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

КОМПЛЕКСНА МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«ПРОГРАМНИЙ ЗАСІБ ЗАХИЩЕНОГО ПЕРЕДАВАННЯ ІНФОРМАЦІЇ. ЧАСТИНА 1. МЕТОД АВТЕНТИФІКОВАНОГО ШИФРУВАННЯ»

Виконав: студент 2 курсу, групи 1БС-24м
спеціальності 125 Кібербезпека та захист інформації
_____ Дмитро РОГАЧЕВСЬКИЙ
Керівник: В. о. зав. кафедри ЗІ, д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«16» грудня 2025 р.
Опонент: д. т. н., проф., зав. каф. ІТЗ
_____ Олександр РОМАНЮК
«16» грудня 2025 р.

Допущено до захисту
В.о. зав. кафедри ЗІ
д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«16» грудня 2025 р.

Вінниця ВНТУ – 2025 року

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

ЗАТВЕРДЖУЮ
В. о. зав. кафедри ЗІ, д. т. н., проф.
Володимир ЛУЖЕЦЬКИЙ
«24» вересня 2025 року

З А В Д А Н Н Я
НА КОМПЛЕКСНУ МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ
СТУДЕНТУ
Дмитру РОГАЧЕВСЬКОМУ

1. Тема роботи: «Програмний засіб захищеного передавання інформації. Частина 1. Метод автентифікованого шифрування», керівник роботи: Володимир ЛУЖЕЦЬКИЙ, д. т. н., проф., в. о. зав. кафедри ЗІ, затверджені наказом ректора ВНТУ від 24 вересня 2025 року №313.
2. Строк подання студентом роботи 16 грудня 2025 р.
3. Вихідні дані до роботи:
   - метод автентифікованого шифрування – на основі поворотної матриці та гамування;
   - розмір поворотної матриці – 3 × 3;
   - генератор гами – на основі множення кватерніонів;
   - режими формування гами та MAC-коду – суміщені;
   - розрядність блоків даних – 64 біт;
   - розрядність секретного ключа – 64 біт;
   - розрядність MAC-коду – 64 біт.
4. Зміст текстової частини: Вступ. 1. Аналіз інформаційних ресурсів. 2. Розробка методу автентифікованого шифрування. 3. Програмна реалізація методу автентифікованого шифрування. 4. Економічна частина. Висновки. Список використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу: відомі підходи до автентифікованого шифрування: GCM та CCM, відомі підходи до автентифікованого шифрування: OCB та EAX, схема процесу зашифрування, схема процесу розшифрування, узагальнена архітектура програмного засобу для захищеного передавання інформації, узагальнений алгоритм роботи функції encrypt, узагальнений алгоритм роботи функції decrypt, оцінка складності реалізації запропонованого алгоритму, порівняльні оцінки складності реалізації алгоритмів, показники економічної ефективності.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Володимир ЛУЖЕЦЬКИЙ, д. т. н., проф., в. о. зав. кафедри ЗІ	25.09.2025	12.12.2025
2	Володимир ЛУЖЕЦЬКИЙ, д. т. н., проф., в. о. зав. кафедри ЗІ	25.09.2025	12.12.2025
3	Володимир ЛУЖЕЦЬКИЙ, д. т. н., проф., в. о. зав. кафедри ЗІ	25.09.2025	12.12.2025
4	Олександр ЛЕСЬКО, зав. каф. ЕПВМ, к. е. н., доц.	12.12.2025	12.12.2025

7. Дата видачі завдання 24 вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	24.09.2025 – 26.09.2025	
2	Аналіз інформаційних джерел за напрямком магістерської кваліфікаційної роботи	27.09.2025 – 07.10.2025	
3	Науково-технічне обґрунтування	11.10.2025 – 22.10.2025	
4	Аналіз методів автентифікованого шифрування	23.10.2025 – 26.10.2025	
5	Аналіз та формування вимог до методу автентифікованого шифрування	27.10.2025 – 02.11.2025	
6	Розробка методу автентифікованого шифрування	03.11.2025 – 10.11.2025	
7	Програмна реалізація та тестування модуля автентифікованого шифрування	11.11.2025 – 17.11.2025	
8	Розробка розділу економічного обґрунтування доцільності розробки	18.11.2025 – 22.11.2025	
9	Оформлення пояснювальної записки	23.11.2025 – 29.11.2025	
10	Попередній захист та доопрацювання МКР	29.11.2025 – 11.12.2025	
11	Перевірка на наявність текстових запозичень	12.12.2025 – 15.12.2025	
12	Представлення МКР до захисту, рецензування	16.12.2025 – 19.12.2025	
13	Захист МКР	19.12.2025 – 23.12.2025	

Студент _____ Дмитро РОГАЧЕВСЬКИЙ

Керівник комплексної магістерської кваліфікаційної роботи _____ Володимир ЛУЖЕЦЬКИЙ

АНОТАЦІЯ

УДК 004.056.5

Рогачевський Д. Програмний засіб захищеного передавання інформації. Частина 1. Метод автентифікованого шифрування. Комплексна магістерська кваліфікаційна робота зі спеціальності 125 – Кібербезпека та захист інформації, освітня програма – Безпека інформаційних і комунікаційних систем. Вінниця: ВНТУ, 2025. 86 с.

