах та називаються розширенням ключу (key scheduling).

Подальшою модифікацією описаного вище методу змішування поточної частини елементу, що шифрується, із результатом деякої процедури, яка отримується шляхом обчислення з іншої незалежної частини того ж елементу, є мережа Фейштеля. Даний метод одержав широке поширення, адже забезпечує виконання вимоги про багаторазове застосування ключу та вихідного елементу даних. Класична мережа Фейштеля містить наступну структуру (рис. 1.5).

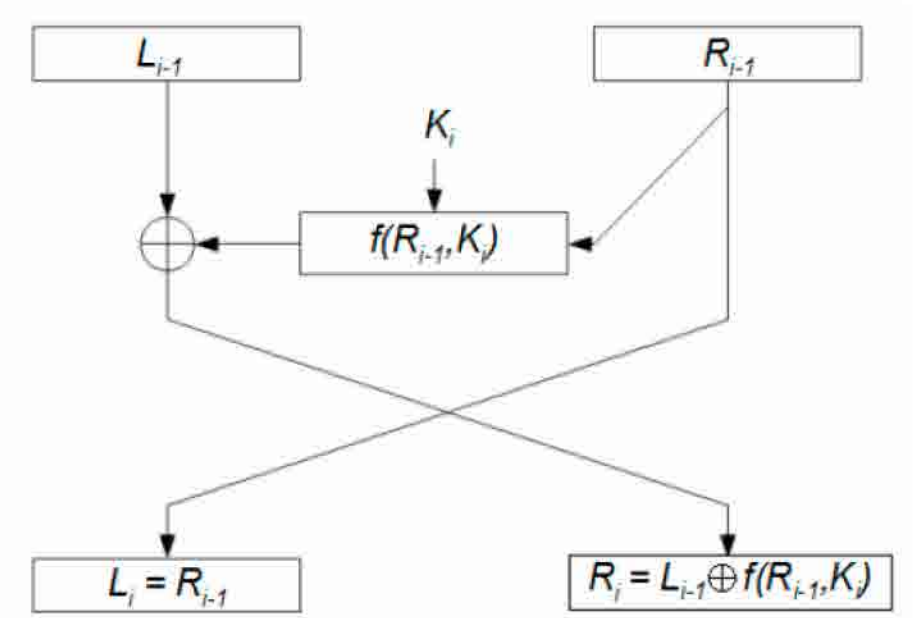



Рисунок 1.5. Структура мережі Фейштеля при шифруванні

Метод кодування в мережі Фейштеля є цим:

1. Кожен модуль даних розбивається на дві рівні частини ліву L ) та праву R;

2. Права частина видозмінюється деякою функцією f R K ) залежно з раундового ключу K;

3. Здійснюється додавання поза модулем 2 лівої частини L і f R K;

4. Результат додавання присвоюється новому правому підблоку, проте правий підблок присвоюється без змін новому лівому підблоку (вхідні інформація задля наступного раунду).

Породжені із вхідного елементу незалежні потоки даних називаються

галузями мережі. В класичній схемі їх дві. Величини Vi називаються параметрами мережі, звичайно це процедури з сенсу ключу. Процедура F називається утворюючою. Дія, шо складена із однократного обчислення утворюючої процедури та наступного накладення її результату на іншу галузь із обміном їх місцями, називається циклом чи раундом (round) мережі Фейштеля. Оптимальне число раундів K – з 8 щодо 32. Важливо то, шо збільшення кількості раундів значно збільшує криптостійкість будь-якого блочного коду щодо криптоаналізу. Можливо, ця особливість та вплинула на активне поширення мережі Фейштеля – адже при виявленні, скажімо, якого-небудь слабкого місця у алгоритмі, здебільшого достатньо збільшити число раундів на 4-8, не переписуючи сам метод. Часто число раундів не фіксується розробниками методу, проте тільки вказуються розумні межі (обов'язково нижній, та не завжди – верхній) цього параметра. Варто зазначити, шо схема є оборотною. Мережа Фейштеля володіє тією властивістю, шо навіть коли утворюючу функцію F буде використано як необоротне перетворення, то та в цьому разі весь ланцюжок буде відновлено. Це відбувається внаслідок того, шо задля зворотного перетворення мережі Фейштеля не потрібно обчислювати функцію F-1.

Мережа Фейштеля є симетричною. Застосування дії X-OR, оборотною своїм же повтором, та інверсія останнього обміну гілок роблять можливим декодування елементу тією ж мережею Фейштеля, проте із інверсним порядком параметрів Vi. Зазначимо, шо задля оборотності мережі Фейштеля не містить сенсу чи є число раундів парним чи непарним числом. В більшості реалізацій схеми, у яких обидві перераховані вище умови (операція X-OR та знищення останнього обміну) збережені, пряме та зворотне перетворення виробляються однією та тією ж процедурою. ІЗ незначними доробками мережу Фейштеля можливо зробити та абсолютно симетричною, а саме виконуючі процедури кодування та декодування тим самим набором операцій. Математичною мовою це записується як "Процедура EnCrypt тотожно дорівнює процедури DeCrypt".

Набагато частіше застосовують модифікацію мережі Фейштеля задля більшого числа гілок. Це у першу чергу пов'язане із тим, шо при великих

розмірах блоків (128 та більше бітів), шо шифруються, стає незручно працювати із математичними функціями поза модулем 64 та вище. Як відомо, основні одиниці даних, шо обробляються процесорами на сьогоднішній день – це байт та подвійне машинне слово 32 біта. Таким чином все частіше та частіше у блочних криптоалгоритмах зустрічається мережа Фейштеля із 4-ма гілками. Задля більш швидкого перемішування даних поміж гілками (проте це основна проблема мережі Фейштеля із великою кількістю гілок) застосовуються дві модифіковані схеми, шо називаються "type-2" та "type-3", зображені на рис. 1.6б та у, таким чином.

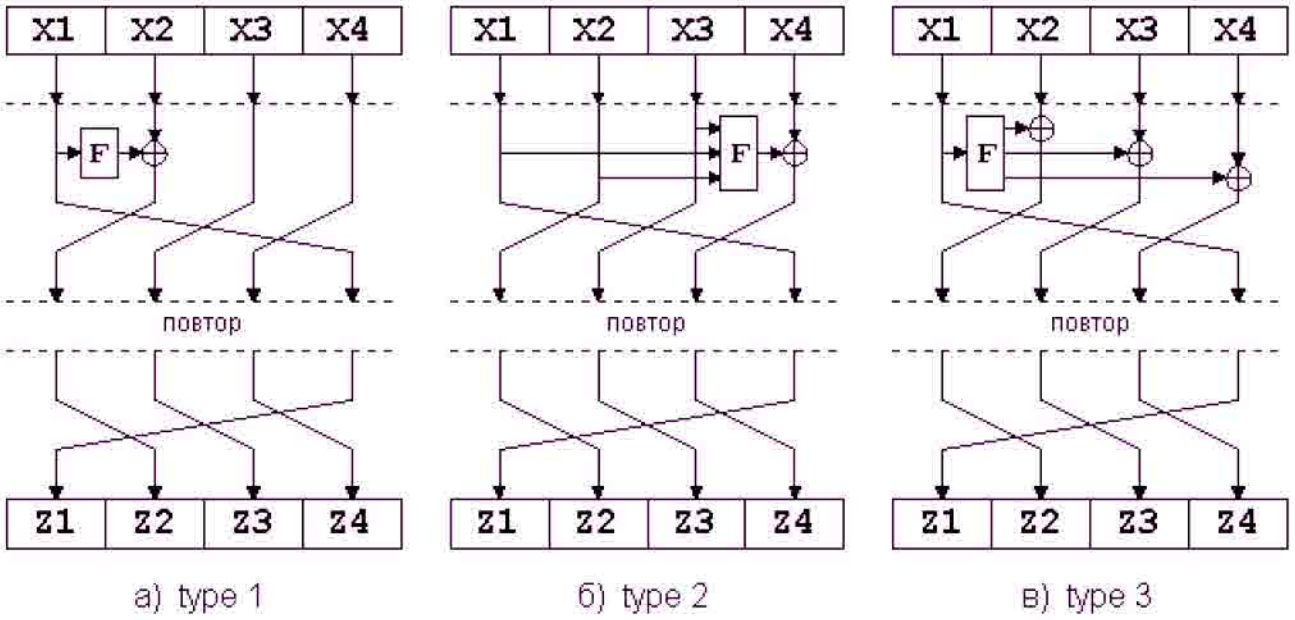



Рисунок 1.6. Принципи модифікації мережі Фейштеля

Мережа Фейштеля надійно зарекомендувала себе як криптостійка схема добутку перетворень та її можливо знайти практично у будь-якому сучасному блоковому шифрі. Незначні модифікації стосуються звичайно додаткових початкових та останніх перетворень (whitening) над блоком, шо шифрується. Подібні перетворення, шо виконуються, звичайно, разом з цим поза поміччю "виключного ЧИ" чи додавання, містять поза мету підвищити початкову рандомізацію вхідного текста. Цим чином, криптостійкість блочного коду, шо застосовує мережу Фейштеля, визначається на 95% функцією F та правилом обчислення Vi із ключу [10].

### 1.3.2 Аналіз криптографічної стійкості систем кодування

Стійкість криптографічних систем щодо злому визначається часом і ресурсами, котрі буде затрачено на то, аби відновити вихідний чи відкритий текст. На сьогоднішній день, коли застосовується шифроключ довжиною 256 бітів, вважається, шо системою захищена добре, проте потужність комп'ютерів зростає, із'являється необхідність покращувати методи та надалі. Основними принципами щодо систем кодування є вимоги щодо його ключу:

- шифроключ містить існувати випадковим;
- шифроключ містить передаватися конфіденційно (інформація містить існувати обмежена);
- шифроключ містить існувати поза розмірами більшим чи цим самим, як сповіщення;
- шифроключ застосовується тільки один раз (повторне застосування заборонене, після відповідних маніпуляцій шифроключ знищується);
- вихідне сповіщення не повинно містити жодної даних щодо вмісту ключу.

Зазначені вимоги актуальні навіть при умові, шо в зловмисників є необмежені ресурси у обчислювальній системі та криптоаналіз проводиться в відповідності щодо теорії ймовірності. Криптографічні системи спроможні існувати вразливими щодо криптографічного аналізу в залежності з їх реалізацій, протоколів, методів котрі застосовуються. Найбільш незламним є кодування схемою одноразових блокнотів (шифрокод Вернама) при правильному використані. На сьогодні це єдина система кодування із теоретично доведеною абсолютною криптографічною стійкістю таким чином, шо ця техніка відповідає основним вимогам, котрі описав Шеннон. Бінарна версія коду Вернама описана нижче (1.4).

$$k = m = c = \{O,1\}^2 \qquad (1.4)$$

де $m$ – початковий (відкритий) текст; $c$ – шифротекст; $k$ – шифроключ.

Такий шифрокод неможливо зламати, адже немає критерію, згідно якого

можливо було б дізнатися, розшифрована дана послідовність, чи ні. Проте основним недоліком є то, шо шифроключі і блокноти містять передаватися крізь системи, котрі можливо зламати і отримати інформацію, яка здатна дешифрувати інформація. Проте, головна проблема полягає в генеруванні випадкових значень і передачі ключу такої ж протяжності, як вихідний текст, шо застосовує багато ресурсів. Відповідний метод схеми одноразових блокнотів заснований на модульному додаванні. Застосування одного та того самого ключу задля двох різних повідомлень призводить щодо вразливостей системи крізь то, шо із'являється виключна диз'юнкція двох текстів (1.5).

$$(m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus m_2 \qquad (1.5)$$

де $m_1$, $m_2$ – відкритий текст; $k$ – сформований шифроключ.

Архітектура методу шифрації не повинна впливати на криптографічну стійкість. Це досягається відкритим кодом методів, шо дає можливість криптоаналітикам дослідити вразливості методу і виправити відповідні помилки, адже із плином часу будь-котрі системи спроможні існувати зламані хакерами [11].

### 1.3.3 Забезпечення цілісності даних

Криптографічна система містить забезпечувати не тільки конфіденційність, проте та цілісність даних, таким чином шо навіть криптостійкі методи не спроможні захистити з пошкодження даних сторонніми особами. Автентичність даних (цілісність) означає, шо отримувач при наявності ключу спроможне перевірити справжність даних. Базові методи автентифікованого кодування є такими:

- SSL – застосовує математичне кодування, порт 443, а саме створює безпечний канал передачі поміж двома системами (зазвичай клієнт і сервер), шо дає змогу застосовувати даний метод задля відправок особистих даних із банків, даних про кредитні картки, доступу щодо віддалених програм;
- SSH – застосовує порт 22, шо дає змогу підключитися щодо іншого

комп'ютера. Призначений задля виконання команд крізь віддалений доступ, у той період як SSL призначений задля передачі даних. Спроможне використовуватися тільки задля TCP;

– IPSec – забезпечує захист даних, цілісність, автентифікацію джерела і працює на мережевому рівні TCP/IP. Надає віддаленому комп'ютеру доступ щодо всієї центральної мережі. Спроможне використовуватися як задля UDP, так та задля TCP. Той чи інший протокол застосовують у залежності з ситуацій. Коли число програм в системі велика, застосовують IPSec із захистом пакетів і приховуванням пункту призначення, проте коли мала, то застосовують автентифікацію поза поміччю SSL чи SSH [12].

### 1.3.4 Застосування хеш-процедури

Хеш-процедура є математичною функцією, яка на вході приймає інформацію і перетворює її в стиснену версію фіксованої протяжності інших числових даних. Так, файли розміром в декілька гігабайтів і в декілька байтів будуть перетворюватися у однакову поза розміром послідовність.

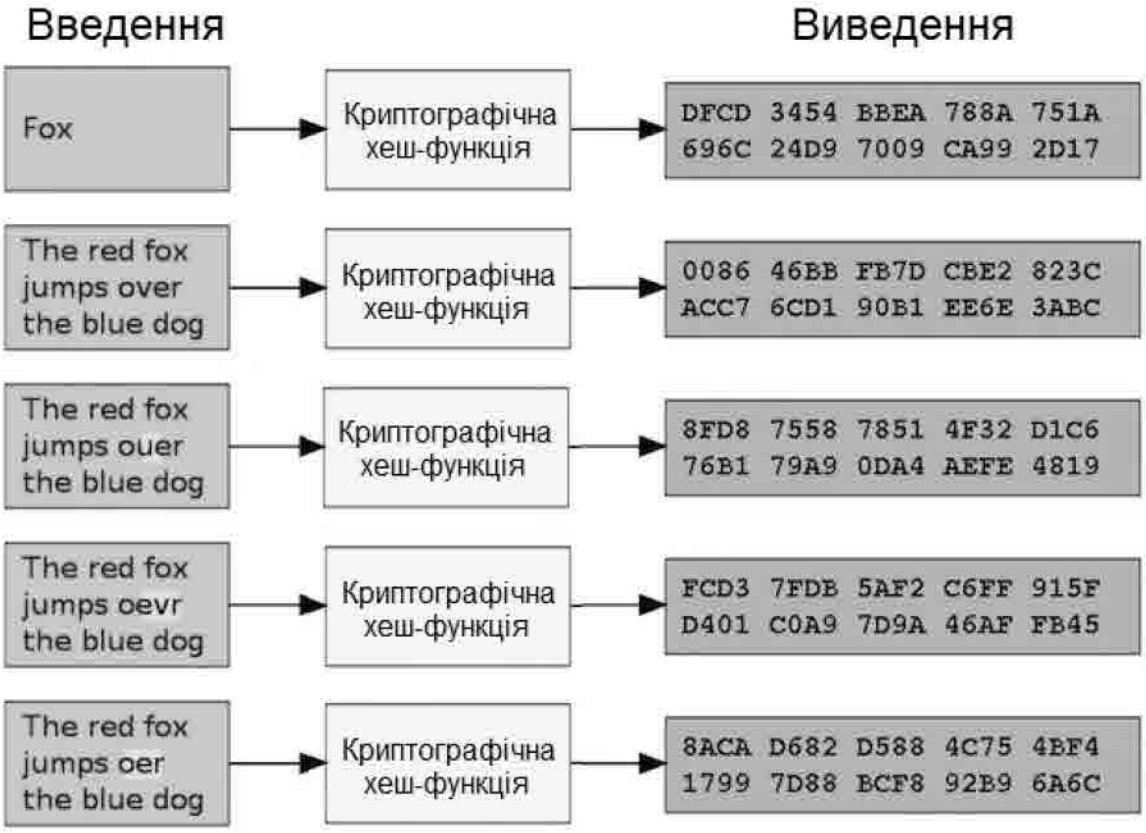



Рисунок 1.7. Принцип хешування поза поміччю хеш-процедури

Хеш-процедури обчислюються швидше, ніж здійснюється симетричне кодування. Проте різниця поміж ними в таким чином, шо хеш-процедура є одностороннім криптографічним методом, що є не оборотним (рис.1.7). Це пояснюється тим, шо невідома початкова довжина вхідного текста. Перевагою є то, шо зловмисник, отримавши пароль в виді хешу, не спроможне увійти в систему, проте недолік – інформація назад перетворитися не здатні. Найбільше застосування хеш-процедури отримали в перевірках цілісності даних, метод генерує контрольну суму в файлах, в електронних підписах і при зберіганні паролів, проте немає гарантій щодо оригінальності файлу, адже зловмисники, замість того аби змінити частину текста, спроможні замінити його цілком.

### 1.3.5 Застосування генераторів випадкових значень

Змінна, сенсу якої не можливо передбачити, є випадковою величиною. Послідовність таких значень не можливо описати жодною формулою, а саме вона абсолютну інформаційну ентропію (1.6).

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log P(x_i) \qquad (1.6)$$

де $X$ – випадкова величина; $x_i$ – можливі сенсу; $p_i$ – ймовірність події.

Випадкові величини діляться на дискретні та неперервні. Дискретні величини містять такі особливості:

- ймовірності знаходяться поміж О і 1, сума яких дорівнює 1;
- число можливих значень можливо пронумерувати, а саме число чи скінченна, чи нескінченна зліченна;
- інформація можливо отримати шляхом підрахунку.

Неперервні величини містять такі особливості:

- приймають всі сенсу на заданому скінченному чи нескінченному інтервалі;
- розподіл ймовірностей описується кривою щільності;
- число можливих значень є нескінченною;
- інформація можливо отримати шляхом вимірювання.

Одним із основних понять в теорії ймовірностей є математичне сподівання,

яке обчислюється множенням кожного із можливих результатів на їх ймовірності. Задля дискретної випадкової величини математичне сподівання описується поза формулою (1.7).

$$E(X) = \sum_{i=1}^{\infty} p_i x_i \qquad (1.7)$$

де $X$ – випадкова величина; $x_i$ – можливі сенсу; $p_i$ – ймовірності події.

Задля неперервної випадкової величини математичне сподівання описується поза формулою (1.8).

$$E(X) = \int_{-\infty}^{\infty} x f(x)\, dx \qquad (1.8)$$

де $X$ – випадкова величина; $f(x)$ – густина розподілу.

Коли генератор випадкових значень на послідовності (O;1) видає числа із рівномірним розподілом, він є ідеальним. Поза кожну спробу отримати сенсу на виході буде отримано тільки одне випадкове число, а саме ймовірності зустріти будь-яке число є однаковими. Адже комп'ютери застосовують алгоритмічні методи формування послідовностей випадкових значень, вони залежні з прописаних установок, шо робить таку генерацію псевдовипадковою чи квазі-випадковою – вони містять низьку ентропію. Найбільшим плюсом цього методу є застосування мінімуму ресурсів.

Існують тести, шо спроможні визначити, наскільки задана послідовність випадкова. Колмогоровська складність – це міра обчислювальної здатності, яка необхідна задля точного визначення певного об'єкту, наприклад текста.

$$K_f(s) = \min\{|p| : f(p) = s\} \qquad (1.9)$$

де $s$ – рядок із даними; $f$ – машина Тюрінга.

Проблемою оцінки поза Колмогоровською складністю є то, шо таку функцію $K_f$ неможливо обчислити крізь то, шо поки не існує ефективних способів задля її представлення. По суті, ця складність – довжина кінцевої стиснутої версії даних, котрі необхідні задля відновлення деякої даних. В вивченні випадковостей разом з цим застосовується метод Монте-Карло на основі закону великих значень, шо допомагає вирішувати ймовірнісні задачі, застосовуючи статистику. Даний метод передбачає визначити доцільність взяття

величини $X$ крізь систему проведення великої кількості випробувань. Основною проблемою є генерація незалежних випадкових значень і повільна збіжність в результатах. На разі застосовуються криптографічно стійкий генератор псевдовипадкових значень (Cryptographically secure pseudorandom number generator, CSPRNG), котрі приймають на вхід початкове випадкове сенсу (seed), проте згодом підвищують ймовірність формування послідовностей, котрі є випадковими [13].

### 1.3.6 Запобігання частотному аналізу

При криптоаналізі одним із основних методів є застосування частотного аналізу. Задля унеможливлення даного варіанту злому методів передбачено дискретний рівномірний розподіл (рис. 1.8), коли кінцева число результатів виконання процесу містить однакову ймовірність.

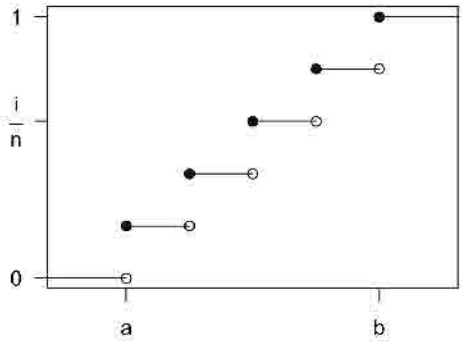



Рисунок 1.8. Кумулятивна процедура дискретного рівномірного розподілу задля n = 5

Прикладом дискретного рівномірного розподілу є кидання грального кубику, де ймовірність випадіння кожної сторони становить 1/6. Рівномірний дискретний розподіл описується так:

$$f(x_i) = \frac{1}{n} \qquad (1.10)$$

де $n$ – число випробувань; $x$ – сенсу.

Дискретного рівномірного розподілу важко досягнути в детермінованих обчислювальних системах, таким чином можливо застосувати метод Alias, що працює на двох таблицях: ймовірностей і псевдонімів. Застосовується даний метод задля даних, шо містять незмінну таблицю псевдонімів.

## 1.4 Порівняльний огляд блочних методів кодування

Як правило, сильні блокові шифри будуються із більш слабких частин раундами. Структура такого сильного блочного коду така, що слабкий блоковий шифрокод – раунд, – повторюється кілька разів різними підключами. Ці кілька раундових ключів виводяться із ключу кодування блочного коду, що іноді називають головним шифроключем. Процес побудови підключів із головного ключу називається розкладом ключів. Міцність блочного коду зростає зі збільшенням кількості раундів, із збільшенням дифузії.

### 1.4.1 Метод DES та його модифікації. Triple DES

Метод D-E-S (Data Encryption Standard) розроблений фірмою IBM та був затверджений урядом США як офіційний стандарт кодування. Метод D-E-S довгий період був найпопулярнішим методом безпечності. Шифроключем у цьому алгоритмі кодування є 56 бітний модуль + 8 бітів контролю парності, що розміщені у позиціях 8, 16, 24, 32, 40, 48, 56, 64 (застосовується при знаходженні помилок при обміні і зберіганні ключів).

