Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістра
(назва освітнього ступеня)

на тему: Метод підвищення криптостійкості симетричних алгоритмів шифрування

Виконав(ла): студент(ка) 6 курсу, групи СІм-61
спеціальності 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

(підпис) Семеген Б. В. (прізвище та ініціали)

Керівник (підпис) Лупенко С. А. (прізвище та ініціали)

Нормоконтроль (підпис) Луцик Н. С. (прізвище та ініціали)

Завідувач кафедри (підпис) Осухівська Г.М. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2021

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Осухівська Г.М.
(підпис) (прізвище та ініціали)
« » 20___ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістра
(назва освітнього ступеня)

за спеціальністю 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

студенту Семегену Богдану Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Метод підвищення криптостійкості симетричних алгоритмів шифрування

Керівник роботи Лупенко Сергій Анатолійович, д.т.н., професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*28*» *жовтня* 2021 року № *4-7/976*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Алгоритми шифрування даних, генерація перестановок, високорівнева мова програмування PureBasic та низькорівнева мова програмування Асемблер

4. Зміст роботи (перелік питань, які потрібно розробити)
1. Аналітична частина
2. Математичний опис симетричних алгоритмів шифрування даних та методу підвищення їх криптостійкості
3. Програмна реалізація системи шифрування
4. Охорона праці та безпека в надзвичайних ситуаціях

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Актуальність
2. Мета, об'єкт, предмет, методи дослідження, наукова новизна і практичне значення
3. Структурна схема шифрування даних
4. Структурна схема дешифрування зашифрованих даних
5. Приклад генерації перестановки
6. Файл з відкритим текстом
7. Файл з перестановленими фрагментами даних
8. Файл із зашифрованими даними
9. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М., зав. каф. КС		
Безпека в надзвичайних	Стадник І.Я., проф. каф. ОХ		
ситуаціях			

7. Дата видачі завдання 28 жовтня 2021 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	28.10.2021 р.	Виконано
2	Аналіз завдання	01.11.2021 р.	Виконано
3	Виконання розділу 1	10.11.2021 р.	Виконано
4	Виконання розділу 2	14.11.2021 р.	Виконано
5	Виконання розділу 3	20.11.2021 р.	Виконано
6	Виконання розділу 4	24.11.2021 р.	Виконано
7	Оформлення пояснювальної записки	02.12.2021 р.	Виконано
8	Оформлення графічного та презентаційного матеріалу	09.12.2021 р.	Виконано
9	Перевірка роботи на антиплагіат	14.12.2021 р.	Виконано
10	Попередній захист	15.12.2021 р.	Виконано
11	Захист		

Студент _____ Семеген Б. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Лупенко С. А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Метод підвищення криптостійкості симетричних алгоритмів шифрування // Кваліфікаційна робота магістра// Семеген Богдан Васильович // ТНТУ, комп'ютерна інженерія, група СІм-61 // Тернопіль, 2021 // С. – 86, рис. – 35, табл. – 3, аркушів А1 – 9, додат. – 2, бібліогр. – 28.

Ключові слова: ШИФРУВАННЯ ДАНИХ, КРИПТОСТІЙКІСТЬ, СИМЕТРИЧНІ АЛГОРИТМИ, ПЕРЕСТАНОВКИ, ЗАХИСТ ДАНИХ, МЕТОД.

Мета цієї роботи полягає у розробці методу для підвищення криптостійксті симетричних алгоритмів шифрування та програмній його реалізації.

У кваліфікаційній роботі проаналізовано симетричні алгоритми шифрування, що дало змогу визначити слабкі сторони їхнього застосування. Описано основні методи криптоаналізу, які можуть зробити шифри уразливими і становити загрозу розкриття відкритого тексту. Також було розглянуто способи генерації перестановок та створено новий алгоритм генерації унікальної перестановки.

В роботі було розроблено метод підвищення криптостійкості на основі генерації унікальних перестановок, для кожного блоку даних, за допомогою згенерованого псевдовипадкового числа.

Створено програмне забезпечення в якому реалізовано даний метод.

ANNOTATION

Method of increasing symmetric encryption algorithms' cryptosecurity // Qualification work of the master // TNTU, Computer Engineering // Semehen Bohdan // group SIm-61

// Ternopil, 2021 // p. – 86, fig. – 35, tab. – 3, Sheets A1 – 9, Add – 2, Ref. – 28.

Keywords: DATA ENCRYPTION, CRYPTOSECURITY, SYMMETRIC ALGORITHMS, PERMUTATIONS, DATA PROTECTION, METHOD.

The aim of the work is to develop an algorithm to increase the cryptographic strength of symmetric encryption algorithms and its implementation

Symmetric encryption algorithms were analyzed in the thesis, which allowed to identify the weaknesses of their application. The main methods of cryptanalysis that can make ciphers vulnerable and threaten the disclosure of plaintext are described.

In the thesis a method was developed to increase cryptosecurity based on the generation of unique permutations for each block of data using the generated pseudo-random number.

The software in which this method is implemented was created.

ЗМІСТ

ВСТУП

Актуальність роботи. Оскільки існує велика кількість методів криптоаналізу, що дають змогу зробити вразливими багато алгоритмів шифрування, симетричних зокрема, то питання підвищення криптостійкості симетричних алгоритмів шифрування буде актуальним для галузі криптографії, адже в сучасному світі цифрових технологій і великого потоку інформації є важливим захист даних, які передаються через різні канали зв’язку.

Також для забезпечення швидкого передавання даних і відповідно зв’язку, майже всі дані передаються в різних системах чи пристроях за допомогою симетричного шифрування, які є швидшими за асиметричні, проте можуть бути уразливими. Є різні способи криптоаналізу. Три із чотирьох основних базуються на способі атаки відкритим текстом. І в шифрованих каналах передачі даних, де є можливість атаки адаптивно підібраним відкритим текстом, існує особлива небезпека для конфіденційності даних, адже цей метод дає ширші можливості стороні, яка проводить атаку на канал зв’язку із симетричним шифруванням. Цим і пояснюється актуальність теми роботи, враховуючи також те, що розробляються пристрої, у яких для зменшення енергоспоживання застосовують недостатньо стійкі симетричні шифри. Можливе застосування цих пристроїв у сферах, де необхідним є забезпечення надійності передавання даних, вказує на потребу пошуку просто способу підвищити їхню криптостійкість, значно не збільшуючи при цьому споживання енергії.

Метою дослідження є розробка методу підвищення криптостійкості симетричних алгоритмів шифрування даних та програмної реалізації.

Для досягнення цієї мети потрібно:

- опрацювати матеріали по симетричним алгоритмам шифрування та визначити їхні слабкі сторони;
- проаналізувати методи атак на алгоритми шифрування;

- розробити метод підвищення криптостійкості для симетричних алгоритмів;
- створити програмну реалізацію розробленого алгоритму.

Об’єкт дослідження: симетричні алгоритми шифрування даних.

Предмет дослідження: метод підвищення криптостійкості симетричних алгоритмів шифрування даних.

Методи дослідження: аналіз симетричних алгоритмів та їх вразливостей; пошук способів підвищення криптостійкості вразливих алгоритмів шифрування; реалізація програмного забезпечення на основі способу.

Наукова новизна отриманих результатів вперше розроблено алгоритм, що підвищує криптостійкість методом перестановок підблоків, залежно від чисел, що генеруються окремо для кожного блоку даних.

Практичне значення одержаних результатів: розроблений метод та його програмна реалізація дають змогу підвищити криптостійкість симетричних алгоритмів з невисокими вимогами до обчислювальних потужностей і використовувати алгоритм для захисту передачі даних у малопотужних пристроях.

Публікації. Результати дослідження апробовано на IX Науково-технічній конференції «Інформаційні моделі, системи та технології» (8–9 грудня 2021 року) Тернопільського національного технічного університету імені Івана Пулюя у вигляді тез конференцій [27,28].

Структура роботи: робота складається з пояснювальної записки та графічної частини. Пояснювальна записка складається із вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Обсяг роботи: пояснювальна записка – 86, арк. формату А4, графічна частина – 9 аркушів формату А1.

РОЗДІЛ 1
АНАЛІТИЧНА ЧАСТИНА

1.1 Аналіз існуючих алгоритмів шифрування та їх поділ за типами

Симетричні алгоритми шифрування – це такі алгоритми в яких при передаванні даних для їхнього шифрування використовується один і той же криптографічний ключ. Ключ алгоритму не повинен розкриватися жодною із сторін, що обмінюються даними. Тобто тримається у секреті обома сторонами. Таке шифрування було єдиним існуючим способом для таємної передачі інформації аж до винаходу асиметричних алгоритмів шифрування даних. Алгоритми, що шифрують і дешифрують дані використовуються як в комп'ютерній техніці, так і в системах приховування конфіденційної чи комерційної інформації, від неправомірного отримання доступу та некоректного використання.

Базовою є ідея, що сторона, яка приймає повідомлення, має відповідний алгоритм шифрування, а також ключ для зворотнього перетворення, без якого передана інформація не матиме змісту. Симетричні алгоритми здійснюють перетворення над невеликим блоком (1 або 32-128 біт; можливе і понад 128) даних і як саме це відбувається залежить від ключа. Як результат – прочитати повідомлення можна лише знаючи цей ключ [9, 10].

Класифікація симетричних криптоалгоритмів.

Є дуже багато криптографічних алгоритмів, зокрема, і симетричних. Усі вони мають де призначення – захист інформації. Симетричні алгоритми, які є криптоалгоритмами з ключем поділяються на: потокові шифри та блочні шифри.

Потокові шифри – обробка інформації відбувається побітно. Шифрування і дешифрування при такій системі може перериватися в довільний момент часу, щойно виявляється, що потік що передається перервався, і також відновлюється при виявленні зв’язку та факту продовження передачі. Потокові шифри в свою

чергу можна поділити на: синхронні та самосинхронізуючі. Програмні чи апаратні реалізації потокових шифрів називаються скремблерами [1, 2, 3].

Блочні шифри – здійснюють перетворення блоку вхідної інформації фіксованої довжини і отримують в результаті блок того ж обсягу (спосіб використовується при пакетній передачі інформації і для кодування).

В свою чергу блокові шифри також можуть поділяти на ті, які базуються на:

- мережа Фейштеля;
- мережа замін-перестановок або SP-мережа.

Мережа Фейштеля. Даний алгоритм використовує метод оборотних перетворень тексту, при цьому значення, обчислені від однієї з частин тексту, певним чином накладаються на інші частини. Доволі часто структура мережі виконується так, що для шифрування і дешифрування використовується один і той же алгоритм – різниця полягає лише в порядку використання інформації ключа [11].

Мережа замін-перестановок або SP-мережа – спосіб шифрування при якому на вхід подається блок відкритого тексту, а також ключ і опісля йде застосування певної кількості раундів S- та P-скринь [8].

Потокові шифри.

Існує велика кількість досліджень в напрямку розробки потокових шифрів, оскільки симетричні алгоритми почали розвиватися значно швидше асиметричних. Для поточних шифрів характерним є побітова обробка інформації (рис.1.1). Обробка інформації такого типу може бути представлена у вигляді автомату, що на кожному такті:

- генерує за певним методом біт шифруючої послідовності;
- деяким зворотнім перетворенням накладає на один біт відкритого потоку інформації шифруючий біт і в результаті отримує зашифрований біт.

Причиною достатності одного біта на біт вихідного тексту при шифруванні послідовності є наявність оборотних операцій арифметики за модулем 2, а саме: додавання за модулем два та заперечення.

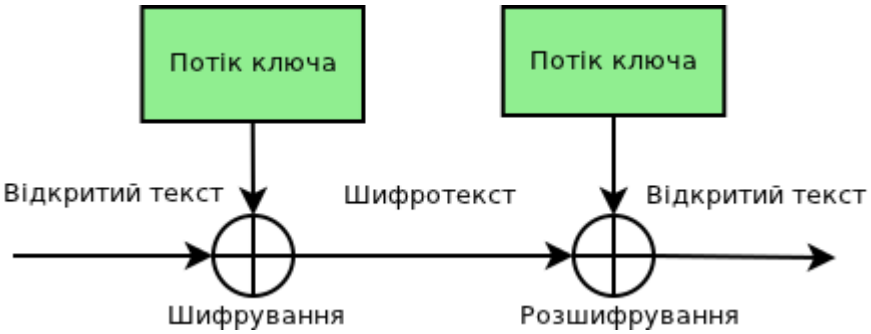

Рис. 1.1. Загальна схема передачі інформації потоковими шифрами

Обортною операцією є та, в якій при відомому результаті та всіх операндах, крім одного, можна відновити цей невідомий операнд. Іншими словами: при таких перетвореннях не втрачається частина інформації. Отже при шифруванні можливим є застосування лише оборотніх операцій і фактичного усі алгоритми такого шифрування організовані таким чином, що на один біт первісного тексту накладаються шифруючі біти (скільки б їх не було створено) шляхом комбінацій з зазначених вище двох операцій – XOR та заперечення. Заперечення можна вставити в середину операції XOR, тобто для будь яких значень a і b є вірним: NOT (a XOR b) = a XOR (NOT b) XOR b . Звідси висновок, що композицію з вихідного та шифруючих бітів, можна розділити та представити у вигляді: p XOR F (g1, g2, g3), де p – первісний біт, g – шифруючий біт, F – певна функція, що містить XOR та заперечення. В підсумку вся формула матиме вигляд: c = p XOR g

Загальна схема шифрування поточним шифром наведена на рис.1.2.

Рис. 1.2. Узагальнена схема шифрування поточними шифрами (із прикладом)

За даною схемою працюють усі поточні шифри. Символом гама (γ) прийнято позначати біт шифрування чи цілий набір таких біт, що з'являються на кожному кроці автомату. Через це такі алгоритми іменують деколи гама шифрами. За умови апаратної реалізації ці шифри значно швидші за блочні, а при програмній зазвичай не мають переваг у швидкості.

Три основних складових над якими обчислюється функція, що породжує гаму:

- ключ;
- номер поточного кроку шифрування;
- найближчі від поточної позиції біти первісного і (або) зашифрованого тексту.

Блокові алгоритми.

Блокові шифри – це алгоритми шифрування, які виконують перетворення інформації блоками фіксованої довжини та на виході отримують аналогічної довжини блок, недоступних для сторонніх осіб даних. Можна сказати, що в багатьох блокових шифрах застосовується деякий набір бієктивних математичних перетворень, тобто таких, які є оборотними.

Отже коротко робота блокових криптосистем описується функціями $Z = EnCrypt(X, Key)$ та $X = DeCrypt(Z, Key)$ – шифрування та дешифрування, відповідно. Для багатьох криптосистем блокові шифри є основою, оскільки дозволяють швидко та надійно передавати важливу, конфіденційну інформацію.

Salsa20 – це симетрична система потокового шифрування, яка має програмне застосування, що володіє високою пропускною здатністю. Розробив і представив на конкурсі даний алгоритм Деніел Бернштейн. Конкурс був організований задля створення європейських стандартів шифрування і захисту даних, які передавалися поштовими системами.

Шифрування salsa20 базується на використанні таких операцій:

- додавання 32-бітних чисел;
- побітове сумування по модулю 2;

- зсув бітів.

Базові перетворення виконувані алгоритмом схожі до AES, а сам алгоритм використовує хеш-функцію з 20-ма циклами. Шифр має високу швидкодію через те, що існує можливість розпаралелювання процесу необхідних обчислень, адже перетворення кожного стовпця і кожного рядка відбуваються окремо і, відповідно, є не залежними одні від одних. В алгоритмі фактично відсутні накладні обчислення перед початком циклів шифрування. Складнощів для змін ключа також немає. Багатьом системам шифрування необхідні попередні обчислення, які зберігаються в кеші першого рівня у процесорі і, оскільки це залежить від ключа, то у випадку зміни всі обчислення виконувати заново. У випадку Salsa20 потрібно всього лише завантажувати ключ у пам'ять. На основі Salsa20 розроблено простіші варіанти шифру – Salsa20/8 та Salsa20/12, в яких, відповідно, кількість раундів шифрування 8 та 12, а в той час у попередньому – 20. Хоча Salsa20/8 швидкий та має кращу швидкодію, ніж деякі інші шифрувальні системи (блоковий AES чи потоковий RC4), але на нього виявлена криптоатака, проте вона має велику часову складність, а от Salsa20/20 розроблявся аби мати значну стійкість.

Тим самим автором також запропонована модифікація шифру Salsa20, де довжину nonce було збільшено із 64 біт до 192, та названо її XSalsa20. Збільшення величини nonce дає можливість отримати криптографічно стійкий генератор випадкових чисел для його генерації, оскільки при надійному шифруванні 64-бітним потрібно використовувати лічильник для уникнення колізій.

Було проведено криптоаналіз даного алгоритму Паулем Кровлі та в 2005 році заявлено про атаку (що базувалася на спрощеному диференціальному крипто аналізі) на Salsa20/5 з складністю по часу 2^{165}. Роком пізніше стало відомо про атаку на Salsa20/6 що мала складність 2^{117}, а 2008 року п’ятеро спеціалістів (Омасон, Фішер, Казаї, Меєр і Решбергер) повідомили про атаку на Salsa20/8 із часовою складністю 2^{251}. Цифри після слешу вказують на кількість раундів

шифрування [6]. Але наразі не має інформації про атаки на повну Salsa20/20 і Salsa20/12.

DES – це блоковий симетричний алгоритм шифрування даних, який при створені мав малу довжину ключа. Зараз даний алгоритм через причину попередньо згадану, а також через малу довжину блоку (64 біти) вважається ненадійним, хоча критиці піддавався і від часу його створення у 1976 році, зокрема через таємність деяких елементів його роботи та структури. Це в свою чергу викликало побоювання аналітиків щодо можливості контролю зі сторони Національного Агентства Безпеки Сполучених Штатів Америки, адже довжина ключа 56 біт є недостатньою для забезпечення конфіденційності і держава могла володіти необхідними потужностями для аналізу даних. Саме DES як і аналіз його стійкості став підґрунтям для уявлень про блокові алгоритми шифрування, а також криптоаналіз і уважне вивчення предмету. Йому на зміну прийшов AES, який з 2002 року є стандартом у США і фактично витіснив його в застосуванні. Відверто говорячи є різниця між самим стандартом DES і алгоритмом DEA, який організований відповідно. Розробка DESу почалася у 1970-х через запит Національного Бюро Стандартів (нині це NIST), оскільки була потреба зі сторони уряду в таємній, захищеній передачі важливих даних органами влади. Заявка була подана компанією IBM на другий конкурс 1974 році в серпні (27-го), тобто більш, ніж через рік часу з часу оголошення першого конкурсу, а такж було запропоновано алгоритм, що базувався на шифрі, розробленому Хорстом Фейстелем [4, 5].

