ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка алгоритму шифрування зображень на основі зміщення байтів по код-паролю»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
(код, найменування спеціальності)
освітньо-професійної програми «Інженерія програмного забезпечення»
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Андрій СКУРАТОВСЬКИЙ
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-64
Андрій СКУРАТОВСЬКИЙ

Керівник: Олесь ДІБРІВНИЙ
доктор філософії (PhD)

Рецензент: _____
науковий ступінь, вчене звання *Ім'я, ПРІЗВИЩЕ*

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Скуратовському Андрію Володимировичу

1. Тема кваліфікаційної роботи: Розробка алгоритму шифрування зображень на основі зміщення байтів по код-паролю

керівник кваліфікаційної роботи Олесь ДІБРІВНИЙ доктор філософії (PhD),

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» жовтня 2023 р. №145.

2. Строк подання кваліфікаційної роботи «29» грудня 2023р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, алгоритм симетричного шифрування.

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

1. Дослідження принципів роботи алгоритмів шифрування
2. Аналіз існуючих алгоритмів шифрування та захисту інформації
3. Розробка алгоритму шифрування зображень

5. Перелік графічного матеріалу: *презентація*

1. Мета, об’єкта та предмет дослідження
2. Порівняння існуючих симетричних алгоритмів
3. Процес взаємодії користувача
4. Процес обробки зображення
5. Модифікований процес взаємодії користувача
6. Модифікований процес обробки зображення
7. Алгоритм шифрування зображень
8. Приклад шифрування зображення по код-паролю
9. Розрахунок якості алгоритму
10. Розрахунок якості алгоритму
11. Порівняльний аналіз швидкості алгоритмів
12. Порівняльний аналіз алгоритмів за показником використання пам’яті
13. Порівняльний аналіз ефективності алгоритмів
14. Висновки
15. Публікації та апробація роботи

6. Дата видачі завдання«19» жовтня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	19.10-05.11.23	
2	Вивчення матеріалів для аналізу існуючих алгоритмів шифрування та захисту інформації	06.11-12.11.23	
3	Дослідження криптографії та захисту інформації	13.11-19.11.23	
4	Аналіз симетричних алгоритмів шифрування та їх особливості	20.11-26.11.23	
5	Створення та розробка алгоритму шифрування	27.11-03.12.23	
6	Застосування розробленого алгоритму	04.12-10.12.23	
7	Оформлення роботи: вступ, висновки, реферат	11.12-20.12.23	
8	Розробка демонстраційних матеріалів	21.12-29.12.23	

Здобувач вищої освіти _____ Андрій СКУРАТОВСЬКИЙ
(підпис)

Керівник
кваліфікаційної роботи _____ Олесь ДІБРІВНИЙ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 77 стор., 8 табл., 21 рис., 13 джерел.

Мета роботи – підвищення ефективності та простоти шифрування зображень за рахунок розробки симетричного алгоритму шифрування байтів.

Об'єкт дослідження – процес симетричного шифрування байтів.

Предмет дослідження – методи та підходи до реалізації алгоритмів симетричного шифрування даних.

Короткий зміст роботи: Ця дослідницька робота заглиблюється в шифрування даних та інформаційну безпеку, зосереджуючись на симетричних алгоритмах шифрування, де один ключ використовується як для шифрування, так і для дешифрування. У ній ретельно розглядаються існуючі методи симетричного шифрування, оцінюються їхні сильні та слабкі сторони в різних контекстах.

У роботі також представлено новий алгоритм шифрування, розроблений для підвищення ефективності обробки та використання пам'яті, зокрема, для шифрування великих даних, таких як зображення високої роздільної здатності. Цей алгоритм, розроблений з використанням механізму код-пароль, має на меті спростити взаємодію користувача з шифрувальними сервісами, роблячи його доступним для користувачів незалежно від їхніх технічних знань.

Значний акцент у цьому алгоритмі зроблено на дружньому до користувача аспекті, що забезпечує безпечне та просте шифрування зображень. Розробка та ефективність нового алгоритму підтверджена ретельним тестуванням і порівняльним аналізом із встановленими стандартами симетричного шифрування, що демонструє його потенціал перевищити ефективність існуючих рішень у сфері шифрування даних.

КЛЮЧОВІ СЛОВА: ШИФРУВАННЯ, СИМЕТРИЧНИЙ АЛГОРИТМ, ЗСУВ БАЙТІВ, XOR ОПЕРАЦІЇ, ЗАХИСТ ДАНИХ.

ABSTRACT

The text part of the qualification work for the master's degree: 77 pages, 8 table, 21 figures, 13 sources.

The purpose of the work is to increase the efficiency and simplicity of image encryption by developing a symmetric byte encryption algorithm.

Object of research is the process of symmetric byte encryption.

Subject of research are methods and approaches to the implementation of symmetric data encryption algorithms.

Summary of the work: The presented research paper delves into the intricate realm of data encryption and information security, offering a comprehensive analysis of the underlying principles that govern these crucial areas. The focus primarily lies in the exploration of symmetric encryption algorithms, where the same key is employed for both encryption and decryption processes. This exploration involves a thorough examination of existing symmetric encryption methodologies, assessing their strengths, vulnerabilities, and applicability in various scenarios.

In addition to the analysis, the paper introduces a novel encryption algorithm, meticulously designed via a code-password mechanism symmetric encryption. The development of this new algorithm is not merely an academic exercise but is driven by practical considerations, particularly the need to enhance the efficiency of the encryption process. By optimizing encryption speed and memory consumption, the algorithm addresses one of the significant challenges in cryptographic practices, especially in contexts where large volumes of data, such as high-resolution images, require secure handling.

Furthermore, the paper places a strong emphasis on user interaction with encryption services. Recognizing that the complexity of cryptographic systems can often be a barrier to widespread adoption, the new algorithm is crafted with a user-centric approach. This approach simplifies the end-user experience without

compromising the integrity and security of the encryption process. The algorithm's integration into an image encryption service is meticulously engineered to ensure that users, irrespective of their technical expertise, can securely encrypt their images with ease and confidence.

The development process of this new algorithm is underpinned by rigorous testing and benchmarking against existing popular symmetric algorithms. This comparative analysis ensures that the new system potentially exceeds efficient parameters of current encryption solutions.

KEYWORDS: ENCRYPTION, SYMMETRIC ALGORITHM, BYTE SHIFT, XOR OPERATIONS, DATA PROTECTION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

XOR	–	eXclusive OR (виключна диз’юнкція)
AES	–	Advanced Encryption Standard
DES	–	Data Encryption Standard
3DES	–	Triple DES
DLP	–	Data Loss Prevention
VPN	–	Virtual Private Network

ВСТУП

У сучасному світі інформація є одним з найважливіших ресурсів. Вона може бути використана як для позитивних, так і для негативних цілей. Тому захист інформації є однією з найважливіших задач.

Одним з поширених способів захисту інформації є шифрування. Шифрування – це процес перетворення інформації в нерозбірливий вид, який називається шифротекстом. Для розшифрування зашифрованого тексту необхідний ключ, який відомий лише тим, хто має право доступу до інформації.

Застосування шифрування є найважливішим аспектом користування сайтами, сервісами і іншими додатками. Через відсутність захисту даних користувачів є загрози витоку даних у світ, або цією відсутністю може скористатися зловмисник. Саме тому все частіше виникають нові ідеї та варіації різних типів шифрування, задля забезпечення різноманіття та ускладнення захисту даних. Але не кожен алгоритм шифрування є зручним та ефективним з точки зору як кінцевого користувача продуктом, так і технічним спеціалістом який використовує той чи інший алгоритм.

У кожного алгоритму є свої недоліки та переваги, саме тому завдання розробки нового алгоритму, що є ефективним та зручним є актуальним та сучасним.

Мета роботи – підвищення ефективності та простоти шифрування зображень за рахунок розробки симетричного алгоритму шифрування байтів.

Об’єкт дослідження – процес симетричного шифрування байтів.

Предмет дослідження – методи та підходи до реалізації алгоритмів симетричного шифрування даних.

Методи дослідження – методи теорії захисту даних, методи теорії інформації, методи проектування та розробки програмного забезпечення, технології об’єктно-орієнтованого програмування.

Практична значущість результатів полягає у використанні розробленого алгоритму для підвищення ефективності роботи систем шифрування зображень та спрощення взаємодії кінцевого користувача з системою, аналіз впровадження нового алгоритму шифрування в системи зберігання зображень.

Для досягнення мети вирішувалися наступні завдання:

1. Аналіз існуючих методів та підходів до захисту інформації;
2. Розробка симетричного алгоритму шифрування даних;
3. Розробка програмного забезпечення для проведення тестувань алгоритму;
4. Проведення тестування та аналізу симетричного алгоритму шифрування зображень, збір статистичних даних та порівняння ефективності й зручності з іншими алгоритмами;

1 АНАЛІЗ МЕТОДІВ ТА ПІДХОДІВ ДО ЗАХИСТУ ІНФОРМАЦІЇ

1.1 Загальна класифікація методів та підходів до захисту інформації

На даний час існує велика потреба в захисті інформації, як від зловмисників так і від звичайних ситуацій з витоком даних через халатність працівників або програмні помилки. Тому все частіше люди прибігають до застосування різних методів, підходів та засобів по збереженню цілісності даних, власних чи корпоративних, мережевих чи фізичних. Кожен з видів даних потребує свого засобу/методу захисту, або ж їх потрібно комбінувати між собою.

Існує чимала кількість методів, засобів та підходів до захисту інформації, які нерідко комбінують між собою, утворюючи вищий рівень захисту, більш складні лабіринти з перепонами на шляху до інформації для зловмисників.

Одним з прикладів можна представити звичайну серверну кімнату, де стоїть фізичний сервер у вигляді комп’ютерної машини з можливістю фізичного доступу та підключенням до внутрішньої системи компанії. Тут можна скомбінувати як всі варіанти захисту комплексно, так і окремо кожен. Фізичний захист самого сервера за допомогою код-панелей, паролів, ключів доступу. З організаційного боку проінструктувати персонал будівлі про критичну важливість приміщення та видати відповідні розпорядження про права та можливості доступу до сервера. Також майже в 100% випадків використовують мережевий захист з резервним копіюванням, як на фізичні, так і на хмарні пристрої. [13]

Загальні секції представлені на рисунку 1.1:

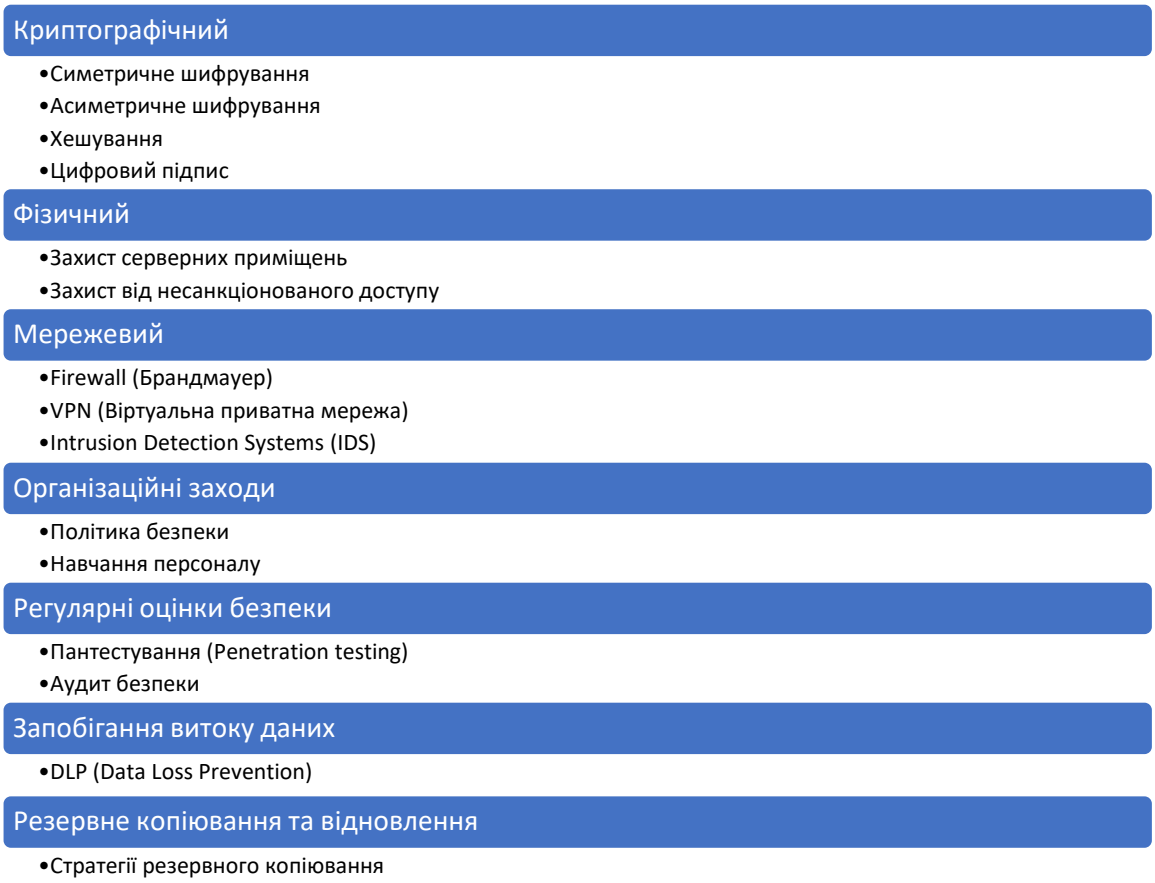


Рис. 1.1 Загальна класифікація методів та підходів до захисту інформації

Тож існує 7 загальних секцій на які можна поділити методи/заходи/підходи до захисту інформації.

1.2 Криптографічний захист

Криптографічний захист є фундаментальною складовою у системах захисту інформації. Він містить різні методи та алгоритми для забезпечення конфіденційності, цілісності, автентичності та невідтворюваності інформації. Основні елементи криптографічного захисту включають:

1.2.1 Симетричне шифрування

Симетричне шифрування – це метод шифрування, при якому для шифрування та розшифрування даних використовується один і той же ключ (рис.

1.2). Цей ключ повинен бути відомий лише тим, хто має право доступу до інформації.

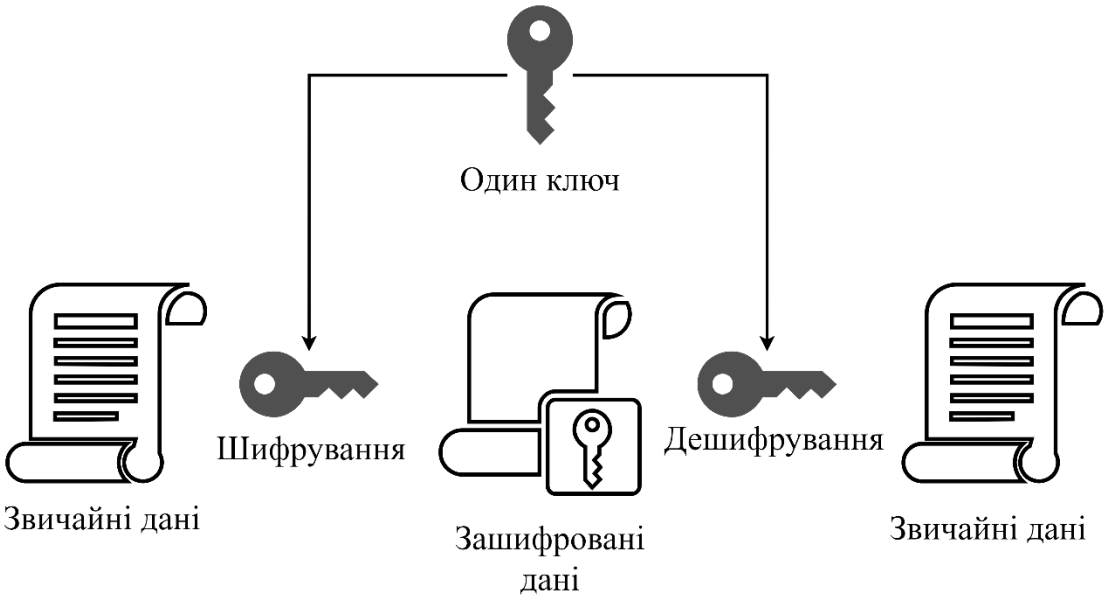


Рис. 1.2 Симетричне шифрування

Симетричне шифрування є більш швидким і ефективним, ніж асиметричне шифрування, але воно вимагає, щоб ключ був переданий стороні, яка отримує зашифровані дані.[6]

Приклади симетричних алгоритмів шифрування:

- DES (Data Encryption Standard), 3DES;
- AES (Advanced Encryption Standard);
- RC2, RC4;
- Blowfish;

1.2.2 Асиметричне шифрування

Асиметричне шифрування – це метод шифрування, при якому для шифрування та розшифрування даних використовуються два ключі: відкритий (публічний) ключ і закритий (приватний) ключ (рис. 1.3). Відкритий ключ може бути переданий будь-кому, а закритий ключ зберігається в таємниці.

Для шифрування даних використовується відкритий ключ, а для розшифрування даних - закритий ключ.

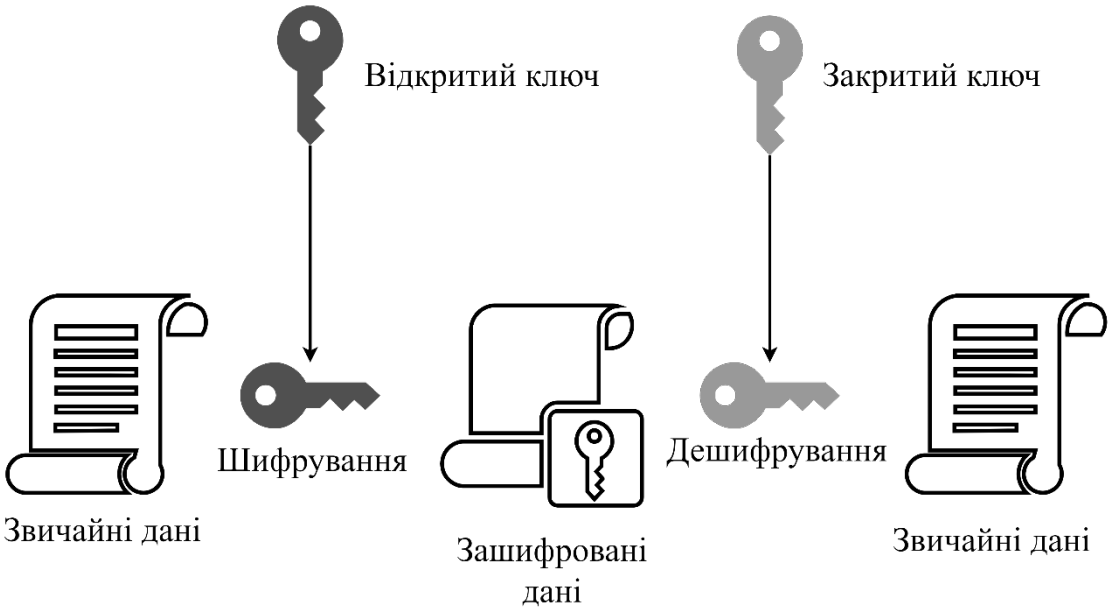


Рис. 1.3 Асиметричне шифрування

Асиметричне шифрування більш безпечне, ніж симетричне шифрування, оскільки відкритий ключ не може бути використаний для розшифрування даних без закритого ключа. Однак воно також є менш ефективним, ніж симетричне шифрування. [11]

Приклади асиметричних алгоритмів шифрування:

- RSA;
- ElGamal;
- Diffie-Hellman;

1.2.3 Хешування

Хешування - це метод перетворення інформації в короткий хеш-код, який неможливо відновити до початкового значення. Хеш-коди можуть використовуватися для забезпечення цілісності даних, а також для аутентифікації користувачів.

