Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА:

на тему: «Метод шифрування даних на основі їх характеристичних ознак»

Виконав: студент 1 курсу групи 2БС-22м
спеціальності 125 - Кібербезпека
_____Назарій Гуцуляк

Керівник: д. т. н., проф. зав. каф. ЗІ
_____Володимир ЛУЖЕЦЬКИЙ
«18» 12 2023 р.

Опонент: д.т.н., проф., зав. каф. ПЗ
_____Олександр РОМАНЮК
«18» 12 2023р.

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н., проф.
_____ Лужецький В. А.
«18» 12 2023 р.

Вінниця ВНТУ – 2023

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти II (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 125 Кібербезпека
Освітньо-професійна програма – Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри ЗІ, д. т. н., проф.
_____Володимир ЛУЖЕЦЬКИЙ
«19» 09 2023 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Гуцуляку Назарію Олеговичу

1. Тема роботи: «Метод шифрування даних на основі їх характеристичних ознак.»

Керівник роботи: Лужецький Володимир Андрійович , зав. каф. ЗІ , д. т. н., проф., затверджені наказом ректора ВНТУ від 18 вересня 2023 року №247.

2. Строк подання студентом роботи 18 грудня 2023 року

3. Вихідні дані до роботи:

Метод шифрування – на основі характерестичних ознак байтів;

Характеристичні ознаки –код байту даних по модулю 8;

Розрядність ключа – 64 біт;

Мова програмування - C#;

Обсяг файлу що підлягає шифруванню – не більше 10 мб.

4. Зміст текстової частини: Вступ. 1. Аналіз інформаційних джерел. 2. Розробка методу шифрування. 3. Розробка програмного засобу. 4. Економічна частина. Висновки. Перелік використаних джерел. Додатки.

5. Перелік ілюстративного матеріалу: Мета та задачі магістерської кваліфікаційної роботи (плакат А4). Схема процесу зашифрування (плакат А4). Схема процесу розшифрування (плакат А4). Байтові операції для зашифрування та розшифрування (плакат А4). Результат тестування

лавинного ефекту (плакат А4).. Сформований пакет із ... методом шифрування (плакат А4).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Лужецький В.А., зав. каф. ЗІ , д. т. н., проф.	19.09	25.09
2	Лужецький В.А., зав. каф. ЗІ , д. т. н., проф.	19.09	10.10
3	Лужецький В.А., зав. каф. ЗІ , д. т. н., проф.	19.09	26.10
4	Ольга РАТУШНЯК, к.т.н., доц. каф ЕВПМ	19.09	27.11

7. Дата видачі завдання 1 вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	01.09.2023 – 10.09.2023	
2	Аналіз літературних джерел за напрямком магістерської кваліфікаційної роботи	10.09.2023 – 15.09.2023	
3	Науково-технічне обгрунтування	16.09.2023 – 22.09.2023	
4	Розробка технічного завдання	23.09.2023 – 29.09.2023	
5	Розробка рішень	30.09.2023 – 12.10.2023	
6	Практична реалізація, моделювання, експериментування, результати	14.10.2023 – 10.11.2023	
7	Розробка розділу економічного обгрунтування доцільності розробки	11.11.23 – 17.11.2023	
8	Аналіз виконання ТЗ, висновки	18.11.2023 – 24.11.2023	
9	Оформлення пояснювальної записки	25.11.2023 – 30.11.2023	
10	Попередній захист та доопрацювання МКР	28.11.2023 – 01.12.2023	
11	Перевірка магістерської роботи на наявність плагіату	02.12.2023 – 10.12.2023	
12	Представлення МКР до захисту	11.12.2023 – 14.12.2023	
13	Захист МКР	14.12.2023 – 21.12.2023	

Студент _____ Назарій ГУЦУЛЯК

Керівник роботи _____ Володимир ЛУЖЕЦЬКИЙ

АНОТАЦІЯ

Магістерська робота присвячена розробці методу шифрування даних на основі їх характеристичних ознак.

В першому розділі було проведено комплексний аналіз існуючих методів шифрування даних. Наведено приклади застосування цих методів у сучасних системах з урахуванням їхньої ефективності та потенційних обмежень.

Другий розділ присвячено розробці нового методу шифрування, базованого на характеристичних ознаках. Описано технічні аспекти методу, його унікальні особливості та потенційні переваги порівняно з існуючими методами. Наведено деталі його реалізації та можливості його використання в різних сферах.

У третьому розділі представлена практична реалізація розробленого методу шифрування даних на основі характеристичних ознак. Описано програмний засіб, що реалізує розроблений метод шифрування.

Четвертий розділ містить аналіз економічних аспектів розробленого методу шифрування. Обговорюються витрати на реалізацію та впровадження, можливість зменшення витрат часу чи ресурсів завдяки новому методу, а також його конкурентоспроможність на ринку і потенційні переваги для різних секторів.

ABSTRACT

The master's thesis is devoted to the development of a method of data encryption based on their characteristic features.

In the first section, a comprehensive analysis of existing data encryption methods was conducted. Examples of the application of these methods in modern systems are given, taking into account their effectiveness and potential limitations.

The second chapter is devoted to the development of a new encryption method based on characteristic features. The technical aspects of the method, its unique features and potential advantages compared to existing methods are described. The details of its implementation and the possibilities of its use in various areas are given.

The third section presents the practical implementation of the developed data encryption method based on characteristic features. The software tool that implements the developed encryption method is described.

The fourth section contains an analysis of the economic aspects of the developed encryption method. Implementation and implementation costs, the possibility of reducing time or resource costs due to the new method, as well as its competitiveness in the market and potential benefits for different sectors are discussed.in the market and potential benefits for different sectors are discusse.

ЗМІСТ

ВСТУП

Багато сфер діяльності сучасного суспільства залежать від комп’ютерних технологій. У зв’язку з цим гостро постає питання конфіденційності інформації.

У світі, де наші персональні дані, банківські рахунки, особиста інформація знаходяться в електронному вигляді є великий ризик не зберегти їх конфіденційність. У багатьох сферах, від фінансів до медицини, конфіденційні дані стають все більш вразливими перед загрозами кібернападів, тому потреба у безпекових шифрах постійно зростає.

Наразі, безпека інформації є не менш важливою, ніж сама інформація. З постійними загрозами кібератак і зростанням об'ємів цифрової інформації, пошук нових, більш надійних методів шифрування залишається важливим. Нові блокові шифри можуть допомогти у вдосконаленні стандартів криптографії, забезпечуючи більшу безпеку для зберігання та передачі даних. Розвиток хмарних технологій та розумних систем підкреслює потребу у шифруванні даних на рівні пристроїв та передаваних інформаційних потоків. Законодавство щодо захисту даних, ставить більший акцент на захист особистих даних, змушуючи компанії та організації активніше застосовувати ефективні методи шифрування. Саме тому розробка даного застосунку є актуальною.

Об’єкт дослідження – процес шифрування даних.

Предмет дослідження – метод пришвидшеного шифрування даних та засіб, що реалізує його.

Метою роботи є пришвидшення процесу шифрування даних шляхом перетворення байтів на основі операцій, що керуються характеристичною ознакою даних.

Наукова навизна – вперше запропоновано метод шифрування даних, який відрізняється від відомих тим, що використовує перетворення байтів на основі операцій, що керуються характеристичною ознакою даних і забезпечує пришвидщення процесу шифрування.

Практичну цінність має програмний засіб, що реалізує запропонований метод пришвидшеного шифрування даних.

1 АНАЛІЗ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1.1 Основні поняття

У світі сьогодення інформація вважається цінним активом, а гарантування її конфіденційності та цілісності стає важливим завданням. Шифрування відіграє ключову роль у захисті інформації від несанкціонованого доступу та різних загроз. Процес шифрування даних полягає в перетворенні відкритого тексту в нерозбірливий шифрований варіант за допомогою певних алгоритмів та ключа. Такий захист дозволяє зберегти конфіденційність та цілісність інформації [1].

Відкритий текст (Plaintext) є початковою інформацією, яка піддається шифруванню [3]. Шифрований текст (Ciphertext) представляє собою результат застосування шифру до відкритого тексту за допомогою ключа, що робить його нерозбірливим для сторонніх осіб.

Ключ (Key) використовується в алгоритмі шифрування для гарантування унікальності та безпеки шифру. Це може бути випадково згенероване число або послідовність символів. Важливо добре захищати ключ, оскільки він визначає можливість розшифрування даних.

Історія шифрування налічує тисячоліття і почалася з необхідності захистити конфіденційність важливих повідомлень та даних. Шифр Цезаря вважається одним із перших відомих методів шифрування, де кожна буква тексту замінювалася іншою, зсуваючи її на певну кількість позицій у алфавіті [4].

Історично розвивалися різні методи шифрування, включаючи шифри Віженера, Шифр Атбаш, шифр Аффіна та багато інших. Проте, з підвищенням складності сучасних атак, які використовують статистичний аналіз, з'явилися більш потужні та складні методи шифрування.

У 20 столітті відбувся значний прогрес у криптографії, особливо під час Другої світової війни. Однією з найбільш відомих подій було розкриття німецького коду Енігма союзниками. Це значно підштовхнуло розвиток

криптографії та створення нових методів шифрування [5].

Значення шифрування даних у сучасному світі важко переоцінити. Однією з його ключових переваг є здатність захищати конфіденційність інформації, чи то в мережі, на сховищі або під час передачі. Несанкціонований доступ до цієї інформації може мати серйозні наслідки, включаючи втрату фінансових активів, порушення приватності та можливість зловживання та крадіжки особистих даних [6].

Шифрування також є важливим для забезпечення інформаційної безпеки у підприємств, урядових організацій та для особистих користувачів. Враховуючи поширення кіберзлочинності та доступу до Інтернету, важливість збереження цілісності даних підкреслює необхідність ефективних методів шифрування, що можуть запобігти несанкціонованому доступу та атакам на інформаційні системи.

Все це підтверджує, що шифрування даних залишається однією з найважливіших складових інформаційної безпеки в сучасному світі, де приватність та безпека інформації є надзвичайно цінними ресурсами.

1.2 Блокові шифри та принципи їх побудови

Блокові шифри належать до типу симетричних шифрів, які операційно використовують набір блоків з фіксованою кількістю бітів із відкритого тексту. Ці шифри обробляють дані блоками фіксованої довжини і використовують певний алгоритм шифрування для кожного блоку, при цьому для кожного блоку використовується свій ключ. В сучасний час багато криптографічних алгоритмів ґрунтуються на блокових шифрах [7].

Важливо відзначити, що всі повідомлення, зашифровані тексти та ключі представлені у вигляді послідовностей бітів. Блоковий шифр може бути описаним як перетворення, що має форму:

$$Ek : Vn \times K \rightarrow Vn,$$

де Vn — n-бітові рядки, які ми трактуємо як вхідні повідомлення (ВТ),

K — множина ключів (зазвичай це множина бітових рядків іншої довжини),

Vn — множина всіх ШТ.

Блокові шифри - це тип симетричних шифрів, які операцію шифрування чи розшифрування виконують над фіксованими блоками даних однакового розміру. Основні принципи їх побудови включають:

Режими шифрування: Це способи застосування блокових шифрів для обробки довільних кількостей даних. Найпоширеніші режими: ECB (Electronic Codebook), CBC (Cipher Block Chaining), CTR (Counter), OFB (Output Feedback), CFB (Cipher Feedback) [6].

Блочні перетворення: Блокові шифри використовують перетворення над блоками фіксованого розміру. Це може бути замішання, змішування бітів та змішування ключа.

Ключ: Важливо використовувати надійні ключі для блокових шифрів. Якщо ключ легко передбачити, це може вразливості шифру.

Підтвердження цілісності: Деякі блокові шифри включають в себе методи для підтвердження цілісності даних, такі як MAC (Message Authentication Code), які забезпечують не лише конфіденційність, а й автентифікацію.

Розмір блоку: Розмір блоку важливий для блокових шифрів. Зазвичай використовуються блоки розміром 64 або 128 бітів.

Необхідність випадковості: Деякі режими блокових шифрів вимагають випадкових значень, таких як IV (Initialization Vector), для запобігання повторних шифрувань однакових блоків даних [6].

Блокові шифри є важливою складовою у шифруванні даних в багатьох сферах, включаючи кібербезпеку, захист інформації та забезпечення конфіденційності. Вони використовуються для шифрування повідомлень, файлів, баз даних та інших даних, що потребують захисту.

Режими шифрування - це способи застосування блокових шифрів для обробки довільних кількостей даних.

ECB (Electronic Codebook): Цей режим ділить вхідне повідомлення на блоки однакового розміру і кожен блок шифрується незалежно. Проте це може призвести до проблеми, коли однакові блоки даних у вихідному повідомленні будуть шифруватися однаково, що може викликати зміни у вихідних даних.

CBC (Cipher Block Chaining): У цьому режимі перед шифруванням кожен блок даних комбінується з попереднім за допомогою операції XOR. Крім того, використовується IV (Initialization Vector), який додає випадковість до першого блоку. Це дозволяє уникнути проблеми з однаковими блоками даних, які могли виникнути в ECB.

CTR (Counter): У цьому режимі кожен блок шифрується за допомогою значення лічильника, яке комбінується з ключем для генерації псевдовипадкової послідовності. Ця послідовність потім використовується для шифрування блоків даних. Один з його головних плюсів - можливість паралельного шифрування, що полегшує обробку даних [7].

OFB (Output Feedback): У цьому режимі блок шифрується, а результат XOR-иться з вхідним текстом, щоб згенерувати шифртекст. Потім використовується зміщення для генерації наступного шифртексту. OFB може бути використаний для створення потокового шифру.

CFB (Cipher Feedback): Цей режим схожий на OFB, але він використовує шифрований блок як зсув для генерації наступного шифртексту, який потім XOR-иться з вхідним текстом для створення шифртексту.

Ці режими надають можливість використовувати блокові шифри для шифрування довільних кількостей даних, забезпечуючи високий рівень захисту та безпеки. Вибір конкретного режиму залежить від потреб системи та конкретних вимог до захисту інформації.

Блочні перетворення є основним елементом блокових шифрів, які використовуються для шифрування даних у фіксованих блоках. Ось кілька ключових аспектів блочних перетворень:

Субституція (Substitution): Це процес заміни бітів або байтів на інші за певними правилами. Зазвичай, ця заміна базується на ключі шифрування і відбувається відповідно до конкретного алгоритму. Це дозволяє змішувати й змінювати дані для унікального шифрування.

Перестановка (Permutation): Це переміщення бітів у заданих позиціях. Перестановка виконується відповідно до ключа шифрування, що змішує дані у блоках, щоб ускладнити процес розшифрування без ключа.

Раунди (Rounds): Багато блочних шифрів використовують послідовність операцій, які називають раундами. Кожен раунд може включати комбінацію субституції та перестановки, змінюючи стан блоку даних кожного разу [7].

Функції змішування (Mixing Functions): Це операції, які забезпечують додаткове перемішування даних у межах блоку. Вони допомагають ускладнити структуру блока та розподілити вплив ключа шифрування по всьому блоку.

Операції над ключем (Key Operations): Ключ шифрування використовується для перетворення блоку даних. Операції з ключем можуть включати XOR, зсуви, підстановки та інші маніпуляції для впливу на блок.

Зворотні перетворення (Inversion Transformations): Блочні перетворення також мають можливість розшифрування даних. Вони використовують обернені операції для відновлення оригінального блоку з шифрованого.

Блочні перетворення враховують важливі аспекти безпеки, такі як змішування даних, використання ключа шифрування та операцій над блоками даних для забезпечення ефективного та безпечного шифрування.