Укр. мовою. Бібліогр.: 28 назв; рис. 21; табл.: 13.

Комплексна магістерська кваліфікаційна робота присвячена розробці методу та програмного засобу автентифікованого шифрування для забезпечення конфіденційності, цілісності та автентичності даних у системах захищеного передавання інформації. У роботі проведено аналіз сучасних методів автентифікованого шифрування та режимів роботи блокових шифрів.

Запропоновано метод автентифікованого шифрування, що поєднує матричне перетворення на основі ортогональної поворотної матриці, сформованої з використанням кватерніонів, та шифрування гамуванням. Особливістю методу є суміщення процесів формування гами та MAC-коду, що дозволяє зменшити обчислювальні витрати та підвищити ефективність шифрування.

Розроблено програмний засіб, який реалізує запропонований метод, та проведено його тестування криптографічної стійкості. Виконано економічне обґрунтування доцільності впровадження розробленого програмного засобу.

Ілюстративна частина складається з 10 плакатів.

Ключові слова: автентифіковане шифрування, кватерніони, поворотна матриця, гамування, MAC-код, кібербезпека, тестування криптостійкості.

ABSTRACT

Rohachevskyi D. Software tool for secure information transfer. Part 1. Authenticated encryption method. Comprehensive master's thesis in the field of 125 – Cybersecurity and Information Protection, educational programme – Information and Communication Systems Security. Vinnytsia: VNTU, 2025. 85 p.

In Ukrainian. Bibliography: 28 titles; fig. 21; tabl.: 13.

This comprehensive master's thesis is devoted to the development of a method and software tool for authenticated encryption to ensure the confidentiality, integrity, and authenticity of data in secure information transmission systems. The thesis analyses modern methods of authenticated encryption and block cipher modes of operation.

A method of authenticated encryption is proposed that combines matrix transformation based on an orthogonal rotation matrix formed using quaternions and encryption by masking. A distinctive feature of the method is the combination of gamma and MAC code formation processes, which reduces computational costs and increases encryption efficiency.

A software tool that implements the proposed method has been developed and tested for cryptographic resistance. An economic justification for the feasibility of implementing the developed software tool has been provided.

The illustrative part consists of 10 posters.

Keywords: authenticated encryption, quaternions, rotation matrix, hashing, MAC code, cybersecurity, cryptographic strength testing.

ЗМІСТ

ВСТУП

Актуальність дослідження зумовлена зростанням обсягів передавання даних і потребою у методах захисту, що одночасно забезпечують конфіденційність, цілісність і автентичність. Традиційні алгоритми не завжди гарантують комплексний захист, що створює умови для модифікації повідомлень та атак повторного відтворення. Тому важливим напрямом криптографії є розроблення автентифікованого шифрування.

Сучасні алгоритми автентифікованого шифрування (AE/AEAD-алгоритми), попри високу криптографічну стійкість, часто мають значну обчислювальну складність, що обмежує їх використання в мобільних застосунках і системах реального часу. Це висуває вимогу щодо розробки нових підходів, які поєднують захищеність із ефективністю.

Перспективним рішенням є застосування кватерніонів і ортогональних поворотних матриць, що забезпечують сильну дифузію та складні нелінійні перетворення. Комбінація такого матричного шифрування з потоковим підходом і вбудованим MAC-кодом дозволяє створити метод, який водночас є продуктивним і стійким до криптоаналітичних атак.

Значний внесок у розвиток методів автентифікованого, інкрементного та кватерніонного шифрування зробили такі дослідники, як Ф. Рогавей, Дж. Кац, Дж. Блек, В. Шуп, Ч. Рекордс, М. Абаджі, М. Дзвонський [1-5].

Об’єктом дослідження є процес автентифікованого шифрування.

Предмет дослідження – метод автентифікованого шифрування з використанням кватерніонних обчислень та програмний засіб, що його реалізує.

Метою дослідження є пришвидшення процесу автентифікованого шифрування, шляхом розробки методу, що базується на використанні кватерніонних і матричних перетворень та шифрування на основі гамування [6].