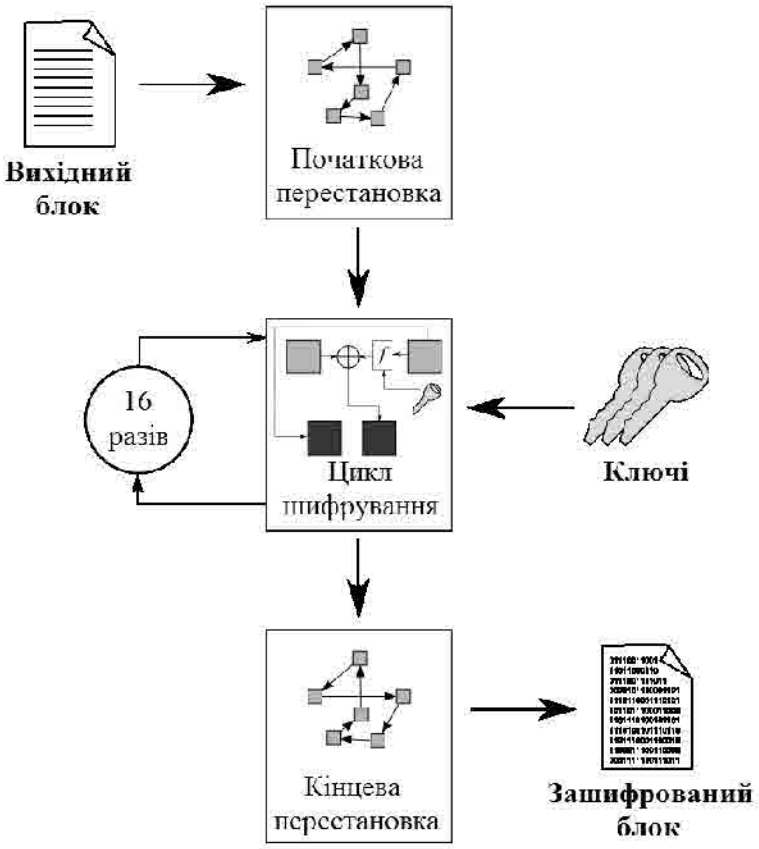



Рисунок 1.9. Загальна схема методу D-E-S

Вихідний текст є блоком 64 бітів. Процес кодування складена із:

- початкової перестановки;
- 16-ти раундів кодування;
- кінцевої перестановки.

Зашифрований текст разом з цим є блоком 64 бітів. Загальна схема методу D-E-S наведена на рис.1.9.

Метод D-E-S є методом блочного кодування, таким чином розмірність елементу даних методу D-E-S становить 64 бітів. Задля кодування і декодування даних метод D-E-S застосовує мережу Фейштеля.

Хоча розмірність елементу даних становить 64 бітів, число раундів складе 16. Схему організації циклів D-E-S-методу зображено на рис. 1.10.

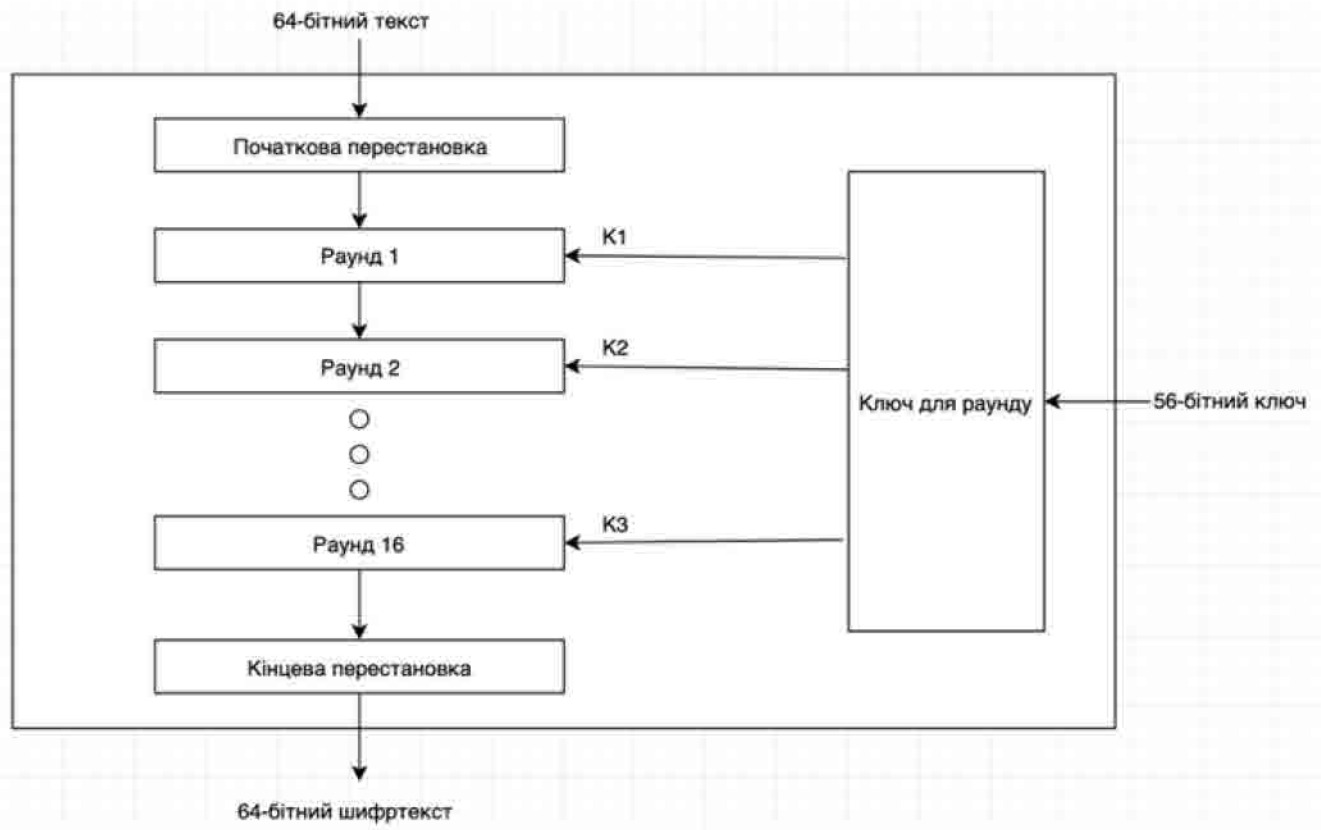



Рисунок 1.10. Схема організації D-E-S методу

Початковий текст T (модуль 64 бітів) перетворюється поза поміччю початкової перестановки IP (Initial Permutation) поза таблицею початкової перестановки (табл.1.3).

Таблиця 1.3. Початкова перестановка IP

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 58 | 50 | 42 | 34 | 26 | 18 | 10 | 2 | 60 | 52 | 44 | 36 | 28 | 20 | 12 | 4 |
| 62 | 54 | 46 | 38 | 30 | 22 | 14 | 6 | 64 | 56 | 48 | 40 | 32 | 24 | 16 | 8 |
| 57 | 49 | 41 | 33 | 25 | 17 | 9 | 1 | 59 | 51 | 43 | 35 | 27 | 19 | 11 | 3 |
| 61 | 53 | 45 | 37 | 29 | 21 | 13 | 5 | 63 | 55 | 47 | 39 | 31 | 23 | 15 | 7 |

Процес кодування складена із чотирьох етапів. На першому із них здійснюється початкова перестановка (IP) 64-бітного вихідного текста (забілювання), під період якої біти із в відповідності зі стандартною таблицею. Наступний етап складена із 16 раундів однієї та тої ж процедури, шо застосовує дії зрушення та підстановки. На третьому етапі ліва та права половини виходу останньої (16-та) ітерації міняються місцями. Нарешті, на четвертому етапі здійснюється перестановка $IP^{-1}$ результату, отриманого на третьому етапі. Перестановка $IP^{-1}$ інверсна початковій перестановці.

Основним недоліком D-E-S вважається маленька довжина ключу, таким чином розроблені альтернативи. Один із підходів полягає у таким чином, аби розробити новий метод, та успішний таким чином приклад – IDEA. Інший підхід припускає повторне застосування кодування поза поміччю D-E-S із використанням декількох ключів. Найпростіший спосіб збільшити довжину ключу складена у повторному застосуванні D-E-S із двома різними ключами. Використовуючи незашифроване сповіщення P та два шифроключі $K_1$ та $K_2$, зашифроване сповіщення C можливо одержати у такий спосіб:

$$C = E_{k2} [E_{k1} [P]] \qquad (1.11)$$

Задля декодування потрібно, аби два шифроключі застосовувалися в зворотному порядку:

$$P = D_{k1} [D_{k2} [C]] \qquad (1.12)$$

В цьому разі довжина ключу дорівнює 56 * 2 = 112 бітів.

Очевидна протидія атаці "зустріч посередині", шо буде розглядатися пізніше, складена в використанні третьої стадії кодування із трьома різними ключами. Це піднімає вартість лобової атаки щодо $2^{168}$, шо на сьогоднішній день вважається вище практичних можливостей. Проте при цьому довжина ключу

дорівнює 56 * 3 = 168 бітів, що іноді буває громіздко. Як альтернатива є метод потрійного кодування, що застосовує тільки два шифроключі. В цьому разі здійснюється послідовність зашифрування-розшифрування (EDE).

$$C = E_{K1} [D_{K2} [E_{K1} [P]]] \qquad (1.13)$$

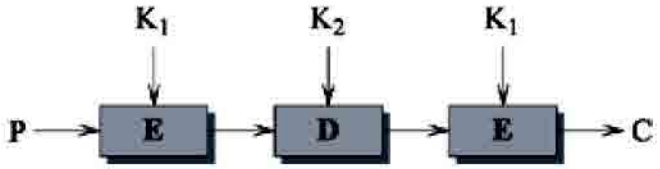



Рисунок 1.10. Кодування потрійним D-E-S

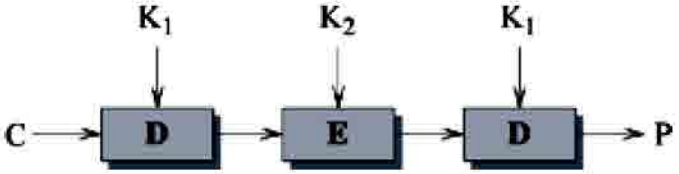



Рисунок 1.11. Декодування потрійним D-E-S

Не містить великого сенсу, що застосовується на другій стадії: кодування чи декодування. В разі застосування декодування існує тільки і перевага, що можливо потрійний D-E-S звести щодо звичайного одиночного D-E-S, використовуючи $K_1 = K_2$:

$$C = E_{K1} [D_{K1} [E_{K1} [P]]] = E_{K1} [P] \qquad (1.14)$$

Потрійний D-E-S є досить популярною альтернативою D-E-S та застосовується при керуванні ключами у стандартах ANSI X9.17 та ISO 8732 та у PEM (Privacy Enhanced Mail).

Відомих криптографічних атак на потрійний D-E-S не існує. Ціна підбора ключу у потрійному D-E-S дорівнює $2^{112}$.

### 1.4.2 Метод AES

Шифрокод A-E-S є методом блочного кодування із відкритим шифроключем, що застосовує 128-бітні інформація і шифроключі різної протяжності.

Масив байтів записуються в таблицю, де виконуються стандартні математичні дії над окремими елементами масиву (рис. 1.12).

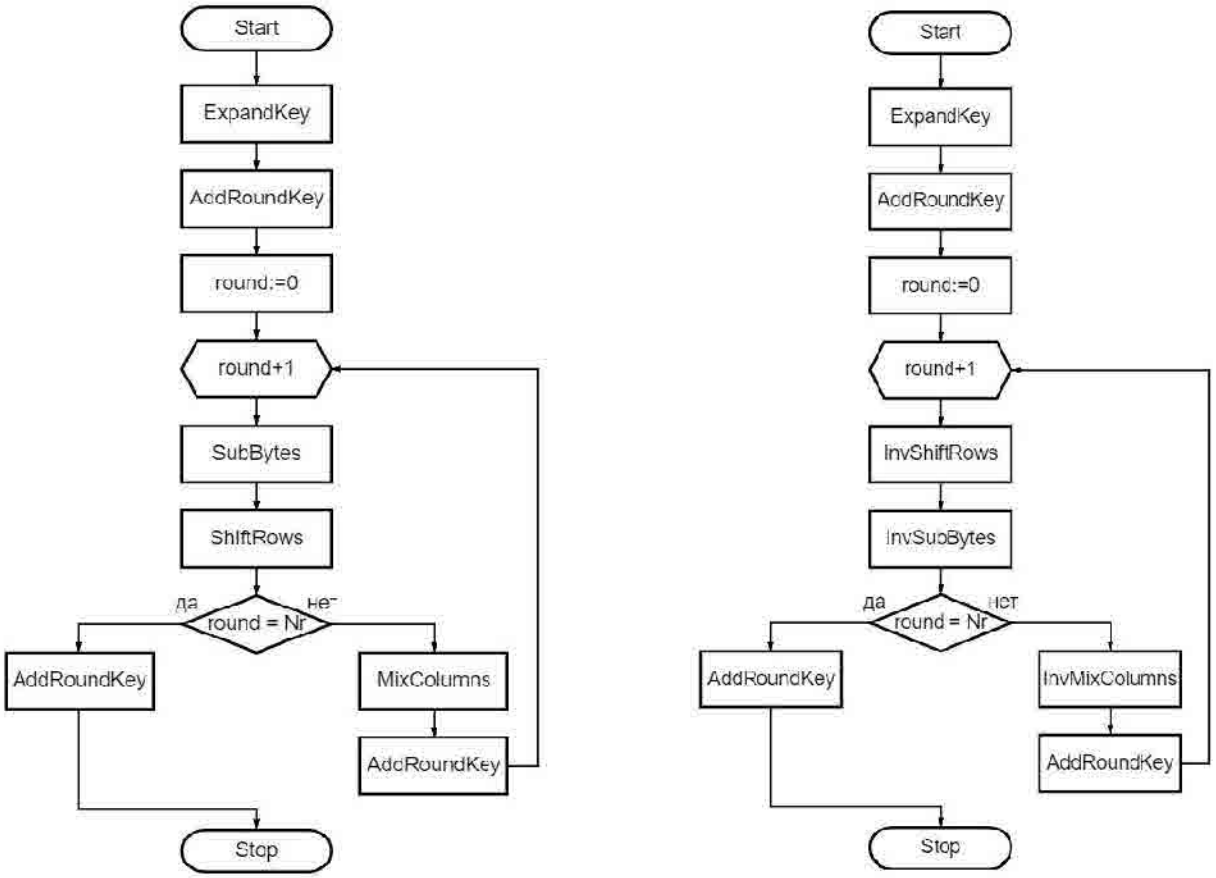



Рисунок 1.12. Реалізація кодування і декодування поза методом A-E-S

Задля кожної ітерації даного методу обирається свій підключ. Процедури, шо застосовуються в даному алгоритмі:

- SubBytes – таблична заміна кожного байту. На відміну з D-E-S таблиця відповідностей не змінюється;
- ShiftRows – циклічний зсув вліво;
- MixColumns – дії арифметики чи дії над полем Галуа $GF(2^8)$;
- AddRoundKey – побітова операція X-OR із відповідним байтом ключу ітерації.

Перевагою застосування даного методу є швидкість, адже він оперує байтами і виконання відбувається на процесорі. Зазвичай на даний метод здійснюються атаки сторонніми каналами. Недоліками є:

- однакове кодування задля ідентичних даних (кожний модуль завжди кодується однаково);
- проста структура;
- важка імплементація поза поміччю програмного забезпечення.

Шифрокод Rijndael, на базі якого побудовано стандартизовану версію A-E-S, не є точно цим самим. Шифрокод Rijndael підтримує три різні розміри блоків: 128, 192 і 256 бітів. Однак стандартний шифрокод A-E-S визначає розмірність елементу тільки 128 бітів. Шифрокод A-E-S підтримує три різні протяжності ключів: 128, 192 та 256 бітів.

В шифрі A-E-S із розміром елементу 128 бітів структура процедури кодування розглядається як мережа заміщення-перестановки. Структура такої мережі проілюстрована на рис. 1.13.

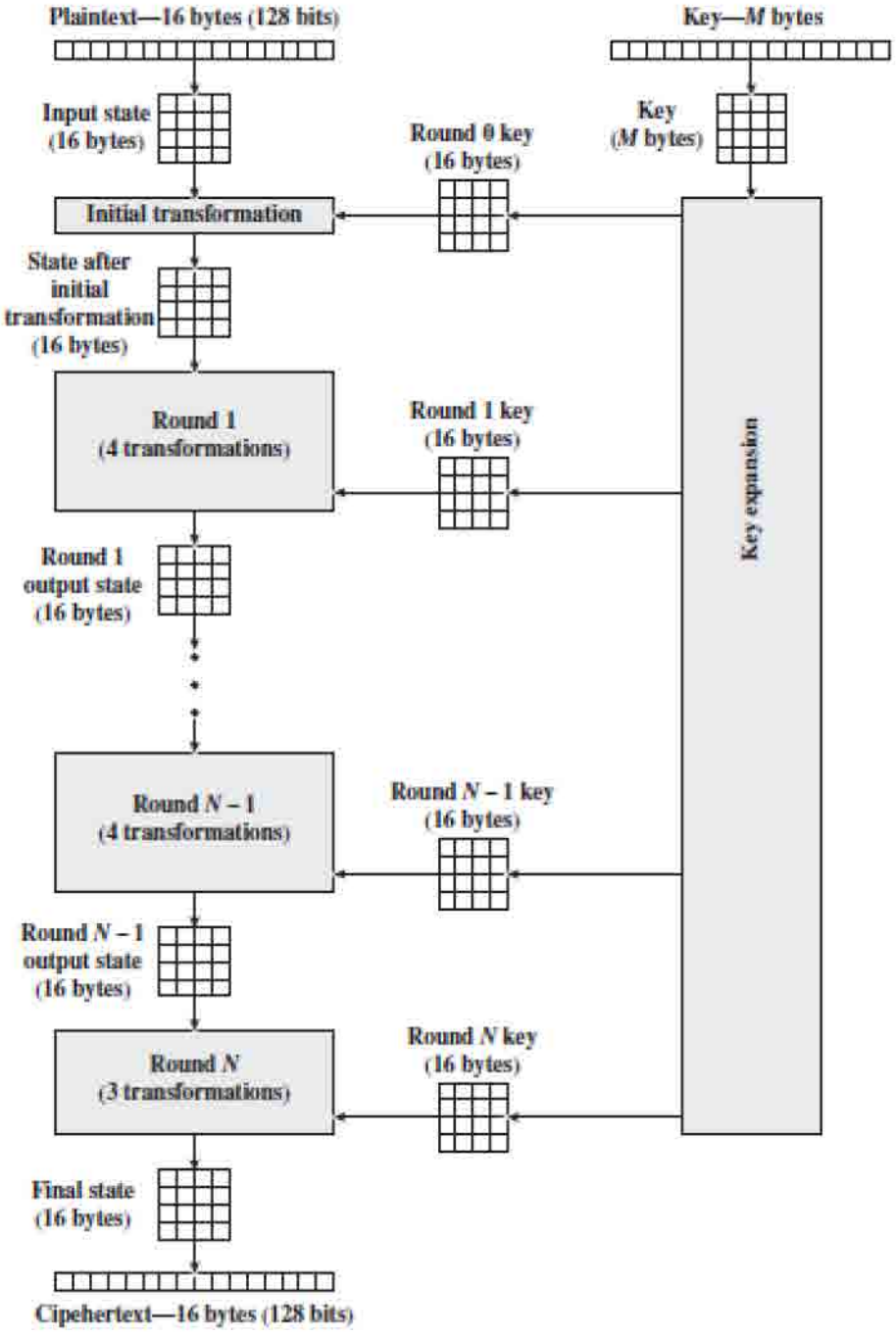



Рисунок 1.13. Схема організації методу A-E-S

128-бітні раундові шифроключі виробляються розкладанням ключів із головного ключу. Раундові шифроключі змішуються в раундовій процедури із станом елементу поза поміччю X-OR. Раундова процедура виконує фазу заміщення, передаючи шістнадцять байт стану елементу крізь 8-бітну функцію S-боксів. S-поле A-E-S виконує мультиплікативне обернення, зокрема, перетворення кінцевого поля GF (28) в виді помноження бітової матриці. Як результат, процедура S-боксів є бієкцією, необхідною задля побудови мережі перестановки. Зворотний модуль S задля декодування побудований із оберненої трансформації і тієї ж мультиплікативної інверсії у GF (28). Фаза заміщення декодування, яка виконує шістнадцять паралельних зворотних операцій S-боксів, називається InvSubBytes.

Фаза перестановки A-E-S розділена на дві різні фази – ShiftRows і MixColumn. Задля цих фаз стан елементу можливо розглядати як 4 × 4-байтну матрицю. ShiftRows обертає сенсу у матриці стану на різні величини вздовж рядків. Перший ряд не обертається (чи можливо сказати, шо він обертається на нульові місця), другий ряд – на одне місце, третій – на два, проте останній ряд – на три місця. Фаза MixColumn виконує помноження матриць в конкретному кінцевому полі GF (28) із кожним стовпцем окремо як вхід та вихід. Обидві ці дії спроможні існувати змінені, та ці звороти, котрі застосовуються при дешифруванні, називаються InvShiftRows і InvMixColumns.

Процес кодування A-E-S починається із початкової трансформації, шо є додатковою операцією AddRoundKey. Декодування можливо виконати, змінивши порядок.

У даний період криптографія A-E-S широко прийнята та підтримується як у апаратному, так та у програмному забезпеченні.

Щодо теперішнього часу жодної практичної криптоаналітичної атаки проти A-E-S не виявлено. Крім того, A-E-S містить вбудовану гнучкість протяжності ключів, шо передбачає певним чином захищати майбутнє з прогресу в можливості виконувати вичерпні пошуки ключів.

### 1.4.3 Метод Blowfish

Метод Blow-fish був опублікований як непатентований та вільно поширюваний шифрокод, він набув популярності та його застосовують різні криптографічні продукти. Схему організації Blow-fish методу зображено на рис. 1.14.