AES або Rijndael як черговий алгоритм (і відповідно стандарт) симетричного шифрування прийшов на зміну DES на початку 2000-х, адже недоліки останнього ставали все більш очевидними, а саме: потреба в апаратній реалізації для забезпечення відповідної швидкодії, із зростанням швидкодії процесорів з'являлися можливість його взлому методом грубої із сили, його модифікації, що мали підвищувати криптостійкість були повільними при

застосуванні [7]. Осінню 1997-го Національний інститут стандартів США оголосив конкурс аби обрати наступника DES і поставив наступні вимоги:

- шифр має бути блоковим;
- довжина одного блоку 128 біт;
- ключі можуть бути довжиною 128, 192, 256 біт.

Алгоритм відповідає вимогам, що були поставленні НІСТом.

Blowfish – в криптографії алгоритм, що реалізує блочне симетричне шифрування, авторства Брюса Шнаєра. Розроблений на доволі швидких та простих операція таких як: виключне або, додавання та підстановка. По суті базується на мережі Фейстеля. Свого часу став популярним через те, що не був запатентованим на відміну від решти відносно надійних алгоритмів, а ті, що були вільними для використанням, являли собою переважно ненадійні шифри [13]. Відповідно являв собою альтернативу застарілому DES. При розробці автор орієнтувався на такі критерії:

- Швидкість (всього за 26 тактів відбувається шифрування на 32-бітних процесорах);
- Простота (так як використовуються простих операцій, що зменшують імовірність помилки реалізації алгоритму);
- Компактність;
- Можливість налаштування стійкості.

1.2 Типи атак на алгоритми шифрування і їх перелік

Задача в роботі криптоаналітика полягає у визначенні ключа K, і послідовності даних P або і першого, і другого. Але він також може працювати із деякою ймовірнісною інформацією про P, наприклад знаючи, що дані є: цифровим фотоматеріалом, українським текстом, оцифрованим звуком або ще чимось. Ціль криптографії — зберегти відкритий текст або ключ (чи і перше, і друге) від несанкціонованого доступу особами, які прагнуть перехопити

повідомлення. При цьому робиться припущення про те, що лінії зв'язку між відправником і отримувачем повністю контролюються зловмисниками. Сенсом криптографії є вирішення питань конфіденційності передаваної інформації. Вона вирішує питання таємності, перевірки справжності (чи авторства), цілісності інформації та нечесності зі сторони людей. Спробу чи процес криптоаналізу називають розкриттям. Брюсом Шнаєром було виділено чотири основних криптоатаки:

Атака із використанням шифротексту – спосіб атаки, коли у криптоаналітика є тільки шифротексти декількох повідомлень, що зашифровані одним і тим же алгоритмом шифрування. Задача криптоаналітика в цьому випадку полягає в знаходженні відкритих текстів у якомога більшій кількості повідомлень або отриманні ключа чи ключів, які використовувалися для шифрування повідомлень, задля подальшого дешифрування інших повідомлень, що були зашифровані тими же ж ключами. Зашифровані тексти отримуються при звичайному перехоплені повідомлень, коли зловмисник слухає їх через відкриті канали зв'язку. Отже маємо: $C_1 = E_k(P_1)$, $C_2 = E_k(P_2)$,…, $C_i = E_k(P_i)$, а потрібно отримати або якийсь з відкритих текстів P_1, P_2, … , P_i або ключ K, або алгоритм як отримувати P_{i+1} з $C_{i+1} = E_k(P_{i+1})$

Атака із використанням відкритих текстів – вид атаки при якій у крипто аналітика є доступ не тільки до шифротекстів, але і до частини або цілком відкритого повідомлення. Завданням є отримання ключа, який шифрує повідомлення. Надалі стає можливим читати інші повідомлення. В загальному схема методу є такою, що на подані P_1, $C_1 = E_k(P_1)$, P_2, $C_2 = E_k(P_2)$,…, P_i, $C_i = E_k(P_i)$ потрібно отримати або ключ K, або алгоритм отримання P_{i+1} з $C_{i+1} = E_k(P_{i+1})$

Атака із використанням підібраного відкритого тексту – це такий вид атаки, коли у криптоаналітика є не тільки доступ до шифротектів та відкритих текстів, але і можливість обирати відкритий текст для шифрування. Це дає більше можливостей, аніж атака з використанням відкритого тексту. Оскільки

криптоаналітик може вибирати блоки відкритого тексту для шифрування, що може надати більше інформації про ключ. Його завдання в цьому випадку полягає у знаходженні ключа для дешифрування повідомлень або алгоритму, який би давав можливість дешифрувати повідомлення, що були зашифровані тим же ж ключем. Тобто маємо: $P_1,\ C_1 = E_k(P_1),\ P_2,\ C_2 = E_k(P_2),\ldots,\ P_i,\ C_i = E_k(P_i),$ де криптоаналітик може вибирати відкриті тексти $P_1,\ P_2,\ \ldots\ ,\ P_i$ і завданням є отримати або ключ K, або алгоритм отримання P_{i+1} з $C_{i+1} = E_k(P_{i+1})$

Атака з адаптивно підібраним відкритим текстом – метод криптоаналізу, який є частковим випадком атаки з підібраним шифротекстом, та при якому, криптоаналітик може обирати наступні відкриті тексти під час атаки, тобто на базі попередньо отриманих результатів шифрування частини відкритого тексту. При атаці на основі відкритих текстів спеціаліст володіє парами текст-шифротекст, а більш зручною за цю є атака з підібраним шифротекстом, коли відкриті тексти (або один великий блоку) добираються ще до початку шифрування і аналізу. Тобто Проте адаптивний добір модернізує цей метод і надає йому перевагу перед попереднім та іншими, оскільки фахівець може обирати один блок відкритих даних, а на основі опрацювань потім інші, які потрібні. Даний вид атаки використовувався спеціалістами в галузі криптології ще в першій половині 20 століття. Для прикладу, британські військові мінували певні зони Північного моря і, в результаті, маючи координати зон, знали частину відкритих даних, які пересилали німецькі військові.

Є ще також три додаткових методи крипто атак(варто згадати ще два додатки:

Атака на базі підібраного шифротексту – спосіб атаки при якому криптоаналітик має доступ чи володіє деякою системою (так званим чорним ящиком), яка дешифрує зашифровані повідомлення. Він має можливість обирати різні шифротексти для дешифрування, а також отримувати дешифровані відкриті тексти. Коротко описується так: $C_1,\ P_1 = D_k(C_1),\ C_2,\ P_2 = D_k(C_2),\ldots,\ C_i,\ D_i = D_k(C_i)$, а завданням є отримати ключ k. Такий тип атаки зазвичай застосовується

до алгоритмів з відкритим ключем, проте в деяких випадках є ефективним і проти симетричних криптосистем. Через подібність до атаки з використанням підібраного відкритого тексту їх деколи разом називають атакою з використанням підібраного тексту.

Атака з використанням вибраного ключа – тип атаки, коли криптоаналітик має деяку інформацію про певний зв’язок між ключами. Цей метод не означає простого перебору ключів, а мається на увазі, наприклад, визначення деякого математичного співвідношення. При застосуванні часто є непрактичним.

Бандитський криптоаналіз – отримання необхідних даних силовими чи нечесними методами (використовується людський фактор): шантаж, погрози, дача хабаря і інші. Зловмисники можуть обрати його у випадку стійких криптосистем [12].

Окремо виділяється інженерний криптоаналіз (атака), з допомогою якого розробляються атаки на програмні, апаратні чи програмно-апаратні реалізації шифрів. При такому підході вивчають: час за який здійснюються криптографічні перетворення, інформацію про споживання обсягів електроенергії, що дає деяку інформацію про шифр чи, наприклад, який саме алгоритм використовується, а також спосіб аналізу при впливі електричними і магнітними полями, йонізуючим опроміненням чи великими температурами.

1.3 Види аналізу криптостійкості алгоритмів шифрування

Диференціальний криптоаналіз – один з методів криптоаналізу, що застосовується до блокових шифрів, але може застосовним і до інших алгоритмів (потокових, геш-функцій). Диференціальний криптоаналіз базується на досліджені змін між певними значеннями, які шифруються на різних раундах шифрування. Тобто як зміни у вхідних значеннях впливають на дані на виході. Даний алгоритм запропонували Елі Біхам та Аді Шамір (обидва ізраїльські спеціалісти). Призначався для атак на DES та подібні йому системи, але найкраще

розкривав шифр FEAL. Також варто зазначити, що стійкість DES залежить від структури S-блоків та від кількості раундів і, відповідно, може бути підвищена за умови змін у них. При 19 і більше етапах шифрування час аналізу стає більшим, аніж метод грубої сили. Алгоритм можна вважати стійким до даного виду аналізу, якщо кількість відкритих текстів необхідних для нього перевищує теоретичну кількість відкритих текстів, які можливі при певній довжині блоку.

В основному цей метод є теоретичним, оскільки в реальному застосуванні є обмеження вимогами по часу і об'єму даних.

Лінійний криптоаналіз – метод для крипто аналізу алгоритмів, який застосовує лінійні наближення для розкриття роботи алгоритму. Створювався спочатку для розкриття алгоритмів DES і FIAL та був запропонований у 1993 році Міцуро Мацуі – японським криптоаналітиком. Проте пізніше даний метод використовувався і для розкриття інших алгоритмів та поряд з диференціальним криптоаналізом є одним з найпоширеніших. Також з його допомогою можна атакувати і потокові шифри. Для атаки на DES із 16 етапами шифрування достатньо 50 днів і процесора з тактовою частотою орієнтовно 60-70 МГц, а також кількість текстів 2^{43}. Якщо ключ належить до класу так званих слабких ключів, то цей криптоаналіз також ефективний і проти RC5. Також цим методом розкриваються NUSH і NOEKEON. В даному методі є два етапи криптоаналізу. Спочатку будуються співвідношення між зашифрованим текстом, відкритими даними і ключем. Далі йде застосування поданих співвідношень в купі з відомими парами (відкритий текст)-(зашифрований текст), щоб отримати біти ключа. Тобто суть алгоритму в тому, аби отримати наступні співвідношення:

$$P_{i_1} \oplus P_{i_2} \oplus \ldots \oplus P_{i_a} \oplus C_{j_1} \oplus C_{j_2} \oplus \ldots \oplus C_{j_b} = K_{k_1} \oplus K_{k_2} \oplus \ldots \oplus K_{k_c}, \quad (1.1)$$

де $P_n\ C_n\ K_n$ – n-ні біти відкритого тексту, зашифрованого та ключа.

Ймовірність правильності такого співвідношення наближено дорівнює 0,5 для деяких вибраних бітів множин P, C та K.

Інтерполяційний криптоаналіз – тип криптографічного аналізу в якому для представлення S-скринь використовується алгебраїчна функція. Це, для прикладу, може бути поліноміальна, квадратична чи раціональна функція над полем Галуа. Коефцієнти цієї функції можна визначити за допомогою звичайних методів інтерполяції Лагранжа, використовуючи відомі відкриті тексти як точки даних. Також атаку можна оптимізувати, спростивши рівняння обраними відкритими текстами. Спрощено являє собою многочлен від зашифрованого тексту і, за умови невеликого числа невідомих коефіцієнтів, а також маючи набори шифротексту-тексту, можна відновити поліном. Це дасть інформацію про шифрування без точних даних про ключ. Був запропонований в кінці 90-х років Ларсом Кнудсеном в співробітництві з Томасом Якобсеном.

Частотний криптоаналіз – це такий криптоаналітичний метод, що базується на визначені частоти появи знаків шифротексту. Тобто робиться припущення про наявність деякого статистичного розподілу певних символів чи їх послідовностей в незашифрованих даних, а також те, що така статистика відображається і у шифротексті. Простіше кажучи, цей метод опирається на частотний розподіл літер у текстах деякої мови, тобто те, що певна літера зустрічається з однаковою частотою з’являється у достатньо довгих, але різних текстах тієї ж мови. Даний вид криптоаналізу в основному використовується для навчання, оскільки орієнтовно із середини 20 століття значна частина алгоритмів розроблялася з метою зробити їх стійкими до нього[11, 15, 16, 17].

1.4 Висновки до розділу 1

У цьому розділі було:

- Описано типи симетричних алгоритмів шифрування даних;
- Описано найбільш популярні шифри та їхню криптостійкість до різних методів криптоаналізу;

- Розглянуто та описано найбільш популярні криптоатаки на алгоритми шифрування, а також додаткові способи атак спрямованих на розкриття ключа чи відкритого тексту;
- З'ясовано та наведено інформацію про основні види криптоаналізу.

РОЗДІЛ 2
МАТЕМАТИЧНИЙ ОПИС СИМЕТРИЧНИХ АЛГОРИТМІВ ШИФРУВАННЯ ДАНИХ ТА МЕТОДУ ПІДВИЩЕННЯ ЇХ КРИПТОСТІЙКОСТІ

2.1 Аналіз та математичний опис алгоритму шифрування TEA як прикладу нестійких шифрів

Варто окремо розглянути алгоритм TEA. Оскільки автори алгоритму TEA не патентували його, то він широко використовується

В даному алгоритмі відкритий текст перед подачею на шифрування розбиваються на блоки довжина кожного з них становить 64 біти. Оскільки дані можуть бути не кратними 64 бітам, а криптоалгоритм є блоковим, то, щоб доповнити блок і зробити дані кратними, значення решти байтів встановлюють як 0b00000001 (в двійковій системі). Ключ K, довжина якого 128 біт (16 байт)розділяють на чотири підключі довжиною 32 біти кожен і позначають їх як K_0 – перший, решта три – K_1, K_2, K_3. Потім опісля цієї підготовки кожен блок (64-бітний) опрацьовується в 64 раунди (32 цикли) за алгоритмом, який описується нижче (рис.2.1).

Якщо подається ліва (L_n) та права (R_n) частини на вхід кожного раунду n (тут $1 \leq n \leq 64$), то на виході кожного такого раунду будуть уже ліва (L_{n+1}) та права (R_n), яких обчислюють за наступними правилами:

- За умови, що n є непарними, тобто $\mathrm{n} = 2 * \mathrm{i} - 1$ для всіх $1 \leq i \leq 32$, тоді $R_{n+1} = L_n \boxplus (\{[R_n \ll 4] \boxplus K[0]\} \oplus \{R_n \boxplus i * \delta\} \oplus \{[R_n \gg 5] \boxplus K[1]\})$;
- За умови, що n є парними, тобто $\mathrm{n} = 2 * \mathrm{i}$ для всіх $1 \leq i \leq 32$, тоді $R_{n+1} = L_n \boxplus (\{[R_n \ll 4] \boxplus K[2]\} \oplus \{R_n \boxplus i * \delta\} \oplus \{[R_n \gg 5] \boxplus K[3]\})$;
- Ліва частина $L_{n+1} = R_n$.

Тут $\delta = \left(\sqrt{5} - 1\right) * 2^{31} = 2654435769 = 9E3779B9_h$ є сталою, яка виводиться із золотого перетину.

$X \boxplus Y$ – операція додавання деяких частин обчислення або чисел X та Y по модулю 2^{32}.

$X \oplus Y$ – побітова операція XOR (Виключне АБО) деяких частин обчислення або чисел X та Y.

$X \ll Y$ та $X \gg Y$ – побітовий зсув деякого числа X на певну кількість біт (позначено Y) вліво або вправо.

В кожному раунді відбувається множення константи на номер циклу i,а було це зроблено, щоб запобігти атакам, які базуються на симетрії раундів та є досить прості в реалізації. Перевага алгоритму є нескладна реалізація, але те, що в основній функції перетворень відсутні нелінійні операції, компенсовується великою кількістю раундів [14].

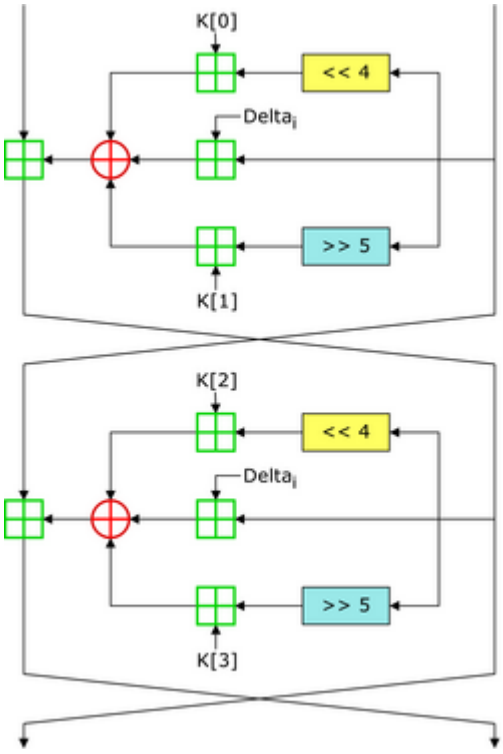

Рис. 2.1. Структурна схема алгоритму роботи TEA

Над розробкою алгоритму шифрування працював Девід Вілер разом із Роджером Нідгемом.

Алгоритм має таку ж надійність як і шифр IDEA, оскільки операції з тих же ортогональних алгебраїчних груп застосовуються і у ньому, що в теорії надає захищеність від методів на основі лінійного криптоаналізу.

Найбільш уразливим цей алгоритм є до атаки пов'язаними ключами, оскільки розклад ключів в ньому зроблено дуже по-простому, тобто його немає як такого. Тут використовуються в парних раундах підключі K_2 і K_3, а в непарних застосовуються K_0 і K_1. Також виявлено, що кожен із чотирьох ключів має ще три йому еквівалентних і, відповідно, ефективними є лише 126 біт довжини ключа, а не 128. Криптограф Брюс Шнаєр включно із ще двома авторами в 1997 році представив три атаки засновані на цьому методі. Із ймовірністю 2 деякі із них (ті що простіші) мають диференціальна характеристику з й найбільш простих серед даних способів виявляє. Серед найбільш простих даних способів для отримання диференціальної характеристики в якій ймовірність рівна одиниці, що означає знаходження усіх бітів ключа, потрібно 2^{34} відкритих текстів, тобто після 32 раундів алгоритму вона має ймовірність 2^{-32}. Інша атака, складніша в реалізації так як в ній поєднані диференціальний криптоаналіз і згадана атака з відкритими ключами, але вона має ймовірність диференціальної характеристик рівну 2^{-11} і 2^{23} вибраних відкритих текстів і час 2^{32}, що означає кількість бітових операцій становить 2^{32}. Цей шифр при 10 раундах виявляє хорошу стійкість і при диференціальному крипто аналізі потрібно $2^{52,5}$ вибраних відкритих тексті та часова складність становить 2^{84}. Проте при 17 раундах шифрування для атаки потрібно 1920 відкритих даних і часова складність тут наближено становить 2^{123}. Розроблені посиленні та модифіковані варіанти для даного шифру: XTEA, XXTEA, RTEA та Raiden. Варіант XTEA має довжину блоку 64 біти та 128-бітний ключ. При шифруванні тут застосовується 64 раунди мережі Фейштеля. Дана версія була опублікована у 1997 році авторами попереднього варіанту. Було усунено вразливість до атаки на пов'язаних ключах шляхом введення алгоритму розкладу ключів, проте зменшена стійкість до диференціального криптоаналізу. Є модифікації цього алгоритму, де є 32 раунди мережі Фейстеля (XTEA-1) замість

64, або ж розмір блоку збільшений у два рази до 128 біт (XTEA-2), а у XTEA-3 збільшений і розмір блоку (довжина 128 біт), і довжина ключа до 256 бітів. Через рік після оприлюднення XTEA був опублікований розширений варіант цього алгоритму та був названий XXTEA. В 2007 році з'явилася модифікація TEA, що отримала назву RTEA і кількість раундів мережі Фейштеля була збільшена до сорока восьми. В табл. 2.1 наведено порівняння алгоритмів на основі TEA по їх базовим характеристикам.

