МАГІСТЕРСЬКА РОБОТА

МР. ІПм - 16.00.00.000 ПЗ

Група ІПм-24-1

Гуменяк Арсен

2025

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Гуменяк Арсен Іванович

(прізвище, ім'я, по батькові)

УДК 004.9

(індекс)

МАГІСТЕРСЬКА РОБОТА

Постквантові моделі та методи криптоаналізу даних

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Гуменюк А.І.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Романишин Тарас Любомирович, к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Нормоконтроль

доц. Вовк Р.Б.

(посада) (підпис) (дата) (ініціали та прізвище)

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

Івано-Франківськ – 2025

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**

доц. В.В. Бандура

“ 04 ” вересня 2025 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Гуменяк Арсен Іванович

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи “ Постквантові моделі та методи криптоаналізу даних”

керівник проекту (роботи) Романишин Т.Л., к.т.н., доцент

затверджені наказом закладу вищої освіти від “ 05 ” листопада 2025 р. № 695/7

2. Строк подання студентом проекту (роботи) 15 грудня 2025 р.

3. Вихідні дані до проекту (роботи) Концепції та формальні моделі побудови функціонування інформаційних технологій криптографічного аналізу даних

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

1. Дослідження предметної області криптографічного аналізу даних та постквантових моделей
2. Дослідження моделей, методів та концепцій криптографічного шифрування
3. Імплементація методів криптоаналізу даних для розробки алгоритму шифрування
4. Емпіричний аналіз продуктивності та ефективності пропонованого алгоритму шифрування

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Графічна інтерпретація багатовимірної криптографії (рис. 1.1)
2. Алгоритм перевірки цілісності даних за допомогою геш-функції (рис. 1.2)
3. Візуалізація криптографії на основі лізогеній (рис. 1.3)
4. Алгоритм роботи блокового шифру (рис. 1.4)
5. Приклад роботи потокового шифру (рис. 1.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Перевірка на плагіат	доц., к.т.н. Вовк Р.Б.	

7. Дата видачі завдання 04 вересня 2025 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури по темі магістерської роботи	15.09.2025	виконано
2	Дослідження предметної області криптографічного аналізу даних та постквантових моделей	29.09.2025	виконано
3	Дослідження моделей, методів та концепцій криптографічного шифрування	15.10.2025	виконано
4	Імплементація методів криптоаналізу даних для розробки алгоритму шифрування	08.11.2025	виконано
5	Емпіричний аналіз продуктивності та ефективності пропонованого алгоритму шифрування	22.11.2025	виконано
6	Затвердження пояснювальної записки роботи завідувачем кафедри	14.12.2025	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 80 с., 16 рис., 10 табл., 36 джерел.

Тема: Постквантові моделі та методи криптоаналізу даних

Мета дослідження - розробка, формалізація та експериментальна оцінка постквантового симетричного алгоритму шифрування, що забезпечує підвищену стійкість до криптоаналізу.

Об'єкт дослідження - процеси криптографічного захисту даних у постквантових обчислювальних середовищах.

Предмет дослідження - моделі, методи та алгоритми симетричного шифрування і криптоаналізу даних, адаптовані до умов постквантової криптографії.

Результати дослідження

В роботі виконано формалізацію процедури генерації ключів, яка мінімізує ризик прогнозованості за рахунок комбінування випадкових і криптографічно модифікованих компонент.

Висновок

Формалізовано та реалізовано власний симетричний алгоритм шифрування, побудований з урахуванням постквантових принципів безпеки. Алгоритм характеризується комбінованим підходом до генерації ключів, що поєднує випадкові параметри та методи їх криптографічного перетворення.

ПОСТКВАНТОВА КРИПТОГРАФІЯ, КРИПТОАНАЛІЗ, СИМЕТРИЧНЕ ШИФРУВАННЯ, АЛГОРИТМ ШИФРУВАННЯ, КРИПТОСТІЙКІСТЬ, ГЕНЕРАЦІЯ КЛЮЧІВ, ІНФОРМАЦІЙНА БЕЗПЕКА

ABSTRACT

Master Thesis: 80 pp., 16 fig., 10 tab., 36 sources.

Topic: Post-quantum models and methods of data cryptanalysis

The purpose of the study is the development, formalization and experimental evaluation of a post-quantum symmetric encryption algorithm that provides increased resistance to cryptanalysis.

The object of the study is the processes of cryptographic data protection in post-quantum computing environments.

The subject of the study is models, methods and algorithms of symmetric encryption and data cryptanalysis adapted to the conditions of post-quantum cryptography.

Research results

The paper formalizes key generation procedures, which minimizes the risk of predictability by combining random and cryptographically modified components.

Conclusion

A proprietary symmetric encryption algorithm built taking into account post-quantum security principles has been formalized and implemented. The algorithm is characterized by a combined approach to key generation that causes random parameters and methods of their cryptographic transformation.

POST-QUANTUM CRYPTOGRAPHY, CRYPTANALYSIS, SYMMETRIC ENCRYPTION, ENCRYPTION ALGORITHM, CRYPTOSTABILITY, KEY GENERATION, INFORMATION SECURITY

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

R-LWE - Robust Lightweight Encryption

ATM - Automated Teller Machine

uot - Unified Office Text

PHT - Pseudo-Hadamard Transform

SAFER - Secure and Fast Encryption Routine

CAST - Carlisle Adams and Stafford Tavares

3DES - Triple Data Encryption Algorithm

IPKE - Inverse Public Key Encryption

RSA - Ron Rivest, Adi Shamir and Leonard Adleman

LWC - Lightweight Cryptography

RFID - Radio-Frequency Identification

JTC - Joint Technical Committee

ВСТУП

Актуальність теми.

Сучасний розвиток інформаційних технологій супроводжується безпрецедентним зростанням обсягів даних, що обробляються, та ускладненням способів їхнього захисту. У контексті глобальної цифровізації питання забезпечення конфіденційності, цілісності та автентичності інформації набуває стратегічного значення. Класичні криптографічні системи, які протягом десятиліть вважалися надійними, втрачають свою ефективність через появу нових обчислювальних парадигм, зокрема квантових комп'ютерів. Алгоритми Шора та Гровера демонструють потенційну здатність розв'язувати задачі, які раніше вважалися обчислювально нездійсненними, що ставить під загрозу традиційні механізми захисту даних, засновані на факторизації та дискретному логарифмі.

У цьому контексті постквантова криптографія (Post-Quantum Cryptography, PQC) постає як новий напрям досліджень, спрямований на створення моделей і алгоритмів, здатних протистояти квантовим атакам. Її мета полягає у розробці методів, що зберігають високий рівень безпеки навіть у постквантовому середовищі, при цьому залишаючись сумісними з класичними обчислювальними платформами. Зокрема, особливої уваги потребує розвиток симетричних алгоритмів шифрування, які можуть бути адаптовані до нових вимог без суттєвих втрат у продуктивності.

Магістерська робота присвячена дослідженню постквантових моделей і методів криптоаналізу даних, спрямованих на підвищення стійкості алгоритмів шифрування та створення ефективних рішень для захисту інформації у майбутніх квантових обчислювальних системах.

Актуальність дослідження зумовлена стрімким розвитком квантових технологій, що призводить до появи нових загроз у сфері інформаційної безпеки. У той час як сучасні криптографічні системи — такі як RSA, ECC

або AES — покладаються на складність певних математичних задач, квантові алгоритми потенційно здатні зруйнувати ці припущення. Це створює критичну потребу у переході до криптографічних рішень, стійких до атак з боку квантових комп'ютерів.

Додатковим аспектом актуальності є необхідність забезпечення високої ефективності шифрування в умовах обмежених ресурсів, зокрема в системах Інтернету речей (IoT), де полегшені (lightweight) криптографічні схеми відіграють ключову роль. Поєднання цих напрямів — постквантової стійкості та ефективності — визначає перспективність дослідження.

Таким чином, робота є актуальною як у теоретичному, так і у прикладному аспектах, оскільки сприяє розв'язанню однієї з найважливіших проблем сучасної кібербезпеки — створенню адаптивних криптографічних систем, здатних протистояти майбутнім квантовим загрозам.

Метою магістерської роботи є розробка, формалізація та експериментальна оцінка постквантового симетричного алгоритму шифрування, що забезпечує підвищену стійкість до криптоаналізу.

Об'єктом дослідження є процеси криптографічного захисту даних у постквантових обчислювальних середовищах.

Предметом дослідження є моделі, методи та алгоритми симетричного шифрування і криптоаналізу даних, адаптовані до умов постквантової криптографії.

Для досягнення **поставленої мети необхідно виконати такі завдання**:

- Провести теоретичний аналіз стану сучасних криптографічних систем та оцінити їхню вразливість до квантових атак.

- Дослідити математичні основи постквантових моделей криптографії та визначити їхні переваги й обмеження.

- Розробити методологічні принципи побудови симетричного постквантового алгоритму шифрування.

- Реалізувати експериментальний зразок алгоритму та здійснити тестування його роботи з різними типами даних.

- Провести емпіричний аналіз продуктивності та стійкості розробленого алгоритму до криптоаналітичних впливів.

У роботі застосовано комплекс методів:

- аналітичний метод — для аналізу теоретичних основ криптографії та постквантових моделей;

- порівняльний аналіз — для оцінки ефективності існуючих алгоритмів шифрування;

- моделювання — для побудови математичної моделі алгоритму симетричного шифрування;

- експериментальний метод — для тестування продуктивності розробленого алгоритму на різних наборах даних.

Наукова новизна роботи полягає у розробленні моделі симетричного алгоритму шифрування, що поєднує постквантові принципи безпеки з підвищеною обчислювальною ефективністю.

Практичне застосування результатів

Результати дослідження можуть бути використані для розробки та впровадження систем шифрування у постквантових інформаційних інфраструктурах; підвищення рівня безпеки комунікаційних систем у сферах фінансів, охорони здоров’я та створення гібридних криптографічних рішень, сумісних із класичними протоколами.

Структура магістерської роботи. Робота складається зі вступу, трьох розділів, висновків та додатків. Загальний обсяг роботи становить 80 сторінок, і містить 16 рисунків, 10 таблиць, список використаних джерел із 36 найменувань.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ КРИПТОГРАФІЧНОГО АНАЛІЗУ ДАНИХ ТА ПОСТКВАНТОВИХ МОДЕЛЕЙ

1.1. Постановка задачі проектування та оцінки продуктивності криптографічного алгоритму

В умовах стрімкої цифровізації та динамічного розвитку інформаційних технологій, забезпечення безпеки цифрових даних набуває першочергової важливості. Зі зростанням обсягів конфіденційної комунікації зростає і потреба у надійних засобах захисту каналів зв'язку. Ключову роль у захисті інформації, зокрема конфіденційних документів, від несанкціонованого доступу відіграють криптографічні методи. Криптографічні алгоритми класифікують на два основні типи: асиметричні (з відкритим ключем) та симетричні.

У даній роботі запропоновано новий надійний та полегшений криптографічний алгоритм симетричного шифрування. Метою розробки є підвищення рівня безпеки за одночасного зниження обчислювальних витрат. Дослідження проводилося з використанням дедуктивного методу, що включав такі етапи: аналіз існуючих алгоритмів симетричного шифрування, формулювання гіпотези, розробка нового алгоритму, проведення експериментальної перевірки на наборах даних, аналіз отриманих результатів та формулювання висновків.

Результати експериментів демонструють, що запропонований алгоритм демонструє вищу продуктивність та нижчий рівень помилок порівняно з більшістю існуючих полегшених шифрів. Його надійність та валідність підтверджено тестуванням на значному масиві даних. Обмеженням запропонованого підходу є його ефективність переважно для обробки невеликих обсягів даних. Проте, виходячи з проведеного аналізу, можна стверджувати, що алгоритм є перспективним рішенням для індивідуальних

користувачів та малих підприємств, які потребують ефективного захисту даних за мінімальних витрат.

1.2. Проблематика та актуальність захисту даних в сучасних інформаційних системах

В умовах перманентної цифровізації, безпека інформаційних технологій є динамічною галуззю досліджень, що характеризується появою нових сфер застосування та постійним удосконаленням апаратно-програмних платформ.

З розвитком інформаційно-комунікаційних технологій зростає критична значущість захищеної передачі даних, що висуває підвищені вимоги до рівня її безпеки. Криптографія є фундаментальною дисципліною, що забезпечує теоретичну та практичну основу для захисту інформації. Вона є ключовим компонентом безпеки комунікацій та базовим елементом сучасної архітектури інформаційної безпеки.

Основна функція криптографії полягає у захисті конфіденційності комунікації від несанкціонованого доступу, де методи шифрування виступають основним інструментом для запобігання атакам. Починаючи з XX століття, криптографія еволюціонувала в одну з ключових наукових галузей. У цьому розділі детально аналізуються проблеми інформаційної безпеки та обґрунтовуються передумови для дослідження методів симетричного шифрування та дешифрування на даних.

Інформаційна безпека відіграє центральну роль у функціонуванні сучасних цифрових комунікацій, зокрема в системах електронних транзакцій. Хоча її необхідність може бути неочевидною для неспеціалізованих користувачів, вона є абсолютною вимогою в умовах щоденного обміну даними між мільйонами суб'єктів [1]. Існують різноманітні криптографічні методи, що забезпечують інструментарій для безпечної комерції, приватних комунікацій та захисту автентифікаційних даних [2].

Криптографічні алгоритми широко інтегровані у фінансовий сектор для гарантування безпеки транзакцій. Приклади включають захист конфіденційності та цілісності даних, таких як персональний ідентифікаційний номер (PIN), у транзакціях через банкомати (ATM), а також автентифікацію сторін в системах інтернет-банкінгу, що детально розглянуто в роботах [1–4]. Загальноприйнятою є класифікація криптографічних алгоритмів на два основні типи: асиметричні (з відкритим ключем) та симетричні (з симетричним ключем) [5].

Симетричні шифри переважно використовуються для забезпечення конфіденційності, тоді як для гарантування цілісності та автентичності застосовуються коди автентифікації повідомлень (MAC) або цифрові підписи. Потреба в таких механізмах захисту існує не лише у великих корпоративних системах, а й на рівні індивідуальних користувачів та малих підприємств. Для цих категорій користувачів існуючі криптографічні рішення часто є надмірними з точки зору обчислювальних ресурсів.

Дослідження, зокрема праця [5], підтверджують дефіцит криптографічних алгоритмів, оптимізованих для ефективної роботи з файлами малого розміру (наприклад, текстовими документами) при збереженні достатнього рівня надійності. Всупереч поширеній думці, що криптографія орієнтована переважно на великі масиви даних, для невеликих обсягів вимоги до мінімізації часової складності та забезпечення швидкого криптоаналізу стають не менш критичними. Це визначає актуальність даного дослідження та формує логічне підґрунтя для його проведення.

1.3. Аналіз задачі розробки алгоритму симетричного шифрування для криптографічного захисту даних

Проблема даного дослідження полягає у дефіциті надійних, полегшених криптографічних алгоритмів, що ускладнює захист даних малого обсягу в освітніх та комерційних організаціях. Робота включає аналіз

існуючих підходів до розробки симетричних шифрів та зосереджена на прототипуванні нового алгоритму, призначеного для швидкої криптографічної обробки даних обсягом від 1 до 512 КБ на кросплатформенній основі (Linux, Windows, Mac OS). Таким чином, ключова мета дослідження — розробити надійний та ефективний алгоритм симетричного шифрування, який дозволить організаціям захищати дані малого обсягу та протидіяти несанкціонованому доступу за одночасного зниження обчислювальної складності.

1.3.1. Наукова та практична значущість дослідження

Значущість дослідження визначається його внеском у вирішення актуальних завдань у сфері мережевих систем та інформаційної безпеки. З огляду на обмежену кількість алгоритмів, оптимізованих для шифрування даних малого обсягу на різних операційних системах, дана робота пропонує доступне рішення, що забезпечує стабільність, інтеграцію та конфіденційність таких даних.