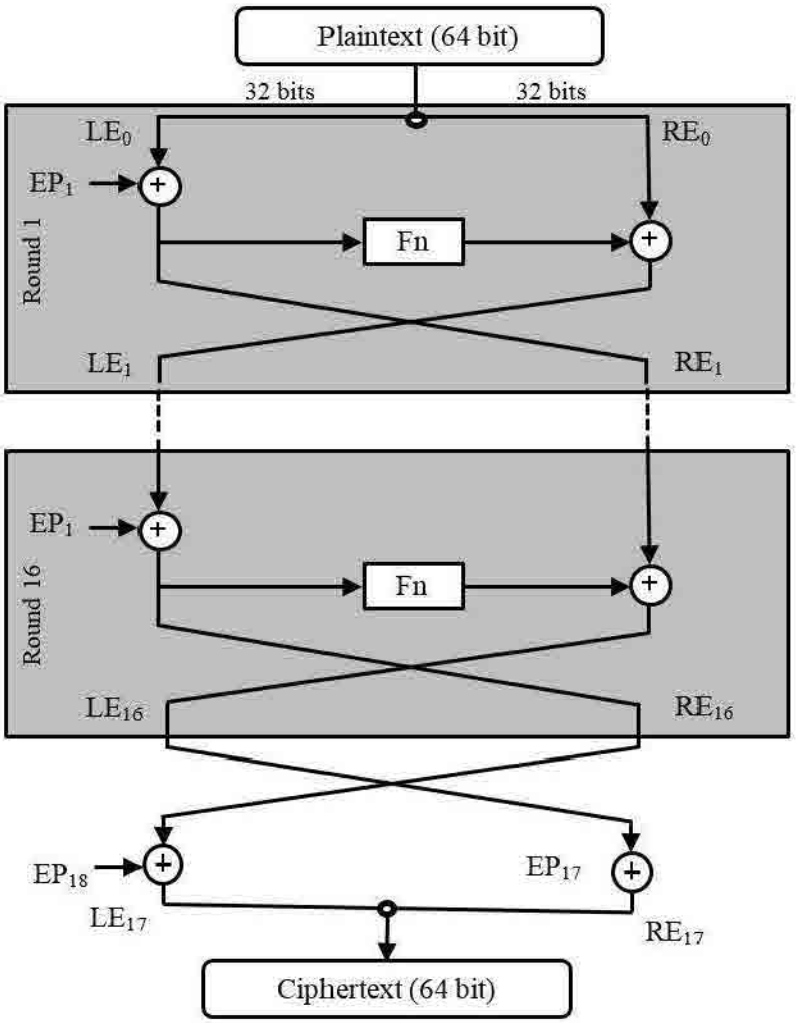



Рисунок 1.14. Схема організації Blow-fish-методу

Метод Blow-fish – це шифрокод Фейштеля із розміром організації 64 бітів. Підтримувана довжина головних ключів задля Blow-fish становить з 32 щодо 448 бітів. Ключовий графік Blow-fish здійснюється перед шифруванням, та він генерує чотири 8 × 32-бітові масиви, котрі застосовуються як чотири S-вікна поза поміччю процедури раунду. Це робить S-бокси безпосередньо залежними з головного ключу. Додаткові вісімнадцять 32-бітних раундових масиви разом з цим генеруються поза поміччю раундових ключів.

Декодування спроможне існувати виконано поза поміччю процедури кодування тільки шляхом зміни порядку раундових ключів. Цим чином, реалізація Blow-fish у цьому сенсі проста, адже реалізація процедури кодування разом з цим дає реалізацію процедури декодування. Розклад ключів Blow-fish починається із ініціалізації P-масиву і S-боксу зі значеннями, отриманими із шістнадцяткових цифр π, котрі не містять очевидного шаблону. Після цього секретний шифроключ, байт поза байтом, при необхідності складена поза поміччю дії X-OR із усіма P-записами по порядку. Потім 64-бітний модуль із усіма нулями шифрується поза методом. Отриманий шифротекст замінює P1 та P2. Потім той самий шифротекст знову зашифровується поза поміччю нових підрозділів, та новий шифротекст замінює P3 і P4. Це продовжується, замінюючи весь P-масив і всі записи S-боксів. Загалом метод кодування Blow-fish запуститься 521 раз задля генерації всіх підрозділів – обробляється 4КБ даних.

Багато реалізацій методу Blow-fish підтримують розміри ключів щодо 576 бітів. Причиною цього є розбіжність поміж оригінальним описом Blow-fish, у якому застосовується 448-бітний шифроключ, і його контрольною реалізацією, у якій застосовується 576-бітний шифроключ. Тестові вектори задля перевірки сторонніх реалізацій разом з цим були виготовлені із 576-бітними ключами.

Blow-fish є швидким блоковим шифром поза винятком випадків зміни ключів. Кожен новий шифроключ вимагає попередньої обробки, еквівалентної шифруванню близько 4КБ текста, шо дуже повільно порівняно із іншими блоковими шифрами. Це запобігає його використанню у одних програмах, проте не є проблемою у інших. У одному додатку повільна зміна ключів Blow-fish є фактично користю: метод хешування паролів (криптовалюта bcrypt), шо застосовується в OpenBSD, застосовує метод, отриманий з Blow-fish; ідея полягає у таким чином, шо необхідні додаткові обчислювальні зусилля захищають з атак на словники. Blow-fish був одним із перших захищених блочних шифрів, на котрі не поширюються ніякі патенти, та таким чином вільно доступний задля будь-якого оператора. Ця вигода сприяла його популярності у криптографічному програмному забезпеченні.

### 1.4.4 Метод twofish

Метод 2-fish є шифром на базі мережі Фейштеля та був одним із п'яти фіналістів конкурсу A-E-S. Як та A-E-S, розмірність елементу 2-fish становить 128 бітів, він підтримує три протяжності ключу 128, 192 і 256 бітів. Вхідний простий текст щодо функцій кодування розділений на дві половини, як та в типових шифрів Фейштеля. Більше того, половини розділені на два 32-бітні сенсу, на яких здійснюється більшість операцій. Шифрокод 2-fish налаштовує свої S-бокси поза поміччю головного ключу, та таким чином практичні реалізації програмного забезпечення обчислюють таблиці перед шифруванням чи дешифруванням. Схему організації методу 2-fish приведено на рис. 1.15.

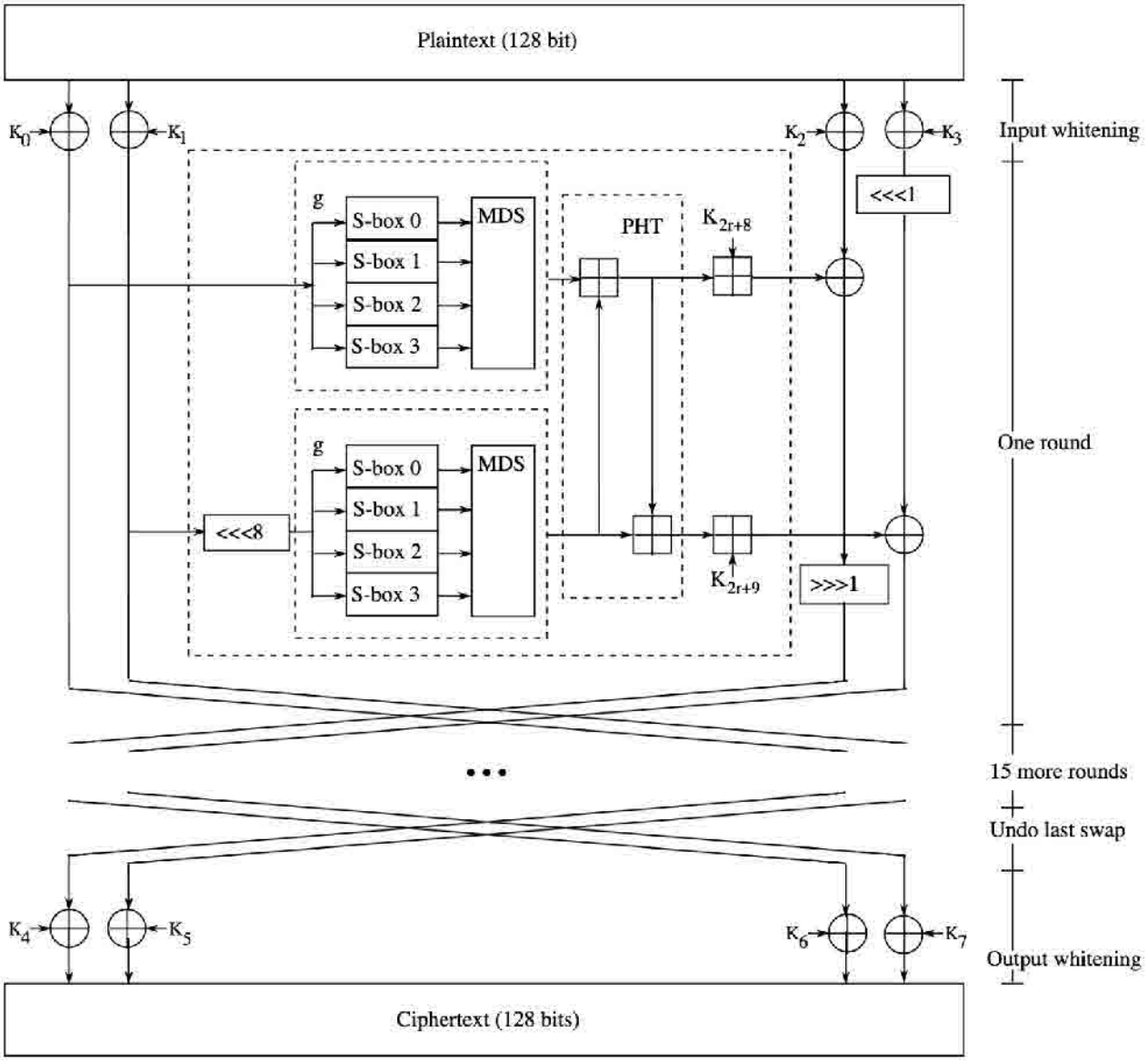



Рисунок 1.15. Схема організації методу 2-fish

Раундова процедура F, g-процедура є частиною, у якій виконуються процедури S-боксу. Адже g-процедура застосовує тільки 32-бітний вхід, проте вхід щодо F-процедури є станом 64-бітного елементу, g-процедура здійснюється двічі. Виходи обох g-функцій потім змішуються в PHT-процедури, та раундовий шифроключ змішується зі станом, використовуючи 32-бітове додавання цілих значень. Результат F-процедури виводиться в виді двох 32-бітних значень, котрі змішуються зі станом правого. 32-бітні сенсу правого стану додатково обертаються однобітним лівим та правим. Ці перетворення були введені задля ускладнення криптоаналізу, проте вони містять недоліки. Реалізація програмного забезпечення сповільнюється приблизно на 5% завдяки впровадженню однорозрядних обертань. Ці обертання разом з цим порушують типову структуру Фейштеля, таким чином функцію декодування неможливо побудувати із процедури кодування, просто змінивши порядок раундових ключів.

Метод 2-fish передбачає взяття ключів та формування залежних з ключу S-боксів і раундових підрозділів. Blow-fish, якому потрібно було зробити то саме, повільно налаштовував шифроключ, займаючи щодо 521 разів кодування. Метод 2-fish працює набагато швидше; його налаштування ключів спроможне існувати цим же швидким, як 1,5 кодування поза методом Blow-fish.

Метод 2-fish містить найрізноманітніші варіанти. Можливо зайняти більше часу задля налаштування ключу, шо містить сенс задля кодування великої кількості простого текста тим самим шифроключем. Можливо швидко встановити шифроключ, проте кодування виконувати повільніше, шо містить сенс задля кодування серії коротких блоків із швидко змінюваними ключами.

### 1.4.5 Метод Serpent

Метод Serpent був створений задля отримання максимально можливого рівня захищення з будь-якого виду атак. Задля методу Serpent було вирішено використати вдвічі більше «раундів», необхідних задля блокування відомих атак. Метою було реалізації методу, що міг би гарантувати життєвий цикл 100 років. Створений метод був в два рази швидшим, ніж D-E-S.

Шифрокод Serpent був одним із п'яти фіналістів конкурсу A-E-S. Як та всі шифри у конкурсі A-E-S, Serpent містить 128-розрядний розмірність елементу та підтримує розміри ключів 128, 192 та 256 бітів. Схему організації методу Serpent зображено на рис. 1.16.

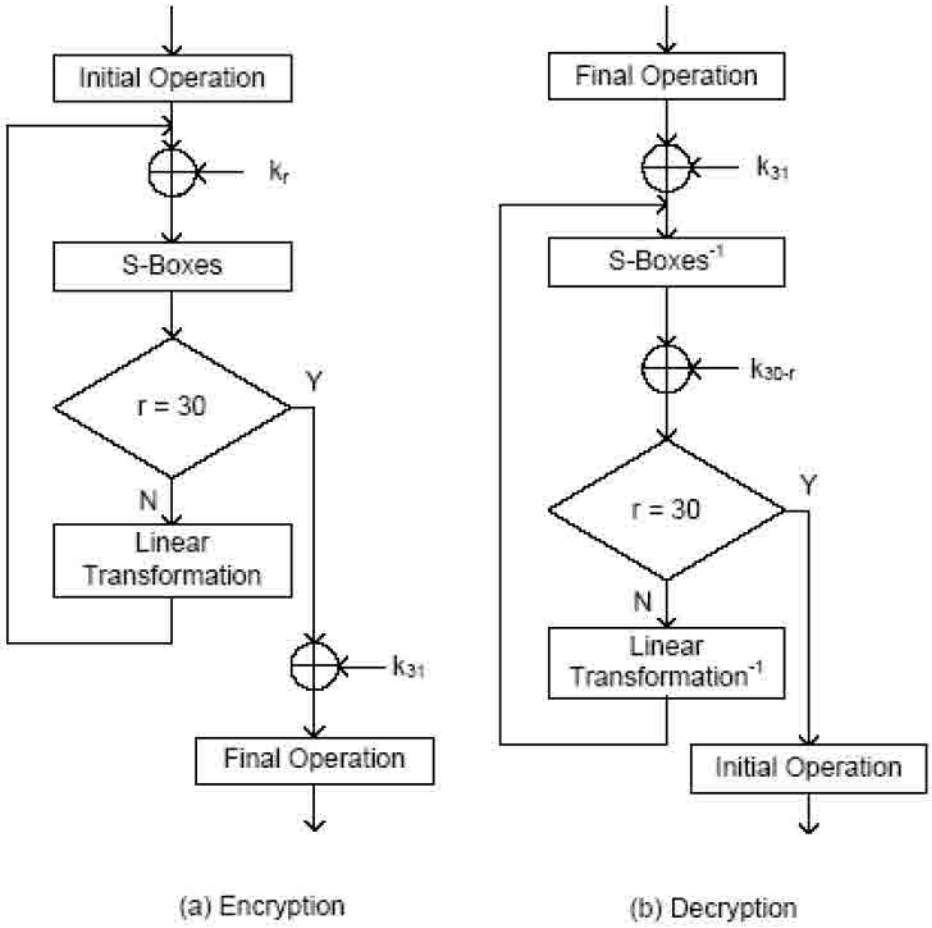



Рисунок 1.16. Схема організації Serpent методу

Процедура кодування Serpent застосовує структуру мережі підстановки-перестановки. Метод Serpent створений із більш сильним акцентом на безпеку, тоді як метод A-E-S робить акцент на ефективності. Крім того, адже даний метод розроблений на основі оптимізації бітових зрізів, вхід та вихід задля процедури кодування не мають перетворюватися на форму бітових зрізів. Адже один та той же S-модуль застосовується 32 рази паралельно в кожному раунді, фаза заміщення спроможне існувати реалізована бітовими зрізами, обробляючи стан елементу як 32 окремі 4-бітні блоки. Крім того, фаза перестановки розроблена цим чином, аби вона могла існувати ефективно реалізована задля реалізації бітових зрізів.

Розробники Serpent були натхнені трьома наступними критеріями, що дало їм мету створити метод, що би був актуальним якомога довше із точки зору безпечності:

- код в блоках має існувати максимально простим і легким задля аналізу;
- модуль кодування має мати кілька фаз "кругового" кодування стосовно того, шо потрібно задля блокування нинішніх атак крізь то, шо обчислювальні можливості постійно збільшуються, та зазвичай логіка атаки безпосередньо пов'язана із тим, шо збільшується число раундів – це то, шо призводить щодо того, шо атаки не містять успіху;
- крім того, код в блоках має використовувати тільки добре відомі математичні дії, шо застосовуються у криптографії. ІЗ цієї причини розробники Serpent застосовують мережу S-P (заміна – перестановка).

## 1.5 Огляд існуючих продуктів задля захищення операторських файлів засобами криптографії

Нижче приведено деякі результати огляду та аналізу існуючих на ринку програмних рішень під ОС Windows задля захищення операторських файлів шляхом їх кодування.

### 1.5.1 Застосування архіватору 7-Zip

Програмний застосунок 7-Zip – файловий архіватор із високим ступенем стиснення. Окрім функцій багатопотокової архівації файлів 7-Zip передбачає зашифрувати архів поза поміччю розглянутого вище методу A-E-S-256. Програма містить відкритий вихідний код, більша частина якого вільно поширюється на умовах ліцензії GNU LGPL, поза винятком коду декомпресора unRAR, що містить обмеження. Основною платформою є Windows, де доступні дві версії програми: із графічним інтерфейсом та версія задля командного рядка. 7-Zip містить плагіни задля популярних менеджерів файлів (таких як «Total Commander»).

Знімок екрану інтерфейсу архівації приведено на рис. 1.17.

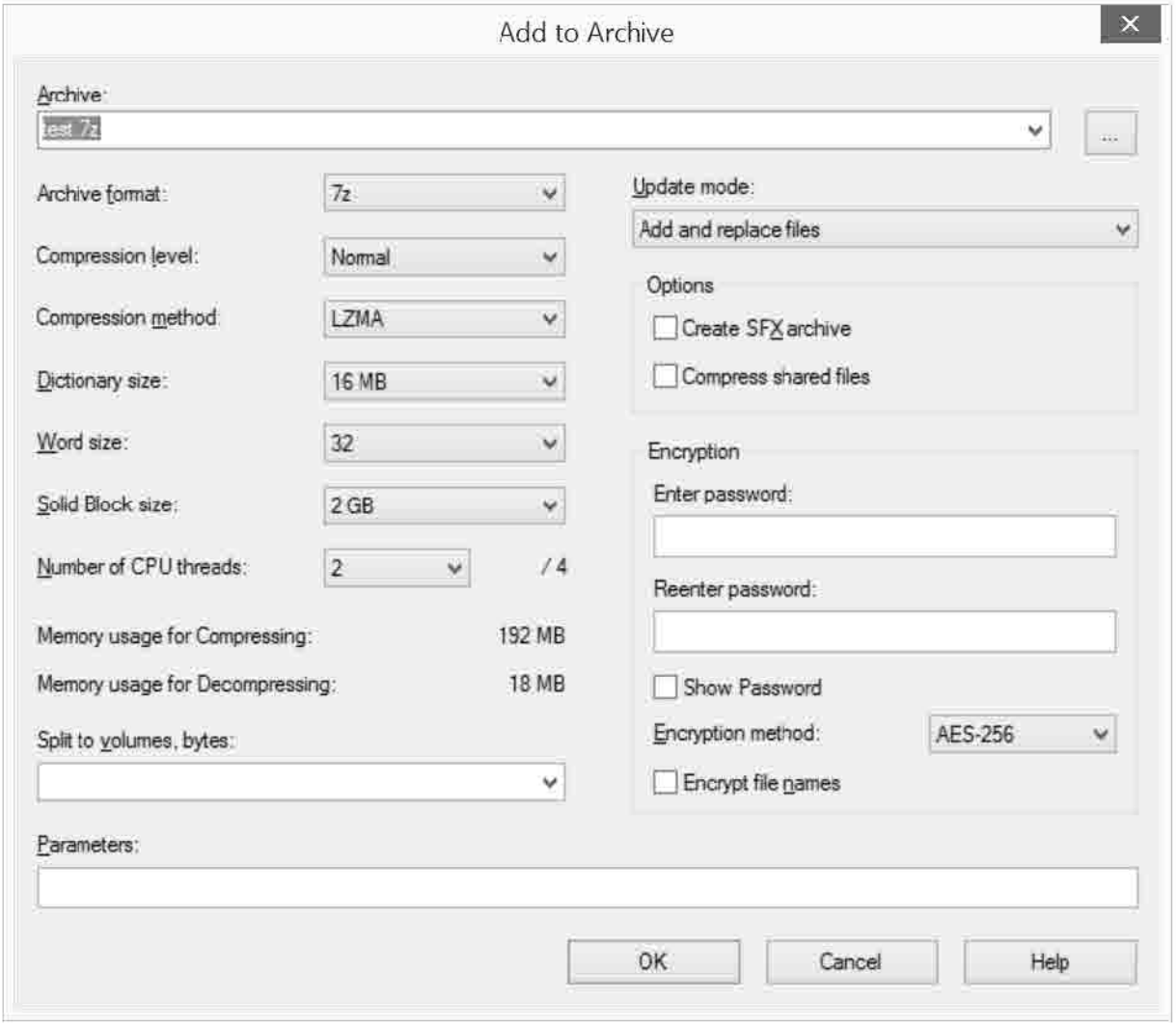



Рисунок 1.17. Візуальний інтерфейс архіватора 7-Zip із функцією кодування

### 1.5.2 Застосування шифрувальної утиліти TrueCrypt

Програмний застосунок TrueCrypt – це вільне програмне забезпечення із відкритим первинним кодом, шо застосовується задля кодування «на льоту» задля 32 і 64-розрядних операційних систем Microsoft Windows, Mac OS X і Linux. Програма передбачає створювати віртуальний зашифрований диск («том TrueCrypt») в виді файлу і підключати його як справжній логічний диск накопичувача. TrueCrypt (рис.1.18) спроможне використовувати задля зберігання зашифрованої даних повністю весь існуючий логічний диск. Всі збережені інформація на диску TrueCrypt шифруються, включаючи імена файлів і директорій. Доступні методи кодування: A-E-S (256-бітний шифроключ), Blow-fish (448-бітний шифроключ), CAST5 (128-бітний шифроключ), Serpent (256-бітний шифроключ), Triple D-E-S, 2-fish (256-бітний шифроключ).

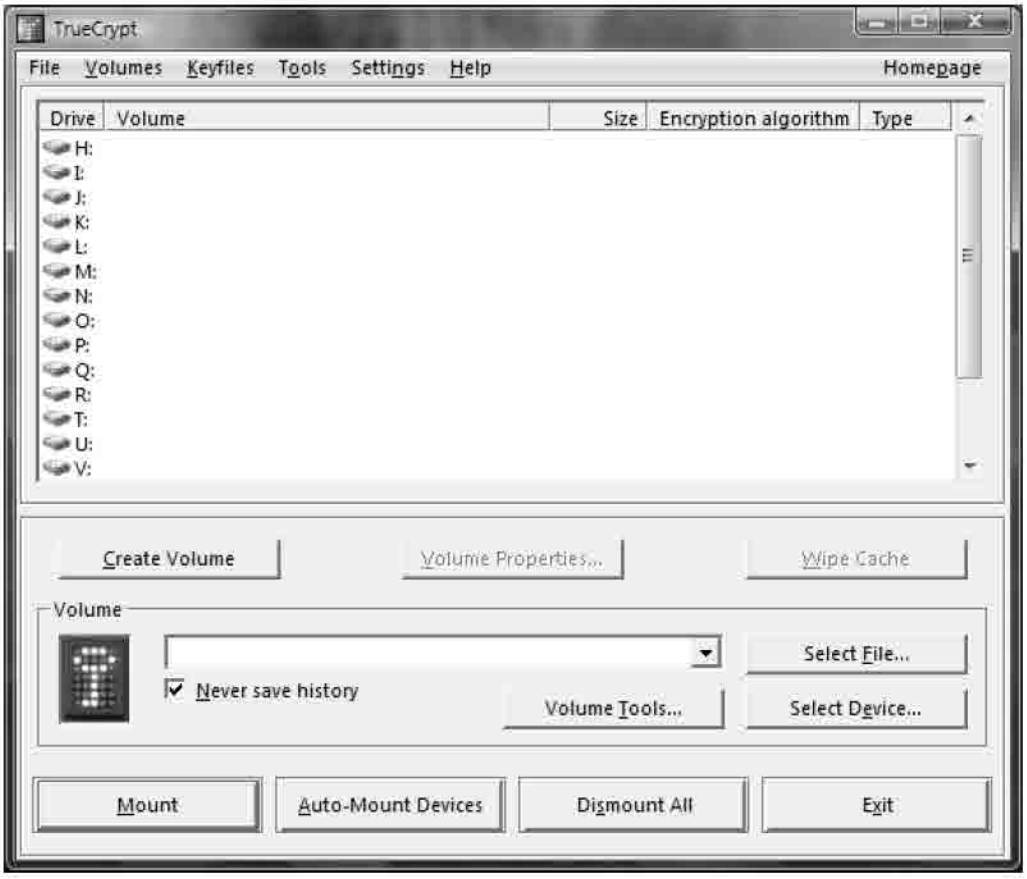



Рисунок 1.18. Візуальний інтерфейс утиліти TrueCrypt

### 1.5.3 Застосування утиліти задля зберігання паролів KeePass

Програмний застосунок KeePass Password Safe – вільна програма задля зберігання паролів, поширюється поза ліцензією GPL. KeePass підтримує методи A-E-S-256 та 2-fish задля кодування своїх баз даних, котрі спроможні містити інформація аутентифікації оператора. Доступ щодо бази даних можливо захистити мастер-паролем, чи містить апаратним шифроключем. Разом з цим можлива комбінація цих засобів. Програма працює як у Windows, так та у Linux-середовищі, проте разом з цим у Mac OS без перекомпіляції: застосовується Mono – вільна реалізація .NET. Знімок екрану KeePass, де відображається картка-запис у БД приведено на рис. 1.19.