Використання випадкових значень, таких як Initialization Vector (IV),

є критично важливим для деяких режимів блокових шифрів. IV - це додатковий параметр, який використовується разом з ключем для шифрування, і його основна мета полягає у запобіганні виникнення однакових шифртекстів при шифруванні однакових блоків даних.

Основна необхідність використання IV полягає в тому, що при шифруванні в тих же самих умовах, з однаковим ключем, блоковий шифр може виводити однаковий шифртекст для однакових блоків вхідних даних. Це вразливість, яка може допомогти атакувачеві отримати додаткову інформацію про вміст даних [6].

IV вирішує цю проблему, оскільки він вводить випадковість у процес шифрування. Кожен блок даних шифрується з використанням свого унікального IV, навіть якщо вміст блоків однаковий. Це перешкоджає виведенню однакового шифртексту для ідентичних блоків даних, оскільки шифрування відбувається з врахуванням унікальності IV.

Основні принципи побудови та властивості блокового шифрування:

Функція відображення Ek повинна бути взаємно однозначною (бієктивною) для всіх $k \in K \Rightarrow$ для будь-якого k: $Dk \equiv E$-1k, де Dk — відкрите повідомлення (ВТ);

Ключі мають оптимальну довжину: вони повинні бути достатньо короткими для зручного використання, але досить довгими, щоб забезпечити стійкість;

Застосування принципів Шеннона, таких як розсіювання та перемішування;

Шифри повинні відповідати криптографічним властивостям стійкості, таким як нелінійність, імунітет до кореляцій, ефект лавини та інші;

Шифрування виконується ітеративно. На рис.1.1 схемі шифрування показана загальна схема, де (k1, k2, ..., kr) = KS(k) — результат ключового розкладу, F — раундове шифруюче перетворення.

$$Y = E\text{k}(\text{x}) = F\text{k}\ (F\text{kr-1}\ (...F\text{k1}\ (\text{x})...))$$

перетворення має бути оберненим самому до себе (інволютивним), це означає, що: $\forall x : \phi(\phi(x)) = ;$

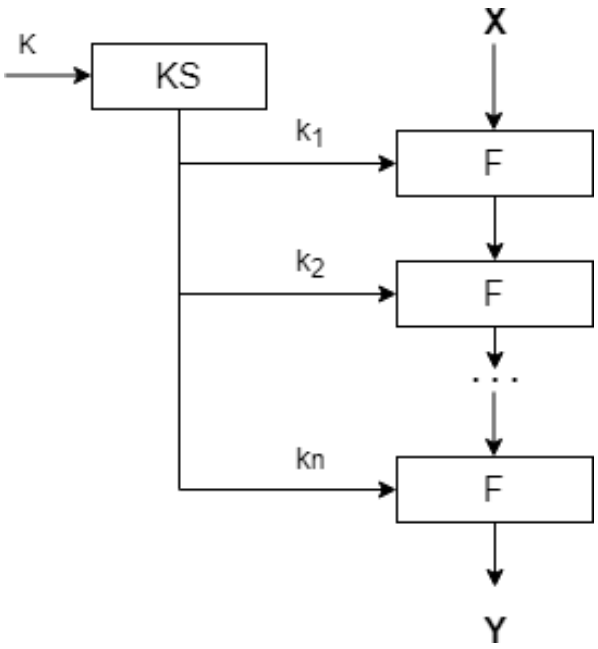


Рисунок 1.1 – Загальна схема ітеративного бловокого шифру.

У процесі шифрування виконується наступна послідовність операцій. Представимо K як випадковий двійковий ключ певної фіксованої довжини. Цей ключ використовується для створення N раундових ключів, які позначаються як $K1$, $K2$, ..., KN . Сукупність цих раундових ключів ($K1$, $K2$, ..., KN) є ключовим розкладом, створеним з K за допомогою певних фіксованих та загальноприйнятих алгоритмів [8].

Раундова функція, позначена як f, приймає два параметри: раундовий ключ Kr та поточний стан, позначений як $w\ r{-}1$. Наступний стан обчислюється як $w\ r = f(w\ r{-}1\ ,\ Kr\ )$, де $w\ r{-}1$ визначається як початковий стан "x", що є вхідним відкритим текстом. Після N раундів шифрування шифротекст "y" отримується шляхом послідовних ітераційних перетворень вхідного відкритого тексту. Таким чином, процес шифрування відбувається в наступній послідовності:

$$x \rightarrow w^0$$

$$f(w^0, K^1) \rightarrow w^1$$

$$f(w^1, K^2) \rightarrow w^2$$

$$\dots$$

$$f(w^{N-1}, K^N) \rightarrow w^N$$

$$w^N \rightarrow y$$

Щоб мати можливість розшифрувати, функція f повинна бути ін'єктивною, що означає: для будь-яких значень w та y, $f^{-1}(f(w, y), y) = w$. Таким чином, процес розшифрування виконується наступним чином:

$$y \rightarrow w^N$$

$$f^{-1}(w^N, K^N) \rightarrow w^{N-1}$$

$$\dots$$

$$f^{-1}(w^2, K^2) \rightarrow w^1$$

$$f^{-1}(w^1, K^1) \rightarrow w^0$$

$$w^0 \rightarrow x$$

1.3 SP-мережа

SP-мережа, або Substitution-Permutation Network (SPN), є класом блокових шифрів, що складаються з раундів, у яких повторюються ряд математичних операцій. Один з основних алгоритмів, який ґрунтується на SP-мережі, це AES. Кожен раунд шифрування має структуру, що включає три етапи: додавання ключа, нелінійну заміну бітів та лінійну перестановку бітів, як зображено на рис.1.2. Розшифрування виконується у зворотньому напрямку [9].

"SP-мережа" (Substitution-Permutation Network) - це структура, що використовується в блочних шифрах для створення шифруючої функції. Ця

конструкція полягає у поєднанні двох основних операцій: субституції (заміни) та перестановки (переміщення).

Основна ідея SP-мережі полягає в тому, що вона розбиває блоки даних на менші частини, які потім проходять через послідовність раундів, кожен з яких включає в себе операції заміни та переміщення (перестановки).

- Субституція (Substitution): Ця операція заміни замінює вхідні біти або байти за певними правилами на основі ключа та структури шифру. Це дозволяє змішувати та перетворювати дані в кожному раунді, ускладнюючи процес розшифрування без ключа.
- Перестановка (Permutation): Це операція переміщення бітів у заданих позиціях. Перестановка також відбувається відповідно до ключа шифрування та специфікацій шифру.

SP-мережі застосовуються у багатьох сучасних блочних шифрах, таких як AES (Advanced Encryption Standard) та DES (Data Encryption Standard). Вони надають ефективний спосіб захисту даних шляхом послідовної обробки блоків даних через раунди з операціями заміни та переміщення [9].

Ця конструкція дозволяє створювати сильні шифри, оскільки змішування та перестановка даних у кожному раунді ускладнюють аналіз шифру та здійснення атак. Важливою є правильна реалізація операцій субституції та перестановки, а також ефективне керування ключем для надійного шифрування та розшифрування даних.

Розглянемо ближче принцип роботи SP-мережі. Нехай l та m - цілі натуральні числа. Відкритий текст (ВТ) та зашифрований текст (ШТ) є двійковими векторами довжини lm (тобто, lm - довжина блоку шифрування). SP-мережа складається з двох компонент: πs та πp.

$$\pi_s : \{0,1\}^l \to \{0,1\}^l,$$

Перестановка $2l$ бітових рядків довжини l виражає процес перенесення, або перестановки, двох послідовностей бітів, кожна з яких має довжину

$$l.\pi_p : \{0,\ldots,l\mathrm{m}\} \to \{0,\ldots,l\mathrm{m}\},$$

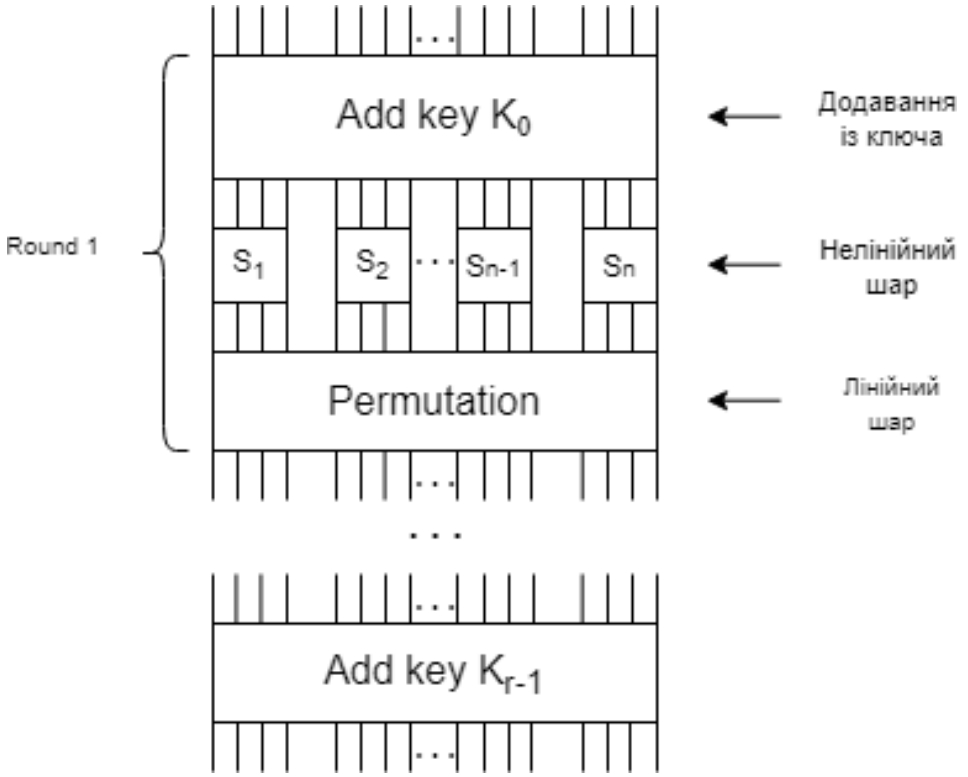


Рисунок 1.2 – Загальна схема шифрування SP-мережі.

SP-мережа допомагає ускладнити процес розшифрування без ключа, оскільки вона використовує послідовність операцій, які перетворюють вхідні блоки у шифртекст з урахуванням ключа шифрування. Використання SP-мережі дозволяє створювати шифри з високим рівнем захисту даних [9].

Перестановка цілих чисел від 1 до lm представляє собою процес перенесення, або перестановки, послідовності чисел в діапазоні від 1 до lm. Це може бути перестановка порядку цифр у послідовності чи зміна їх розташування.

Перестановка πs відома як S-блок, що використовується для заміни l бітів на інший набір бітів довжиною l. У той час, πp використовується для перестановки lm бітів, змінюючи їх порядок. Якщо розглядати lm-бітовий двійковий рядок, наприклад, x = (x1, x2, . . . , xlm), ми можемо розглядати

x як конкатенацію з m l-бітових підрядків, які позначимо як x<1>, x<2>, . . . , x<m> [10]. Таким чином,

$$x = x_{<1>} \parallel x_{<2>} \parallel \cdots \parallel x_{<m>}, \text{ де}$$

$$x_{<i>} = (x_{(i-1)\,l+1}, x_{(i-1)+2}, \ldots, x_{il}), \ = \overline{1,m}$$

SP-мережа складається з N раундів, в кожному з яких (окрім останього) виконується m замін, використовуючи π_s, після чого йде перестановка π_p. Перед кожною операцією заміни буде відбуватися додавання раундового ключа. Зазвичай, це проста операція XOR. Розглянемо алгоритм шифрування SP-мережі на прикладі шифру Serpent [11]. На вході ми маємо ВТ x, на виході маємо отримати ШТ z (Рис. 1.3).

Алгоритм шифрування SP-мережі на прикладі шифру Serpent складається з N раундів, кожен з яких (за винятком останнього) виконує множину замін, використовуючи πs, після чого слідує перестановка πp. Перед кожною операцією заміни здійснюється додавання раундового ключа, яке зазвичай представлене як проста операція XOR.

Serpent - це симетричний блоковий алгоритм шифрування, спроектований Россом Андерсоном, Елі Біхамом і Ларсом Кнудсеном. Він був одним із п'ятьох фіналістів конкурсу Advanced Encryption Standard (AES), проведеного Національним інститутом стандартів і технологій (NIST) для визначення нового стандарту шифрування. Розмір блоку становить 128 біт, а можливі довжини ключа - 128, 192 або 256 біт. Всі операції можуть бути виконані паралельно, використовуючи 32 1-бітових "потоки" [12].

Кроки шифрування включають початкову перестановку, 32 раунди, кожен з яких складається з:

- Операції змішування з 128-бітовим ключем
- Виключного АБО
- Табличної підстановки (S-box)
- Лінійного перетворення

У останньому раунді лінійне перетворення замінюється додатковим накладанням ключа. Після чого відбувається кінцева перестановка.

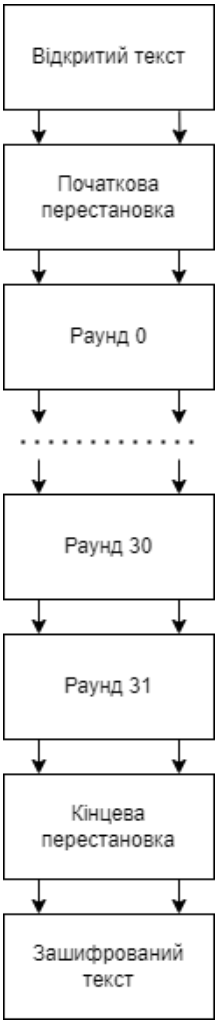


Рисунок 1.3 – Схема роботи шифру Serpent.

Додавання ключа як перша та остання операція в SP-мережі відоме як "відбілювання". Це вважається корисним методом для запобігання зловмисникам виконувати операції шифрування або дешифрування, якщо ключ невідомий [13].

SP-мережі мають декілька привабливих особливостей. По-перше, їх дизайн є простим і дуже ефективним як у апаратному, так і у програмному забезпеченні. У програмному забезпеченні зазвичай S-блоки реалізуються у вигляді пошукових таблиць.

1.4 Схема Фейстеля

Схема Фейстеля, відома як шифр Фейстеля, була названа на честь Хорста Фейстеля, який розробив її, працюючи у компанії IBM. У 1973 році він та його колега Дон Копперсміт опублікували шифр під назвою "Люцифер", який став першим публічним прикладом шифрування з використанням структури Фейстеля [14]. Пізніше на основі Люцифера був розроблений шифр DES, що став національним стандартом шифрування в 1977 році. Раундове шифрування схеми Фейстеля можна побачити наступним чином рис. 1.4.

Основна ідея схеми Фейстеля полягає у тому, що вхідний блок даних розділяється на дві частини, які проходять через послідовність раундів, які включають в себе функцію заміни, функцію перестановки та операції з ключем. Після кожного раунду блоки обмінюються місцями і знову обробляються.

Кожен раунд схеми Фейстеля містить кілька операцій.Функція заміни (Substitution Function): Вона отримує частину блока даних та ключ шифрування і виконує операцію заміни бітів чи байтів у вхідному блоку.

Функція перестановки (Permutation Function): Ця операція переміщує біти чи байти в певних позиціях вхідного блоку відповідно до конкретних правил.

Операції з ключем: Ключ шифрування використовується для зміни вхідного блоку на кожному раунді.

Після завершення всіх раундів, обидві частини блоку об'єднуються, створюючи шифртекст. При розшифруванні процес відбувається у зворотньому порядку, із тим же самими ключами, але в іншому порядку.

Схема Фейстеля є важливим компонентом багатьох блочних шифрів та дозволяє створювати шифри з високим рівнем безпеки та стійкості.