Це є особливо актуальним для малих підприємств, для яких інвестиції у дороге програмне забезпечення безпеки є фінансово невиправданими. Як зазначається в [6], розробка полегшених, надійних та швидких алгоритмів шифрування стане пріоритетним напрямком у найближчі роки. Цю тезу підтверджують у дослідженні алгоритму Hummingbird [7], вказуючи, що прототип полегшеного симетричного шифру не лише прискорює криптографічну обробку даних, але й зменшує часові та обчислювальні витрати.

Отже, дане дослідження спрямоване на надання організаціям економічно ефективної криптографічної альтернативи, сумісної з різними ОС. Очікується, що розроблений алгоритм буде здатен обробляти різні формати файлів (.doc, .docx, .txt) розміром 1–512 КБ, слугуючи валідним інструментом для захисту даних. Таким чином, дослідження робить вагомий

внесок у розвиток симетричної криптографії, пропонуючи надійне та ефективне рішення.

1.3.2. Межі та обсяг дослідження

Це дослідження має експериментальний характер. Воно включало аналіз існуючих алгоритмів симетричного шифрування за такими параметрами, як часова та просторова складність, архітектурні переваги та недоліки, а також кількість раундів шифрування. Найкращі характеристики проаналізованих алгоритмів були синтезовані для розробки нового алгоритму шифрування, орієнтованого на дані малого обсягу.

Обсяг дослідження чітко обмежений тестуванням на масивах даних розміром від 1 до 512 КБ у форматах .doc, .docx, .txt та їх аналогах. Експериментальна перевірка проводилася на трьох операційних системах: Windows, Linux та Mac OS. Очікується, що результати дослідження матимуть значний вплив на сферу захисту даних малого обсягу — сегмент, який досі залишався недостатньо дослідженим, — пропонуючи рішення зі зниженою складністю та підвищеною часовою ефективністю.

1.4. Особливості постквантових моделей та пост квантової криптографії

Постквантові моделі (або постквантова криптографія, PQC) — це криптографічні алгоритми, які розроблені для роботи на класичних комп'ютерах, але при цьому є стійкими до атак з боку як класичних, так і майбутніх квантових комп'ютерів. Головна ідея — замінити поточні криптографічні стандарти на нові, математичні проблеми яких залишаються складними навіть для квантових обчислювачів.

Сучасна асиметрична криптографія (алгоритми RSA, ECC), яка захищає майже все — від банківських транзакцій до вебсайтів (HTTPS) та месенджерів — спирається на дві основні математичні проблеми:

- Проблема факторизації, оскільки дуже складно розкласти велике число на два прості множники.

- Проблема дискретного логарифму полягає у складності знайти показник степеня в певній математичній групі.

Для класичних комп'ютерів ці задачі є настільки складними, що на їх вирішення потрібні мільйони років. Однак у 1994 році математик Пітер Шор розробив алгоритм Шора, який може вирішити обидві ці проблеми експоненційно швидко, але лише за умови його запуску на достатньо потужному квантовому комп'ютері.

Важливо розуміти, що PQC не потребує квантових комп'ютерів для своєї роботи. Це алгоритми, що працюють на наших звичних пристроях (серверах, ноутбуках, смартфонах), але їх математична основа стійка до злому за допомогою алгоритму Шора та інших квантових алгоритмів.

Вони базуються на абсолютно інших, складніших математичних проблемах.

1.4.1. Основні напрямки (моделі) PQC (Post-Quantum Cryptography)

Існує кілька основних сімейств постквантових алгоритмів, кожне з яких використовує свою унікальну математичну проблему:

- Криптографія на ґратках (Lattice-based) - лідер у стандартизації. Безпека заснована на складності знаходження найкоротшого вектора у багатовимірному просторі (ґратці). Уявіть собі нескінченну тривимірну решітку з точок; завдання — знайти точку, найближчу до центру, що є надзвичайно складно у просторах із сотнями вимірів.

- Криптографія на основі кодів (Code-based) використовує теорію кодів, що виправляють помилки. Шифрування схоже на додавання до повідомлення специфічних, розрахованих "помилок", а розшифрувати його може лише той, хто знає секретний код для виправлення цих помилок.

- Багатовимірна криптографія (Multivariate) базується на складності розв'язання систем нелінійних поліноміальних рівнянь з багатьма змінними.

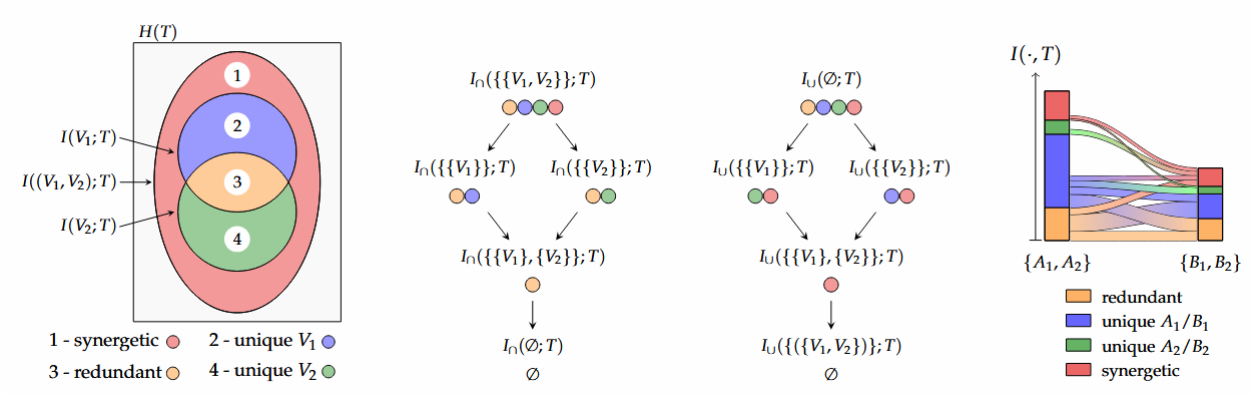

Рис. 1.1. Графічна інтерпретація багатовимірної криптографії

На рис. 1.1 зображено складну систему перетину поверхонь, що описуються рівняннями. Завдання полягає в тому, щоб знайти єдину точку, в якій перетинаються всі ці поверхні одночасно

- Криптографія на основі геш-функцій (Hash-based) - використовує властивості криптографічних геш-функцій (наприклад, SHA-256). Це один із найнадійніших і найстаріших підходів, але його головний недолік — обмежена кількість підписів, які можна згенерувати одним ключем.

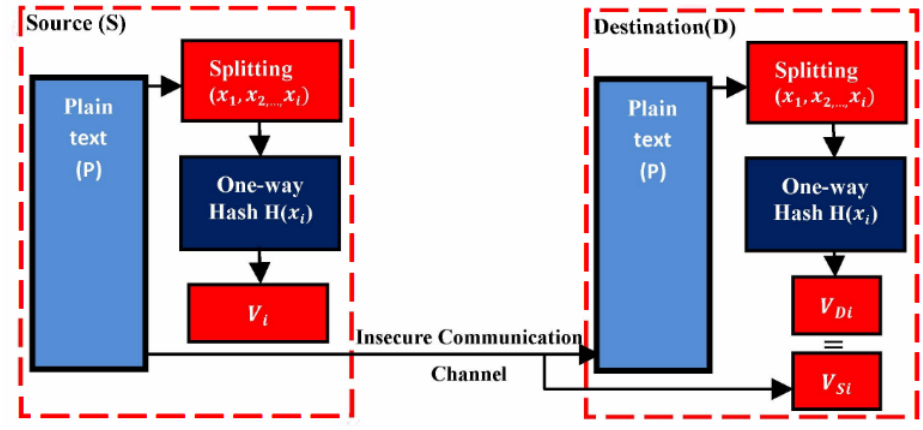

Рис. 1.2. Алгоритм перевірки цілісності даних за допомогою геш-функції

На рис. 1.2 подана схема алгоритму перевірки цілісності даних за допомогою геш-функції. Його головна мета — не зашифрувати дані, а переконатися, що вони не були змінені під час передачі через незахищений канал.

Схема складається з двох частин: Відправник (Source) та Одержувач (Destination).

1. На боці Відправника (S):

Крок 1: Береться вихідне повідомлення (Plain text (P)).

Крок 2: Це повідомлення відправляється одержувачу напряму через незахищений канал зв'язку.

Крок 3 (Паралельно): Повідомлення розбивається на менші частини (Splitting (x_1, x_2, ..., x_i)).

Крок 4: Кожна частина x_i пропускається через односторонню геш-функцію (One-way Hash $H(x_i)$). В результаті для кожного шматочка даних створюється унікальний короткий "відбиток" — геш.

Крок 5: Усі ці геші об'єднуються в єдиний верифікаційний код V_{si} (це і є наша "цифрова пломба").

Крок 6: Цей код V_{si} також відправляється одержувачу.

2. На боці Одержувача (D):

Крок 1: Одержувач отримує два елементи: саме повідомлення P та верифікаційний код V_{si}.

Крок 2: Він бере отримане повідомлення P і виконує з ним ті ж самі дії, що й відправник: розбиває на частини (Splitting) і гешує кожну з них (One-way Hash).

Крок 3: На основі цих гешів він генерує свій власний верифікаційний код V_{ai}.

Крок 4 (Ключовий момент): Одержувач порівнює свій згенерований код V_{ai} з тим кодом V_{si}, який він отримав від відправника.

Результат перевірки:

- Якщо $V_{ai} \equiv V_{si}$ (коди ідентичні), то це означає, що повідомлення дійшло в первозданному вигляді. "Пломба" не пошкоджена, і даним можна довіряти.

- Якщо $V_{ai} \neq V_{si}$ (коди відрізняються) - це сигнал тривоги. Навіть найменша зміна в повідомленні (видалення коми, зміна однієї літери)

повністю змінить геш, і коди не співпадуть. Це означає, що дані були пошкоджені або навмисно змінені в дорозі. Такі дані вважаються недійсними.

Цей метод не забезпечує конфіденційність, оскільки Plain text передається у відкритому вигляді. Його задача — виключно гарантія цілісності.

- Криптографія на основі ізогеній (Isogeny-based) - найновіший і математично складний напрямок, що використовує маршрути між еліптичними кривими. Пропонує відносно короткі ключі, але є менш вивченим.
-

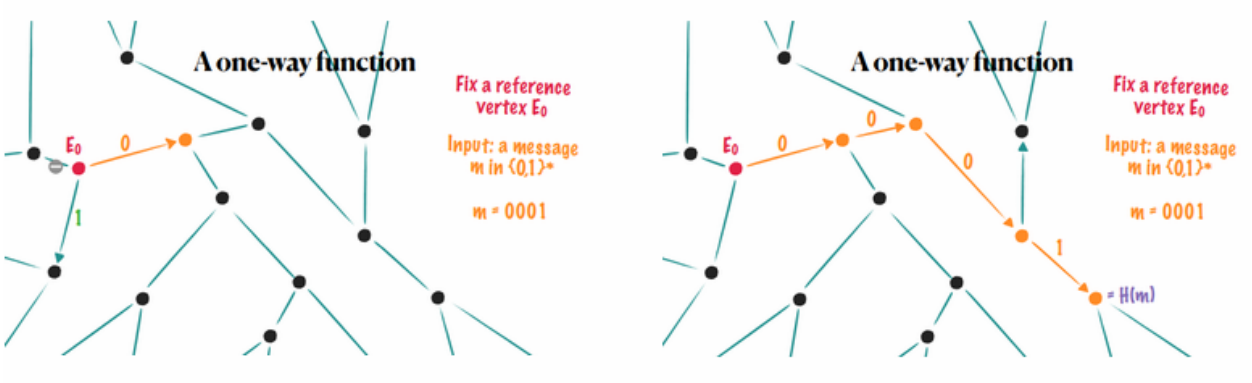

Рис. 1.3. Візуалізація криптографії на основі лізогеній

Візуалізація (рис. 1.3) показує дві різні еліптичні криві та складний, заплутаний "шлях" (ізогенію) між ними. Секрет полягає саме в цьому унікальному маршруті, а не в кінцевих точках.

1.4.2. Важливість постквантових алгоритмів

Хоча повномасштабних квантових комп'ютерів, здатних зламати RSA, ще не існує, загроза є актуальною через атаку "Harvest now, decrypt later" ("Збирай зараз, розшифровуй пізніше"). Зловмисники можуть вже сьогодні перехоплювати та зберігати зашифрований трафік (наприклад, державні таємниці, корпоративні дані), щоб розшифрувати його в майбутньому, коли з'являться відповідні квантові комп'ютери.

Перехід світової інфраструктури на нові криптографічні стандарти — це процес, що триватиме багато років. Тому починати його потрібно заздалегідь.

Головним рушієм у цій сфері є Національний інститут стандартів і технологій США (NIST), який у 2016 році запустив глобальний конкурс для відбору майбутніх постквантових стандартів. У 2022-2024 роках були обрані перші переможці, зокрема алгоритми, засновані на ґратках (CRYSTALS-Kyber для обміну ключами та CRYSTALS-Dilithium для цифрових підписів), які стануть основою для майбутніх стандартів безпеки.

1.5. Аналіз стану досліджень у галузі симетричної криптографії та теоретична основа розробки криптографічних алгоритмів

Огляд наукової літератури є систематичним, експліцитним та відтворюваним методом ідентифікації, оцінки та синтезу існуючого обсягу робіт, опублікованих дослідниками та науковцями [11]. Він являє собою критичний аналіз вторинних джерел, що включає ключові результати, теоретичні концепції та методологічні підходи з обраної теми, і не передбачає представлення нових експериментальних даних. Основна мета цього розділу — не лише реферувати наявні знання, а й виявити невирішені проблеми та "білі плями" (прогалини в знаннях), що обґрунтовує актуальність та необхідність проведення даного дослідження. Якісний огляд літератури є фундаментом для генерації нових ідей та розробки інноваційних підходів у галузі симетричної криптографії.

1.5.1. Структура огляду літератури

Структура даного огляду літератури організована за тематичним принципом для забезпечення логічної послідовності викладу матеріалу. В основу структури покладено концептуальну карту ключових термінів та напрямків дослідження.

1. Теоретичне підґрунтя.

Спочатку розглядається теоретична рамка дослідження, в межах якої визначаються фундаментальні поняття інформаційної безпеки та шифрування. Це створює концептуальну основу для подальшого аналізу.

2. Техніки шифрування.

Детально аналізуються основні техніки симетричного та асиметричного шифрування, їхні переваги, недоліки та сфери застосування.

3. Специфіка полегшеної криптографії.

Далі фокус зміщується на концепцію даних малого обсягу та сучасний стан розробок у сфері полегшеного симетричного шифрування. Аналізуються ключові ініціативи, існуючі алгоритми, їхні масштаби та архітектурні особливості.

4. Прикладне значення.

Оцінюється роль та потенціал застосування полегшеного шифрування в освітньому та корпоративному секторах. Розглядаються проблеми, з якими стикаються організації при впровадженні ресурсоємних систем безпеки, та переваги полегшених альтернатив.

5. Ідентифікація прогалин у знаннях.

Проводиться критична оцінка проаналізованої літератури з метою виявлення недостатньо досліджених аспектів. На основі цього аналізу формулюються дослідницькі питання та уточнюються цілі роботи.

Розділ завершується узагальненням ключових висновків, зроблених на основі огляду літератури.

1.5.2. Теоретичне підґрунтя дослідження

Теоретичне підґрунтя (або теоретична рамка) являє собою структуру, що систематизує дослідження, слугуючи "системою відліку для спостережень, визначення понять, проєктування дослідження та інтерпретації результатів" [13]. Це не просто опис теорій, а їх інтеграція для формування логічної основи, що пояснює досліджувану проблему.

Дане дослідження ґрунтується на синтезі кількох фундаментальних теорій.

1. Теорія інформації. Заснована Клодом Шенноном, вона надає математичний апарат для аналізу комунікаційних систем. Для криптографії ключовими є поняття ентропії (міра невизначеності), надлишковості даних, а також принципи побудови стійких шифрів: розсіювання (diffusion) та перемішування (confusion), які є основою більшості сучасних блокових шифрів.