Принцип роботи хешування полягає в тому, що дані перетворюються в хеш-код за допомогою деякої математичної функції (рис. 1.4). [7, с. 279-301]

Рис. 1.4 Хешування

Цей метод має такі властивості:

- Для будь-якого набору даних, хеш-код є унікальним;
- Неможливо відновити початкові дані з хеш-коду;

Приклади хеш-функцій:

- MD5;
- SHA-1;
- SHA-256;

1.2.4 Цифровий підпис

Цифровий підпис – це електронний аналог звичайного підпису, який використовується для підтвердження авторства та цілісності документа. Цифровий підпис створюється за допомогою асиметричного шифрування та сертифікату отримувача.

Для створення цифрового підпису автор документа використовує свій закритий ключ і сертифікат отримувача. Цей ключ використовуються для шифрування деяких даних документа, або повністю всього документа. Після чого документ підписується сертифікатом отримувача.

Для перевірки цифрового підпису використовується відкритий ключ автора документа. Цей ключ використовується для розшифрування даних, які були

зашифровані за допомогою закритого ключа автора і потім йде порівняння документу за допомогою сертифікату отримувача.

Якщо дані, які були розшифровані за допомогою відкритого ключа автора, збігаються з даними документа та контрольна сума (хеш сума) збігається, то це означає, що документ був підписаний автором і не був змінений з моменту підписання (рис. 1.5). [11]

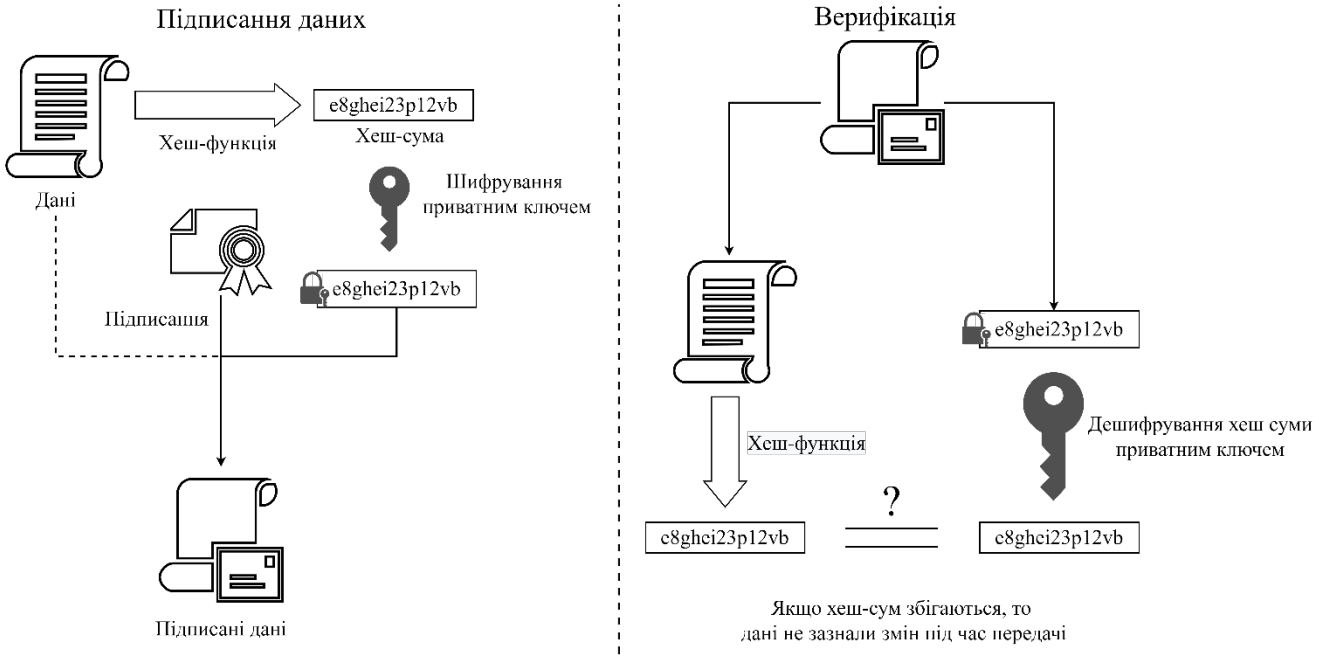


Рис. 1.5 Цифровий підпис

Цифрові підписи використовуються в багатьох сферах, включаючи електронний документообіг, електронну торгівлю та електронну пошту.

1.3 Фізичний захист

Фізичний захист - це комплекс заходів, спрямованих на захист інформації від несанкціонованого доступу, спотворення або знищення за допомогою фізичних засобів. Фізичний захист є одним з найважливіших складових комплексної системи захисту інформації.

1.3.1 Захист серверних приміщень

Серверна кімната - це приміщення, в якому розміщуються сервери та інше обладнання, яке використовується для зберігання та обробки інформації. Серверна кімната повинна бути захищена від несанкціонованого доступу, спотворення або знищення інформації, яка зберігається та обробляється в ній.

До заходів фізичного захисту серверних приміщень відносяться:

– Контроль доступу: обмеження доступу до серверної кімнати лише уповноваженим особам. Контроль доступу може здійснюватися за допомогою фізичних бар'єрів (двері, замки, турнікети), електронних систем контролю доступу (СКД), а також за допомогою біометричних даних (відбитки пальців, обличчя, голос);

– Охоронна сигналізація: виявлення несанкціонованого вторгнення в серверну кімнату. Охоронна сигналізація може бути сповіщальною або сповіщально-охоронною. Сповіщальна сигналізація просто сповіщає про несанкціонований вторгнення, а сповіщально-охоронна сигналізація також дає команду на відключення обладнання або на інші охоронні заходи;

– Пожежна сигналізація: виявлення пожежі в серверній кімнаті. Пожежна сигналізація повинна бути спроектована таким чином, щоб вона могла швидко і точно виявити пожежу та сповістити про неї;

– Відеоконтроль: відеоспостереження за серверною кімнатою. Відеоспостереження може використовуватися для фіксації несанкціонованого вторгнення в серверну кімнату, а також для виявлення злочинів, які були вчинені в серверній кімнаті;

– Доступ до електроживлення: обмеження доступу до електроживлення серверної кімнати. Доступ до електроживлення серверної кімнати повинен бути обмежений лише уповноваженим особам, які відповідають за обслуговування обладнання;

– Доступ до повітря: обмеження доступу до повітря в серверній кімнаті. Доступ до повітря в серверній кімнаті повинен бути обмежений лише уповноваженим особам, які відповідають за обслуговування обладнання;

1.3.2 Захист від несанкціонованого доступу

Захист від несанкціонованого доступу - це комплекс заходів, спрямованих на захист інформації від несанкціонованого перегляду, копіювання або видалення. Несанкціонований доступ до інформації може бути здійснений як фізично, так і дистанційно.

До заходів захисту від несанкціонованого доступу відносяться:

– Контроль доступу: обмеження доступу до інформації лише уповноваженим особам. Контроль доступу може здійснюватися за допомогою фізичних бар'єрів (двері, замки, турнікети), електронних систем контролю доступу (СКД), а також за допомогою біометричних даних (відбитки пальців, обличчя, голос);

– Шифрування: перетворення інформації в нерозбірливий вид, який неможливо відновити без ключа. Шифрування використовується для захисту інформації від несанкціонованого перегляду та копіювання;

– Конфіденційність інформації: обмеження доступу до інформації лише тим особам, які мають необхідні для її використання повноваження. Конфіденційність інформації може забезпечуватися за допомогою заходів організаційного характеру, а також за допомогою технічних засобів захисту інформації;

Найбільш ефективним способом захисту інформації є комплексний підхід, який включає в себе як фізичні, так і криптографічні заходи захисту.

1.4 Мережевий захист

Мережевий захист – це комплекс заходів, спрямованих на захист інформації, яка передається по мережі, від несанкціонованого доступу, спотворення або знищення. Мережевий захист є одним з найважливіших складових комплексної системи захисту інформації.

1.4.1 Брандмауер

Брандмауер (firewall) – це пристрій або програмне забезпечення, яке контролює вхідний і вихідний трафік в мережі. Брандмауер може використовуватися для блокування несанкціонованого доступу до мережі, а також для фільтрації трафіку, щоб запобігти поширенню шкідливого програмного забезпечення.

Існує два основних типи брандмауерів:

– Фізичні брандмауери: це пристрої, які встановлюються між мережами. Фізичні брандмауери можуть бути як апаратними, так і програмними;

– Програмні брандмауери: це програмне забезпечення, яке встановлюється на комп'ютері або сервері. Програмні брандмауери можуть бути як автономними, так і інтегрованими в операційну систему;

1.4.2 VPN (Віртуальна приватна мережа)

VPN (Virtual Private Network) – це технологія, яка дозволяє створювати зашифроване з'єднання між двома або більше комп'ютерами, які знаходяться в різних мережах. VPN використовується для захисту інформації, яка передається по відкритій мережі, наприклад, Інтернету.

VPN-з'єднання створюється за допомогою VPN-клієнта і VPN-сервера. VPN-клієнт встановлюється на комп'ютері, який хоче підключитися до VPN-мережі. VPN-сервер встановлюється на комп'ютері, який є кінцевою точкою VPN-з'єднання.

1.4.3 IDS (Intrusion Detection Systems)

IDS (Intrusion Detection System) – це система, яка використовується для виявлення вторгнень в мережу. IDS аналізує трафік в мережі і виявляє ознаки несанкціонованого доступу або злому.

IDS може бути як відомою, так і невідомою. Знайома IDS аналізує трафік в мережі за допомогою набору правил, які визначають ознаки несанкціонованого доступу або злому. Невідома IDS аналізує трафік в мережі за допомогою

штучного інтелекту і намагається виявити аномалії в поведінці користувачів або пристроїв.[12]

Мережевий захист забезпечує наступні переваги:

- Захист від несанкціонованого доступу;
- Захист від спотворення інформації;
- Захист від знищення інформації;
- Захист від поширення шкідливого програмного забезпечення;

Найбільш ефективним способом захисту інформації є комплексний підхід, який включає в себе як брандмауер, так і VPN, і IDS.

1.5 Організаційні заходи

Організаційні заходи – це комплекс заходів, спрямованих на створення ефективної системи управління інформаційною безпекою в організації. Ці заходи включають різні аспекти, від розробки політик і процедур до контролю і моніторингу діяльності працівників. Давайте детально розглянемо ключові компоненти організаційних заходів в контексті захисту інформації.

1.5.1 Розробка політик і процедур

Розробка політик і процедур – це процес визначення цілей, принципів, правил і відповідальності, які регулюють діяльність організації в сфері захисту інформації. Політики і процедури повинні бути відповідними до законодавства, стандартів, рекомендацій і кращих практик, а також враховувати специфіку організації, її бізнес-процесів, ризиків і потреб. Політики і процедури повинні бути документовані, затверджені керівництвом, поширені серед співробітників і періодично переглядатися і оновлюватися. Для розробки політик і процедур можна використовувати такі джерела, як міжнародні стандарти ISO/IEC 27000, національні стандарти ДСТУ 3396.1-96, методичні рекомендації Держспецзв’язку та інші. [3]

Політика безпеки повинна містити такі розділи:

- Цілі та завдання захисту інформації;

– Об'єкти захисту інформації;

– Загальні принципи захисту інформації;

– Конкретні заходи захисту інформації;

Політика безпеки повинна бути доступна для всіх співробітників організації.

1.5.2 Навчання та розвиток персоналу

Навчання та розвиток персоналу – це процес підвищення кваліфікації, компетентності і свідомості співробітників організації в галузі інформаційної безпеки. Навчання та розвиток персоналу повинні бути плановими, цілеспрямованими, регулярними і адаптованими до рівня знань, ролі і функцій кожного співробітника. Навчання та розвиток персоналу повинні включати теоретичні і практичні заняття, тести, ігри, симуляції, кейси, воркшопи, вебінари, тренінги, курси, сертифікації тощо. Навчання та розвиток персоналу повинні бути оцінені за ефективністю, задоволенням і застосуванням отриманих знань і навичок на робочому місці.

Навчання персоналу з питань інформаційної безпеки має включати в себе такі теми:

– Загрози інформаційній безпеці;

– Методи захисту інформації;

– Правила поведінки в інформаційному просторі;

Навчання персоналу з питань інформаційної безпеки слід проводити регулярно, не рідше одного разу на рік.

Організаційні заходи забезпечують наступні переваги:

– Забезпечують розуміння співробітниками важливості захисту інформації;

– Формують у співробітників відповідальність за захист інформації;

– Сприяють дотриманню співробітниками правил захисту інформації;

Найбільш ефективним способом захисту інформації є комплексний підхід, який включає в себе як організаційні, так і технічні заходи захисту інформації.

Приклади використання організаційних заходів захисту інформації:

- Розробка і впровадження політики безпеки;
- Навчання персоналу з питань інформаційної безпеки;
- Розподіл обов'язків і повноважень;
- Контроль доступу до інформації;
- Управління ризиками;
- Розслідування інцидентів інформаційної безпеки;

1.5.3 Аудит безпеки

Аудит безпеки – це процес незалежної і об’єктивної оцінки стану інформаційної безпеки організації за допомогою спеціалізованих методів, технік і інструментів. Аудит безпеки повинен бути здійснений кваліфікованими аудиторами, які мають необхідні сертифікати, досвід і репутацію. Аудит безпеки повинен бути проведений за погодженою програмою, критеріями, стандартами і процедурами.

Аудит безпеки може бути спрямований на виявлення вразливостей у таких областях:

- Політика безпеки;
- Процедури безпеки;
- Контроль доступу;
- Резервне копіювання та відновлення;
- Захист від інцидентів безпеки;

Аудит безпеки дозволяє виявити уразливості, які можуть виникнути внаслідок недосконалості політики безпеки, процедур безпеки або технічних систем.

Зазвичай, регулярні оцінки безпеки проводяться не рідше одного разу на рік.

Оцінки безпеки повинні проводитися незалежними фахівцями, які мають досвід роботи в галузі інформаційної безпеки. Фахівці, які проводять оцінки

безпеки, повинні бути обізнані про сучасні загрози інформаційній безпеці та про методи захисту інформації.

Результати оцінок безпеки повинні бути документовані та передані керівництву організації для вжиття заходів щодо усунення виявлених вразливостей

1.5.4 Пантестування

Пантестування – це вид тестування безпеки, який проводиться для виявлення вразливостей в системі безпеки організації. Пантестування може проводитися як внутрішніми, так і зовнішніми спеціалістами.

Мета пантестування – виявити вразливості в системі безпеки, які можуть бути використані для несанкціонованого доступу до інформації, зміни або знищення інформації, або відмови в обслуговуванні.

Пантестування може бути декількох типів:

– Чорне тестування – тестування, яке проводиться без будь-якої інформації про систему безпеки організації;

– Сіре тестування – тестування, яке проводиться з обмеженою інформацією про систему безпеки організації;

– Біле тестування – тестування, яке проводиться з повною інформацією про систему безпеки організації;

Пантестування може проводитися за допомогою різних методів, наприклад:

– Сканування мережі – сканування мережі для виявлення відкритих портів і небезпечних служб;

– Взлом паролів – спроби зламати паролі користувачів або систем;

– Спроби несанкціонованого доступу – спроби отримати доступ до інформації або систем без дозволу;

– Спроби зміни або знищення інформації – спроби змінити або знищити інформацію;

– Спроби відмови в обслуговуванні – спроби зробити систему недоступною для користувачів;

План пантестування повинен включати в себе такі елементи:

– Цілі – що саме потрібно виявити за допомогою пантестування;
– Обсяг – які системи та процеси будуть тестуватися;
– Методи – які методи будуть використовуватися для тестування;
– Календарізований план– коли буде проводитися пантестування;

Пантестування повинно проводитися в контрольованому середовищі. Це означає, що тестування не повинно впливати на роботу системи в реальному часі.

Під час пантестування тестувальники повинні дотримуватися таких правил:

– Не завдавати шкоди системі;
– Не порушувати конфіденційність користувачів;
– Не використовувати пантестування для особистої вигоди;

Пантестування проводиться за допомогою ручних та автоматизованих методів. Пантестування може бути направлено на виявлення вразливостей у таких областях:

– Фізичний захист;
– Мережевий захист;
– Системи безпеки;
– Програмне забезпечення;
– Процеси та процедури;

Пантестування дозволяє виявити вразливості, які неможливо виявити за допомогою стандартних засобів контролю безпеки.[10]

1.6 Запобігання витоку даних

Запобігання витоку даних (DLP) – це комплекс заходів, спрямованих на захист інформації від несанкціонованого поширення. DLP-системи аналізують інформацію, яка передається по мережі, зберігається на пристроях або видаляється, і виявляють ознаки витоку даних.

DLP-системи можуть бути як програмними, так і апаратними. Програмні системи встановлюються на комп'ютери або сервери, а апаратні встановлюються між мережами або між мережею та комп'ютером.

DLP-системи можуть використовувати такі методи виявлення витоку даних:

– Аналіз вмісту – DLP-система аналізує вміст інформації, яка передається по мережі, зберігається на пристроях або видаляється;

– Аналіз поведінки – DLP-система аналізує поведінку користувачів, наприклад, частоту і обсяг пересилаються даних;

– Аналіз контексту – DLP-система аналізує контекст, в якому передається інформація, наприклад, час і місце передачі даних;

DLP-системи забезпечують наступні переваги:

– Захист від несанкціонованого поширення інформації;

– Зниження ризику витоку конфіденційної інформації;

– Покращення дотримання політики безпеки;

DLP-системи не можуть повністю захистити інформацію від витоку, але вони можуть допомогти значно знизити цей ризик.

DLP-системи повинні бути регулярно оновлюватися для забезпечення захисту від нових загроз.

1.7 Резервне копіювання та відновлення

Резервне копіювання та відновлення – це комплекс заходів, спрямованих на захист інформації від втрати внаслідок аварії, помилки або злому. Резервне

копіювання та відновлення є одним з найважливіших елементів системи захисту інформації.

Стратегії резервного копіювання – це плани, які визначають, як буде проводитися резервне копіювання інформації, які дані будуть копіюватися та як буде відновлюватися інформація в разі її втрати.

Існує кілька основних стратегій резервного копіювання:

– Стратегія повного резервного копіювання – при цій стратегії вся інформація копіюється повністю. Це найбезпечніша стратегія, але вона також є найбільш ресурсомісткою;

– Стратегія інкрементного резервного копіювання – при цій стратегії копіюються лише зміни, які були внесені до інформації з моменту останнього резервного копіювання. Це менш ресурсомістка стратегія, ніж стратегія повного резервного копіювання, але вона також менш безпечна;

– Стратегія диференціального резервного копіювання – при цій стратегії копіюються всі зміни, які були внесені до інформації з моменту останнього повного резервного копіювання. Це більш ресурсомістка стратегія, ніж стратегія інкрементного резервного копіювання, але вона також більш безпечна;

Найчастіше використовується комбінація цих стратегій. Наприклад, може використовуватися стратегія повного резервного копіювання щотижня, а потім інкрементне або диференціальне резервне копіювання щодня.

Інформація, яка повинна бути включена до резервної копії, повинна визначатися на основі важливості інформації. Важлива інформація, така як фінансові дані, конфіденційні документи або дані клієнтів, повинна бути включена до резервної копії.

Резервне копіювання інформації повинно проводитися регулярно. Частота резервного копіювання залежить від важливості інформації та від ризику її втрати. Копії повинні зберігатися в безпечному місці, яке недоступне для несанкціонованого доступу.

Важливо регулярно перевіряти резервні копії. Перевірка резервних копій дозволяє переконатися, що вони можуть бути успішно відновлені в разі необхідності.

Найкращий спосіб відновити інформацію з резервної копії - це використовувати програмне забезпечення для відновлення.

Резервне копіювання та відновлення є важливим елементом системи захисту інформації. Правильно розроблена та впроваджена стратегія резервного копіювання та відновлення може допомогти захистити інформацію від втрати в разі аварії, помилки або злому. [2]

2 РОЗРОБКА СИМЕТРИЧНОГО АЛГОРИТМУ ШИФРУВАННЯ НА ОСНОВІ ЗСУВУ БАЙТІВ

2.1 Аналіз симетричних алгоритмів шифрування

Симетричні алгоритми шифрування, що використовують однаковий ключ для шифрування та розшифрування даних, є одними з найважливіших інструментів у цій галузі.