У схемі Фейстеля кожен стан розділяється на дві рівні частини: Li та Ri (від left та right). Раундова функція має такий вигляд: f (Li-1, Ri-1, Ki), де

Li-1 та Ri-1 - це ліва та права половини попереднього стану, а Ki - це

раундовий ключ, що використовується на поточному раунді. Функція f оперує цими половинами разом з раундовим ключем для створення нових значень Lі та Rі.

$$L^i = R^{i-1}$$

$$R = L^{i-1} \oplus f(R^i - 1, K^i)$$

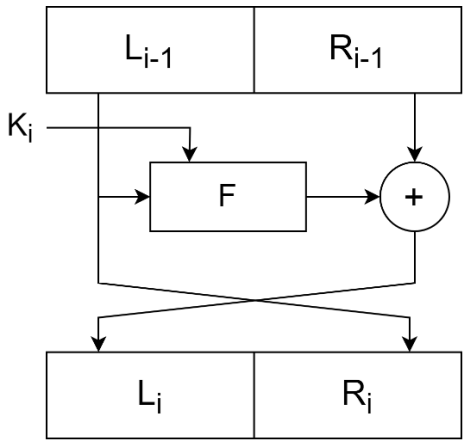


Рисунок 1.4 – Схема раунднового шифрування у шифрі Фейстеля.

Під час кожного раунду вхідний текст розділяється на дві однакові частини. Раундова функція f, що залежить від раундового ключа Kі, використовується для однієї з цих половин вхідного тексту, після чого результат застосовується до другої половини за допомогою операції XOR (зазвичай це операція додавання чи інша логічна операція). Після цього обидві частини тексту міняються місцями. У фінальному раунді ця операція міняння місцями здійснюється двічі, для покращення безпеки шифру.

$$L^i = R^{i-1}$$

$$R = L^{i-1} \oplus f(R^i - 1, K^i)$$

Схема Фейстеля має кілька важливих властивостей. Шифрування лише половини повідомлення в одному раунді виявляється корисним, оскільки реалізація функції від половини бітів є ефективнішою, ніж від усієї довжини повідомлення. Це сприяє покращенню продуктивності та швидкодії алгоритму.

Як і у SP-мереж, схема Фейстеля є інволютивною, тобто процедура розшифрування відрізняється від процедури шифрування тільки порядком застосування раундових ключів.

1.4 Міжнародні стандарти блокових шифрів

Стандарти вкрай необхідні в сучасном житті, щоб забезпечувати бажані характеристики виробів чи послуг, такі як якість, безпеку, надійність, ефективність та інше. Якщо вироби, частини машини й прилади працюють добре, і безпечно, то це часто відбувається тому, що вони задовольняють вимогам стандартів [17]. І організацією, відповідальною за багато тисяч стандартів, що приносять користь всьому світу, є ISO (*International Organization for Standardization*).

Міжнародна організація зі стандартизації (ISO) — це найбільший у світі розробник і видавець міжнародних стандартів. В даний час у світі діє близько 2000 різних стандартів, випущених Міжнародною організацією стандартизації, які застосовуються як у вузькоспеціалізованих областях, так і в міжнаціональному масштабі[18]. Зокрема, вищезгадана система ISO/IEC 18033-3 використовується в якості стандартів блочних шифрів, які були визначені у 2005 році, переглядались у 2020, і станом на 2023 рік є актуальними Серед цих шифрів є:

– 64-бітові блочні шифри: TDEA, MISTY1, CAST-128, HIGHT;

– 128-бітові блочні шифри: AES, Camellia, SEED.

TDEA. Цей блоковий шифр був розроблений Уітфілдом Діффі, Мартіном Хеллманом та Уолтом Тачманном у 1987 році на основі алгоритму DES. Даний шифр побудований за схемою Фейстеля. TDEA також ще називають Triple DES, 3DES або TDES. Швидкість його роботи втричі нижча, ніж швидкість DES, але усувається головний недолік — коротка довжина ключа, через який шляхом повного перебору можна дешифрувати повідомлення [19].

Шифрування TDEA виконується за схемою Фейстеля на трьох ключах, кожен з яких довжиною в 56 бітів, які можуть бути незалежні між собою, співпадати або серез них буде лише одна пара однакових ключів. Варіант, коли три ключа не мають співпадінь між собою є найбільш надійним. На кожному ключі виконується по 16 раундів.

MISTY1. Даний блоковий шифр розробив Міцуру Міцуї з командою спеціалістів для компанії Mitsubishi Electric у 1995 році. Розшифровується як Mitsubishi Improved Security Technology. Шифрування відбувається за схемою Фейстеля, де довжина ключа складає 128 біт, а довжина блоку -64 біт. Кількість раундів может бути будь-якою, але має бути кратною чотирьом, хоча рекомендується використовувати 8 раундів шифрування. MISTY1 був одним з обраних алгоритмів, які шифрують 64-бітні блоки, у европейському проекті NESSIE.

CAST-128. Алгоритм був створений Карлайлом Адамсом та Стаффордом Таваресом у 1996 році. Даний шифр складається з 12 або 16 раундів шифрування на основі мережі Фейстеля. Розмір блоку становить 64 біт, а довжина ключа может бути від 40 до 128 біт, але має бути кратною 8. Якщо довжина ключа перевищує 80 біт, використовують 16 раундів шифрування. Є три різні типи раундових функцій, які схожі між собою, але відрізняються застосовуваними операціями: додавання, віднімання або *XOR*. Використовується у багатьох продуктах, зокрема, за замовчуванням як шифр у деяких версіях GPG і PGP. Він також був схвалений для використання урядом Канади [20].

HIGHT. Розроблений південнокорейськими криптографами у 2006 році. Довжина ключа дорвнює 128 біт, а довжина блоку 64. Реалізація даного алгоритму не потребує великих апаратних ресурсів, щоб його було легко застосовувати у обчислювальних приладах, таких як датчик в USN або RFID-мітка. HIGHT має 32-раундову ітераційну структуру, яка є варіантом узагальненої мережі Фейстеля. Визначною особливістю цього шифру є те, що він складається з простих операцій, таких як *XOR*,

додавання за модулем 2^8 і циклічний зсув вліво.

AES. Цей блоковий шифр був запропонований бельгійськими криптографами Вінсентом Рейменом та Джоаном Даменом на конкурсі NIST на заміну шифру DES і став переможцем. AES (Advanced Encryption Standard) також відомий під назвою Rijndael. Розмір блокудорівнює 128 біт, а довжина ключа може бути 128, 192 або 256 біт. В залежності від довжии ключа, кількість раундів буде 10, 12 або 14 відповідно. Працює на основі SP-мережі. На кожному раунді застосовуються такі операції: AddRoundKey (додавання раундового ключа), SubBytes (ненілійна заміна), ShiftRows (перестановка бітів у стані), MixColumns (лінійна заміна). У останьому раунді лінійна заміна не виконується. Даний алгоритм має ефективну реалізацію на різних архітектур [21].

Camellia. Алгоритм був розроблений у 2000 році японськими компаніями Mitsubishi Electric та NTT для європейського конкурсу NESSIE та став фіналістом. Як структуру, використовує мережу Фейстеля з 18 або 24 раундами та довжиною блоку в 128 біт. Довжина ключа залежить від кільності раундів: 128 біт (для 18 раундів шифрування) та 192 або 256 біт (для 24 раундів). Раундова функція використовує нелінійне перетворення, блок лінійного розсіювання кожні 16 циклів (побайтова операція XOR) і байтову перестановку [22].

SEED. Даний шифр був створений Корейським агентством інформаційної безпеки (KISA) у 1998 році.Він широко використовується в промисловості Південної Кореї, але рідко зустрічається десь ще. Він набув популярності в Кореї, тому що 40-бітове шифрування не вважалосядосить надійним, тому Корейське агентство інформаційної безпеки розробило свій власний стандарт. SEED є мережею Фейстеля з довжиноюключа в 128 біт та блоком розміром у 128-біт. Шифрування виконується за 16 раундів [23]. Серед усього різноманіття алгоритмів та методів криптографічного захисту виділяються ітеративні блокові шифри, які будуються на основі SP-мережі або схеми Фейстеля.

Їм характерний певний ряд позитивних властивостей: розшифрування не потребує окремої схеми чи процедури, тому що виконується у зворотньому напрмку від процедури зашифрування; алгоритми ітеративного блокового шифрування можуть бути реалізовані як на пристроях з певним обмеженням у ресурсах, так і на приладах, які цих обмежень не мають, що говорить про їх універсальність; не мають надскладної будови для реалізації; є 7ітеративних блокових шифрів, що входять до списку міжнародних стандартів за версією ISO.

Усі ці фактори вказують на ефективність та актуальність даного виду шифрування. Підсумовуючи вище написане, можна зробити висновок, що розвиток та аналіз ітеративних блоковихшифрів є важливим та перспективним.

1.5 Постановка завдання

Отже проаналізувавши відомі методи шифрування, можна сформувати вимоги для створення власного методу.

По-перше, розробити та детально описати алгоритм шифрування даних на основі їх характеристичних ознак.

Для розробки методу шифрування потрібно:

- Визначити розмір блоку та можливих довжин ключа.
- Вибрати тип характеристичної ознаки, яка буде використовуватися та операцій що будуть виконуватись під час шифрування.
- Розробити алгоритми для вилучення характеристичних ознак з даних.
- Розробити алгоритми для шифрування та дешифрування даних на основі їх характеристичних ознак.

По-друге, реалізувати програмний засіб, що буде вміщувати роботу розробленого шифру. Під час реалізації необхідно забезпечити безпеку ключа під час всіх етапів алгоритму, необхідно уникнути можливості витоку конфіденційної інформації через вразливості у керуванні ключами.

Також важливо врахувати можливості зміни параметрів алгоритму для різних вимог. Алгоритм повинен бути гнучким та легко розширюваним, дозволяючи змінювати параметри, такі як розмір блоку, довжина ключа та інші, з метою вирішення різних вимог застосувань. Важливо, щоб алгоритм був компактним та легко реалізовуваним в різних програмних та апаратних середовищах.

Для оцінки розробленого методу шифрування необхідно виконати тести на різних вхідних даних та ключах. Провести аналіз безпеки алгоритму, спробувати виявити та виправити можливі вразливості. Перевірити стійкість до кріптоаналітичиних атак. За результатами тестування провести оцінювання ефективності методу з точки зору продуктивності та витрат. Також оцінити розроблений метод за міжнародними стандартами та вимогами до сучасних шифрів.

2 РОЗРОБКА МЕТОДУ ШИФРУВАННЯ

2.1 Узагальнена схема методу шифрування

Для реалізації методу шифрування перш за все необхідно обробити вхідві дані. Маючи 64 бітний ключ та деякі дані для шифрування алгоритм розпочинається з того, що ключ та дані розбивається на байти, відповідно ключ це 8 байт, дані n байт. Після цього, метод переходить до взначення характеристичної ознаки, а саме беремо код байту за модулем 8, саме цей результат буде показником, яку саме операцію для певного набару байтів виконуватиме методу зашифрування.

Маючи всю необхідну інформації, можна описати алгоритм роботи методу шифрування забражену на рис. 2.1.

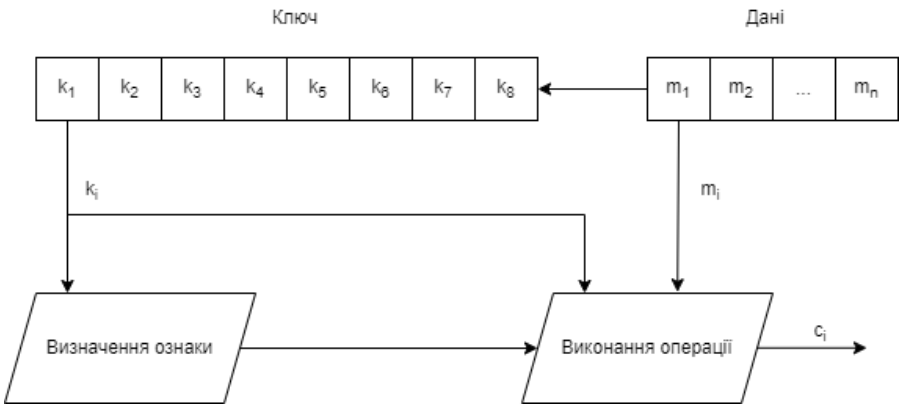


Рисунок 2.1 – Схема процесу зашифрування

Після завершення ітерації алгоритм переходить далі беручи наступний байт ключа, коли ключ буде вичерпано, на місце ключа будуть ставати байти даних що шифруються аж до поки вся інформація не бути зашифрованою.

Для розшифрування необхідно зробити такіж самі кроки, але використовуючи обернені операції, до тих які були використані під час зашифрування, відповідно на вході метод розшийрування ключ та зашифровані дані, як зображено на рис2.2.

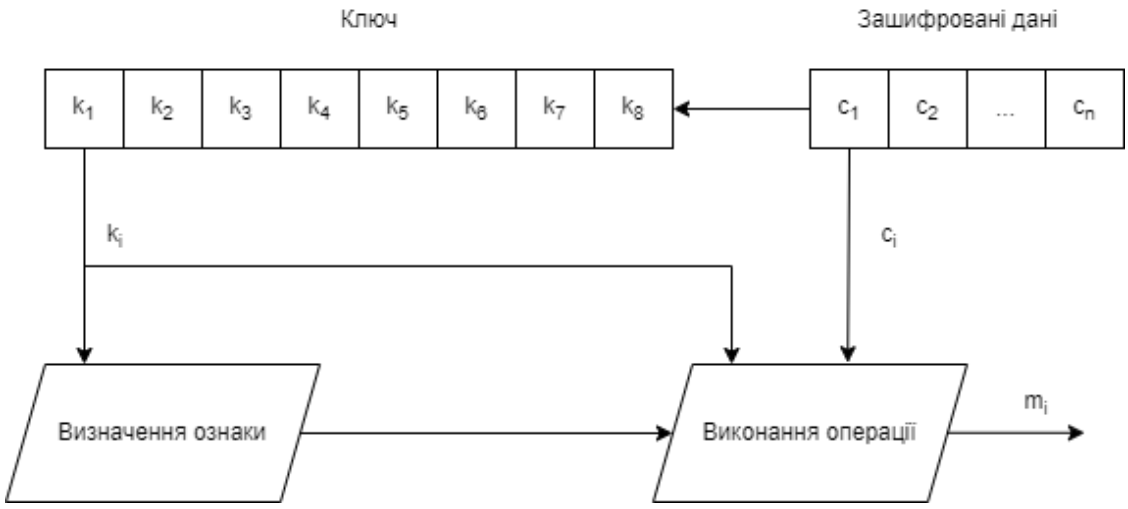


Рисунок 2.2 – Схема процесу розшифрування

Для зручності використання розробленого шифру він повнний поставлятися як стороння бібліотека. Тому алгоритм буде запакований у спеціальний NuGet пакет для подальшої інтергації в код або системи.

Підсумовуючи розділ можна зауважити що розроблений алгорит має високу шфидкість виконання оскільки він не використовує жодних сторонніх бібліотке і оперує з байтами даних. Усі операції з ключами відбуваються в середині розробленого методу, що зменшує ризики на втрату конфеденційності ключа.

2.2 Визначення характеристичної ознаки

Для реалзації методу шифрування можна використовувати будь-яку характеристику даних. У даній роботі було обрано код байту даних за модулем 8. Це означає що для вилучення характеристичної ознаки ключа потрібно взяти код окремо кожного байту за модулем 8 і відповідно до цього формувати наступні дію алгоритму шифрування.

Обчислення числового значення байту за модулем 8 реалізується шляхом логічного множення коду байту на код числа 7. Наприклад, числове значення байту 150, а його двійковий код 10010110. Результатом логічного множення на

код числа 7 є код: 10010110 & 00000111 = 00000110. Який відповідає числу 6. Дійсно, 150 mod 8 = 6.