Таблиця 2.1

Порівняння алгоритмів сімейства TEA за їх основними характеристиками

Найменування алгоритму	Рік оприлюднення	Довжина блоку в бітах	Довжина ключа в бітах	Число раундів мережі Фейстеля
TEA	1994	64	128	64
XTEA	1997	64	128	64
XTEA-1	-	64	128	32
XTEA-2	-	128	128	64
XTEA-3	-	128	256	64
XXTEA	1998	64*M	128	52+12*M
RTEA	2007	64	128 чи 256	48 чи 64

Оскільки все сімейство алгоритмів TEA побудоване на основі мережі Фейштеля, то варто також її розглянути. В алгоритмах на основі даних мереж вхідну інформацію розбивають на блоки однакової довжини, яку можна виразити

часто як двійку в деякому степені. Основні операції, що здійснюються над блоками є такими:

- обраний блок над яким здійснюється операція ділиться на два підблоки однакового розміру, один з яких лівий (L_0), а інши – правий (R_0);
- функцією F змінюється правий підблок R_0, що робиться із застосуванням ключа K_0: $x = F(R_0, K_0)$;
- до лівого блоку L_0 додається по модулю результат тієї функції;
- цей же результат в наступному раунді буде опрацьований як правий підблок R_1 ($R_1 = x$);
- а правий підблок R_0 в даному раунді буде використаний в наступному як лівий L_1 ($L_1 = R_0$);
- за деяким математичним правилом (залежного від конкретного алгоритму) вираховується ключ, який буде застосований в наступному раунді.

На рис. 2.2 зображено перераховані вище операції, а також дешифрування.

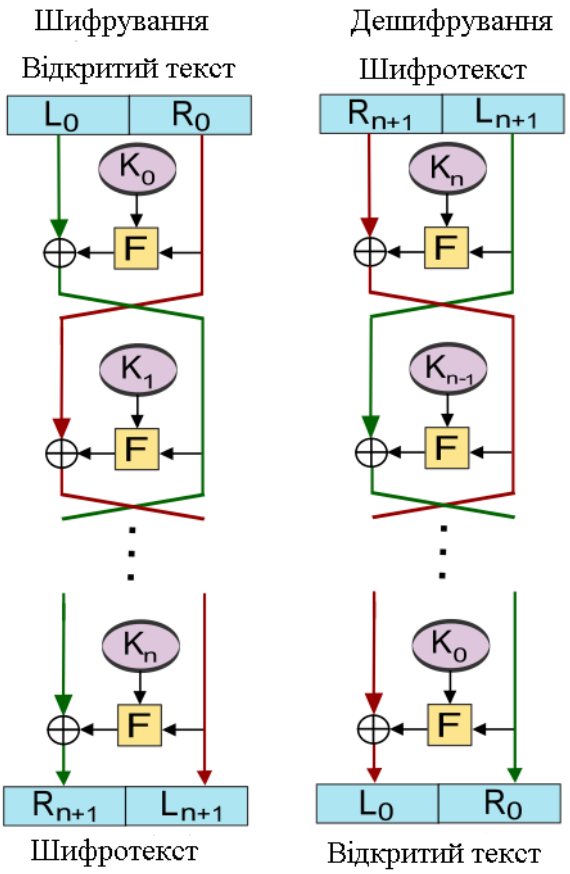

Рис. 2.2. Схема шифрування та дешифрування в мережі Фейстеля

Перераховані операції виконуються в кількості на одиницю меншій від числа раундів, а при переході між раундами змінюються ключі: нульовий змінюється на перший, а перший – на другий і так далі. При розшифруванні виконуються ті ж операції, а ключі застосовуються в зворотньому порядку.

2.2 Генерація унікальних перестановок

Оскільки в описуваному у цій роботі методі використовується генерація перестановки за заданим числом, то варто детальніше розглянути алгоритми із перестановками.

Генерація всіх перестановок для заданої деякої множини чи мультимножини є важливою як для спеціалістів в області комбінаторних задач так і для інших. Є різного роду задачі, де потрібні алгоритми із перестановками. Означення перестановки наступне: перестановкою порядку N називається розміщення N різних об’єктів в ряд за деяким порядком. Наприклад для деяких трьох об’єктів a, b та c існує шість наступних перестановок: abc, acb, bac, bca, cab, cba. Отже для деякої множини із N елементів можна зробити (побудувати) $N!$ різних перестановок: першу позицію можна зайняти N способами, а другу позицію – $N-1$ способом, оскільки один елемент вже зайнятий і далі за таким принципом та, відповідно, на останньому місці можна розмістити тільки один елемент. А загальна кількість варіантів розміщень наступна: $N*(N-1)*(N-2)*\ldots*1=N!$. Опис способів і алгоритмів генерації перестановок варто розпочати з алгоритму L.

Алгоритм L (або генерація лексикографічної перестановки). Для заданої послідовності n елементів $a_1 a_2 \ldots a_n$ початково відсортованих таким чином, що $a_1 \le a_2 \le \cdots \le a_n$ цей алгоритм генерує всі перестановки $\{a_1, a_2, \ldots, a_n\}$ проходячи їх в лексикографічному порядку. Допускається для зручності використовувати допоміжний елемент a_0 і він має бути меншим аніж найбільший елемент a_n. Здійснюється все в чотири кроки. Перший крок – це переглянути

перестановку $a_1 a_2 \dots a_n$. Другий крок – встановити $j \leftarrow (n-1)$ і, якщо $a_j \geq a_{j+1}$, тоді відбувається зменшення j на 1 аж поки не виконається умова, що $a_j < a_{j+1}$, а якщо $j = 0$, то робота алгоритму припиняється, оскільки в такому випадку j є найменшим індексом та, відповідно, тут були переглянуті всі перестановки починаючи з $a_1 \dots a_j$. Це означає, що значення a_j буде збільшеним при наступні перестановці. На третьому кроці встановлюється $l \leftarrow n$ і, якщо $a_j \geq a_l$, то зменшувати до тих пір, поки не буде виконуватися умова $a_j < a_l$ і, після цього потрібно поміняти місцями $a_j \leftrightarrow a_l$.Оскільки $a_{j+1} \geq \dots \geq a_n$, то елемент a_l є найменшим елементом, який більший за a_j, що йде слідом за $a_1 \dots a_{j-1}$ у перестановці. Переду заміною було $a_{j+1} \geq \dots \geq a_{l-1} \geq a_l > a_j \geq a_{l+1} \geq \dots \geq a_n$, а після заміни стало $a_{j+1} \geq \dots \geq a_{l-1} \geq a_j > a_l \geq a_{l+1} \geq \dots \geq a_n$. На четвертому кроці потрібно встановити $k \leftarrow (j+1)$ і $l \leftarrow n$, а опісля, якщо $k < l$, поміняти місцями a_k і a_l $(a_k \leftrightarrow a_l)$ і також $k \leftarrow (k+1)$, $l \leftarrow (l-1)$ і повторювати поки не буде виконана умова $k > l$, потім повернутися до першого кроку [18].

В загальному за допомогою процедури у три етапи можна знайти лексикографічного спадкоємця будь-якого комбінаторного об’єкта $a_1 \dots a_n$:

1) Знайти таке найбільше значення j для якого a_j може бути збільшеним, тобто $a_j < a_{j+1}$

2) Збільшити a_j на найменше можливе значення

3) Віднайти лексикографічно (в перестановці) найкоротший шлях, щоб розширити новий об’єкт $a_1 \dots a_j$ до повного об’єкту.

Даний алгоритм виконує цю загальну процедуру у випадку генерації перестановки (рис.2.3). Є також інші способи і алгоритми генерації перестановок [20, 21].

Рис. 2.3. Приклад лексикографічної перестановки для послідовності 0125330

Суміжні заміни. Варто розглянути метод суміжних замін при перестановці, оскільки найпростіша можлива зміна перестановки полягає в заміні суміжних елементів і будь-яка перестановка може бути впорядкована при підборі правильної послідовності таких замін. Отже, коли піти звітнім шляхом, то можна отримати будь-яку потрібну перестановку, починаючи зі всіх впорядкованих елементів.

Алгоритм простих змін (Алгоритм P) – такий алгоритм, який для заданої послідовності $a_1 a_2 \dots a_n$, що складається з n різних елементів, генерує всі їхні перестановки за рахунок повторних замін у суміжних парах. Він використовує допоміжний масив $c_1 c_2 \dots c_n$, який являє собою таблицю інверсій, і проходить всі послідовності цілих чисел таких що $1 \leq c_j \leq j$ для всіх $1 \leq j \leq n$. А інший масив $o_1 o_2 \dots o_n$ відповідає за напрямок в яких змінюються елементи c_j. Існує ще

Алгоритм T (переходи простих змін). Це алгоритм вичислює таку таблицю $t[1], t[2], \dots, t[n! - 1]$, що дії алгоритме простих змін еквівалентні послідовним обмінам $a_{t[k]} \leftrightarrow a_{t[k]+1}$ для $1 \leq k < n!$ за умови $n \geq 2$. Робота алгоритму описується в п'ять етапів. На першому етапі потрібно встановити $N \leftarrow n!, d \leftarrow \frac{N}{2}, t[1] \leftarrow 1$і $m \leftarrow 2$. Наступний етап – визначити чи $m = n$ і, якщо це так, то завершити, а якщо – ні, то встановити $m \leftarrow (m + 1), d \leftarrow \frac{d}{m}$ і $k \leftarrow 0$. Третій етап – це встановити $k \leftarrow k + d$ і $j \leftarrow (m - 1)$ і до того, поки не виконається умова $j = 0$. Далі четвертим етапом буде встановлення $t[k] \leftarrow (t[k] + 1)$ і $k \leftarrow (k + d)$. На п'ятому етапі поки $j < (m - 1)$, потрібно, щоб $j < (j + 1)$ і встановити $t[k] < j$ та $k \leftarrow (k + d)$, а якщо $k < N$, то повернутися до третього етапу, в іншому випадку – до другого. Наприклад, якщо $n = 4$, то отримаємо таблицю $(t[1], t[2], \dots, t[23]) = (3, 2, 1, 3, 1, 2, 3, 1, 3, 2, 1, 3, 1, 2, 3, 1, 3, 2, 1, 3, 1, 2, 3)$.

Є також загальний генератор перестановок так званий алгоритм G, який для заданої таблиці Сімса $(S_1, S_2, \dots, S_n)$, де кожна множина S_k має кількість k+1 елементів $\sigma(k, j)$, генерує всі перестановки $a_0 a_1 \dots a_{n-1}$ для $\{0, 1, \dots, n - 1\}$, використовуючи додаткову допоміжну таблицю $c_n \dots c_2 c_1$. Важливою перевагою алгоритму G є те, що він проходить всі перестановки $a_0 \dots a_{k-1}$ аж до тих пір, поки не досягне a_k, а потім він виконує ще k! циклів перед зміною a_k і так далі. Відповідно, якщо в певний момент ми дійдемо до останніх елементів $a_k \dots a_{n-1}$, які не мають великого значення для задачі, що вирішується, то можливим є швидко пропустити, всі перестановки, які закінчуються суфіксами, що є неважливими. Також є важливим згадати генерацію всіх перестановок методом Кнута. Ідея полягає в тому, що: якщо побудовано всі перестановки довжини N, то для кожної такої перестановки можна побудувати $N + 1$ перестановку довжини $N + 1$. Наприклад, для перестановки 3241, можна побудува 5 різних перестановок довжини 5: $53241, 35241, 32541, 32451, 32415$ (рис.2.4). Перший спосіб генерації цим методом всіх перестановок для деякої перестановки довжини N полягає у двох діях:

– Додати в кінець перестановки деякі числа, що визначаються $\frac{2i+1}{2}$ $(0 \leq i \leq N;$

– В порядку зростання перенумерувати елементи отриманих перестановок.

3 2 4 1 0.5 → 4 3 5 2 1
3 2 4 1 1.5 → 4 3 5 1 2
3 2 4 1 2.5 → 4 2 5 1 3
3 2 4 1 3.5 → 3 2 5 1 4
3 2 4 1 4.5 → 3 2 4 1 5

Рис. 2.4. Приклад роботи методу Кнута для перестановки 3241

Є також ще метод рекурентного пошуку перестановок (рис.2.5), основою якого є ідея, що потрібно початкову задачу звести до аналогічної на меншому наборі даних.

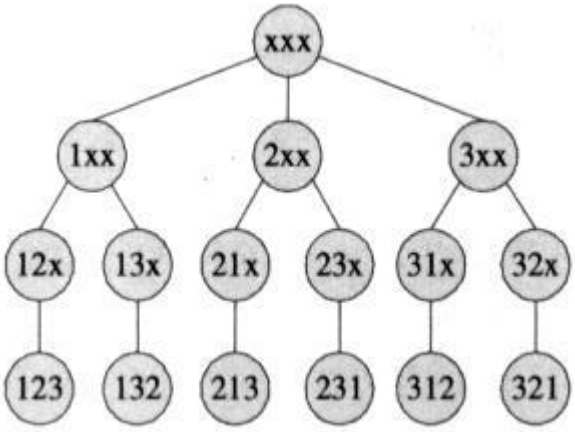

Рис. 2.5 Генерація рекурсією всіх перестановок для множини {1, 2, 3}

Алгоритм рекурентної генерації перестановок можна зобразити деревом і наприклад, для n=3 буде шість листків із результуючими перестановками і три гілки на початку.

2.2 Розробка алгоритму посилення криптостійкості симетричних алгоритмів шифрування даних

Даний метод підвищення криптостійкості можна поділити на три категорії роботи, як показано пунктирними блоками на рис.2.6.

Можна умовно виділити перший етап, який є відмінним від наступних. Під час даного етапу відбувається отримання ключа шифрування, та блоку даних який потрібно опрацювати і передати, що на схемі відноситься до категорії вхідних даних.

Для першого етапу знаходиться геш ключа і певним чином перетворюється, щоб вирівняти його по межам мінімального і максимально можливого значення номеру перестановки підблоків даних.

Дані з блоку розділяються на фрагменти та над ними виконується перестановка по номеру перестановки, який отримано із хешу ключа шифрування даних.

Генерується число псевдовипадковим алгоритмом, яке буде використане для перестановки даних другого блоку.

Пересортовані дані разом із числом заплутування даних, для наступного блоку, подаються у будь-який алгоритм симетричного шифрування даних. Дані і число формують єдиний блок, у якому саме число розміщується вкінці після даних.

Шифровані дані, цим методом, передаються по каналу передачі даних і опрацювання переходить до наступного етапу.

Другий блок даних подібно до першого подається на виконання перестановки над ним, але номером перестановки буде згенероване число, псевдовипадковим алгоритмом, із першого етапу. Дані передаються до отримувача і там виконується процедура дешифрування даних.

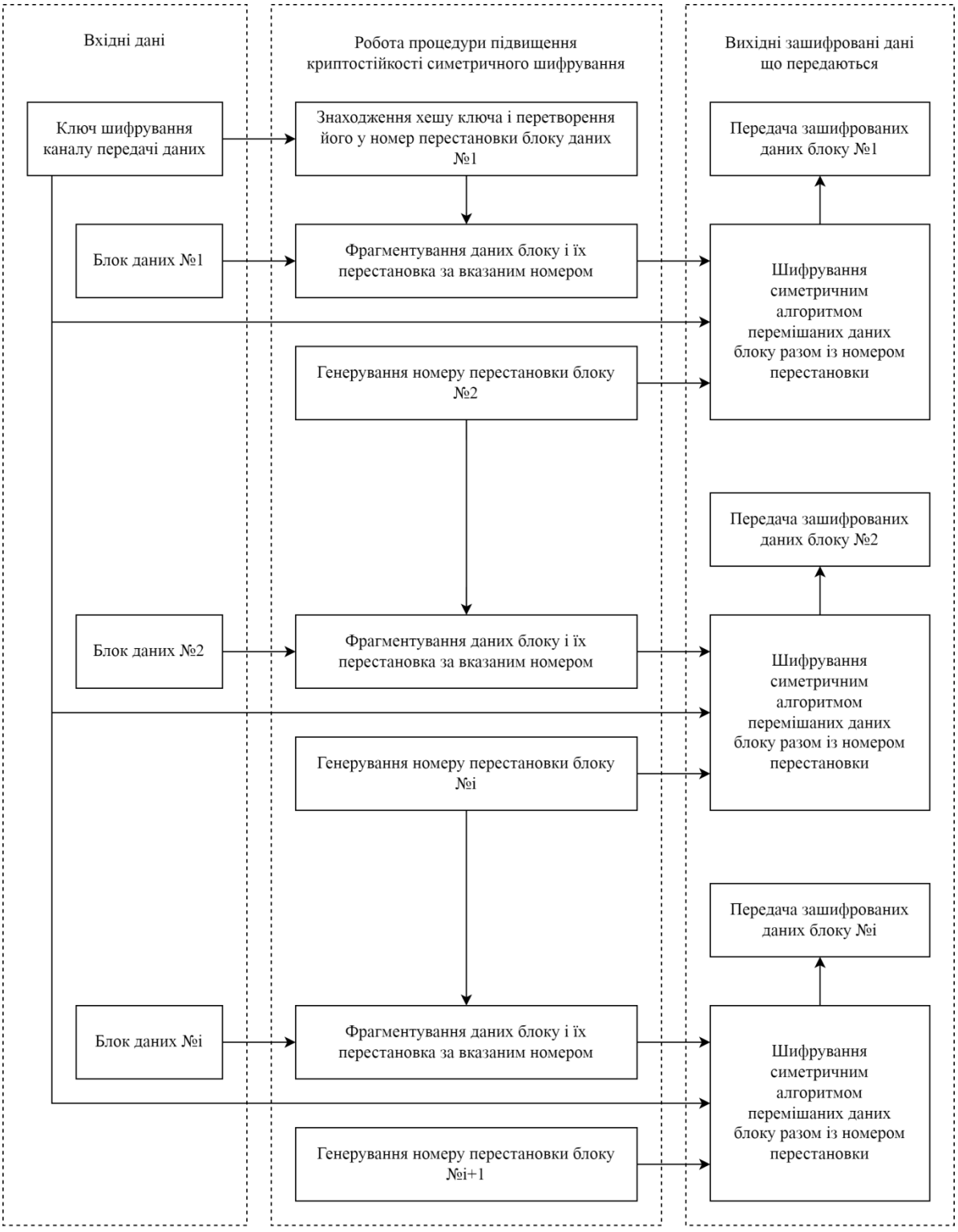

Рис. 2.6. Структурна схема шифрування із застосуванням методу підвищення криптостійкості симетричних алгоритмів шифрування

Для усіх наступних блоків шифрування відбувається аналогічним чином як і для другого блоку даних, що передаються на шифрування та їх передачу по інформаційному каналу.