Для досягнення поставленої мети необхідно виконати такі завдання:

– проаналізувати сучасні методи автентифікованого шифрування та їх обмеження;

– дослідити математичні властивості кватерніонів для застосування у криптографії;

– розробити метод автентифікованого шифрування на основі кватерніонних і матричних перетворень та шифрування на основі гамування;

– розробити програмний засіб, що реалізує метод автентифікованого шифрування;

– провести тестування методу автентифікованого шифрування щодо його криптографічної стійкості.

Наукова новизна роботи полягає у наступному:

– вперше запропоновано метод автентифікованого шифрування, який на відміну від відомих методів поєднує використання поворотної матриці на основі кватерніона та шифрування гамуванням і суміщення процесу формування гами та MAC-коду, що забезпечує пришвидшення процесу автентифікованого шифрування;

– удосконалено метод формування гами шляхом використання множення кодів за модулем 2^{64} степені, що забезпечує сильний лавинний ефект;

Практична цінність роботи полягає в тому, що розроблено програмний засіб для автентифікованого шифрування, що може бути інтегрований у програмні засоби захищеного обміну даними.

Основні результати комплексної магістерської кваліфікаційної роботи обговорювались на міжнародній науково-практичній конференції «DIGITAL TRANSFORMATION: STRENGTHENING THE CYBERSECURITY CAPACITIES IN THE MODERN WORLD» (Польща, Краків) [7] та міжнародній науково-практичній Інтернет-конференції «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ (МН-2026)» (Україна, Вінниця) [8].

1 АНАЛІЗ ІНФОРМАЦІЙНИХ РЕСУРСІВ

1.1 Огляд шифрування в месенджерах

Соціальні мережі, в тому числі месенджери революціонізували глобальну комунікацію та продовжують впливати на неї. Додатки месенджерів стали інтегрованими в наше повсякденне життя.

Основними месенджерами, що використовують люди є:

– Telegram;

– Viber;

– Facebook Messenger;

– WhatsApp;

– Signal.

Неймовірна кількість особистої інформації публікується на цих платформах або надсилається через них за допомогою їхніх повідомлень.

Для забезпечення конфіденційності та заходів безпеки даних, основні додатки месенджерів почали використовувати наскрізне шифрування через протокол Signal [9].

Хоча наскрізне шифрування є безпечною моделлю зв’язку між пристроями, криптографічні примітиви, що лежать в його основі, є життєво важливими для його безпеки. Протокол Signal спочатку використовував AES-256, HMAC-SHA256 та Curve25519, всі три з яких вважаються криптографічно безпечними.

Згідно з даними WhatsApp запровадив повне наскрізне шифрування за допомогою протоколу Signal у 2016 році. Його материнська компанія, Facebook, впровадила додаткове наскрізне шифрування у Facebook Messenger під назвою Secret Conversations.

Facebook також оголосив, що вони працюють над об’єднанням Facebook Messenger, Instagram Direct Message та WhatsApp в єдиний сервіс обміну повідомленнями зі стандартним наскрізним шифруванням.

Telegram запровадив наскрізне шифрування у 2013 році у форматі «Secret Chats», яке базується на власному протоколі MTProto.

Viber у 2016 році оголосив про повне наскрізне шифрування для всіх приватних і групових чатів, а також дзвінків, використовуючи унікальні ключі для кожного пристрою.

Signal від самого початку свого запуску у 2014 році застосовує наскрізне шифрування за замовчуванням, використовуючи однойменний протокол Signal Protocol [9].

Наскрізне шифрування – це спосіб, за допомогою якого два процеси на двох різних пристроях надсилають повідомлення один одному таким чином, що третя сторона, навіть та, що передає повідомлення між ними, не може їх переглянути [10].

Основний принцип полягає у використанні криптографії з відкритим ключем та безпечній передачі ключів між пристроями. Це дозволить лише цим двом процесам розшифрувати будь-які повідомлення, що надсилаються між ними.

Криптографічні примітиви, що лежать в основі наскрізного шифрування, є життєво важливими для його безпеки. Протокол Signal використовує наступні криптографічно захищені примітиви.

Curve25519 – це найшвидша еліптична крива, яку можна використовувати в обміні ключами методом еліптичної кривої Діффі-Геллмана (ECDH). Протокол узгодження ключів ECDH дозволяє двом сторонам встановити спільний секретний ключ через незахищений канал зв’язку. Він створює спільний секретний ключ від двох користувачів, використовуючи закритий ключ одного користувача та відкритий ключ іншого.