Обрана характеристика є унікальною для кожного набору даних та забезпечує високу продуктивність оскільки не вимагає багато ресурсів для того, щоб витягти її з даних.

2.3 Байтові операції

Байтові операції - це операції, які виконуються над окремими байтами даних. Байтові операції використовуються в блоковому шифрі для реалізації різних кроків зашифрування та розшифрування. Для реалізації методу шифрування потрібно обрати та описати, 8 різних операцій для процесу зашифрування і обернених до них операцій для процесу розшифрування. Кількість операцій визначається тим, що маємо 8 різних значень за модулем 8.

Операція “інверсія” (також відома як бітовий NOT) - це бітова операція, яка перевертає значення кожного біту в заданому коді. Якщо біт має значення 0, то після інверсії він стає 1, і навпаки. Наприклад, якщо ми маємо 8-бітове код 00101100, то його побітова інверсія буде 11010011.

Зсув - це операція, яка зміщує біти в числі або бітовому векторі вліво або вправо на певну кількість позицій. Зсуви можуть бути як на одну так і на більше позицій. Є два типи зсувів:

Операція “зсув вліво” (<<). Всі біти в коді байту зсуваються вліво на вказану кількість позицій. Наприклад, якщо код 00001010 (десяткове число 10), то результатом зсуву вліво на одну позицію є код буде 00010100 (десяткове число 20).

Операція “зсуву вправо” (>>): Всі 8 бітів числа зсуваються вправо на вказану кількість позицій. Наприклад, якщо код B - 10101010 (десяткове 170), то вираз B >> 1 зсуне всі 8 бітів вправо на одну позицію, отримаємо 01010101 (десяткове 85).

Операція “XOR” (виключне АБО) - це побітова операція, що виконує

побітове додавання по модулю 2. Вона повертає 1 для кожного біту, де кількість введених бітів зі значенням 1 є непарною, та 0 для бітів з парною кількістю одиниць. Наприклад, якщо кододного байту 10101010, а код другого 11001100, використання побітового XOR між ними (10101010 XOR 11001100) дає результат 01100110.

Побітовий “AND” (&) - бінарна операція , дія якої еквівалентна застосуванню логічного «І» до кожної пари бітів, які стоять на однакових позиціях у двійкових уявленнях операндів. Іншими словами, якщо обидва відповідні біти операндів рівні 1, результуючий двійковий розряд дорівнює 1; якщо хоча б один біт з пари дорівнює 0, результуючий двійковий розряд дорівнює 0. Наприклад, для двох восьмибітових чисел: 10101010 та 11001100, використання побітового AND (10101010 & 11001100) дає результат 10001000.

Побітовий “OR" (|) - бінарна операція, дія якої еквівалентна застосуванню логічного «АБО» до кожної пари бітів, які стоять на однакових позиціях у двійкових уявленнях операндів. Іншими словами, якщо обидва відповідні біти операндів дорівнюють 0, двійковий розряд результату дорівнює 0; якщо хоча б один біт з пари дорівнює 1, двійковий розряд результату дорівнює 1. Наприклад, для двох восьмибітових чисел: 10101010 та 11001100, використання побітового OR (10101010 | 11001100) дає результат 11101110.

Таблиця 2.1 –Байтові операції для зашифрування

A mod 8	Формула операції
0	k_i XOR m_i
1	(k_i + m_i) mod 256
2	k_i XOR (NOT m_i)
3	(NOT k_i) XOR m_i
4	((NOT k_i) + (NOT m_i)) mod 256
5	(k_i>> 2) XOR (m_i << 2)
6	(NOT k_i + m_i) mod 256
7	NOT k_i XOR NOT m_i

У таблицях наведено операції, що в залежності від результату розрахунок по модулю 8 будуть використовуватись в алгоритмі розробленого шифрування, де k_i - це байт ключа шифрування відповідної ітерації; m_i - байт даних, що шифруються відповідної ітерації. Отже, в залежності результату обчислення байту даних по модулю буде виконуватись певна логічна побітова операція.

Отже, було обрано операції для реалізації шифрування, як забражено у таблиці 2.1, та розшифрування у табоиці 2.2, деякі логічні операції було комбіновано для покращення безпекових характеристик шифру.

Варто відмітити, що при реалізації алгоритму можна залишити місце для конфігурацій окремих операцій, оскільки потрібно врахувати той факт, що операції можуть об’єднуватися у більш складні.

Таблиця 2.2 –Байтові операції для розшифрування

A mod 8	Формула операції
0	k_i XOR c_i
1	$(c_i - k_i) \bmod 256$
2	k_i XOR (NOT c_i)
3	(NOT k_i) XOR c_i
4	((NOT c_i) - (NOT k_i)) mod 256
5	(k_i << 2) XOR (c_i>> 2)
6	(c_i - NOT k_i) mod 256
7	(NOT k_i) XOR (NOT c_i)

2.4 Приклад реалізації методу зашифрування

Нехай ключ має вигляд: $k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8$,

де $k_1 = 16, k_2 = 17, k_3 = 18, k_4 = 19, k_5 = 20, k_6 = 21, k_7 = 22, k_8 = 23$.

Повідомлення має вигляд: $m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8$,

де $m_1 = 103$ $m_2 = 74$ $m_3 = 43$ $m_4 = 12$, $m_5 = 212$, $m_6 = 124$, $m_7 = 79$, $m_8 = 187$,

Крок 1. Обчислюємо ознаку $k_1 : 16 \bmod 8 = 0$.

Виконуємо операцію за номером 0 : $c_1 = k_1$ XOR m_1, в результаті маємо c_1 = 16 XOR 103 = 00010000 XOR 01100111 = 01110111 = 119

Крок 2. Обчислюємо ознаку k_2 : 17 mod 8 = 1.

Виконуємо операцію за номером 1 : $c_2 = (k_2 + m_2)$ mod 256, в результаті маємо c_2 = (17 AND 74) mod 256 = (00010001 AND 01001010) mod 256 = (00000000) mod 256 = 0

Крок 3. Обчислюємо ознаку k_3 : 18 mod 8 = 2.

Виконуємо операцію за номером 2 : $c_3 = k_3$ XOR (NOT m_3), в результаті маємо NOT 43 = NOT 00101011 = 11010100

c_3 = 18 XOR 11010100 = 00010010 XOR 11010100 = 11000110 = 198.

Крок 4. Обчислюємо ознаку k_4 : 19 mod 8 = 3.

Виконуємо операцію за номером 3 : $c_4 =$ (NOT k_4) XOR c_4, в результаті маємо NOT 19 = NOT 00010011 = 11101100

c_4 = 11101100 XOR 00001100 = 11101100 XOR 00001100 = 11100000 = 224

Крок 5. Обчислюємо ознаку k_5 : 20 mod 8 = 4.

Виконуємо операцію за номером 4 : $c_5 =$ ((NOT k_5) + (NOT m_5)) mod 256, в результаті маємо (NOT 20) AND (NOT 212)

c_5 = 11101100 AND 11010100 = 00100100 = 39

Крок 6. Обчислюємо ознаку k_6 : 21 mod 8 = 5.

Виконуємо операцію за номером 5 : $c_6 = (k_6 >> 2)$ XOR ($m_6 << 2$), в результаті маємо(21<<2) XOR (124>>2)

c_6 = (21<<2) XOR (124>>2) = (10101 XOR 1111) = 01101000 = 01101000 = 111

Крок 7. Обчислюємо ознаку k_7 : 22 mod 8 = 6.

Виконуємо операцію за номером 6 : $c_7 =$ (NOT $k_7 + m_7$) mod 256, в результаті маємо NOT 22 = 11101001 c_7 = 11101001+1001111 = 100001000 = 56

Крок 8. Обчислюємо ознаку k_8 : 23 mod 8 = 7.

Виконуємо операцію за номером 7 : c_8 = NOT k_8 XOR NOT m_8, в результаті маємо NOT 23 = 11101000, NOT 187 = 01001010

c_8 = 11101000 XOR 01001010 = 10100010 = 187

В рузультаті усіх кроків зашифрованиі дані матимуть вигляд c_1 =119 , c_2 = 0, c_3 = 198 , c_4 = 224, c_5 = 39 , c_6 = 111 , c_7 = 56 , c_8 = 187

Для розшифрування необхідно зробити такіж самі кроки, але використовуючи операції із таблиці байтових операції для розшифрування.

Крок 1. Обчислюємо ознаку k_1 : 16 mod 8 = 0.

Виконуємо операцію за номером 0 : $m_1 = k_1$ XOR c_1, в результаті маємо m_1 = 103

Крок 2. Обчислюємо ознаку k_2 : 17 mod 8 = 1.

Виконуємо операцію за номером 1 : $m_2 = (c_2 – k_2)$ mod 256, в результаті маємо m_2 = 74

Крок 3. Обчислюємо ознаку k_3 : 18 mod 8 = 2.

Виконуємо операцію за номером 2 : $m_3 = k_3$ XOR (NOT c_3) , в результаті маємо m_3 = 43

Крок 4. Обчислюємо ознаку k_4 : 19 mod 8 = 3.

Виконуємо операцію за номером 3 : m_4 = (NOT k_4) XOR c_4, в результаті маємо m_4 = 12

Крок 5. Обчислюємо ознаку k_5 : 20 mod 8 = 4.

Виконуємо операцію за номером 4 : m_5 = ((NOT c_5) - (NOT k_5)) mod 256, в результаті маємо m_5 = 212

Крок 6. Обчислюємо ознаку k_6 : 21 mod 8 = 5.

Виконуємо операцію за номером 5 : $m_6 = (k_6 << 2)$ XOR $(c_6 >> 2)$, в результаті маємо m_6 = 124

Крок 7. Обчислюємо ознаку k_7 : 22 mod 8 = 6.

Виконуємо операцію за номером 6 : $m_7 = (c_7$ - NOT $k_7)$ mod 256, в результаті маємо m_7 = 79

Крок 8. Обчислюємо ознаку k_8 : 23 mod 8 = 7.

Виконуємо операцію за номером 7 : m_8 = (NOT k_8) XOR (NOT c_8), в результаті маємо m_8 = 187

Отже m_1 = 103 m_2 = 74 m_3 = 43 m_4 = 12, m_5 = 212, m_6 = 124, m_7 = 79, m_8 = 187.

3 РЕАЛІЗАЦЯ ПРОГРАМНОГО ЗАСОБУ

3.1 Реалізація методу шифрування

Для досягнення мети створення ефективного і безпечного шифру, були вибрані наступні технології та інструменти. Мова програмування C# була обрана з урахуванням її широкого застосування у розробці програм для платформи .NET. Ця мова надає потужність та можливості для реалізації складних шифрів, використовуючи об'єктно-орієнтований підхід та багатофункціональний API. Використання платформи .NET дозволило ефективно інтегрувати розроблений шифр у великий спектр додатків та систем, що побудовані на цій платформі. Зокрема, використання нового .NET, з його модульністю та підтримкою крос-платформеності, забезпечило гнучкість у використанні шифру на різних пристроях [24].

Для початку реалізуємо клас Common, який містить методи для шифрування та розшифрування даних. У конструкторі ми ініціалізуємо ключ та текстові дані для подальшого шифрування [25]. Важливою частиною цього коду є метод GenerateBitKey, який генерує випадковий ключ заданої довжини.

```
public readonly byte[] key;
public readonly byte[] plainTextBytes;
public readonly BytesOperations _Operations;

public Common(int keyLength, string data)
{
    key = Generate64BitKey(keyLength);
    plainTextBytes = Encoding.UTF8.GetBytes(data);
    _Operations = new BytesOperations();
}
```

Методи Encrypt та Decrypt відповідають за процеси зашифрування та розшифрування відповідно. Вони використовують ключ для застосування операцій над кожним байтом тексту. Ці операції включають зсуви бітів, логічні

операції (XOR, AND, OR, NOT) та інші, які впливають на байти тексту.

Також використовуємо допоміжні методи, які додають байти до масивів байтів та рахують кількість встановлених бітів у байті, розрахунок модуля.

У роботі ми виводимо інформацію про кількість встановлених бітів у ключі під час шифрування та розшифрування, щоб демонструвати вплив цього параметру на процес трансформації даних[26].

```
public byte[] AddByteToArray(byte[] originalArray, byte byteToAdd)
{
    byte[] newArray = new byte[originalArray.Length + 1];

    Array.Copy(originalArray, newArray, originalArray.Length);

    newArray[newArray.Length - 1] = byteToAdd;

    return newArray;
}

public int CountSetBits(byte value)
{
    int count = 0;

    for (int i = 0; i < 8; i++)
    {
        if ((value & (1 << i)) != 0)
        {
            count++;
        }
    }

    return count;
}
public static int CalculateModulo(byte value, int modulo)
{
    // Обчислення остачі від ділення байту на вказаний модуль
    int result = value % modulo;
    return result;
}
```

Для шифрування та розшифрування використовуєтсья відповідні одноіменні методи. Їх унікальність полягає у тому що вони використовуються

відповідний метод роботи з байтами за умови певного результату обчислення модуля, як наведено нижче [27].

```
                switch (count)
                {
                    case 0:
                        encryptedTextByte = _Operations.shiftedLeft(2, b2);
                        result = AddByteToArray(result, encryptedTextByte);
                        break;
                    case 1:
                        encryptedTextByte = _Operations.shiftedLeft(1, b2);
                        result = AddByteToArray(result, encryptedTextByte);
                        break;
                    case 2:
                        encryptedTextByte = _Operations.XOR(b1, b2);
                        result = AddByteToArray(result, encryptedTextByte);
                        break;
                    case 3:
                        encryptedTextByte = _Operations.modulus(256, _Operations.AND(b1, b2));
                        result = AddByteToArray(result, encryptedTextByte);
                        break;
                    case 4:
                        encryptedTextByte = _Operations.XOR(b1, _Operations.NOT(b2));
                        result = AddByteToArray(result, encryptedTextByte);
                        break;
                    case 5:
                        encryptedTextByte = _Operations.XOR(_Operations.NOT(b1), b2);
                        result = AddByteToArray(result, encryptedTextByte);
                        break;
                    case 6:
                        encryptedTextByte = _Operations.modulus(256,
_Operations.AND(_Operations.NOT(b1), _Operations.NOT(b2)));
                        result = AddByteToArray(result, encryptedTextByte);
                        break;
                    case 7:
                        encryptedTextByte = _Operations.shiftedRight(2, b2);
                        result = AddByteToArray(result, encryptedTextByte);
                        break;
                    case 8:
                        encryptedTextByte = _Operations.shiftedRight(1, b2);
                        AddByteToArray(result, encryptedTextByte);
                        break;
                    default:
                        break;
                   }
```

На цьому етапі необхідно реалізувати додатковий клас де буде описано побітові операції. Клас BytesOperations є інструментом для виконання операцій над байтами, які використовуються під час шифрування та розшифрування даних.

Початково цей клас створений для спрощення реалізації різних логічних та арифметичних операцій над байтами. Методи в цьому класі виконують такі операції, як виключне або (XOR), зсуви бітів вправо та вліво, обернення бітів (NOT), а також операції AND, OR та модуль [28].