Саме ці алгоритми становлять фундамент для забезпечення конфіденційності та цілісності інформації у різноманітних застосуваннях, від захисту особистих даних до забезпечення безпеки національних комунікаційних систем. Ефективність симетричних алгоритмів шифрування лежить не тільки у їх криптографічній стійкості, але й у їх здатності працювати швидко та ефективно навіть на обмежених у ресурсах системах.

Цей аналіз симетричних алгоритмів шифрування має на меті оглянути ключові аспекти цих алгоритмів, включаючи їхню структуру, режими роботи, переваги, недоліки та сфери застосування. Від класичних алгоритмів, таких як DES та 3DES, до сучасних стандартів, таких як AES, цей аналіз має на меті надати глибоке розуміння важливості та складності симетричного шифрування в сучасному світі кібербезпеки. [2]

2.1.1 AES

Алгоритм шифрування AES (Advanced Encryption Standard) – це симетричний алгоритм блочного шифрування, який використовується для захисту інформації. Він був обраний у 2001 році як стандарт шифрування для урядових і комерційних застосувань у Сполучених Штатах Америки.

AES є 128-бітним алгоритмом, що означає, що блоки даних, які він шифрує або розшифровує, мають розмір 128 біт. Він також має три варіанти довжини ключа: 128, 192 і 256 біт. Ключ використовується для генерації секретної таблиці підстановок (S-box), яка є основою алгоритму.

AES складається з 14 раундів, які повторюються доти, доки не буде зашифрований весь вхідний блок. Кожен раунд складається з чотирьох етапів.

Суббайтова Підстановка (SubBytes) – перший етап шифрування підстановки є першим етапом кожного раунду в алгоритмі AES. На цьому етапі кожен байт вхідного блоку замінюється на інший байт, який вибирається з таблиці S-box.

Таблиця S-box є 16x16 таблицею, яка складається з 256 різних байтів. Кожен байт вхідного блоку використовується як індекс у таблиці S-box, щоб отримати відповідний байт для заміни.

Зсув рядів (Shift Row) – це операція, яка циклічно зсуває рядки даних на різну кількість байтів. Ця операція забезпечує необхідну дифузію даних, тобто розподіл впливу одного байта на багато інших байтів.

Мішання стовпів (Mix Column) – це операція, яка виконує математичні операції над кожним стовпцем даних, щоб змінити його значення. Ця операція також забезпечує дифузію даних, тобто розподіл впливу одного байта на багато інших байтів

XOR з ключем (Add Round Key) – етап XOR з ключем є четвертим і останнім етапом кожного раунду в алгоритмі AES. На цьому етапі до вхідного блоку додається ключ, який є одним з 14 ключів, що генеруються з основного ключа.

Ключ, який використовується на цьому етапі, залежить від номера раунду. Перший раунд використовує основний ключ, а наступні раунди використовують ключі, які генеруються з основного ключа.

Розшифровування за допомогою AES є зворотним процесом шифрування. На кожному етапі розшифровування використовуються ті ж самі етапи, що і на етапі шифрування, але в зворотному порядку.

Переваги алгоритму AES:

– Висока безпека – AES є дуже ефективним алгоритмом, який важко зламати. Він був протестований на міцність проти різноманітних атак, і він досі не був зламаний;

– Швидкість – AES є відносно швидким алгоритмом, який може шифрувати дані з високою швидкістю. Це робить його придатним для широкого спектру застосувань, включаючи мережеві протоколи, такі як HTTPS і SSH;

– Ефективність – AES є ефективним алгоритмом, який споживає відносно мало ресурсів. Це робить його придатним для використання на пристроях з обмеженими ресурсами, таких як мобільні телефони і смарт-пристрої;

Недоліки алгоритму AES:

– Ключова довжина – AES має три варіанти довжини ключа: 128, 192 і 256 біт. Ключова довжина 128 біт вважається достатньою для більшості застосувань, але ключова довжина 192 або 256 біт може бути більш бажаною для високочутливих даних. Також деякі користувачі хочуть робити більш довгі ключі або більш короткі в залежності від побажань;

– Складність реалізації – AES є складним алгоритмом, який може бути важко реалізувати правильно. Це може призвести до помилок, які можуть знизити безпеку шифрування; [1]

2.1.2 DES

DES - це алгоритм симетричного шифрування, який використовує один і той самий ключ для шифрування та розшифрування даних. Він був стандартом шифрування у США з 1977 до 2001 року, коли його замінив AES.

Перетворення – це перший етап, коли вхідні дані розбиваються на дві половини по 32 біти кожна і перетворюються за допомогою спеціальної таблиці.

Генерація ключів – це етап, коли початковий ключ довжиною 64 біти перетворюється на 16 підключів довжиною 48 біт кожен за допомогою операцій зсуву та перестановки.

Етапи Фейстеля – це 16 раундів, які повторюються для кожної половини даних. На кожному раунді виконуються такі операції:

– Розширення – це операція, яка перетворює 32-бітну половину даних на 48-бітну за допомогою дублювання деяких бітів;

– XOR – це операція, яка виконує операцію XOR (виключне АБО) між 48-бітною половиною даних та 48-бітним підключем етапу;

– S-Box – це операція, яка замінює кожні 6 бітів даних на 4 біти за допомогою восьми фіксованих таблиць замін, які називаються S-Box;

– P-Box – це операція, яка перетворює 32-бітні дані за допомогою спеціальної таблиці перестановки, яка називається P-Box;

– Обмін – це операція, яка обмінює дві половини даних місцями;

Зворотне перетворення – це фінальний етап, коли дві половини даних об’єднуються в один 64-бітний блок і перетворюються за допомогою зворотної таблиці перетворення.

Переваги:

– Надійність – на момент його створення DES вважався дуже надійним і був широко використаний у фінансових і урядових установах;

– Стандартизація – як один з перших стандартизованих алгоритмів, DES зіграв ключову роль у розвитку сучасної криптографії;

– Перевіреність – був ретельно вивчений і тестований протягом десятиліть, що довело його ефективність проти багатьох видів атак;

Недоліки DES:

– Обмежена довжина ключа – головним недоліком DES є його 56-бітний ключ, що з часом став вразливим до атак грубою силою (brute-force attacks);

– Вразливість до сучасних атак – з розвитком обчислювальної потужності атака грубою силою стала практично можливою, що призвело до заміни DES на більш безпечні алгоритми, такі як AES;

– Недостатньо складні S-бокси – S-бокси DES не достатньо складні для протистояння деяким сучасним методам криптоаналізу;[5]

2.1.3 3DES

Трьохкратний DES (Triple DES або 3DES) є розширенням алгоритму DES (Data Encryption Standard). Він був розроблений для збільшення рівня безпеки

DES шляхом виконання послідовності операцій шифрування DES кілька разів з різними ключами.

3DES є 64-бітним алгоритмом, що означає, що блоки даних, які він шифрує або розшифровує, мають розмір 64 біти. Він також має один варіант довжини ключа: 56 біт. Ключ використовується для генерації секретної таблиці підстановок (S-box), яка є основою алгоритму.

3DES складається з 16 раундів, які повторюються доти, доки не буде зашифрований весь вхідний блок. На кожному етапі використовується один з трьох ключів, які генеруються з основного ключа.

Етапи шифрування 3DES:

– Перший етап – дані шифруються за допомогою DES з використанням першого ключа (K1);

– Другий етап – результат першого етапу шифрування потім розшифровується за допомогою DES з другим ключем (K2);

– Третій етап – це зашифрування розшифрованих даних за допомогою третього ключа за алгоритмом DES. Це зроблено для забезпечення сумісності з DES, якщо третій ключ співпадає з першим;

Переваги 3DES:

– Підвищена безпека – завдяки використанню трьох ключів або двох ключів у трьох етапах шифрування, 3DES значно стійкіший до атак грубою силою, ніж оригінальний DES;

– Сумісність – 3DES може використовувати інфраструктуру, розроблену для DES, що робить його зручним варіантом для організацій, які вже використовували DES;

– Перевіреність – як і DES, 3DES був ретельно вивчений і тестований, що забезпечує високий рівень довіри до його безпеки;

Недоліки 3DES:

– Низька швидкість – триетапне шифрування робить 3DES значно повільнішим, ніж більшість сучасних алгоритмів, таких як AES;

– Старіння DES – оскільки 3DES базується на DES, він успадковує обмеження і потенційні вразливості цього алгоритму;

– Обмеження розміру ключа – хоча 3DES використовує декілька ключів, ефективний розмір ключа не є таким великим, як може здатися, через проблему "зустрічі посередині" (meet-in-the-middle attack);

– Складність управління ключами – необхідність управління декількома ключами може створювати додаткові виклики в термінах логістики та безпеки.

– Застарівання – в сучасному світі, де швидкість та ефективність є критичними, 3DES все частіше замінюється на більш сучасні та ефективні алгоритми, такі як AES;

2.1.4 RC4

RC4 – це потоковий шифр, який використовує один ключ для шифрування та розшифрування даних. Він був розроблений Роном Рівестом у 1987 році і широко застосовувався в різних протоколах безпеки, таких як SSL, TLS, WEP і WPA. Однак, він має деякі вразливості, які роблять його небезпечним для сучасного використання.

Етапи RC4 алгоритму:

– Ініціалізація – це етап, коли створюється масив S з 256 байтів, які містять числа від 0 до 255. Потім масив S перетасовується за допомогою ключа, який може мати довжину від 1 до 256 байтів. Це здійснюється за допомогою двох змінних i та j, які початково дорівнюють 0. Для кожного i від 0 до 255 виконуються наступні кроки:

- j прирівнюється до (j + S[i] + ключ[i mod довжина ключа]) mod 256;
- Значення S[i] та S[j] обмінюються місцями;

– Генерація – це етап, коли генерується псевдовипадковий потік байтів, який називається K. Для кожного байта, який потрібно зашифрувати або розшифрувати, виконуються наступні кроки:

- i прирівнюється до (i + 1) mod 256;
- j прирівнюється до (j + S[i]) mod 256;
- Значення S[i] та S[j] обмінюються місцями;
- t прирівнюється до (S[i] + S[j]) mod 256;
- K прирівнюється до S[t];

– Шифрування – це етап, коли вхідний потік байтів, який називається M, зашифровується за допомогою псевдовипадкового потоку байтів K. Це здійснюється за допомогою операції XOR (виключне АБО) між кожним байтом M та K. Результатом є вихідний потік байтів, який називається C.

Переваги RC4

– Простота та швидкість – однією з основних переваг RC4 є його простота і висока швидкість обробки, особливо на програмному рівні;

– Гнучкість ключа – RC4 підтримує ключі різної довжини, що робить його досить гнучким для різних застосувань;

– Малі вимоги до пам'яті – RC4 не вимагає багато пам'яті для роботи, що робить його підходящим для пристроїв з обмеженими ресурсами;

Недоліки RC4:

– Проблеми безпеки – було виявлено кілька слабких місць у RC4, особливо пов'язаних зі стартовою частиною потоку ключів, що може дозволити певні види криптографічних атак:

- Початок вихідного потоку ключів не відкидається, що призводить до повторного використання набору ключів;
- Використовуються не випадкові або споріднені ключі;
- Використовуються короткі або однорідні відкриті тексти;

– Застарівання – у зв'язку з виявленими слабкими місцями, RC4 більше не рекомендується для використання в нових застосуваннях безпеки;

2.1.5 RC2

Алгоритм RC2 використовує серію раундів шифрування для перетворення вхідних даних (початкового тексту) в зашифрований текст. Кожен раунд включає кілька кроків, які поєднують лінійні та нелінійні операції для забезпечення безпеки шифрування. Давайте розглянемо кожен етап раунду шифрування RC2 детальніше:

Перед початком раундів шифрування ключ, поданий користувачем, розширюється до 128 байтів за допомогою спеціального алгоритму. Цей розширений ключ використовується на різних етапах кожного раунду.

Вхідний блок даних розбивається на 16-бітні слова. RC2 працює з кожним із цих слів окремо, використовуючи різні комбінації операцій.

На кожному етапі раунду виконуються різні операції з 16-бітними словами, включаючи:

– Міксування та перетворення даних (Mixing and transforming):
  - Додавання – слова додаються до відповідних частин розширеного ключа;
  - Побітове XOR – слова комбінуються з іншими словами за допомогою XOR;
  - Перемішування – виконуються різні операції перемішування, такі як циклічний зсув вліво;

– Функція Міксування (Mixing Function) – ключовий елемент, що забезпечує дифузію та забезпечує безпеку шифру. Вона включає нелінійні перетворення, які залежать від значень вхідних слов і частини розширеного ключа;

Переваги RC2:

– Ефективність – може шифрувати дані з високою швидкістю;

– Простота реалізації – RC2 є відносно простим алгоритмом, який легко реалізувати;

Недоліки:

– Низька дифузія – має низьку дифузію даних, тобто маленька зміна вхідних даних призводить до маленької зміни вихідних даних;

– Слабкий ключ – це алгоритм, який має обмежену довжину ключа, яка не може перевищувати 128 біт, що може бути недостатнім для сучасних стандартів безпеки;

– Вразливість до атак повторного використання – RC2 вразливий до атак повторного використання. Це означає, що атакуючий може використовувати зашифровані дані, щоб отримати інформацію про ключ;

2.2 Постановка задачі створення та розробки нового симетричного алгоритму шифрування даних

За поточних обставин у світі, питання захисту особистих даних та забезпечення не тільки самих себе, а й інших є однієї з найголовніших проблем які потрібно вирішувати. За останні роки неодноразово було зафіксовано витік особистих даних лише з одних сервісів швидких знімків екрану з можливістю зберігання скриншотів. До прикладу Lightshot – за цією програмою неодноразово помічали дефекти системи і саме через відсутність захисту особистих даних. Бодай кожен кому не ліньки, може замінити просто пару символів в посиланні на сервіс зберігання скриншотів Lightshot і отримати знімок екрану/зображення будь-якої людини, без жодних обмежень, як це показано на рис. 2.1. [4, 8, 9]

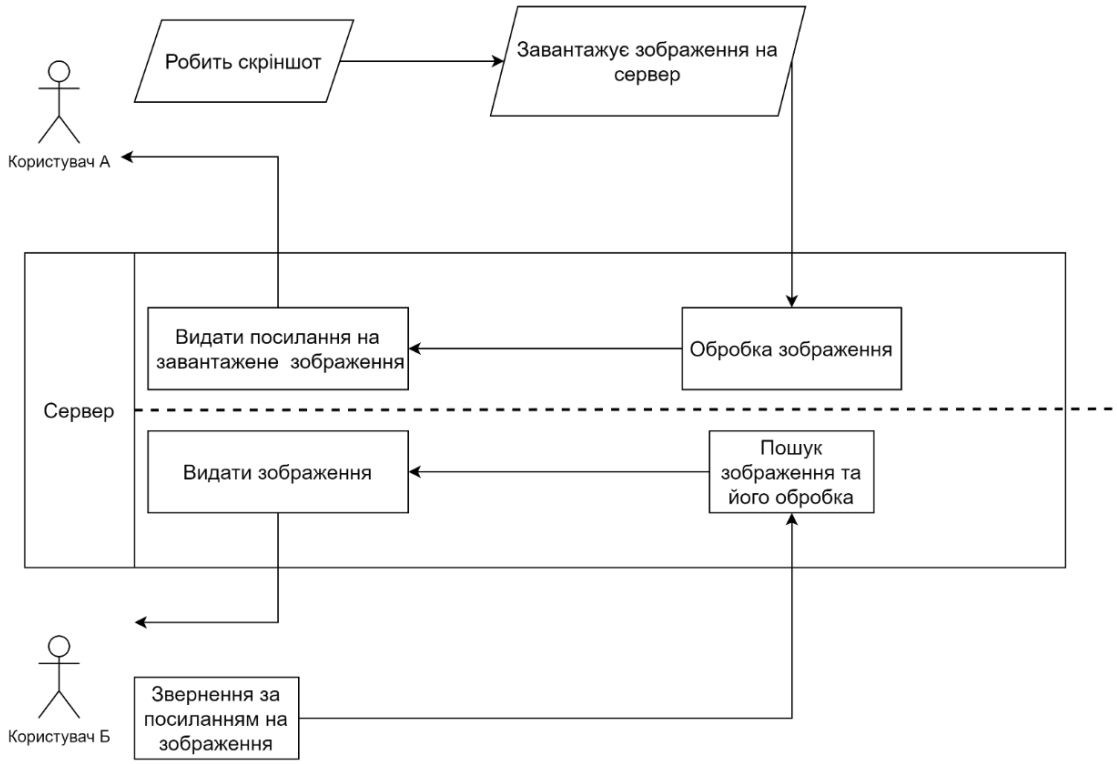


Рис. 2.1 Процес взаємодії користувачів з сервісом швидких знімків екрану з можливістю зберігання скриншотів

Існує декілька реальних причин можливої відсутності використання захисту інформації: мала ефективність або складність використання алгоритму розробником, або ж складність користування системою через обмеження способів шифрування даних, що могло би забезпечити доступ до зображень користувачів по код-паролю який відомий людині яка завантажила зображення та відповідно встановила пароль до фото. Відповідно «отримувачі» цього зображення – особи які намагаються отримати зображення за посиланням в мережі, повинні ввести пароль доступу щоб отримати розшифроване зображення. Для такого процесу взаємодії можна віднести алгоритми симетричного шифрування.

Шифрування даних є одним з найефективніших засобів захисту даних в розрізі проблеми завантаження зображень, їх отримання, та захисту цих самих зображень в мережі. З точки зору користувача, та його взаємодії з сервісом, найкращий варіант є застосування алгоритму симетричного шифрування завдяки одному ключу як для шифрування, так і для дешифрування даних (рис. 1.2).

Але одним з головних аспектів використання алгоритмів в цьому процесі є зручність взаємодії кінцевого користувача. Нажаль, існуючі методи симетричного шифрування не сильно можуть похизуватись зручністю для кінцевого користувача, а тим паче для розробника який буде впроваджувати той алгоритм шифрування, як це видно по таблиці 2.1, де поле довжина ключа є обмеженим для кожного алгоритму, що дуже обмежує користувачів у різноманітті паролів які можна використовувати для захисту зображень. [5]

Таблиця 2.1

Основні алгоритми симетричного шифрування

Особливості	AES	DES	3DES	RC4	RC2
Довжина ключа	128, 196, 256 біт	56 біт	112 або 118 біт	8 – 2048 біт	До 128 біт
Розмір блоку	128 біт	64 біт	64 біт	Немає (потоковий шифр)	64 біт
Підхід / метод	Заміна та перестановка байтів	Мережа Фейстеля	3 цикли DES	Потоковий шифр	Мережа Фейстеля
Швидкість	Дуже висока	Низька	Дуже низька	Висока (але не рекомендується через вразливості)	Висока

Згідно таблиці 2.1, можна побачити, що деякі алгоритми є доволі незручними для кінцевого користувача, що зумовлено обмеженнями довжини ключа, який можна використовувати, а один з алгоритмів взагалі не

рекомендується використовувати через можливі вразливості. Тому слід створити та розробити новий симетричний алгоритм шифрування для збільшення зручності користування ним.

2.3 Створення алгоритму симетричного шифрування даних на основі зміщення байтів

В основу створення симетричного алгоритму шифрування покладено метод циклічного зсуву бітів у байтах та XOR (виключна диз'юнкція) операція над байтами.

Зсув байтів - це операція, яка змінює порядок байтів у даних, де біти в бінарних числах або інших бінарних даних зсуваються вліво або вправо. Це може змінювати значення даних і часто використовується для різних цілей, включаючи шифрування, оптимізацію обчислень і маніпуляцію з даними.