2. Теорія обчислювальної складності. Ця теорія є центральною для полегшеної криптографії. Вона дозволяє класифікувати алгоритми за необхідними для їх виконання ресурсами (час, пам'ять). Аналіз асимптотичної складності (напр., нотація "великого О") є інструментом для оцінки ефективності алгоритмів та їх придатності для пристроїв з обмеженими обчислювальними можливостями.

3. Фундаментальні принципи криптографії. Дослідження спирається на загальновизнані принципи, зокрема принцип Керкгоффса, згідно з яким надійність криптосистеми має залежати лише від секретності ключа, а не від секретності самого алгоритму.

Така теоретична основа зміцнює дослідження, оскільки вона пов'язує його з існуючим науковим знанням, надає базу для формування гіпотез та вибору методів, а також дозволяє перейти від простого опису явищ до їх узагальнення та пояснення. Вона чітко визначає межі дослідження та ключові змінні, що впливають на аналізовані процеси.

Висновки до розділу

У першому розділі виконано всебічний аналіз предметної області криптографічного аналізу даних та досліджено основні напрями розвитку постквантової криптографії. Визначено проблематику сучасних методів захисту інформації та окреслено їхню вразливість у контексті появи

квантових обчислень. Проведено систематизацію основних моделей PQC, таких як граткові, кодові та ізогенійні схеми, що є перспективними для створення стійких до квантових атак систем. Здійснено постановку задачі проектування алгоритму шифрування, який забезпечує баланс між безпекою та ефективністю. У результаті сформовано теоретичну основу для розробки постквантового симетричного криптографічного алгоритму, що враховує обмеження сучасних систем безпеки.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ МОДЕЛЕЙ, МЕТОДІВ ТА КОНЦЕПЦІЙ КРИПТОГРАФІЧНОГО ШИФРУВАННЯ

2.1. Аналіз криптографічних методів та концепція полегшеного шифрування

2.1.1. Основи інформаційної безпеки та шифрування

Інформаційна безпека (InfoSec) — це науково-практична дисципліна, що охоплює комплекс заходів для захисту інформації від несанкціонованого доступу, використання, модифікації, розголошення чи знищення, незалежно від форми її представлення (електронної чи фізичної). Ключовими принципами інформаційної безпеки є забезпечення конфіденційності, цілісності та доступності даних (тріада CIA). Спорідненим поняттям є гарантія інформації (Information Assurance), яка фокусується на підтримці функціонування систем та збереженні даних під час критичних інцидентів, таких як стихійні лиха чи технічні збої, часто шляхом резервного копіювання. Шифрування є фундаментальним методом забезпечення конфіденційності. Це процес перетворення вихідної інформації, або відкритого тексту (plaintext), у шифротекст (ciphertext) за допомогою криптографічного алгоритму та ключа. Цей процес не запобігає перехопленню даних, але робить їх нечитабельними для неавторизованих осіб. Дешифрування можливе лише за наявності відповідного ключа.

2.1.2. Класифікація алгоритмів шифрування

Криптографічні алгоритми можна класифікувати за кількома основними критеріями. Нижче наведено огляд ключових типів та історично значущих прикладів.

У симетричній криптографії один і той самий ключ використовується як для шифрування, так і для дешифрування. Ці алгоритми поділяються на блокові та потокові.

Блокові шифри обробляють дані блоками фіксованого розміру. Їх розвиток почався з DES (Data Encryption Standard), який був стандартом у США з 1977 року. Через недостатню довжину ключа (56 біт) його було замінено на Triple DES (3DES), що застосовував DES тричі. Потреба у новому, більш надійному стандарті призвела до конкурсу AES (Advanced Encryption Standard), переможцем якого у 2001 році став алгоритм Rijndael. Серед фіналістів конкурсу AES були також потужні алгоритми Serpent та Twofish. Іншими відомими блоковими шифрами є Blowfish (попередник Twofish) та CAST-128.

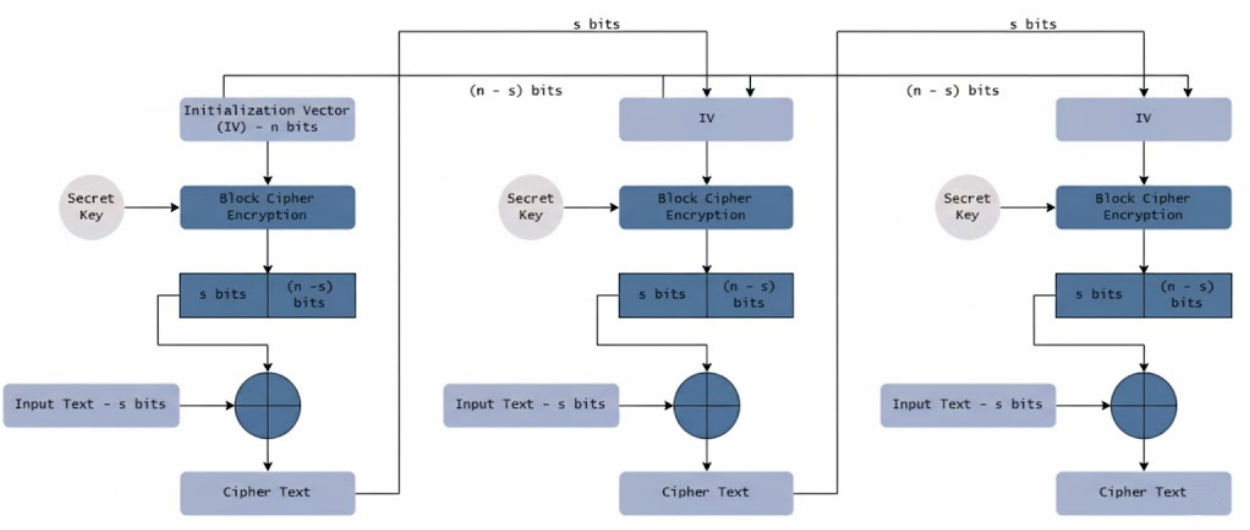

Рис. 2.1. Алгоритм роботи блокового шифру

Ця схема (рис. 2.1) чітко показує, як текст поділяється на рівні частини, і кожна з них проходить через алгоритм шифрування, перетворюючись на блок шифротексту.

Потокові шифри шифрують дані побітово або побайтово в режимі реального часу. Найвідомішим представником є RC4, який завдяки своїй простоті та швидкості був широко поширений, проте на сьогодні має відомі вразливості, що обмежують його використання в нових системах. Генератор псевдовипадкової послідовності створює ключовий потік (keystream), який

потім "накладається" на потік вхідних даних (зазвичай за допомогою операції XOR).

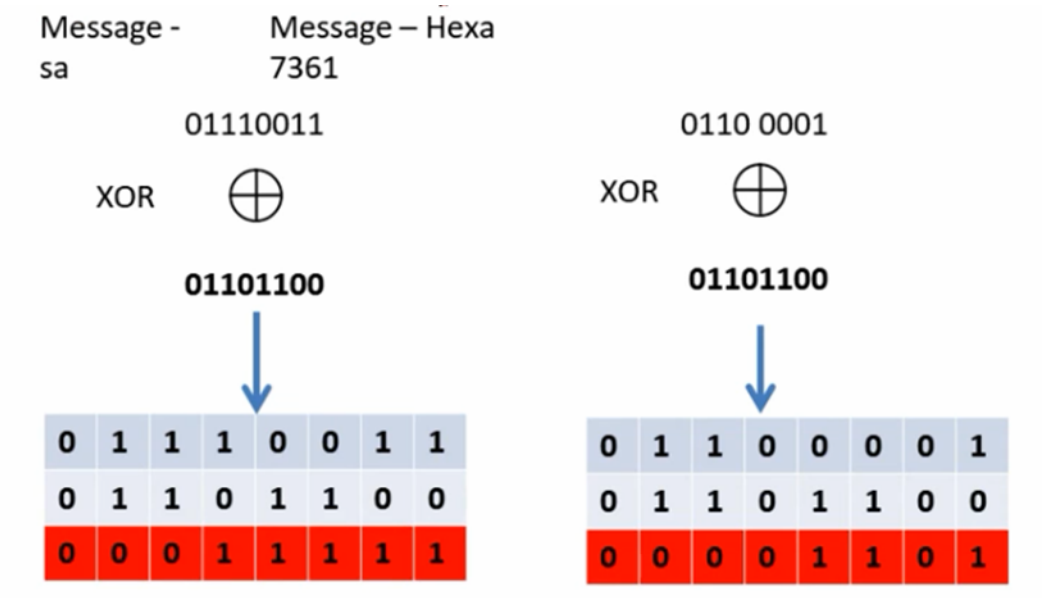

Рис. 2.2. Приклад роботи потокового шифру

На рисунку 2.2 видно, як генератор створює нескінченний потік ключів, і кожен біт цього потоку поєднується з відповідним бітом відкритого тексту, створюючи шифротекст.

В асиметричних системах використовується пара ключів: відкритий (для шифрування) та закритий (для дешифрування). Цей підхід вирішує проблему безпечної передачі ключа.

Обмін ключами Діффі-Геллмана (D-H). Один із перших практичних протоколів, що дозволяє двом сторонам встановити спільний секретний ключ через незахищений канал.

Розглянемо приклад на якому дозволяється двом сторонам (Алісі та Бобу) створити спільний секретний ключ через незахищений канал, навіть якщо їхнє спілкування прослуховується.

Візуалізація даного алгоритму подана на рисунку 2.3. Це класична аналогія з фарбами. Кожен починає зі спільного публічного кольору (жовтий). Додає свій секретний колір (червоний або бірюзовий). Публічно

обмінюється змішаними кольорами. Кожен знову додає свій секретний колір до отриманої суміші. В результаті обидві сторони отримують однаковий кінцевий секретний колір (коричневий), який неможливо отримати, знаючи лише публічні компоненти.

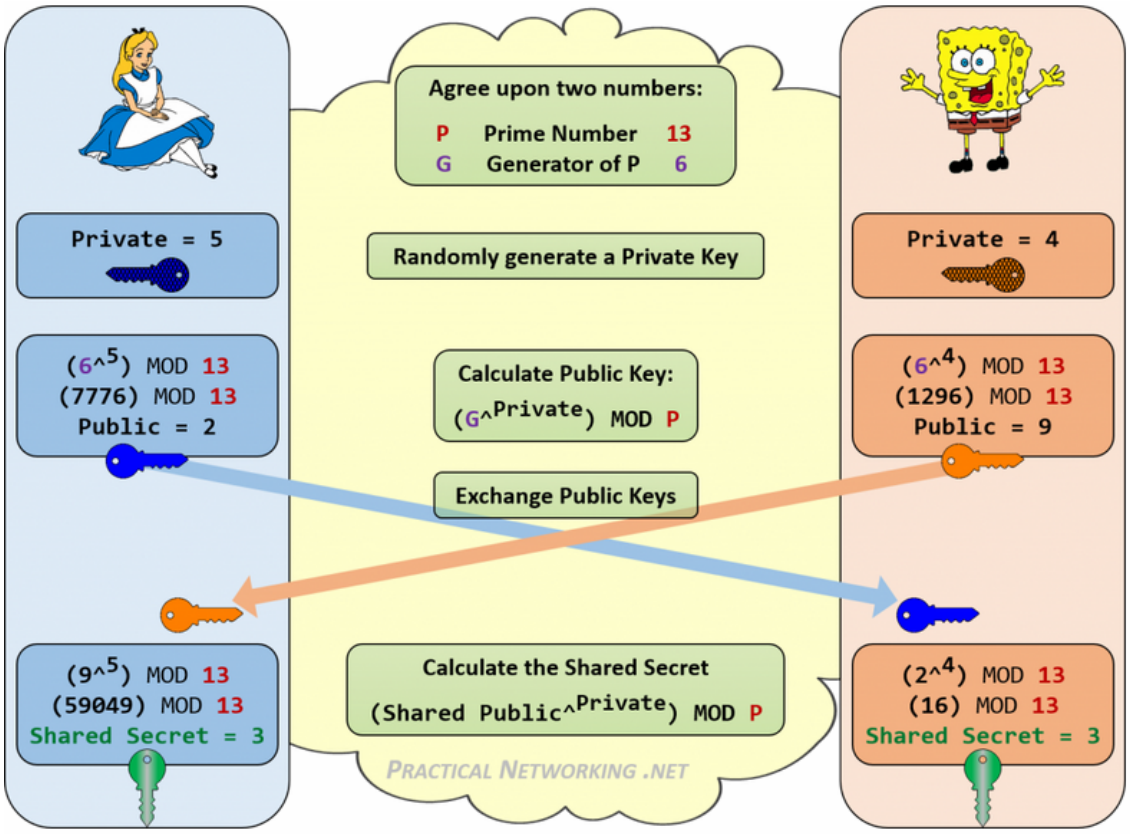

Рис. 2.3. Візуалізація алгоритму обміну ключами Діффі-Геллмана (D-H)

Алгоритм RSA може використовуватись як для шифрування, так і для цифрового підпису. Його стійкість базується на складності задачі факторизації великих чисел.

Це універсальний алгоритм, який використовується як для шифрування (забезпечення конфіденційності), так і для цифрового підпису (забезпечення автентичності).

Рисунок 2.4 ілюструє обидва процеси:

1. Шифрування - відкритий текст шифрується відкритим ключем одержувача і може бути розшифрований лише його закритим ключем.

2. Підпис - хеш повідомлення шифрується закритим ключем відправника (створюючи підпис) і може бути перевірений будь-ким за допомогою його відкритого ключа.

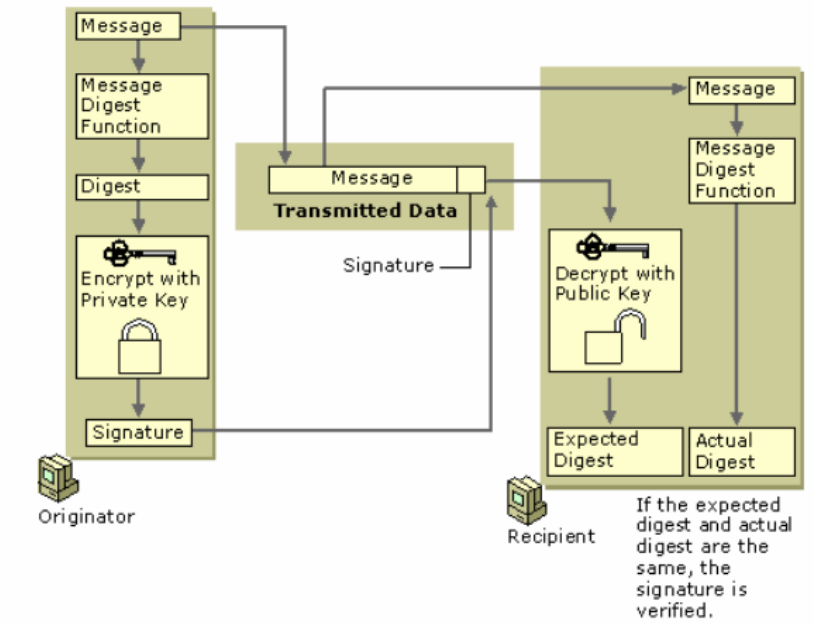

Рис. 2.4. Базовий процес цифрового підпису RSA

DSA (Digital Signature Algorithm) - федеральний стандарт США, призначений виключно для створення цифрових підписів, що гарантують автентичність та цілісність даних.