Цей спільний секретний ключ потім використовується обома сторонами для шифрування та автентифікації повідомлень між собою. Спільні секретні ключі криптографії еліптичної кривої значно менші за стандартні спільні секретні ключі, зберігаючи при цьому порівнянну криптографічну стійкість. Ця різниця швидко зростає зі збільшенням розмірів ключів. Ключ еліптичної кривої

довжиною 512 бітів дорівнює стандартному асиметричному ключу довжиною 15 360 бітів. Менший розмір ключа також збільшує швидкість обміну ключами.

Використання Curve25519 у протоколі Signal сприяє загальній безпеці та продуктивності протоколу, забезпечуючи швидкий та безпечний механізм узгодження ключів, який можна використовувати для встановлення спільних секретних ключів між пристроями.

AES (Advanced Encryption Standard) – це широко використовуваний стандарт симетричного шифрування, який стосується набору блокових шифрів. Шифри розрізняються за розміром ключа, причому AES-256 (256-бітний ключ) є найчастіше використовуваним.

Шифрування AES виконується в серії раундів, під час яких дані підставляють, переставляють та зсувають. Кількість раундів залежить від розміру ключа, причому AES-256 вимагає 14 раундів. AES широко вважається безпечним стандартом шифрування, і уряд Сполучених Штатів схвалив його використання для державних таємниць. AES також вважається захищеним від атак методом грубої сили за допомогою сучасних технологій.

Протокол Signal використовує AES-256 для шифрування повідомлень між клієнтами. Це забезпечує високий рівень безпеки повідомлень, оскільки AES-256 є дуже безпечним алгоритмом шифрування. Секретні розмови WhatsApp та Facebook Messenger використовують AES-256-CBC для шифрування повідомлень. CBC (Cipher Block Chaining) – це режим роботи блокових шифрів, який робить шифрування більш стійким до певних типів атак, проте він не включає жодної автентифікації повідомлень, тому обидві платформи використовують для цього HMAC-SHA256 [10].

Загалом, використання AES-256 у поєднанні з режимом роботи CBC та HMAC-SHA256 забезпечує високий рівень безпеки повідомлень, що надсилаються через ці платформи.

HMAC-SHA256 – це геш-алгоритм, який використовується для автентифікації та перевірки цілісності повідомлень. HMAC розшифровується як геш-код автентифікації повідомлення.

SHA-256 – це геш-алгоритм, який використовується для створення коду автентифікації. Безпека коду автентифікації залежить від базової геш-функції. SHA-256 – це специфічний геш-алгоритм у стандарті SHA-2, який наразі вважається криптографічно безпечним.

Коли користувач намагається надіслати повідомлення, спільний секретний ключ, створений під час обміну Діффі-Геллмана за еліптичною кривою, використовується для отримання двох ключів, які будуть гешовані разом із повідомленням. HMAC-SHA256 гешує комбінацію першого ключа та повідомлення. Потім він гешує комбінацію отриманого гешу та другого ключа.

Результатом є унікальна комбінація, яку можна відтворити лише за допомогою точно такої ж комбінації повідомлення та ключів, отриманих зі спільного секретного ключу [9, 10].

1.2 Основні відомості про автентифіковане шифрування

Існують десятки термінів та акронімів шифрування, які можуть бути невідомими навіть тим, хто вже деякий час працює з шифруванням. Коли люди вперше починають працювати з шифруванням, вони часто зосереджуються лише на простому завданні зашифрування та розшифрування даних.

Однак вони можуть не помічати інших важливих концепцій, таких як автентифіковане шифрування. Розуміння основ шифрування є життєво важливим першим кроком, але лише опанування основ недостатньо [11].

Шифрування не схоже на інші розробки програмного забезпечення; запущена програма не означає, що робота виконана. Хоча розробник міг зашифрувати дані, щоб захистити їх від сторонніх очей, системи або дані все ще можуть бути вразливими до атаки.

У своїй найпростішій формі шифрування – це процес захисту певної інформації та створення шифротексту, який в ідеалі неможливо відрізнити від випадкових даних.

Шифрування – це оборотний процес, у якому для розшифрування шифротексту та повернення вихідної інформації використовується правильний ключ. Приклад базового зашифрування представлено рис. 1.1 [11].