Основні різновиди зсуву бітів:

а) Логічний зсув (рис. 2.2):

1) Логічний зсув вліво – біти зсуваються вліво, а на вакантні місця праворуч вставляються нулі. Це еквівалентно множенню на 2;
2) Логічний зсув вправо – біти зсуваються вправо, а на вакантні місця зліва вставляються нулі. Це еквівалентно діленню на 2, проте без врахування знаку;

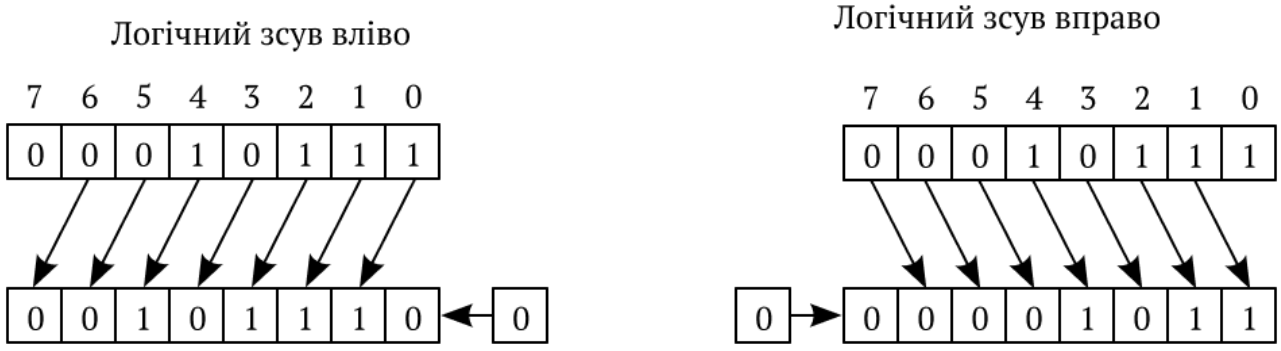


Рис. 2.2 Логічний зсув

б) Арифметичний зсув (рис. 2.3):

1) Арифметичний зсув вліво – схожий на логічний зсув вліво, але зазвичай використовується для цілих чисел;
2) Арифметичний зсув вправо – біти зсуваються вправо, але зберігається знак числа (старший біт). Тобто, якщо число було від'ємним, то старший біт залишиться 1, а якщо додатним – 0;

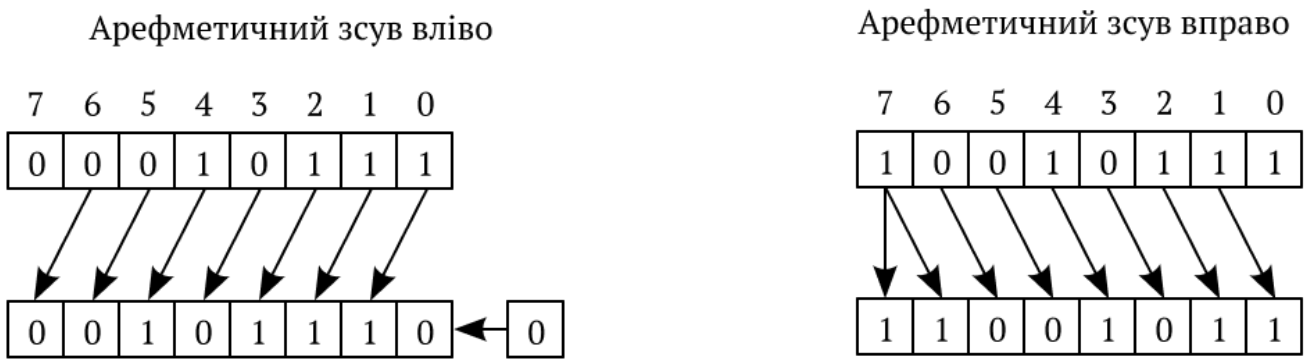


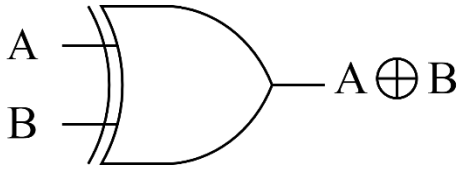
Рис. 2.3 Арифметичний зсув

в) Циклічний зсув (рис. 2.4):

1) Циклічний Зсув Вліво/Вправо – під час зсуву біт, який "виходить" за межі числа, переміщується на протилежний кінець. Це створює ефект "циклічності", де всі біти в числі обертаються в одному напрямку;

Рис. 2.4 Циклічний зсув

Операція XOR, також відома як виключна диз'юнкція або порозрядне додавання за модулем два, є логічною і бітовою операцією, яка виконується над двома числами. Результатом операції XOR є 1, якщо один із операндів є 1, а інший – 0, і 0, якщо обидва операнди є 0 або 1 (рис. 2.5).



A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Рис. 2.5 XOR операція

Операція XOR має кілька цікавих властивостей, які роблять її корисною для різних цілей. Наприклад, операція XOR є:

– Комутативна – порядок операндів не впливає на результат;

– Асоціативна – результат операції XOR не змінюється, якщо операнди переставляються в інший порядок;

– Ідемпотентна – операція XOR над одним і тим самим операндом завжди повертає 0;

Операція XOR використовується в різних сферах, наприклад:

– У криптографії операція XOR використовується для шифрування даних. Наприклад, при шифруванні за допомогою алгоритму DES дані шифруються шляхом застосування XOR операції з ключа шифрування;

– У програмуванні операція XOR використовується для обробки даних. Наприклад, операція XOR може використовуватися для інвертування бітів або для перевірки відповідності даних певному шаблону;

– У електроніці операція XOR використовується в логічних схемах. Наприклад, операція XOR може використовуватися для побудови мультиплексорів або дешифраторів;

2.4 Математична модель

Алгоритм симетричного шифрування, який буде розроблюватись заснований на операції зсуву байтів та XOR операції над байтами зображення по код-паролю. У цьому алгоритмі зсув байтів виконується на величину, яка визначається відповідним байтом код-паролю.

Для створення математичної моделі, потрібно визначити зображення для шифрування, код пароль та математичний алгоритм шифрування.

Існують різні види шифрування зображень:

– Матричний – зображення розбивається на рівні матриці байт 2x2, 4x4, 6x6 (2.1) і кожна з них шифрується побайтово окремо;

$$\begin{bmatrix} a_{00} & \cdots & a_{0j} \\ \vdots & \ddots & \vdots \\ a_{i0} & \cdots & a_{ij} \end{bmatrix} => \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \begin{bmatrix} a_{02} & a_{03} \\ a_{12} & a_{13} \end{bmatrix} \begin{bmatrix} a_{20} & a_{21} \\ a_{30} & a_{31} \end{bmatrix} \begin{bmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{bmatrix} \quad \cdots \tag{2.1}$$

де, *i* – кількість байт в рядку (ширина зображення в кількості байт);

j – кількість байт в стовпчику (висота зображення в кількості байт);

– Блочний – зображення розбивається на рівні блоки байт і кожен блок шифрується окремо, або послідовно кожен на основі попереднього блоку;

$$\begin{bmatrix} a_{00} & \cdots & a_{0j} \\ \vdots & \ddots & \vdots \\ a_{i0} & \cdots & a_{ij} \end{bmatrix} => [a_{00} \quad a_{01} \quad a_{02} \quad a_{03}][a_{04} \quad a_{05} \quad a_{06} \quad a_{07}] \quad \cdots \quad (2.2)$$

– Потоковий – кожен байт зображення шифрується послідовно;

$$\begin{bmatrix} a_{00} & \cdots & a_{0j} \\ \vdots & \ddots & \vdots \\ a_{i0} & \cdots & a_{ij} \end{bmatrix} => (a_n) : a_1, a_2, a_3, a_4, \dots, a_n \quad (2.3)$$

де $n = i \cdot j$ –загальна кількість байт в зображенні

Для реалізації алгоритму симетричного шифрування візьмемо потоковий варіант, з можливістю подальшої реалізації як блочного так і матричного шифрування.

Фактично код-пароль являє собою просто набір символів, які можна комбінувати між собою у будь якому виді, маленькі, великі літери, цифри, символи і т.д., це все може бути код-паролем. Цей же код пароль грає роль у шифруванні зображення за рахунок кожного байту цього код-паролю, що можна визначити як послідовність елементів k з мінімальною довжиною 6 символів для забезпечення мінімального рівня паролю.

$$(k_m) : k_1, k_2, k_3, k_4, \dots, k_m \quad (2.4)$$

де $m \geq 6$ – кількість символів код-паролю

Математично алгоритм можна представити таким чином:

$s = 3$ – кількість бітів для зсуву;

Проходимо по кожному елементу (a_n) :

а) Взяти черговий байт k_i з ключа (k_m) по циклічній індексації, де $i = n \; mod \; len(k_m)$;

б) Визначити операцію над поточним байтом a_j з масиву (a_n), де j є $[0; n]$ і записати результат операції у байт b_j вихідної послідовності (b_n) :

1) Якщо індекс j ділиться на 3 без залишку, виконати правий циклічний зсув на s позицій :

$$b_j = a_j \gg s \tag{2.5}$$

2) Якщо індекс ділиться на 2 без залишку, виконати лівий циклічний зсув на s позицій :

$$b_j = a_j \ll s \tag{2.6}$$

3) Інакше, додати до байту a_j результат XOR операції над байтом k_i з ключа і числа 90:

$$b_j = a_j + (k_i \oplus 90) \tag{2.7}$$

в) Виконати XOR операцію над отриманою сумою байтів b_j і k_i з ключа і числом 162, і записати результат у вихідну послідовність:

$$b_j = (b_j + k_i) \oplus 162 \tag{2.8}$$

2.5 Функціональна модель

Функціональну модель цього алгоритму можна представити як поліпшення загального процесу завантаження та обробки зображення (рис. 2.1) .

Тобто створений і розроблений алгоритм можна буде впроваджувати в цей процес, додавши до нього політику безпеки даних та захистивши користувачів від несанкціонованого отримання чужих зображень (рис. 2.6).

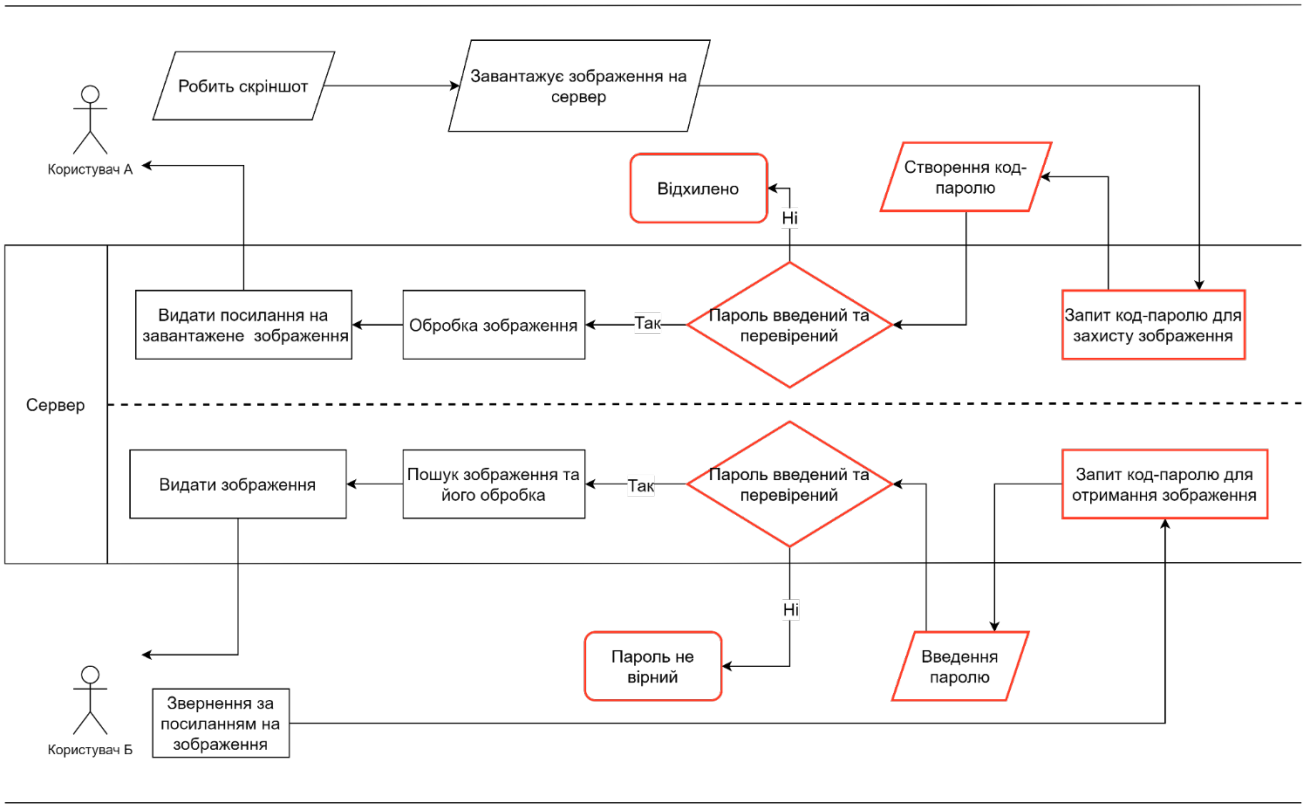


Рис. 2.6 Модифікований алгоритм шифрування, модифікований процес взаємодії користувача

Згідно рис. 2.6 можна побачити що було додано кроки політики безпеки, які сприяють збільшенню загального рівню захисту інформації користувачів. Але загальна обробка зображення на сервері потребує більшого роз’яснення, для відображення всієї картини необхідності додавання алгоритму шифрування в процесс взаємодії користувача з сервісом.

Послідовний алгоритм взаємодії користувача до впровадження алгоритму можна представити схематично (рис.2.7):

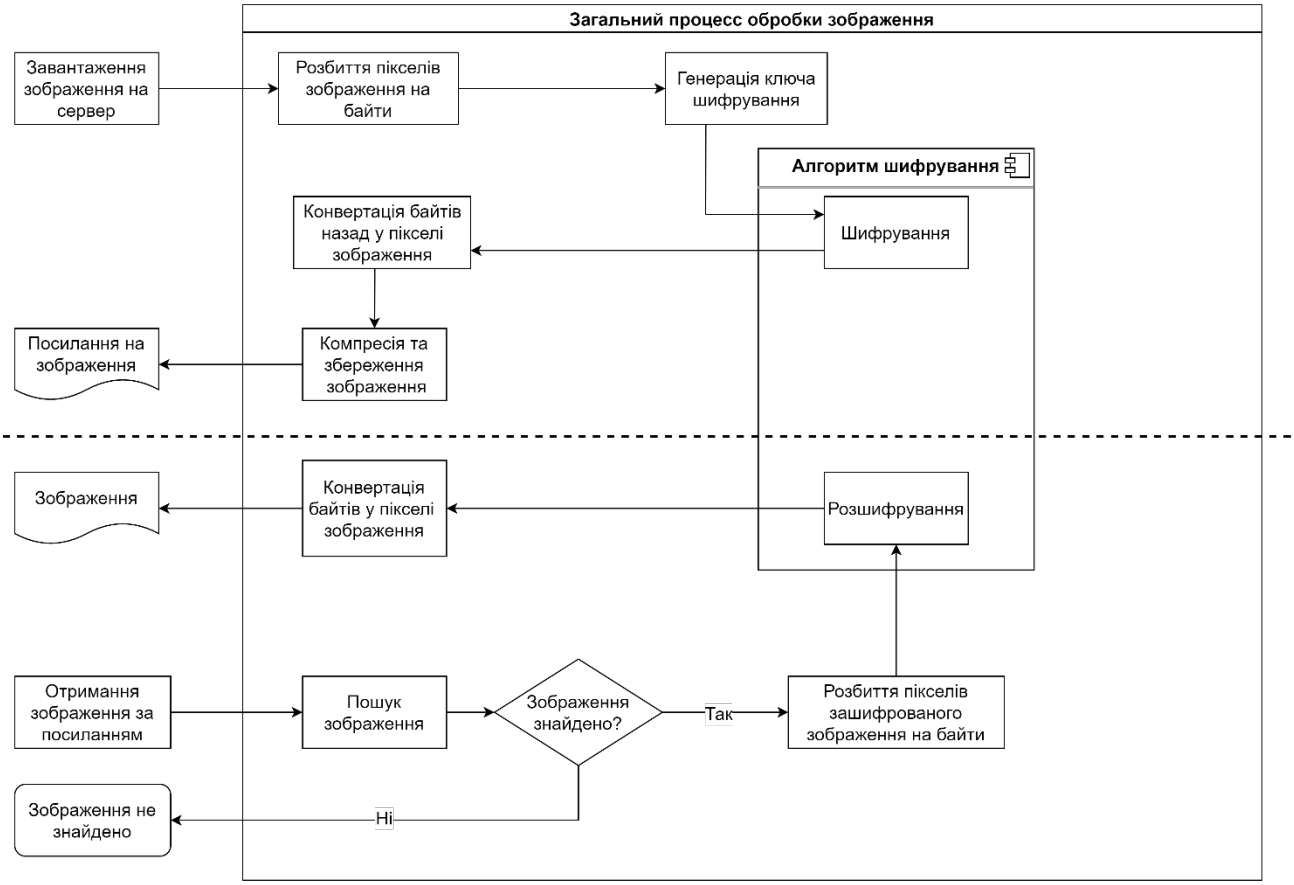


Рис. 2.7 Загальний процес обробки зображення сервером

Покроково діаграма поділена на 2 паралельних процеси :

а) Завантаження на сервер зображення з боку користувача А:

1) Сервер приймає зображення у відповідних форматах встановлених правилами сервісу, наприклад: .jpg, .png, .bmp, .jpeg і т.п.;

2) Прийняте зображення представляється у вигляді пікселів та розбивається на байти, найчастіше 1 піксель = 3 байти, що пояснюється 3 кольорам: червоний, зелений, синій;

3) Байти отримані з зображення передаються далі у алгоритм шифрування, але перед цим на сервері генерується ключ шифрування який необхідний для шифрування зображення відповідним алгоритмом;

4) Згодом за допомогою використання алгоритму шифрування, зображення кодується, задля забезпечення захисту інформації на стороні серверу, тобто в базі даних компанії, у внутрішньому колі;

5) Закодоване зображення (послідовність байтів) конвертується назад у пікселі та компресується для зменшення об’єму займаної пам’яті в місці зберігання;

6) Збережене зображення має посилання, щоб можна було отримати це зображення (згідно 2 процесу) з зовнішньої мережі, з інтернету;

б) Отримання зображення за посиланням користувача А користувачем Б:

1) Сервер отримує посилання за яким повинно знаходитись зображення, завантажене користувачем А;

2) Відносно посилання на зображення сервер намагається знайти зображення в місці зберігання. У випадку якщо зображення не буде знайдено користувачу буде видана помилка про відсутність зображення;

3) Коли зображення буде знайдено, це закодоване зображення потрібно у вигляді пікселів розбити на послідовність байтів, задля подальшого розшифрування;

4) Подальше розшифрування байтів зображення на основі ключа шифрування;

5) Розшифровані байти потрібно конвертувати назад у пікселі;

6) Видача зображення користувачу Б;

Згідно цього опису процесу обробки зображення, можна відслідкувати той момент, що сам користувач цим сервісом не має захисту даних, якщо знайти посилання на його зображення, або методом перебору, або ж просто викрасти його, якраз через відсутність захисту зовнішнього кола. Внутрішньо так, зображення шифруються для захисту даних серверів, але не для захисту користувачів.

Тож краще прийняти бік захисту користувачів, що дасть вибір користувачу особисто який пароль йому задавати для зображення, що відповідно може уберегти дані користувачів від витоку назовні, або спроби викрадення даних (рис. 2.8).