Кожен метод виконує конкретну операцію над одним чи двома байтами і повертає результат цієї операції. Цей підхід дозволяє використовувати цей клас для виконання різних операцій під час шифрування та розшифрування даних у моїй магістерській роботі. Клас BytesOperations є свого роду "бібліотекою" функцій для маніпулювання байтами, що спрощує та структурує роботу з даними на бітовому рівні.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace NHCypher
{
    public class BytesOperations
    {
        public byte XOR(byte byte1, byte byte2)
        {
            return (byte)(byte1 ^ byte2);
        }
        public byte shiftedRight(int number, byte byte1)
        {
            return (byte)(byte1 >> number);
        }
        public byte shiftedLeft(int number, byte byte1)
        {
            return (byte)(byte1 << number);
        }
        public byte NOT(byte byte1)
        {
```

```
            return (byte)(~byte1);
        }
        public byte AND(byte byte1, byte byte2)
        {
            return (byte)(byte1 & byte2);
        }
        public byte OR(byte byte1, byte byte2)
        {
            return (byte)(byte1 | byte2);
        }
        public byte modulus(int number, byte byte1)
        {
            return (byte)(byte1 % number);
        }
    }
}
```

У взаємодії ці два класи представляються реалізацію методу шифрування даних на основі характеристичних ознак. Він є легко розширювальним, що дає змогу без додаткових затрат змінювати кахактеристичну ознаку по якій виконувати бітові операції, самі бітові операці, розмір ключа, тощо.

3.2 Інтеграція та тестування

Для перевірки функціональності та коректності роботи шифру були використані фреймворки для тестування. NUnit та MSTest дозволили створювати автоматизовані тести, які перевіряли різні аспекти роботи шифру, включаючи правильність шифрування та розшифрування, взаємодію з різними форматами даних та інші аспекти [29].

Для цього необхдно створити новий тестовий проект та підключити до нього описані вищі фреймворки. Загальний тест для методу шифрування це той тест, який на вхід отримає дані що маються бути зашифровані, шифрує їх, потім розшифровує і перевіряє із початковими. Це тест вигладає нуступним чином.

```
    public class Tests
    {
        private BytesOperations _operations;
        private Common _common;
        private readonly int keyLength = 8;
```

```
private readonly string plaintext = "Test for NHChypher! Vinnytsia 2023***";

[SetUp]
public void Setup()
{
    _common = new Common(keyLength, plaintext);
    _operations = new BytesOperations();
}

[Test]
public void TestEncryptionDecryption()
{

    byte[] encrypted = _common.Encrypt(_common.key, _common.plainTextBytes);

    string decryptedText = _common.Decrypt(_common.key, encrypted);

    Assert.AreEqual(plaintext, decryptedText);
}
```

Для првильного та повного тестування за різними метадологіями необхідно поривати майже увесь код тестами. А це означає, що кожна операція, кожен цикл має бути протестований

Розуміючи важливість тестування шифру для переконання, що він працює правильно у всіх сценаріях. Щоб покрити свій шифр тестами, використано юніт-тести для перевірки кожної окремої частини свого коду на коректність (Рис.3.1). Для початку, створено набір тестів для класу BytesOperations. У цих тестах перевірено кожен метод цього класу з різними вхідними значеннями, щоб переконатися, що вони повертають очікувані результати.

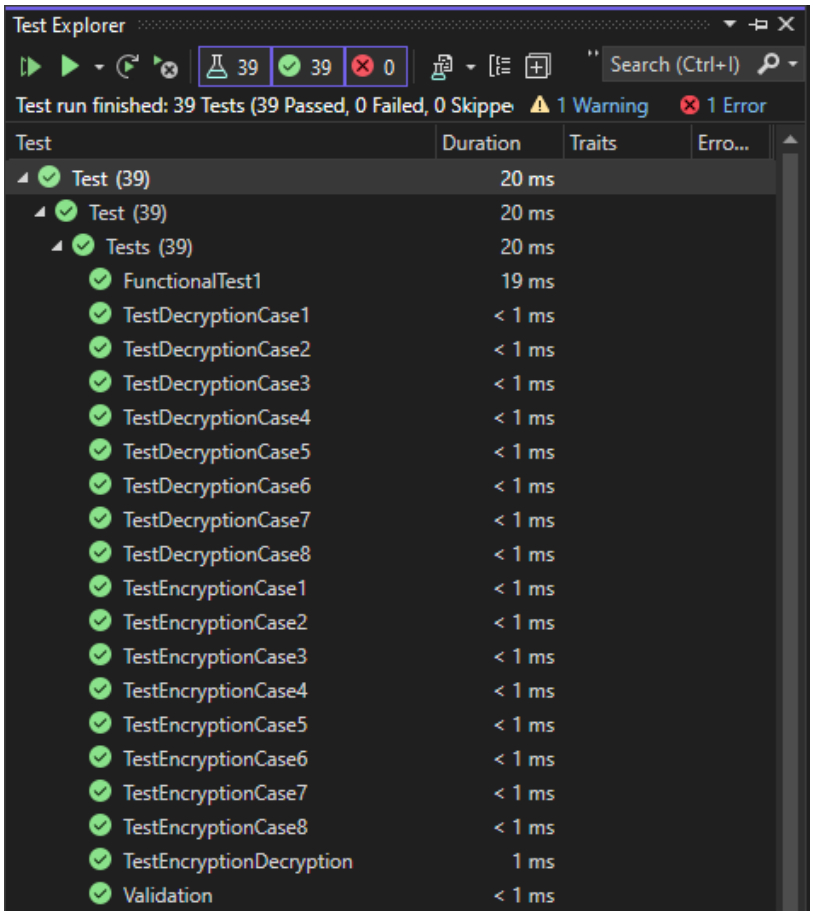


Рисунок 3.1 – Тестування розробленого методу

Наступним кроком було створення тестів для класу Common, що відповідає за шифрування та розшифрування даних. Реалізовано тести для методів Encrypt та Decrypt, подавши різні вхідні дані та ключі, щоб переконатися, що вони повертають очікувані результати та розшифровують дані правильно.

Окрім цього, я б також покрито код тестами на випадки роботи з помилками. Оскільки методи передбачають обробку помилок, наприклад, неправильні вхідні дані або невірний формат ключа, тому створено тести, які перевіряють, чи вони викликають очікувані винятки та обробляють помилки належним чином.

Після реалізації шифру було вирішено здійснити його розповсюдження через пакувальну систему NuGet. Це рішення спростило процес установки та використання шифру для розробників, дозволяючи легко і швидко додавати його до своїх проектів.

(NuGet packages) - це засіб для розповсюдження бібліотек, фреймворків та

різноманітних компонентів програмного забезпечення в середовищі .NET. Це своєрідний контейнер, який містить в собі скомпільований код, ресурси, метадані та інші файли, які можна легко встановити в проект, що працює на платформі .NET [30].

Ось кілька плюсів використання NuGet:

Легкість використання: NuGet дозволяє легко розповсюджувати свій код. Інші розробники можуть легко встановити ваш пакет у свої проекти за допомогою Visual Studio або командного рядка.

Управління залежностями: NuGet дозволяє вам вказувати залежності вашого пакету від інших пакетів, що полегшує управління залежностями вашого проекту.

Автоматичне оновлення: Розробники, які встановлюють ваш пакет, можуть отримувати автоматичні оновлення, якщо ви випускаєте нові версії пакету.

Версіонування: NuGet дозволяє працювати з версіями вашого пакету, що полегшує управління версіями для користувачів.

Щоб запакувати шифр у NuGet-пакет, необхідно створити файл NuGet-специфікації (файл .nuspec) для проекту [31]. В цьому файлі вказуються метадані пакету, такі як назва, версія, залежності тощо.

Після цього необхідно скомпілювати проект проект та згенерував NuGet-пакет за допомогою інструментів відповідної або командного рядка за допомогою nuget pack зображено на рис.3.2.

Структура нюгет-пакету включає в себе кілька ключових елементів, які дозволяють належним чином організувати та управляти компонентами програмного забезпечення:

lib: Це каталог, де зберігаються бінарні файли бібліотеки, зазвичай різні версії для різних фреймворків (наприклад, net45, netstandard2.0 тощо).

content: Тут можуть розміщуватися ресурси для розгортання в проект, такі як шаблони, конфігураційні файли, зображення тощо.

build: Містить скрипти MSBuild, які виконуються під час встановлення або видалення пакету. Вони можуть виконувати додаткові дії, наприклад, копіювання файлів, зміни конфігурацій тощо.

tools: Тут розміщуються інструменти, необхідні для роботи з пакетом, наприклад, утиліти командного рядка або інші допоміжні програми.

package: Містить метадані пакету, такі як файли .nuspec, які містять опис пакету, залежності, версію тощо.

docs: Каталог для документації пакету, яка може бути відображена у NuGet Package Manager або інших інструментах.

tests: Містить тести для пакету, що допомагають перевірити його працездатність.

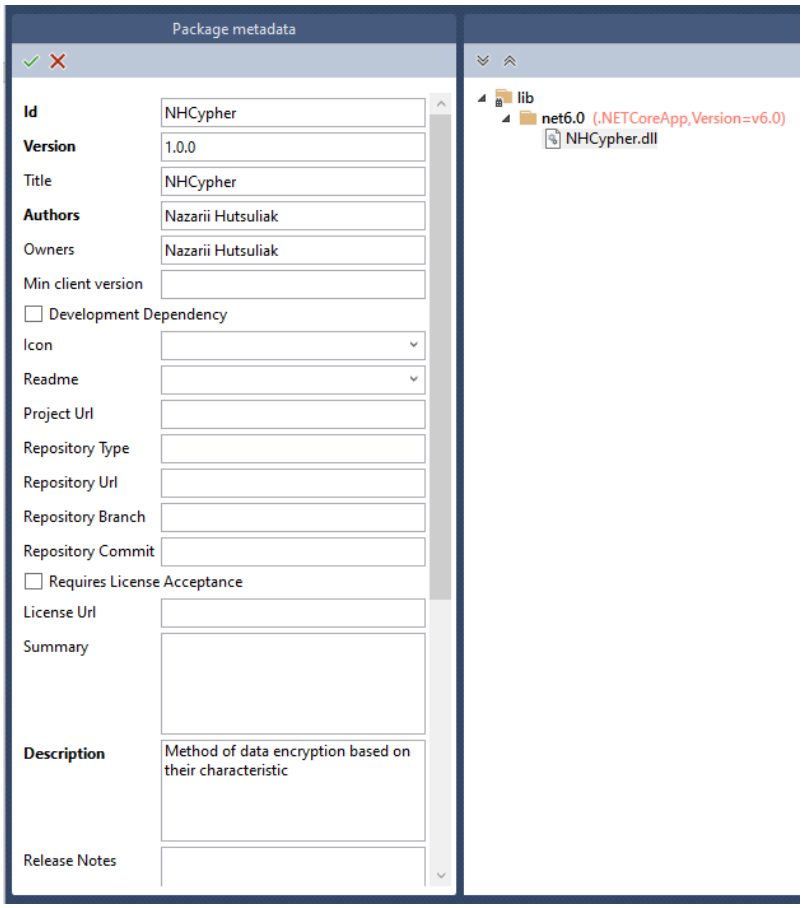


Рисунок 3.2 – Сформований пакет із реалізованим методом

Останній крок це публікування цього пакету на сервері NuGet, щоб інші розробники могли легко знайти та встановити його у свої проекти.

3.3 Порівняльні оцінки методів шифрування

Розроблений застосунок відповідає усім критеріям поставленого завдання. У своєму роді він є унікальним та може скласти конкуренцію існуючим методам шифрування. Для порівняння розробленого засобу було обрано такі критерії:

- Безпека: Найважливіший аспект - це стійкість до злому. Методи, що базуються на сучасних алгоритмах шифрування, таких як AES (Advanced Encryption Standard) чи RSA (Rivest-Shamir-Adleman), зазвичай вважаються стійкими.
- Швидкість: Деякі методи шифрування можуть бути більш обчислювально витратними, що може впливати на швидкодію обробки даних.
- Гнучкість: Деякі методи можуть бути більш гнучкими у використанні та можуть підтримувати різні типи даних чи дозволяти різні режими шифрування.
- Ресурсомісткість: Деякі методи можуть вимагати більше ресурсів (пам'яті, обчислювальної потужності) для шифрування та розшифрування.
- Простота використання та інтеграції: Як метод інтегрується з існуючими системами, наскільки просто його використовувати.

Порівняння з існуючим засобами для моніторингу, а саме AES, RSA висвітлено у наведеній нижче таблиці. Аналоги широко використовуються на різних підприємствах та є задіяними у формуванні політик безпеки. AES Є стандартом серед методів шифрування. DES є застралім методом, хоча раніше він і був стандартом для використання, але наведений для кращого розуміння загальної картини.

Безпека оцінюється на високому рівні у всіх методів, але, наприклад, AES відомий своєю високою швидкодією, але не так гнучким, як мій метод, що має високий показник гнучкості. RSA, хоча і має високий рівень безпеки, але має низьку швидкість та гнучкість у порівнянні з іншими методами.

Таблиця 3.1 – Порівняння розроблений засобу з аналогами

Продукт	Розроблений у ході виконання роботи	DES	AES	RSA
Безпека	Середня	Низька	Висока	Висока
Швидкість	Середня	Середня	Висока	Низька
Гнучкість	Висока	Низька	Середня	Низька
Ресурсомісткість	Висока	Середня	Середня	Висока
Простота використання та інтеграції	Висока	Висока	Висока	Низька

Виходячи з результатів порівняння очевидно, що створений засіб має перевагу у деяких показниках. Швидкість реалзованого методу на рівні із найкращими, але все ж таки цього не достатньо. Невелика затрата ресурсів, також дуже суттєвий показник. Простота ітеграції виявилась найвищою, що добревідібється на популярності, а ось над безпекою потрібно попрацювати. Необхідно зауважити, що зміна характеристичних ознак та бітових операцій зможе покращити показники найдійності.

Хоча і AES відзначається високою швидкодією, тоді як RSA має свої особливості в застосуваннях, де важлива асиметрична криптографія, за результатами аналізу можна визначити, що для певних сценаріїв застосування існуючі розроблений методи шифрування може бути відмінним вибором.

Враховуючи унікальні потреби та контекст застосування, вибір методу шифрування може бути здійснений на основі конкретних вимог до безпеки, ефективності та інтеграції. Важливо обирати метод шифрування, який найкращим чином відповідає потребам конкретного проекту чи системи.

3.4 Лавинний тест

"Лавинний ефект" або "лавинний тест" у контексті криптографії

використовуватися для методів шифрування. У криптографії, лавинний ефект — це явище, коли навіть найменша зміна в вхідних даних при шифруванні призводить до значних змін у вихідному шифротексті. Це важлива властивість сучасних криптографічних алгоритмів.

Криптографічні алгоритми, які мають лавинний ефект, вважаються стійкими та надійними, оскільки навіть дрібні зміни у вхідних даних призводять до значних змін у вихідному шифротексті. Це робить важким передбачення або аналізування шаблонів у шифротексті, що зроблює злам шифру важким завданням.

Лавинний ефект є важливою характеристикою безпеки багатьох сучасних криптографічних алгоритмів, таких як AES (Advanced Encryption Standard), і він сприяє забезпеченню високого рівня стійкості шифрування даних.

Для цієї перевірки було реалізовано низку тестів, під час виконання яких, було навмисно змінено один байт, що перевірити чи за лавинним ефектом усі наступні байти будуть у нерозбірливому вигляді. Код тесту наведено нижче.