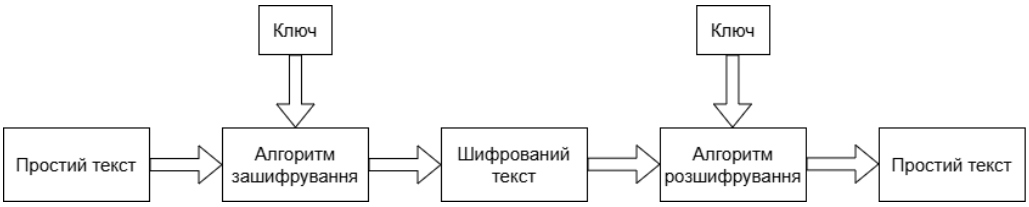


Рисунок 1.1 – Приклад базового зашифрування

Шифрування захищає дані від сторонніх очей. Однак багато алгоритмів шифрування просто намагаються розшифрувати будь-який наданий шифротекст. Іншими словами, ніщо не перевіряє цілісність чи автентичність шифротексту, щоб гарантувати, що він не був змінений. Така відсутність перевірки може призвести до низки проблем.

Для прикладу, розглянемо шифрування інструкцій банку щодо банківського переказу, що містять дані для переказу 100,00 доларів США з рахунку A на рахунок B [11].

Багато хто може вважати, що цього достатньо, оскільки зловмисник не зможе визначити суму грошей або рахунки, що беруть участь у транзакції, якщо дані зашифровані; однак, якщо банк обрав неавтентифікований алгоритм шифрування, його може чекати неприємний сюрприз. Приклад вразливості неавтентифікованого шифрування наведено на рис. 1.2.

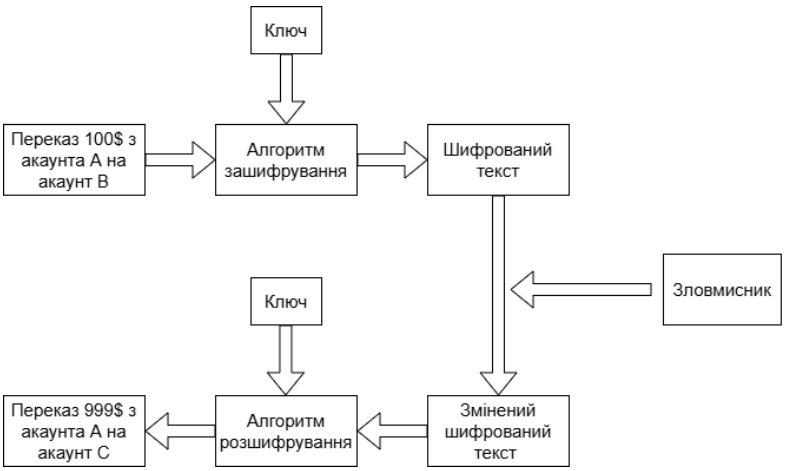


Рисунок 1.2 – Потенційна слабкість неавтентифікованого шифрування

Оскільки шифртекст і ключ не перевірені, зловмисник може змінити шифртекст таким чином, що процес розшифрування не усвідомить, що дані були змінені, і коли змінений шифртекст буде розшифровано, інструкції щодо банківського переказу міститимуть інструкції щодо переказу 999,00 доларів США з рахунку А на рахунок С [11].

Знову ж таки, за допомогою неавтентифікованого алгоритму шифрування зловмисник може зробити це, не будучи виявленим та не знаючи ключа шифрування. Хоча здається, що це не повинно бути можливо, певні алгоритми та режими шифрування вразливі до певних атак, таких як ця.

Аналогічно, деякі неавтентифіковані алгоритми шифрування просто намагатимуться розшифрувати будь-які дані, представлені з будь-яким наданим ключем. Такі алгоритми можуть бути більш вразливими до атак методом перебору, які можуть призвести до отримання ключа шифрування або часткового розшифрування зашифрованого тексту.

Саме тут і з’являється автентифіковане шифрування. Існує кілька варіацій та підходів, але всі вони є різними режимами забезпечення конфіденційності та автентичності зашифрованих даних. Загалом, процес створює код автентифікації повідомлення (MAC, Message authentication code), який також зазвичай називають тегом автентифікації, що використовується для перевірки автентичності даних [11].

Приклад автентифікованого зашифрування та розшифрування представлено на рис. 1.3.