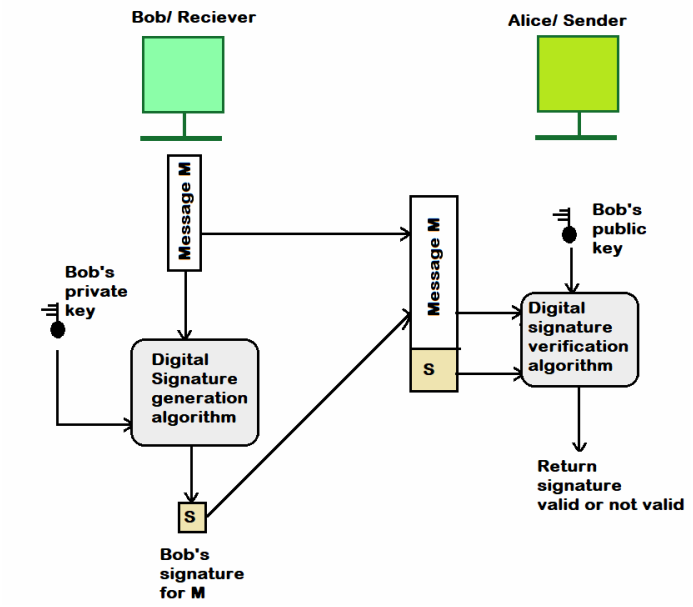

Рис. 2.5. Використання DSA алгоритму

Діаграма на рисунку 2.5 демонструє процес:

- Створення підпису. Від повідомлення береться геш, який потім обробляється за допомогою закритого ключа відправника та випадкового числа для створення підпису.

- Перевірка підпису. Одержувач, використовуючи відкритий ключ відправника, повідомлення та підпис, виконує обчислення. Якщо результат збігається з частиною підпису, то він вважається дійсним.

2.1.3. Становлення та принципи полегшеної криптографії

З поширенням пристроїв з обмеженими обчислювальними ресурсами (RFID-мітки, сенсори, Інтернет речей) виникла потреба в криптографічних алгоритмах, адаптованих до таких середовищ. Традиційні шифри, як-от AES, можуть бути надто ресурсоємними для таких систем.

Полегшена криптографія (Lightweight Cryptography, LWC) — це напрям криптографії, що фокусується на створенні алгоритмів з низьким споживанням ресурсів:

- Мінімальний розмір апаратної реалізації (площа чіпа).

- Низьке енергоспоживання.

- Невеликий обсяг програмного коду та оперативної пам'яті.

Цей підхід є особливо актуальним для обробки даних малого обсягу (small data), де критично важливою є не лише безпека, а й мінімальна затримка. Розробка LWC не завжди означає компроміс із безпекою, а скоріше пошук оптимального балансу між надійністю та ефективністю. Важливість цього напрямку підтверджується стандартизацією в рамках ISO/IEC 29192.

Поточна структура інформаційної безпеки в організаціях значною мірою залежить від оцінки ризиків (ISRA), що дозволяє ідентифікувати активи, загрози та обирати адекватні методи захисту. В умовах постійного зростання обсягів даних та ускладнення загроз спостерігається тенденція до спрощення архітектури безпеки.

Оскільки симетричне шифрування є значно швидшим за асиметричне, поширеною є гібридна модель (наприклад, у протоколі TLS): дані шифруються швидким симетричним ключем, а сам цей ключ шифрується надійним асиметричним алгоритмом (наприклад, RSA) для безпечної передачі. LWC ідеально вписується в цю парадигму, пропонуючи ефективні симетричні рішення для захисту даних як у стані спокою, так і під час передачі, що є особливо корисним для індивідуальних користувачів, малих компаній та систем Інтернету речей.

2.2. Аналіз полегшених симетричних шифрів у контексті сучасних загроз інформаційній безпеці

Фундаментальна різниця у продуктивності між симетричними та асиметричними криптографічними алгоритмами є визначальним фактором при проєктуванні сучасних систем захисту. Перевага у швидкості симетричних шифрів є настільки суттєвою, що вона лежить в основі гібридних криптосистем: дані шифруються за допомогою ефективного симетричного ключа, а сам ключ, у свою чергу, захищається за допомогою асиметричного алгоритму, наприклад, RSA. Оскільки симетричні шифри за своєю природою є обчислювально менш затратними, полегшена симетрична криптографія розвиває цю перевагу, пропонуючи ще більшу ефективність з точки зору часових витрат та простоти імплементації. Відповідно, кінцевою метою є створення спеціалізованих криптографічних систем, оптимізованих для безпечної обробки даних невеликого обсягу.

2.2.1. Концептуальна структура та стандартизація полегшеного шифрування

Пропозиційна структура в контексті даного дослідження визначає концептуальні рамки та план розробки нового криптографічного рішення. Основою для таких рішень слугує симетричне шифрування, яке

використовує ідентичні ключі для шифрування та дешифрування. Ця властивість усуває обчислювальну складність, пов'язану з обробкою пари ключів, що робить його значно швидшим та придатним для широкого кола завдань, зокрема на рівні кінцевого користувача. Технологічний розвиток у криптографії є неперервним процесом, що включає інтенсивне вивчення нових векторів атак, а також інноваційних методів проєктування та реалізації алгоритмів. Одним із передових напрямків у цій галузі є "Полегшена криптографія" (Lightweight Cryptography, LWC).

LWC визначається як клас криптографічних алгоритмів та протоколів, спеціально адаптованих для імплементації в середовищах з обмеженими ресурсами. До таких середовищ належать RFID-мітки, бездротові сенсори, безконтактні смарт-карти, імплантовані медичні пристрої та інші компоненти Інтернету речей (IoT).

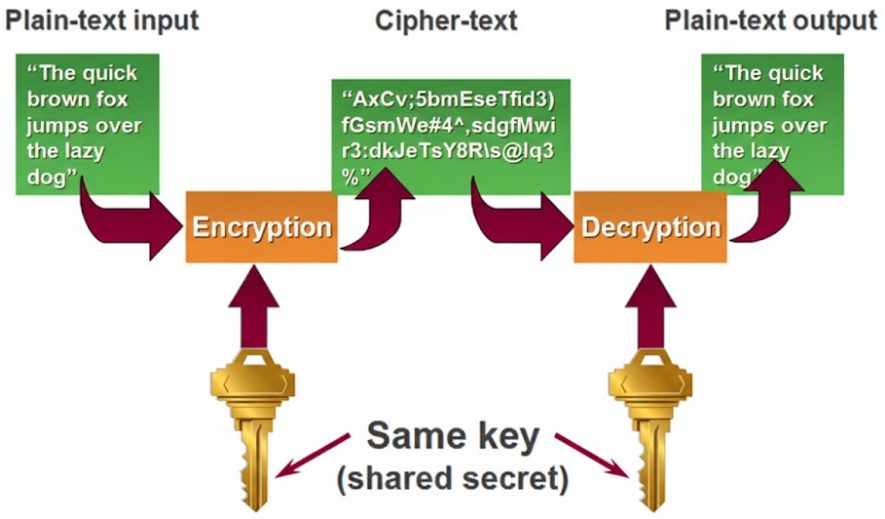

Рис. 2.6. Симетричне шифрування

Для апаратних реалізацій критичними метриками є фізичний розмір кристала та енергоспоживання. Для програмних реалізацій пріоритет надається мінімізації розміру коду та обсягу використовуваної оперативної пам'яті. За цими показниками полегшені криптографічні примітиви значно перевершують традиційні алгоритми, що використовуються в стандартних інтернет-протоколах, таких як IPsec та TLS. Важливо підкреслити, що LWC

забезпечує належний рівень безпеки і не завжди є результатом компромісу між надійністю та ефективністю.

На сучасному етапі інформаційна безпека є критично важливою інфраструктурною складовою як для освітніх установ, так і для комерційних організацій. Освітній сектор висуває високі вимоги до надійності захисту даних, оскільки значна частина фінансових транзакцій, наукових досліджень та освітнього процесу відбувається в цифровому середовищі. За відсутності належних заходів безпеки цінні наукові дані, інтелектуальна власність та персональна інформація можуть бути несанкціоновано модифіковані або викрадені, що може завдати непоправної шкоди як окремим дослідникам, так і репутації установи. Аналогічно, майже всі організації оперують з конфіденційною інформацією, яка потребує захисту від зовнішніх та внутрішніх загроз. Для вирішення цих завдань освітні та організаційні сектори традиційно впроваджують ресурсоємні криптографічні системи. Ці системи вимагають значних інвестицій у вигляді фінансових, часових та людських ресурсів, що створює подвійний ефект: з одного боку, вони забезпечують високий рівень безпеки, з іншого — є дорогими та складними в обслуговуванні.

2.2.2. Оцінка ролі полегшеного шифрування в контексті захисту даних

Настання епохи повсюдних обчислень (ubiquitous computing) характеризується масовим розповсюдженням інтелектуальних пристроїв, які через жорсткі економічні обмеження мають суттєво лімітовані ресурси пам'яті, обчислювальної потужності та заряду батареї. У цьому контексті закон Мура інтерпретується інакше: замість подвоєння продуктивності відбувається зниження вартості за одиницю обчислювальної потужності. Оскільки багато прикладних застосувань, від RFID-міток до біометричних сенсорів, обробляють чутливі дані, попит на ефективні та недорогі криптографічні компоненти стрімко зростає. Саме для таких реалізацій і призначена "полегшена криптографія".

Розробка LWC-рішень неминуче стикається з необхідністю пошуку компромісу між трьома ключовими параметрами: безпекою, вартістю та продуктивністю. Зазвичай, оптимізувати будь-які дві з трьох цілей є відносно простою задачею. Наприклад, безпечна та високопродуктивна апаратна реалізація може бути досягнута за допомогою конвеєрної, стійкої до атак по бічних каналах архітектури, що призводить до високих вимог до площі кристала і, відповідно, до високих витрат. З іншого боку, можна спроєктувати безпечну та недорогу апаратну реалізацію, але з обмеженою продуктивністю. Таким чином, якщо вдається досягти оптимального балансу між цими трьома цілями, полегшене шифрування може відіграти ключову роль у захисті організаційних та освітніх даних, забезпечуючи економію ресурсів та часу завдяки вищій швидкодії та простоті.

Незважаючи на наявність складних інструментів моніторингу та передових методів захисту, зловмисники постійно знаходять способи проникнення навіть у добре захищені системи. Аналіз даних про атаки з реальних великомасштабних середовищ використовується для моделювання поведінки зловмисників та виявлення недоліків в інфраструктурі безпеки. Ці дослідження демонструють, що навіть найсучасніші та найдорожчі архітектури інформаційної безпеки не є невразливими. Це створює для організацій економічну дилему: інвестувати значні кошти у великомасштабні, складні системи, які все одно можуть бути скомпрометовані, чи розглянути альтернативні підходи. У цьому контексті полегшена криптографія постає як економічно вигідніша альтернатива, що вимагає менше ресурсів та фінансових вкладень для досягнення адекватного рівня захисту.

2.2.3. Вплив полегшеної криптографії на тенденції розвитку інформаційної безпеки

Полегшена симетрична криптографія має значний потенціал для впливу на майбутні тенденції та розробки в галузі інформаційної безпеки.

Практика показує, що великомасштабні системи безпеки не завжди забезпечують пропускну здатність та продуктивність, співмірні з інвестованими в них ресурсами. Якщо критерії продуктивності, вартості та ефективності можуть бути оптимально збалансовані в рамках полегшеної криптосистеми, необхідність у розгортанні громіздких архітектур може суттєво зменшитися. Це здатне змінити парадигму сучасної індустрії безпеки, змістивши фокус з централізованих, дорогих систем на більш гнучкі, інтегровані та економічно ефективні рішення. Зростаючий інтерес до цієї теми з боку дослідників та комерційних компаній, а також поява нових перспективних результатів свідчать про те, що полегшене симетричне шифрування формує новий важливий напрямок у розвитку індустрії кібербезпеки.

2.3. Представлення сучасного стану та перспективи розвитку симетричної криптографії даних невеликого об’єму

Аналіз існуючої літератури виявляє ключове обмеження полегшених шифрів: їхня архітектура, оптимізована для даних малого обсягу, унеможливлює їх ефективне застосування у великомасштабних системах, де домінують інші криптографічні підходи. Водночас, саме ця спеціалізація дозволяє значно прискорити процеси безпечної передачі інформації в освітньому та організаційному секторах. У цьому контексті головним науково-технічним викликом є подальше підвищення ефективності алгоритмів та зниження коефіцієнта помилок при їх функціонуванні. Відповідно, рекомендується, щоб майбутні дослідження були зосереджені на цій проблематиці з метою досягнення оптимального балансу між трьома ключовими метриками: вартістю (обчислювальними ресурсами), ефективністю та продуктивністю (пропускною здатністю).

Проведений аналіз стану досліджень у галузі полегшеної симетричної криптографії дозволяє зробити висновок про значний потенціал для

подальшого розвитку. Наявні дослідження з цієї теми є недостатньо вичерпними, кількість розроблених полегшених симетричних шифрів обмежена, а їхні показники ефективності та продуктивності не завжди відповідають сучасним вимогам.

Виходячи з цієї ідентифікованої прогалини, центральною проблемою даного дослідження є відсутність криптографічної системи, що поєднує низькі вимоги до ресурсів з високою ефективністю та мінімальним коефіцієнтом помилок при обробці даних невеликого обсягу. Таким чином, основна мета даної роботи — спроєктувати та розробити таку систему шифрування. Очікується, що результати цього дослідження зроблять значний внесок у розвиток галузі та сприятимуть переходу від використання виключно великомасштабних криптосистем до застосування спеціалізованих симетричних рішень для захисту даних малого обсягу.

2.3.1. Обґрунтування методологічного підходу

Для досягнення поставлених цілей дослідження було обрано специфічну теоретичну основу, яка визначає концептуальну рамку, термінологію та актуальність роботи. Дослідницькі цілі включають:

- Систематичний аналіз характеристик існуючих криптографічних алгоритмів.

- Розробку полегшеного алгоритму.

- Зменшення часової та просторової складності для обробки даних невеликого обсягу.

- Мінімізацію коефіцієнта помилок у процесі шифрування.

Для реалізації цих цілей було обрано дедуктивний підхід. Ця методологія передбачає рух від загальних теоретичних положень, вивчених у літературі, до формулювання конкретної гіпотези. На основі існуючих теорій буде побудована гіпотеза про можливість створення ефективнішого алгоритму, яка згодом буде емпірично перевірена шляхом тестування на наборах даних та аналізу отриманих результатів.

2.3.2. Підсумкові висновки з огляду літератури

Узагальнюючи весь проаналізований матеріал, можна сформулювати три ключові висновки.

По-перше, підтверджено фундаментальну перевагу симетричних алгоритмів у швидкодії порівняно з асиметричними, що робить їх основним інструментом для шифрування безпосередньо даних.

По-друге, виявлено парадокс: незважаючи на те, що полегшена криптографія є оптимальним, надійним та економічно доцільним рішенням для захисту даних у невеликих масштабах, їй не приділяється належної уваги. Організації продовжують інвестувати значні фінансові та людські ресурси у великомасштабні системи безпеки, які, як показує практика, не гарантують абсолютного захисту і залишаються вразливими. Водночас, багато дослідників сходяться на думці, що полегшене симетричне шифрування здатне забезпечити вищий рівень ефективності за значно нижчих витрат.

По-третє, аналіз тенденцій свідчить, що полегшені криптосистеми є не просто нішевою технологією, а новою парадигмою, яка, ймовірно, визначатиме вектор розвитку індустрії безпеки в майбутньому, особливо в контексті розповсюдження Інтернету речей та інших систем з обмеженими ресурсами.

Висновки до розділу

Другий розділ присвячено дослідженню існуючих криптографічних методів і концепції полегшеного шифрування, що є актуальною для систем із обмеженими ресурсами. Проведено класифікацію алгоритмів шифрування за принципами симетричності, асиметричності та стійкості до криптоаналізу. Виявлено ключові переваги lightweight-криптографії, зокрема її гнучкість, мінімальне споживання ресурсів і можливість інтеграції в IoT-пристрої. Розглянуто сучасні стандарти та тенденції розвитку симетричних шифрів,

здатних адаптуватися до постквантових вимог. На основі аналізу зроблено висновок, що полегшені алгоритми можуть бути фундаментом для створення нових постквантових рішень із високою ефективністю та безпекою.