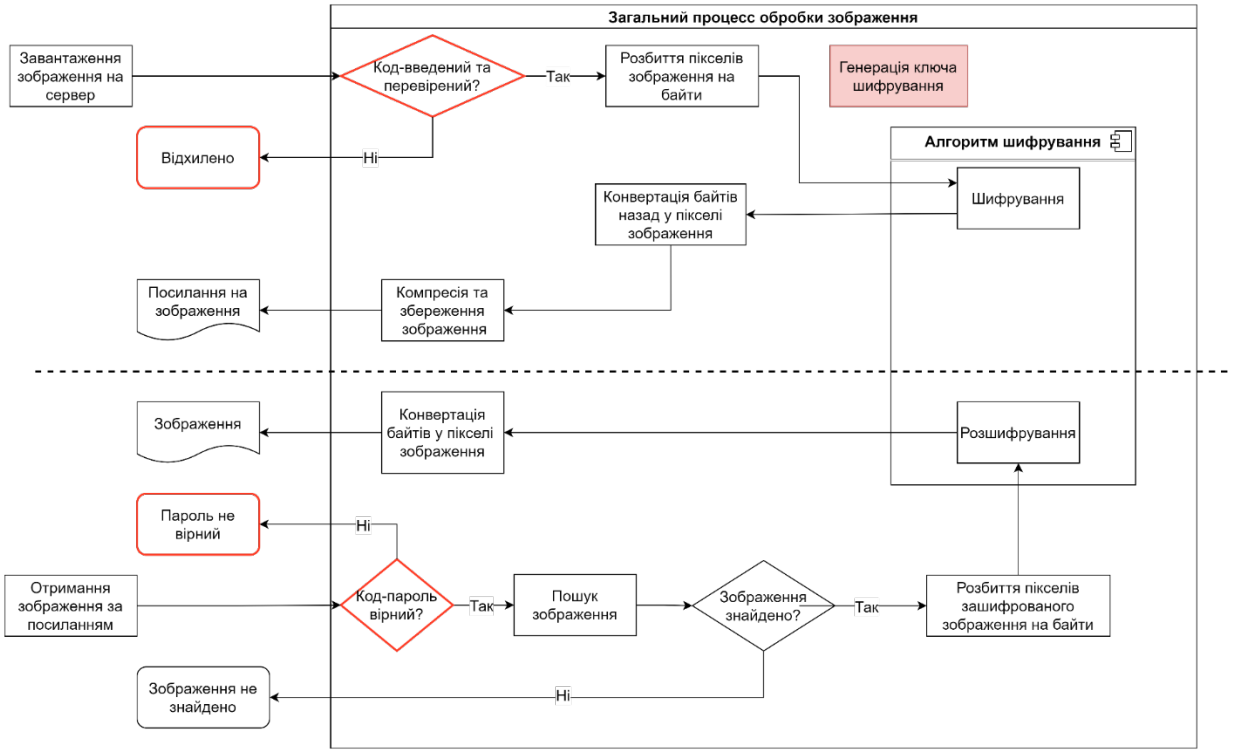


Рис. 2.8 Модифікований процес обробки зображення

Як видно по рис. 2.8 автоматична генерація ключа шифрування була видалена з загального процесу. Це дало змогу інтегрувати політику безпеки у вигляді задання коду шифрування з боку користувача.

Загалом математичне представлення алгоритму можна зобразити функціонально діаграмою (рис. 2.9):

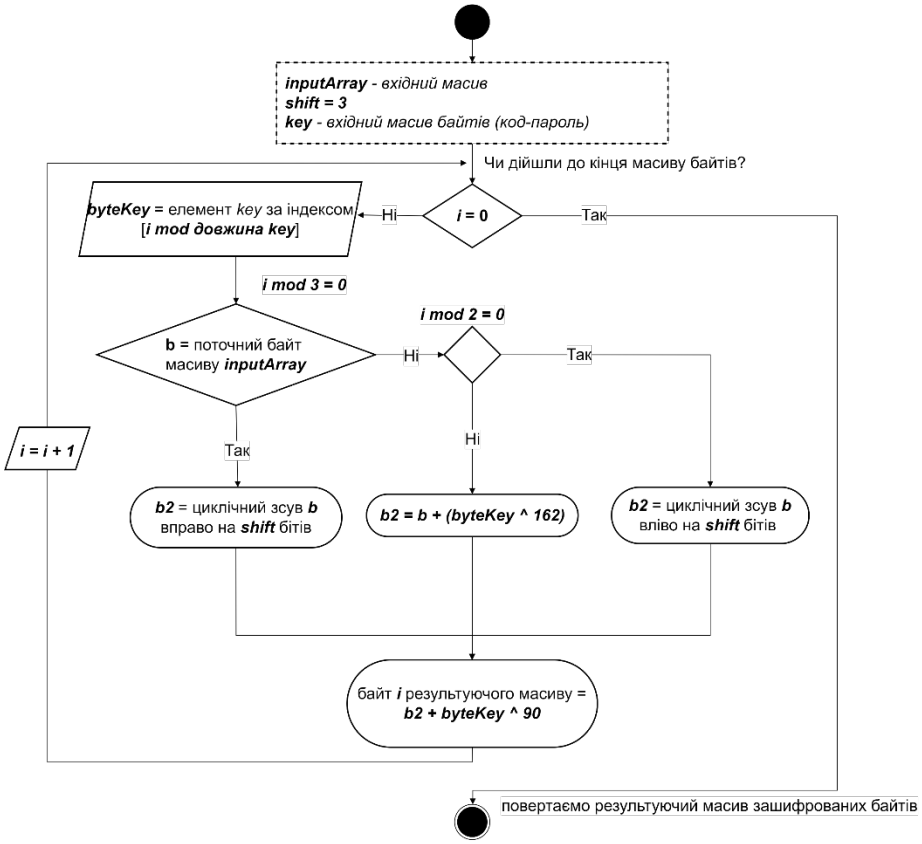


Рис. 2.9 Функціональне представлення алгоритму симетричного шифрування

2.6 Програмна реалізація симетричного алгоритму шифрування

Програмна реалізації алгоритму була виконана із застосуванням ноутбуку Acer Nitro 5 з наступними харектеристиками:

- Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz 2.50 GHz;
- ОЗП 32,0 ГБ DDR4-3200 (1600 МГЦ);
- Графічний адаптер: Intel HD Graphics (інтегрований);

Програмна реалізація проводилась під операційною системою Windows 11. Алгоритм може функціонувати на будь яких операційних системах. Приклад використання алгоритму наведений в додатку Б.

2.6.1 Опис використаних програмних засобів

Програмну реалізацію алгоритму було виконано у середовищі розробки Visual Studio 2022 Professional версії із застосуванням мови програмування C#.

Visual Studio – це інтегроване середовище розробки (IDE) для програмістів на .NET і C++ на Windows. Воно має багато функцій і можливостей, таких як GitHub Copilot, автодоповнення коду, налагодження, тестування, публікація та інші. За допомогою Visual Studio ви можете створювати веб-, хмарні, настільні, мобільні програми, сервіси та ігри. Visual Studio підтримує широкий спектр мов програмування, включаючи C++, C#, Visual Basic, JavaScript, Python і Java. Воно також підтримує різні типи проектів, включаючи веб-додатки, мобільні додатки, десктопні додатки та ігри.

Visual Studio є потужним і багатофункціональним IDE, який може використовуватися для розробки широкого спектру програмного забезпечення. Воно є популярним вибором для розробників, які працюють на платформах Microsoft.

C# (вимовляється як "See Sharp") – це сучасна високорівнева об'єктно-орієнтована мова програмування з безпекою типів. C# дозволяє розробникам створювати багато типів безпечних і надійних додатків, які працюють в .NET. Мова C# походить із сімейства мов C і буде знайома програмістам, які володіють мовами C, C++, Java та JavaScript.

C# – це об'єктно-орієнтована, компонентно-орієнтована мова програмування. C# надає мовні конструкції для безпосередньої підтримки цих концепцій, що робить C# природною мовою для створення та використання програмних компонентів. З моменту свого виникнення C# додала функції для підтримки нових робочих навантажень та нових методів проектування програмного забезпечення. За своєю суттю C# є об'єктно-орієнтованою мовою. Ви визначаєте типи та їхню поведінку.

Декілька функцій C# допомагають створювати надійні та довговічні додатки. Збір сміття автоматично звільняє пам'ять, зайняту недоступними невикористаними об'єктами. Типи, що обнуляються, захищають від змінних, які не посилаються на виділені об'єкти. Обробка винятків забезпечує структурований та розширюваний підхід до виявлення та відновлення помилок. Лямбда-вирази підтримують методи функціонального програмування. Синтаксис Language

Integrated Query (LINQ) створює загальний шаблон для роботи з даними з будь-якого джерела. Підтримка мовою асинхронних операцій забезпечує синтаксис для побудови розподілених систем. C# має уніфіковану систему типів. Всі типи C#, включаючи примітивні типи, такі як int та double, успадковуються від одного кореневого типу об'єкта. Всі типи поділяють набір спільних операцій. Значення будь-якого типу можна зберігати, транспортувати та оперувати з ними однаковим чином. Крім того, C# підтримує як користувацькі типи посилань, так і типи значень. C# дозволяє динамічно розподіляти об'єкти та зберігати в потоці легкі структури. C# підтримує узагальнені методи та типи, які забезпечують підвищену безпеку типів та продуктивність. C# надає ітератори, які дозволяють реалізаторам класів колекцій визначати власну поведінку для клієнтського коду. Опис структури проекту.

2.6.2 Опис структури проєкту

Опис структури проєкту «F_Project» наведено на рисунку 2.10.

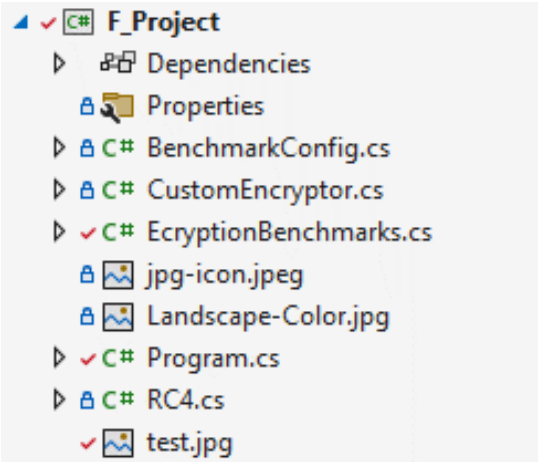


Рис. 2.10 Структура проєкту «F_Project»

Структура проєкту складається з наступних файлів та папок:

– Dependencies – мета-розділ проєкту в котрому зберігають залежності проєкту від деяких зовнішніх плагінів, для забезпечення роботи проєкту. Там також зберігаються посилання на системні компоненти проєкту;

– Properties – папка в які можуть знаходитись інформація про збірку: версія, інформація про авторство, налаштування та інші технічні файли;

– BenchamarkConfig.cs – файл, в якому клас BenchamarkConfig містить налаштування компоненту тестування алгоритмів для подальшого порівняння;

– CustomEncryptor.cs – файл, в якому клас CustomEncryptor містить реалізацію розробленого алгоритму шифрування;

– EncryptionBenchmarks.cs – файл, в якому клас EncryptionBenchmarks містить методи з реалізаціями шифрування різних симетричних алгоритмів, для використання в подальшому для порівняння ефективності алгоритмів;

– jpg-icon.jpeg, Landscape-Color.jpg, text.jpg – файли зображень для тестування алгоритмів шифрування;

– Program.cs – головний файл програми, в якому міститься метод Main - точка входу в програму;

– RC4.cs – файл, в якому клас RC4Encryptor містить реалізацію алгоритму шифрування RC4 для подальшого тестування і порівняння з іншими алгоритмами;

2.6.3 Опис розроблених класів

Клас BenchmarkConfig є налаштуванням для проведення бенчмаркінгу (вимірювання продуктивності) у програмі. Він наслідується від ManualConfig, що означає, що це конфігурація, яку можна налаштовувати вручну для BenchmarkDotNet пакету. Елементи цього класу включають:

а) Конструктор BenchmarkConfig() – налаштовує різні аспекти бенчмаркінгу:

1) Діагностика Пам'яті: Додається діагностувальник пам'яті (MemoryDiagnoser), який забезпечує інформацію про використання пам'яті під час бенчмаркінгу;

2) Експортери: Додаються різні експортери через метод AddExporter(). Ці експортери визначають, в яких форматах будуть виводитися результати бенчмаркінгу (наприклад, XML, CSV, JSON тощо);

3) Статистичні Колонки: Додається колонка статистики, яка включає кількість ітерацій кожного бенчмарку за допомогою AddColumn(StatisticColumn.Iterations);

б) Властивість AdditionalExporters – є ітератором, що повертає колекцію об'єктів IExporter. Кожен IExporter представляє різні формати для експорту результатів бенчмаркінгу. До них відносяться:

1) XmlExporter.Full – експортує результати бенчмаркінгу у повному форматі XML;

2) CsvExporter.Default – виводить результати у стандартному форматі CSV (Comma-Separated Values);

3) RPlotExporter.Default – генерує скрипти та дані для візуалізації результатів бенчмаркінгу в середовищі мови програмування R. Використовується для створення графіків та діаграм, що візуально представляють результати бенчмаркінгу, зручно для аналізу та презентацій;

4) JsonExporter.Full – експортує результати бенчмаркінгу у форматі JSON (JavaScript Object Notation) у повному вигляді;

5) BenchmarkReportExporter.Default – стандартний експортер для створення звітів про бенчмарки, забезпечує детальний звіт про виконані бенчмарки, включаючи різноманітну статистику;

6) MarkdownExporter.Default та MarkdownExporter.StackOverflow – експорт результатів у форматі Markdown, з двома варіантами - стандартний Markdown та спеціально адаптований для сайту StackOverflow;

7) PlainExporter.Default – експортує результати в простому текстовому форматі, ідеально підходить для випадків, коли потрібна максимальна сумісність або для швидкого перегляду результатів без потреби в спеціальному програмному забезпеченні.

Клас CustomEncryptor являє собою реалізацію розробленого алгоритму, механізму шифрування та дешифрування. Він наслідує інтерфейс IDisposable, що дозволяє коректно відпустити ресурси, коли клас більше не використовується.

Поля класу:

– private byte[] _key – масив байтів, який використовується як ключ шифрування;

– private int _s – ціле число, що використовується для зсуву бітів вліво;

– private int _sr – ціле число, що використовується для зсуву бітів вправо (ініціалізується значенням 8 - _s);

Конструктори:

– public CustomEncryptor(byte[] key) – приймає масив байтів як ключ шифрування. Перевіряє, чи ключ не є null або порожнім, та ініціалізує змінні _s та _sr = 8 - _s;

Методи:

– public byte[] Encrypt(byte[] data) – метод шифрування масиву байтів data за допомогою ключа _key. Використовує побітові операції для модифікації даних, включаючи зсув байтів та побітову операцію XOR. Повертає зашифровані дані у формі масиву байтів;

– public byte[] Decrypt(byte[] encryptedData) – метод дешифрації масиву байтів encryptedData, який був зашифрований методом Encrypt. Проводить обернені операції до тих, що використовувались у Encrypt, для відновлення оригінальних даних. Повертає розшифровані дані у формі масиву байтів;

– private byte L(byte value) та private byte R(byte value) – внутрішні допоміжні методи для циклічного зсуву бітів вліво (L) та вправо (R);

– public void Dispose() – метод Dispose з інтерфейсу IDisposable забезпечує коректне очищення ресурсів, у цьому випадку обнулення ключа та інших змінних;

Клас EcryptionBenchmarks, призначений для вимірювання продуктивності різних алгоритмів. Він використовує бібліотеку BenchmarkDotNet, що є потужним інструментом для бенчмаркінгу коду в .NET.

Атрибути Класу:

– [Config(typeof(BenchmarkConfig))] – застосовує конфігурацію бенчмаркінгу визначену в класі BenchmarkConfig;

– [AllStatisticsColumn] – вказує на включення всіх статистичних колонок у вивід результатів бенчмарку;

– [Benchmark] – кожен метод з цим атрибутом представляє окремий тест продуктивності для конкретного алгоритму шифрування (AES, TripleDES, DES, RC2, RC4, та користувацьке шифрування);

Поля класу:

– private byte[] dataToEncrypt – масив в якому містяться дані для шифрування;

– private byte[] generalEncryptionKey, private byte[] desEncryptionKey, private byte[] tripleDesEncryptionKey – поля, масииви байтів які містять ключі для різних алгоритмів шифрування;

– [Params(...)] public int N – визначає розмір даних dataToEncrypt для шифрування;

Методи класу:

– [GlobalSetup] public void Setup() – ініціалізує дані та ключі перед кожним виконанням бенчмарку;

– public byte[] AesEncryption() – метод тестування шифрування за допомогою алгоритму AES;

– public byte[] TripleDesEncryption() – метод тестування шифрування за допомогою алгоритму TripleDES;

– public byte[] DesEncryption() – метод тестування шифрування за допомогою алгоритму DES;

– public byte[] RC2Encryption() – метод тестування шифрування за допомогою алгоритму RC2;

– public byte[] RC4Encryption() – метод тестування шифрування за допомогою алгоритму RC4;

– public byte[] CustomEncryption() – метод тестування шифрування за допомогою користувацького алгоритму шифрування;

3 ПРОВЕДЕННЯ ТЕСТУВАННЯ АЛГОРИТМУ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

Тестування та аналіз алгоритмів шифрування - найважливіший крок у визначенні їхньої ефективності та придатності для використання в реальних додатках. У цьому розділі наведено докладне порівняння відомих алгоритмів симетричного шифрування (AES, DES, 3DES, RC4 і RC2) і розробленого алгоритмів. Мета цього порівняння - виявити сильні та слабкі сторони кожного алгоритму та визначити, як ці характеристики впливають на практичне використання.

Процес тестування охоплює кілька важливих аспектів, як-от швидкість шифрування та дешифрування, використання ресурсів, таких як пам'ять. Крім того, особлива увага приділяється аналізу впливу довжини ключа на безпеку і продуктивність алгоритму.

3.1 Умови проведення тестування

3.1.1 Варіації кількості байтів шифрування (N)

Тестування виконувалося на 13 різних розмірах даних, представлених у байтах. Це дозволяє оцінити продуктивність алгоритмів на різних об'ємах даних, що є критично важливим для розуміння масштабованості та ефективності шифрування. Варіації N включають: 1 024, 2 048,4 096, 8 192, 16 384, 32 768, 65 536, 131 072, 262 144, 524 288, 1 048 576 (1 МБ), 2 097 152 (2 МБ), 4 194 304 (4 МБ), 8 388 608 (8 МБ) байт.

3.1.2 Вибрані Алгоритми Шифрування

Для тестування було обрано шість різних алгоритмів шифрування. Це включає як широко використовувані стандартні алгоритми, так і власний розроблений алгоритм:

– AES (Advanced Encryption Standard) – Широко використовуваний і рекомендований стандарт;

– DES (Data Encryption Standard) – Старіший стандарт, що все ще використовується в деяких застосуваннях;

– 3DES (Triple Data Encryption Standard) – Розширена версія DES з підвищеною безпекою;

– RC4 – Популярний алгоритм змінного потоку;

– RC2 – Блоковий шифр, що був популярним у 90-х роках;

– Розроблений алгоритм – алгоритм в основі якого лежить зсув байтів;

3.1.3 Бенчмаркінг і Статистичні Дані

Кожен алгоритм було запущено 20 разів для кожного розміру даних, що дозволило зібрати достатньо статистичних даних для оцінки продуктивності. Цей підхід забезпечує надійність отриманих результатів і допомагає виявити будь-які випадкові відхилення або нерегулярності в продуктивності.

3.2 Швидкодія алгоритму

Згідно 20 тестових запусків алгоритмів шифрування для кожного з 13 розмірів даних, було отримано такі результати по кожному алгоритму:

Таблиця 3.1

Мінімальний та максимальний час шифрування алгоритмів

Алгоритм	t_{min} шифрування	t_{max} шифрування
AES	3,0228	14265,0024
DES	36,6072	296523
3DES	36,7896	287722,92
RC4	7,3884	48593,4372
RC2	24,0096	176965,29

Продовження таблиці 3.1

Алгоритм	t_{min} шифрування	t_{max} шифрування
Розроблений	3,13	31532,7936

Примітка: t_{min}, t_{max} – час в мікросекундах.

Отже, згідно даних, як видно на таблиці 3.1, DES, 3DES та RC2 є явними аутсайдерами за мінімальним часом шифрування. Але ці дані не можна брати за початковий вектор розрахунку порівняння, через велику розбіжність в результатах і відсутності результатів аналізу по кожному розміру окремо.