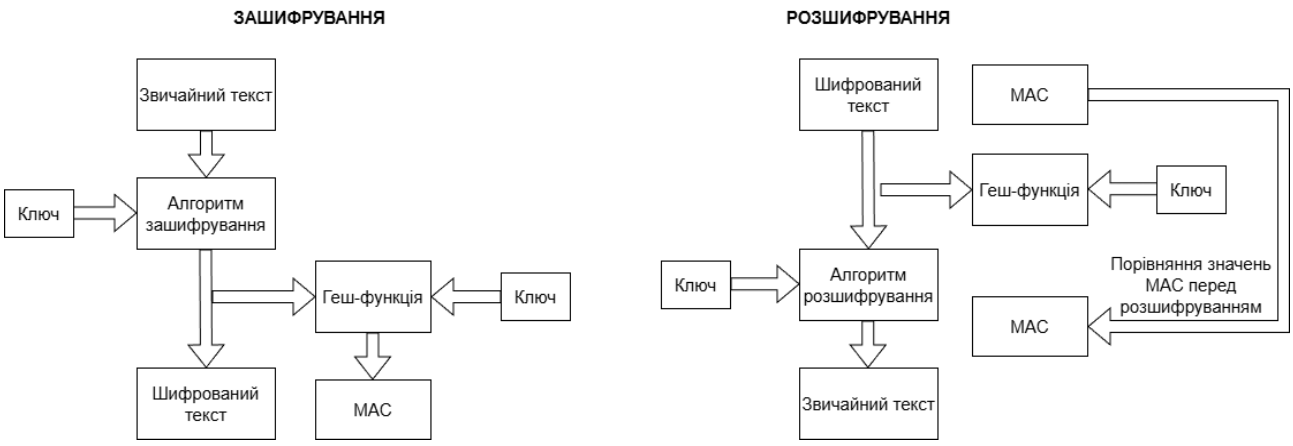


Рисунок 1.3 – Схема автентифікованого зашифрування та розшифрування

Існує три основні підходи до автентифікованого шифрування з використанням MAC-кодів: Encrypt-then-MAC, Encrypt-and-MAC, MAC-then-Encrypt. Як випливає з назви, різниця полягає в тому, коли створюється MAC-адреса та які вхідні дані використовуються для її створення.

На рисунку 3.1 показано основний процес для Encrypt-then-MAC, який вважається більш безпечним, оскільки зашифрований текст і MAC-код можна перевірити без розшифрування будь-яких даних.

Це означає, що технологія може бути перервана, якщо виявить, що ключ, зашифрований текст або тег автентифікації були підроблені або не збігаються. У випадку вищезазначених інструкцій щодо банківського переказу процес, що працює із зашифрованими даними, негайно розпізнає, що зловмисник змінив дані, і відхилить інструкції щодо банківського переказу [11].

1.3 Режими роботи блокового шифру

Оскільки автентифіковане шифрування базується на симетричних блокових алгоритмах, доцільно розглянути відомі режими їх роботи.

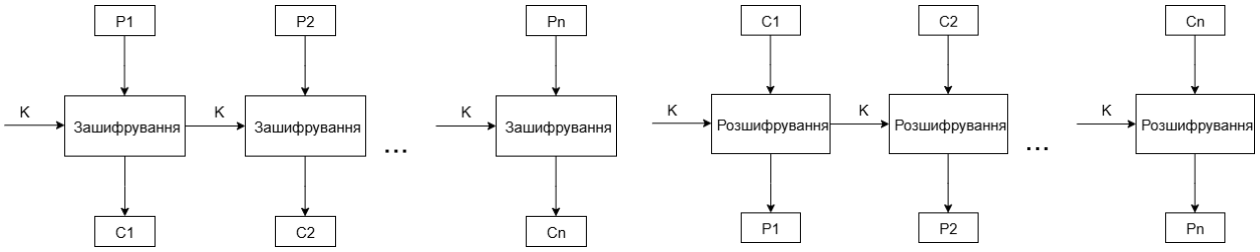
Блоковий шифр – це алгоритм шифрування, який приймає вхідні дані фіксованого розміру (наприклад, b бітів) та створює шифротекст з b бітів. Якщо вхідні дані більші за b бітів, їх можна розділити далі.

Існує кілька режимів роботи блокового шифру, кожен з яких підходить для різних застосувань та використання. Ось кілька поширених режимів:

- електронна книга кодів (ECB, Electronic Codebook);
- ланцюгування шифроблоків (CBC, Cipher Block Chaining);
- зворотній зв’язок за шифротекстом (CFB, Cipher Feedback);
- зворотній зв’язок за виходом (OFB, Output Feedback);
- лічильник (CTR, Counter Mode) [12].