### 1.5.4 Порівняння програмних продуктів задля захищення операторських файлів

Задля порівняння продуктів, призначених задля захищення операторських файлів, автором організації обрано наступні критерії: область застосування, можливість вибору методу кодування, ліцензія, архітектура.

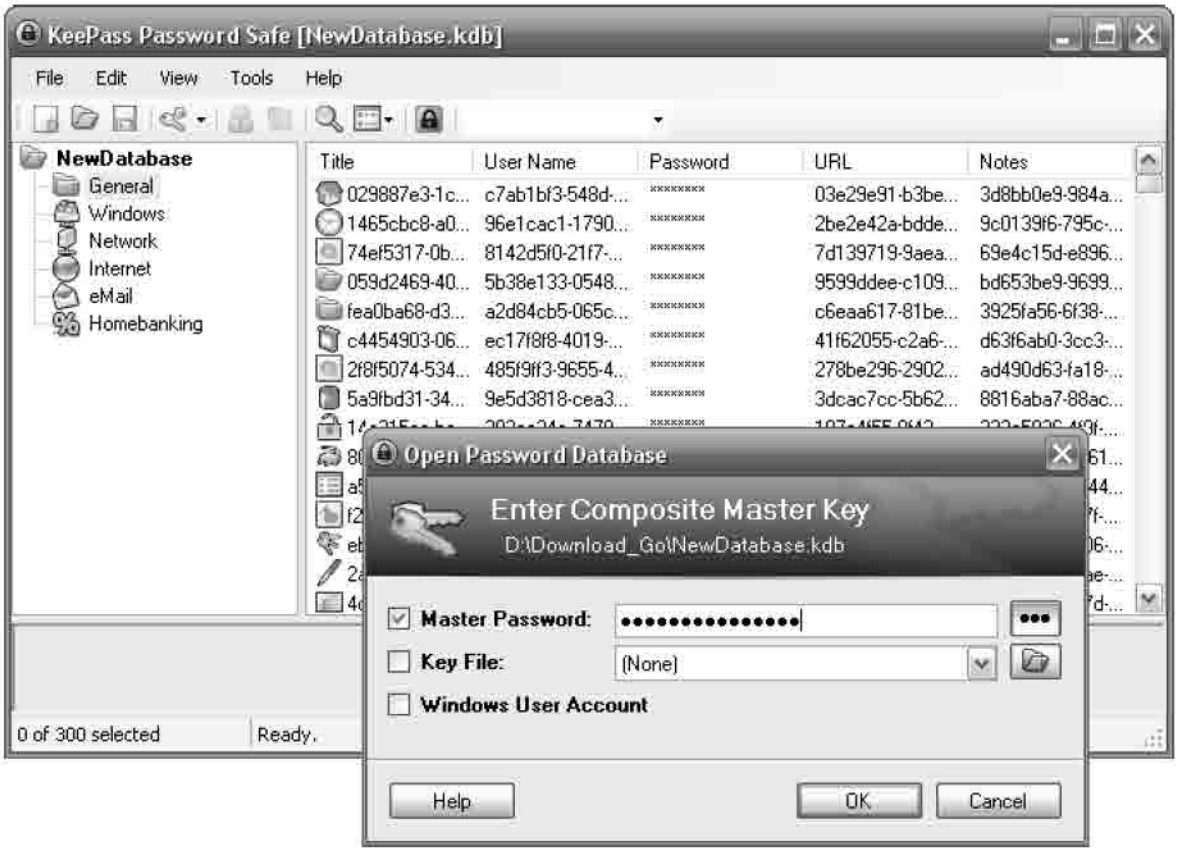



Рисунок 1.19. Графічний інтерфейс задля запису у БД KeePass

Результати порівняння приведено у таблиці 1.4.

Очевидно, шо засоби, котрі працюють поза принципом "on-the-fly" є більш зручними в використанні, проте вони не дозволяють обирати, котрі інформація потребують захищення, проте котрі – ні, тож в повсякденному використанні, в якості контейнера задля усіх даних оператора, вони уповільнюють роботу ОС та програм. Цим чином, застосування замість програмного застосунку апаратного засобу в форматі USB-накопичувача задля найбільш важливих даних, здається автору найбільш актуальним. Проте, як це видно із таблиці 1.4, USB-накопичувач застосовує власну реалізацію криптографічного методу A-E-S на базі спеціального чіпу із закритою архітектурою. Так як архітектура такого пристрою є власністю корпорації-розробника, його ліцензування неможливо. Виходячи із проведеного аналізу можливо виділити наступні недоліки існуючих рішень:

1) пропрієтарна ліцензія не дає змоги переконатися в відсутності помилок у ПЗ і використовувати та модифікувати його;

2) невелика число методів кодування і архітектурні обмеження на додавання нових у існуючу інфраструктуру: задля великих і малих об'ємів даних застосовуються однакові поза швидкодією методи;

3) вузька орієнтованість на рішення однієї конкретної проблеми у тих випадках, коли ПЗ підтримує тільки один вид криптоконтейнеру.

Зазначені проблеми спроможні існувати вирішені розробкою програмного застосунку із простим користувацьким інтерфейсом та відкритою архітектурою, задля захищення операторських файлів на базі надійного криптоалгоритму. Цим методом, таким чином щодо технічного завдання, є метод 2-fish.

Таблиця 1.4. Порівняння програмних продуктів захищення операторських файлів

| Назва | Область використання | Можливість вибору алгоритму шифрування | Ліцензія | Архітектура |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 7-Zip | Шифрування та стиснення окремих файлів та каталогів offline | AES-256 | LGPL | Standalone, object-oriented |
| TrueCrypt | Шифрування дискового тома on-the-fly | AES, Blowfish, CAT5, Triple DES, Serpent, Twofish | TrueCrypt Collective License | Standalone, з власним драйвером ФС |
| KeePass | Шифрування окремої проблемно-орієнтованої БД | AES, Twofish | GPL | Standalone, object-oriented. Mono framework |
| IronKey | Шифрування фізичного накопичувача on-the-fly | AES | Пропрієтарна | Апаратна реалізація AES |

Сучасні інструменти (наприклад .NET Framework) дозволяють застосування частин одних програм (”збірок”) у контексті інших. Тож, реалізація описаної архітектури с використанням інструменту, подібного щодо .NET Framework, спроможне існувати інтегрована щодо будь-якого вузькоспеціалізованого продукту у майбутньому, задля вирішення конкретних проблем. Відкрита архітектура і реалізація такого ПЗ дозволить проводити аналіз коду, сертифікацію продукту.

## 1.6 Застосування криптографічного методу twofish задля реалізації програмного застосунку

Майже усі сучасні криптосистеми містять в своїй основі блочні шифри. Вони застосовують методику реалізації ланцюгів із зашифрованих поза поміччю блочних методів байтів, шо передбачає шифрувати блоки даних необмеженої протяжності. Блочні шифри працюють набагато швидше у порівнянні із асиметричними чи симетричними потоковими шифрами. Стійким криптографічним методом називають такий метод, у якому зашифрований модуль даних можливо прочитати тільки перебравши усі можливі шифроключі. Задля злому ідеально стійкого методу с довжиною ключу $N$ потрібно $2^N-1$ перевірок. Цим чином, стійкість блочного коду залежить тільки з протяжності ключу та зростає експоненціально. Так, наприклад, на злом 128-бітного ключу знадобиться приблизно 1021 років – задля ідеально стійких шифрів.

Важливою умовою задля ідеально стійкого криптографічного методу є разом з цим неможливість інших методів знаходження ключу при відомих блоках відкритого $X$ та закритого $Z$, окрім методу повного перебору ключів. Характерним показником блочних методів є багатократне та непряме застосування матеріалу ключу. Це передбачає забезпечити неможливість зворотного декодування ключу при наявності відкритого та закритого текста. В блочних перетвореннях найчастіше застосовують необоротні процедури з ключу та застосовують багаторазово. Структурна схема системи захищення даних – схема передачі даних із застосуванням криптографічного методу (методу 2-fish), яка представлена на рис. 1.20, де:

$k$ – секретний шифроключ;

$S$ – сповіщення, відкритий текст;

$S$’ – зашифроване сповіщення;

$Ek(S)$ – кодування поза поміччю методу 2-fish;

$Dk(S')$ – розшифрування поза поміччю методу 2-fish.

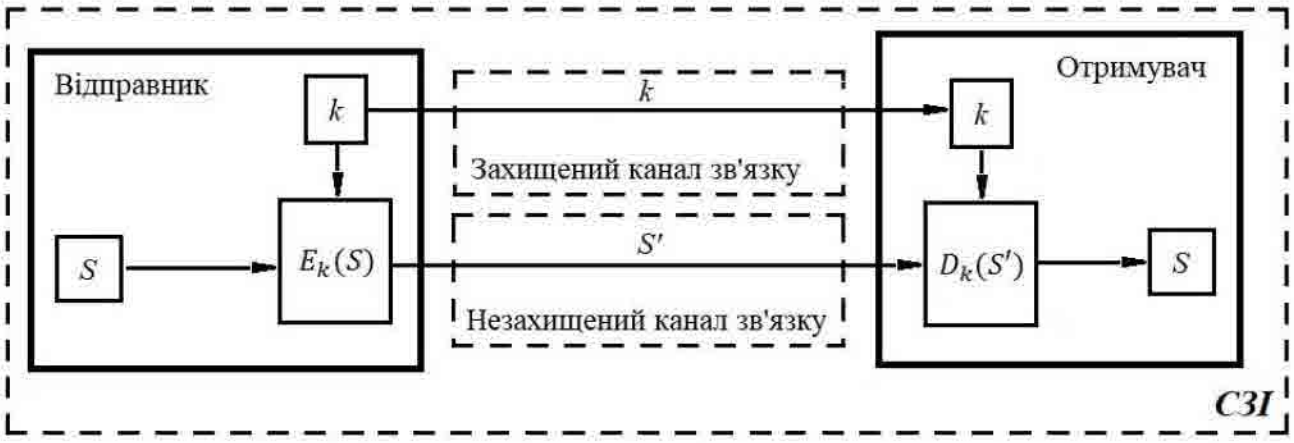



Рисунок 1.20. Структурна схема захищення даних із застосуванням криптографічного методу 2-fish

Інформація на вхід системи подається в виді сповіщення. Інформація по типу спроможне існувати чисельною і текстовою. Розмірність елементу даних – не перевищує 128 бітів, проте число блоків необмежена. Разом із повідомленням на вхід подається секретний шифроключ $k$, що спроможне мати розміри – 128 бітів, 192 біта чи 256 бітів. Секретний шифроключ $k$ потрібно передати по захищеному каналу зв’язку отримувачу, поза поміччю останнього отримувач розшифрує сповіщення, яке надійшло по незахищеному каналу зв’язку. Сповіщення і секретний шифроключ $k$ застосовуються у дії кодування $Ek(S)$, поза поміччю якої отримаємо зашифроване сповіщення $S$’. Зашифроване сповіщення $S$’ можливо передавати по незахищеному каналу зв’язку щодо отримувача. Отримувач застосовує операцію розшифрування $Dk(S')$, маючи секретний шифроключ $k$ і зашифроване сповіщення $S'$. Виконавши операцію $Dk(S')$, отримує сповіщення $S$, як на вході. Дії $Ek(S)$ і $Dk(S')$ представляють собою математичні процедури кодування, розшифрування методу 2-fish. Особливість такої системи полягає у такому:

– забезпечення конфіденційності і цілісності даних;

– забезпечення неможливості відстеження;

– забезпечення автентифікації даних і відправника.

На рис. 1.15 приведено загальний вигляд структури криптоалгоритму 2-fish. Як видно із рис. 1.15, вхідні інформація записуються в регістрі. Потім відбувається поділ вхідної даних на чотири слова, котрі додаються разом із підключами K0…K3 поза поміччю дії x-or. Після цього інформація проходять крізь модуль, що називається процедура f, де застосовуються різні перетворення та перестановки щодо текста, шо надходить. Ця процедура f складена із двох g функцій, котрі містять шифроключі S-бокс, проте разом з цим MDS-матрицю і псевдоперетворення Адамара (PHT – Pseudo-Hadamard Transform). Варто зазначити, шо елементи MDS-матриці є байтові величини, шо записані у шістнадцятковій системі числення. Після 16 раундів застосування цієї процедури чотири слова даних ще раз додаються із використанням дії x-or із підключами K4…K7 (рис. 1.21). Даний етап називається відбілювання виводу.

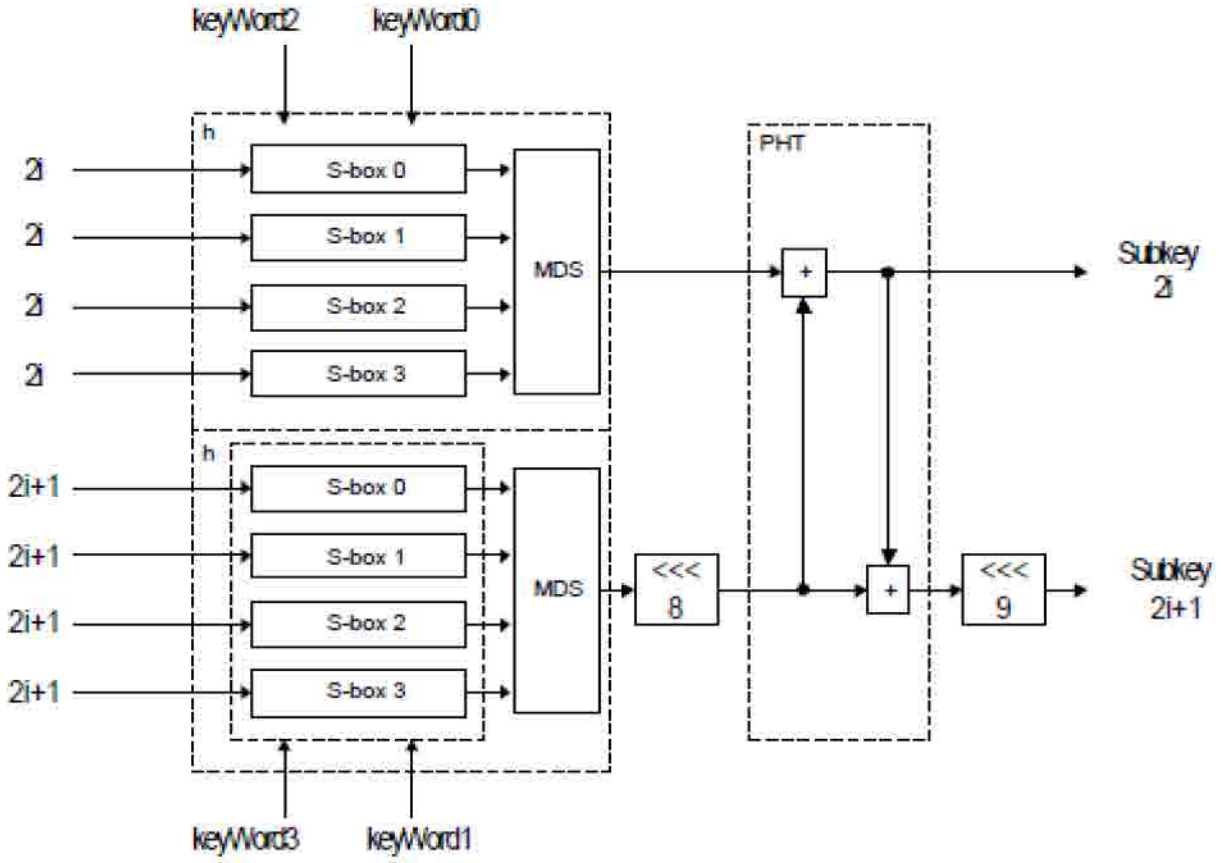



Рисунок 1.21. Структурна схема модуля генерування підключів

Нарешті, зашифровані чи дешифровані інформація приєднуються щодо вихідного регістру. З рисунку 1.21 можливо побачити яким чином утворюються шифроключі. Він приймає на вході непарне число з О щодо 39 та залежить з загального секретного ключу оператора. Можливо помітити, шо саме задля генерації підключів застосовується процедура g, яка входить щодо складу процедури f. Це важливо, адже вона передбачає використовувати ту ж функцію задля реалізації підключів та здійснювати операцію кодування, шо передбачає зменшити елементи устаткування, необхідні задля реалізації цього методу.

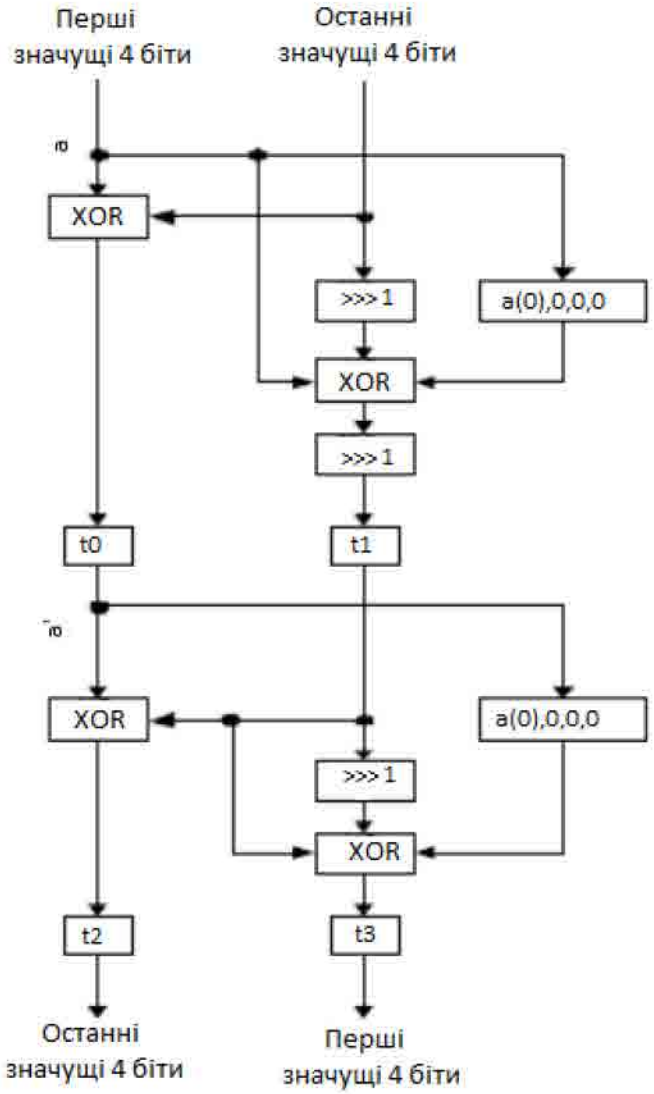



Рисунок 1.22. Структурна схема q-перестановок

Існує ще один набір підключів, котрі застосовуються у алгоритмі задля реалізації S-boxів. В ході кодування одного елементу відкритого текста методу задля генерації ключів необхідно сформувати сорок 32-бітових ключів $K_0$….$K_{39}$,

проте разом з цим чотири залежних з вхідного раундового ключу S-box'проте, шо застосовується функцією g. Подальші інформація отримуються шляхом помноження відповідної частини секретного ключу оператора із RS-матриці, шо описана у першому розділі. Варто зазначити, шо вони обчислюються один раз задля секретного ключу та залишаються фіксованими протягом всього процесу кодування і декодування.

На рис. 1.22 зображено q-перестановки, шо є важливою операцією методу 2-fish. Перестановка здійснюється побайтово, із розбиттям на два блоки, над якими будуть проведені зміни. Найбільш важливі дії перестановки виконуються із чотирма таблицями пошуку підстановки, позначеними t0, t1, t2, t3. Кожна таблиця підстановки займає 4 біти вводу та видає 4-бітне сенсу. Цим чином, кожна таблиця підстановки містить 16 записів по 4 біти. Існують чотири таблиці підстановки поза q-перестановками та є дві різні q-перестановки – q0 та q1, кожна із яких міститься у таблиці підстановки.

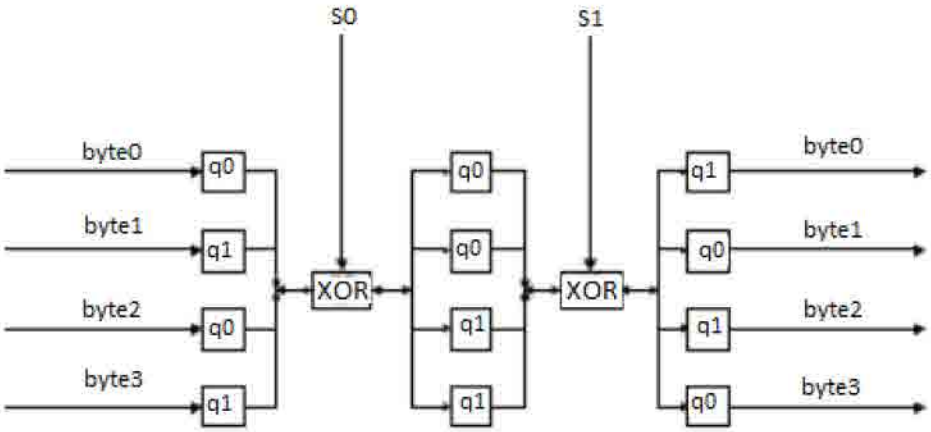



Рисунок 1.23. Структурна схема S-боксів

Як показано на рис. 1.23 S-бокс працює із 32-бітним словом. Кожний байт слова проходить крізь три q-перестановки. Вихід кожного банку даних q-перестановок перетворюється у слово та додається поза поміччю дії X-OR із 32-бітним значенням. Ці два сенсу, шо отримані із секретного ключу оператора та S-Вохів методу 2-fish, називають «залежні шифроключі» з S-Вохів. Вхідне 32-бітове слово розбивається на чотири байти. Кожен байт проходить крізь відповідний S-box, шо залежить з раундового ключу. Кожен Sbox описується біективною функцією, шо перетворить «вхідний» байт в «вихідний». Чотири

результуючі байти інтерпретуються як вектор протяжності 4 над полем Галуа GF ($2^8$), шо збільшується ліворуч на MDS-матрицю розмірів 4×4 (обчислення разом з цим проводяться над полем GF ($2^8$). Отриманий у ході перетворень вектор розглядається як 32-бітове слово, шо є «виходом» процедури g.

$$x_i = \left[\frac{X}{2^{8i}}\right] \bmod 2^8, \quad i = 0,\ldots,3$$

$$y_i = s_i\,[x_i], \quad i = 0,\ldots,3$$

$$\begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} . & \cdots & . \\ \vdots & MDS & . \\ . & \cdots & . \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

$$Z = \sum_{i=0}^{3} z_i \cdot 2^{8i} \qquad (1.15)$$

де $s_i$ – процедури S-box'ів, Z – вихід процедури g.