```
[TestMethod]
public void AvalancheTest(byte[] originalData)
{
    // Закодуємо оригінальні дані
    byte[] encryptedData = Encrypt(originalData);

    // Зміна одного байту у середині закодованих даних
    encryptedData[encryptedData.Length / 2] ^= 0x01;

    // Розкодуємо змінені дані
    byte[] decryptedData = Decrypt(encryptedData);

    // Порівняння розкодованих даних з оригіналом
    CollectionAssert.AreNotEqual(originalData, decryptedData);
}
```

За результатами тесту видно, що заміна одного байту під час розшифрування призводить до лавинного ефекту так, що наступні ітерації реалізованого методу призводить до змін шифротексту, що унемоливлює його подальше розшифрування (Рис. 3.3).

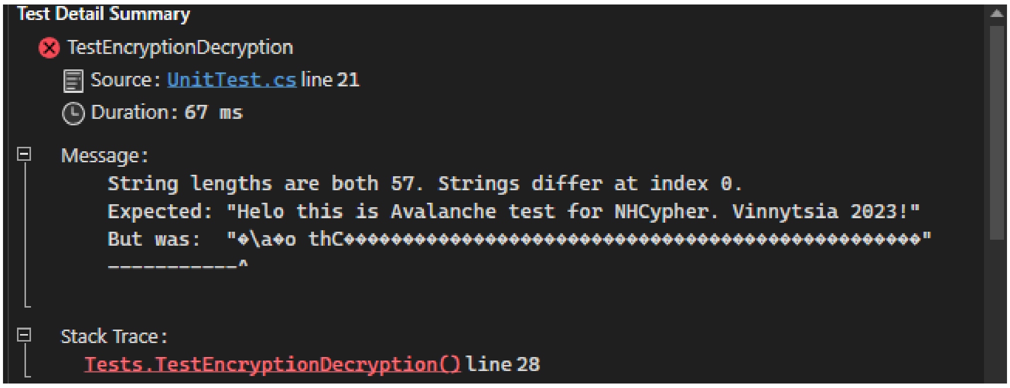


Рисунок 3.3 – Перевірка методу на наявність лавинного ефекту

Подальше тестування виявило, що попередній тест був найвдаліший, оскільки замінений байт був перший, але в середнього примусова зміна байту під час шифрування призводить до зміни усі подальший байтів, як зображено на рис. 3.4

.

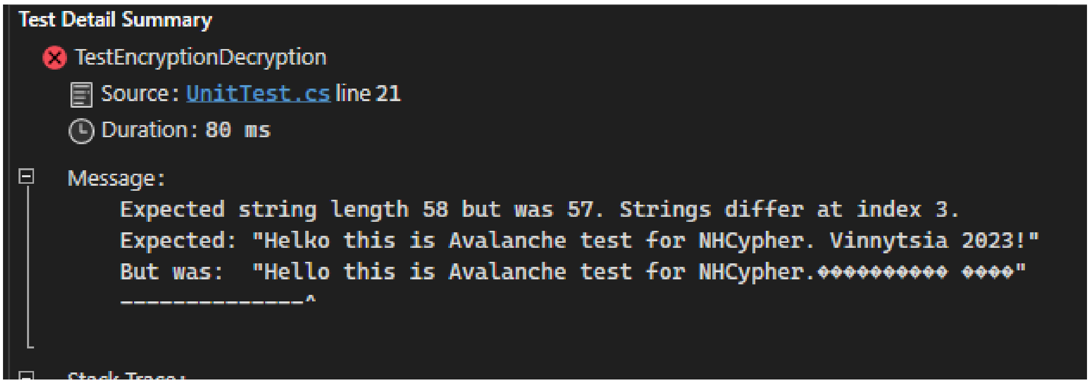


Рисунок 3.4 – Результат лавинного тесту реалізованого методу шифрування.

У підсумку виявилось, що в залежності від того де буде зміна байту розроблений методу шифрування покаже або надзвичайно високий показник, або низький, за статистикою зібраною під час проведення тестування випадковим метод, реазіовний метод показав відсоток лавинного ефекту 35 %.

Наявність лавинного ефекту свідчить про важливу характеристику криптографічного алгоритму, яка забезпечує стійкість та безпеку у разі невеликих змін в вхідних даних.

4 ЕКОНОМІЧНА ЧАСТИНА

Для успішного впровадження науково-технічної розробки важливо, щоб вона відповідала сучасним вимогам науково-технічного прогресу та враховувала економічні аспекти. Оцінка економічної ефективності результатів науково-дослідної роботи є важливою частиною цього процесу. Дослідження, представлене у магістерській роботі та присвячене розробці та вивченню "Методу шифрування даних на основі їх характеристичних ознак", віднесено до науково-технічних робіт, спрямованих на виведення на ринок. Рішення щодо комерціалізації розробки може бути прийняте протягом виконання самої роботи, відкриваючи можливості для подальшого виведення на ринок. Цей напрямок визначається як пріоритетний, оскільки розроблені результати можуть бути корисними для різних зацікавлених сторін і приносити економічні вигоди. Проте для успішної реалізації цього процесу важливо залучити зацікавленого інвестора, який виявить інтерес до втілення даного проекту, і переконати його у доцільності інвестування у цю розробку. З метою досягнення цього завдання передбачені наступні етапи виконання робіт:

1. Проведення комерційного аудиту науково-технічної розробки, включаючи визначення науково-технічного рівня та комерційного потенціалу.

2. Розрахунок витрат на реалізацію науково-технічної розробки.

3. Проведення розрахунку економічної ефективності впровадження та комерціалізації науково-технічної розробки для потенційного інвестора, а також обґрунтування економічної доцільності комерціалізації з точки зору інвестора.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Метод шифрування даних на основі їх характеристичних ознак» є розширення функціональних можливостей програмного забезпечення для захисту файлів.

Оцінювання науково-технічного рівня розробки та її комерційного

потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно	Технічні та споживчі влас-тивості продукту трохи гірші, ніж	Технічні та споживчі властивості продукту на рівні	Технічні та споживчі властивості продукту трохи	Технічні та споживчі властивості продукту значно
5	Експлуатаційні витрати значно вищі, ніж в	Експлуатаційні витрати дещо вищі, ніж в	Експлуатаційні витрати на рівні експлуатаційних	Експлуатаційні витрати трохи нижчі, ніж в	Експлуатаційні витрати значно нижчі, ніж в
Ринкові перспективи					
6	Ринок малий і не має позитивної	Ринок малий, але має позитивну	Середній ринок з позитивною	Великий стабільний ринок	Великий ринок з позитивною
7	Активна конкуренція великих	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї	Потрібні незначні фінансові ресурси. Джерела фінансування	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовують ся у військово	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці. Для оцінки науково-технічного рівня і комерційного потенціалу розробки експертами було запрошено трьох незалежних експертів: Захар Окрутний. Middle+ FullStack Developer. ТОВ «Delphi Software», Борис Чорновол. Middle Backend Developer. ТОВ «Delphi Software», Микола Седлецький Solution Architect ТОВ «Delphi Software».

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	Захар	Борис Чорновол.	Микола

	Окрутний.	Middle Backend Developer. ТОВ «Delphi Software»	Седлецький Solution Architect ТОВ «Delphi Software»
	Бали, виставлені експертами:		
1. Технічна здійсненність концепції	4	4	5
2. Ринкові переваги (наявність аналогів)	2	3	3
3. Ринкові переваги (ціна продукту)	4	4	3
4. Ринкові переваги (технічні властивості)	3	3	4
5. Ринкові переваги (експлуатаційні витрати)	2	2	3
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	3	3	3
8. Практична здійсненність (наявність фахівців)	5	5	5
9. Практична здійсненність (наявність фінансів)	2	3	2
10. Практична здійсненність (необхідність нових матеріалів)	4	5	5
11. Практична здійсненність (термін реалізації)	3	4	5
12. Практична здійсненність (розробка документів)	4	5	4
Сума балів	$СБ_1$=39	$СБ_2$=44	$СБ_3$=45
Середньоарифметична сума балів $СБ_c$	43		

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок

щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 [32].

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Метод шифрування даних на основі їх характеристичних ознак» становить 43 бали, що, відповідно до таблиці 4.3 рівень комерційного потенціалу розробки високий, що свідчить про комерційну важливість проведення даних досліджень.

Магістерська кваліфікаційна робота «Метод шифрування даних на основі їх характеристичних ознак» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок, тобто при цьому відбувається комерціалізація науково-технічної розробки. Цей напрямок є для нас пріоритетним, оскільки результатами розробки можуть користуватися не тільки самі розробники, а й інші споживачі, отримуючи при цьому суттєвий економічний ефект.

Розробка програмного засобу для шифрування даних на основі їх характеристичних ознак може бути реалізована в різних сферах і мати різний спектр користувачів. Ось деякі можливі шляхи реалізації:

1. Корпоративні системи безпеки даних: Такий засіб може бути використаний у великих компаніях або урядових установах для шифрування конфіденційних даних, які зберігаються на серверах чи передаються через мережі.

2. Клієнтські застосунки: Це може бути частиною захисту конфіденційних даних у додатках для мобільних пристроїв або комп'ютерів, де користувачі мають можливість шифрувати та розшифровувати дані за допомогою характеристик, таких як паролі, відбитки пальців тощо.

3. Інтернет безпека: Реалізація цієї технології може включати її використання у браузерах чи розширеннях для шифрування передачі даних через Інтернет, забезпечуючи безпеку від перехоплення інформації під час передачі.

4. Системи зберігання в хмарі: Шифрування даних на основі характеристичних ознак може бути корисним для забезпечення приватності та безпеки даних, що зберігаються в хмарних сервісах.

5. Медичні системи зберігання даних: В галузі медицини, де конфіденційність даних - критичний аспект, такий засіб може бути використаний для захисту особистої медичної інформації.

Реалізація може варіюватися в залежності від цільової аудиторії і конкретних потреб в безпеці даних. Користувачі можуть бути IT-професіоналами, звичайними споживачами, організаціями або будь-якою галуззю, яка потребує захисту конфіденційної інформації.

4.2 Визначення рівня конкурентоспроможності розробки

В процесі визначення економічної ефективності науково-технічної розробки також доцільно провести прогноз рівня її конкурентоспроможності за сукупністю параметрів, що підлягають оцінюванню.

Як аналог для програмного засобу шифрування даних на основі їх характеристичних ознак можна розглянути систему шифрування на основі атрибутів (Attribute-Based Encryption - ABE).

ABE - це криптографічна схема, де розшифрування даних залежить від певних атрибутів, які має користувач. Це може бути використано для шифрування даних у великих організаціях або системах, де доступ до інформації контролюється за допомогою різних рівнів доступу та політик безпеки.

Одиничний параметричний індекс розраховуємо за формулою [32]

$$q_i = \frac{P_i}{P_{базі}}. \quad (4.1)$$

де q_i – одиничний параметричний індекс, розрахований за i-м параметром;

P_i – значення i-го параметра виробу;

$P_{базі}$ – аналогічний параметр базового виробу-аналога, з яким проводиться порівняння.

Загальні технічні та економічні характеристики розробки представлено в таблиці 4.4.

Таблиця 4.4 – Основні техніко-економічні показники аналога та розробки, що проектується

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
1	2	3	4	5
Доступність сервісу,%	98	100	1,02	15%
Швидкодія сервісу, мс	150	120	1,25	10%
Вартість обслуговування на одного користувача, грн	35	5	7	20%
Кількість виявлених врахливостей, шт	10	3	3,33	30%
Час резервного копіювання, хв	120	15	8	15%

Нормативні параметри оцінюємо показником, який отримує одне з двох значень: 1 – пристрій відповідає нормам і стандартам; 0 – не відповідає.

Груповий показник конкурентоспроможності за нормативними параметрами розраховуємо як добуток частинних показників за кожним параметром за формулою[32]:

$$I_{НП} = \prod_{i=1}^{n} q_i \ , \qquad (4.2)$$

де $I_{нп}$ – загальний показник конкурентоспроможності за нормативними параметрами;

q_i – одиничний (частинний) показник за i-м нормативним параметром;

n – кількість нормативних параметрів, які підлягають оцінюванню.

За нормативними параметрами розроблюваний пристрій відповідає вимогам ДСТУ, тому $I_{нп} = 1$.

Значення групового параметричного індексу за технічними параметрами визначаємо з урахуванням вагомості (частки) кожного параметра [32]

$$I_{ТП} = \sum_{i=1}^{n} q_i \cdot \alpha_i \ , \qquad (4.3)$$

де $I_{тп}$ – груповий параметричний індекс за технічними показниками (порівняно з виробом-аналогом);

q_i – одиничний параметричний показник i-го параметра;

α_i – вагомість i-го параметричного показника, $\sum_{i=1}^{n} \alpha_i = 1$;

n – кількість технічних параметрів, за якими оцінюється конкурентоспроможність.

Проведемо аналіз параметрів згідно даних таблиці 4.4.

$$I_{mn} = 1{,}02 \cdot 0{,}15 + 1{,}25 \cdot 0{,}1 + 7 \cdot 0{,}2 + 3{,}33 \cdot 0{,}3 + 8 \cdot 0{,}15 = 3{,}8.$$

Груповий параметричний індекс за економічними параметрами розраховуємо за формулою [32]:

$$I_{ЕП} = \sum_{i=1}^{m} q_i \cdot \beta_i \ , \qquad (4.4)$$

де $I_{ЕП}$ – груповий параметричний індекс за економічними показниками;

q_i – економічний параметр i-го виду;

β_i – частка i-го економічного параметра, $\sum_{i=1}^{m} \beta_i = 1$;

m – кількість економічних параметрів, за якими здійснюється оцінювання. Проведемо аналіз параметрів згідно даних таблиці .

$$I_{ЕП}=0{,}75 \cdot 0{,}5 + 0{,}86 \cdot 0{,}5 = 0{,}80.$$

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розрахуємо інтегральний показник конкурентоспроможності за формулою [32]:

$$K_{ІНТ} = I_{НП} \cdot \frac{I_{ТП}}{I_{ЕП}} , \qquad (4.5)$$

$$K_{ІНТ} = 1 \cdot 3{,}8 / 0{,}80 = 4{,}8.$$

Інтегральний показник конкурентоспроможності $К_{ІНТ} > 1$, отже розробка переважає відомі аналоги за своїми техніко-економічними показниками.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Метод шифрування даних на основі їх характеристичних ознак», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [32]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.6)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=21 дні.

$$З_о = 18000 \cdot 5 / 21 = 4091 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту розробки та дослідження	18000	818,2	5	4091
Інженер-програміст 1-ї категорії	22000	1000,0	38	38000
Всього				42091

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Метод шифрування даних на основі їх характеристичних ознак» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i \,, \tag{4.7}$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} \,, \tag{4.8}$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6500 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [32].

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 21 дн;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1 \cdot 1{,}65 / (21 \cdot 8) = 65{,}8$ грн.

$З_{р1} = 65{,}8 \cdot 1 = 65{,}8$ грн.

Таблиця 4.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
1.Підготовчі	16	1	65,8	1052,9
2.Монтажні	40	3	88,8	3553,4
3.Складальні	40	5	111,9	4474,6
4.Налагоджувальні	40	2	72,4	2895,4
5.Випробувальні	16	4	59,8	957,1
Всього				12933,4

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.9)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (42091+ 12933,4) · 11 / 100% = 6052,67 грн.

4.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.10)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

$Зн$ = (42091+ 12933,4+6052,67) · 22 / 100% = 13436,93 грн.

4.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні

та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Метод шифрування даних на основі їх характеристичних ознак».

Витрати на матеріали (M), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}, \qquad (4.11)$$

де H_j – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

K_j – коефіцієнт транспортних витрат, ($K_j = 1,1 \ldots 1,15$);

B_j – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

Проведені розрахунки зведемо до таблиці.

Таблиця 4.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Вартість витраченого матеріалу, грн
Папір канцелярський офісний (А4)	180	1	180
Папір для заміток (А5)	110	1	110
Органайзер офісний	183	1	183
Всього			473
З врахуванням коефіцієнта транспортування			520,3

4.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Інформаційна система підтримки підприємств малого бізнесу у сфері послуг».

Витрати на комплектуючі К, що були використані під час виконання даного етапу роботи, розраховуються за формулою:

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i \text{ грн.,} \quad (4.12)$$

де $Н_i$ – кількість комплектуючих *i*-го виду, шт.;

$Ц_i$ – ціна комплектуючих *i*-го виду, грн.;

$К_i$ – коефіцієнт транспортних витрат, $К_i = (1{,}1 \ldots 1{,}15)$;

n – кількість видів комплектуючих.

Зроблені розрахунки бажано звести до таблиці:

Таблиця 4.8 – Витрати на комлектуючі

Найменування комплектувальних	Кількість	Ціна за штуку, грн.	Сума, грн.
Intel Xeon E5-2620, 64 ГБ оперативної пам'яті, 1 ТБ SSD	2	100000	200000
PostgreSQL	1	15000	15000
Мереже обладнання	1	10000	10000
JetBrains WebStorm	3	1000	3000
SSL-сертифікат	1	2000	2000
DataStorage	4	60000	240000
Всього з врахування коефіцієнт транспортних витрат			517000,00

4.3.5 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду

обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \qquad (4.13)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (62000 \cdot 1) / (2 \cdot 12) = 2583{,}33$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.9 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
MacBook Pro 14 2021	62000	2	1	2583,33
Робоче місце розробника ПЗ	85700	20	2	714,17
Всього				4697,38

4.3.6 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($B_е$) розраховуємо за формулою:

$$B_е = \sum_{i=1}^{n} \frac{W_{уі} \cdot t_i \cdot Ц_е \cdot K_{впі}}{\eta_i}, \qquad (4.14)$$

де $W_{уі}$ – встановлена потужність обладнання на визначеному етапі розробки,

кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е$ = 7,5 грн;

$К_{вті}$ – коефіцієнт, що враховує використання потужності, $К_{вті} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$В_е = 0,25 \cdot 290,0 \cdot 7,5 \cdot 0,5 / 0,8 = 339,84$ грн.

4.3.7 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Метод шифрування даних на основі їх характеристичних ознак» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \qquad (4.15)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$В_{св} = (42091 + 12933,4) \cdot 20 / 100\% = 11004,86$ грн.

4.3.8 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми

основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.16)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$I_в$ = (42091+ 12933,4) · 50 / 100% = 27512,15 грн.

4.3.8 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.17)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$В_{нзв}$ = (42091+ 12933,4) · 100 / 100% = 55024,3 грн.

Витрати на проведення науково-дослідної роботи на тему «Метод шифрування даних на основі їх характеристичних ознак» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.18)$$

$В_{заг}$=42091+ 12933,4+6052,67+13436,93 +520,3+517000,00+4697,38+339,84 +11004,86 +27512,15 +55024,3 = 689212,87 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної)

роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (4.19)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,5.

ЗВ = 689212,87 / 0,5 = 1378425,73 грн.