ECB – це найпростіший режим функціонування блокового шифрування. Вона простіша завдяки прямому шифруванню кожного блоку вхідного відкритого тексту, а вихідний код надходить у вигляді блоків зашифрованого

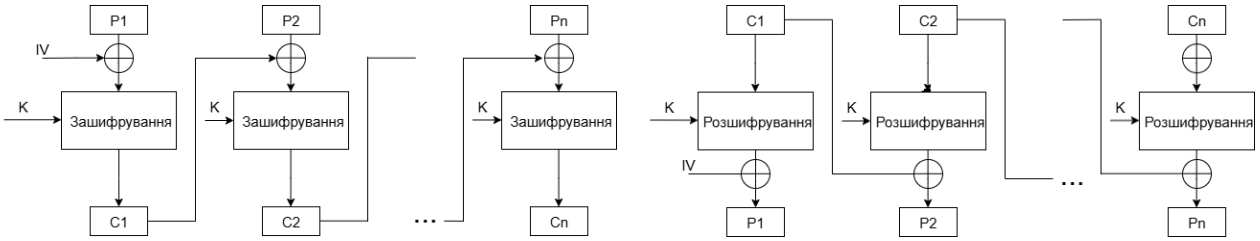
шифротексту. Зазвичай, якщо розмір повідомлення перевищує *b* бітів, його можна розбити на кілька блоків, і процедура повторюється. Процедуру блокового шифрування ECB зображено на рис. 1.4 [12].



а) б)

Рисунок 1.4 – Схема ECB: а – зашифрування, б – розшифрування

CBC – це вдосконалення ECB, оскільки ECB поступається деяким вимогам безпеки. У CBC попередній шифрований блок подається як вхід для наступного алгоритму шифрування після операції XOR з вихідним блоком відкритого тексту. Коротко кажучи, шифрований блок створюється шляхом шифрування виходу XOR попереднього шифрованого блоку та поточного блоку відкритого тексту. Процес шифрування CBC представлено на рис. 1.5 [12].

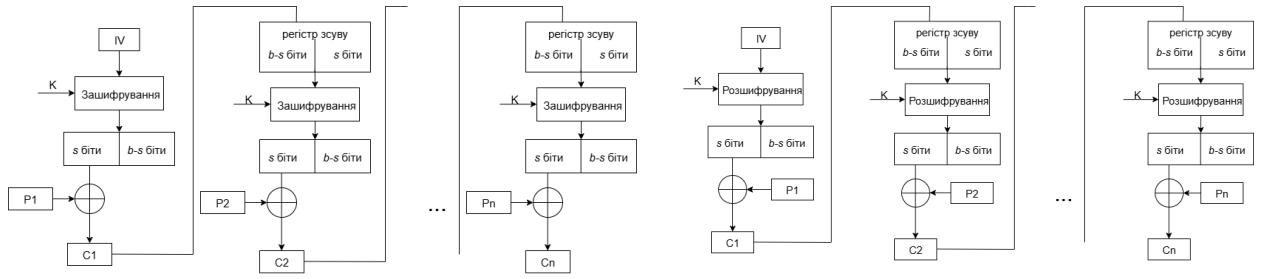


а) б)

Рисунок 1.5 – Схема CBC: а – зашифрування, б – розшифрування

CFB подається як зворотний зв’язок до наступного блоку шифрування з деякими новими специфікаціями: спочатку початковий вектор IV використовується для першого шифрування, а вихідні біти поділяються на набір *s* та *bs* бітів. Ліві *s* біти вибираються разом із бітами відкритого тексту, до яких застосовується операція XOR.

Результат подається на вхід до регістра зсуву, що має *bs* бітів зліва, *s* бітів справа, і процес продовжується. Процес шифрування CFB показано на рис. 1.6.

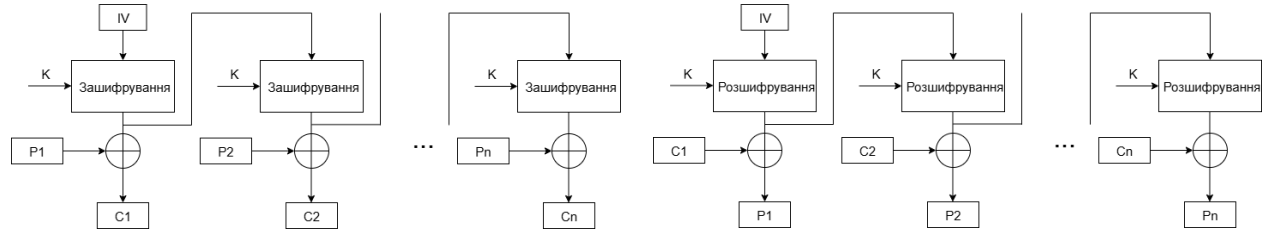


а) б)

Рисунок 1.6 – Схема CFB: а – зашифрування, б – розшифрування

OFB на виході майже такий самий, як і CFB за шифром, за винятком того, що він надсилає зашифрований вихід як зворотний зв’язок замість фактичного шифру, який є виходом XOR. У цьому режимі зворотного зв’язку на виході надсилаються всі біти блоку замість вибраних *s* бітів.