Тож було в таблиці 3.2, рисунку 3.1 наведені зібрані дані по алгоритмам та на рисунку 3.2 наведено кожен алгоритм окремо, а саме – середній час серед 20 запусків шифрування кожного розміру даних:

$\overline{t_{nA}}$ = середній час виконання 1 операції шифрування (шифрування N кількості байт):

$$\overline{t_{nA}} = \frac{1}{S}\sum_{i=1}^{S} t_{nA_i} \tag{3.1}$$

де t_{nA_i} - час виконання операції під час *i* запуску операції алгоритму;

n – операція шифрування [1; 13] з кроком 1;

N – кількість байт для шифрування (14 варіантів – $[2^{10}; 2^{23}]$ з кроком степеня 1);

(n операція відповідає N кількості байт шифрування – 1 (n) = 2^{10} (N), 2 (n) = 2^{11} (N) …, 3 (n) = 2^{23}(N));

A – алгоритм шифрування в контексті якого виконується операції;

S = 20 (Кількість запусків кожної операції алгоритму для збору статистичних даних);

Таблиця 3.2

Середній час шифрування 1 операції ($\overline{t_{nA}}$)

N	Aes	3Des	Des	RC2	RC4	Custom
1024	3,07	36,79	36,80	24,01	7,39	3,15
2048	4,38	71,50	71,35	44,88	13,44	6,48
4096	7,62	140,21	140,49	86,32	24,48	12,87
8192	12,97	283,25	280,44	169,83	46,95	25,43
16384	23,72	552,34	557,02	335,07	94,62	56,29
32768	46,75	1130,03	1107,72	665,31	183,56	100,53
65536	132,89	2285,41	2241,15	1370,47	361,81	202,53
131072	375,96	4583,50	4596,17	2840,79	767,14	437,79
262144	738,88	9179,94	9272,46	5662,66	1556,64	881,26
524288	872,26	17759,86	17726,37	10763,18	3121,20	1790,98
1048576	2104,60	35697,65	35666,65	21682,77	5892,80	3577,21
2097152	4196,69	71510,02	71909,85	43417,91	11727,54	6604,96
4194304	10104,44	144956,18	143799,68	87091,22	23595,00	12999,05
8388608	14199,00	286597,53	292950,31	175109,57	47416,94	31334,82

Примітка: $\overline{t_{nA}}$ – час в мікросекундах

З таблиці видно, що швидкість шифрування зростає зі збільшенням розміру ключа. Це пов'язано з тим, що більші ключі дозволяють використовувати більш складні алгоритми шифрування, які є більш ефективними.

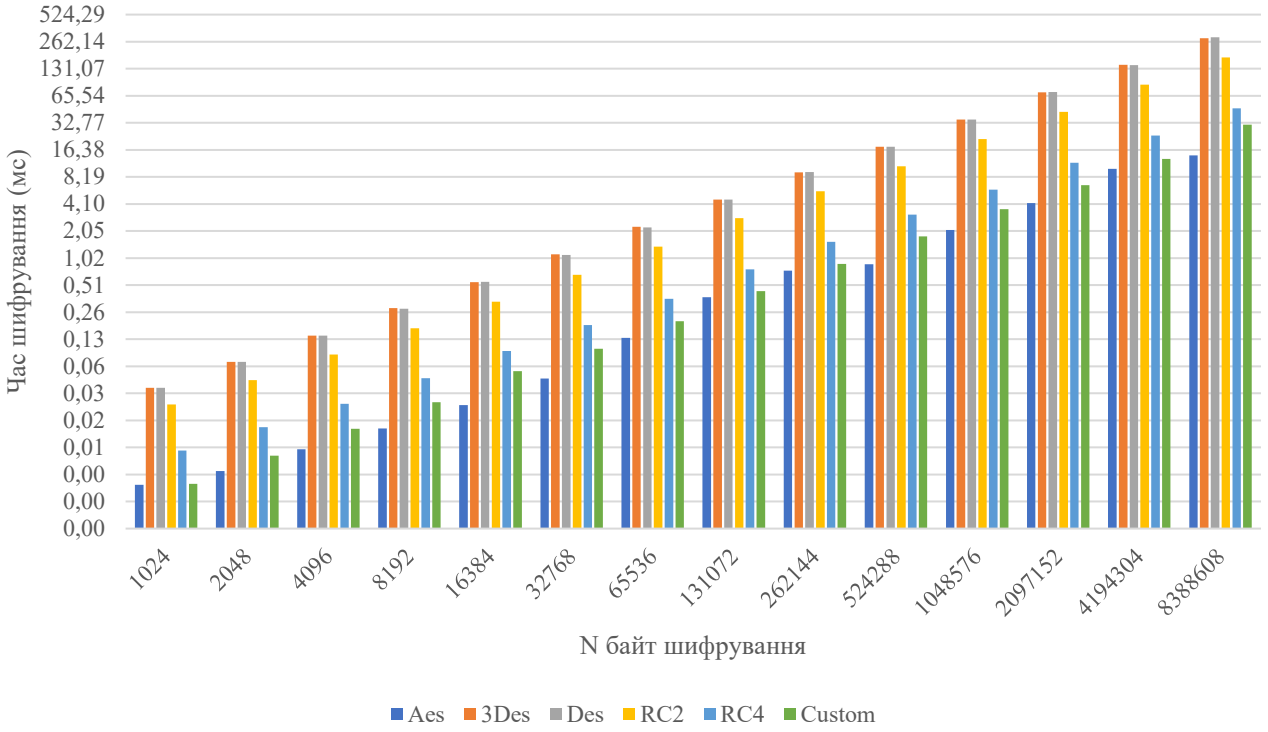


Рис. 3.1 Середній час шифрування 1 операції ($\overline{t_{nA}}$)

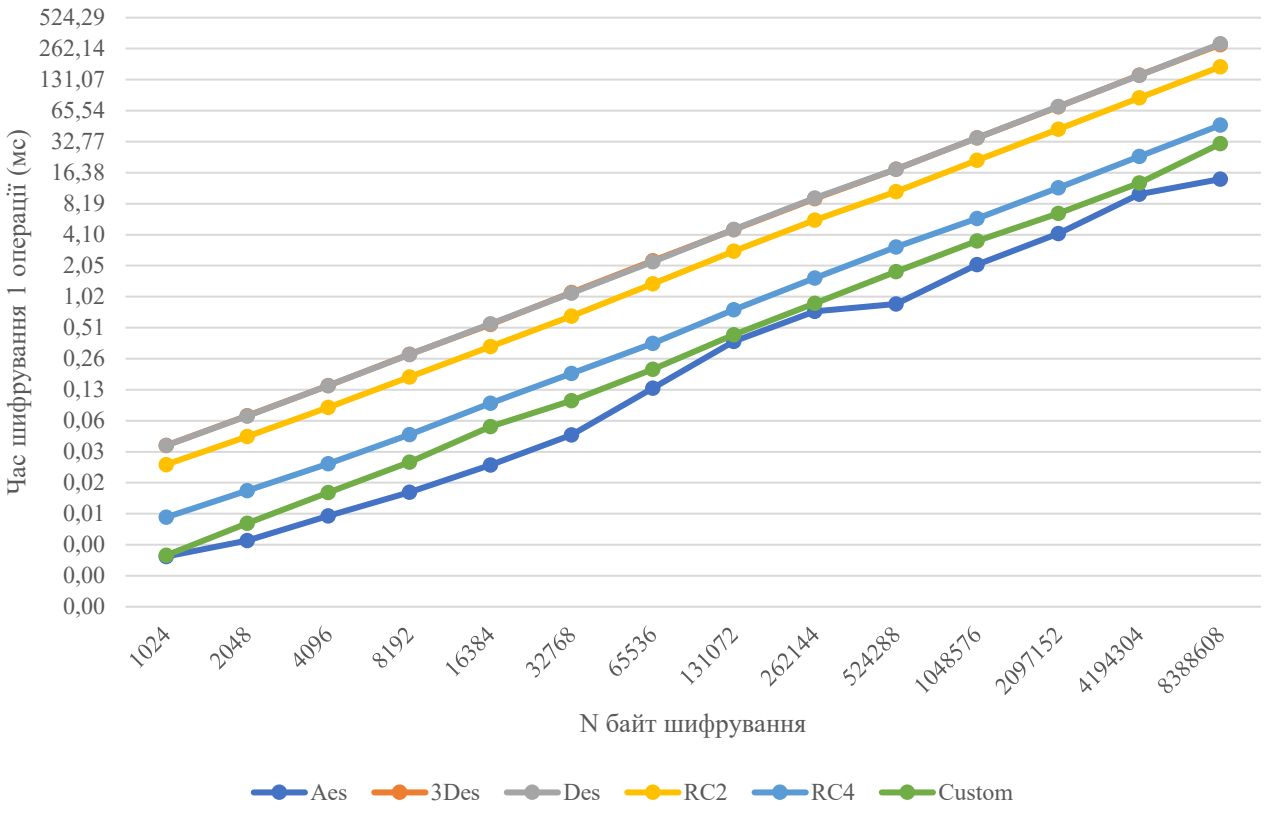


Рис. 3.2 Середній час шифрування алгоритмом 1 операції ($\overline{t_{nA}}$)

Серед алгоритмів, що тестуються, найкращу швидкість шифрування демонструє AES. Він є найновішим і найсучаснішим алгоритмом з усіх представлених. Також на початкових етапах від нього не відстає і розроблений алгоритм, навіть за умови що алгоритм AES вже максимально можливо оптимізований на рівні процесору завдяки завчасно закладених реалізацій всередині операційних систем.

Також добре себе показують алгоритми 3-DES і DES. Вони є менш ефективними, ніж AES, але все ще досить швидкими.

Алгоритми RC2 і RC4 мають найнижчу швидкість шифрування. Вони є застарілими алгоритмами, які не рекомендуються для використання в сучасних системах через проблеми з безпекою алгоритму.

3.3 Кількість виділених ресурсів на алгоритм

Згідно аналізу швидкодії алгоритмів, також потрібно зробити аналіз ресурсів, а саме кількість оперативної пам’яті, яка необхідна на виконання 1 операції шифрування задля визначення потенційної потребі в ресурсах. В подальшому на основі цих даних буде визначено ефективність алгоритмів.

На рисунку 3.3 та таблиці 3.3 представлено використання пам’яті у вигляді середньої кількості виділених ресурсів (байт пам’яті) на 1 операцію, згідно формули розрахунку:

$\overline{v_{nA}}$ = середня кількість байт (ресурсів) виділених для виконання 1 операції:

$$\overline{v_{nA}} = \frac{1}{S}\sum_{i=1}^{S} v_{nA_i} \tag{3.2}$$

де v_{nA_i} - кількість виділених кілобайт під час *i* запуску операції алгоритму;

n – операція шифрування [1; 13] з кроком 1;

N – кількість байт для шифрування (14 варіантів – $[2^{10}; 2^{23}]$ з кроком степеня 1);

(n операція відповідає N кількості байт шифрування – 1 (n) = 2^{10} (N), 2 (n) = 2^{11} (N) …, 3 (n) = 2^{23}(N));

A – алгоритм шифрування в контексті якого виконується операції;

S = 20 (Кількість запусків кожної операції алгоритму для збору статистичних даних);

Таблиця 3.3

Кількість виділених ресурсів (байт) на 1 операцію

N	Aes	3Des	Des	RC2	RC4	Custom
1024	5120	5140	5202	5089	7798	1075
2048	9216	9236	9298	9185	14841	2099
4096	17408	17428	17490	17377	26751	4147
8192	33792	33812	33874	33761	48770	8243
16384	66560	66580	66642	66540	101206	16435
32768	132106	132127	132188	132086	195238	32829
65536	263209	263229	263301	263178	375964	65597
131072	525455	525496	525558	525466	789903	131144
262144	1049897	1049958	1050020	1049938	1672269	262236
524288	2099476	2099743	2100111	2099630	3326733	524411
1048576	4197816	4198277	4198461	4198308	6101465	1048709
2097152	8390277	8392540	8392622	8391404	12116133	2097316
4194304	16778824	16780114	16779028	16779080	24379657	4194570
8388608	33555773	33558211	33556347	33556255	49759680	8388977

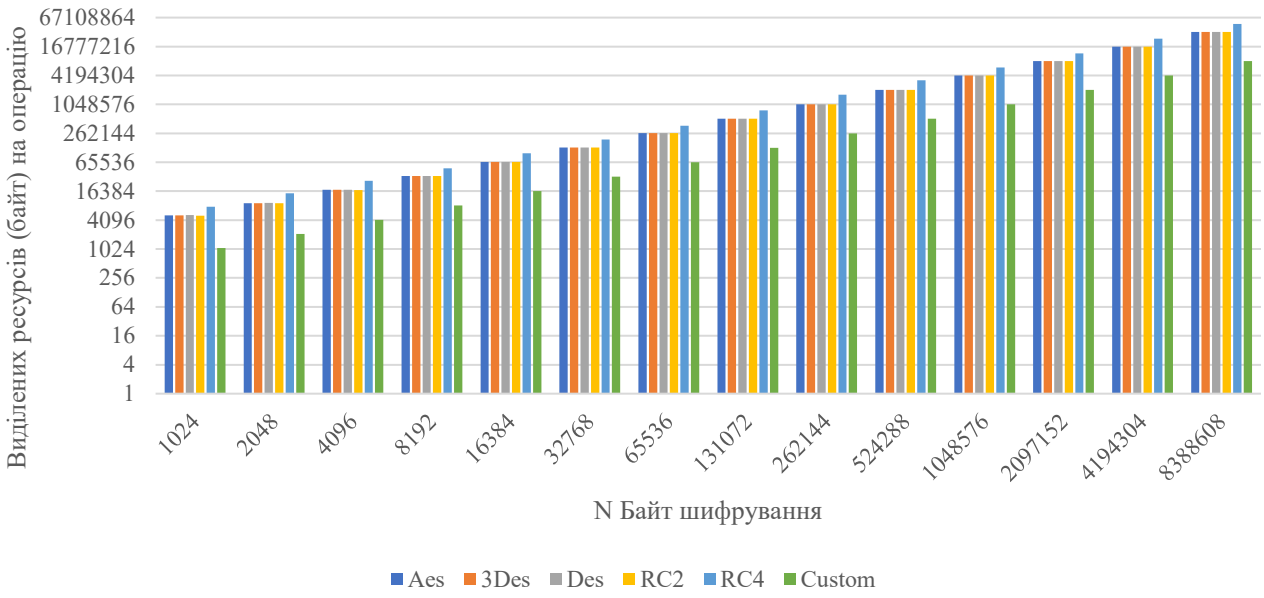


Рис. 3.3 Кількість виділених ресурсів (байт) на 1 операцію шифрування

З таблиці видно, що використання оперативної пам'яті зростає зі збільшенням довжини ключа. Це пов'язано з тим, що більші ключі вимагають більшого обсягу пам'яті для зберігання.

Серед алгоритмів, що тестуються, найбільше оперативної пам'яті споживає алгоритм AES, що показує що у контексті масштабованості цей алгоритм використовувати потрібно на машинах, які мають достатню кількість ресурсів.

Також багато оперативної пам'яті споживають алгоритми 3-DES і DES. Вони є менш ефективними, ніж AES, але все ще досить швидкими.

Алгоритми RC2 і RC4 споживають найменше оперативної пам'яті. Вони є застарілими алгоритмами, які не рекомендуються для використання в сучасних системах.

Менш за всіх споживає розроблений алгоритм, майже в 4 рази менше ніж AES, та майже у 6,5 разів менше за RC4, що показує, що розроблений алгоритм тепер переважає серед інших, хоча в розрізі швидкої він не дотягував до алгоритму AES. Тому слід провести аналіз ефективності всіх алгоритмів.

3.4 Аналіз ефективності алгоритму

Згідно попередніх результатів, потрібно проаналізувати спільний фактор швидкодії та використання пам’яті, задля визначення ефективності кожного алгоритму. Для порівняння кожного алгоритму, за основу було взято таку схему:

$\overline{t_A}$ = середній час шифрування 1 байту:

$$\overline{t_A} = \frac{1}{m}\sum \frac{\overline{t_{nA}}}{N} \tag{3.3}$$

$\overline{v_A}$ = середня кількість байт (ресурсів) виділених на шифрування 1 байту:

$$\overline{v_A} = \frac{1}{m}\sum \frac{\overline{v_{nA}}}{N} \tag{3.4}$$

де m – кількість операцій шифрування ($n_{\max}$);

P_A = показник ефективності алгоритму A

$$P_A = 1 - \overline{t_A} \cdot \overline{v_A} \tag{3.5}$$

Згідно формул 3.3, 3.4, можливо розрахувати середній час шифрування 1 байту та середню кількість байт (ресурсів) виділених на шифрування 1 байту. (рис. 3.4, 3.5)

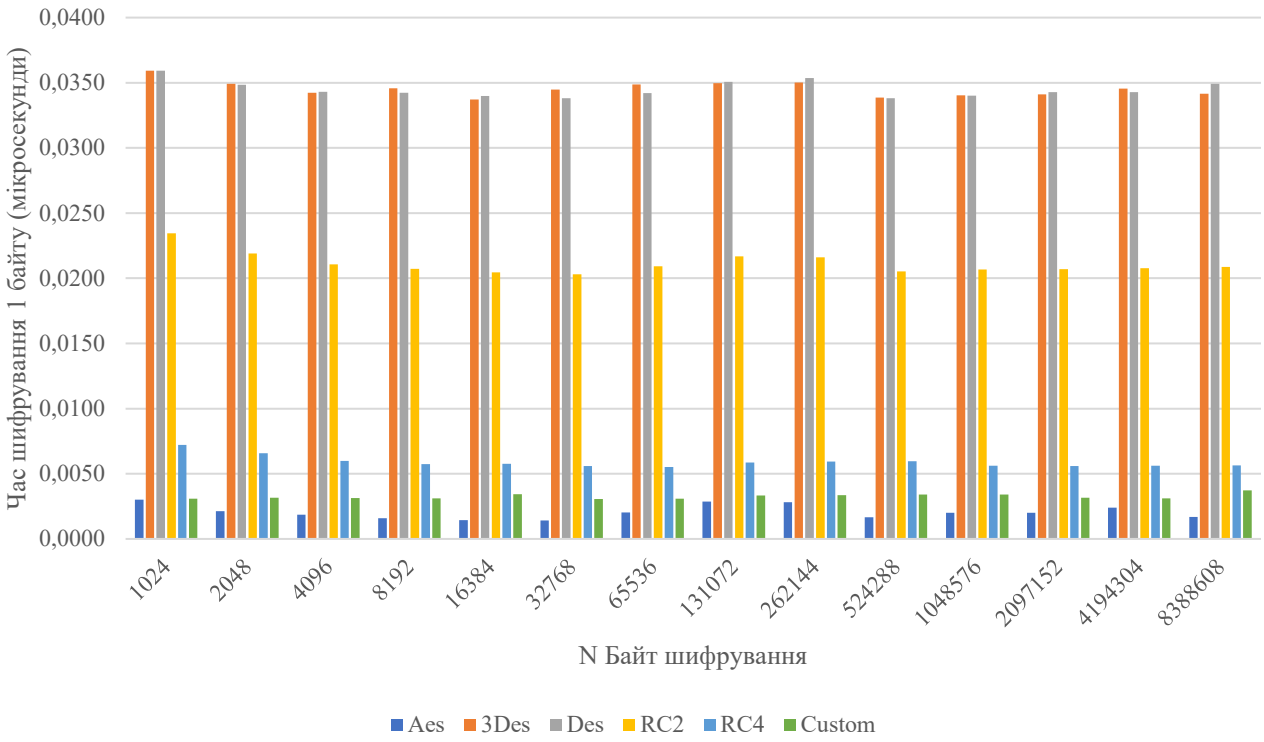


Рис. 3.4 Час шифрування 1 байту (мікросекунди)

Для точного розподілу та порівняння вирахувано середній час шифрування 1 байту по кожному алгоритму згідно даних з рисунку 3.4 (таблиця 3.4):

Таблиця 3.4

Середній час шифрування 1 байту (мікросекунди)

	Aes	3Des	Des	RC2	RC4	Custom
$\overline{t_A}$	0,002068	0,034532	0,034505	0,021118	0,005901	0,003257

За тим самому принципом, згідно формули 3.4 розраховуємо середню кількість виділених ресурсів на 1 байт (таблиця. 3.5).