РОЗДІЛ 3. ІМПЛЕМЕНТАЦІЯ ПОСТКВАНТОВИХ МОДЕЛЕЙ ТА МЕТОДІВ КРИПТОАНАЛІЗУ ДАНИХ ДЛЯ РОЗРОБКИ АЛГОРИТМУ ШИФРУВАННЯ

3.1. Методологічне обґрунтування розробки та аналізу криптографічного алгоритму

Дослідження, за своєю суттю, є систематичним та науковим пошуком знань і нової інформації з певної теми. Методологія дослідження являє собою теоретичне обґрунтування вибору та застосування методів, що використовуються для проведення цього пошуку. Вона не пропонує готових рішень, а натомість створює концептуальну рамку, яка дозволяє зрозуміти, який набір методів, принципів та "кращих практик" є найбільш доцільним для вирішення конкретної дослідницької проблеми [20]. Даний розділ присвячено опису та обґрунтуванню методології, що була застосована для досягнення цілей даної роботи.

3.1.1. Дедуктивний підхід як основа дослідження

Дослідницька парадигма визначається як "основоположні припущення та інтелектуальна структура, на якій базуються дослідження та розробки в певній галузі" [25]. В основі даного дослідження лежить дедуктивна парадигма. Дедуктивний підхід передбачає розробку гіпотези на основі існуючої теорії з подальшим проєктуванням дослідницької стратегії для її емпіричної перевірки [27].

Ця парадигма була реалізована в даній роботі через наступну послідовність етапів:

- Теорія. На першому етапі було проведено глибокий аналіз існуючих теорій криптографії, зокрема принципів симетричного шифрування та концепцій полегшеної криптографії.

- Гіпотеза. На основі теоретичного підґрунтя та виявлених прогалин була сформульована гіпотеза про можливість створення нового полегшеного симетричного алгоритму, який демонструватиме вищу ефективність та точність при роботі з даними малого обсягу на різних платформах.

- Спостереження (Експеримент). Для перевірки гіпотези було проведено серію експериментів, що включали імплементацію розробленого алгоритму та його тестування на контрольних наборах даних.

- Підтвердження/Спростування. Отримані експериментальні дані були систематизовані та проаналізовані для підтвердження або спростування висунутої гіпотези, що дозволило сформулювати обґрунтовані висновки.

3.1.2. Використання пошуково-експериментальної моделі

Дизайн дослідження визначає систематичний план та структуру, створену для пошуку відповідей на дослідницькі питання. Враховуючи характер поставлених завдань, дизайн даного дослідження є комплексним і поєднує елементи пошукового та експериментального підходів.

Пошуковий аспект дослідження реалізовано на початкових етапах, коли проблема ще не була чітко визначена. Цей підхід дозволив провести глибокий аналіз наявної літератури, визначити концептуальні рамки, ідентифікувати дослідницькі прогалини та сформулювати основну ідею нового алгоритму. Пошуковий етап покладався на вторинні джерела та якісний аналіз існуючих рішень.

Експериментальний аспект є центральним для даної роботи. Він відповідає визначенню експерименту, в якому дослідник активно маніпулює незалежними змінними (наприклад, архітектурними особливостями алгоритму, довжиною ключа) для спостереження за їхнім впливом на залежні змінні (час шифрування, використання пам'яті, коефіцієнт помилок). Цей підхід дозволив провести контрольоване тестування розробленого алгоритму, порівняти його продуктивність з існуючими аналогами та зробити обґрунтовані висновки щодо його ефективності.

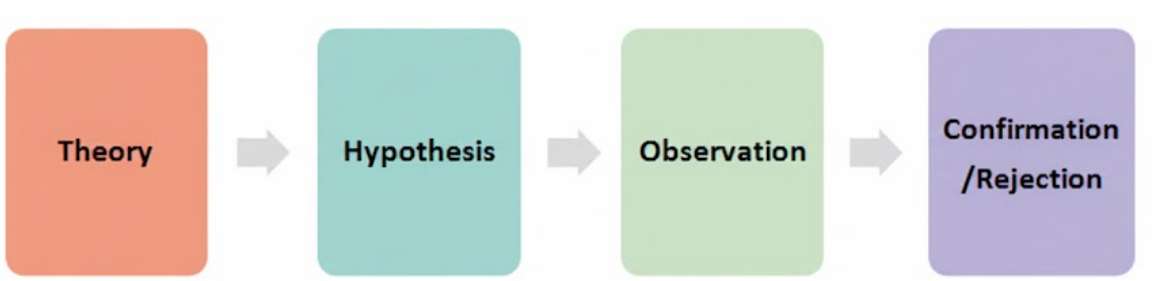

Рис. 3.1. Дедуктивний підхід до дослідження

Дослідницька стратегія являє собою методологічний підхід (general plan), що забезпечує систематичне вивчення та відповіді на поставлені дослідницькі питання.

Ефективна дослідницька стратегія характеризується чітко визначеними цілями, сформульованими дослідницькими питаннями, плануванням ресурсів для збору даних, а також урахуванням різноманітних обмежень (constraints), які можуть впливати на процес дослідження. До таких обмежень відносять ліміти доступу, часові та просторові рамки, фінансові ресурси та етичні норми. Вибір конкретної стратегії дозволяє досліднику обґрунтувати застосовані методики для забезпечення ефективності дослідження та визначити специфічні методи збору даних для емпіричної підтримки аргументів. Реалізація стратегії включає збір довідкової інформації та аналіз даних для досягнення конкретного наукового висновку.

Серед ключових дослідницьких стратегій виділяють: аналіз огляду літератури, кейс-стаді, інтерв'ю, спостереження, експерименти та опитування.

Для досягнення цілей та завдань даного дослідження було використано комбінацію стратегій: опитування та аналізу наукових статей. Стратегія опитування сприяє збору якісних даних та інформації, дозволяючи зафіксувати загальні погляди респондентів. Водночас, аналіз наукових журнальних статей слугує для отримання релевантної інформації та емпіричних даних, безпосередньо пов'язаних із дослідницькими питаннями.

Синтез цих двох стратегій забезпечує збір достовірних та надійних даних, необхідних для успішного виконання дослідницьких завдань.

3.2. Розробка та реалізація алгоритму

Структура дослідницької роботи передбачає поділ на три основні частини, детально розглянуті в наступних підрозділах.

3.2.1. Призначення випадкового ключа

На початковому етапі обирається чотирисимвольний випадковий ключ (random key) (наприклад, abcd,efgh,ijkl тощо). Вибір такої довжини є оптимальним, оскільки занадто короткий ключ знижує криптографічну стійкість (security), роблячи його вразливим до атак грубої сили (brute-force attacks), тоді як надмірно довгий ключ негативно впливає на швидкість виконання алгоритму (algorithmic efficiency). Додатковим обґрунтуванням є кореляція з довжиною більшості PIN-кодів та кодів безпеки. Ключ підлягає зміні (оновленню) на кожній ітерації, що значно підвищує складність злому алгоритму. Кількість ітерацій алгоритму визначається як загальна кількість символів у вихідному реченні, включаючи пробіли та спеціальні символи.

3.2.2. Шифрування ключа та генерація нового ключа

Процедура генерації нового ключа відбувається наступним чином:

- Обчислення та сумація ASCII-значень. Спочатку обчислюються ASCII-значення символів поточного ключа, які потім підсумовуються.

- Модульне перетворення. Обчислюється залишок від ділення отриманої суми на 100 (сума mod 100).

- Бінарне представлення та сегментація. Отриманий результат перетворюється на 8-бітне бінарне число і ділиться на чотири блоки, кожен з яких містить два біти.

- Конкатенація та ітерація. Кожен з чотирьох 2-бітних блоків підлягає конкатенації (послідовному з'єднанню) з трьома бінарними бітами (у діапазоні від 000 до 111) в обох напрямках (прямому та зворотному). На першій ітерації використовується 000, на другій — 001, і так далі. Після восьми ітерацій послідовність повертається до 000. Таким чином, кожен блок набуває довжини у вісім бітів.

- Генерація нового ключа. Десяткові значення (decimal values) цих нових 8-бітних блоків визначаються як ASCII-значення символів нового ключа.

Ця процедура забезпечує ітеративну генерацію (iterative generation) нового ключа з попереднього і продовжується до завершення останньої ітерації алгоритму.

3.2.3. Процедура шифрування та дешифрування даних

Процес шифрування даних реалізується ітеративно, як показано на рисунку 3.2.

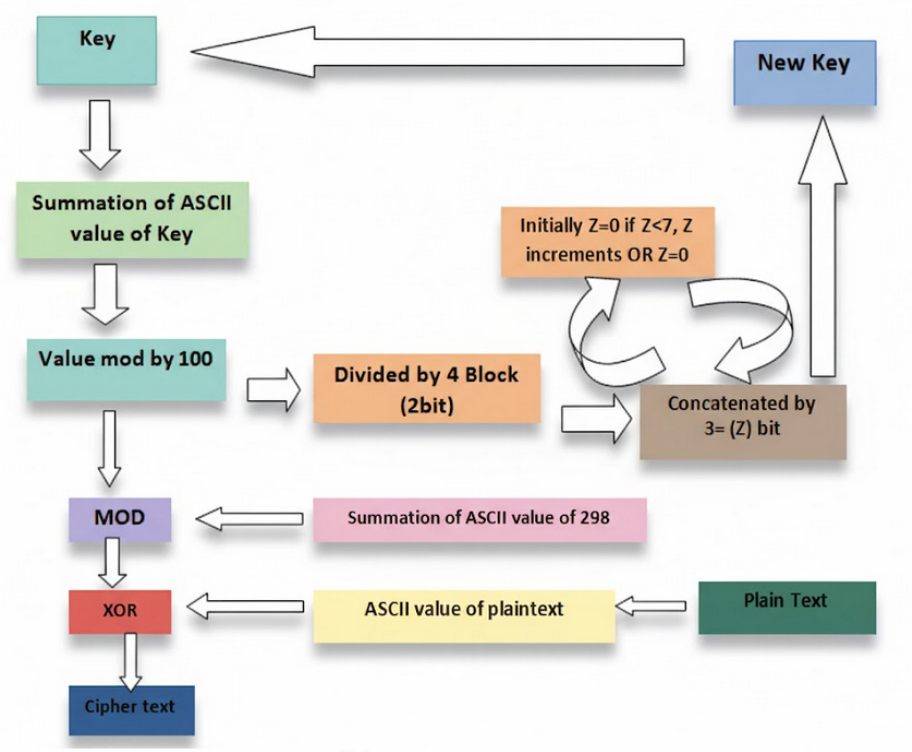

Рис. 3.2. Блок-схема шифрування

1. Початкове перетворення.

Символу відкритого тексту (plaintext) ставиться у відповідність його ASCII-значення, яке далі конвертується у двійкове представлення.

2. Генерація маски XOR.

Використовується фіксоване випадкове число 298. Обчислюється залишок від ділення (modulo) числа 298 на проміжне значення mi (отримане як ∑ASCII(ключ) mod 100 на поточній ітерації). Це проміжне значення mi (або MOD на схемі) слугує для генерації маски.

3. Криптографічна операція.

Результат модульного перетворення (маска) конвертується у двійковий формат і виконується операція логічного виключного АБО (XOR) з двійковим представленням ASCII-значення відкритого тексту.

4. Фінальне перетворення.

Результат операції XOR повертається до десяткового формату. Отримане десяткове значення є ASCII-значенням шифротексту, якому відповідає кінцевий символ шифротексту (ciphertext).

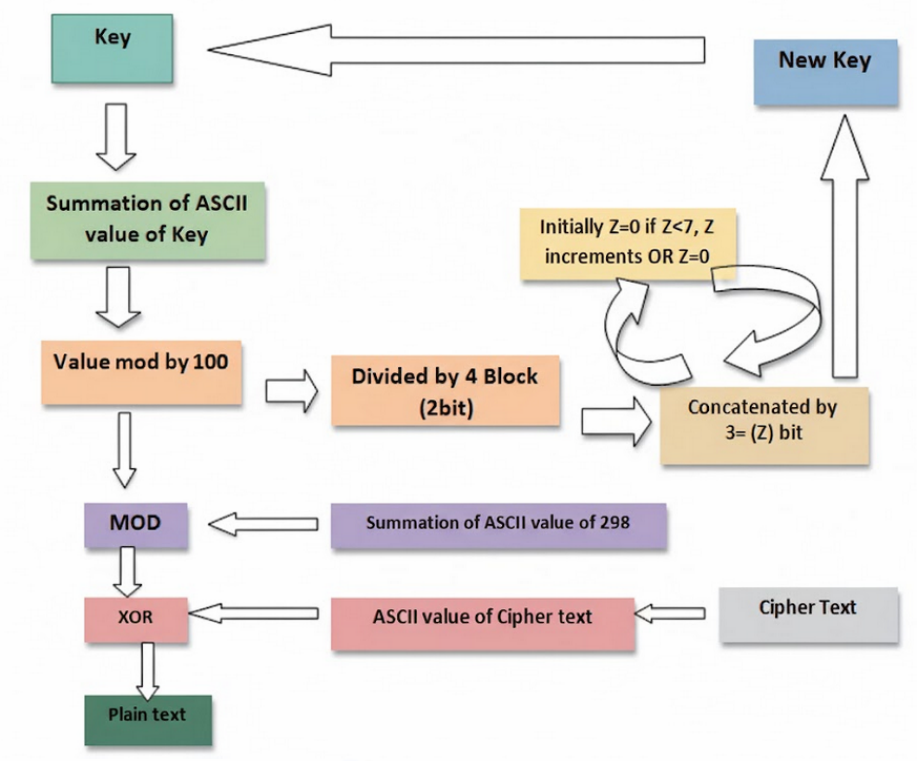

Рис. 3.3. Блок-схема дешифрування

Дешифрування є оберненою (інверсною) процедурою і повністю повторює кроки шифрування завдяки властивості інволютивності (self-inverting property) операції XOR ($A \oplus B \oplus B = A$). Процес представлено на рисунку 3.3.

1. Початкове перетворення.

Символу шифротексту ставиться у відповідність його ASCII-значення, конвертоване у двійкове представлення.

2. Генерація маски XOR.

Використовується те саме фіксоване випадкове число 298. Обчислюється залишок від ділення числа 298 на проміжне значення ni (еквівалентне mi з етапу шифрування).

3. Криптографічна операція.

Результат модульного перетворення (маска) перетворюється у двійковий формат, і виконується операція XOR з двійковим значенням ASCII-значення шифротексту.

4. Фінальне перетворення.

Результат операції XOR є ASCII-значенням відкритого тексту, якому відповідає кінцевий дешифрований символ.

3.2.4. Представлення прикладу застосування

Для наочної демонстрації функціонування алгоритму наведено приклад шифрування та дешифрування речення "MY MIST". Як алгоритм із симетричним ключем (symmetric key algorithm), він використовує однаковий ключ для обох процесів. Вибрано початковий випадковий ключ: "abcd".

Процес є посимвольним та ітераційним, що означає, що генерація нового ключа відбувається перед обробкою кожного наступного символу відкритого тексту.

Шифрування (Ітерації 1-7)

Ітерація 1 (Символ 'M')

- Генерація ключа:

- Початковий ключ: abcd.

- ∑ASCII(abcd)=394.

- Проміжне значення m1=394 mod 100=94.

- Процедура конкатенації з бітами '000' генерує новий ключ 8,8,24,16.

- Шифрування:

- ASCII('M') =77.

- Маска XOR: 298 mod m1=298 mod 94=16.

- 77 ⊕ 16=93.

- ASCII шифротексту 93⟹ ']'.

Ітерація 2 (Символ 'Y')

- Генерація ключа:

- Попередній ключ 8,8,24,16⟹∑ASCII=56.

- Проміжне значення m2=56 mod 100=56.

- Конкатенація з бітами '001' генерує новий ключ 33,57,49,33.

- Шифрування:

- ASCII('Y') =89.

- Маска XOR: 298 mod m2=298 mod 56=18.

- 89 ⊕ 18=75.

- ASCII шифротексту 75⟹ 'K'.

Процес продовжується до сьомої ітерації, генеруючи проміжні ключі та шифротекст.

Ітерація 7 (Символ 'T')

- Генерація ключа:

- Проміжне значення m7=8.