Дешифрування відбувається подібним чином до шифрування, але у зворотному порядку із деякими відмінностями (рис.2.7).

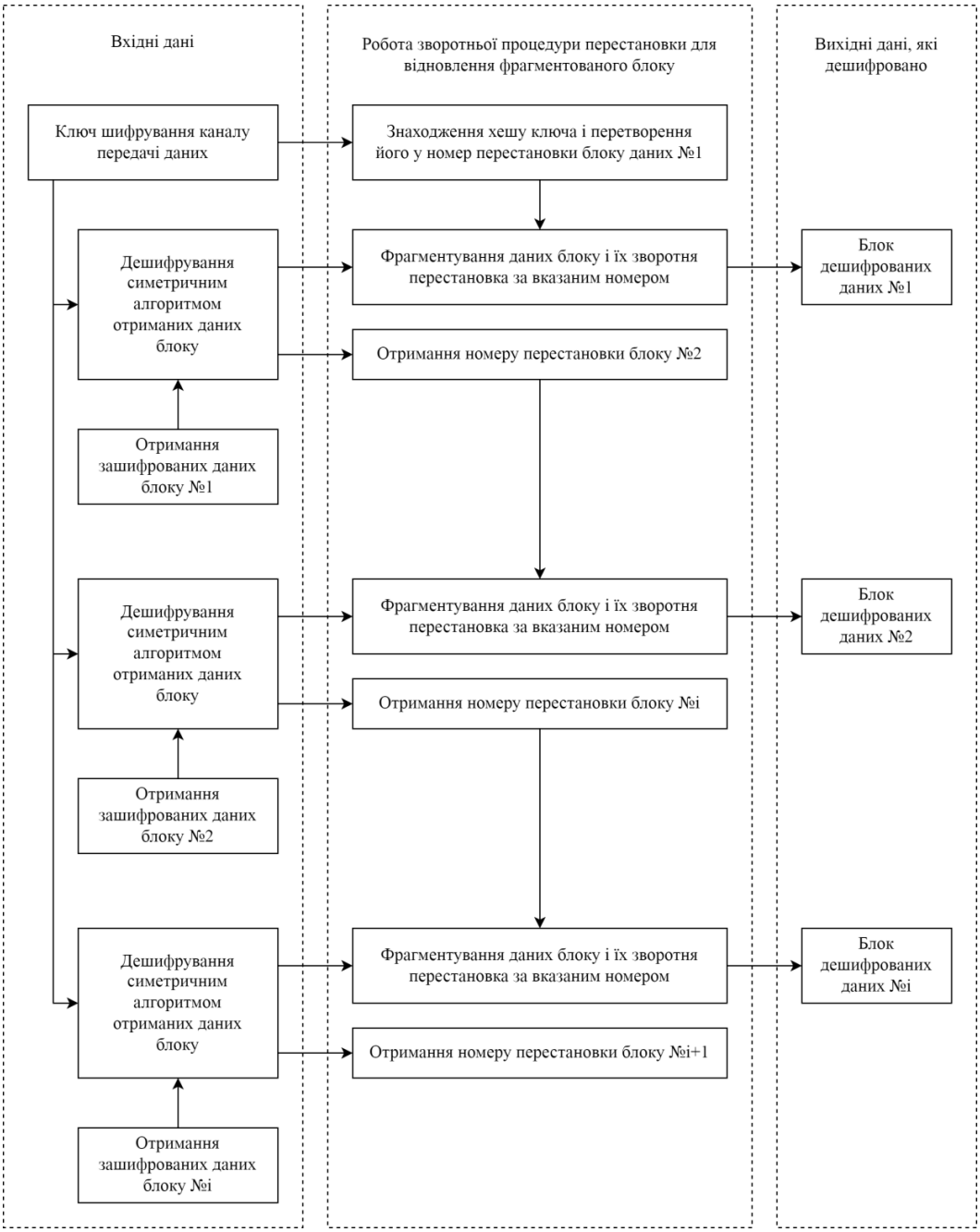

Рис. 2.7. Структурна схема дешифрування зашифрованих даних із застосованим методом заплутування інформації

Дані які надходять від передавача подаються разом із ключем у алгоритм симетричного дешифрування разом із спільним ключем даного закритого каналу передачі даних.

Ключ, шифрування та дешифрування даних, хешується і отримується номер перестановки, для першого блоку розшифрованих даних, подібно до того як це було у прямому порядку шифрування.

Частина даних, яка несе собою корисне інформаційне навантаження, передається до алгоритму зворотної перестановки де розсортовується у оберненому порядку, для відновлення початкового вигляду даних.

Алгоритм зворотної перестановки виконує перестановки підблоків у зворотному порядку, відповідно починаючи із останнього блока.

Номер блоку перестановки, у даному випадку, обчислюється діленням числа перестановки на факторіал із кількості опрацьованих блоків, яка починається з одиниці. У прямій перестановці число ділилося не на факторіал із кількості опрацьованих блоків, а просто на їхню кількість. Далі як і в прямому алгоритмі перестановки остача від ділення є номером блоку із яким поточний блок потрібно переставити відносно попередніх опрацьованих блоків, а ціла частина використовується для опрацювання наступних блоків у зворотному порядку.

Після зворотного розсортування підблоків, інформація є відновленою і передається далі куди потрібно.

Із даного блоку зчитується номер перестановки для наступного блоку.

Другий блок дешифрується, але розсортування вже проходить за числом перестановки отриманим із попереднього блоку.

Для всіх подальших блоків дешифрування інформації відбувається як і для другого блоку.

Приклад генерації перестановки.

Для описання алгоритму перестановки на рис.2.8 було зображено блок даних, який розділено на п’ять підблоків, як приклад, оскільки зображати всі етапи опрацювання більшої кількості підблоків, просто не доцільно. Така кількість блоків була вибрана виключно для того щоб розглянути деякі випадки перестановки і відобразити зміст усіх операцій на кожному етапі даного опрацювання

У дійсності варто обрати кількість субблоків більше 12-и, оскільки потрібно щоб була велика кількість варіантів перестановки, адже саме це добавляє невизначеності при спробі виконати криптоаналіз шифротексту.

Факторіал з п’яти буде 120. Дане число є максимальною кількістю варіантів перестановки субблоків. Оскільки воно є меншим за 255, то поміщається у один байт. Для алгоритму 128-бітного симетричного шифрування блок буде мати розмір 16 байт, із яких один байт буде виділено під номер перестановки, а решту 15 байт буде поділено на п’ять блоків меншого розміру по три байти, над якими й виконуватиметься процедура перестановки. Можна використати і інший варіант коли блок ділиться на 12 частин, а для збереження номеру перестановки вже потрібно 4 байти і із шістнадцяти байт залишається 12 байт які і будуть собою складати 12 частин по одному байту. Такий варіант вже матиме вагому криптостійкість, оскільки забезпечує майже пів мільярда варіантів перестановки, а так як для криптоаналізу потрібен не один блок, а велика кількість їх, то ця складність примножується із кожним отриманим блоком.

Підблоки рекомендовано робити як можна меншими, щоб кількість підряд розміщених даних, які можуть бути відомі криптоаналітику, була низькою це знижуватиме кількість відомої інформації криптоаналітику навіть якщо даним каналом передачі даних саме він передав власні дані.

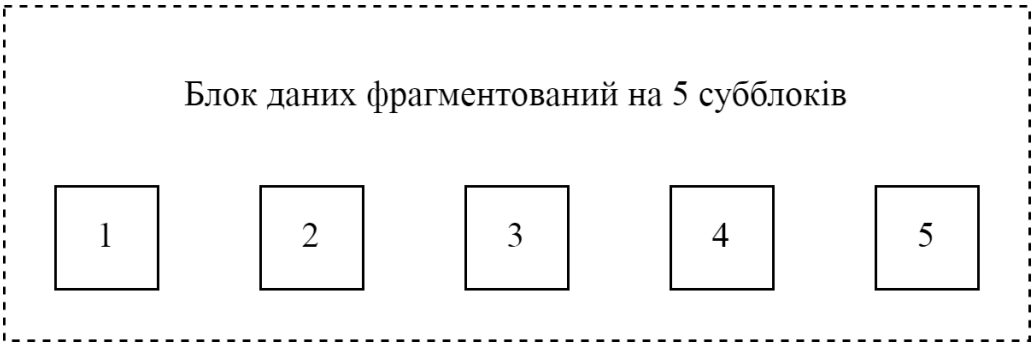

Рис 2.8. Приклад розділення блоку на 5 фрагментів меншого розміру

Оскільки в подальшому описі перестановки, будуть розглядатися виключно підблоки, то інколи вони будуть просто називатися блоками.

На кожному етапі блоки нумеруються зліва на право починаючи із певного блоку, якому присвоюється нульовий номер, наступному блокові – один і так далі.

Спочатку генерується номер перестановки у діапазоні від 0 до 5!-1, нехай буде наприклад 83. На першому кроці потрібно дане число поділити на всю кількість під блоків, тобто на п’ять. $\frac{83}{5} = 16\frac{3}{5}$, де ціла частина, число 16, береться для подальшого опрацювання, а остача, число 3, є номером субблоку із яким буде переставлений місцями нульовий блок, як показано стрілками на рис. 2.9.

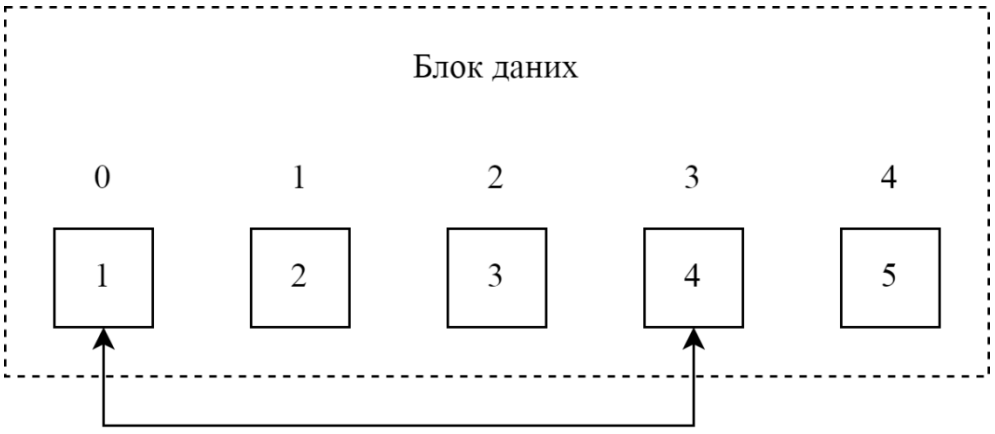

Рис 2.9. Перший етап перестановки

На рис. 2.10 показано, що блоки 1 і 4 переставлені місцями, а далі йде підготовка до другого етапу. Всі блоки, починаючи із блоку який у першому етапі мав номер 1, нумеруються зліва на право починаючи із нуля. Тепер кількість пронумерованих блоків буде на одиницю менше, тобто чотири.

Ціла частина, число 16 що залишилося після ділення на першому етапі, у цьому другому етапі таким же чином ділиться, але вже на 4, оскільки така кількість блоків була пронумерована на даному етапі.

$\frac{16}{4} = 4$, де ціла частина число 4 буде використовуватися на наступному етапі, а щодо цього етапу, то оскільки тут остачі від ділення не було, то це означає, що вона становить нуль і перестановка блоку вказує на сам блок, тобто його не потрібно переставляти із будь-яким іншим блоком і він залишається на своєму місці (див. рис. 2.10), а опрацювання переходить на наступний етап.

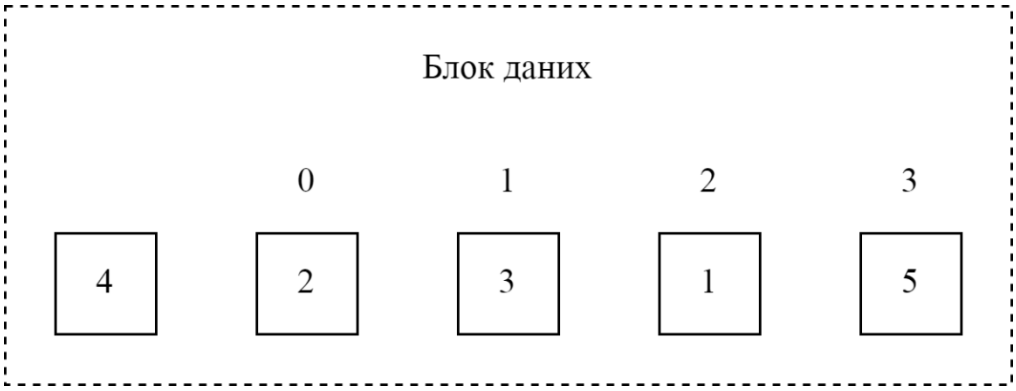

Рис 2.10. Другий етап перестановки

Третій етап, як і попередні етапи, починається із перенумерування блоків. Кількість блоків знову на один стало менше і тепер кількість можливих варіантів звідки може бути переставлений блок у нульову позицію складає 3. Як і у попередніх етапах відбувається обчислення номеру блоку для перестановки $\frac{4}{3} = 1\frac{1}{3}$. Остача від ділення одиниця тому блок який наразі має нульовий номер буде переставлений із сусіднім блоком під номером 1, як показано на рис. 2.11.

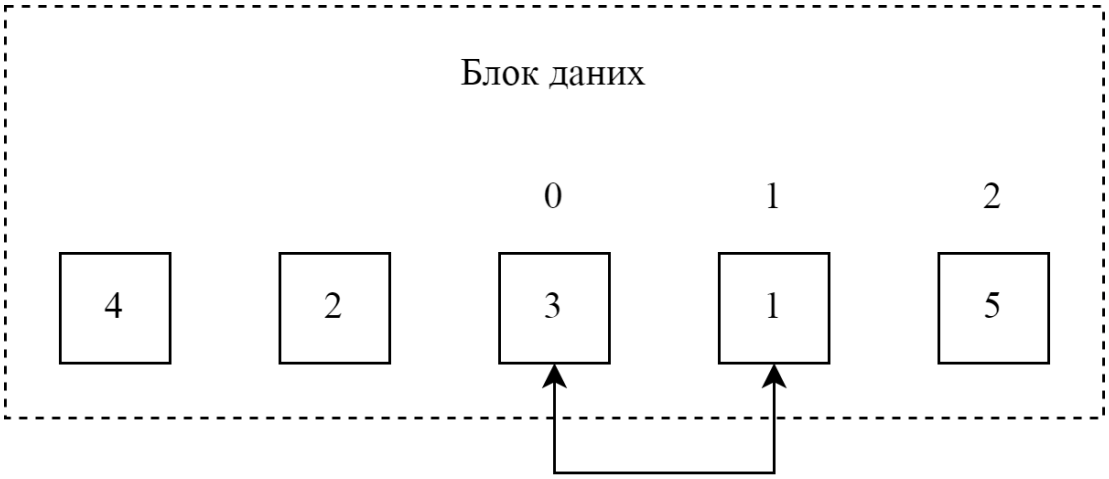

Рис 2.11. Третій етап перестановки

На четвертий етап перестановки залишилося тільки два блоки, тому номер блоку перестановки може бути один або нуль, а в даному випадку є число 1, яке при діленні на два дасть таку ж остачу, тому блок із тимчасовим номером 0 буде переставлено із тим що має номер 1 (рис. 2.12).

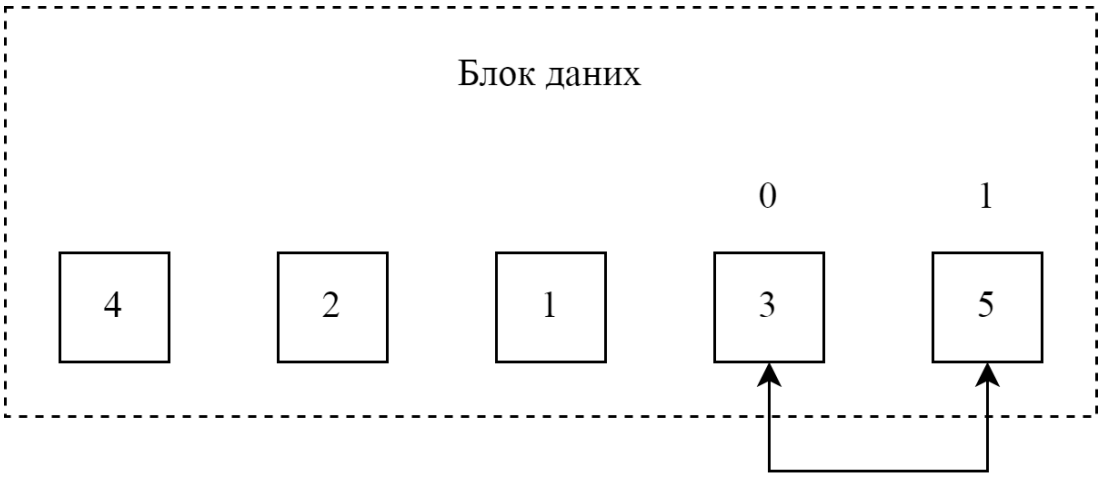

Рис 2.12. Четвертий етап перестановки

Оскільки залишився тільки один елемент, то для нього перестановки не існує і він вже знаходиться на своєму місці.