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Метод шифрування даних на основі їх характеристичних ознак» передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN – збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

N – кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 1 особа

$Ц_б$ – вартість послуги у році до впровадження інформаційної системи, приймемо 10000,00 грн;

$\pm\Delta Ц_о$ – зміна вартості послуги від впровадження результатів, приймемо зростання на 500,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів

від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [32]:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.20)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =40%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ =(1·500+10000·2000)·0,83·0,4·(1-0,18/100%)=3587441,9 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ =(1·500+10000·(2000+1500))·0,83·0,4·(1-0,18/100%)=6278373,9 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ =(1·500+10000·(2000+1500+1000))·0,83·0,4·(1-0,18/100%)=8072052,1 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.21)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки,

роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =18%;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 3587441{,}9 /(1+0{,}18)^1 + 6278373{,}9 /(1+0{,}18)^2 + 8072052{,}1 /(1+0{,}18)^3 =$$
$$= 12042570{,}35 \text{ грн.}$$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{4.22}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 1378425,73 грн.

$PV = k_{інв} \cdot ЗВ = 2 \cdot 1378425{,}73 = 2756851{,}47$ грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.23}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 12042570,35 грн;

PV – теперішня вартість початкових інвестицій, 2756851,47 грн.

$E_{абс} = ПП - PV = 12042570{,}35 - 2756851{,}47 = 9285718{,}88$ грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені

потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.24)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, грн;

PV – теперішня вартість початкових інвестицій, грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1$ = (1+9285718,88 /2756851,47)$^{1/3}$ – 1 = 0,98.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (4.25)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,1+0,25 = 0,35 < 0,98 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія онтологічного моделювання бази знань з організації бібліотеки» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \qquad (4.26)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,98 = 1 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Висновки до розділу

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Метод шифрування даних на основі їх характеристичних ознак» становить 43 бали, що, свідчить про комерційну важливість проведення даних досліджень оскільки рівень комерційного потенціалу розробки високий.

При оцінюванні рівня конкурентоспроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 4,8 рази.

Також термін окупності становить 1 рік, що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Метод шифрування даних на основі їх характеристичних ознак».

ВИСНОВКИ

В даній роботі було досліджено та розроблено метод шифрування даних, базований на їх характеристичних ознаках. Він використовує унікальні характеристики даних для шифрування, що забезпечує високий рівень безпеки. У ході виконання роботи запропоновано метод шифрування даних, який відрізняється від відомих тим, що використовує перетворення байтів на основі операцій, що керуються характеристичною ознакою даних і забезпечує пришвидщення процесу шифрування.

Результати нашої роботи підтверджують його ефективність та потенціал в різних сферах, де зберігається важлива інформація. Аналізуючи існуючі підходи в області криптографії та методів шифрування, було виявлено переваги та недоліки таких методів і визначено їхній потенціал у сучасних системах збереження даних.

Розроблений у цій роботі метод шифрування дозволяє забезпечити високий рівень безпеки шляхом використання унікальних характеристик даних для захисту конфіденційності. Реалізація цього методу показала його ефективність та потенціал у різних галузях, враховуючи важливість збереження безпеки інформації в сучасному світі. Реалізований шифр можем допомогти у вдосконаленні стандартів криптографії, забезпечуючи більшу безпеку для зберігання та передачі даних.

Додатково, аналіз економічних аспектів підтвердив потенційні переваги використання цього методу, включаючи його вартість впровадження та можливість заощаджень ресурсів. Робота відкриває нові горизонти у сфері криптографії та захисту даних, демонструючи потенційні переваги використання характеристичних ознак для шифрування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dafydd Stuttard, Marcus The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws. 2019, 912 p.
2. Andy Greenberg Sandworm: A New Era of Cyberwar and the Hunt for the Kremlin's Most Dangerous Hackers. Doubleday. 2019, 368 p.
3. Операційні системи. Шеховцов В. А. 2017, 215 ст.
4. Jean-Philippe Aumasson Serious Cryptography: A Practical Introduction to Modern Encryption. 2017, 316 p.
5. William Stallings Cryptography and Network Security: Principles and Practice. 2017, 784 p.
6. William Sutcliffe We See Everything. Bloomsbury. 2018, 272 p.
7. Цмоць І. Г., Батюк А. Є., Яворський А. В., Теслюк Т. В. Система моніторингу технологічних процесів. Львівська політехніка. 2018, 74 с.
8. Michael W Lucas SSH Mastery. Tilted Windmill Press. 2018, 242 p.
9. Олексій Черненко, Сергій Гнатюк, Валентин Петрик Сучасні технології нейролінгвістичного програмування. Центр навчальної літератури. 2020, 240 с.
10. Уілер Тім Алгоритми оптимізації. Діалектика. 2019, 528 с.
11. Роберт Мартін Чиста архітектура. Фабула. 2021, 368 с.
12. Ед Вілсон Моніторинг і аналіз мереж. Лорі. 2021, 350 с.
13. David Kahn The Codebreakers. Scribner. 2017 1200 p.
14. Cath Senker Cybercrime and the Darknet: Revealing the hidden underworld of the internet. Arcturus. 2017, 194 p.
15. Joe Mayo C# Cookbook. O'Reilly Media. 2021, 480c.
16. Debbie Lafferty, Andrei Alexandrescu Generic Programming and Design Patterns Applied. Addison-Wesley Professional. 2017, 360 p.
17. Jort Rodenburg Code Like a Pro in C#. Manning. 2021, 416 p.
18. Kate Fazzini Kingdom of Lies: Unnerving Adventures in the World of Cybercrime. St. Martin's Press. 2019, 256 p.

19. Milenkovic, M. (2020). Internet of Things: Concepts and System Design. Springer.
20. Armstrong, J. (2020). Migrating to AWS: A Manager's Guide: How to Foster Agility, Reduce Costs, and Bring a Competitive Edge to Your Business. O'Reilly Media.
21. Tulloch, M., et al. (2012). Introducing Microsoft System Center 2012 R2. Microsoft Press.
22. Martin, T. (2022). Designing Secure IoT Devices with the Arm Platform Security Architecture and Cortex-M33. Elsevier.
23. D. Catteddu and G. Hogben. Cloud Computing Security Risk Assessment. Technical report, European
24. D. Catteddu and G. Hogben. Cloud Computing Information Assurance Framework. Technical report, European Network and Information Security Agency, November 2009.
25. NIST Special Publication 800-56A: "Шифрування блочного."
26. Alfred J. Menezes; Paul C. van Oorschot; Scott A. Vanstone. Handbook of Applied Cryptography (вид. Fifth printing). CRC Press. ISBN 0-8493-8523.
27. Моргун О.М. Криптографічні методи захисту інформації. [Електронний ресурс] – Режим доступу: http://a-morgun.narod.ru/a10-01/LK02.pdf
28. "Elliptic Curves in Cryptography" by I.A. Antoniou, A.A. Triantafyllidis, and D.C. Varsamis.
29. Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno."Cryptography Engineering: Design Principles and Practical Applications"
30. Simon Singh. 2019"The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography"
31. Shannon Bray "Implementing Cryptography Using Python"
32. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТКИ

72

Додаток А

ПРОТОКОЛ ПЕРЕВІРКИ
МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Метод шифрування даних на основі їх характеристичних ознак

Автор роботи: Гуцуляк Назарій Олегович

Тип роботи: магістерська кваліфікаційна робота

Підрозділ кафедра захисту інформації ФІТКІ
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність – 82,1 %. Схожість – 7,9 %.

Аналіз звіту подібності (відмітити потрібне):

√ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

☐ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

☐ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Валентина КАПЛУН
(підпис)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Назарій ГУЦУЛЯК
(підпис)

Керівник роботи _____ Володимир ЛУЖЕЦЬКИЙ
(підпис)

Додаток Б

Текст програм

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace NHCypher
{
    public class BytesOperations
    {
        public byte XOR(byte byte1, byte byte2)
        {
            return (byte)(byte1 ^ byte2);
        }
        public byte shiftedRight(int number, byte byte1)
        {
            return (byte)(byte1 >> number);
        }
        public byte shiftedLeft(int number, byte byte1)
        {
            return (byte)(byte1 << number);
        }
        public byte NOT(byte byte1)
        {
            return (byte)(~byte1);
        }
        public byte AND(byte byte1, byte byte2)
        {
            return (byte)(byte1 & byte2);
        }
        public byte OR(byte byte1, byte byte2)
        {
            return (byte)(byte1 | byte2);
        }
        public byte modulus(int number, byte byte1)
        {
            return (byte)(byte1 % number);
        }
    }
}
using System;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using NUnit.Framework.Internal;
```

```
namespace NHCypher
{
    public class Common
    {
        public readonly byte[] key;
        public readonly byte[] plainTextBytes;
        public readonly BytesOperations _Operations;

        public Common(int keyLength, string data)
        {
            key = Generate64BitKey(keyLength);
            plainTextBytes = Encoding.UTF8.GetBytes(data);
            _Operations = new BytesOperations();
        }

        public byte[] Generate64BitKey(int keyLength)
        {
            byte[] key = new byte[keyLength];

            using (RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider())
            {
                rng.GetBytes(key);
            }

            return key;
        }



        public byte[] Encrypt(byte[] key, byte[] plainTextBytes)
        {
            byte encryptedTextByte;
            byte[] result = { };
            foreach (var b in plainTextBytes)
            {
                Console.WriteLine(b);
            }
            Console.WriteLine($"Encrypt");
            StringBuilder builder = new StringBuilder();
            for (int p = 0; p < plainTextBytes.Length; p++)
            {
                byte b1;
                byte b2;


                if (p > key.Length - 1)
                {
                    b1 = key.First();
                    b2 = plainTextBytes[p];

                }
```

```
                else
                {
                    b1 = key[p];
                    b2 = plainTextBytes[p];

                }

                int count = CountSetBits(b1);


                builder.Append(count.ToString());

                switch (count)
                {
                    case 0:
                        encryptedTextByte = _Operations.shiftedLeft(2, b2);
                        result = AddByteToArray(result, encryptedTextByte);
                        break;
                    case 1:
                        encryptedTextByte = _Operations.shiftedLeft(1, b2);
                        result = AddByteToArray(result, encryptedTextByte);
                        break;
                    case 2:
                        encryptedTextByte = _Operations.XOR(b1, b2);
                        result = AddByteToArray(result, encryptedTextByte);
                        break;
                    case 3:
                        encryptedTextByte = _Operations.modulus(256, _Operations.AND(b1, b2));
                        result = AddByteToArray(result, encryptedTextByte);
                        break;
                    case 4:
                        encryptedTextByte = _Operations.XOR(b1, _Operations.NOT(b2));
                        result = AddByteToArray(result, encryptedTextByte);
                        break;
                    case 5:
                        encryptedTextByte = _Operations.XOR(_Operations.NOT(b1), b2);
                        result = AddByteToArray(result, encryptedTextByte);
                        break;
                    case 6:
                        encryptedTextByte = _Operations.modulus(256, _Operations.AND(_Operations.NOT(b1),
_Operations.NOT(b2)));
                        result = AddByteToArray(result, encryptedTextByte);
                        break;
                    case 7:
                        encryptedTextByte = _Operations.shiftedRight(2, b2);
                        result = AddByteToArray(result, encryptedTextByte);
                        break;
                    case 8:
                        encryptedTextByte = _Operations.shiftedRight(1, b2);
                        AddByteToArray(result, encryptedTextByte);
                        break;
```

```
            default:
                break;
        }
    }
    Console.WriteLine($"count = {builder}");
    foreach (var b in result)
    {
        Console.WriteLine(b);
    }
    return result;
}

public string Decrypt(byte[] key, byte[] encryptedTextBytes)
{
    byte decryptedTextByte;
    byte[] result = { };
    string log;
    Console.WriteLine($"Decrypt");
    StringBuilder builder = new StringBuilder();
    for (int e = 0; e < encryptedTextBytes.Length; e++)
    {
        byte b1;
        byte b2;
        if (e > key.Length - 1)
        {
            b1 = key.First();
            b2 = encryptedTextBytes[e];

        }
        else
        {
            b1 = key[e];
            b2 = encryptedTextBytes[e];

        }

        int count = CountSetBits(b1);


        builder.Append(count.ToString());

        switch (count)
        {
            case 0:
                decryptedTextByte = _Operations.shiftedRight(2, b2);
                result = AddByteToArray(result, decryptedTextByte);
                break;
            case 1:
                decryptedTextByte = _Operations.shiftedRight(1, b2);
                result = AddByteToArray(result, decryptedTextByte);
                break;
```

```
                    case 2:
                        decryptedTextByte = _Operations.XOR(b1, b2);
                        result = AddByteToArray(result, decryptedTextByte);
                        break;
                    case 3:
                        decryptedTextByte = _Operations.modulus(256, _Operations.OR(b2, b1));
                        result = AddByteToArray(result, decryptedTextByte);
                        break;
                    case 4:
                        decryptedTextByte = _Operations.XOR(b1, _Operations.NOT(b2));
                        result = AddByteToArray(result, decryptedTextByte);
                        break;
                    case 5:
                        decryptedTextByte = _Operations.XOR(_Operations.NOT(b1), b2);
                        result = AddByteToArray(result, decryptedTextByte);
                        break;
                    case 6:
                        decryptedTextByte = _Operations.modulus(256, _Operations.OR(_Operations.NOT(b2),
_Operations.NOT(b1)));
                        result = AddByteToArray(result, decryptedTextByte); ;
                        break;
                    case 7:
                        decryptedTextByte = _Operations.shiftedLeft(2, b2);
                        result = AddByteToArray(result, decryptedTextByte);
                        break;
                    case 8:
                        decryptedTextByte = _Operations.shiftedLeft(1, b2);
                        result = AddByteToArray(result, decryptedTextByte);
                        break;
                    default:
                        break;
                }
            }
            Console.WriteLine($"count = {builder}");
            foreach (var b in result)
            {
                Console.WriteLine(b);
            }
            return Encoding.UTF8.GetString(result);
        }

        public byte[] AddByteToArray(byte[] originalArray, byte byteToAdd)
        {
            byte[] newArray = new byte[originalArray.Length + 1];

            Array.Copy(originalArray, newArray, originalArray.Length);

            newArray[newArray.Length - 1] = byteToAdd;

            return newArray;
        }
```

```
        public int CountSetBits(byte value)
        {
            // mod 8
            ///
            int count = 0;

            for (int i = 0; i < 8; i++)
            {
                if ((value & (1 << i)) != 0)
                {
                    count++;
                }
            }

            return count;
        }
    }
}
public static int CalculateModulo(byte value, int modulo)
        {
            // Обчислення остачі від ділення байту на вказаний модуль
            int result = value % modulo;
            return result;
  }
```