- Шифрування:

- ASCII('T') =84.

- Маска XOR: 298 mod m7=298 mod 8=2.

- 84 ⊕ 2=86.

- ASCII шифротексту 86⟹ 'V'.

Фінальний шифротекст:]K*o[iV [369].

Дешифрування (Ітерації 1-7)

Процес дешифрування дзеркально повторює процедуру шифрування, використовуючи ті ж самі ітераційно згенеровані ключі та маски XOR, що є фундаментальною вимогою для симетричних криптосистем.

Ітерація 1 (Шифротекст ']')

- Генерація ключа:
    - ∑ASCII(abcd)=394.
    - Проміжне значення n1=94.
- Дешифрування:
    - ASCII(']') =93.
    - Маска XOR: 298 mod n1=16.
    - 93 ⊕ 16=77.
    - ASCII відкритого тексту 77⟹ 'M'.

Ітерація 2 (Шифротекст 'K')

- Генерація ключа:
    - Проміжне значення n2=56.
- Дешифрування:
    - ASCII('K') =75.
    - Маска XOR: 298 mod n2=18.
    - 75 ⊕ 18=89.
    - ASCII відкритого тексту 89⟹ 'Y'.

Процес продовжується до 7 ітерації.

Ітерація 7 (Шифротекст 'V')

- Генерація ключа:
    - Проміжне значення n7=8.
- Дешифрування:
    - ASCII('V') =86.
    - Маска XOR: 298 mod n7=2.
    - 86 ⊕ 2=84.

- ASCII відкритого тексту 84⟹ 'T'.

Фінальний відкритий текст: MY MIST.

3.3. Формалізований опис пропонованого криптографічного алгоритму

Запропонований алгоритм є симетричним потоковим шифром, розробленим для обробки даних малого обсягу. Його функціонування базується на двох ключових процесах: генерації псевдовипадкового ключового потоку та ітеративній еволюції внутрішнього стану ключа. Нижче наведено формалізований опис процедур шифрування та дешифрування.

3.3.1. Алгоритм 1. Процедура шифрування

Процедура шифрування, представлена як Алгоритм 1, виконує перетворення відкритого тексту (P) на шифротекст (C) шляхом посимвольної обробки.

Опис процесу:

1. Ініціалізація. Внутрішній стан S ініціалізується сумою ASCII-значень символів 4-байтного ключа. Лічильник ітерацій i та лічильник еволюції ключа conV встановлюються на початкові значення.

2. Ітераційний процес. Для кожного символу P[i] вхідного потоку відкритого тексту виконується цикл операцій.

3. Генерація ключового потоку: На кожній ітерації генерується байт ключового потоку Y шляхом обчислення Y ← 298 mod (S mod 100).

4. Шифрування. Символ шифротексту C[i] отримується шляхом виконання побітової операції XOR між символом відкритого тексту P[i] та згенерованим байтом ключового потоку Y.

5. Еволюція ключа. Після кожної ітерації внутрішній стан S оновлюється. Це відбувається шляхом перетворення S mod 100 у 8-бітний двійковий рядок, розбиття його на чотири 2-бітні блоки, конкатенації

кожного блоку з 3-бітним значенням лічильника conV, перетворення отриманих 8-бітних блоків назад у десяткові числа та їх підсумовування для отримання нового значення S. Лічильник conV інкрементується циклічно від 0 до 7.

```
АЛГОРИТМ 1: Шифрування відкритого тексту (P) в шифротекст (C)
-----------------------------------------------------------------
ВХІДНІ ДАНІ:
    P: Масив символів відкритого тексту.
    Key: 4-символьний ключ шифрування.

ВИХІДНІ ДАНІ:
    C: Масив символів шифротексту.

ПЕРЕДУМОВА: Довжина(Key) = 4 та Довжина(P) > 0.

ПРОЦЕДУРА:
1.  S ← ASCII(Key[1]) + ASCII(Key[2]) + ASCII(Key[3]) + ASCII(Key[4])
2.  conV ← 0
3.  ДЛЯ КОЖНОГО i ВІД 1 ДО Довжина(P):
4.      m ← S mod 100
5.      Y ← 298 mod m
6.      C[i] ← P[i] ⊕ Y  // Побітова операція XOR
7.
8.      // --- Еволюція ключа для наступної ітерації ---
9.      Z ← десяткове_в_8-бітне_двійкове(m)
10.     B1 ← Z[1..2]; B2 ← Z[3..4]; B3 ← Z[5..6]; B4 ← Z[7..8]
11.     conB ← десяткове_в_3-бітне_двійкове(conV)
12.
13.     B1_new ← конкатенація(conB, B1, conB)
14.     B2_new ← конкатенація(conB, B2, conB)
15.     B3_new ← конкатенація(conB, B3, conB)
16.     B4_new ← конкатенація(conB, B4, conB)
17.
18.     S ← двійкове_в_десяткове(B1_new) + ... + двійкове_в_десяткове(B4_new)
19.
20.     ЯКЩО conV ≥ 7 ТОДІ
21.         conV ← 0
22.     ІНАКШЕ
23.         conV ← conV + 1
24.     КІНЕЦЬ ЯКЩО
25. КІНЕЦЬ ДЛЯ
```

Виконаємо реалізацію алгоритму шифрування, представленого вище, мовою програмування Python.

Лістинг 3.1. Реалізація алгоритму шифрування

```
def lwe_encrypt(plaintext: bytes, key: list[int]) -> bytes:
    """
    Реалізація алгоритму шифрування

    Args:
        plaintext (bytes): Вхідні дані для шифрування у вигляді байтів.
        key (list[int]): Ключ шифрування, що складається з 4-х цілих чисел.

    Returns:
        bytes: Зашифровані дані у вигляді байтів.

    Raises:
        ValueError: Якщо ключ не містить 4 елементи або вхідний текст порожній.
    """
    # Require: L.Key == 4 ∨ L.P ≠ 0
    # Перевірка вхідних умов
    if len(key) != 4:
        raise ValueError("Ключ повинен складатися з 4-х цілих чисел.")
    if not plaintext:
        raise ValueError("Вхідний текст не може бути порожнім.")

    # S ← Key(1) + Key(2) + Key(3) + Key(4)
    # Ініціалізація стану S сумою елементів ключа
    s_state = sum(key)


    # conV ← 0
    # Ініціалізація лічильника conV
    conv_counter = 0

    ciphertext = bytearray()

    # while i→L.P do
    # Цикл по кожному байту вхідного тексту
    for p_byte in plaintext:
        # S ← S mod 100
        s_state = s_state % 100

        # Запобігання діленню на нуль, якщо s_state стає 0.
        # У псевдокоді цей випадок не оброблено.
        if s_state == 0:
            s_state = 100 # Використовуємо 100 як ненульове значення за замовчуванням

        # Y ← 298 mod S
        y_val = 298 % s_state

        # F ← Y XOR P(i)
        # Операція шифрування (XOR)
        f_val = y_val ^ p_byte
```

```
            # C(i) ← F
            # Додавання зашифрованого байта до результату
            ciphertext.append(f_val)

            # Z ← Bin(S, 8)
            # Перетворення стану S у 8-бітний двійковий рядок
            z_bin = format(s_state, '08b')

            # Розбиття Z на 2-бітні частини
            b1 = z_bin[0:2]
            b2 = z_bin[2:4]
            b3 = z_bin[4:6]
            b4 = z_bin[6:8]

            # conB ← Bin(conV, 3)
            # Перетворення лічильника conV у 3-бітний двійковий рядок
            conv_bin = format(conv_counter, '03b')

            # Перезапис змінних B1-B4.
            # Примітка: згідно з псевдокодом, ці нові значення
            # не використовуються для оновлення стану s_state.
            b1 = conv_bin + b1 + conv_bin
            b2 = conv_bin + b2 + conv_bin
            b3 = conv_bin + b3 + conv_bin
            b4 = conv_bin + b4 + conv_bin

            # Оновлення лічильника conV (циклічно від 0 до 7)
            conv_counter += 1
            if conv_counter > 7:
                conv_counter = 0

    return bytes(ciphertext)

# --- Приклад використання ---
if __name__ == "__main__":
    # Вхідні дані
    my_key = [65, 10, 201, 88] # Ключ з 4-х чисел
    my_plaintext = "This is a test message for LWE algorithm.".encode('utf-8')

    print(f"Відкритий текст: {my_plaintext.decode('utf-8')}")
    print(f"Ключ: {my_key}")
    print("-" * 30)

    # Шифрування
    try:
        encrypted_data = lwe_encrypt(my_plaintext, my_key)
        print(f"Зашифровані дані (байти): {encrypted_data}")
        print(f"Зашифровані дані (hex):   {encrypted_data.hex()}")
    except ValueError as e:
        print(f"Помилка: {e}")
```

Представлений код реалізує функцію lwe_encrypt, яка приймає байти відкритого тексту (plaintext) та список з 4-х цілих чисел (key) і повертає зашифровані байти.

Алгоритм є потоковим шифром. Для кожного байта вхідного тексту він генерує псевдовипадкове число Y на основі внутрішнього стану S. Потім це число Y поєднується з байтом тексту за допомогою операції XOR (^), щоб отримати зашифрований байт. Після шифрування кожного байта внутрішній стан S та лічильник conV оновлюються.

3.3.2. Алгоритм 2. Процедура дешифрування

Процедура дешифрування, представлена як алгоритм 2, є симетричною до процедури шифрування та виконує перетворення шифротексту (C) назад у відкритий текст (P).

Процес дешифрування є ідентичним процесу шифрування. Використовуючи той самий початковий ключ Key, алгоритм генерує точно таку саму послідовність байтів ключового потоку Y. Завдяки властивості операції XOR, де (A ⊕ B) ⊕ B = A, повторне застосування того ж байта Y до символу шифротексту C[i] відновлює вихідний символ відкритого тексту P[i].

```
АЛГОРИТМ 2: Дешифрування шифротексту (C) у відкритий текст (P)
-----------------------------------------------------------------
ВХІДНІ ДАНІ:
    C: Масив символів шифротексту.
    Key: 4-символьний ключ шифрування (той самий, що й для шифрування).

ВИХІДНІ ДАНІ:
    P: Масив символів відновленого відкритого тексту.

ПЕРЕДУМОВА: Довжина(Key) = 4 та Довжина(C) > 0.

ПРОЦЕДУРА:
1.  S ← ASCII(Key[1]) + ASCII(Key[2]) + ASCII(Key[3]) + ASCII(Key[4])
2.  conV ← 0
3.  ДЛЯ КОЖНОГО i ВІД 1 ДО Довжина(C):
4.      m ← S mod 100
5.      Y ← 298 mod m
6.      P[i] ← C[i] ⊕ Y  // Побітова операція XOR
```

```
7.
8.      // --- Еволюція ключа (ідентична Алгоритму 1) ---
9.      Z ← десяткове_в_8-бітне_двійкове(m)
10.     B1 ← Z[1..2]; B2 ← Z[3..4]; B3 ← Z[5..6]; B4 ← Z[7..8]
11.     conB ← десяткове_в_3-бітне_двійкове(conV)
12.
13.     B1_new ← конкатенація(conB, B1, conB)
14.     B2_new ← конкатенація(conB, B2, conB)
15.     B3_new ← конкатенація(conB, B3, conB)
16.     B4_new ← конкатенація(conB, B4, conB)
17.
18.     S ← двійкове_в_десяткове(B1_new) + ... + двійкове_в_десяткове(B4_new)
19.
20.     ЯКЩО conV ≥ 7 ТОДІ
21.         conV ← 0
22.     ІНАКШЕ
23.         conV ← conV + 1
24.     КІНЕЦЬ ЯКЩО
25. КІНЕЦЬ ДЛЯ
```

Алгоритм дешифрування є дзеркальним відображенням алгоритму шифрування. Він використовує той самий ключ для генерації ідентичної послідовності псевдовипадкових чисел Y. Завдяки властивості операції XOR (де (A XOR B) XOR B = A), повторне застосування того ж ключового потоку до шифротексту відновлює вихідний відкритий текст.

Лістинг 3.2. Реалізація алгоритму дешифрування

```python
def lwe_decrypt(ciphertext: bytes, key: list[int]) -> bytes:
    """
    Реалізація алгоритму дешифрування

    Args:
        ciphertext (bytes): Вхідні зашифровані дані у вигляді байтів.
        key (list[int]): Ключ шифрування, що складається з 4-х цілих чисел.

    Returns:
        bytes: Розшифровані дані (відкритий текст) у вигляді байтів.

    Raises:
        ValueError: Якщо ключ не містить 4 елементи або шифротекст порожній.
    """
    # Require: L.Key == 4 ∨ L.C ≠ 0
    # Перевірка вхідних умов
    if len(key) != 4:
        raise ValueError("Ключ повинен складатися з 4-х цілих чисел.")
    if not ciphertext:
        raise ValueError("Шифротекст не може бути порожнім.")
```

```
    # S ← Key(1) + Key(2) + Key(3) + Key(4)
    # Ініціалізація стану S сумою елементів ключа
    s_state = sum(key)

    # conV ← 0
    # Ініціалізація лічильника conV
    conv_counter = 0

    plaintext = bytearray()

    # while i→L.C do
    # Цикл по кожному байту шифротексту
    for c_byte in ciphertext:
        # Генерація ключового потоку (ідентична до шифрування)
        s_state = s_state % 100

        if s_state == 0:
            s_state = 100

        y_val = 298 % s_state

        # F ← Y XOR C(i)
        # Операція дешифрування (XOR)
        f_val = y_val ^ c_byte

        # P(i) ← F
        # Додавання розшифрованого байта до результату
        plaintext.append(f_val)

        # Оновлення стану (ідентичне до шифрування)
        z_bin = format(s_state, '08b')
        b1, b2, b3, b4 = z_bin[0:2], z_bin[2:4], z_bin[4:6], z_bin[6:8]
        conv_bin = format(conv_counter, '03b')
        b1, b2, b3, b4 = (conv_bin + b + conv_bin for b in (b1, b2, b3, b4))

        conv_counter += 1
        if conv_counter > 7:
            conv_counter = 0

    return bytes(plaintext)
```

Оскільки логіка генерації ключового потоку для шифрування та дешифрування абсолютно однакова, її можна винести в окремий генератор. Це робить код чистішим та уникає дублювання, як показано в лістингу 3.3.

Лістинг 3.3. Реалізація повного циклу (шифрування + дешифрування)

```
def _lwe_keystream_generator(key: list[int], data_length: int):
    """Генератор, що створює ключовий потік Y для алгоритму LWE."""
    if len(key) != 4:
        raise ValueError("Ключ повинен складатися з 4-х цілих чисел.")

    s_state = sum(key)
    conv_counter = 0

    for _ in range(data_length):
        s_state = s_state % 100
        if s_state == 0:
            s_state = 100

        yield 298 % s_state # Повертаємо наступне значення ключового потоку Y

        # Оновлення стану (логіка залишається, хоч і не впливає на Y)
        z_bin = format(s_state, '08b')
        b1, b2, b3, b4 = z_bin[0:2], z_bin[2:4], z_bin[4:6], z_bin[6:8]
        conv_bin = format(conv_counter, '03b')
        b1, b2, b3, b4 = (conv_bin + b + conv_bin for b in (b1, b2, b3, b4))

        conv_counter += 1
        if conv_counter > 7:
            conv_counter = 0

def lwe_process(data: bytes, key: list[int]) -> bytes:
    """
    Уніфікована функція для шифрування та дешифрування.
    Працює однаково для обох операцій завдяки властивостям XOR.
    """
    if not data:
        raise ValueError("Вхідні дані не можуть бути порожніми.")

    processed_data = bytearray()
    keystream = _lwe_keystream_generator(key, len(data))

    for data_byte, key_byte in zip(data, keystream):
        processed_data.append(data_byte ^ key_byte)

    return bytes(processed_data)

# --- Приклад повного циклу ---
if __name__ == "__main__":
    # Вхідні дані
    my_key = [65, 10, 201, 88] # Той самий ключ
    my_plaintext = "This is a test message for LWE algorithm.".encode('utf-8')

    print(f"Відкритий текст:   {my_plaintext.decode('utf-8')}")
    print(f"Ключ:              {my_key}")
    print("-" * 40)
```

```
# 1. Шифрування
encrypted_data = lwe_process(my_plaintext, my_key)
print(f"Зашифровані дані (hex): {encrypted_data.hex()}")

# 2. Дешифрування
decrypted_data = lwe_process(encrypted_data, my_key)
print(f"Розшифровані дані: {decrypted_data.decode('utf-8')}")
print("-" * 40)

# 3. Перевірка
assert my_plaintext == decrypted_data
print("✅ Перевірка успішна: вихідний текст і розшифрований збігаються.")
```

3.3.3. Процедура формування та характеристики тестового набору даних

Для емпіричного оцінювання розробленого алгоритму було сформовано тестовий набір даних, класифікований за сімома основними категоріями на основі складу символів:

- Тільки символи (Characters only).
- Тільки числа (Numerals only).
- Тільки спеціальні символи (Special symbols only).
- Символ-спеціальний символ (Character-Special symbol).
- Символ-число (Character-Numeral).
- Число-спеціальний символ (Numeral-Special symbol).
- Символ-число-спеціальний символ (Character-Numeral-Special symbol).