Таблиця 3.5

Середня кількість байт виділених на шифрування 1 байту

	Aes	3Des	Des	RC2	RC4	Custom
$\overline{v_A}$	4,14	4,15	4,16	4,14	6,24	1,01

Отримавши результати середніх часу виконання та кількості виділених ресурсів на шифрування 1 байту, потрібно зіставити ці дані (рис. 3.5),

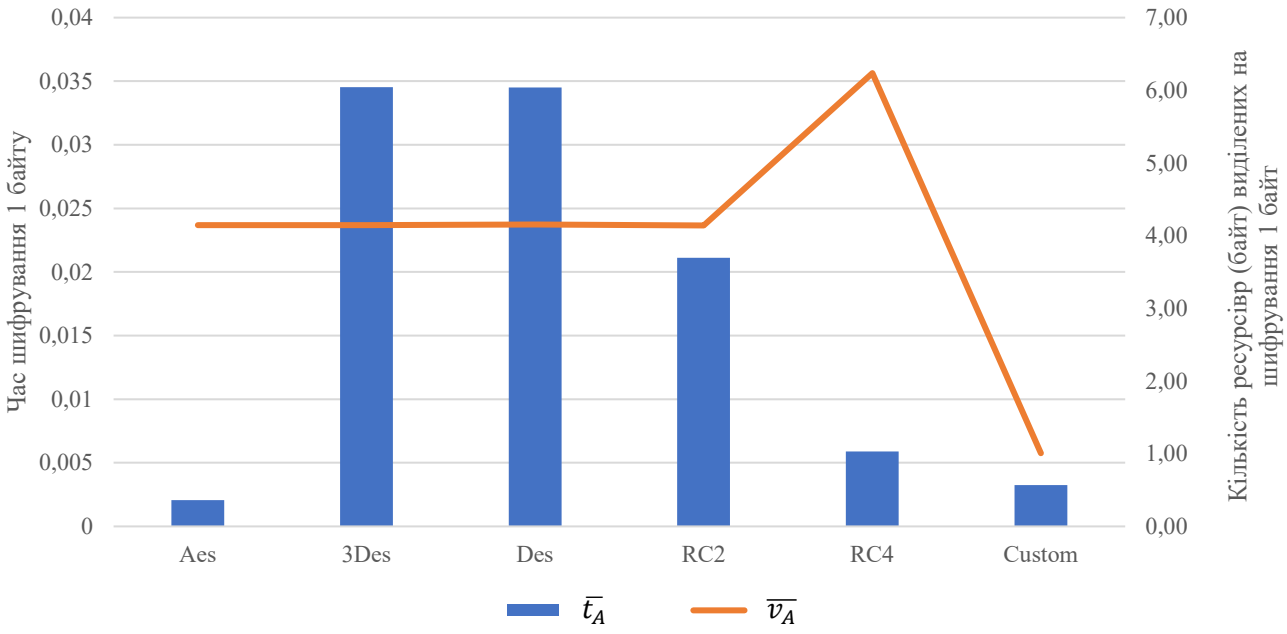


Рис. 3.5 Співвідношення середнього часу шифрування 1 байту та виділених ресурсів

Та оцінити ефективність алгоритмів згідно формули 3.5 (таблиця 3.6, рис. 3.5)

Таблиця 3.6

Показник ефективності кожного алгоритму

	Aes	3Des	Des	RC2	RC4	Custom
$\overline{v_A}$	0,991432	0,856813	0,856629	0,912583	0,963198	0,996720

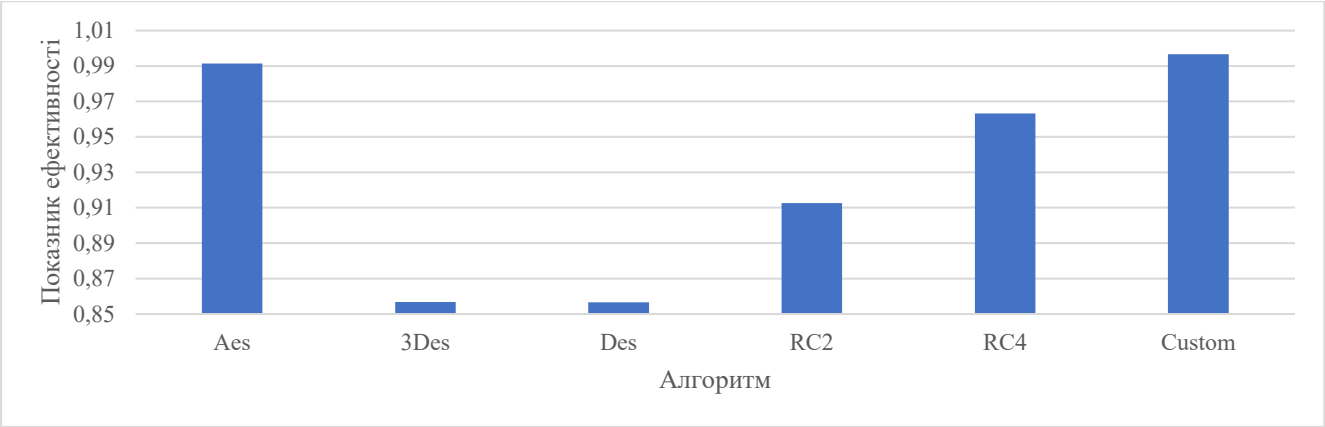


Рис. 3.6 Показник ефективності алгоритму

Згідно отриманих результатів аналізу ефективності, можна з впевненістю сказати, що розроблений алгоритм виграє у AES алгоритму (прямого конкурента) та можна вважати розроблений алгоритм кращим за порівняні алгоритми (таблиця 3.7).

Таблиця 3.7

Порівняльна таблиця розробленого алгоритму з іншими симетричними алгоритмами шифрування

Особливості	AES	DES	3DES	RC4	RC2	Розроблений
Довжина ключа	128, 196, 256 біт	56 біт	112 або 118 біт	8 – 2048 біт	До 128 біт	Від 48 біт
Розмір блоку	128 біт	64 біт	64 біт	Немає (потоковий шифр)	64 біт	Немає (потоковий шифр), але можна інтегрувати

Продовження таблиці 3.7

Особливості	AES	DES	3-DES	RC4	RC2	Розроблений
Підхід / метод	Заміна та перестановка байтів	Мережа Фейстеля	3 цикли DES	Потоковий шифр	Мережа Фейстеля	Зсув байтів, XOR операції
Швидкість	Дуже висока	Низька	Дуже низька	Висока (але не рекомендується через вразливості)	Висока	Висока, майже наближена до AES
Ефективність - $\mathbf{P}_A$	0,991432	0,856813	0,856629	0,912583	0,963198	0,996720

ВИСНОВКИ

В результаті виконання магістерської роботи було розроблено симетричний алгоритм шифрування зображень на основі зміщення байтів по код-паролю та розроблено програмне забезпечення для тестування та аналізу розробленого алгоритму.

При виконанні роботи було виконано наступні задачі:

1. Проведено аналіз алгоритмів симетричного шифрування даних та було виявлено що використання симетричних алгоритмів є більш релевантним для користувача, за рахунок використання одного ключа як для шифрування так і для розшифрування даних;

2. Розроблено алгоритм, за допомогою якого можливо впроваджувати більш гнучкі та безпечні системи зберігання та обробки зображень, за рахунок можливості інтеграції блочного шифрування та використання код-паролю необмеженої довжини починаючи з 48 біт;

3. Розроблено програмного забезпечення для проведення тестувань та аналізу розробленого алгоритму симетричного шифрування даних на основі зміщення байтів по код-паролю;

4. Проведено аналіз результатів впровадження алгоритму. Впроваджений алгоритм дозволяє збільшити ефективність шифрування зображення в середньому у 1,2 рази. Перевага алгоритму складає у меншій кількості ресурсів (пам’яті) необхідної для виконання операції шифрування у порівнянні з іншими алгоритмами шифрування;

ПЕРЕЛІК ПОСИЛАНЬ

1. Алгрен М. Що таке шифрування AES-256 і як воно працює?. *Website Rating*. URL: https://www.websiterating.com/uk/cloud-storage/what-is-aes-256-encryption/ (дата звернення: 12.12.2023).

2. Гребінников В. В. Комплексні системи захисту інформації. Проектування, впровадження, супровід : зб. лекцій. 2019. 161 с. URL: https://dspace.uzhnu.edu.ua/jspui/handle/lib/10070 (дата звернення: 17.12.2023).

3. ДСТУ 3396.1-96. ДСТУ 3396.1-96 Захист інформації. Технічний захист інформації. Порядок проведення робіт. Чинний від 1997-07-01. Вид. офіц. 6 с. URL: https://tzi.com.ua/downloads/DSTU%203396.1-96.pdf (дата звернення: 14.12.2023).

4. Ситнік А. Захищайтеся: 6 сервісів з дірками у приватності, якими ви користуєтесь кожен день. *Platfor.ma*. URL: https://www.platfor.ma/topic/zahyshhajtesya-6-servisiv-z-dirkamy-u-pryvatnosti-yakymy-vy-korystuyetes/ (дата звернення: 14.12.2023).

5. A comprehensive evaluation of cryptographic algorithms: DES, 3DES, AES, RSA and blowfish / P. Patil та ін. *Procedia computer science*. 2016. Т. 78. С. 617–624. URL: https://doi.org/10.1016/j.procs.2016.02.108 (дата звернення: 17.12.2023).

6. Alenezi M. N., Alabdulrazzaq H., Mohammad N. Q. Symmetric encryption algorithms: review and evaluation study. *International journal of communication networks and information security*. 2020. Т. 12, № 2. С. 256–272. URL: https://www.researchgate.net/publication/349324592_Symmetric_Encryption_Algorithms_Review_and_Evaluation_study (дата звернення: 17.12.2023).

7. Introduction to algorithms, fourth edition / T. H. Cormen та ін. MIT Press, 2022. 1332 с. URL: https://www.google.com.ua/books/edition/Introduction_to_Algorithms_fourth_editio/RSMuEAAAQBAJ?hl=uk&gbpv=0 (дата звернення: 13.12.2023).

8. Kwan J. This hugely popular screenshot app is a privacy nightmare. *WIRED UK*. URL: https://www.wired.co.uk/article/lightshot-chrome-screenshot-app (дата звернення: 14.12.2023).

9. Mashraki S. Lightshot phishing attack. *Medium*. URL: https://medium.com/@shahar_78444/lightshot-phishing-attacks-97498bea0e22 (дата звернення: 14.12.2023).

10. Straubinger P., Fraser G. A survey on what developers think about testing. *2023 IEEE 34th international symposium on software reliability engineering (ISSRE)*, м. Florence, Italy, 9–12 жовт. 2023 р. 2023. URL: https://doi.org/10.1109/issre59848.2023.00075 (дата звернення: 28.12.2023).

11. Stubbs R. Classification of cryptographic keys. *Cryptomathic*. URL: https://www.cryptomathic.com/news-events/blog/classification-of-cryptographic-keys-functions-and-properties (дата звернення: 17.12.2023).

12. What is an intrusion detection system (IDS)? | IBM. *IBM in Deutschland, Österreich und der Schweiz | IBM*. URL: https://www.ibm.com/topics/intrusion-detection-system (дата звернення: 15.12.2023).

13. What is information security (infosec)?. *Cisco*. URL: https://www.cisco.com/c/en/us/products/security/what-is-information-security-infosec.html (дата звернення: 12.12.2023).

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО - КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Магістерська робота

«РОЗРОБКА АЛГОРИТМУ ШИФРУВАННЯ ЗОБРАЖЕНЬ НА ОСНОВІ ЗМІЩЕННЯ БАЙТІВ ПО КОДПАРОЛЮ»

Виконав: студент групи ПДМ-64 Скуратовський Андрій Володимирович

Керівник: доктор філософії, доцент кафедри ІПЗ Дібрівний Олесь Андрійович

Київ - 2024

МЕТА, ОБ'ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: підвищення ефективності та простоти шифрування зображень за рахунок розробки симетричного алгоритму шифрування байтів.

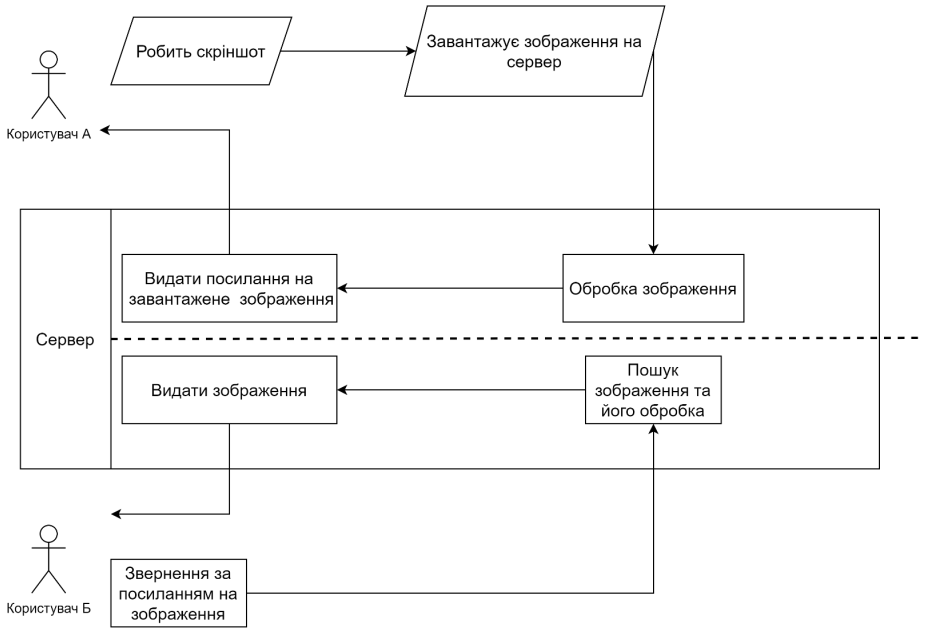
Об'єкт дослідження: процес симетричного шифрування байтів.

Предмет дослідження: методи та підходи до реалізації алгоритмів симетричного шифрування даних.

ПОРІВНЯННЯ ІСНУЮЧИХ СИМЕТРИЧНИХ АЛГОРИТМІВ

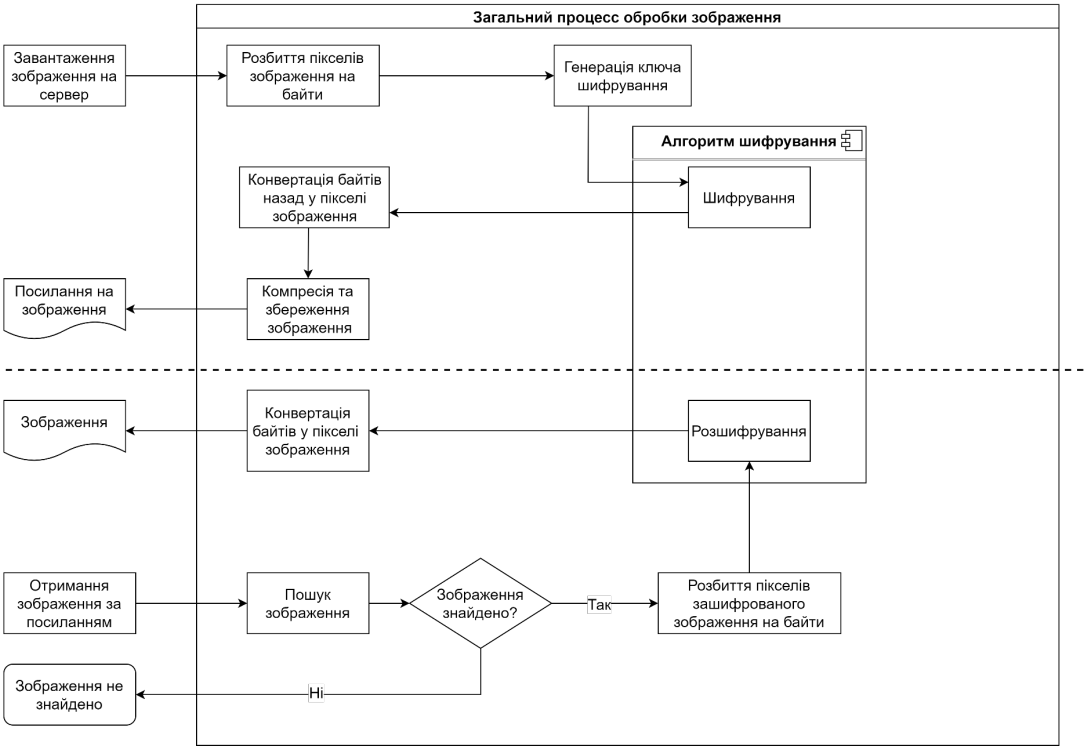
Особливості	AES	DES	3-DES	RC4	RC2	Розроблений
Довжина ключа	128, 196, 256 біт	56 біт	112 або 118 біт	8 – 2048 біт	До 128 біт	Від 48 біт
Розмір блоку	128 біт	64 біт	64 біт	Немає (потоковий шифр)	64 біт	Немає (потоковий шифр), але можна інтегрувати
Підхід / метод	Заміна та перестановка байтів	Мережа Фейстеля	3 цикли DES	Потоковий шифр	Мережа Фейстеля	Зсув байтів, XOR операції
Швидкість	Дуже висока	Низька	Дуже низька	Висока (але не рекомендується через вразливості)	Висока	Висока, майже наближена до AES
Ефективність - P_A	0,991432	0,856813	0,856629	0,912583	0,963198	0,996720