Підблоки пересортовано і отримана послідовність підблоків: 4, 2, 1, 5, 3 (рис.2.13).

Рис 2.13. Результат перестановки підблоків

Дана послідовність, яка згенерована таким алгоритмом перестановки, є унікальною для номеру 83 і більше не повторяється для всіх інших чисел у діапазоні від 0 до 119.

Послідовність готова до шифрування і може бути подана у будь-який алгоритм симетричного шифрування, разом із числом перестановки для наступного блоку даних.

2.3 Висновки до розділу 2

У цьому розділі була здійснена робота по розробці алгоритму для підсилення криптостійкості симетричних алгоритмів шифрування та наведено математичний опис і аналіз нестійкого алгоритму TEA.

Також було детально описано основні алгоритми та способи генерації перестановок.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ШИФРУВАННЯ

3.1 Опис середовища розробки PureBasic

PureBasic являє собою зручний для розробки компілятор з широкими можливостями для написання програмного коду. В пакеті із середовищем йде дуже багато прикладів і кодів базових програм. Був створений таким чином, щоб могли працювати як початківці, так і експерти.

Переваги та основні особливості Pure Basic:

- має великий набір вбудованих команд, яких є більше тисячі восьмиста, що дає можливість швидко і легко створити будь-який додаток, програму;
- створює оптимізований виконуваний код та є швидким компілятором;
- виконувані файли не потребують додаткових інтерпретаторів чи різних бібліотек (DLL);
- підтримуються всі ключові слова, які є в стандартному BASIC;
- для професійних розробників є цілком повний доступ до системних функцій (OS API);
- наявність простий, зрозумілого та швидкого 2D-рушія для розробки ігор (HGE, SDL, DirectX);
- має просту та дуже якісну реалізацію графічної 3D-розробки на рушії OGRE, який націлений спрости процес розробки програм із тривимірною графікою та є орієнтованим на сцену;
- вихідний код є сумісним між різними платформами;
- ефективне використання апаратних засобів, які доступні і ,відповідно, обчислювальних потужностей за рахунок використання максимально оптимізованих команд;

- наявність підтримки простору імен для легкого повторного використання коду;
- строга типізація та синтаксис для уникнення помилок при програмування;
- є підтримка процедур і структур для професійних програмістів;
- повна підтримка юнікоду;
- вбудований візуальний GUI-редактор;
- вбудовані три типи контейнерів: список, масив і мапа;
- має потужний дебагер і профайлер, що спрощує виявлення і аналіз помилок у коді.

Безкоштовна версія обмежує восьмиста рядками код, адже є пробною. Середовище є кросплатформним і підтримує розробку додатків на різні системи. Компілятор є багато прохідним та складається з інтерпретатора коду в асемблерний, компонувальника та асемблера (в середовищі застосовується саме FASM). Наявність асемблера дає змогу використовувати в коді асемблерні вставки.

Однойменна мова даного середовища базується на синтаксисі BASIC та є мовою високого рівня. Її можна віднести до мультипарадигмових мов програмування, адже в ній можна писати як імперативний код, так і процедурний та структурний. Типізація у мові є статичною. Перелік типів даних наведений у табл. 3.1 За допомогою синтаксичного елементу Structure користувач може створити свій власний тип даних, окрім наявних вбудованих елементарних типів.

Все середовище було оптимізовано аби надати максимальну швидкість і компактність програмам створеним з його допомогою. Швидкість програм є на рівні майже будь-якого професійного компілятора, а розмір виконуваного файлу є оптимальним та відносно малим, який також не потребує виконання інтерпретатором в режимі реального часу [19].

Таблиця 3.1

Основні типи даних PureBasic

Тип	Числовий діапазон	Використовуваний розмір пам’яті	Суфікс
Byte	Від -128 до 127	8 біт (1 байт)	.b
Character	Від 0 до +255	8 біт (1 байт)	.c
ASCII	Від 0 до +255	8 біт (1 байт)	.a
Word	Від -32768 до +32768	16 біт (2 байти)	.w
Unicode	Від 0 до +65535	16 біт (2 байти)	.u
Character	Від 0 до +65535	2 байти, 16 біт, (юнікод)	.c
Long	Від -2147483648 до +2147483647	32 біти (4 байти)	.l
Integer	Від -2147483648 до +2147483647	32 біти, 4 байти (для x86)	.i
Float	Залежить від відношення між цілою і дробовою частинами	32 біти (4 байти)	.f
Integer	Від -9223372036854775808 до + 9223372036854775808	64 біта, 8 байт (для x64)	.i
Double	Залежить від відношення між дробовою та цілою частинами	64 біти (8 байт)	.d
Quad	Від -9223372036854775808 до + 9223372036854775808	64 біти (8 байт)	.q
String	Немає обмежень	Довжина рядка+1 (в байтах)	.s
Fixed String	Немає обмежень	Довжина рядка (у байтах)	.s{довжина рядка (в байтах)}

Якщо купити професійну версію, то всі майбутні оновлення будуть безкоштовними. Компанія-розробник – Fantaisie Software – на своєму сайті заявляє, що найперший користувач, зареєстрований у 1998 році, отримує і надалі всі оновлення та такий підхід до продаж вони не будуть змінювати, але лише для тих, хто зареєстрований на їхньому сайті. Також, користувачі, які придбали середовище PureBasic можуть працювати в ньому та писати код на різних операційних системах або завантажувати його на інші комп'ютери, оскільки ліцензія за якою поширюється ця програма є користувацькою.

В табл. 3.2 Наведено опис усіх основних версій (етапів розвитку) цього середовища.

Таблиця 3.2

Основні версії середовища PureBasic

Версія	Дата випуску	Опис
PureBasic v1.00	1 вересня 1999 р.	Перша версія програми, яка була зорієнтована на роботу в AmigaOS
PureBasic v1.10	10 вересня 1999 р.	З'явилася повноцінна підтримка під AmigaOS мікропроцесорної PowerPC
PureBasic v1.20	30 листопада 1999 р.	Вбудовано редактор і наявна підтримка бібліотек AmigaOS
PureBasic v1.60	9 вересня 2000 р.	Реалізовано парадигму програмування для AmigaOS без втрат в бета-версіях для Microsoft Windows і Linux
PureBasic v2.00 Final (кінцева, повна)	17 грудня 2000 р.	Перша повноцінна версія для ОС Windows доступна широку колу користувачів
PureBasic v4.00 Beta	26 жовтня 2006 р.	Стабільна версія для AmigaOS і Linux

Продовж. табл. 3.2

PureBasic v4.00 Final	8 травня 2006 р.	Стабільна версія для Windows
PureBasic v4.10 Final	9 листопада 2007 р.	Перший одночасний випуск для всіх підтримуваних платформ (Linux, Microsoft Windows та Mac OS X)
PureBasic v5.00	5 листопада 2012 р.	Версія для трьох ОС (як 32-розрядних, так і для 64-розрядних): Linux, Windows та Mac OS X
PureBasic v5.30	23 липня 2014 р.	Реалізовано простір імен
PureBasic v5.40 LTS	16 жовтня 2015	Оновлення до версії 5.40 з довготерміновою підтримкою
PureBasic v5.50	25 липня 2016 р.	Підтримка юнікоду в компіляторі і додані функції для роботи з 3D
PureBasic v5.60	2 травня 2017 р.	Додано GIF-декодер та багато іншого
PureBasic v5.61	12 вересня 2017 р.	Виправлена частина поширених помилок в бібліотеках і компіляторі.
PureBasic v5.70 LTS	2 січня 2019 р.	Розширено підтримку кросплатформності, зроблено підсистему QT для Linux, додано функцію для роботи з базами даних.
PureBasic v5.71	16 серпня 2019 р.	Виправлені помилки в компіляторі.

Є оновлена версія 5.73 випущена 2020 року в листопаді з виправленими помилками та декілька функцій, одна з яких створена для підтримки старих серверів і стосується HTTP – запитів.

3.2 Опис коду програмного забезпечення

Код програми написано двома мовами програмування. Основною із них є мова високого рівня “PureBasic” у якій використовуються вставки низькорівневої мови програмування “Assembler”.

На початку коду розташовано добавлення із файлу “IncludeFile_Window.pb” коду інтерфейсу програми. Нижче об’явлення процедур, які необхідні для роботи програми (рис.3.1).

```
IncludeFile "IncludeFile_Window.pb"

Procedure.l fl(n.l)
  res.l = 1
  For i=2 To n
    res*i
  Next
  ProcedureReturn res
EndProcedure

Procedure.l Random32()
  n.l = 2<<16 - 1
  ProcedureReturn Random(n)*256 + Random(n)
EndProcedure
```

Рис. 3.1. Вставка файлу і об’явлення процедур

Виконання коду розпочинається із визначення змінних, які будуть в подальшому будуть використовуватися (рис.3.2).

```
;------- Визначення змінних -------
PathToSourceFile.s = ""
PathToDestinationFile.s = ""
passwordString.s = ""


;12 Блоків по 4 Байти, 4 Байти ном
n.l = 12
fl_n.l = fl(n)

PermutationNumber.l = 0

pointerMessageString.l = 0
pointerResMessageMemory.l = 0
;--------------------------------
```

Рис. 3.2. Визначення змінних

Якщо потрібно, деякі змінні можуть бути визначенні під час виконання коду програми.

Далі виконується виклик процедури “ Open_Window_0”, яка розміщена у файлі “ IncludeFile_Window.pb” (рис.3.3).

```
Open_Window_0()

Repeat
  Event = WindowEvent()
  If Event = #PB_Event_Gadget
        ●     ●     ●
  EndIf
Until Event = #PB_Event_CloseWindow

End
```

Рис. 3.3. Запуск і опрацювання подій вікна

Код процедури запуску вікна інтерфейсу програми та створення його віджетів можна побачити на рис.3.4.

```
Procedure Open_Window_0()
  If OpenWindow(#Window_0, 0, 0, 600, 140, "Шифрамор",  #PB_Win
    If CreateGadgetList(WindowID(#Window_0))
      TextGadget(#Text_0, 20, 20, 80, 20, "Вхідний файл:")
      StringGadget(#String_0, 140, 20, 320, 20, "")
      ButtonGadget(#Button_0, 480, 20, 100, 20, "Вибрати файл")
      TextGadget(#Text_1, 20, 60, 80, 20, "Вихідний файл:")
      StringGadget(#String_1, 140, 60, 320, 20, "")
      ButtonGadget(#Button_1, 480, 60, 100, 20, "Вибрати файл")
      TextGadget(#Text_2, 20, 100, 120, 20, "Ключ шифрування:")
      StringGadget(#String_2, 140, 100, 200, 20, "")
      ButtonGadget(#Button_2, 360, 100, 100, 20, "Шифрувати")
      ButtonGadget(#Button_3, 480, 100, 100, 20, "Дешифрувати")

    EndIf
  EndIf
EndProcedure
```

Рис.3.4. Код створення інтерфейсу програми

Коли вікно створене розпочинається основний цикл роботи програми. Процедурою “WindowEvent”, зчитуються події, які відбуваються із вікном і потребують опрацювання відповідним кодом програми (див.рис.3.3).

Вкінці циклу відбувається перевірка чи не була натиснута кнопка закривання вікна, якщо – так, то цикл завершується виконання коду зупиняється командою “End”.

Однією із подій є опрацювання віджетів вікна, яке перевіряється константою “#PB_Event_Gadget”. Для їхнього опрацювання викликається процедура “EventGadget”, яка повертає номер віджета із яким відбулася певна подія (рис.3.5).

```
EventGadget = EventGadget()
Select EventGadget
  Case #Button_0
    PathToSourceFile = OpenFileRequester("Выберіть ф
    SetGadgetText(#String_0, PathToSourceFile)
  Case #Button_1
    PathToDestinationFile = OpenFileRequester("Выбер
    SetGadgetText(#String_1, PathToDestinationFile)
  Default
    PathToSourceFile = GetGadgetText(#String_0)
    PathToDestinationFile = GetGadgetText(#String_1)
    passwordString = GetGadgetText(#String_2)
    Select EventGadget
      Case #Button_2
        CALL l_encrypteddata
      Case #Button_3
        CALL l_decrypteddata
    EndSelect
EndSelect
```

Рис.3.5. Опрацювання віджетів вікна

В даному коді йде опрацювання натискання кнопки вікна. Перші дві кнопки “#Button_0” і “#Button_1” відповідають за вибір вхідного і вихідного файлу, відповідно.

Наступні дві кнопки “#Button_2” та “#Button_3” виконують виклик функцій опрацювання шифрування і дешифрування даних, відповідно.

Коли файли для роботи вказано і виконується виклик процедури шифрування даних, то

Функція шифрування даних, розпочинається із коду, який відкриває і завантажує вхідний файл у виділену пам’ять по розміру даного файлу (рис.3.6).

```
;-------------------------------------------------------
If OpenFile(0, PathToSourceFile)
lenMessageString.l = Lof(0)
MessageString.l = AllocateMemory(lenMessageString)

nBlocks = Round(lenMessageString/48, #PB_Round_Up)
lenResMesageMem.l = nBlocks*64
resMessageMemory = AllocateMemory(lenResMesageMem)

ReadData(0, MessageString, lenResMesageMem)
CloseFile(0)
;-------------------------------------------------------
```

Рис.3.6. Читання вхідного файлу

Зчитується розмір файлу і виконується визначення кількості блоків, які потрібні для опрацювання даного файлу. Під дану кількість блоків виділяється пам’ять із якою наділі буде працювати код даної функції.

Далі код виконує обчислення хеш-значення, від ключа шифрування даних, та вирівнює його значення в межах від 0 до 12!-1 (рис.3.7).

```
;{ Обчислення номеру першої пересмановки
passFPHex.s = MD5Fingerprint(@passwordString, Len(passwordString))

fPN_formig.q = 0
;{ 16 байт md5 перемворюються діляться на дві частини по 8 байт і с
Hex2Dec(Mid(passFPHex, 1,8))
MOV dword [v_fPN_formig + 4], eax
Hex2Dec(Mid(passFPHex, 9,8))
MOV dword [v_fPN_formig], eax

Hex2Dec(Mid(passFPHex, 17,8))
XOR dword [v_fPN_formig + 4], eax
Hex2Dec(Mid(passFPHex, 25,8))
XOR dword [v_fPN_formig], eax           ;-8122604762126187740
;}

MOV eax, dword [v_fPN_formig]
MUL dword [v_fl_n]
MOV ecx, edx
MOV eax, dword [v_fPN_formig + 4]
MUL dword [v_fl_n]
ADD eax, ecx
JNC l_notaddcarry
INC edx
notAddCarry:
MOV [v_PermutationNumber], edx
;}
```

Рис.3.7. Обчислення номеру першої перестановки

Виконується підготовка змінних для роботи із пам’яттю і основним циклом опрацювання даного алгоритму (рис.3.8).

```
MOV eax, [v_MessageString]
; MOV eax, [eax];
MOV [v_pointerMessageString], eax
MOV eax, [v_resMessageMemory]
; MOV eax, [eax]
MOV [v_pointerResMessageMemory], eax


inBlocks = lenMessageString
SUB [v_inBlocks], 48
JL l_endcycleblocksprocessing
BeginCycleBlocksProcessing:


MOV ecx, 48
```

Рис.3.8. Підготовка змінних для роботи із пам’яттю і початок циклу

Починаючи із мітки “BeginCycleBlocksProcessing” йде код основного циклу даного опрацювання даних.

З мітки “LastBlockProcessing”, йде цикл зчитування 48 байт для виконання над ними перестановки фрагментів по чотири байти (рис.3.9).

```
MOV ecx, 48

LastBlockProcessing:

MOV esi, [v_pointerMessageString]
ADD dword [v_pointerMessageString], 48
MOV edi, [v_pointerResMessageMemory]
ADD dword [v_pointerResMessageMemory], 48

REP MOVSB
```

Рис.3.9. Зчитування 48-ми байт для опрацювання блоку

Генеруються випадкові дані, як показано на рис.3.10. У змінну “NextPermutationNumber” генерується випадкове число, яке буде номером перестановки для наступного блоку даних.

```
NextPermutationNumber = Random(fl_n)

MOV edi, [v_pointerResMessageMemory]
MOV [edi], eax
ADD dword [v_pointerResMessageMemory], 4

; Додавання випадкових даних
Random(2<<32-1)
MOV edi, [v_pointerResMessageMemory]
ADD dword [v_pointerResMessageMemory], 4
MOV [edi], eax
Random(2<<32-1)
MOV edi, [v_pointerResMessageMemory]
ADD dword [v_pointerResMessageMemory], 4
MOV [edi], eax
Random(2<<32-1)
MOV edi, [v_pointerResMessageMemory]
ADD dword [v_pointerResMessageMemory], 4
MOV [edi], eax
```

Рис.3.10. Генерування випадкових даних

На рис.3.11 показано цикли сумування по модулю два трьох блоків випадкових даних із вхідними даними.

```
MOV esi, [v_pointerResMessageMemory]
SUB edi, 60; всіх даних у блоці 64 б
MOV edx, 3; Кількість повторень цикл
XorCycle:
MOV ecx, 4; Кількість повторень цикл
SUB esi, 16
XorSmallCycle:
LODSD; MOV eax, [esi]
XOR eax, [edi]  ;MOV eax, [edi]
STOSD
LOOP l_xorsmallcycle
DEC edx
JG l_xorcycle

SUB esi, 64
```

Рис.3.11. Цикл виконання суми по модулю над даними

Дальше виконується цикл який виконує перестановку місцями фрагментів даних по чотири байти (рис.3.12).

```
MOV ecx, dword [v_n]
; DEC ecx
MOV eax, dword [v_PermutationNumber]
PermutationCycle:
XOR edx, edx; Якщо не обнулити, то к
DIV ecx
MOV ebx, [esi]
XCHG dword [edx*4+esi], ebx
MOV dword [esi], ebx
ADD esi, 4
LOOP l_permutationcycle
```

Рис.3.12. Основний цикл перестановки фрагментів даних

Після завершення всіх циклів опрацювання даних вони передаються у процедуру симетричного шифрування даних. Зашифровані дані зберігаються у файл і виконується повернення із даної процедури опрацювання даних (рис.3.13).

```
AESEncoder(resMessageMemory, resMessageMemory, lenResMesageMem

If OpenFile(0, PathToDestinationFile)
WriteData(0, resMessageMemory, lenResMesageMem)
CloseFile(0)

EndIf
EndIf

RET
```

Рис.3.13. Шифрування і збереження даних у файл

Коли натиснута кнопка дешифрування даних, то виконується функція “DecryptedData”. Дана функція подібна до функції шифрування, але із певними відмінностями. Спочатку відбувається дешифрування даних, а далі передається у цикл зворотної перестановки фрагментів даних.