Для кожної із зазначених категорій було відібрано 300 тестових файлів, що в сукупності становить 2100 тестових одиниць (300×7). Розмір файлів обмежувався діапазоном від 1 до 512 КБ, оскільки алгоритм призначений для опрацювання даних невеликого обсягу. Типи файлів включали формати .txt та .doc.

Для відбору даних використовувався метод випадкової вибірки (random sampling). Цей підхід є доцільним у випадках відбору з існуючої сукупності (популяції), зокрема при аналізі історичних або пакетних даних.

Ключова перевага випадкової вибірки полягає в тому, що кожна одиниця в генеральній сукупності має рівну ймовірність бути включеною у вибірку. Це забезпечує захист від упередженості в процесі відбору і сприяє отриманню репрезентативної вибірки.

Як правило, випадкова вибірка реалізується шляхом присвоєння числового ідентифікатора кожній одиниці сукупності з подальшою генерацією необхідного списку номерів за допомогою таблиці випадкових чисел або спеціалізованого статистичного програмного забезпечення (наприклад, Minitab). За відсутності попередньої інформації про фактори стратифікації сукупності, випадкова вибірка є рекомендованим початковим кроком для отримання обґрунтованих зразків.

Емпіричні тести файлів проводилися на різних операційних системах, включаючи Windows та Linux. Таким чином, результати дослідження та аналізу демонструють незалежність алгоритму від платформи.

3.4. Представлення результатів експериментальної оцінки алгоритму шифрування та дешифрування з симетричним ключем

У цьому розділі представлено результати експериментальної оцінки, аналіз та обговорення характеристик запропонованого та розробленого алгоритму шифрування та дешифрування з симетричним ключем.

Варто зазначити, що сфера застосування алгоритму є вузькоспеціалізованою: він не розглядає та не порівнюється з характеристиками складних або асиметричних криптографічних алгоритмів. Його функціональність обмежена роботою виключно з символами англійської мови та орієнтована на невеликі обсяги текстових даних: файли .txt розміром 1 КБ до 512 КБ та файли .doc розміром 1 КБ до 66.5 КБ.

Внаслідок такої оптимізації, процес шифрування та дешифрування характеризується мінімальними часовими витратами, низькою обчислювальною складністю та незначним відсотком помилок. Алгоритм є

легким у впровадженні і забезпечує покращену безпеку під час передачі даних. Він підтримує текстові файли, що містять окремі або комбіновані символи, числа та спеціальні символи.

Для моделювання та симуляції алгоритму було використано програмне середовище Matlab.

З метою аналізу ефективності було здійснено шифрування та дешифрування файлів у зазначених діапазонах розмірів. Вимірювався час виконання (execution time) для кожної операції. Тестовий набір даних був класифікований на сім категорій залежно від складу символів.

Сім категорій тестових файлів включали:

- Дані, що містять лише звичайні символи.
- Дані, що містять лише звичайні числа.
- Дані, що містять лише звичайні спеціальні символи.
- Комбінації звичайних символів та чисел.
- Комбінації звичайних спеціальних символів та чисел.
- Комбінації звичайних символів та спеціальних символів.
- Комбінації звичайних символів, спеціальних символів та чисел.

3.4.1. Проведення експерименту використання тільки звичайних символьних даних

Ця категорія включала текстові файли та файли типу .doc/.docx (з відповідними еквівалентами .odt,fodt,.uot) з даними, що містять виключно звичайні символи. Діапазони розмірів становили 1 КБ – 512 КБ для текстових файлів та 1 КБ – 66.5 КБ для файлів .doc.

Для аналізу результатів цієї категорії було розраховано співвідношення між типом файлу та середнім часом виконання.

Таблиця 3.1 демонструє результати симуляції алгоритму на різних платформах для тестових файлів фіксованого розміру (150 КБ для .txt та 20 КБ для .doc).

Таблиця 3.1.

Розрахунок співвідношення типу файлу до середнього часу виконання для звичайних символьних даних

Категорія файлу	Тип файлу	Використана платформа	Довжина ключа	Середня кількість ітерацій	Час виконання за ітерацію (ітер./сек)	Загальний час виконання (сек)
Категорія (150 КБ для txt, 20 КБ для doc)	.txt	Windows	4	164,41	1232	133.393
	.doc/.docx	Windows	4	164,41	1215	135.252
	.txt	Linux	4	164,41	1213	135.459
	.odt/fodt/.uot	Linux	4	164,41	1213	135.323
	.txt	Mac OS	4	164,41	1204	136.489
	.page	Mac OS	4	164,41	1206	134.256

Параметри, представлені в таблиці, мають наступне визначення:

- Категорія файлу: Позначає одну з семи визначених груп тестових даних.

Використана платформа: Операційна система, на якій проводилася симуляція.

Довжина ключа: Стандартно фіксована на 4 символи, хоча користувач може обирати будь-яку комбінацію чотирисимвольного ключа.

Середня кількість ітерацій: Кількість ітерацій алгоритму, яка залежить від загальної кількості символьних даних у тестовому файлі.

Час виконання за ітерацію: Середнє число виконаних ітерацій за одну секунду.

Загальний час виконання: Сукупний час, необхідний для завершення середньої кількості ітерацій для даного файлу.

3.4.2. Проведення експерименту використання лише звичайних числових даних

Наступна категорія включала тестові файли, які містили виключно числові дані. Діапазони розмірів файлів відповідали загальним умовам тестування: 1 КБ – 512 КБ для текстових файлів (.txt) та 1 КБ – 66.5 КБ для файлів .doc. Для оцінки продуктивності було проаналізовано співвідношення між типом файлу та середнім часом виконання.

Таблиця 3.2.

Розрахунок співвідношення типу файлу до середнього часу виконання при використанні лише звичайних числових даних

Категорія файлу	Тип файлу	Використана платформа	Довжина ключа	Середня кількість ітерацій	Час виконання за ітерацію (ітер./сек)	Загальний час виконання (сек)
Категорія (150 КБ для txt, 20 КБ для doc)	.txt	Windows	4	115,393	902	127.876386
	.doc/.docx	Windows	4	115,393	902	127.1125
	.txt	Linux	4	115,393	887	129.97856
	.odt/fodt/.uot	Linux	4	115,393	886	128.555
	.txt	Mac OS	4	115,393	891	129.48796
	.page	Mac OS	4	115,393	890	128.1235

Таблиця 3.2 містить результати симуляції полегшеного алгоритму криптографії з симетричним ключем на платформах Windows, Linux та Mac OS для фіксованих тестових розмірів 150 КБ (.txt) та 20 КБ (.doc).

Середня кількість ітерацій для алгоритму у цій категорії визначається загальною кількістю числових даних у файлі. Час виконання за ітерацію являє собою загальну кількість ітерацій, виконаних за одну секунду, а Загальний час виконання — сукупний час, необхідний для завершення

середньої кількості ітерацій. Довжина ключа, хоча і є фіксованою (4 символи), може бути представлена користувачем будь-якою чотирисимвольною комбінацією.

3.4.3. Проведення експерименту використання звичайних спеціальних символів

Тестовий набір у цій категорії складався з файлів, що містили виключно спеціальні символи. Розміри файлів відповідали встановленим діапазонам: 1 КБ – 512 КБ для .txt та 1 КБ – 66.5 КБ для .doc. Аналіз продуктивності ґрунтувався на співвідношенні типу файлу до середнього часу виконання.

Таблиця 3.3.

Розрахунок співвідношення типу файлу до середнього часу виконання при використанні звичайних спеціальних символів

Категорія файлу	Тип файлу	Використана платформа	Довжина ключа	Середня кількість ітерацій	Час виконання за ітерацію (ітер./сек)	Загальний час виконання (сек)
Категорія (150 КБ для txt, 20 КБ для doc)	.txt	Windows	4	115,393	902	127.000
	.doc/.docx	Windows	4	115,393	902	127.19755
	.txt	Linux	4	115,393	887	129.125
	.odt/fodt/.uot	Linux	4	115,393	886	128.578
	.txt	Mac OS	4	115,393	891	129.900
	.page	Mac OS	4	115,393	890	128.8934

Таблиця 3.3 представляє результати симуляції пропонованого алгоритму на різних операційних системах (Windows, Linux, Mac OS) для

тестових файлів категорії з фіксованими розмірами 150 КБ (.txt) та 20 КБ (.doc).

3.4.4. Проведення експерименту використання комбінації символьних та числових даних

Тестовий набір даних складався з файлів, що містили комбінацію звичайних символів та числових даних. Розміри файлів відповідали встановленим діапазонам: 1 КБ – 512 КБ для текстових файлів та 1 КБ – 66.5 КБ для файлів .doc. Оцінювання продуктивності алгоритму проводилося шляхом аналізу співвідношення між типом файлу та середнім часом виконання.

Таблиця 3.4.

Розрахунок співвідношення типу файлу до середнього часу виконання при використанні використання комбінації символьних та числових даних

Категорія файлу	Тип файлу	Використана платформа	Довжина ключа	Середня кількість ітерацій	Час виконання за ітерацію (ітер./сек)	Загальний час виконання (сек)
Категорія (150 КБ для txt, 20 КБ для doc)	.txt	Windows	4	116,504	945	127.000
	.doc/.docx	Windows	4	116,504	941	131.955
	.txt	Linux	4	116,504	911	130.000
	.odt/fodt/.uot	Linux	4	116,504	910	128.157
	.txt	Mac OS	4	116,504	909	129.900
	.page	Mac OS	4	116,504	909	128.9634

У таблиці 3.4 наведено результати симуляції алгоритму на платформах Windows, Linux та Mac OS для фіксованих тестових розмірів. Середня

кількість ітерацій у цій категорії визначається загальною кількістю символьних та числових даних у файлі.

3.4.5. Проведення експерименту використання комбінації символьних та спеціальних символьних даних

Категорія включала файли, що містили поєднання звичайних символів та спеціальних символів. Розміри файлів зберігалися в діапазоні 1 КБ – 512 КБ для текстових файлів та 1 КБ – 66.5 КБ для файлів .doc.

Таблиця 3.5.

Розрахунок співвідношення типу файлу до середнього часу виконання при використанні використання комбінації символьних та спеціальних символьних даних

Категорія файлу	Тип файлу	Використана платформа	Довжина ключа	Середня кількість ітерацій	Час виконання за ітерацію (ітер./сек)	Загальний час виконання (сек)
Категорія (150 КБ для txt, 20 КБ для doc)	.txt	Windows	4	116,504	980	127.000
	.doc/.docx	Windows	4	109,405	972	133.000
	.txt	Linux	4	109,405	969	132.900
	.odt/fodt/.uot	Linux	4	109,405	969	131.157
	.txt	Mac OS	4	109,405	964	131.900
	.page	Mac OS	4	109,405	963	130.340

Таблиця 3.5 ілюструє симуляцію пропонованого алгоритму на різних платформах. Середня кількість ітерацій залежить від сукупної кількості символьних даних та спеціальних символьних даних у файлах. Час виконання за ітерацію відображає кількість ітерацій, виконаних за секунду, а Загальний час виконання — повний час, необхідний для завершення

середньої кількості ітерацій. Довжина ключа залишається фіксованою на 4 символи.

3.4.6. Проведення експерименту використання комбінації числових та спеціальних символьних даних

Категорія включала файли, які містили комбінацію звичайних числових даних та спеціальних символів. Діапазони розмірів файлів незмінні (1 КБ – 512 КБ для .txt, 1 КБ – 66.5 КБ для .doc). Проаналізовано співвідношення типу файлу до середнього часу виконання.

Таблиця 3.6.

Розрахунок співвідношення типу файлу до середнього часу виконання при використанні використання комбінації числових та спеціальних символьних даних

Категорія файлу	Тип файлу	Використана платформа	Довжина ключа	Середня кількість ітерацій	Час виконання за ітерацію (ітер./сек)	Загальний час виконання (сек)
Категорія (150 КБ для txt, 20 КБ для doc)	.txt	Windows	4	164,41	1232	133.393
	.doc/.docx	Windows	4	164,41	1215	135.252
	.txt	Linux	4	164,41	1213	135.459
	.odt/fodt/.uot	Linux	4	164,41	1213	135.323
	.txt	Mac OS	4	164,41	1204	136.489
	.page	Mac OS	4	164,41	1206	134.256

Середня кількість ітерацій у цій категорії відповідає загальній кількості символів у тексті, включаючи числові та спеціальні символи. Час виконання за ітерацію показує кількість ітерацій за секунду, а загальний час виконання — сумарний час виконання.

3.4.7. Проведення експерименту використання комбінації символьних, числових та спеціальних символьних даних

Дана категорія являє собою найбільш складний тестовий набір, що містить комбінацію звичайних символів, числових даних та спеціальних символів. Діапазони розмірів файлів відповідають стандартним умовам. Виконано розрахунок співвідношення типу файлу до середнього часу виконання.

Таблиця 3.7.

Розрахунок співвідношення типу файлу до середнього часу виконання при використанні використання комбінації символьних, числових та спеціальних символьних даних

Категорія файлу	Тип файлу	Використана платформа	Довжина ключа	Середня кількість ітерацій	Час виконання за ітерацію (ітер./сек)	Загальний час виконання (сек)
Категорія (150 КБ для txt, 20 КБ для doc)	.txt	Windows	4	151,912	1201	130.900
	.doc/.docx	Windows	4	151,912	1196	130.000
	.txt	Linux	4	151,912	1195	129.800
	.odt/fodt/.uot	Linux	4	151,912	1193	128.000
	.txt	Mac OS	4	151,912	1194	128.100
	.page	Mac OS	4	151,912	1193	127.500

Таблиця 3.7 представляє результати симуляції алгоритму шифрування на платформах Windows, Linux та Mac OS. Середня кількість ітерацій залежить від сукупної кількості всіх символьних, числових та спеціальних символьних даних у файлах. Довжина ключа є фіксованою, незважаючи на можливість вибору користувачем його комбінації. Час виконання за ітерацію

та загальний час виконання інтерпретуються відповідно як кількість ітерацій за секунду та сумарний час виконання.

3.5. Емпіричний аналіз продуктивності та ефективності пропонованого алгоритму симетричного шифрування

Кількісний аналіз — це методологія, що використовує математичне та статистичне моделювання для об'єктивного вимірювання та дослідження поведінки систем. Шляхом присвоєння числових значень змінним, цей підхід дозволяє створювати формалізовані моделі реальних процесів. У криптографії кількісний аналіз успішно застосовується для оцінки витоку інформації по бічних каналах, вимірювання втрати конфіденційності та оцінки рівня безпеки в анонімних мережах.

У контексті полегшеної симетричної криптографії застосування кількісного аналізу є необхідним для валідації ефективності та надійності алгоритмів, хоча й становить складну задачу через комплексність сучасного програмного забезпечення. Метою даного аналізу є емпіричне вимірювання продуктивності запропонованого алгоритму, визначення його зв'язку з мовами програмування та техніками верифікації, а також чітке окреслення його функціональних обмежень.