ПРОЦЕС ВЗАЄМОДІЇ КОРИСТУВАЧА



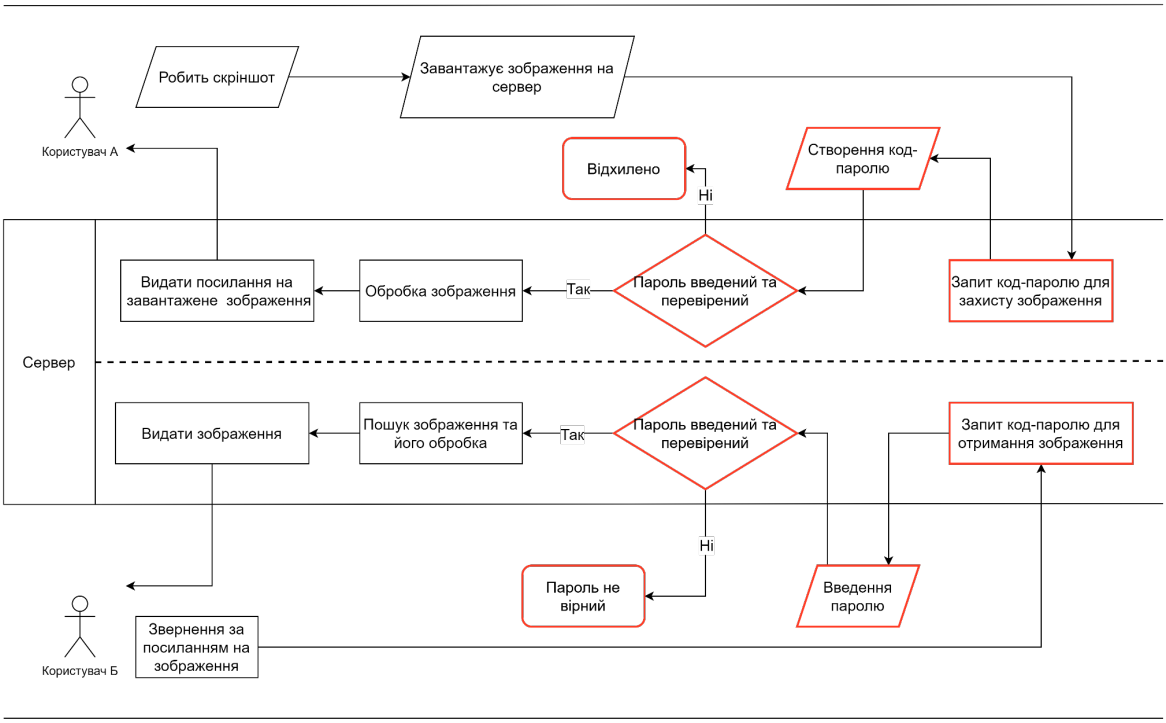
4

ПРОЦЕС ОБРОБКИ ЗОБРАЖЕННЯ



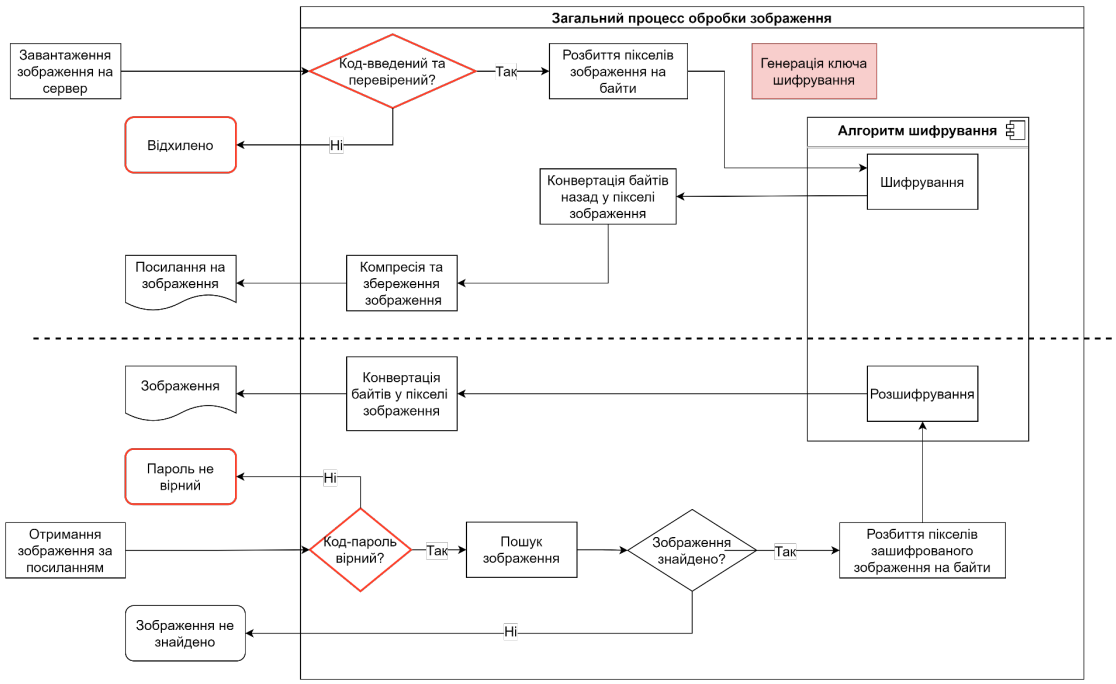
5

МОДИФІКОВАНИЙ ПРОЦЕС ВЗАЄМОДІЇ КОРИСТУВАЧА



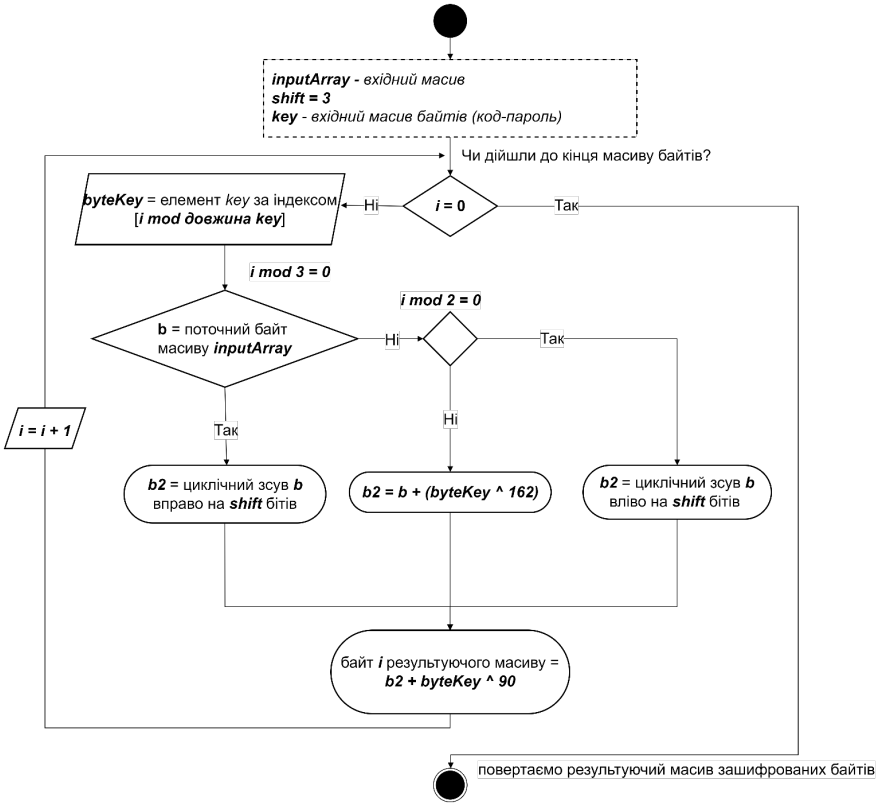
6

МОДИФІКОВАНИЙ ПРОЦЕС ОБРОБКИ ЗОБРАЖЕННЯ



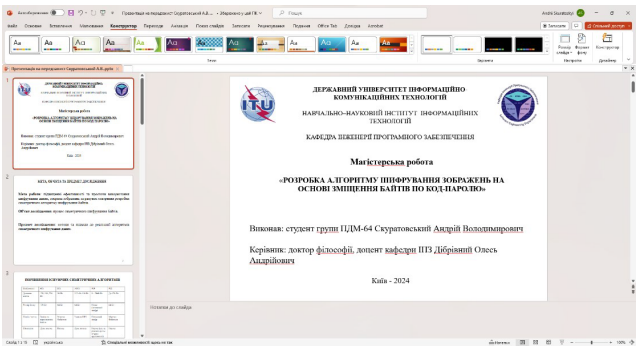
7

АЛГОРИТМ ШИФРУВАННЯ ЗОБРАЖЕНЬ



8

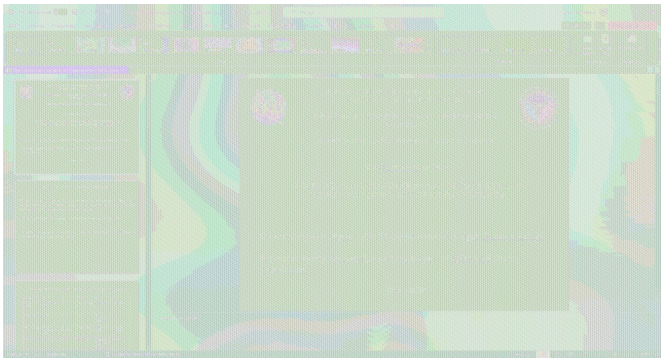
ПРИКЛАД ШИФРУВАННЯ ЗОБРАЖЕННЯ ПО КОД-ПАРОЛЮ



Вхідне зображення

Код-пароль:
BB#VJ(fEqQgr%on9z#%EHM0E@#91

Вихідне (зашифроване) зображення



9

РОЗРАХУНОК ЯКОСТІ АЛГОРИТМУ

$\overline{t_{nA}}$ = середній час виконання 1 операції шифрування (шифрування N кількості байт):

$$\overline{t_{nA}} = \frac{1}{S}\sum_{i=1}^{S} t_{nA_i} \quad (1)$$

$\overline{v_{nA}}$ = середня кількість байт (ресурсів) виділених для виконання 1 операції :

$$\overline{v_{nA}} = \frac{1}{S}\sum_{i=1}^{S} v_{nA_i} \quad (2)$$

де t_{nA_i} - час виконання операції під час *i* запуску операції алгоритму;

v_{nA_i} - кількість виділених кілобайт під час *i* запуску операції алгоритму;

n – операція шифрування [1; 13] з кроком 1;

N – кількість байт для шифрування (14 варіантів – $[2^{10}; 2^{23}]$ з кроком степеня 1);

(n операція відповідає N кількості байт шифрування – 1 (n) = 2^{10} (N), 2 (n) = 2^{11} (N) …, 3 (n) = 2^{23}(N) *);*

A – алгоритм шифрування в контексті якого виконується операції;

*S = 20 (*Кількість запусків кожної операції алгоритму для збору статистичних даних);

РОЗРАХУНОК ЯКОСТІ АЛГОРИТМУ

$\overline{t_A}$ = середній час шифрування 1 байту:

$$\overline{t_A} = \frac{1}{m}\sum \frac{\overline{t_{nA}}}{N} \quad (3)$$

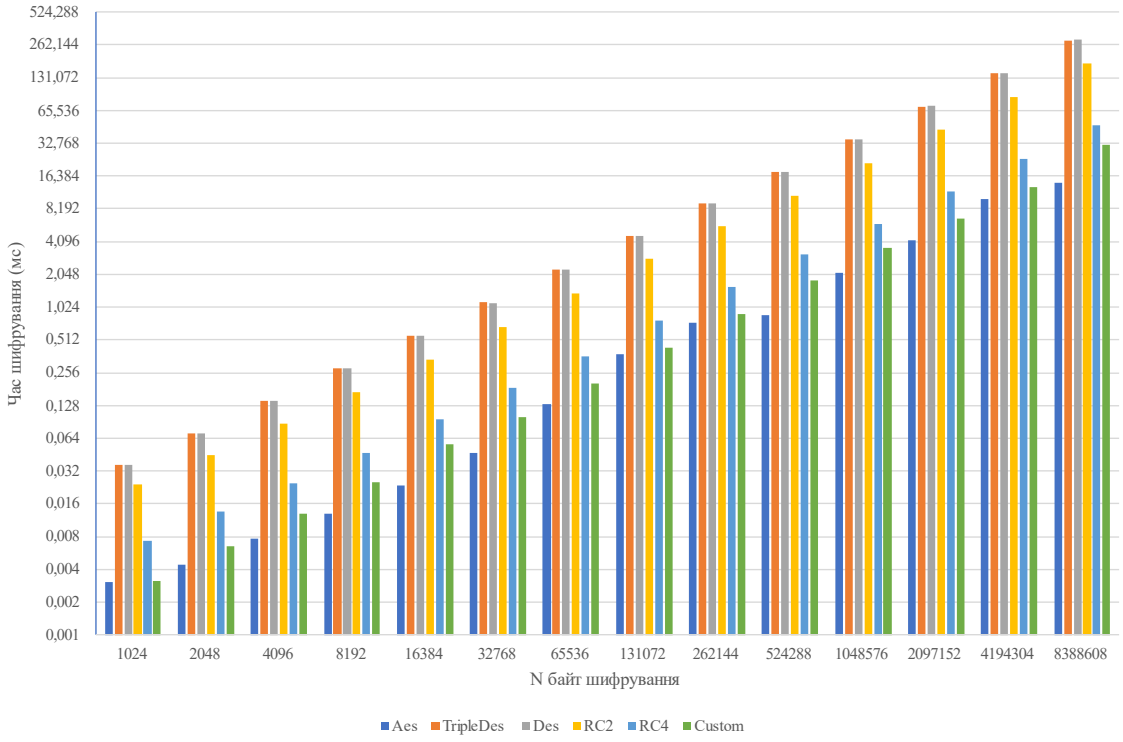
$\overline{v_A}$ = середня кількість байт (ресурсів) виділених на шифрування 1 байту:

$$\overline{v_A} = \frac{1}{m}\sum \frac{\overline{v_{nA}}}{N} \quad (4)$$

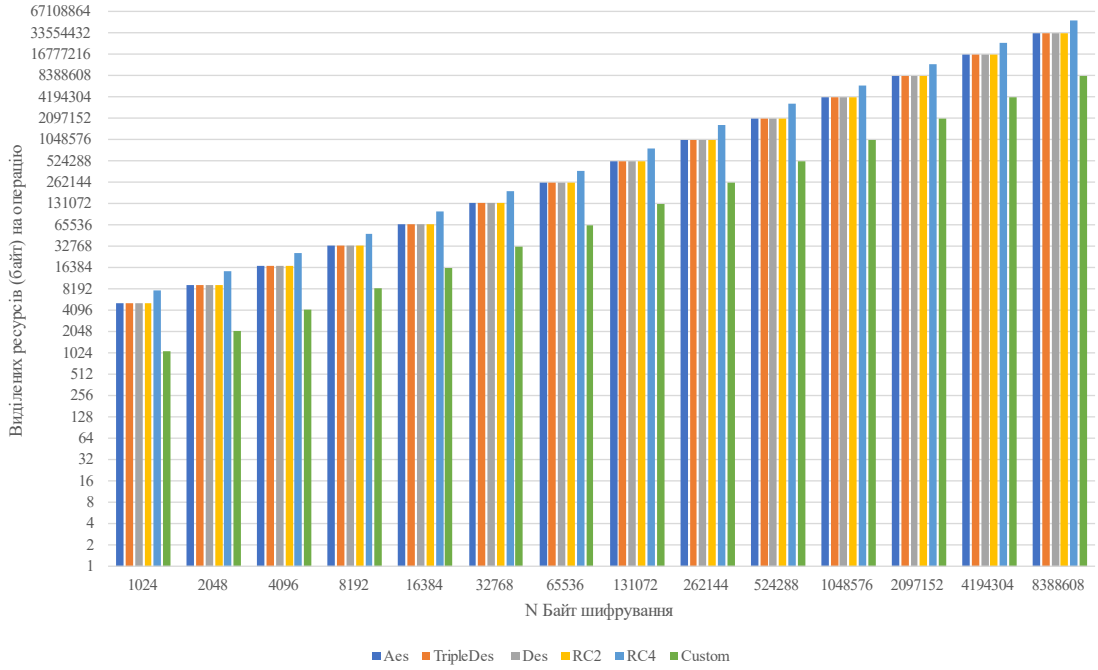
де *m* – кількість операцій шифрування ($n_{\max}$)

$P_A =$ показник ефективності алгоритмуA

$$P_A = 1 - \overline{t_A} \cdot \overline{v_A} \quad (5)$$

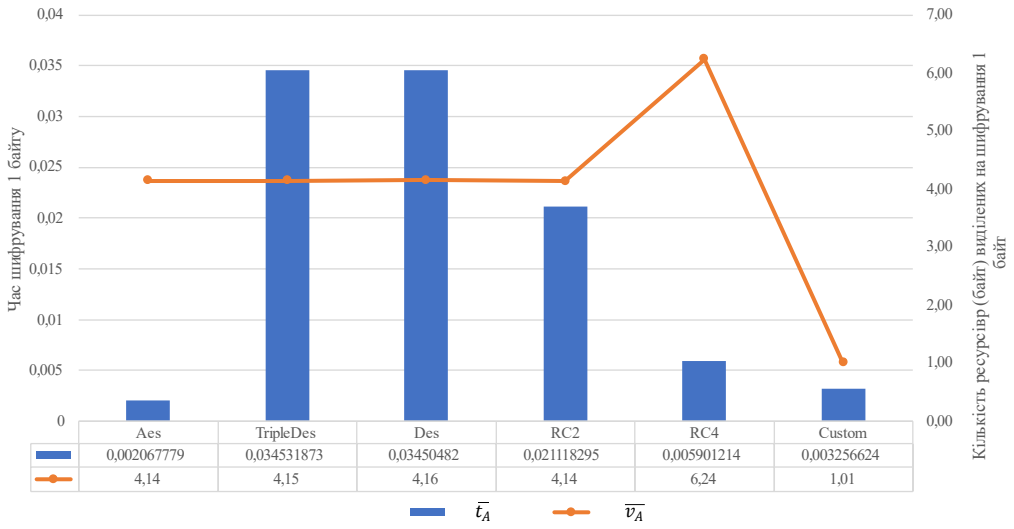

ПОРІВНЯЛЬНИЙ АНАЛІЗ ШВИДКОСТІ АЛГОРИТМІВ
Час шифрування (мс)
524,288
262,144
131,072
65,536
32,768
16,384
8,192
4,096
2,048
1,024
0,512
0,256
0,128
0,064
0,032
0,016
0,008
0,004
0,002
0,001
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
1048576
2097152
4194304
8388608
N байт шифрування
Aes
TripleDes
Des
RC2
RC4
Custom

ПОРІВНЯЛЬНИЙ АНАЛІЗ АЛГОРИТМІВ ЗА ПОКАЗНИКОМ ВИКОРИСТАННЯ ПАМ'ЯТІ



13

ПОРІВНЯЛЬНИЙ АНАЛІЗ ЕФЕКТИВНОСТІ АЛГОРИТМІВ



	Aes	TripleDes	Des	RC2	RC4	Custom
P_A	0,991432	0,856813	0,856629	0,912583	0,963198	0,996720

14

ВИСНОВКИ

1. Проведено аналіз алгоритмів симетричного шифрування даних та було виявлено що використання симетричних алгоритмів є більш релевантним для користувача, за рахунок використання одного ключа як для шифрування так і для розшифрування даних.

2. Розроблено алгоритм, за допомогою якого можливо впроваджувати більш гнучкі та безпечні системи зберігання та обробки зображень, за рахунок можливості інтеграції блочного шифрування та використання код-паролю необмеженої довжини починаючи з 48 біт.

3. Розроблено програмного забезпечення для проведення тестувань та аналізу розробленого алгоритму симетричного шифрування даних на основі зміщення байтів по код-паролю.

4. Проведено аналіз результатів впровадження алгоритму. Впроваджений алгоритм дозволяє збільшити ефективність шифрування зображення в середньому у 1,2 рази. Перевага алгоритму складає у меншій кількості ресурсів (пам’яті) необхідної для виконання операції шифрування у порівнянні з іншими алгоритмами шифрування.

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Тези доповідей:

1. Скуратовський А.В. «Забезпечення безпеки та конфіденційної інформаційних технологій хмарних сховищ зберігання скриншотів» // V Всеукраїнська конференція молодих учених «Актуальні питання розвитку інформаційних технологій – 2023», (27.11.2023), Державний вищий навчальний заклад “Приазовський державний технічний університет”, Маріуполь, Україна. (збірка поки ще не опублікована)
2. Скуратовський А.В. «Метод зміщення байтів, як метод симетричного шифрування зображень» // VI Міжнародної наукової конференції «Науковий простір: актуальні питання, досягнення та інновації» (15.12.2023), “Міжнародний центр наукових досліджень”, м. Київ, Україна (прийнято до публікації)

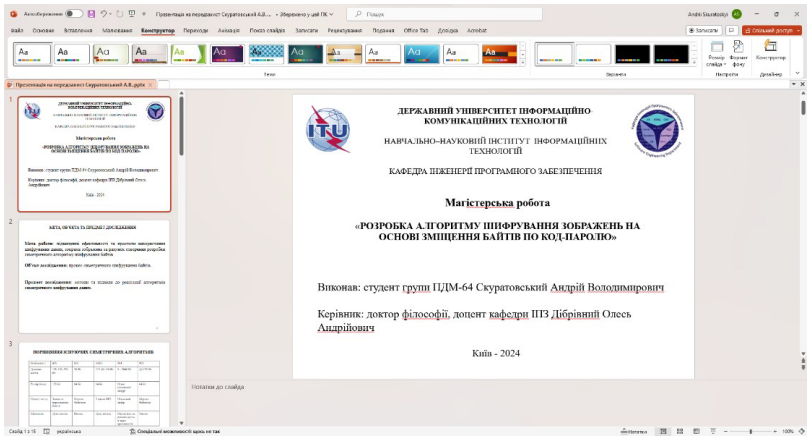
Стаття:

Дібрівний О.А., Скуратовський А.В. «Використання методу зміщення байтів, як алгоритм симетричного шифрування зображень» // Журнал «Зв’язок» Державний університет інформаційно-телекомунікаційних технологій, м. Київ, Україна (подано на розгляд комісії)

ДЯКУЮ ЗА УВАГУ!

ДОДАТОК А

ПРИКЛАД ВИКОРИСТАННЯ РОЗРОБЛЕНОГО АЛГОРИТМУ



Вхідне зображення

Код-пароль:
BB#VJ(fEqQgr%on9z#%EHM0E@#91

Вихідне (зашифроване) зображення