Цикл зворотної перестановки даних дуже подібний до прямої перестановки, тільки ділення числа перестановки відбувається не на кількість не опрацьованих підблоків, а на факторіал із їхньої кількості.

Результат дешифрування даних зберігається у вихідний файл.

3.3 Результати тестування

Програма написана виключно для тестування дієздатності цього методу підвищення криптостійкості симетричного шифрування даних, тому виконана у простому варіанті.

Для тесту програми потрібно створити файл із певними даними, так званим повідомленням, який буде зашифровано даним способом (рис.3.14).

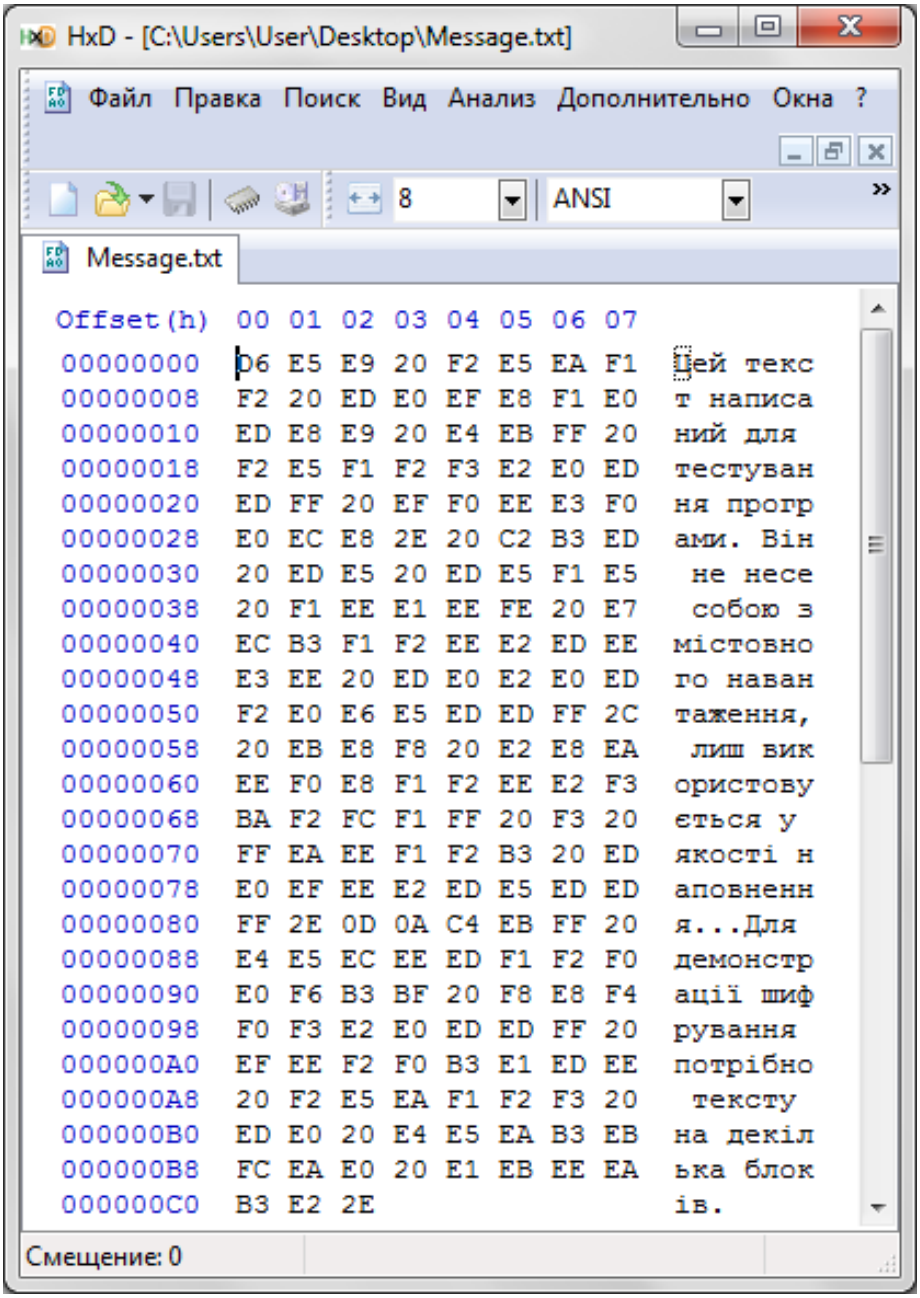

Рис. 3.14. Файл з даними для зашифрування

Програма генерує число перестановки, для першого блоку із “MD5” хешу ключа шифрування даних. Виконує перестановку фрагментів даних, розміром по чотири байти. Результат такого перемішування даних, вказаного повідомлення, показано на рис.3.15.

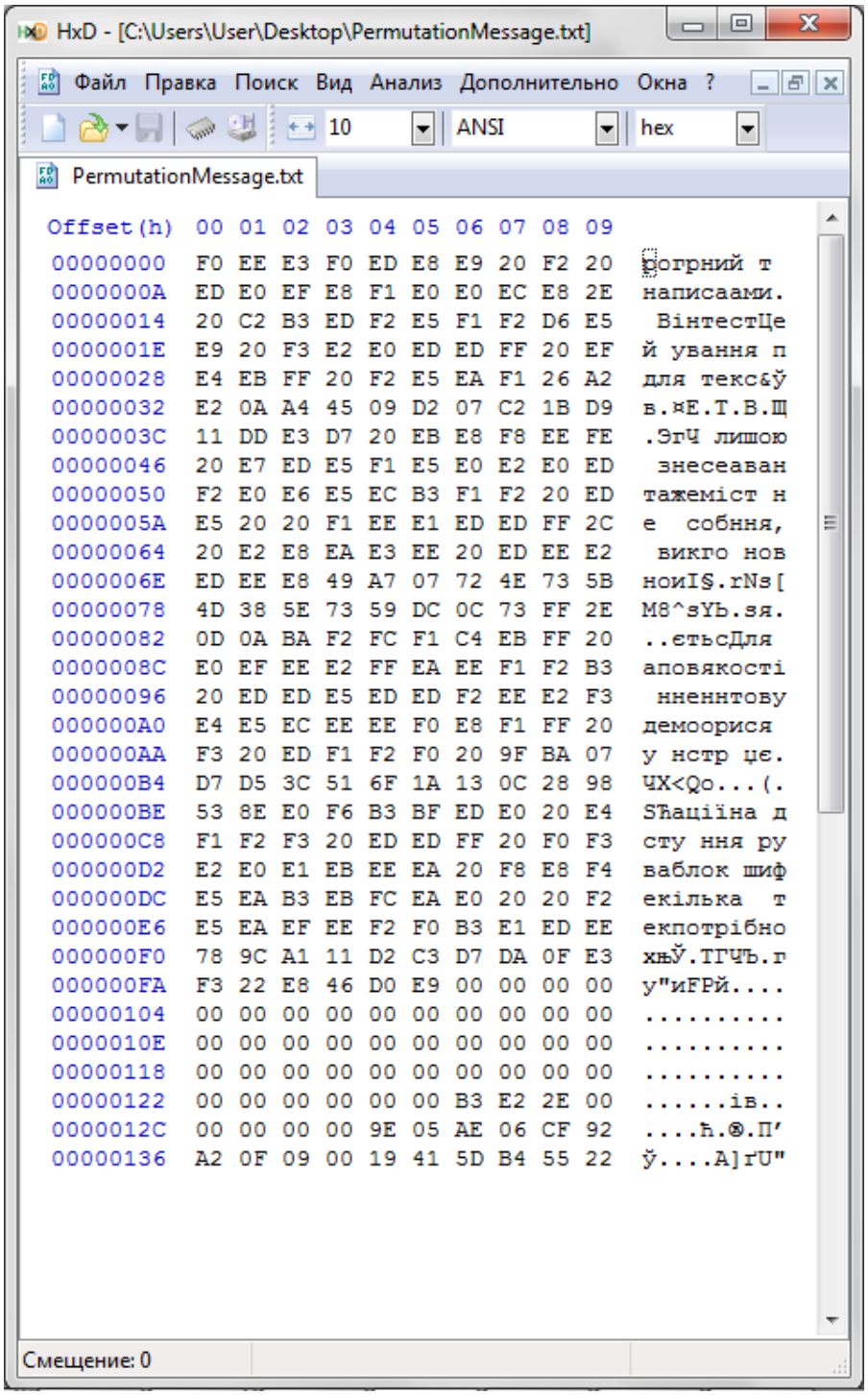

Рис. 3.15. Файл з перестановленими фрагментами даних

Як видно із рис.3.15 після виконання перестановки фрагментів даних, текст вже став не читабельними, але все ще може бути відновлений певним не простим методом.

Коли ж дані було перемішано, то в кінці розміщується номер для генерації перестановки наступного блоку і дані подаються у алгоритм симетричного шифрування даних. Зашифрований файл зображено на рис.3.16.

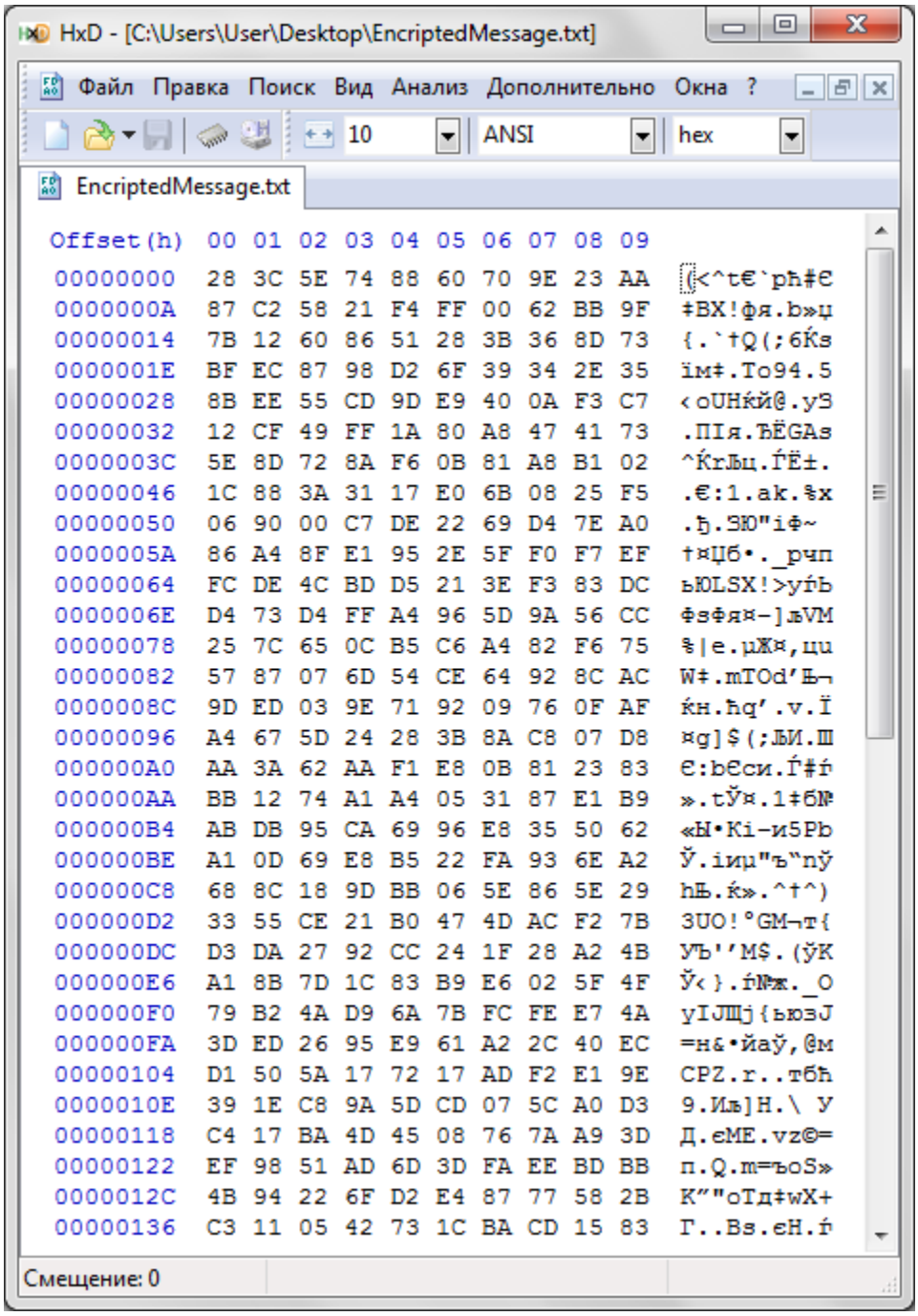

Рис. 3.16. Файл із зашифрованими даними

Коли виконано шифрування даних, то їх відновити без ключа шифрування, стає майже неможливою задачею.

Тепер отримані дані потрібно дешифрувати, для того щоб показати дієздатність даного алгоритму (рис.3.17).

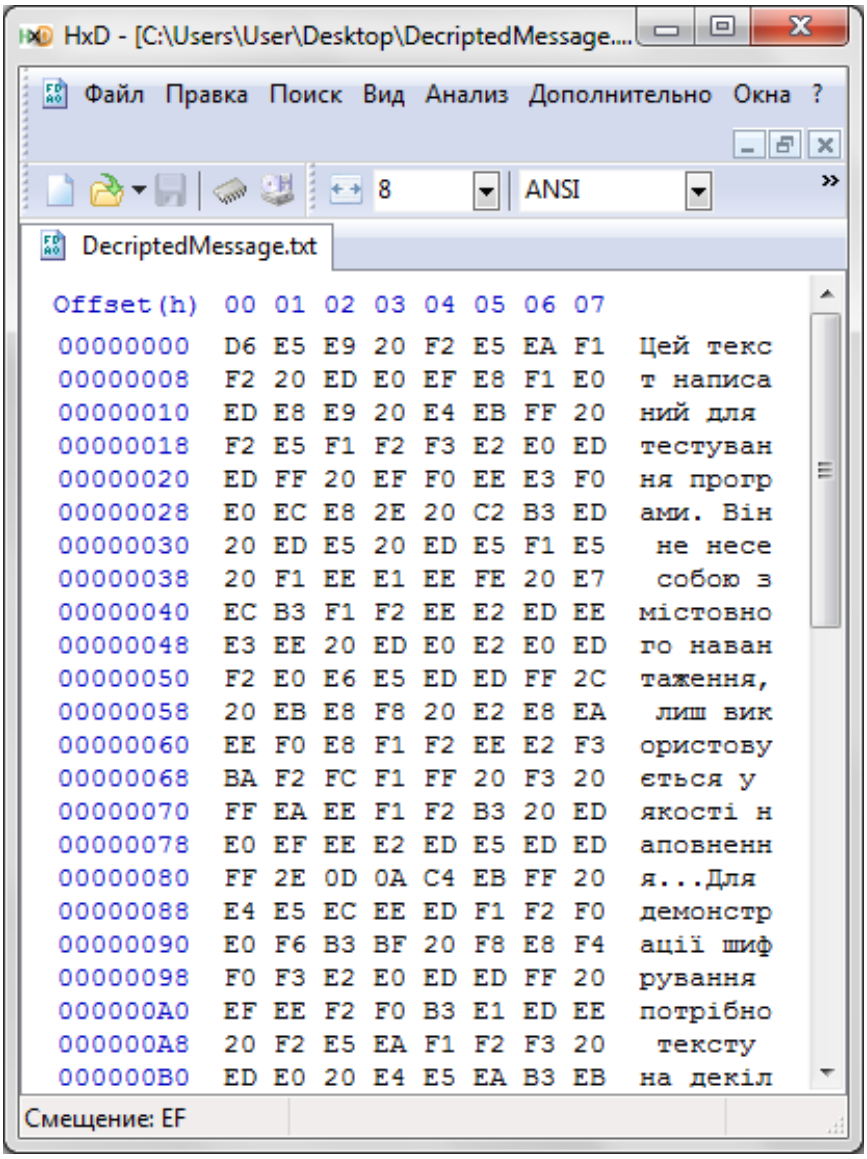

Рис. 3.17. Дешифрований файл

Для іншого прикладу можна навести ще один текст щоб показати дієздатність алгоритму і на інших даних. Вхідні дані показано на рис.3.18.

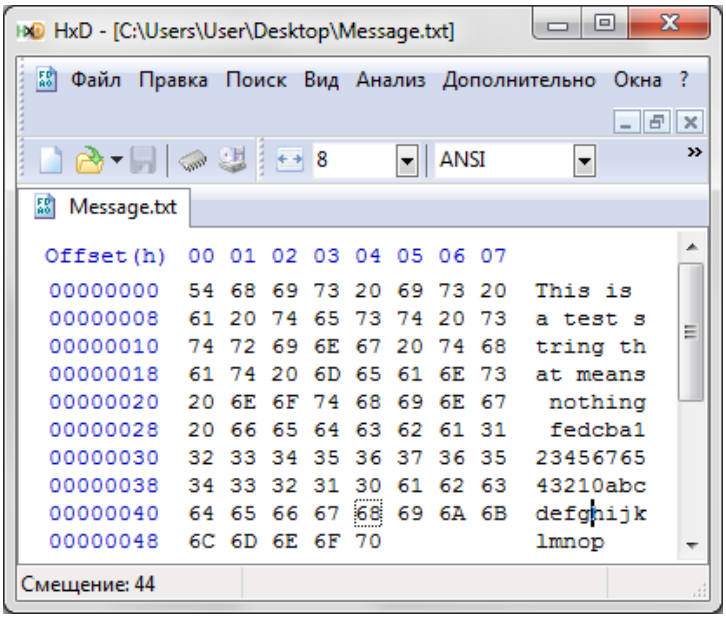

Рис. 3.18. Вхідне повідомлення в алгоритм шифрування даних

Подібно виконується заплутування даних через перестановку їх фрагментів і подається разом із ключем в алгоритм шифрування даних. Шифротекст показано на рис.3.19.