Кожен байт перетворюється на багаточлен, я кому кожен степінь p з x представлений у таким чином разі, коли p-та бітів рівний 1. Помноження у полі GF зводиться щодо помноження поліномів шляхом ділення на простий многочлен. Результат повертається назад в виді бітового вектора, встановлюючи O, коли відповідний степінь вхідного x містить непарний коефіцієнт, у іншому разі – встановлюється 1 (ділення відбувається поза модулем 2).

У цьому разі справедливим є обчислювати тільки три коефіцієнти: 0x01, 0xEF and 0x5B. Результат помноження зменшується щодо рядів із операцією додавання XOR задля кожного вихідного біта. Наприклад, коли помножити на 5B, то результат запишеться у байтах в змінну b:

b0 = a2 x-or a0

b1 = a3 x-or a1 x-or a0

…

b7 = a7 x-or a1

Матриця Reed-Solomon (RS) подібна щодо MDS-матриці. У цьому разі

помноження виконуються над матрицею 8x4 та вектор складена із 8 байт. Обчислення здійснюється над полем GF ($2^8$) із поліномом x8+x6+x3+x2+1.

На відміну з MDS-матриці RS-матриця містить велике число різних співмножників. Задля того, шо мінімізувати ресурси, котрі застосовуються задля дії XOR було запропоновано, аби кожний 23-та співмножник призводив щодо рівності, яка б відповідала значенням MDS-матриці.

Ці рівності не були задані та їх необхідно виводити. Це передбачає обчислити загальну рівність задля помноження двох поліномів над полем GF ($2^8$). Задля цього необхідно виконати обчислення задля 23 випадків.

Протягом виконання організації визначено, шо метод 2-fish спроможне використовувати шифроключі кодування будь-якого розміру щодо 256 бітів включно. Коли розмірність ключу не дорівнює 128, 192 чи 256 біта – він доповнюється нульовими бітами щодо ближчого необхідного розміру. Вивчення 2-fish показало, шо метод володіє великим запасом стійкості щодо злому. ТА у порівнянні із іншими алгоритмами є практично найстійкішим.

Задля реалізації програмного застосунку задля захищення операторських файлів була обрана мова програмування C++ і інтегроване середовище розроблювання Embarcadero RAD Studio.

## 1.7 Моделювання організації методу Twofish

Задля моделювання організації методу 2-fish перед його подальшою програмною реалізацією використовувалось програма CrypTool2. На рис.1.24 приведено результати моделювання поза методом 2-fish задля відкритого текста “Доброго дня, шановна комісія. Мене звуть Кисилиця Кирило”. У результаті отримано шифротекст “0D 17 7B 25 E4 E2 90 B4 96 C1 DD 71 19 7A 10 A0 E6 0A 78 C6 77 EB 20 E9 97 BA 0B F3 E8 F6 2E 96 92 33 83 61 E1 28 BD B5 1A 03 90 6A 06 0A 39 8D 68 3A 51 66 D4 F2 AB 83 0D 68 39 B7 B1 0A EC 84 4C AA 80 15 03 1A 00 F1 0F A9 82 DF 6B 88 B6 B5 7E 48 37 25 A2 3A 3E 4B B7 27 3E 67 50 99 2C 40 43 E3 76 02 E0 37 C2 E9 61 32 83 86 BB A3 4A 8C”.

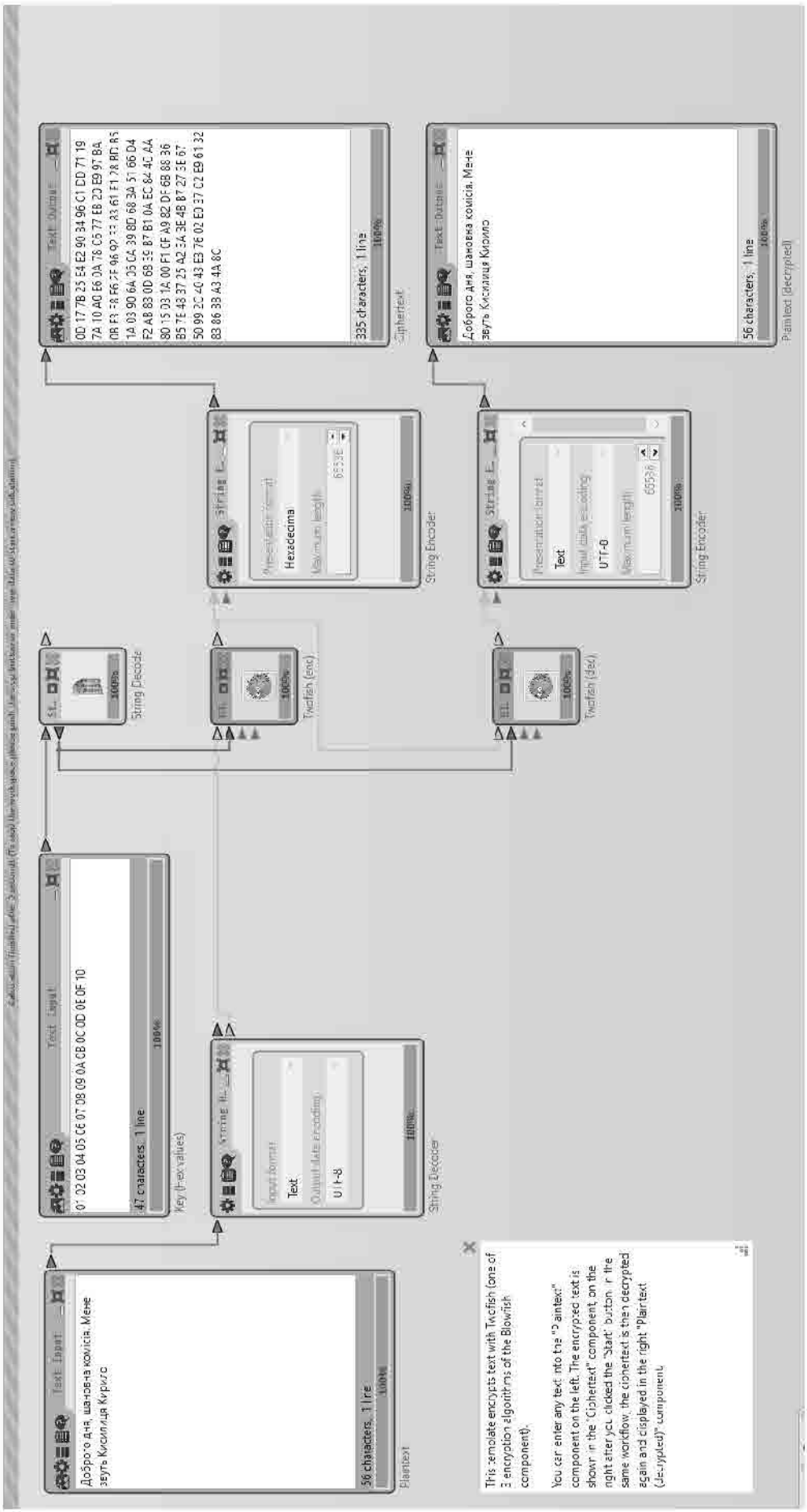



Рисунок 1.24. Результат моделювання поза методом Twofish

## 1.8 Реалізації програмного застосунку задля захищення операторських файлів

### 1.8.1 Формалізація вимог щодо програмного застосунку

Сформульовано наступний список вимог щодо програмного застосунку:

1. Функціональні вимоги:

проте) програма містить працювати у середовищі Windows NT, існувати сумісною із ОС Windows 7-11;

б) програма містить виконувати кодування файлів оператора поза методом 2-fish, причому:

– користувач має мати змогу обрати чи створити шифроключ кодування;

– користувач має мати змогу обрати тип криптоконтейнеру;

– користувач має мати змогу отримати довідку про принципи організації та властивості, криптостійкість методу 2-fish.

у) програма містить виконувати декодування зашифрованих файлів оператора коли користувач знає тільки шифроключ кодування (чи містить збережений шифроключ в файлі);

– програма повинна перевіряти шифроключ кодування так, аби в разі помилкового ключу користувач отримував про це інформацію;

– програма повинна мати змогу перевіряти цілісність криптоконтейнеру.

2. Нефункціональні вимоги:

проте) користувач має мати змогу використовувати програмний застосунок у зручному та простому візуальному інтерфейсі;

б) у процесі організації програма не повинна блокувати взаємодію із файлами, котрі вона опрацьовує.

### 1.8.2 Розробка UML-діаграму варіантів застосування

Діаграми варіантів застосування дозволяють створювати список операцій, котрі виконує система. Як правило даний вид діаграм називають діаграмами функцій, адже набір таких діаграм створює список вимог щодо системи та визначається із множиною функцій, шо виконує система.

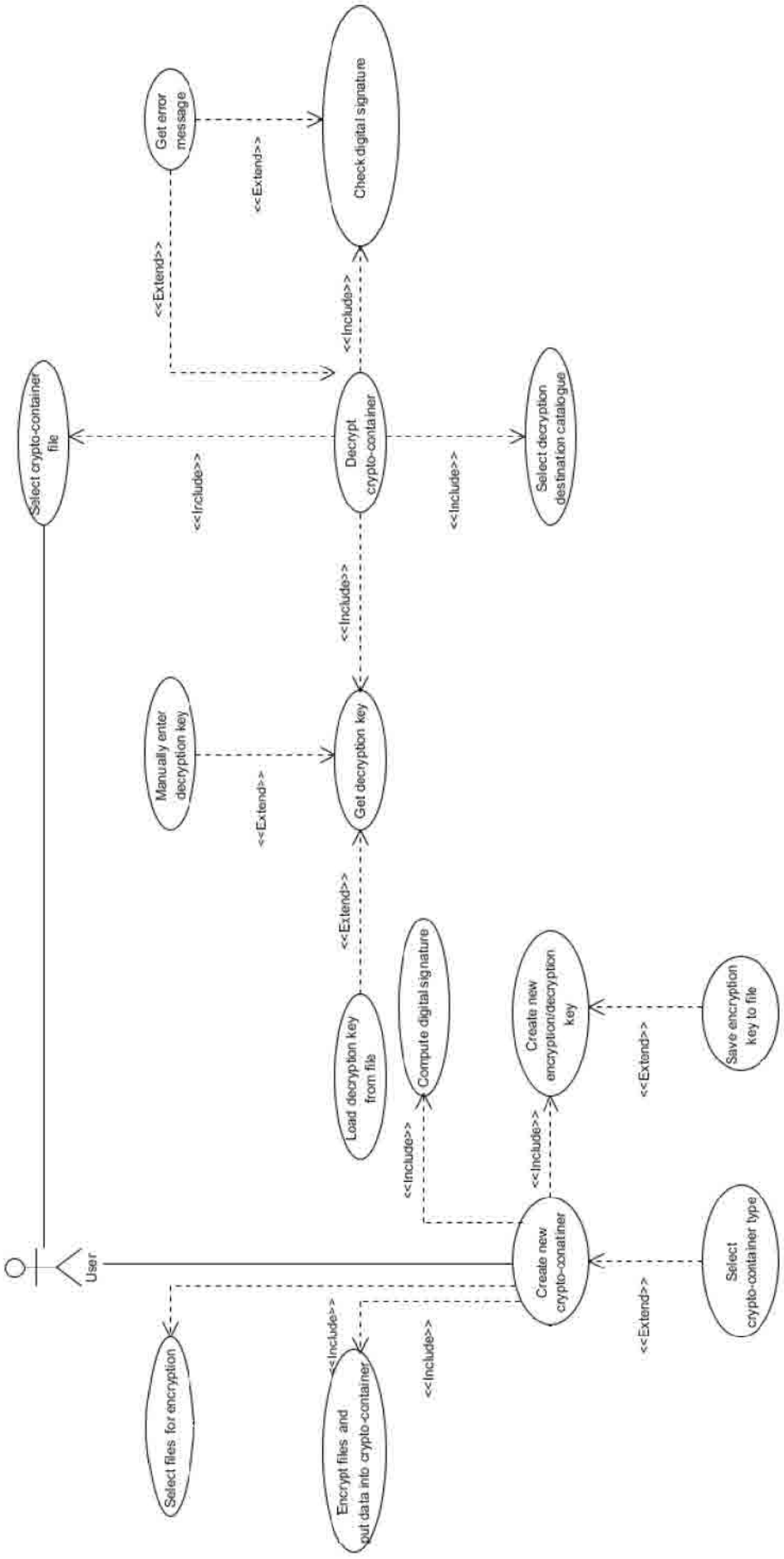



Рисунок 1.25. UML-діаграма варіантів застосування

Кожна діаграми варіантів застосування представляє собою опис послідовності дій і взаємодії поміж акторами і іншими компонентами системи.

ІЗ одним чи кількома варіантами застосування пов'язано дійову особу («actor»). Дійові особи не тотожні фізичним особам, проте визначаються безліччю ролей, котрі користувачі виконують під період взаємодії із системою.

Дійові особи і варіанти застосування зв'язуються один із одним поза поміччю відносини асоціації, зображуваного на діаграмі суцільною прямою лінією. Поміж окремими варіантами застосування можливо вводити зв'язку типу «узагальнення», проте разом з цим типу залежностей зі стереотипами, «включення» («include») та «розширення» («extend»). Даний зв'язок зображується пунктирною лінією зі стрілкою.

Зв'язок типу «розширення» застосовується у тих випадках, коли один варіант застосування подібний іншому, проте несе дещо більше навантаження. Зазвичай «розширення» відображає деякі спеціальні варіанти, шо розширюють можливість основного.

Зв'язок типу «включення» застосовується у тих ситуаціях, коли є що-небудь фрагмент поведінки системи, що доцільно виділити у окремий варіант застосування, наприклад, коли даний фрагмент спроможне існувати включений у кілька інших варіантів застосування. Застосування UML мотивовано тим, шо вона є уніфікованою мовою моделювання, шо широко застосовується в парадигмі об'єктно-орієнтованого програмування. Разом з цим одним із основних мотивів застосування UML стала її поширеність: дуже багато CASE-засобів дозволяють проектувати програмне забезпечення саме мовою UML (Visual Paradigm, Rational Rose, тощо). Задля формалізації функціональних вимог розроблено UML-діаграму варіантів застосування (рис. 1.25).

### 1.8.3 Вибір системної архітектури програмного застосунку

В даній роботі здійснюється реалізація системної архітектури задля standalone-застосунку, а саме програмного забезпечення, шо повністю здійснюється на локальному комп'ютері оператора. Програмний застосунок

містить опрацьовувати файли оператора будь-якого типу і шифрувати їх поза поміччю методу 2-fish, зберігати їх в зашифрованому виді, проте разом з цим виконувати зворотні процедури – дешифрувати зашифровані файли та відновлювати їх початковий стан. Цим чином, узагальнена діаграма розміщення містить вигляд, що приведено на рис. 1.26.

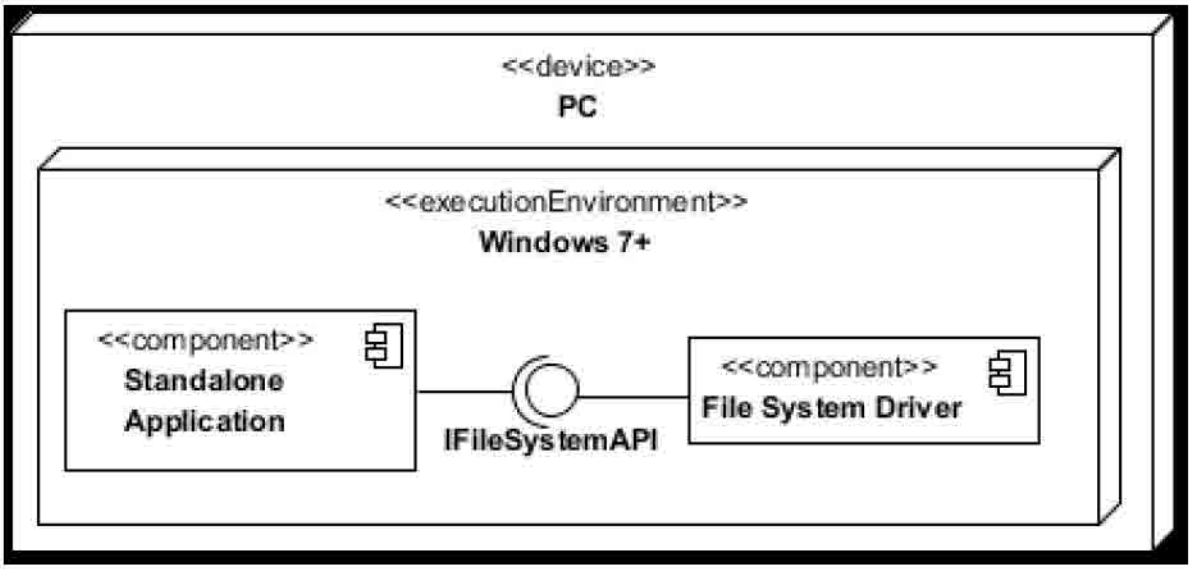



Рисунок 1.26. Deployment UML-діаграма задля програмного застосунку

### 1.8.4 Реалізації діаграми класів задля програмного застосунку

Діаграма класів (class diagram) є статичним представленням структури моделі. Такі діаграми відображають статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст і відношення. Діаграма класів, разом з цим, спроможне містити позначення задля пакетів і спроможне містити позначення задля вкладених пакетів. У цілому, діаграма класів служить задля представлення статичної структури моделі системи у термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти та кооперації, проте разом з цим їхні відносини.

Розроблену концептуальну діаграму класів задля програмного застосунку задля захищення операторських файлів приведено на рис. 1.27. Архітектура складена із декількох пакетів:

1) GUI містить реалізацію графічного інтерфейса. Зміст пакету показано на концептуальному рівні;

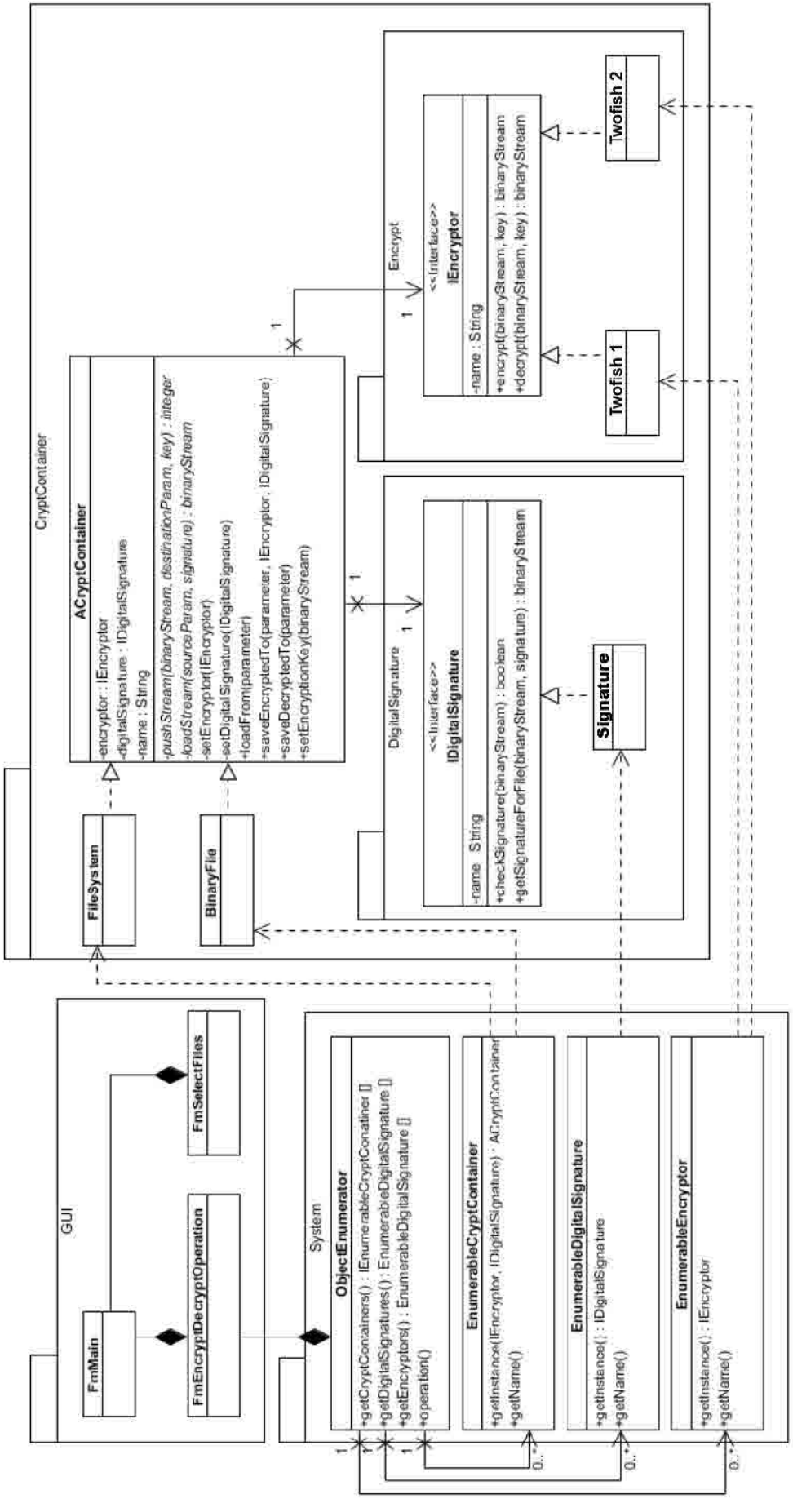



Рисунок 1.27. Концептуальна діаграма класів задля програмного застосунку

2) System містить клас ObjectEnumerator, що призначений задля реєстрації всіх доступних методів (розміщення, кодування, підпису) у системі. У класах EnumerableCryptContainer, EnumerableDigitalSignature, EnumerableEncryptor реалізовано патерн «Factory» задля більш зручного доступу щодо різних методів.

3) CryprtContainer містить абстрактний клас ACryptContainer, що репрезентує деякий криптоконтейнер, і його конкретні реалізації.

проте) DigitalSignature містить інтерфейс IDigitalSignature, що реалізує метод 2-fish.

б) Encrypt містить інтерфейс IEncryptor, що репрезентує метод кодування 2-fish.

У цілому система, використовуючи властивості поліморфізму, спроможне підтримувати будь-яки типи криптоконтейнерів: всі дії задекларовані у абстрактних класах і інтерфейсах, конкретні реалізації тільки слідують їм.

### 1.8.5 Реалізації інтерфейсу і складання програмного коду

Текст основної процедури кодування поза методом 2-fish розробленої програми мовою C++ приведено в Додатку А. Розробку виконано в інтегрованому середовищі розроблювання Embarcadero RAD Studio під керуванням операційної системи Windows 10 Pro x64. На етапі розроблювання візуального інтерфейсу оператора на форму-конструктор були встановлені компоненти кнопок, діалогів, полів вводу, міток (рис. 1.28). При цьому застосовувались об'єкти TButton, TEdit, TLabel, TImage, TOpenDialog, TSaveDialog. Інтерфейс головного вікна розробленої програми задля захищення операторських файлів приведено на рис. 1.29. Передбачені кнопки задля кодування і декодування, генерації нового ключу, довідки задля оператора і виходу із програми. При натисненні кнопки кодування відкривається діалогове модальне вікно, де треба обрати користувацький файл, що треба захистити поза розглянутим методом 2-fish. Перед цим необхідно згненерувати чи ввести уручну шифроключ кодування.