Додаток В

ІЛЮСТРАТИВНА ЧАСТИНА

Мета та задачі магістерської кваліфікаційної роботи

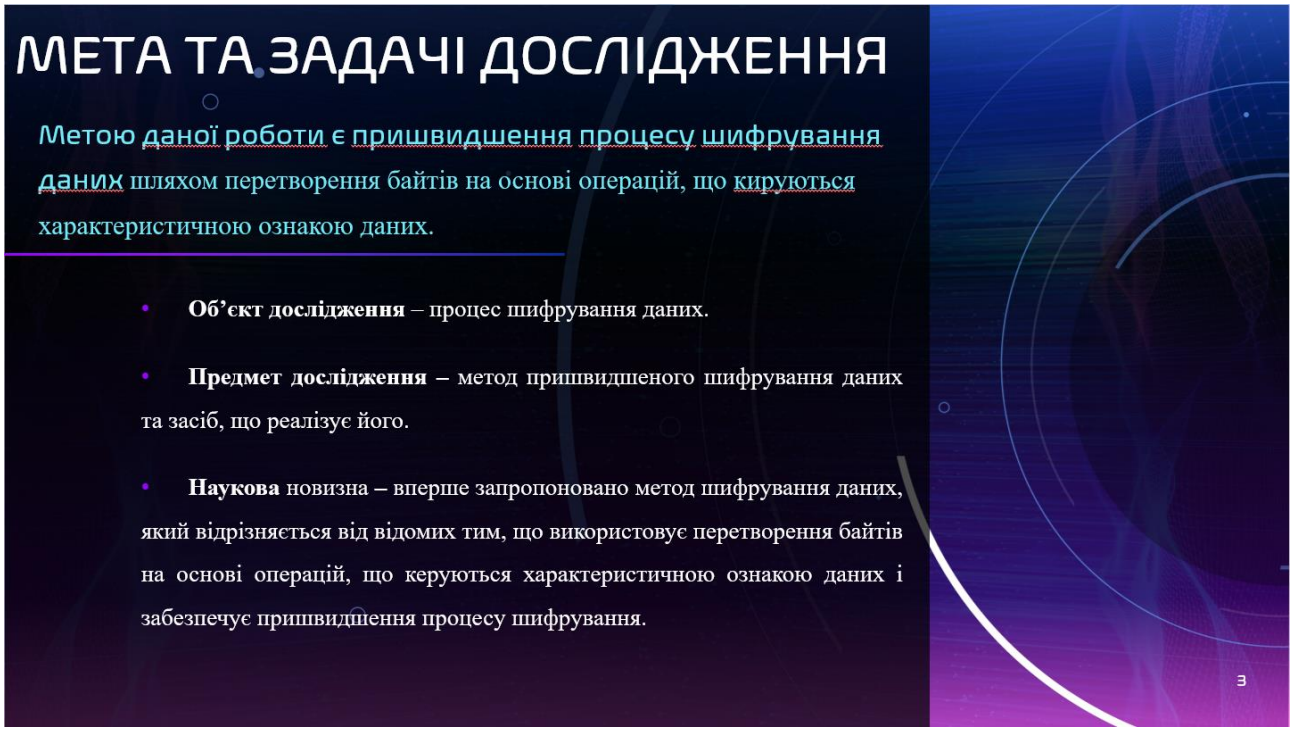


Схема процесу зашифрування

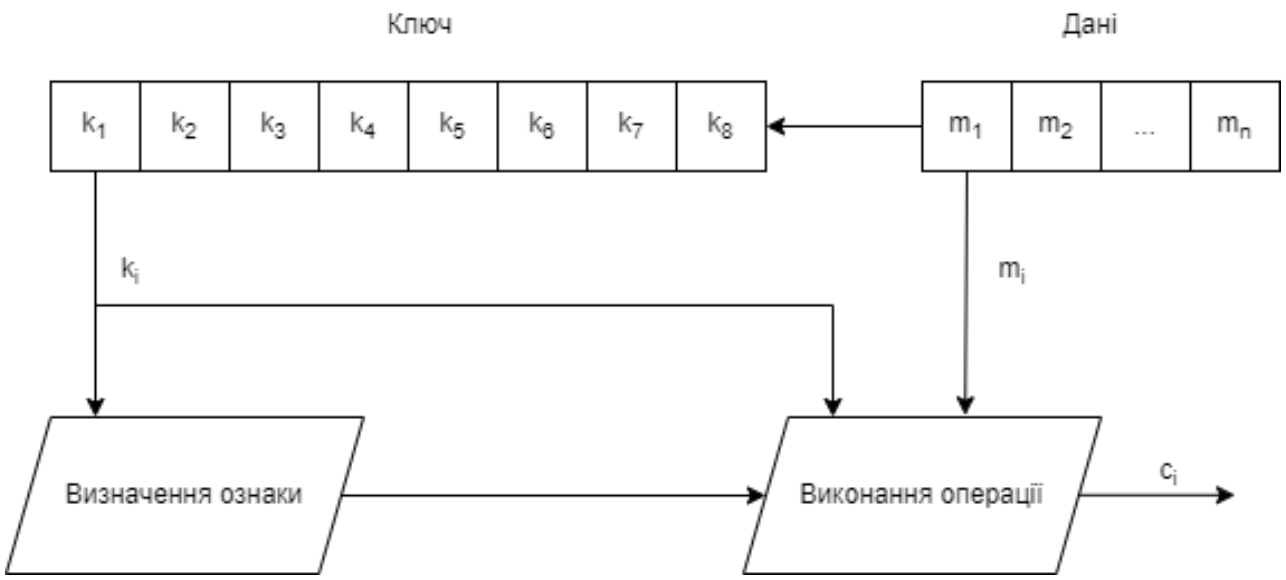
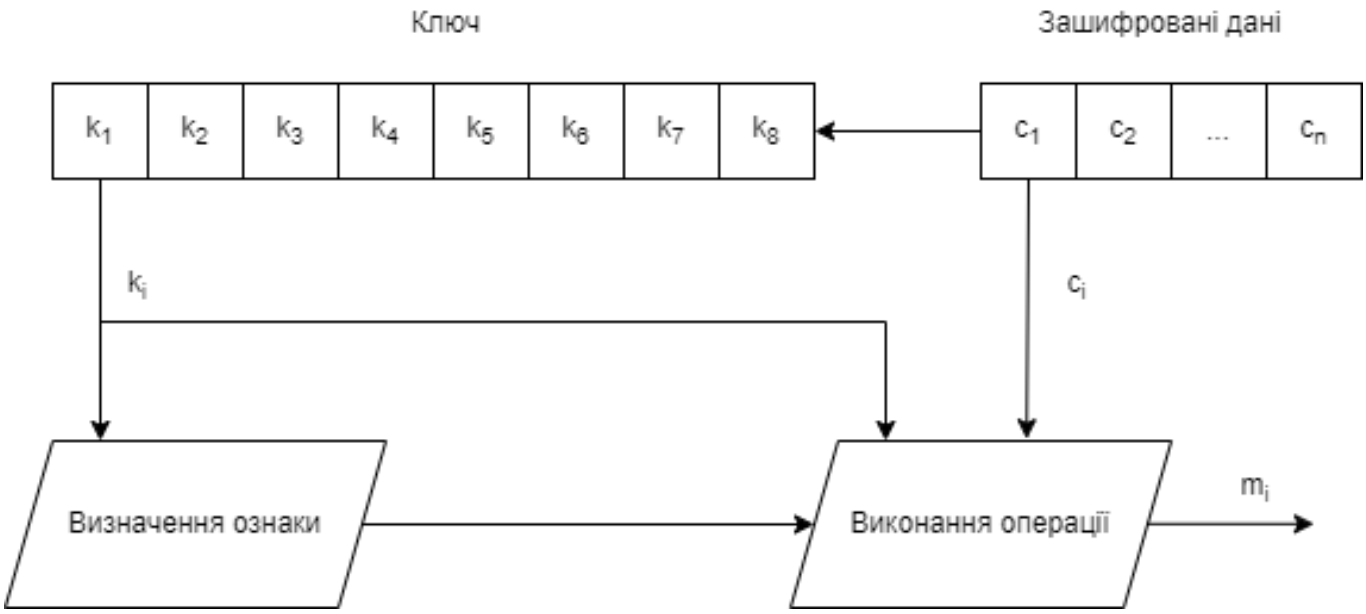


Схема процесу розшифрування



Байтові операції для зашифрування та розшифрування

Байтові операції

k_i - байт ключа

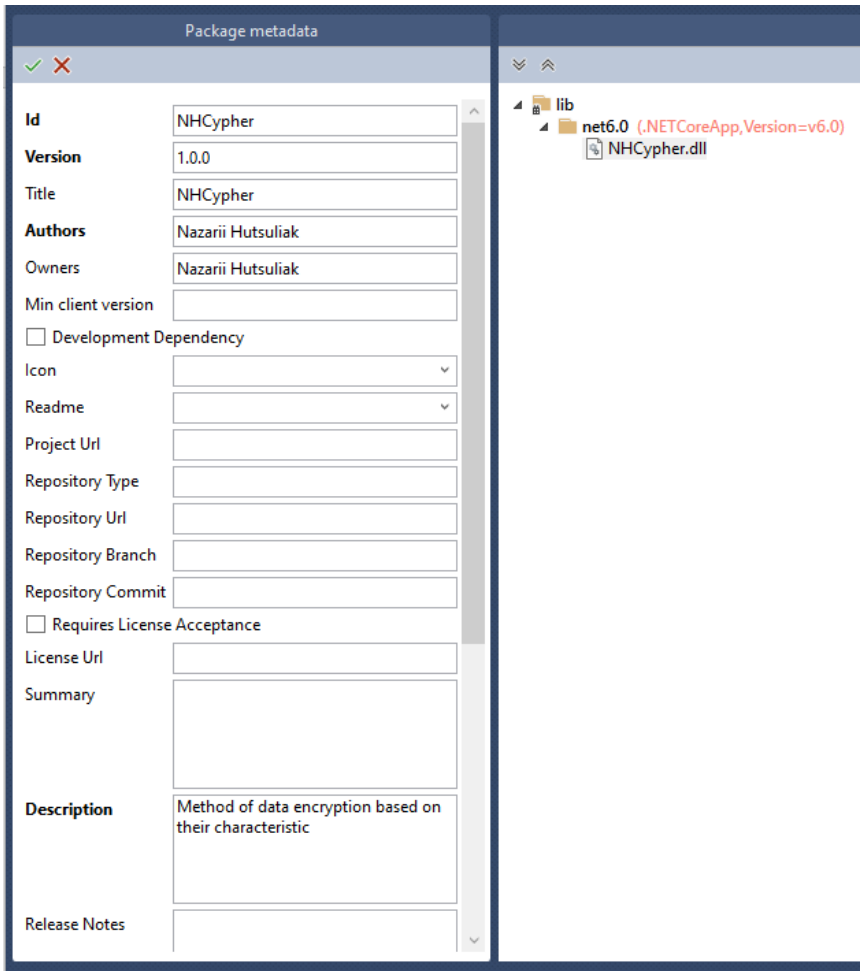
m_i - байт даних, що шифруються

c_i - байт шифрованих даних, що розшифровуються

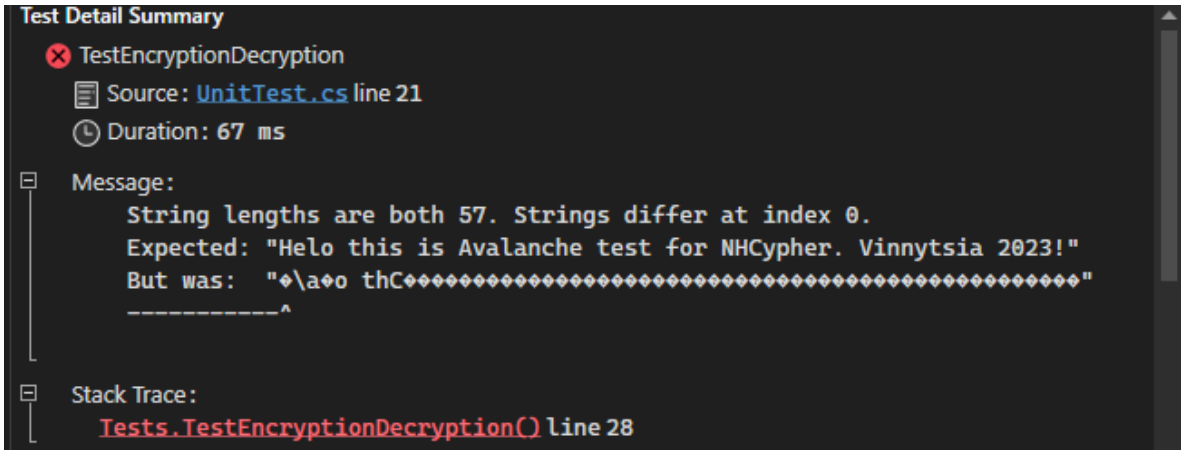
A - байт характеристичну ознаку якого аналізує метод

A mod 8	Операція для шифрування	Операція для розшифрування
0	k_i XOR m_i	k_i XOR c_i
1	$(k_i + m_i) \bmod 256$	$(c_i - k_i) \bmod 256$
2	k_i XOR (NOT m_i)	k_i XOR (NOT c_i)
3	(NOT k_i) XOR m_i	(NOT k_i) XOR c_i
4	((NOT k_i) + (NOT m_i)) mod 256	((NOT c_i) - (NOT k_i)) mod 256
5	(k_i>> 2) XOR (m_i << 2)	(k_i << 2) XOR (c_i>> 2)
6	(NOT k_i + m_i) mod 256	(c_i - NOT k_i) mod 256
7	NOT k_i XOR m_i	NOT k_i XOR c_i

Сформований пакет із реалізованим методом шифрування



Результат тестування лавинного ефекту


Test Detail Summary
TestEncryptionDecryption
Source: UnitTest.cs line 21
Duration: 67 ms
Message:
String lengths are both 57. Strings differ at index 0.
Expected: "Helo this is Avalanche test for NHCypher. Vinnytsia 2023!"
But was: "◆\a◆o thC◆◆◆