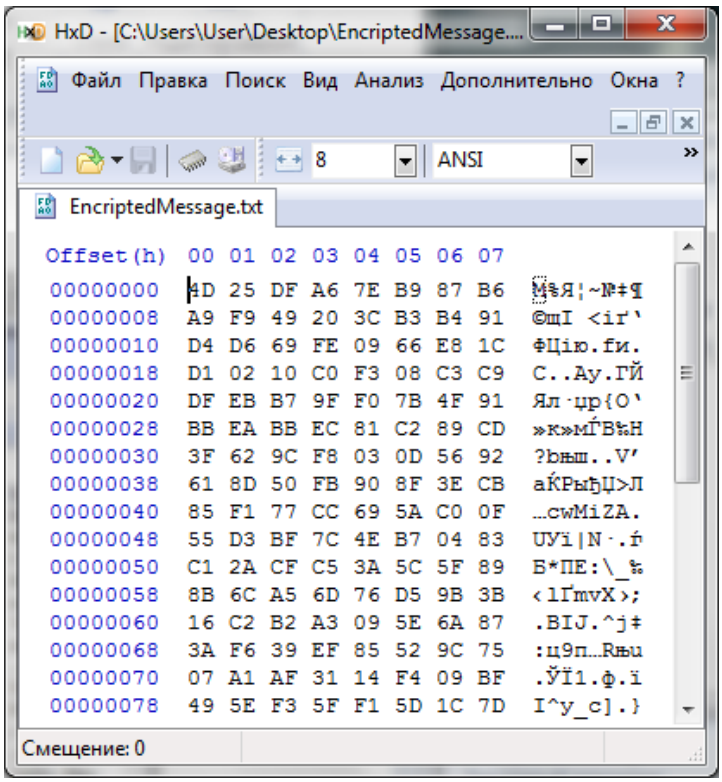

Рис. 3.19. Шифротекст файлу з повідомленням

І останнім кроком буде дешифрування цих даних. Відновлений файл, алгоритмом дешифрування даних, показано на рис.3.20.

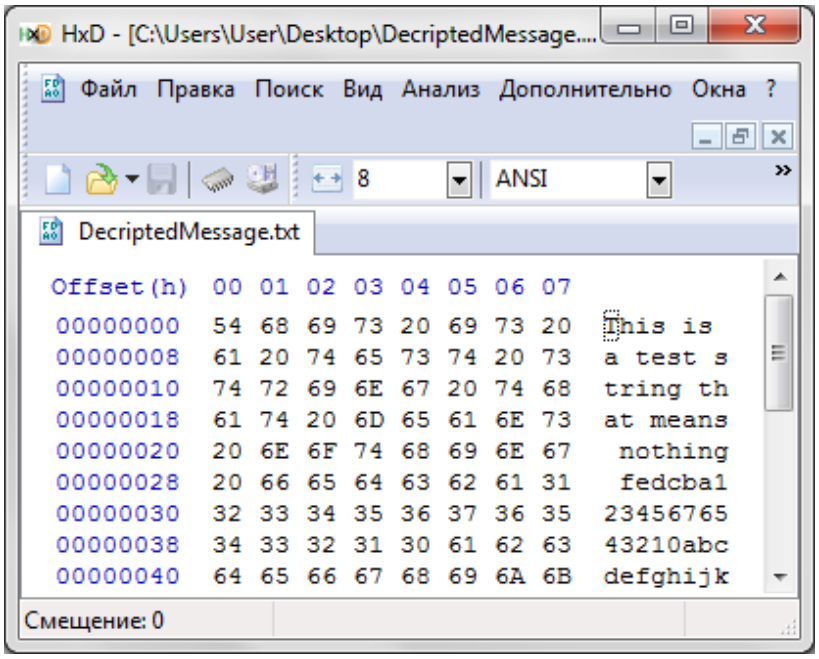

Рис. 3.20. Дешифрований файл повідомлення

На цих прикладах можна побачити, що алгоритм справно функціонує і правильно виконує шифрування та дешифрування даних. Дані зашифровані таким алгоритмом можна безпечно відправляти відкритими каналами даних, оскільки спроба відновлення даних із використанням криптоаналізу стає непосильною для сучасних методів виконання такої задачі та комп'ютерних засобів на яких здійснюється реалізовуються обчислення задачі криптоаналізу.

3.4 Висновки до розділу 3

У цьому розділі було виконано:

- математичний опис та аналіз нестійкого симетричного алгоритму TEA, як прикладу слабких алгоритмів, що можуть мати широке застосування;

- наведено основну теорію по генерації перестановок, адже в алгоритмі є етапи, де здійснюються перестановки;
- розроблено алгоритм для підсилення криптостійкості слабких симетричних алгоритмів шифрування із детальним його описом.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

Робота з комп'ютером чи комп'ютерними системами не передбачає зазвичай якого надмірного впливу на здоров'я, оскільки немає шкідливих речовин. Проте є різні фактори ризику, які можуть становити небезпеку для здоров'я та зашкодити правильному функціонуванню деяких органів тіла. Мається на увазі небезпека для зору при тривалій роботі за комп'ютером чи спини. Є також ризики для здоров'я пов'язані із проблемою надмірної ваги, яка з'являється як наслідок сидячої роботи та малорухомості.

Оскільки в даній роботі розроблено алгоритм та його програмну реалізацію, яка не потребує роботи із системами, що можуть становити небезпеку для здоров'я, то загалом потрібно лише дотримуватися правил безпеки при роботі із комп'ютерами. Вони регулюються законодавчими та нормативно-правовими актами, які, зокрема, визначають обов'язки роботодавця щодо забезпечення працівникам комфортних та безпечних умов для виконання поставлених задач. Ці обов'язки, а також права працівників щодо умов праці передбачені частиною 2 ст.2 і частиною 1 ст.21 КЗпП, а також ст.13 Закону України «Про охорону праці», у яких прописані основні положення з реалізації конституційного права працівників [24].

Законодавчі та нормативно-правові акти встановлюють єдиний порядок організації охорони праці в Україні, а також відносини між роботодавцем та працівником з питань безпеки, гігієни праці та виробничого середовища. На їх основі розроблені такі документи як правила, інструкції, норми, державні санітарні правила та ін., якими мають керуватись роботодавці та які регламентують вимоги щодо конструкції електронно-обчислювальної техніки та особливості її розміщення.

На сьогодні основними документами, які регламентують питання охорони праці при використанні працівниками персональних комп’ютерів, є такі підзаконні акти: НПАОП 0.00-7.15-18 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

У відповідності з цими документами, необхідно вжити всіх необхідних заходів з охорони праці та розробити відповідні документи. Приміщення, в яких виконується робота за комп’ютером, повинні відповідати проектній документації будинку, яка погоджена з уповноваженими державними органами. Крім того, мають бути дотримані санітарні нормативи освітлення, вимоги до параметрів мікроклімату (температура, відносна вологість), ступеня і сили вібрації, звукового шуму і вогнестійкості приміщення, а також характеристики електромагнітного, ультрафіолетового та інфрачервоного полів.

Освітлення робочого місця працівника, який виконує роботу з використанням екранних пристроїв має відповідати встановленим вимогам. Відповідно до ДБН В.2.5-28:2018 “Природне і штучне освітлення” приміщення з постійним перебуванням людей повинні мати природне освітлення. Природне освітлення поділяється на бокове, верхнє і комбіноване. Що до штучного освітлення воно поділяється на робоче, аварійне, охоронне і чергове. Мікроклімат приміщень з робочими місцями працівників з екранними пристроями має підтримуватись на постійному рівні та відповідати вимогам Санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99, затверджених постановою Головного державного санітарного лікаря України від 01 грудня 1999 року № 42.

Обладнання та організація робочого місця ВДТ ЕОМ повинна відповідати характеру трудової діяльності і задовольняти вимоги ДСТУ 7299:2013 «Дизайн і ергономіка. Робоче місце оператора. Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки».

Дотримання всіх необхідних вимог з охорони праці забезпечує комфортні умови при виконанні поставлених задач під час та відсутність шкоди для здоров’я, що сприяє підвищенню продуктивності праці.

4.2. Функціональні заходи у сфері державного регулювання та контролю захисту населення і територій.

4.2.1 Державна стандартизація

Стандартизація становить основу проведення державної експертизи, контролю, нагляду, ліцензування видів діяльності, декларування безпеки промислових об'єктів і сертифікації для встановлення норм, правил та характеристик з метою забезпечення:

- безпеки продукції, робіт та послуг для довкілля, життя і здоров'я людей;
- якості продукції, робіт і послуг відповідно до рівня розвитку науки, техніки і технологій;
- єдності вимірів;
- безпеки об'єктів господарювання з урахуванням ризику природних і техногенних катастроф та інших надзвичайних ситуацій.

4.2.1 Державна експертиза

Державна експертиза проектів і рішень щодо об'єктів виробничого та соціального призначення і процесів, що можуть спричинити надзвичайні ситуації та вплинути на стан захисту населення і територій, організується і проводиться відповідно до законодавства спеціально уповноваженими центральними і місцевими органами виконавчої влади, виконавчими органами рад.

У разі потреби експертиза проектів і рішень щодо об'єктів виробничого та соціального призначення, окремих технологічних процесів, які можуть викликати надзвичайні ситуації або вплинути на стан захисту населення і територій від

надзвичайних ситуацій, може проводитися об'єднаннями громадян, незалежними експертами та спеціалістами міжнародних експертних організацій у порядку, встановленому законодавством України.

4.2.2 Державний нагляд і контроль

Державний нагляд і контроль організується з метою перевірки повноти і якості заходів щодо запобігання надзвичайним ситуаціям, забезпечення готовності органів управління, сил і засобів системи захисту населення і територій, посадових осіб до дій у разі виникнення цих ситуацій. Нагляд і контроль полягають у забезпеченні:

- безаварійного функціонування об'єктів економіки та зменшення масштабів можливих збитків у разі виникнення надзвичайних ситуацій природного та техногенного походження;
- готовності органів управління системи захисту населення і територій, підприємств та їх спеціалізованих аварійно-рятувальних формувань до виконання поставлених перед ними завдань;
- охорони довкілля та збереження природних ресурсів, додержання порядку і умов користування надрами з метою запобігання небезпечним екологічним і геологічним процесам;
- готовності технічних засобів і споруд інженерного захисту населення і територій від лісових, торф'яних та інших пожеж, повеней, підтоплень та інших небезпечних дій води;
- фінансових гарантій відшкодувань збитків здоров'ю та життю громадян, захисту майнових інтересів фізичних та юридичних осіб від збитків, спричинених надзвичайними ситуаціями природного та техногенного походження.

Державний нагляд і контроль проводиться згідно із законодавством України спеціально уповноваженими центральними та місцевими органами виконавчої

влади, виконавчими органами рад відповідно до завдань, покладених на систему захисту населення і територій.

4.2.3 Ліцензування окремих видів діяльності

Ліцензування окремих видів діяльності здійснюється з метою проведення єдиної державної політики для забезпечення життєво важливих інтересів громадян, суспільства, держави. Ліцензуванню підлягає діяльність щодо забезпечення промислової, пожежної та транспортної безпеки, охорони довкілля та інших сфер.

4.2.4 Декларування безпеки промислових об'єктів

Декларування безпеки промислових об'єктів здійснюється з метою забезпечення контролю за додержанням заходів безпеки на етапах їх введення в експлуатацію, експлуатації та виводу з експлуатації. Декларування передбачає:

- оцінку ризику виникнення на промислових об'єктах надзвичайних ситуацій з урахуванням визначення джерел загроз, умов розвитку і можливих наслідків надзвичайних ситуацій, у тому числі викидів у довкілля шкідливих речовин;
- оцінку готовності до експлуатації потенційного об'єкта відповідно до вимог промислової безпеки;
- аналіз достатності й ефективності вжитих заходів щодо запобігання, локалізації та ліквідації надзвичайної ситуації на промисловому об'єкті;
- розроблення заходів, спрямованих на зменшення масштабів і величини негативних наслідків від надзвичайних ситуацій.

4.2.5 Сертифікація

Сертифікація організується і здійснюється з метою підтвердження відповідності продукції встановленим вимогам, включаючи контроль небезпечної продукції для життя та здоров'я людей, довкілля та майна. Сертифікація може мати обов'язковий і добровільний характер.

4.2.5 Страхування

Головною метою страхування є забезпечення економічної підтримки заходів щодо запобігання надзвичайним ситуаціям, які здійснюються центральними і місцевими органами виконавчої влади, виконавчими органами рад, підприємствами та організаціями незалежно від форм власності, і страхового покриття збитків у разі їх виникнення. Страхування організується і здійснюється на підставі договору про страхування, який є сукупністю видів страхування, що передбачають обов'язок страховика щодо страхових виплат у розмірі повної або часткової компенсації збитків, завданих об'єкту страхування. Страховий захист населення і територій від надзвичайних ситуацій забезпечується обов'язковим і добровільним страхуванням [26].

4.3. Висновки до розділу 4

У даному розділі було розглянуто питання охорони праці при використанні працівниками персональних комп’ютерів, адже розроблений алгоритм у цій роботі потребує роботи з ними.

У другому підрозділ було описано функціональні заходи у сфері державного регулювання та контролю захисту населення і територій.

ВИСНОВКИ

В результаті написання магістерської роботи було розроблено алгоритм, що підвищує криптостійкість симетричних алгоритмів шифрування та надає додаткову стійкість від атак на основі відкритих текстів, а також створено його програмну реалізацію. В процесі було отримано такі результати:

- Опрацьовано матеріали по типам симетричних алгоритмів шифрування і описано найпоширеніші із них;
- Проаналізовано методи здійснення атак на алгоритми шифрування та способи криптоаналізу для знаходження ключа шифрування даних;
- Обрано та описано приклад слабкого шифру;
- Проаналізовано матеріали по створенню унікальних перестановок, описано основні із них та створено власний метод генерації перестановок, який використано для розробки алгоритму посилення криптостійкості;
- Розроблено схему алгоритму підвищення криптостійкості симетричних алгоритмів шифрування даних та описано деталі його роботи;
- Створено програму для демонстрації роботи алгоритму шифрування на основі даного методу;
- Протестовано створене програмне забезпечення та роз’яснено деталі користування даною програмою;
- Показано та описано результати роботи програми на прикладах шифрування даних.

СПИСОК ЛІТЕРАТУРИ

1. Beth, Thomas; Piper, Fred (1985). The Stop and Go Generator (PDF). EUROCRYPT '84. pp. 88–92. URL: https://link.springer.com/content/pdf/10.1007/3-540-39757-4_9.pdf (дата звернення 1.11.2021)

2. Matt J. B. Robshaw, Stream Ciphers Technical Report TR-701, version 2.0, RSA Laboratories, 1995. URL: http://www.networkdls.com/Articles/tr-701.pdf (дата звернення: 4.11.2021)

3. SVG анімація простого потокового шифру. URL: https://web.archive.org/web/20120326211358/http://l-system.net.pl/crypto/simple_stream_cipher.svg (дата звернення: 3.11.2021)

4. Christof Paar, Jan Pelzl, "Stream Ciphers", Chapter 2 of "Understanding Cryptography, A Textbook for Students and Practitioners", Springer, 2009. URL: https://archive.ph/20121208212741/http://wiki.crypto.rub.de/Buch/movies.php (дата звернення: 2.11.2021)

5. Diffie, Whitfield; Hellman, Martin E. (June 1977). "Exhaustive Cryptanalysis of the NBS Data Encryption Standard". URL: https://web.archive.org/web/20140226205104/http://origin-www.computer.org/csdl/mags/co/1977/06/01646525.pdf (дата звернення: 2.11.2021)

6. Snuffle 2005: the Salsa20 encryption function. URL: http://cr.yp.to/snuffle.html

7. Jeff Moser. A Stick Figure Guide to the Advanced Encryption Standard (AES). URL: https://www.webcitation.org/65Yj0G6mK?url=http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html (дата звернення: 2.11.2021)

8. "Principles and Performance of Cryptographic Algorithms" by Bart Preneel, Vincent Rijmen, and Antoon Bosselaers. URL: https://www.drdobbs.com/algorithm-alley/184410756 (дата звернення: 2.11.2021)

9. Методы и средства защиты информации. URL: http://citforum.ru/internet/infsecure/ (дата звернення: 2.11.2021)

10. Технології захисту інформації: підручник для студ. спеціальності 122 «Комп’ютерні науки», спеціалізацій «Інформаційні технології моніторингу довкілля», «Геометричне моделювання в інформаційних системах» / Ю. А. Тарнавський; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 2,04 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2018. – 162 с.

11. Menezes A. J., Oorschot P. v., Vanstone S. A. Handbook of Applied Cryptography (англ.) — CRC Press, 1996. — 816 p.

12. Шнайер Б. Прикладная криптография, 2-е издание: протоклы, алгоритмы и исходные тексты на языке С. – (перевод с оригинла Applied Cryptography, Second Edition: Protocols, Algorithms and Source Code in C (cloth) Publisher: John Wiley & Sons, Inc. Author(s): Bruce Schneier ISBN: 0471128457 Publication Date: 01/01/96).

13. Bruce Schneier (1993). "Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)". Fast Software Encryption, Cambridge Security Workshop. URL: https://link.springer.com/chapter/10.1007/3-540-58108-1_24 (дата звернення: 2.11.2021)

14. Алгоритмы блочного симметричного шифрования. Алгоритм TEA. URL: https://intellect.icu/algoritmy-blochnogo-simmetrichnogo-shifrovaniya-algoritm-tea-tiny-encryption-algorithm-5741 (дата звернення: 2.11.2021)

15. Дж. Л. Месси. Введение в современную криптологию. // ТИИЭР, т.76, №5, Май 88 – М, Мир, 1988, с.24-42.

16. А. В. Спесивцев и др. Защита информации в персональных компьютерах. – М., Радио и связь. 1992, с.140-149.

17. Hal Tipton and Micki Krause. Handbook of Information Security Management – CRC Press LLC, 1998.

18. Генерация перестановок. URL: http://zonakoda.ru/generaciya-perestanovok.html (дата звернення: 2.11.2021)

19. PureBasic — The Perfect Cross-Platform & Native Development Language. URL: https://www.codeproject.com/Articles/853831/PureBasic-The-Perfect-Cross-Platform-Native-Develo (дата звернення: 2.11.2021)

20. Алгоритм: Как найти следующую лексикографическую перестановку. URL: https://habr.com/ru/post/428552/ (дата звернення: 2.11.2021)

21. Knuth, D. E. The Art of Computer Programming. — Addison-Wesley, 2005. — Vol. 4.12. URL: https://www-cs-faculty.stanford.edu/~knuth/taocp.html (дата звернення: 2.11.2021)

22. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин. Офіційний вебпортал парламенту України. URL: https://zakon.rada.gov.ua/rada/show/v0007282-98#Text (дата звернення: 17.11.2021).

23. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. Офіційний вебпортал парламенту України. URL: https://zakon.rada.gov.ua/laws/show/z0508-18#Text (дата звернення: 7.10.2021).

24. Про охорону праці: Закон України від № 49, ст.668. Офіційний вебпортал парламенту України. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 17.11.2021)

25. В. Жельников. Криптография от папируса до компьютера. – М., ABF, 1996.

26. Про Концепцію захисту населення і територій у разі загрози та виникнення надзвичайних ситуацій: Указ Президента України від 26.03.1999 №284/99- Л. Кучма. Київ: АПУ, 1999.