Рисунок 1.28. Етап розроблювання візуального інтерфейсу застосунку

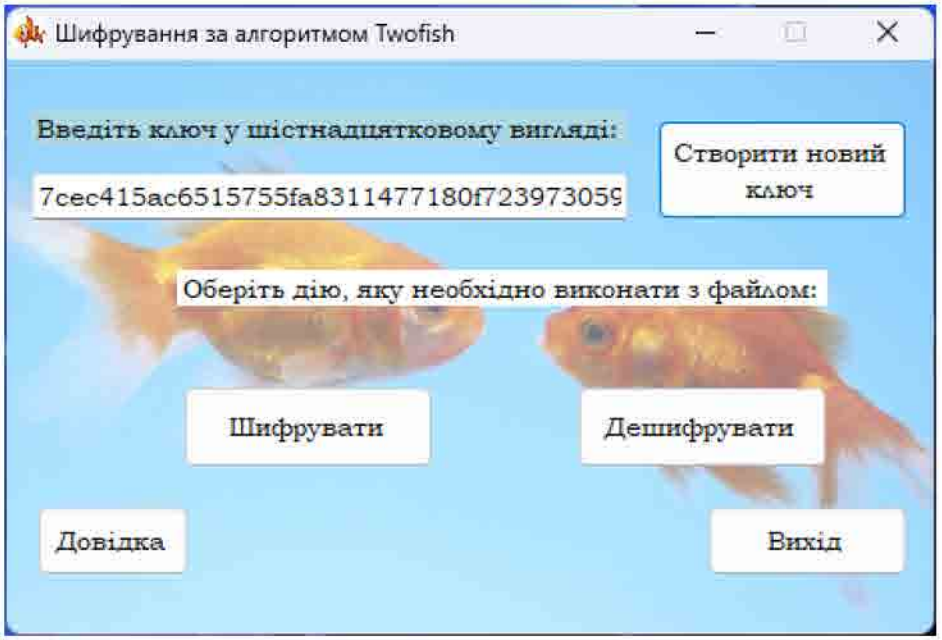



Рисунок 1.29. Головне вікно застосунку задля захищення операторських файлів

Після кодування отриманий зашифрований файл додається в ту саму директорію і отримує нове ім'я із закінченням -enc (рис.1.30). Даний файл вже є захищеним та користувачеві його неможливо використати без попереднього декодування. Задля декодування цього зашифрованого файлу необхідно ввести той самий пароль в шістнадцятковій формі, що використовувався задля кодування, обрати зашифрований файл поза поміччю кнопки «Дешифрувати» (рис. 1.30).

Після закриття вікна вибору файлу відбувається перехід щодо основного вікна. При натисненні кнопки «Дешифрувати» в той самій директорії буде відновлено початковий файл, що отримає в своєму імені закінчення -enc-dec (рис. 1.30). При відсутності згенерованого ключу чи при його невірному форматі кодування/декодування повідомлень буде невдалим, у результаті чого передбачене виведення сповіщення про помилку (рис. 1.31).

Задля перевірки цілісності даних генерується хеш із вихідних даних і порівнюється із тим, що був згенерований щодо цього. В разі, коли вони не співпадають, система видає помилку про то, шо інформація були змінені (рис. 1.32). При натисканні «Yes» програма буде намагатися декодувати пошкоджені інформація, шо призведе щодо неправильних вихідних даних (спотворення даних оператора).

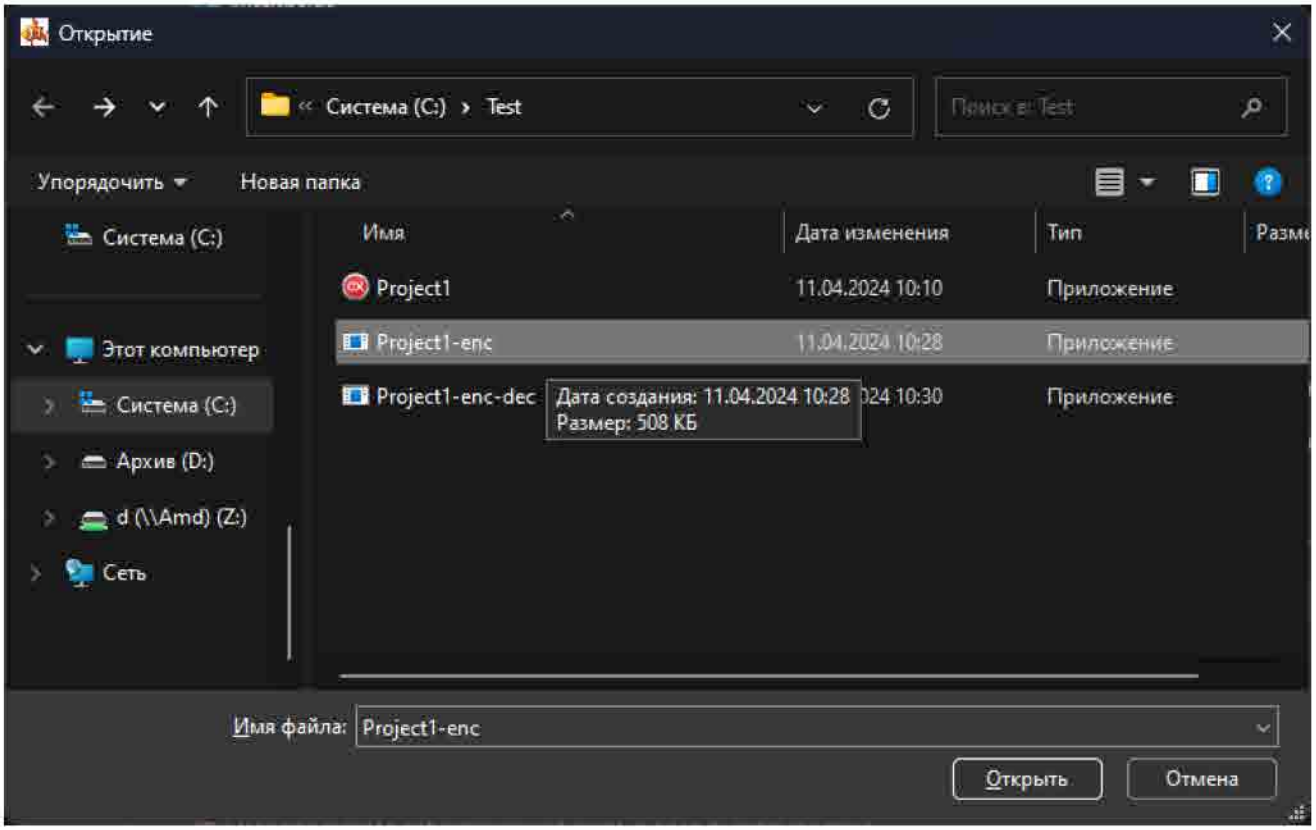



Рисунок 1.30. Вибір зашифрованого файлу задля подальшого декодування

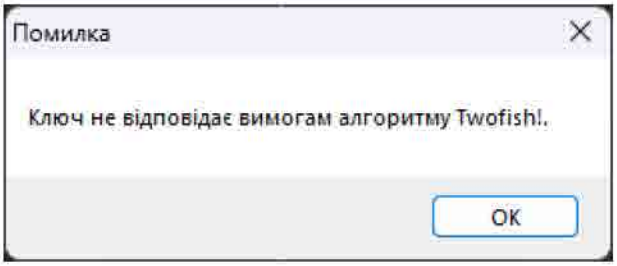



Рисунок 1.31. Сповіщення про помилку ключу

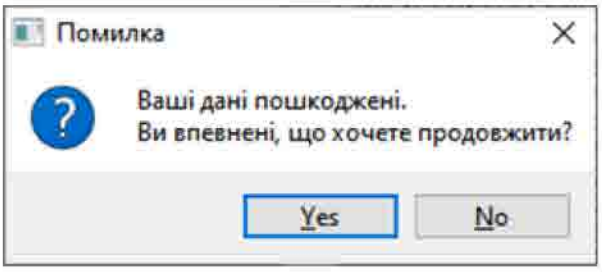



Рисунок 1.32. Сповіщення про порушення цілісності даних

# 2 ЕКОНОМІЧНИЙ РОЗДІЛ

## 2.1 Резюме

Тема даного дипломного проекту «Програмна реалізація методу кодування 2-fish задля захищення операторських файлів». У роботі здійснюється розробка моделі кодеку симетричного блочного методу кодування 2-fish. Упровадження цієї системи дозволить збільшити швидкість обробки даних задля кодування із забезпеченням захищення з зловмисників.

Ефективність кожного програмного продукту визначається його якістю і ефективністю процесу розроблювання. Якість ПП визначається наступними складовими: із точки зору оператора; із позиції застосування ресурсів; виконання вимог щодо програмного забезпечення. Оцінка якості програмного продукту включає визначення трудомісткості та вартості його реалізації.

## 2.2 Визначення трудомісткості розроблювання програмного забезпечення

Тривалість розроблювання програмного продукту залежить з його обсягу, трудомісткості розроблювання, кваліфікації виконавців, проте разом з цим планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, в тисячах умовних машинних команд програми аналога

Таблиця 2.1. Каталог аналогів

| **Найменування ПП** | **Обсяг процедури ПП – $V_o$, усл. машинних командах.** |
|---|---|
| 1. ПП автоматизованих розрахунків | 1300 – 8600 |
| 2. ПП загальної математики та ПП імітаційного моделювання | 1800 – 8800 |
| 3. ПП введення даних | 1060 – 5750 |

В таблиці 2.1 представлені аналоги програмного забезпечення, процедури яких, в більшому чи меншому ступені, виконує розроблений програмний продукт. Задля нашого варіанта виділено сірим кольором.

Вибравши аналог ПП, шо містить Vo у умовних машинних командах, трудомісткості визначати на основі табл.2.2

Таблиця.2.2. Норма часу

| **Обсяг ПП, тис.умов.машинних команд** | **Норма часу, люд/год** |
|---|---|
| 1.00 | 229 |
| 2.00 | 244 |
| 3.00 | 262 |

На підставі отриманого сенсу, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розроблювання ПП, а саме у умовах комп'ютера, Кк=О,7÷О,8): I = 244 х О,7 = 170,08 (люд/годин).

Трудомісткість програмного продукту визначаємо по кожному етапу розроблювання окремо на підставі трудомісткості аналога із урахуванням складності

розроблювання, ступеня новизни та ступеня застосування у розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \qquad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \qquad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \qquad (2.3)$$

Задля розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розроблювання (див. табл. 2.3.);

Кн – поправочний коефіцієнт, шо враховує ступінь новизни (див. табл. 2.4.);

Кт – поправочний коефіцієнт, що враховує ступінь застосування у розробці типових програм (див. табл. 2.5).

Таблиця 2.3. Сенсу питомих коефіцієнтів трудомісткості стадії у загальній трудомісткості розроблювання ПП

| **Код стадії** | **Ступінь новизни** | | |
|---|---|---|---|
| | ПРОТЕ | Б | У |
| ТЗ ($L_1$) | О,15 | О,12 | О,12 |
| ТП ($L_2$) | О,16 | О,15 | О,11 |
| РП ($L_3$) | О,55 | О,58 | О,61 |

Задля нашого варіанта виділено сірим кольором.

Таблиця 2.4. Сенсу поправочного коефіцієнта, що враховує ступінь новизни

| **Код ступеня новизни** | **Ступінь новизни** | **Сенсу $К_н$** |
|---|---|---|
| ПРОТЕ | Принципово нові ПП | 1,75 – 1,2 |
| Б | ПП – розвиток визначеного параметричного ряду | 1,О – О,8 |
| У | ПП маючий аналог | О,7 |

Задля нашого варіанта виділено сірим кольором.

Таблиця 2.5. Сенсу коефіцієнта ступеня застосування у розробці типових програм

| **Ступінь охоплення реалізованих функцій розроблювального ПП типовими програмами, %** | **Сенсу $К_т$** |
|---|---|
| 60 та вище | О,6 |
| 40-60 | О,7 |
| 20-40 | О,8 |
| Щодо 20 | О,9 |

Задля нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$Т_{тз}=I*L_1*К_н= 170,08*О,12*О,7= 14,35 \text{ (люд/годин)} \quad (2.4)$$

Трудомісткість розроблювання технічного проекту

$$Т_{тп}=I*L_2*К_н=170,08*О,11*О,7 = 13,10 \text{ (люд/годин)} \quad (2.5)$$

Трудомісткість розроблювання робочого проекту

$$T_{рп}=I*L_з*K_н*K_т=170,08*O,61*O,7*O,7=\ 50,81\ \text{(люд/годин)} \quad (2.6)$$

Задля подальших розрахунків визначили число папера, витраченого на кожен етап: технічне завдання $N_{тз}$= 2 (стр), розробка ТП $N_{тп}$=12 (стр), розробка робочого проекту $N_{рп}$=14(стр), пояснювальна записка таким чином $N_{пз}$=22(стр) Обчислення зведений в таблицю 2.6.

Таблиця 2.6. Обчислення трудомісткості ПП

| **Найменування етапів** | **Обчислення, годин** | | |
|---|---|---|---|
| 1.ТЗ | $T_{Ртз}$=14,35 | $T_{кк}=O,7*N_{тз}=O,7*2=1,4$ | $T_{нк}=O,15*N_{тз}=O,15*2=O,3$ |
| 2.Розробка ТП | $T_{Ртп}$=13,10 | $T_{кк}=O,7*N_{тп}=O,7*12=8,4$ | $T_{нк}=O,15*N_{тп}=O,15*12=1,8$ |
| 3.Розробка РП | $T_{Ррп}$=50,81 | $T_{кк}=O,7*N_{рп}=O.7*14=9,8$ | $T_{нк}=O,15*N_{рп}=O,15*14=2,1$ |
| 4.Розробка ПЗ | $T_{пз}=1,5*22*N_{пз}=33$ | $T_{кк}=O,7*N_{тз}=O,7*22=15,4$ | $T_{нк}=O,15*N_{пз}=O,15*22=3,3$ |
| Усього, у т.ч.: | $\Sigma T$=153,76 | | |
| - на розробку | $\Sigma T_р$=111,26 | | |
| - контроль керівника | | $\Sigma T_{кк}$=35 | |
| - нормоконтроль | | | $\Sigma T_{нк}$=7,5 |

## 2.3 Обчислення ціни програмного продукту

Задля визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, витрати на розробку ПП. Обчислення основної заробітної плати виконавців приведений в таблиці 4.6. Таким чином щодо статті 8 «Закону про Державний бюджет України на 2023» встановлено мінімальну заробітну плату в місячному розмірі із 1 січня 2023 року - 6700 гривень; мінімальну погодинну тарифну ставку – 40.46 грн.

Таблиця 2.7. Обчислення основної заробітної плати виконавців

| **Найменування робіт** | **Трудомісткість робіт, години** | **Погодинна тарифна ставка, грн.** | **Обчислення, грн.** |
|---|---|---|---|
| 1.Розробка ПП | 111,26 | 45,00 | 5006,25 |
| 2.Контроль керівника | 35 | 100,00 | 3500 |
| 3.Нормоконтроль | 7,5 | 80.00 | 600 |
| Усього | - | - | ΣЗо= 9106,25 |

Зробимо обчислення матеріальних витрат на розробку ПП. Обчислення зведемо у таблицю 2.8.

Таблиця 2.8. Обчислення матеріальних витрат на розробку ПО

| **Найменування матеріальних витрат** | **Тип, модель** | **Число** | **Ціна одиниці, грн.** | **Вартість, грн.** |
|---|---|---|---|---|
| Папір | Лист А4 | 50 | З.О | 150,00 |
| Транспортно– заготівельні витрати (10%) | | | | $Втр\_з = 0,1 \times Вм1$ = 150*О,1= 15,О |
| Усього | | | | $В_м = В_{мі} + В_{тр\_із}$= 165,О |

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості у цілому ПП поза формою, приведеною у таблиці 2.9.

Таблиця 2.9. Обчислення статей витрат планової собівартості

| **Стаття витрат** | **Сенсу, грн.** | **Формула розрахунку** |
|---|---|---|
| 1. Матеріали | 165,О | $В_м$= 165,О |
| 2. Основна заробітна плата | 9106,25 | $З_о$ = 9106,25 |
| 3.Додаткова заробітна плата | 910,63 | $Зд = 0,15 \times Зо$ =О,15*9106,25 |
| 4.Відрахування щодо єдиного фонду соціального внеску | 2203,71 | $Вє.с.в. = 0,22 \times (Зо + Зд)$=О,22*(9106,25+ 910,63) |
| 5. Накладні витрати | 4553,12 | $Внак. = 0,3 \times Зо$ =О,5 * 9106,25 |
| 6. Повна собівартість | 16938,71 | $С_{пов}$=$В_м$+$З_о$+$З_д$+Вє.с.у.+Внак.=165,О+ 9106,25+910,63+2203,71+4553,12 |

Розмірність прибутку, шо включається у ціну, визначаємо по наступній формулі:

$$П=(С_п*Р)/100=(16938,71*10)= 1693,87 \qquad (2.7)$$

# 3 РОЗДІЛ ОХОРОНИ ПРАЦІ І ТЕХНІКИ БЕЗПЕЧНОСТІ

Широке впровадження комп'ютерної техніки, шо дає змогу автоматизувати багато рутинних операцій комп'ютерної обробки даних, одержати доступ щодо численних джерел даних, швидко проводити потрібні розрахунки тощо, підвищує продуктивність праці. Проте активне впровадження в практику персональних комп'ютерів містить та негативну сторону – із'являються фактори, котрі несприятливо впливають на здоров'я працюючої людини.

Згідно темі дипломного проекту робоче місце оператора послуг складається із персонального комп'ютеру із програмним забезпеченням і призначено задля розроблювання моделі кодеку на базі методу симетричного кодування. Таким чином задля нього застосовуються звичайні вимоги безпечності праці задля оператора персонального комп'ютеру.

## 3.1 Аналіз небезпечних та шкідливих факторів, шо впливають на оператора ПК

Показниками гігієнічних вимог є рівень освітлення, температура, вологість, шум, вібрація, токсичність, загазованість, обмежена загальна м'язова активність (гіподинамії), дія електростатичного поля, неіонізуючих і іонізуючих електромагнітних випромінювань.

## 3.2 Гігієнічні вимоги щодо виробничого середовища

Державні санітарні правила та норми організації із візуальними дисплейними терміналами електронно-обчислювальних машин ДСАНПіН 3.3.2.007-98 «Гігієнічні вимоги щодо організації організації із візуальними дисплейними терміналами електронно-обчислювальних машин» призначені задля запобігання несприятливої дії на працівників шкідливих факторів, котрі супроводжують роботу із ВДТ.

### 3.2.1 Вимоги щодо приміщення

Розміщення робочих місць із ВДТ ЕОМ та ПЕОМ в підвальних приміщеннях, на цокольних поверхах заборонено. Задля приміщень, котрі призначені задля організації із ВДТ, доцільно обрати орієнтацію вікон на північ чи північний схід. На вікнах мають існувати жалюзі, шо регулюються, чи штори, шо дають можливість іх повністю закривати.

Об'ємно-планувальні рішення будівель і приміщень, де експлуатуються відеодисплейні термінали (ВДТ) містять відповідати вимогам ДСанПІН 3.3.2.007-98. Площа на одне робоче місце задля програмістів повинна складати не менше 6 кв.м., проте об'єм – не менше 20 куб.м. Стіни мають існувати пофарбовані матовою фарбою. При приміщеннях із ВДТ містять існувати обладнані побутові приміщення задля відпочинку, психологічного розвантаження тощо

### 3.2.2 Освітлення

Задля освітлення приміщення, в якому працює програміст, застосовується змішане освітлення, а саме сполучення природного та штучного освітлення. Задля загального освітлення приміщення, де перебуває робоче місце програміста, застосовуються газорозрядні лампи типу ЛД. Нормами задля даних робіт установлена необхідна освітленість робочого місця ЕН=300 лк (задля робіт високої точності, коли найменший розмірність об'єкта розрізнення дорівнює О,3 – О,5 мм). Дана норма освітленості у цілому виконана.

### 3.2.3 Шум

В робочих приміщеннях основними джерелами акустичних шумів є шуми ПЕОМ. ЕОМ є разом з цим джерелами шумів електромагнітного походження (коливання елементів електромеханічних пристроїв під впливом змінних магнітних полів). Крім того, в даних приміщеннях, виникає структурний шум, а саме шум, випромінюваний поверхнями коливних конструкцій стін, перекриттів, перегородок будинку в звуковому діапазоні частот.. Задля усунення чи ослаблення несприятливих шумових впливів доцільно ізолювати робочі

приміщення, розміщаючи їх в частинах будинку, найбільш вилучених з міського шуму - розташованих в глибині будинку, звернених вікнами в двір. Перевіряти герметичність корпусів комп'ютерів і своєчасно міняти вентилятори охолодження.

## 3.3 Вимоги щодо організації робочого місця працівника

Конструкція робочого місця оператора ПК та взаємне розташування всіх його елементів (сидіння, органи керування, засобу відображення даних) відповідають антропометричним, фізіологічним та психологічним вимогам, проте разом з цим характеру організації. Конструкція робочих меблів повинна забезпечувати можливість індивідуального регулювання таким чином росту працюючих задля підтримки зручної пози. Робочий стіл має існувати пофарбований матовою фарбою. Дисплей розташований так, шо його верхній край перебуває на рівні очей на відстані близько 70 см, шо укладається у в припустимі рамки з 60 щодо 90 см. Частота мерехтіння екрана fмер=100 Гц, шо відповідає умові fмер>70 Гц.

Робоче місце розташоване перпендикулярно віконним прорізам, це зроблено із тією метою, аби виключити пряму та відбиту мерехтливість екрана з вікон та приладів штучного освітлення.

## 3.4 Мікроклімат

Параметри мікроклімату, іонного складу повітря, вміст шкідливих речовин на робочих місцях, оснащених ПК мають відповідати вимогам ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень».

Задля підтримки допустимих значень мікроклімату і концентрації позитивних і негативних іонів необхідно передбачати установки чи прилади зволоження і/чи штучної іонізації, кондиціювання повітря. Рівні інфрачервоного випромінювання не мають перевищувати граничних

Рівні інфрачервоного випромінювання не мають перевищувати граничних таким чином щодо ГОСТ 12.1.005. Вміст озону у повітрі робочої зони не має перевищувати О,1мг/м3; вміст оксидів азоту - 5мг/м3; вміст пилу - 4мг/м3.

## 3.5 Електробезпека

Приміщення де застосовуються імпульсні джерела живлення таким чином щодо ОНТП24-86 та ПУЕ-87 відноситься щодо класу приміщень без підвищеної небезпеки поразки персоналу електричним струмом, адже відносна вологість повітря не перевищує 75%, температура не більш 35°С, відсутні хімічно агресивні середовища. Живлення електроприладів усередині приміщення здійснюється з двухфазної мережі із заземленою нейтралю напругою 220 У та частотою 50 Гц із використанням автоматів токового захищення. В приміщенні повинна існувати застосована схема заземлення. Ураження людини електричним струмом спроможне відбутися в разі:

1) дотику щодо відкритих струмоведучих частин;

2) в результаті дотику щодо струмопровідних неструмоведучих елементів устаткування, шо опинилися під напругою у результаті порушення ізоляції чи із інших причин.