Для проведення кількісного аналізу запропонованого алгоритму симетричного шифрування було проведено серію експериментів на масиві з 500 файлів, що включали текстові документи (.txt) та документи формату .doc.

Розміри файлів варіювалися в діапазоні від 1 КБ до 512 КБ.

Для оцінки кросплатформенної сумісності та незалежності алгоритму від операційної системи тестування проводилося на трьох платформах: Windows, Linux та Mac OS.

Для кожного експерименту фіксувалися такі показники продуктивності:

- Середня кількість ітерацій.

- Час виконання однієї ітерації.

- Загальний час виконання процесу шифрування/дешифрування.

Для перевірки надійності алгоритму використовувалися файли з сімома різними категоріями вмісту.

Результати кількісного аналізу продемонстрували високу ефективність та дієвість запропонованого алгоритму в межах визначених умов. Процеси шифрування та дешифрування показали низьку часову та обчислювальну складність, а також мінімальний відсоток помилок. Алгоритм успішно обробив усі сім категорій файлів, що підтверджує його надійність при роботі з різнорідними даними.

Ключовим результатом є підтвердження платформенної незалежності алгоритму, що дозволяє його імплементацію в різноманітних програмних середовищах. Запропоноване рішення забезпечує базові принципи інформаційної безпеки: стабільність, цілісність та конфіденційність даних під час передачі.

З економічної точки зору, алгоритм є економічно ефективною альтернативою для малих підприємств та організацій. Для таких суб'єктів інвестиції у дорогі антивірусні програми та великомасштабні системи безпеки для захисту невеликих обсягів даних є недоцільними. Запропонований алгоритм надає надійний та валідний інструмент для захисту даних за мінімальних витрат.

Незважаючи на позитивні результати, необхідно чітко визначити обмеження даного алгоритму.

Алгоритм коректно обробляє лише символи англійського алфавіту (ASCII), числа та стандартні спеціальні символи. Він не призначений для роботи з іншими мовними кодуваннями.

Ефективність алгоритму гарантується виключно для файлів малого обсягу (в межах 1–512 КБ).

Підсумовуючи, запропонований алгоритм є простим у реалізації, надійним та валідним рішенням для організацій, що потребують ефективного захисту невеликих обсягів даних. Він забезпечує стабільність, цілісність та конфіденційність інформації, є кросплатформенним та становить економічно вигідну альтернативу ресурсоємним системам безпеки, що робить його актуальним для широкого кола користувачів та малих підприємств.

Висновки до розділу

У третьому розділі розроблено та імплементовано алгоритм симетричного шифрування, побудований із використанням постквантових принципів стійкості. Запропонований алгоритм базується на комбінації випадкових ключів і механізмів їх криптографічного перетворення, що забезпечує підвищену ентропію шифротексту. Проведено експериментальні дослідження з використанням різних типів даних, які підтвердили високу швидкодію та надійність алгоритму. Результати емпіричного аналізу показали, що розроблена модель зберігає оптимальний баланс між продуктивністю та криптостійкістю, навіть за умов інтенсивного навантаження. Таким чином, реалізований алгоритм може слугувати ефективною основою для побудови постквантових систем захисту інформації.

ВИСНОВКИ

У магістерській роботі проведено дослідження теоретичних, методологічних та практичних аспектів розробки постквантових моделей і методів криптоаналізу даних, спрямованих на підвищення рівня інформаційної безпеки в умовах стрімкого розвитку квантових обчислень. Робота поєднує системний аналіз сучасних підходів до шифрування даних, оцінку вразливостей класичних криптографічних систем та практичну реалізацію алгоритму симетричного шифрування, адаптованого до вимог постквантового середовища.

У першому розділі здійснено дослідження предметної області криптографічного аналізу даних та постквантових моделей. Проведено глибокий огляд сучасного стану проблеми захисту інформації, визначено основні напрями розвитку постквантової криптографії (PQC), зокрема на основі граткових, кодових, мультилінійних та ізогенійних підходів. У результаті аналізу показано, що більшість класичних алгоритмів (RSA, ECC, Diffie–Hellman) не забезпечують належного рівня безпеки в умовах квантових атак, що вимагає впровадження нових моделей криптографічного захисту, стійких до алгоритмів Шора та Гровера.

У другому розділі проаналізовано існуючі моделі та методи симетричного шифрування, а також концепцію полегшеної криптографії, що є актуальною для систем із обмеженими обчислювальними ресурсами, таких як пристрої Інтернету речей (IoT). Визначено ключові характеристики сучасних симетричних шифрів, розглянуто їхню роль у забезпеченні конфіденційності, цілісності та автентичності даних. Здійснено порівняльний аналіз сучасних алгоритмів полегшеного шифрування та їх придатності до адаптації у постквантовому контексті. Зроблено висновок, що симетричні методи можуть залишатися актуальними за умови модифікації їхніх структурних компонентів і параметрів ключів відповідно до нових криптографічних вимог.

У третьому розділі розроблено, формалізовано та реалізовано власний симетричний алгоритм шифрування, побудований з урахуванням постквантових принципів безпеки. Алгоритм характеризується комбінованим підходом до генерації ключів, що поєднує випадкові параметри та методи їх криптографічного перетворення для зниження ризику передбачуваності. Проведено серію експериментів для різних типів вхідних даних (символьних, числових, спеціальних та змішаних), результати яких продемонстрували високу стійкість алгоритму до криптоаналізу та прийнятну продуктивність при різних умовах навантаження. Отримані результати свідчать про ефективність розробленої моделі та перспективність її використання у сучасних системах захисту даних.

Проведений емпіричний аналіз продуктивності довів, що запропонований алгоритм забезпечує баланс між швидкістю шифрування та криптографічною стійкістю, перевищуючи класичні симетричні схеми за рівнем ентропії шифротексту та варіативністю ключів. Методологічна база дослідження, що включає поєднання дедуктивного, пошуково-експериментального та аналітичного підходів, дозволила побудувати узгоджену модель постквантового криптоалгоритму.

Узагальнюючи результати, можна зробити висновок, що розроблена система є конкурентоспроможною в контексті сучасних вимог інформаційної безпеки та має потенціал для подальшого вдосконалення шляхом інтеграції із гібридними постквантовими протоколами. Робота зробила внесок у розвиток теоретико-практичної бази постквантової криптографії, зокрема у напрямах симетричних шифрів, орієнтованих на високу стійкість до квантового криптоаналізу при збереженні ефективності обчислень.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Digital Signature Algorithm (DSA) – GeeksforGeeks - https://www.geeksforgeeks.org/computer-networks/digital-signature-algorithm-dsa/
2. Diffie-Hellman Key Exchange – Practical Networking .net - https://www.practicalnetworking.net/series/cryptography/diffie-hellman/
3. An Introduction to Block Cipher Mode of Operation | by Helene Kegel | Towards Dev - https://towardsdev.com/an-introduction-to-block-cipher-mode-of-operation-faf3a40f1da1
4. C. E. Shannon, "Communication Theory of Secrecy Systems," Bell System Technical Journal, vol. 28, no. 4, pp. 656-715, 1949.
5. W. Diffie and M. E. Hellman, "New Directions in Cryptography," IEEE Transactions on Information Theory, vol. 22, no. 6, pp. 644-654, 1976.
6. R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, vol. 21, no. 2, pp. 120-126, 1978.
7. B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd ed. John Wiley & Sons, 1996.
8. T. S. Kuhn, The Structure of Scientific Revolutions. University of Chicago Press, 1992.
9. W. Stallings and M. P. Tahiliani, Cryptography and Network Security: Principles and Practice, 7th ed. Pearson Education, 2017.
10. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, Handbook of Applied Cryptography. CRC Press, 1996.
11. Non-Negative Decomposition of Multivariate Information: From Minimum to Blackwell-Specific Information - https://www.mdpi.com/1099-4300/26/5/424

12. Math PQC: Foundations of isogeny-based cryptography – COSIC - https://www.esat.kuleuven.be/cosic/blog/math-pqc-foundations-of-isogeny-based-cryptography/
13. J. Daemen and V. Rijmen, The Design of Rijndael: AES - The Advanced Encryption Standard. Springer, 2002.
14. R. Anderson, E. Biham, and L. Knudsen, "Serpent: A Proposal for the Advanced Encryption Standard," in AES Candidate Conference, 1998.
15. B. Schneier et al., "The Twofish Encryption Algorithm: A 128-Bit Block Cipher," John Wiley & Sons, 1999.
16. T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," IEEE Transactions on Information Theory, vol. 31, no. 4, pp. 469-472, 1985.
17. C. Adams and S. Tavares, "The CAST-128 Encryption Algorithm," RFC 2144, 1997.
18. R. C. Merkle, "Secure Communications over Insecure Channels," Communications of the ACM, vol. 21, no. 4, pp. 294-299, 1978.
19. DILH: Data integrity using linear combination for Hash algorithm - https://www.researchgate.net/publication/286561636_DILH_Data_integrity_using_linear_combination_for_Hash_algorithm?enrichId
20. A. Bogdanov et al., "PRESENT: An Ultra-Lightweight Block Cipher," in Proceedings of CHES 2007, pp. 450-466, 2007.
21. G. Leander, C. Paar, A. Poschmann, and K. Schramm, "New Lightweight DES Variants," in Proceedings of FSE 2007, pp. 196-210, 2007.
22. A. Biryukov and L. Perrin, "State of the Art in Lightweight Symmetric Cryptography," in Designs, Codes and Cryptography, vol. 88, pp. 2905-2936, 2020.
23. D. Engels, X. Fan, G. Gong, H. Hu, and E. M. Smith, "Hummingbird: An Ultra-Lightweight Cryptographic Algorithm for Resource-Constrained Devices," in Proceedings of FC 2010 Workshops, 2010.

24. C. H. Lim and T. Korkishko, "mCrypton – A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors," in Proceedings of WISA 2005, pp. 243-258, 2006.
25. R. Beaulieu, S. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The SIMON and SPECK Lightweight Block Ciphers," in Proceedings of DAC 2015, pp. 1-6, 2015.
26. M. Manifavas, G. Hatzivasilis, K. Fysarakis, and P. Rantos, "A Survey of Lightweight Cryptography for IoT and M2M Communications," in Proceedings of the 10th International Conference on Security for Information Technology and Communications (SecITC), 2017.
27. R. Kumar and R. Aggarwal, "A Lightweight Cryptographic Solution for Mobile Ad-hoc Networks (MANETs)," International Journal of Computer Applications, vol. 48, no. 19, pp. 44-48, 2012.
28. P. W. Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring," in Proceedings of the 35th Annual Symposium on Foundations of Computer Science, pp. 124-134, 1994.
29. D. J. Bernstein and T. Lange, "Post-Quantum Cryptography," Nature, vol. 549, pp. 188-194, 2017.
30. L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehle, "CRYSTALS-Kyber: A CCA-Secure KEM Based on Module-LWE," in Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P), 2018.
31. C. Chen and T. Lange, "Isogeny-Based Cryptography," in Post-Quantum Cryptography, Springer, pp. 241-260, 2017.
32. P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," in Proceedings of CRYPTO'96, pp. 104-113, 1996.
33. E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," Journal of Cryptology, vol. 4, no. 1, pp. 3-72, 1991.

34. S. Vaudenay, "La Scurit de RC4," Report for the French Ministry of Defence, 2005.
35. T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann, and L. Uhsadel, "A Survey of Lightweight-Cryptography Implementations," IEEE Design & Test of Computers, vol. 24, no. 6, pp. 522-533, 2007.
36. J. W. Creswell and J. D. Creswell, Research Design: Qualitative, Quantitative, and Mixed Methods Approaches, 5th ed. Sage Publications, 2018.

ДОДАТКИ

Додаток А

Лістинг А.1. Код шифрування

```
% Початок вимірювання часу
tic;
% --- Читання відкритого тексту з файлу ---
fp    =    fopen('C:\Users\Documents\MATLAB\Thesis\PlainText.txt',
'r');
x = fscanf(fp, '%c');
fclose(fp);
% Визначення довжини тексту
l = length(x);
% Ініціалізація змінної (у цьому коді не використовується)
conValue = 0;
% --- Отримання ключа від користувача ---
% Примітка: 's' означає, що ключ буде прочитано як рядок (string)
key = input('Please give your key input:', 's');
% Ініціалізація стану 'sum' ASCII-сумою перших 4-х символів ключа
sum = key(1) + key(2) + key(3) + key(4);
% --- Відкриття файлу для запису шифротексту ---
fp    =    fopen('C:\Users\Documents\MATLAB\Thesis\CipherText.txt',
'w');
% --- Основний цикл шифрування ---
for i = 1:l
        % Генерація псевдовипадкового значення для XOR
    sum = mod(sum, 100);
    yy = mod(298, sum);
        % Отримання ASCII-коду поточного символу
    decvalue = x(i) / 1; % В MATLAB операції над символами
автоматично використовують їх ASCII-коди
        % Операція шифрування
    xorfinal = bitxor(decvalue, yy);
        % Перетворення числового результату назад у символ
    xorfinal = char(xorfinal);
        % Запис зашифрованого символу у файл
    fprintf(fp, '%c', xorfinal);
        % --- Блок оновлення стану ---
    % для оновлення змінної 'sum' у наступній ітерації.
```

```
        y = dec2bin(sum, 8);
        block1 = strcat(y(1), y(2));
        block2 = strcat(y(3), y(4));
        block3 = strcat(y(5), y(6));
        block4 = strcat(y(7), y(8));
    end
    % --- Завершення роботи ---
    fclose(fp); % Закриття файлу шифротексту
    toc; % Відображення часу, що минув
```

Лістинг А.2. Код дешифрування

```
    % Початок вимірювання часу
    tic;
    % --- Читання шифротексту з файлу ---
    fp   =   fopen('C:\Users\Documents\MATLAB\Thesis\CipherText.txt',
'r');
    x = fscanf(fp, '%c');
    fclose(fp);
    % Визначення довжини шифротексту
    l = length(x);
    % Ініціалізація лічильника
    conValue = 0;
    % --- Отримання ключа від користувача ---
    key = input('Please give your key input:', 's');
    % Ініціалізація стану 'sum' ASCII-сумою перших 4-х символів ключа
    sum = key(1) + key(2) + key(3) + key(4);
    % --- Відкриття файлу для запису розшифрованого тексту ---
    fp                                                          =
fopen('C:\Users\Documents\MATLAB\Thesis\PlainText_from_cipher.txt',
'w');
    % --- Основний цикл дешифрування ---
    for i = 1:l
            % Генерація псевдовипадкового значення для XOR (ідентично
до шифрування)
        sum = mod(sum, 100);
        yy = mod(298, sum);
            % Отримання ASCII-коду поточного символу шифротексту
        decvalue = x(i) / 1;
```

```
        % Операція дешифрування
        xorfinal = bitxor(decvalue, yy);
        % Запис розшифрованого символу у файл
        % Примітка: fprintf з '%c' автоматично перетворює ASCII-код у
символ
        %xorfinal
        fprintf(fp, '%c', xorfinal);
        % --- Повний блок оновлення стану ---
        y = dec2bin(sum, 8);
        block1 = strcat(y(1), y(2));
        block2 = strcat(y(3), y(4));
        block3 = strcat(y(5), y(6));
        block4 = strcat(y(7), y(8));
        conValueBin = dec2bin(conValue, 3);
        block1 = strcat(conValueBin, block1, conValueBin);
        block2 = strcat(conValueBin, block2, conValueBin);
        block3 = strcat(conValueBin, block3, conValueBin);
        block4 = strcat(conValueBin, block4, conValueBin);
        % Оновлення лічильника conValue (циклічно від 0 до 7)
        if (conValue >= 7)
            conValue = 0;
        else
            conValue = conValue + 1;
        end

    end
    % --- Завершення роботи ---
    fclose(fp); % Закриття файлу з розшифрованим текстом
    toc; % Відображення часу, що минув
```