27. Б. Семеген, С. Лупенко. Актуальність розробки методів підвищення криптостійкості слабких алгоритмів шифрування. Матеріали IX Науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі, системи та технології» (8–9 грудня 2021 року). Тернопіль: ТНТУ, 2021. С. 127.

28. Б. Семеген, С. Лупенко. Метод підвищення криптостійкості симетричних алгоритмів шифрування. Матеріали IX Науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі, системи та технології» (8–9 грудня 2021 року). Тернопіль: ТНТУ, 2021. С. 128.

Додаток А

Тези конференції

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

ІХ НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

8–9 грудня 2021 року

ТЕРНОПІЛЬ
2021

УДК 004.056.55:004.77:004.42
Б. Семеген, С. Лупенко д.т.н., проф.
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

АКТУАЛЬНІСТЬ РОЗРОБКИ МЕТОДІВ ПІДВИЩЕННЯ КРИПТОСТІЙКОСТІ СЛАБКИХ АЛГОРИТМІВ ШИФРУВАННЯ

UDC 004.056.55:004.77:004.42
B. Semehen, S. Lupenko

ACTUALITY OF DEVELOPMENT OF METHODS OF INCREASING CRYPTIC RESISTANCE OF WEAK ENCRYPTION ALGORITHMS

На даний час існує велика кількість методів криптоаналізу, які дозволяють зробити вразливими багато алгоритмів шифрування даних. Це становить загрозу конфіденційності інформації. Для збереження високого рівня захисту даних постійно розробляються все нові алгоритми шифрування із врахуванням і виправленням уразливостей, які були знайдені у попередніх алгоритмів. У сучасних комп'ютерних системах автоматизації будинків, пультів керуванням різними пристроями і інших засобах керування і передачі даних продовжують використовуватися слабкі алгоритми шифрування у зв'язку із їх простотою і оптимальністю алгоритму для забезпечення високої швидкодії у малопотужних пристроях для яких це критично важливо. Тому створення нових простих алгоритмів шифрування, або ж покращення стійкості попередніх простих алгоритмів є важливим у розвитку сучасних технологій із спрощеною архітектурою.

Для криптоаналізу поширеними є чотири типи атак на основі: тільки шифротексту, відкритого тексту, підібраного відкритого тексту, адаптивно підібраного відкритого тексту [1]. Три типи криптоаналізу із даних переліку основані на відкритому тексті кожен наступний із яких дає ширші можливості для криптоаналітика.

У зв'язку із тим, що багато шифрованих каналів передачі даних пропускають через себе дані із різних джерел, зокрема відкритих, тому виникає можливість зловмисникові виконувати передачу відомих для нього даних. Стає можливим виконувати криптоаналіз не тільки на основі відкритого тексту, а й на основі адаптивно підібраного відкритого тексту. Для каналу передачі даних який використовує слабкий алгоритм шифрування це становить особливу небезпеку.

В результаті важливою задачею для безпеки сучасних комп'ютерних систем є збільшення криптостійкості алгоритмів шифрування використовуючи методи які є ефективними по ресурсам і відповідно мають високу швидкодію. Для забезпечення підвищеної складності криптоаналізу на основі відкритого тексту важливим є добавлення певної невідомої складової до відкритого тексту, це добавляє невизначеності у шифрування і зникає відповідність між відкритим текстом та шифротекстом. Тому пропонується вводити додаткове визначене перемішування даних для кожного блоку, яке буде різним для кожного наступного блоку даних. Одним із таких є фрагментування даних на менші блоки і виконання над ними перестановки.

Література.

1. Шнайер Б. Прикладная криптография, 2-е издание: протоклы, алгоритмы и исходные тексты на языке С. – (перевод соригинла Applied Cryptography, Second Edition: Protocols, Algorithms and Source Code in C (cloth) Publisher: John Wiley & Sons, Inc. Author(s): Bruce Schneier ISBN: 0471128457 Publication Date: 01/01/96).
2. Кнут, Дональд Эрвин. Искусство программирования, том 4, выпуск 2. Генерация всех кортежей и перестановок.: Пер. с английского - М.: ООО «И.Д. Вильямс», 2008. - 160 с.: ил. - Парал. тит. англ.

УДК 004.056.55:004.77:004.42
Б. Семеген, В. Семеген, С. Лупенко д.т.н., проф.
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

МЕТОД ПІДВИЩЕННЯ КРИПТОСТІЙКОСТІ СИМЕТРИЧНИХ АЛГОРИТМІВ ШИФРУВАННЯ

UDC 004.056.55:004.77:004.42
B. Semehen, V. Semehen, S. Lupenko Prof.

METHODS OF INCREASING SYMMETRIC ENCRYPTION ALGORITHMS' CRYPTOSECURITY

Однією із найбільш поширених атак на алгоритми шифрування є атака відкритим текстом, за умови її здійсненності із подальшим криптоаналізом шифротексту для знаходження секретного ключа даного каналу передачі даних. Тому для забезпечення підвищеної криптостійкості до такого виду атак пропонується використовувати перемішування даних. У цьому разі, відкритий текст матиме подібні характеристики до невідомих даних із точки зору криптоаналізу.

У цьому методі виконується розбиття даних на підблоки і їх пересортування за спеціальним алгоритмом по наперед встановленій їх кількості, шляхом задання певного числа. Число, яке вказується для пересортування підблоків для першого блоку, генерується із гешу ключа, а для усіх наступних блоків це число генерується псевдовипадковим алгоритмом і розташовується вкінці даних попереднього блоку.

Запропонований алгоритм буде надлишковим, але підвищуватиме криптостійкість слабких алгоритмів шифрування при подачі даних опрацьованих вказаним чином.

Ускладнення атаки відкритим текстом відбувається через невизначеність позиції фрагментів відкритих даних, над якими була виконана перестановка і також, шляхом добавлення вкінці блоку певного випадкового числа, яке робить невизначеним шифротекст для одних і тих же відкритих даних блоку при їхній повторній передачі.

Зашифрований текст дешифрується у зворотному порядку, де кінцевим етапом є розстановка підблоків у правильному порядку за вказаним числом по даному алгоритму перестановки.

Описані перетворення над даними не спричинять високого навантаження на комп'ютерну систему і дозволять підвищити криптостійкість алгоритмів шифрування без втручання безпосередньо у самі алгоритми. Перетворення над даними виконується перед подачею даних у алгоритм шифрування.

Використаний метод є простим із точки зору логіки його роботи і, відповідно, програмної реалізації, тому може використовуватись як для більш потужних комп'ютерних систем, так і для малопотужних комп'ютерів у приладах низького енергоспоживання.

Література.

1. Шнайер Б. Прикладная криптография, 2-е издание: протоклы, алгоритмы и исходные тексты на языке С. – (перевод соригинла Applied Cryptography, Second Edition: Protocols, Algorithms and Source Code in C (cloth) Publisher: John Wiley & Sons, Inc. Author(s): Bruce Schneier ISBN: 0471128457 Publication Date: 01/01/96).
2. Кнут, Дональд Эрвин. Искусство программирования, том 4, выпуск 2. Генерация всех кортежей и перестановок.: Пер. с английского - М.: ООО «И.Д. Вильямс», 2008. - 160 с.: ил. - Парал. тит. англ.
3. R.J. Anderson, "Searching for the Optimum Correlation Attack, " K. U. Leuven Workshop on Cryptographic Algorithms, Springer-Verlag, 1995. to appear.

Додаток Б

Код програмного забезпечення методу підвищення криптостійкості симетричних алгоритмів шифрування даних

```
;Not unicode

IncludeFile "IncludeFile_Window.pb"

Procedure.l fl(n.l)
  res.l = 1
  For i=2 To n
    res*i
  Next
  ProcedureReturn res
EndProcedure

Procedure.l Random32()
  n.l = 2<<16 - 1
  ProcedureReturn Random(n)*256 + Random(n)
EndProcedure



;------ Визначення змінних ------
PathToSourceFile.s = ""          ;"C:\Users\Vitaliy\Desktop\Message.txt"
PathToDestinationFile.s = ""     ;"C:\Users\Vitaliy\Desktop\EncriptedMessage.txt"
passwordString.s = ""            ;"Password string to encrypt a message"


;12 блоків по 4 байти, 4 байти номер перестановки і 3 блока випадкових даних по 4 байти для суми по модулю два
n.l = 12
fl_n.l = fl(n)
```

```
PermutationNumber.l = 0

pointerMessageString.l = 0
pointerResMessageMemory.l = 0
;-------------------------------


Open_Window_0()

Repeat
  Event = WindowEvent()
  If Event = #PB_Event_Gadget
    EventGadget = EventGadget()
    Select EventGadget
      Case #Button_0
        PathToSourceFile = OpenFileRequester("Виберіть файл", "", "Text (*.txt)|*.txt;*.bat|All files (*.*)|*.*", 0)
        SetGadgetText(#String_0, PathToSourceFile)
      Case #Button_1
        PathToDestinationFile = OpenFileRequester("Виберіть файл", "", "Text (*.txt)|*.txt;*.bat|All files (*.*)|*.*", 0)
        SetGadgetText(#String_1, PathToDestinationFile)
      Default
        PathToSourceFile = GetGadgetText(#String_0)
        PathToDestinationFile = GetGadgetText(#String_1)
        passwordString = GetGadgetText(#String_2)
        Select EventGadget
          Case #Button_2
            CALL l_encrypteddata
          Case #Button_3
            CALL l_decrypteddata
        EndSelect
    EndSelect
  EndIf
```

```
Until Event = #PB_Event_CloseWindow

End


EncryptedData:
;-----------------------------------------------
If OpenFile(0, PathToSourceFile)
lenMessageString.l = Lof(0)
MessageString.l = AllocateMemory(lenMessageString)

nBlocks = Round(lenMessageString/48, #PB_Round_Up)
lenResMesageMem.l = nBlocks*64
resMessageMemory = AllocateMemory(lenResMesageMem)

ReadData(0, MessageString, lenResMesageMem)
CloseFile(0)
;-----------------------------------------------



;{ Обчислення номеру першої перестановки
passFPHex.s = MD5Fingerprint(@passwordString, Len(passwordString));
4bfef90f3c9a8a38c4b84f0e2a08091c


fPN_formig.q = 0
;{ 16 байт md5 перетворюються діляться на дві частини по 8 байт і занаходиться їхня сума по модулю два
Hex2Dec(Mid(passFPHex, 1,8))
MOV dword [v_fPN_formig + 4], eax
Hex2Dec(Mid(passFPHex, 9,8))
MOV dword [v_fPN_formig], eax
```

```
Hex2Dec(Mid(passFPHex, 17,8))
XOR dword [v_fPN_formig + 4], eax
Hex2Dec(Mid(passFPHex, 25,8))
XOR dword [v_fPN_formig], eax          ;-8122604762126187740
;}

MOV eax, dword [v_fPN_formig]
MUL dword [v_fl_n]
MOV ecx, edx
MOV eax, dword [v_fPN_formig + 4]
MUL dword [v_fl_n]
ADD eax, ecx
JNC l_notaddcarry
INC edx
notAddCarry:
MOV [v_PermutationNumber], edx
;}


MOV eax, [v_MessageString]
; MOV eax, [eax];
MOV [v_pointerMessageString], eax
MOV eax, [v_resMessageMemory]
; MOV eax, [eax]
MOV [v_pointerResMessageMemory], eax


inBlocks = lenMessageString
SUB [v_inBlocks], 48
JL l_endcycleblocksprocessing
BeginCycleBlocksProcessing:


MOV ecx, 48
```

```
LastBlockProcessing:

MOV esi, [v_pointerMessageString]
ADD dword [v_pointerMessageString], 48
MOV edi, [v_pointerResMessageMemory]
ADD dword [v_pointerResMessageMemory], 48

REP MOVSB


NextPermutationNumber = Random(fl_n)

MOV edi, [v_pointerResMessageMemory]
MOV [edi], eax
ADD dword [v_pointerResMessageMemory], 4

; Додавання випадкових даних
Random(2<<32-1)
MOV edi, [v_pointerResMessageMemory]
ADD dword [v_pointerResMessageMemory], 4
MOV [edi], eax
Random(2<<32-1)
MOV edi, [v_pointerResMessageMemory]
ADD dword [v_pointerResMessageMemory], 4
MOV [edi], eax
Random(2<<32-1)
MOV edi, [v_pointerResMessageMemory]
ADD dword [v_pointerResMessageMemory], 4; Зсувв вказівки на 4 байти
MOV [edi], eax
;!
;!edi
;Сума по модулю два
; MOV esi, edi
```

```
MOV esi, [v_pointerResMessageMemory]
SUB edi, 60; всіх даних у блоці 64 байти останні 4 були записані вище тому від вказівки потрібно відняти 60 байт щоб перейти на початок
MOV edx, 3; Кількість повторень циклу XorCycle
XorCycle:
MOV ecx, 4; Кількість повторень циклу XorSmallCycle
SUB esi, 16
XorSmallCycle:
LODSD; MOV eax, [esi]
XOR eax, [edi]  ;MOV eax, [edi]
STOSD
LOOP l_xorsmallcycle
DEC edx
JG l_xorcycle

SUB esi, 64



PUSH ebx
MOV ecx, dword [v_n]
; DEC ecx
MOV eax, dword [v_PermutationNumber]
PermutationCycle:
XOR edx, edx; Якщо не обнулити, то команда DIV сповістить про переповнення цилого числа
DIV ecx
MOV ebx, [esi]
XCHG dword [edx*4+esi], ebx
MOV dword [esi], ebx
ADD esi, 4
LOOP l_permutationcycle
POP ebx

PermutationNumber = NextPermutationNumber
```

```
SUB [v_inBlocks], 48
JGE l_begincycleblocksprocessing
EndCycleBlocksProcessing:

MOV ecx, [v_inBlocks]
ADD ecx, 48
JG l_lastblockprocessing

; OpenFile(0, "C:\Users\Vitaliy\Desktop\PermutationMessage.txt")
; WriteData(0, resMessageMemory, lenResMesageMem)
; CloseFile(0)

AESEncoder(resMessageMemory, resMessageMemory, lenResMesageMem,
@passwordString, 256, 0, #PB_Cipher_ECB)

If OpenFile(0, PathToDestinationFile)
WriteData(0, resMessageMemory, lenResMesageMem)
CloseFile(0)

EndIf
EndIf

RET


DecryptedData:

If OpenFile(0, PathToSourceFile)
lenEncriptedMessage.l = Lof(0)

encriptedMessage.l = AllocateMemory(lenEncriptedMessage)

lenRecoveredMessage = lenEncriptedMessage*48/64
```

```
recoveredMessage.l = AllocateMemory(lenRecoveredMessage)

ReadData(0, encriptedMessage, lenEncriptedMessage)
CloseFile(0)


AESDecoder(encriptedMessage, encriptedMessage, lenEncriptedMessage, @passwordString, 256, 0, #PB_Cipher_ECB)


pointerEncriptedMessage.l = 0
pointerRecoveredMessage.l = 0

;{ Обчислення номеру першої перестановки
passFPHex.s = MD5Fingerprint(@passwordString, Len(passwordString))

fPN_formig.q = 0
;{ 16 байт md5 перетворюються діляться на дві частини по 8 байт і занаходиться їхня сума по модулю два
Hex2Dec(Mid(passFPHex, 1,8))
MOV dword [v_fPN_formig + 4], eax
Hex2Dec(Mid(passFPHex, 9,8))
MOV dword [v_fPN_formig], eax

Hex2Dec(Mid(passFPHex, 17,8))
XOR dword [v_fPN_formig + 4], eax
Hex2Dec(Mid(passFPHex, 25,8))
XOR dword [v_fPN_formig], eax
;}

MOV eax, dword [v_fPN_formig]
MUL dword [v_fl_n]
MOV ecx, edx
MOV eax, dword [v_fPN_formig + 4]
```

```
MUL dword [v_fl_n]
ADD eax, ecx
JNC l_notaddcarry_decrypt
INC edx
notAddCarry_decrypt:
MOV [v_PermutationNumber], edx
;}

MOV eax, [v_encriptedMessage]
; MOV eax, [eax]
MOV [v_pointerEncriptedMessage], eax

MOV eax, [v_recoveredMessage]
; MOV eax, [eax]
MOV [v_pointerRecoveredMessage], eax


inBlocks = lenEncriptedMessage/64

MOV esi, [v_pointerEncriptedMessage]
ADD esi, (48-4)
DEC [v_inBlocks]
JL l_endcycleblocksprocessing_decrypt
BeginCycleBlocksProcessing_decrypt:

PUSH ebx
MOV ecx, 1
MOV eax, dword [v_fl_n]
MOV edi, dword [v_PermutationNumber]

PermutationCycle_decrypt:
CMP ecx, dword [v_n]
JG l_endpermutationcycle
XOR edx, edx; В іншому випадку DIV сповіщить про переповнення цилого числа
```

```
DIV ecx
XCHG edi, eax
XOR edx, edx; В іншому випадку DIV сповіщить про переповнення цилого числа
DIV edi
MOV ebx, [esi]
XCHG [eax*4+esi], ebx
MOV [esi], ebx
SUB esi, 4
MOV eax, edi
MOV edi, edx
INC ecx
JNO l_permutationcycle_decrypt
EndPermutationCycle:

POP ebx

ADD esi, 4
; MOV esi, [v_pointerEncriptedMessage]
MOV edi, [v_pointerRecoveredMessage]
MOV edx, -48
SUB esi, edx;ADD esi, 64
XorCycle_decrypt:
MOV ecx, 4
; CallDebugger
XorSmallCycle_decrypt:
LODSD; MOV eax, [esi]
XOR eax, [esi+edx-4]
STOSD
LOOP l_xorsmallcycle_decrypt
SUB esi, 16
ADD edx, 16
JAE l_xorcycle_decrypt
```

```
MOV eax, [esi]; [v_encriptedMessage+48];
MOV [v_PermutationNumber], eax    ;   PermutationNumber = NextPermutationNumber
ADD esi, 16+(48-4)
; ADD esi, 16;+
; MOV [v_pointerEncriptedMessage], esi
MOV [v_pointerRecoveredMessage], edi

DEC [v_inBlocks]
JGE l_begincycleblocksprocessing_decrypt
EndCycleBlocksProcessing_decrypt:

If CreateFile(0, PathToDestinationFile)
; WriteData(0, encriptedMessage, lenEncriptedMessage)
WriteData(0, recoveredMessage, lenRecoveredMessage)
CloseFile(0)
EndIf
EndIf

RET

;1625218383 - Completed core code
;1640036765-1640042863 - Implement interface
```