Заземлення повинно існувати зроблено поза поміччю гнучкого сплетеного мідного проводу діаметром порядку 1,5 мм2

Задля зменшення значень напруг дотику та відповідних їм величин струмів, при нормальному та аварійному режимах організації устаткування необхідно виконати повторне захисне заземлення нульового проводу.

Таким чином щодо ГОСТ-12.2.007.О-75 устаткування (крім ЕОМ - II клас) відноситься щодо I класу, воно містить робочу ізоляцію таким чином щодо вимог ГОСТ 12.1.009-76. Підключення устаткування виконане таким чином щодо вимог ПБЕ і ПУЕ. Додаткових заходів по електробезпечності не потрібно.

## 3.6 Пожежна безпека

Робоче приміщення таким чином щодо ПБЕ і ОНТП 24–86 по вибухово-пожарній безпеці можливо віднести щодо категорії "У".

Можливими причинами виникнення пожежі у приміщенні є:

1) коротке замикання проводки;

2) користування побутовими електрорадіоприладами.

3) не дотримання умов протипожежної безпечності.

В зв'язку із цим таким чином щодо ПУЕ необхідно передбачити наступні заходи щодо пожежної безпечності: ретельна ізоляція всіх струмоведучих провідників щодо робочих місць; періодичний огляд та перевірка ізоляції; суворе дотримання норм протипожежної безпечності на робочому місці.

Задля гасіння пожеж на робочому місці оператора ПК застосовують вуглекислотні і порошкові вогнегасники.

– Вуглекислотні вогнегасники випускаються як ручні (ВВК-5);

– Порошкові вогнегасники ВП-2, ВП-5, ВП-10 і ін.


Рисунок 3.1. Засоби пожежогасіння

ІЗ метою своєчасного оповіщення, на дільниці необхідно встановити протипожежну сигналізацію. Проходи і запасні виходи мають існувати вільними. Пожежний щит має розміщуватись у доступному місці і містити первинні засоби пожежогасіння: вогнегасник, лопату, відро, простирадло, ящик із піском. Відповідальний поза пожежну безпеку керівник виробничої дільниці.

# ВИСНОВКИ

Під період виконання дипломного проекту були проаналізовані популярні блокові методи симетричного кодування даних, визначені їх переваги і недоліки, методи щодо їх злому, вивчено роботу надійного методу кодування 2-fish, що застосовано задля реалізації простого програмного додатку під ОС Windows задля захищення операторських файлів.

Поза поміччю програмного забезпечення задля моделювання криптографічних методів CrypTool2 виконано моделювання організації методу 2-fish, шо є одним із найбільш ефективних і захищених методів симетричного блочного кодування у даний період. Це дозволило виконати розробку візуального програмного застосунку на базі кодеку 2-fish мовою C++ засобами інтегрованого середовища розроблювання Embarcadero RAD Studio.

Використаний метод 2-fish містить великий запас стійкості щодо злому в порівнянні із іншими. Додатковою перевагою методу 2-fish є швидкість кодування елементу даних. Практична реалізація методу 2-fish дає можливість частково вирішити проблему захищення конфіденційності та цілісності даних. Особливістю і характерним показником блочних методів є багатократне та непряме застосування матеріалу ключу.

Вивчені при розробці програмного забезпечення методи кодування спроможні існувати корисними не тільки задля захищення локальних даних оператора, проте та задля реалізації web-сервісу задля захищення хмарних даних. Розроблений програмний застосунок підтримується операційними системами сімейства Windows і не вимагає багато ресурсів задля своєї організації.

Подальше вдосконалення організації запропонованого програмного забезпечення спроможне полягати в додаванні можливості вибору протяжності ключу (128, 192, 256 бітів) задля кодування даних, застосуванні методів кодування задля передачі ключу, надійного зберігання, можливості застосування генераторів випадкових значень, котрі будуть залежати з величин, шо не можливо передбачити, збільшенні циклів кодування.

# ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ


1. Бобало Ю.Я. Інформаційна безпека: навчальний посібник – Львів: Видавництво Львівської політехніки, 2019. – 580 с.

2. Дудикевич В.Б. Основи інформаційної безпеки. Вінниця: ВНТУ, 2018

3. Глинчук Л.Я. Криптологія: навч.-метод. посіб. / Людмила Ярославівна Глинчук – Луцьк: Вежа-Друк, 2014. – 164 с.

4. Малайчук В.П. Перешкодостійке кодування повідомлення електронного підпису // Актуальні проблеми та інформаційних технологій автоматизації. – Дніпро: ДНУ. Збірник наукових праць. Том 21. 2017. С. 123–131.

5. Креневич А.П. Алгоритми і структури даних. Підручник. – К.: ВПЦ "Київський Університет", 2021. – 200 с.

6. Мосіюк О.О. Операційні системи та системне програмування: навчально-методичний посібник. Житомир: Вид-во ЖДУ ім. Івана Франка, 2022. – 76 с.

7. Богач І. В., Довгалець С. М., Дубовой В. М. Алгоритми розв'язання задач з програмування. Розв'язник. Вінниця: ВНТУ, 2017. – 119 с.

8. Бублик В. В. Об'єктно-орієнтоване програмування. Київ: ІТ-книга, 2015. – 640 с.

9. Stroustrup B. A Tour of C++ (Second Edition). – Addison-Wesley, 2018. – 240 p. – ISBN 978-0-13-499783-4.

10. Authenticated encryption [Електронний ресурс]:
https://www.cryptopp.com/wiki/Authenticated_Encryption

11. Symmetric And Asymmetric Key Cryptography: All You Need To Know In 3 Points. [Електронний ресурс]:
https://www.jigsawacademy.com/blogs/cyber-security/symmetric-and-asymmetric-key-cryptography/

12. Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption [Електронний ресурс]:
https://csrc.nist.gov/publications/detail/sp/800-38g/rev-1/draft

13. Schneier B. Twofish: A 128–Bit Block Cipher. The Twofish Encryption Algorithm (researchgate.net).

# ДОДАТОК А. Фрагмент коду базових функцій шифрування/ дешифрування за алгоритмом TwoFish мовою С++

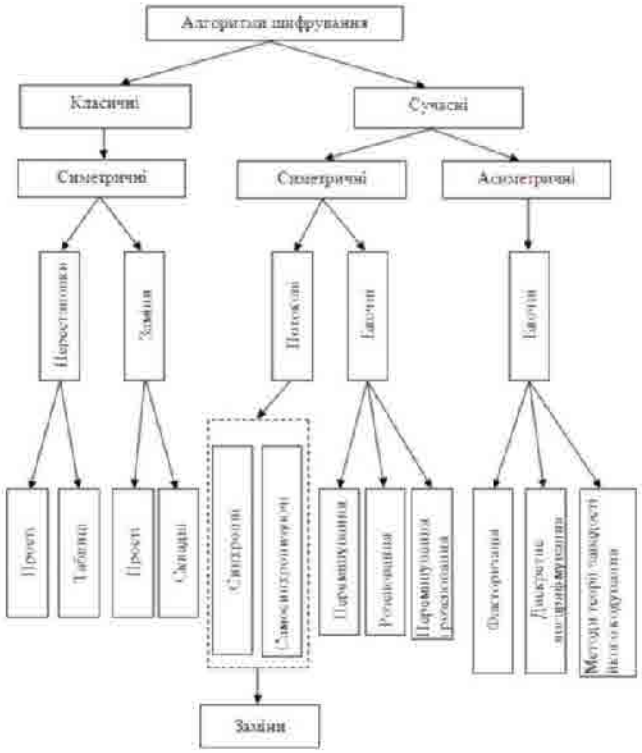
```
#include "pch.h"
#include "twofish.h"
#include "secblock.h"
#include "misc.h"

NAMESPACE_BEGIN(CryptoPP)

// compute (c * x^4) mod (x^4 + (a + 1/a) * x^3 + a * x^2 + (a + 1/a) * x + 1)
// over GF(256)
static inline unsigned int Mod(unsigned int c)
{
    static const unsigned int modulus = 0x14d;
    unsigned int c2 = (c<<1) ^ ((c & 0x80) ? modulus : 0);
    unsigned int c1 = c2 ^ (c>>1) ^ ((c & 1) ? (modulus>>1) : 0);
    return c | (c1 << 8) | (c2 << 16) | (c1 << 24);
}

// compute RS(12,8) code with the above polynomial as generator
// this is equivalent to multiplying by the RS matrix
static word32 ReedSolomon(word32 high, word32 low)
{
    for (unsigned int i=0; i<8; i++)
    {
        high = Mod(high>>24) ^ (high<<8) ^ (low>>24);
        low <<= 8;
    }
    return high;
}

inline word32 Twofish::Base::h0(word32 x, const word32 *key, unsigned int kLen)
{
    x = x | (x<<8) | (x<<16) | (x<<24);
    switch(kLen)
    {
#define Q(a, b, c, d, t) q[a][GETBYTE(t,0)] ^ (q[b][GETBYTE(t,1)] << 8) ^ (q[c][GETBYTE(t,2)] << 16) ^
(q[d][GETBYTE(t,3)] << 24)
    case 4: x = Q(1, 0, 0, 1, x) ^ key[6];
    // fall through
    case 3: x = Q(1, 1, 0, 0, x) ^ key[4];
    // fall through
    case 2: x = Q(0, 1, 0, 1, x) ^ key[2];
            x = Q(0, 0, 1, 1, x) ^ key[0];
    }
    return x;
}

inline word32 Twofish::Base::h(word32 x, const word32 *key, unsigned int kLen)
{
    x = h0(x, key, kLen);
    return mds[0][GETBYTE(x,0)] ^ mds[1][GETBYTE(x,1)] ^ mds[2][GETBYTE(x,2)] ^
mds[3][GETBYTE(x,3)];
}

void Twofish::Base::UncheckedSetKey(const byte *userKey, unsigned int keylength, const
NameValuePairs &)
{
    AssertValidKeyLength(keylength);
```

```
    unsigned int len = (keylength <= 16 ? 2 : (keylength <= 24 ? 3 : 4));
    SecBlock<word32> key(len*2);
    GetUserKey(LITTLE_ENDIAN_ORDER, key.begin(), len*2, userKey, keylength);

    unsigned int i;
    for (i=0; i<40; i+=2)
    {
        word32 a = h(i, key, len);
        word32 b = rotlConstant<8>(h(i + 1, key + 1, len));
        m_k[i] = a+b;
        m_k[i + 1] = rotlConstant<9>(a + 2 * b);
    }

    SecBlock<word32> svec(2*len);
    for (i=0; i<len; i++)
        svec[2*(len-i-1)] = ReedSolomon(key[2*i+1], key[2*i]);
    for (i=0; i<256; i++)
    {
        word32 t = h0(i, svec, len);
        m_s[0*256+i] = mds[0][GETBYTE(t, 0)];
        m_s[1*256+i] = mds[1][GETBYTE(t, 1)];
        m_s[2*256+i] = mds[2][GETBYTE(t, 2)];
        m_s[3*256+i] = mds[3][GETBYTE(t, 3)];
    }
}

#define G1(x) (m_s[0*256+GETBYTE(x,0)] ^ m_s[1*256+GETBYTE(x,1)] ^
m_s[2*256+GETBYTE(x,2)] ^ m_s[3*256+GETBYTE(x,3)])
#define G2(x) (m_s[0*256+GETBYTE(x,3)] ^ m_s[1*256+GETBYTE(x,0)] ^
m_s[2*256+GETBYTE(x,1)] ^ m_s[3*256+GETBYTE(x,2)])

#define ENCROUND(n, a, b, c, d) \
    x = G1 (a); y = G2 (b); \
    x += y; y += x + k[2 * (n) + 1]; \
    (c) ^= x + k[2 * (n)]; \
    (c) = rotrConstant<1>(c); \
    (d) = rotlConstant<1>(d) ^ y

#define ENCCYCLE(n) \
    ENCROUND (2 * (n), a, b, c, d); \
    ENCROUND (2 * (n) + 1, c, d, a, b)

#define DECROUND(n, a, b, c, d) \
    x = G1 (a); y = G2 (b); \
    x += y; y += x; \
    (d) ^= y + k[2 * (n) + 1]; \
    (d) = rotrConstant<1>(d); \
    (c) = rotlConstant<1>(c); \
    (c) ^= (x + k[2 * (n)])

#define DECCYCLE(n) \
    DECROUND (2 * (n) + 1, c, d, a, b); \
    DECROUND (2 * (n), a, b, c, d)

typedef BlockGetAndPut<word32, LittleEndian> Block;

void Twofish::Enc::ProcessAndXorBlock(const byte *inBlock, const byte *xorBlock, byte *outBlock) const
{
    word32 x, y, a, b, c, d;

    Block::Get(inBlock)(a)(b)(c)(d);
```

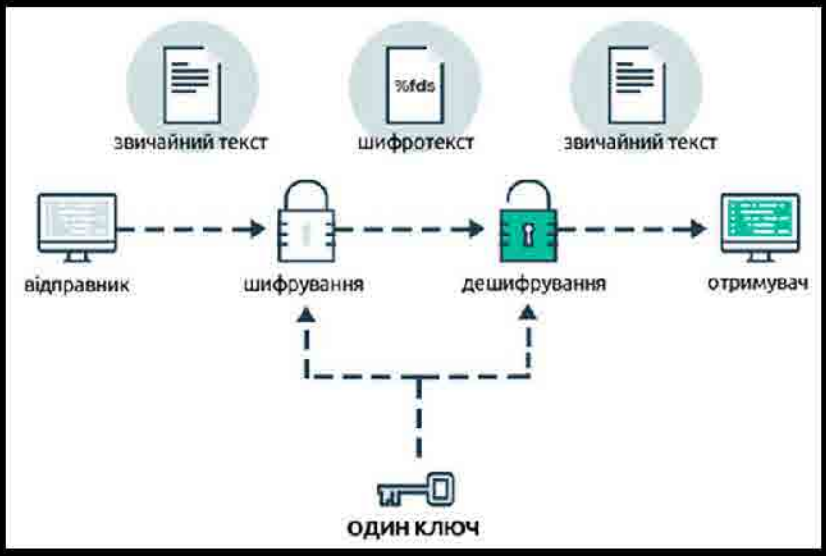
```
    a ^= m_k[0];
    b ^= m_k[1];
    c ^= m_k[2];
    d ^= m_k[3];

    const word32 *k = m_k+8;
    ENCCYCLE (0);
    ENCCYCLE (1);
    ENCCYCLE (2);
    ENCCYCLE (3);
    ENCCYCLE (4);
    ENCCYCLE (5);
    ENCCYCLE (6);
    ENCCYCLE (7);

    c ^= m_k[4];
    d ^= m_k[5];
    a ^= m_k[6];
    b ^= m_k[7];

    Block::Put(xorBlock, outBlock)(c)(d)(a)(b);
}

void Twofish::Dec::ProcessAndXorBlock(const byte *inBlock, const byte *xorBlock, byte *outBlock) const
{
    word32 x, y, a, b, c, d;

    Block::Get(inBlock)(c)(d)(a)(b);

    c ^= m_k[4];
    d ^= m_k[5];
    a ^= m_k[6];
    b ^= m_k[7];

    const word32 *k = m_k+8;
    DECCYCLE (7);
    DECCYCLE (6);
    DECCYCLE (5);
    DECCYCLE (4);
    DECCYCLE (3);
    DECCYCLE (2);
    DECCYCLE (1);
    DECCYCLE (0);

    a ^= m_k[0];
    b ^= m_k[1];
    c ^= m_k[2];
    d ^= m_k[3];

    Block::Put(xorBlock, outBlock)(a)(b)(c)(d);
}

NAMESPACE_END
```

# ДОДАТОК Б. Слайди мультимедійної презентації

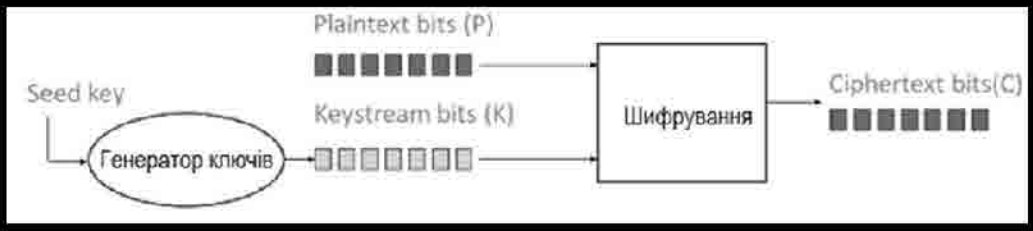
Класифікація існуючих алгоритмів шифрування




Узагальнена схема симетричного шифрування



## Принципи потокового шифрування

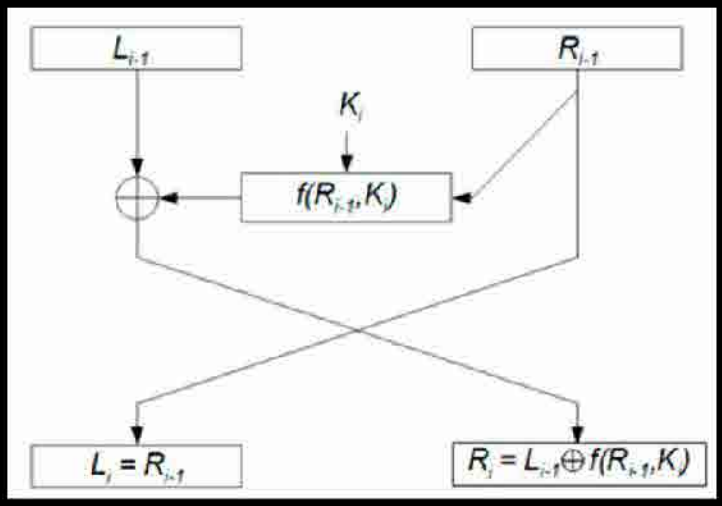



## Принципи блокового шифрування

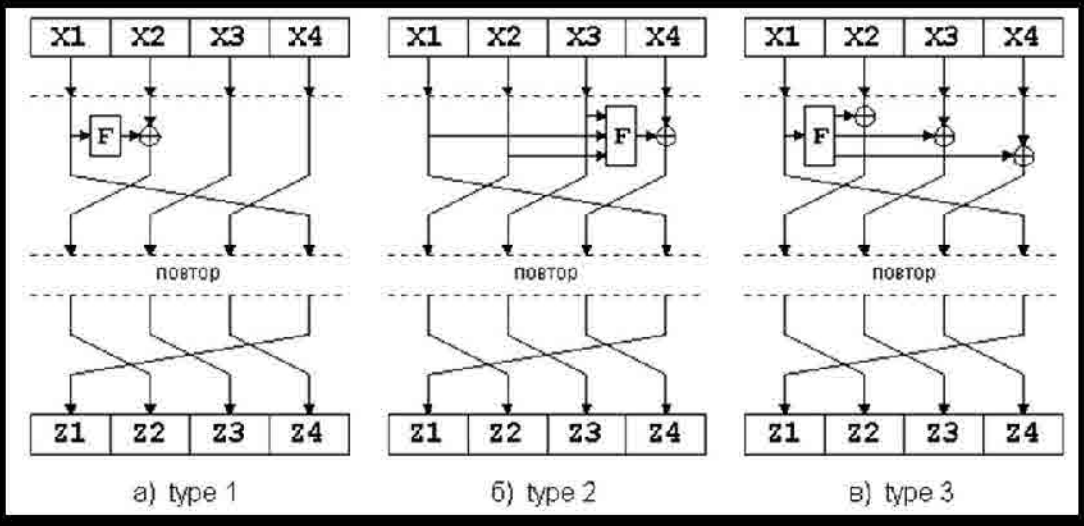

## Бінарні операції при шифруванні

| Бінарні операції | |
|---|---|
| Додавання | $X'=X+V$ |
| Виключне АБО | $X'=X$ XOR $V$ |
| Множення за модулем $2^N+1$ | $X'=(X*V) \bmod (2^N+1)$ |
| Множення за модулем $2^N$ | $X'=(X*V) \bmod (2^N)$ |
| Бітові зсуви | |
| 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 (англ. substitute) | $X'=$Table$[X,V]$ |

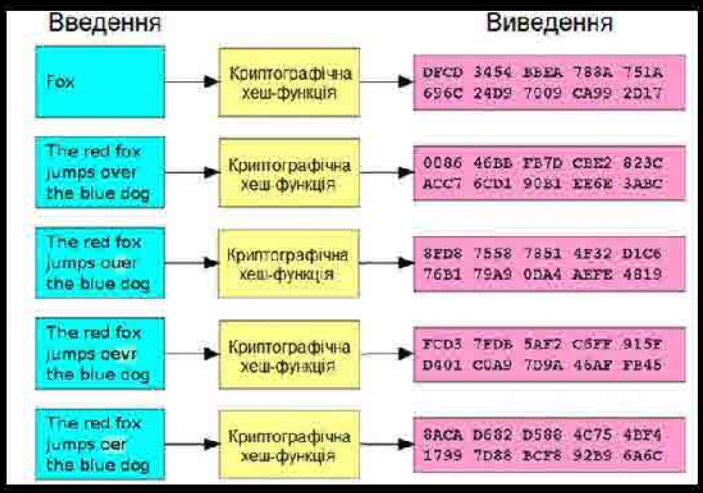
## Структура мережі Фейстеля при шифруванні



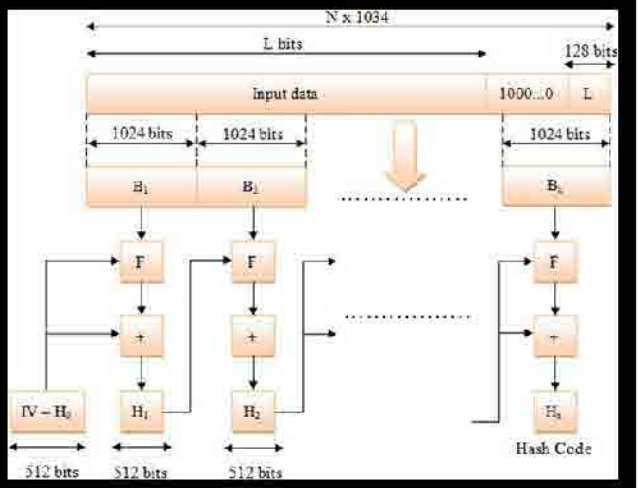

## Принципи модифікації мережі Фейстеля



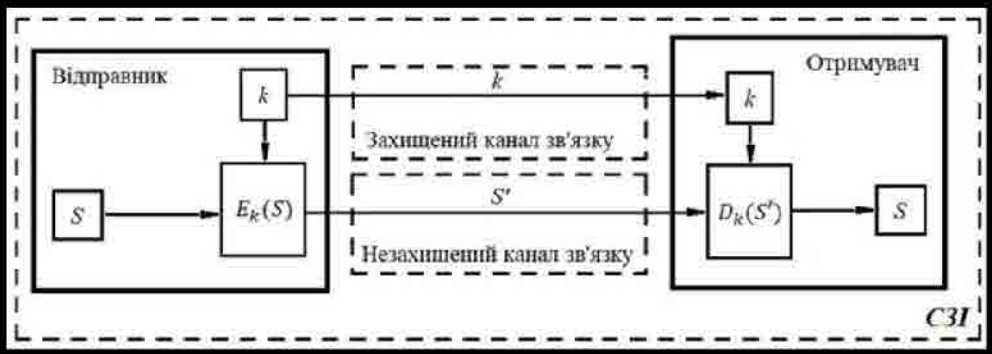
## Принцип хешування за допомогою хеш-функції



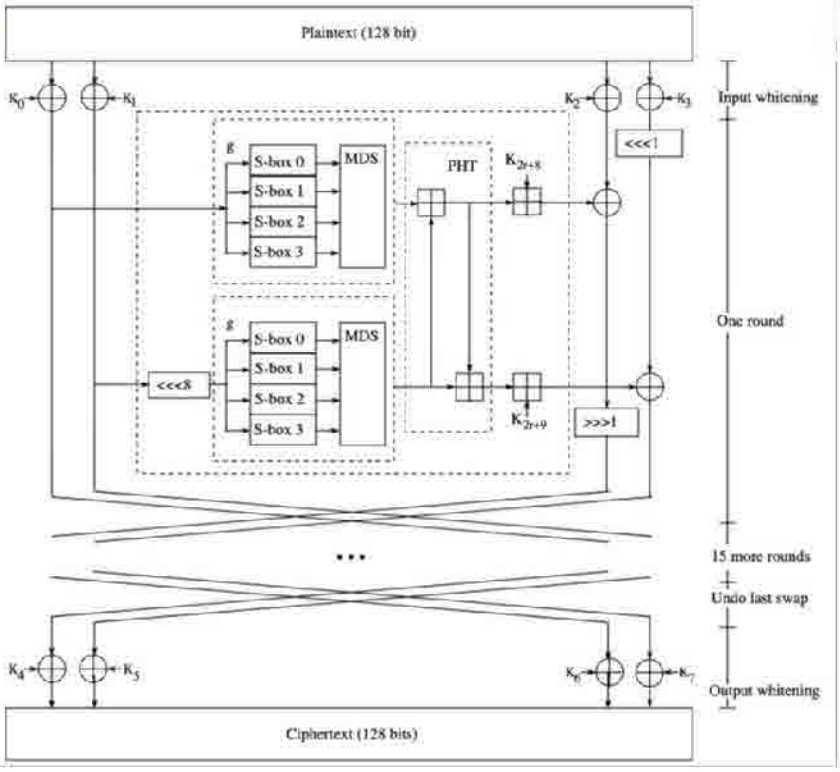

## Структура алгоритму SHA-512



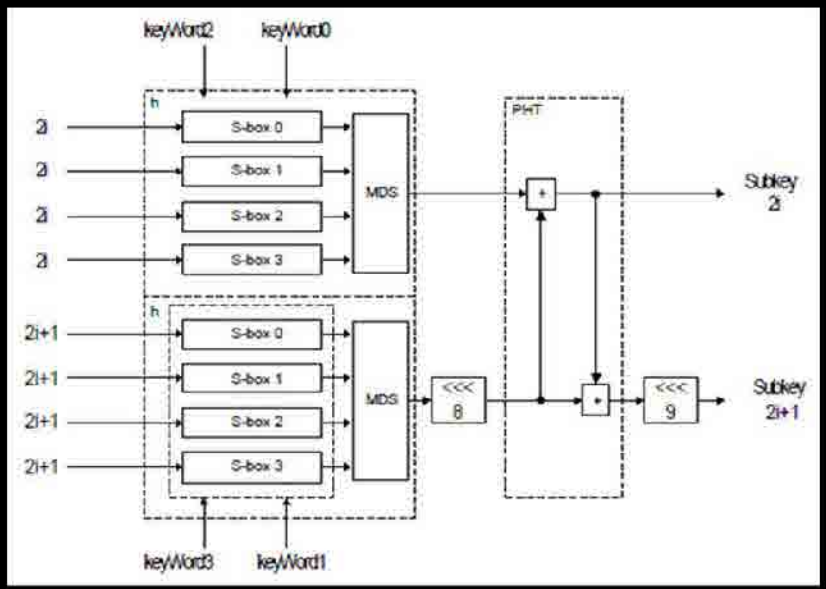

## Структурна схема захисту інформації з застосуванням криптографічного алгоритму Twofish

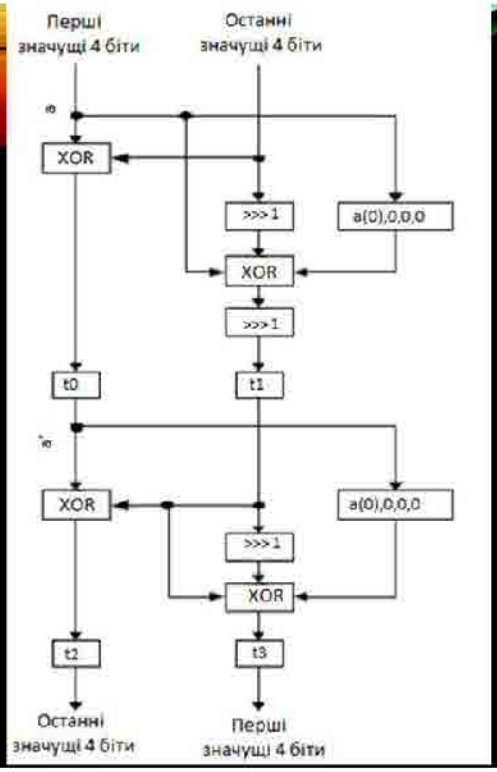


# Схема роботи алгоритму Twofish



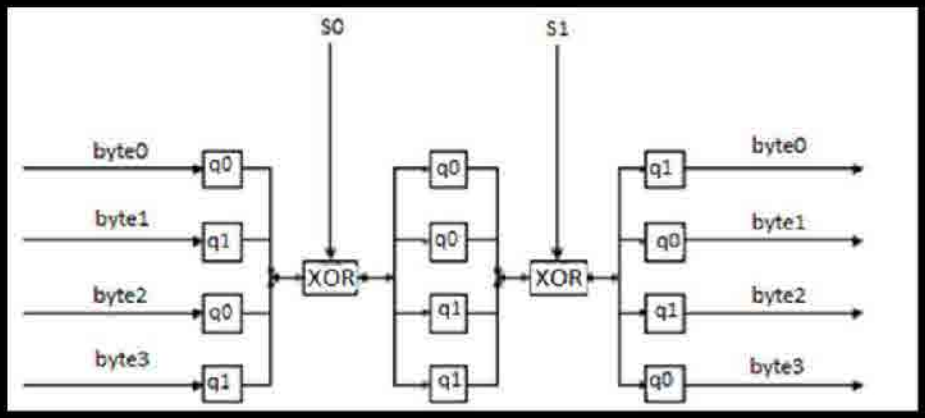

# Структурна схема модуля генерування підключів для алгоритму Twofish



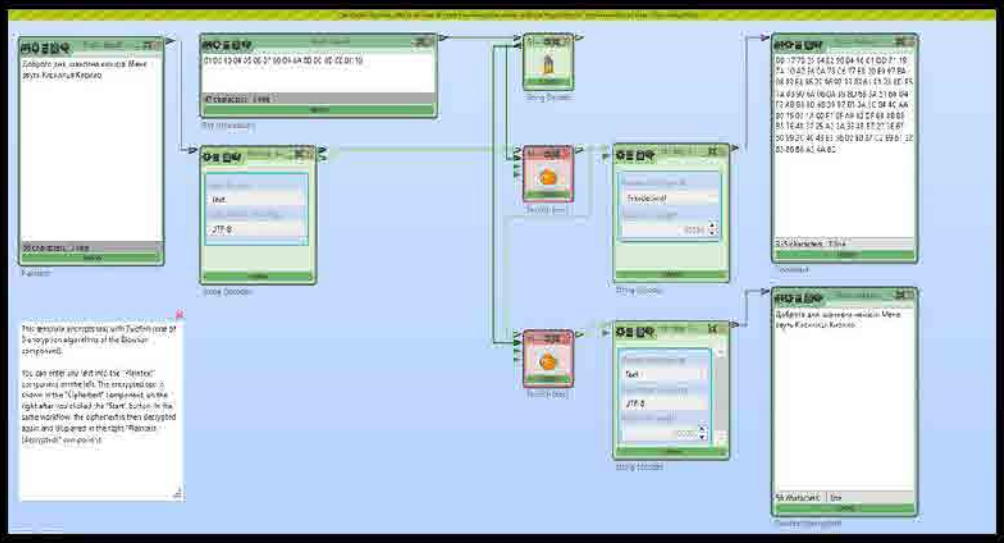



## Структурна схема q-перестановок у алгоритмі Twofish

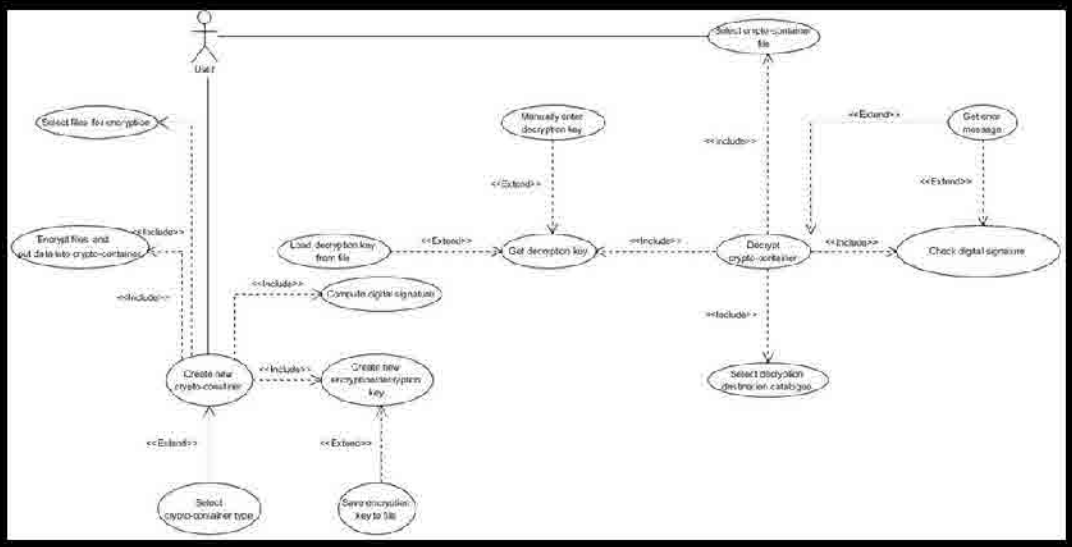
## Структурна схема S-боксів для алгоритму Twofish



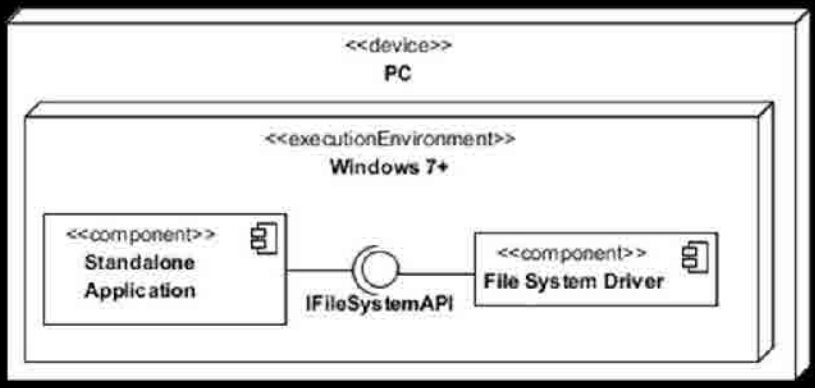
# Результат моделювання за алгоритмом Twofish



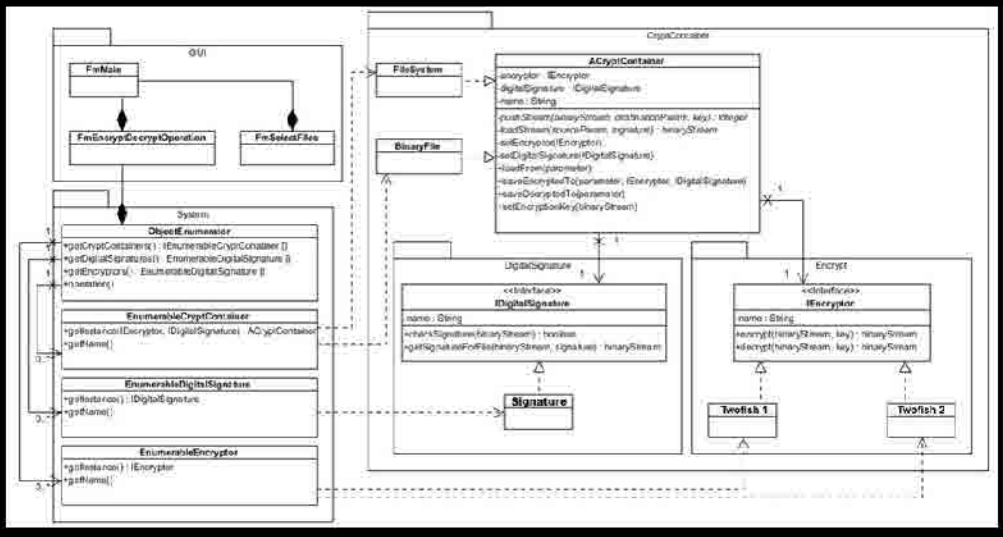
# UML-діаграма варіантів використання



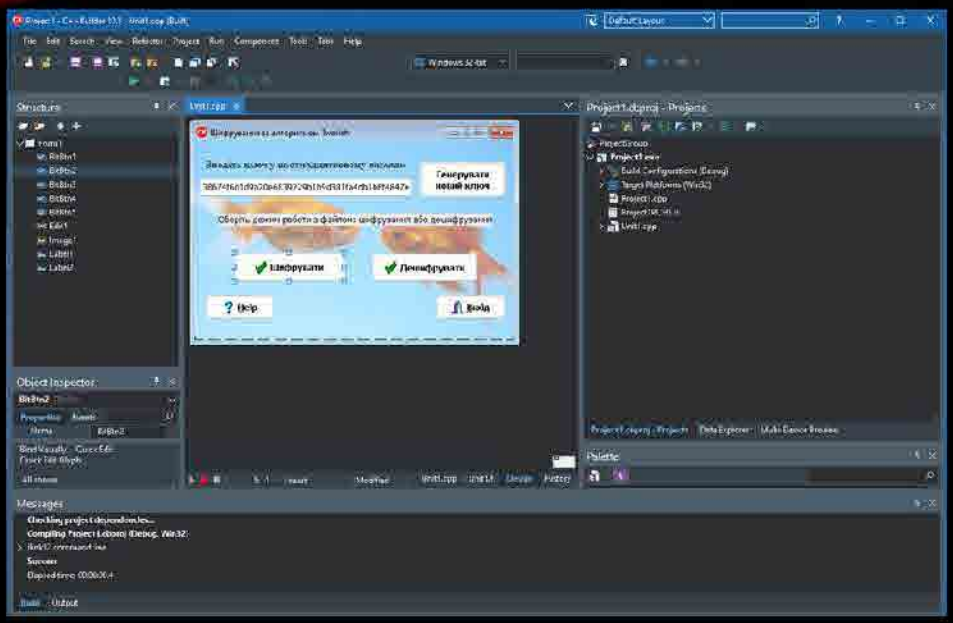
# Deployment UML-діаграма для програмного застосунку

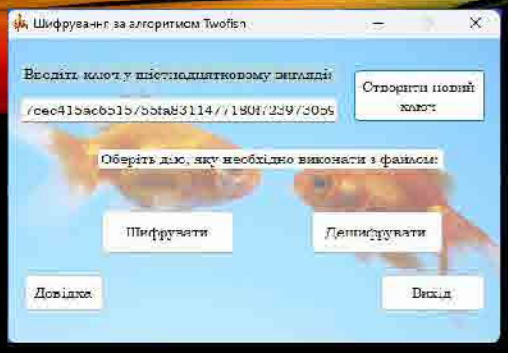



# Концептуальна діаграма класів для програмного застосунку

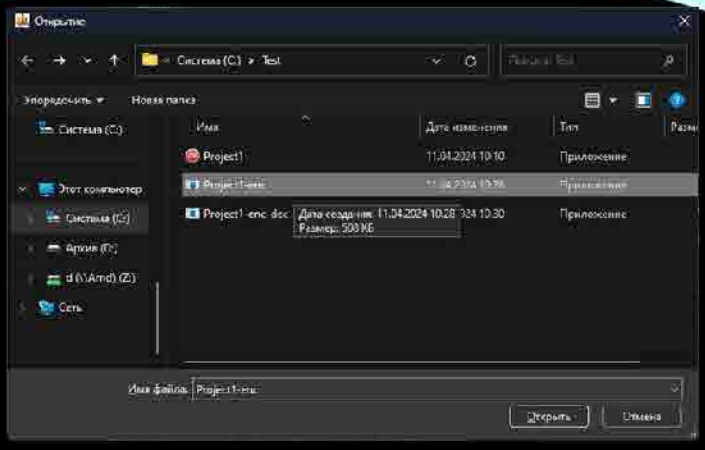


# Етап розробки візуального інтерфейсу застосунку

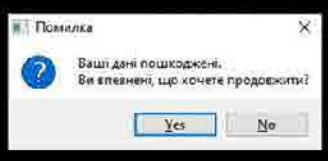


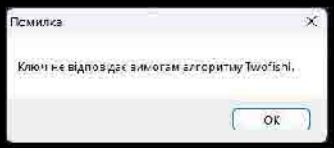



Головне вікно застосунку для захисту користувацьких файлів




Вибір зашифрованого файлу для подальшого дешифрування



ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

# ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

*Кисилиці Кирила Олексійовича*

(прізвище, ім'я та по батькові)

Спеціальність: *123 «Комп'ютерна інженерія»*

Освітня програма: *«Безпека комп'ютерних систем і мереж»*

Тема дипломного проекту: *Програмна реалізація алгоритму шифрування Twofish для захисту користувацьких файлів*

## ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка до дипломного проекту містить 84 сторінки. У пояснювальній записці описана програмна реалізація алгоритму шифрування Twofish для захисту користувацьких файлів. Графічна частина складається з 19 слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів добра.*

б) самостійність роботи над проектом: *Протягом виконання дипломного проекту здобувач освіти Кисилиця Кирило поступово та послідовно виконував всі етапи, проявив ініціативу в створенні загальної концепції та реалізації роботи. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Кисилиця Кирило під час роботи над дипломним проектом вивчив достатньо багато літературних та інтернет-джерел за даною тематикою.*

*Вважаю, що теоретична підготовка дипломника достатня і він готовий до захисту проекту.*

г) вміння розв'язувати виробничі та конструкторські питання *Під час виконання дипломного проекту здобувач освіти Кисилиця Кирило показав вміння організовано працювати над поставленим завданням, застосовувати знання у галузі безпеки комп'ютерних мереж, програмування, використовуючи сучасні комп'ютерні програмні засоби розробки, такі як Embarcadero RAD Studio, CrypTool2.*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові керівника дипломного проекту *Кривченко Анастасія Анатоліївна*

Місце роботи і посада керівника дипломного проекту *ВСП «Одеський технічний фаховий коледж ОНТУ», викладач спецдисциплін циклової комісії комп'ютерних технологій та програмної інженерії, голова методичної комісії комп'ютерної інженерії Одеської області*

Підпис

«10» 06 2024 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

# РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

*Кисилиці Кирила Олексійовича*

(прізвище, ім`я та по батькові)

Спеціальність *123 «Комп'ютерна інженерія»*

Освітня програма *«Безпека комп'ютерних систем і мереж»*

Керівник дипломного проекту (роботи) *Кривченко Анастасія Анатоліївна*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Програмна реалізація алгоритму шифрування Twofish для захисту користувацьких файлів*

Обсяг розрахунково-пояснювальної записки *84* сторінок

Обсяг графічної (презентаційної) частини *19* аркушів (слайдів)

## ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

*Представлений на рецензію дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений проблемі захисту користувацьких файлів та складається з пояснювальної записки, додатку з програмним кодом та мультимедійної презентації, що містить приклади роботи програми.*

б) характеристика виконання кожного розділу дипломного проекту

*Пояснювальна записка складається з основного розділу (аналізу предметної області, проектування застосунку, реалізації застосунку, тестування застосунку), економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано. Розділ охорони праці містить загальну інформацію та вимоги до техніки безпеки оператора КТ. Економічний розділ проекту містить розрахунок витрат на НДР та реалізацію проекту.*

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

*Графічна частина складається з 19 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять ілюстративні схеми, скріншоти роботи програмного застосунку, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання графічної частини проекту та пояснювальної записки відмінна, розробку виконано у повному обсязі.*

UNICHECK
by Turnitin

Ім'я користувача:
**Катерина Григоріївна Краснокутська**

ID перевірки:
**1016235225**

Дата перевірки:
**07.05.2024 20:14:46 EEST**

Тип перевірки:
**Doc vs Internet + Library**

Дата звіту:
**07.05.2024 21:57:49 EEST**

ID користувача:
**100011688**

Назва документа: **4КБ-01_Кирило_Кисилиця**

Кількість сторінок: **65** Кількість слів: **12719** Кількість символів: **90469** Розмір файлу: **2.56 MB** ID файлу: **1016016400**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

## 14.4%
## Схожість

Найбільша схожість: **5.11%** з Інтернет-джерелом (https://card-file.ontu.edu.ua/server/api/core/bitstreams/6c95086b-bff...

14.4% Джерела з Інтернету | 514 ..... Сторінка 67

**Не знайдено джерел з Бібліотеки**

## 0% Цитат

**Вилучення цитат вимкнене**

**Вилучення списку бібліографічних посилань вимкнене**

## 0%
## Вилучень

**Немає вилучених джерел**

## Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | 51

Підозріле форматування | 10 сторінок

г) перелік позитивних якостей дипломного проекту *Обраний алгоритм шифрування є актуальним та надійним. Виконується перевірка на цілісність даних. Передбачено використання надійного генератору випадкових чисел для використання ключу. Реалізовано візуальний інтерфейс із зручним та простим керуванням процесом шифрування / дешифрування.*

д) основні недоліки дипломного проекту

1. *В програмному проекті не передбачено відображення повідомлення про закінчення процесу шифрування/дешифрування*
2. *Деякі ілюстрації у тексті пояснювальної записки мають невисоку якість друку.*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові рецензента *к.т.н. Кірєєв Ігор Анатолійович*

Місце роботи і посада рецензента *Державний університет інтелектуальних технологій і зв'язку, доцент каф. інформаційної безпеки та передачі даних*

Підпис:

ПІДПИС ПОСВІДЧУЮ
НАЧАЛЬНИК ВІДДІЛУ
КАДРІВ ДУІТЗ

«14» 06 2024 р.

**ДОЗВІЛ**
**НА РОЗМІЩЕННЯ**
**ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ**
**(ДИПЛОМНОГО ПРОЕКТУ)**
**В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»**

Ми, що нижче підписалися,

***Кисилиця Кирило Олексійович,***
здобувач освіти гр. 4КБ-01, та

***Кривченко Анастасія Анатоліївна***,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

***«Програмна реалізація алгоритму шифрування Twofish для захисту користувацьких файлів» (автор роботи – Кисилиця К.О., керівник роботи – Кривченко А.А.)***

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2024 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Кисилиця К.О. /

Керівник _____ / Кривченко А.А. /

«10» червня 2024 р.