OFB на виході блокового шифру має високу стійкість до помилок передачі бітів. Він також зменшує залежність або зв’язок шифру з відкритим текстом. Схематичне представлення OFB наведено рис. 1.7.

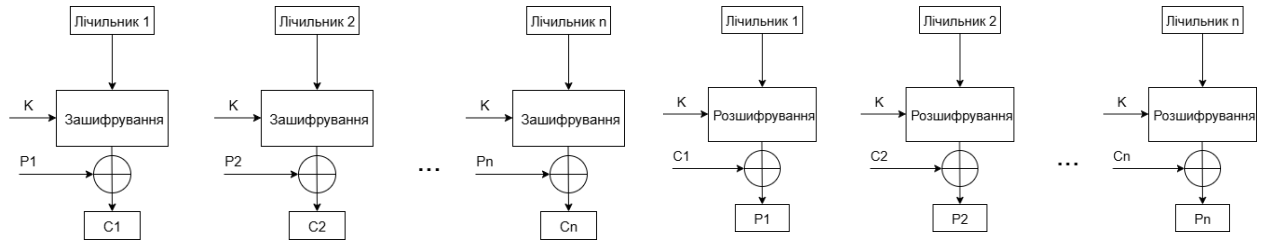


а) б)

Рисунок 1.7 – Схема OFB: а – зашифрування, б – розшифрування

CTR – це проста реалізація блокового шифру на основі лічильника. Щоразу, коли значення, ініціалізоване лічильником, шифрується та передається як вхід для операції XOR з відкритим текстом, що призводить до блоку шифротексту.

Режим CTR не залежить від використання зворотного зв’язку і тому може бути реалізований паралельно [12]. Його проста реалізація показана на рис. 1.8.



а) б)

Рисунок 1.8 – Схема CTR: а – зашифрування, б – розшифрування

Блокові шифри – це методи, що визначають, як безпечно зашифрувати та розшифровувати великі обсяги даних за допомогою блокових шифрів, таких як AES .

Вони гарантують, що один і той самий відкритий текст не завжди створює один і той самий шифротекст, використовуючи такі методи, як ланцюжок та лічильники. Поширені режими включають ECB, CBC, CTR та GCM, кожен з яких пропонує різні рівні безпеки та продуктивності.

Вибір правильного режиму та належне керування елементами, такими як вектори ініціалізації, є важливим для збереження конфіденційності даних та захисту від атак [12].

1.4 Методи автентифікованого шифрування

Автентифіковане шифрування може реалізовуватися у різних варіантах та мати кілька форматів. Зокрема, було визначено й формалізовано концепцію автентифікованого шифрування з пов’язаними даними (AEAD).

Автентифіковане шифрування з пов’язаними даними (AEAD – authenticated encryption with associated data) – це метод, що використовується в криптографії, який забезпечує як шифрування, так і автентифікацію, а також пов’язує додаткові дані разом із зашифрованим повідомленням.

Просто «AD» в «AEAD» означає «Пов’язані дані». Якщо розглядати лише автентифіковане шифрування (AE), воно має здатність забезпечувати як конфіденційність, так і цілісність. Але воно не здатне запобігати атакам повторного відтворення. Для цього «AE» додає пов’язані дані (AD) до

повідомлення, щоб забезпечити автентичність зашифрованого тексту. За допомогою цих додаткових даних можна ідентифікувати будь-які зміни, що відбулися в зашифрованому тексті [13, 14, 15].

Причина важливості AEAD порівняно з традиційними підходами полягає в цих пов’язаних даними. Адреси, порти, порядкові номери тощо можуть бути включені як пов’язані дані. Основна теорія цього механізму полягає в тому, що він пов’язує мережеві пакети із зашифрованими даними.

Наприклад, якщо ми додатково додамо порядковий номер і порт до пакета, це не працюватиме для іншого порядкового номера або іншого порту під час автентифікації.

У порівнянні з традиційними підходами, AEAD має більшу значущість. У табл. 1.1. описано причини її значущості.

Таблиця 1.1 – Порівняльні характеристики традиційного шифрування та AEAD

Критерій	Традиційний підхід	AEAD
Шифрування та автентифікація	Часто виконуються окремо. Напр.: Шифрування – AES, Triple DES Автентифікація – HMAC, цифрові підписи	Виконується однією операцією шляхом поєднання шифрування та автентифікації.
Складність	Реалізація є більш складною через розділення операцій. Якщо їх виконати некоректно, можуть виникати вразливості.	Менш складна завдяки єдиному режиму роботи.
Цілісність даних	Хоча конфіденційність даних забезпечується, цілісність даних не гарантується.	Будь-які зміни в шифротексті або у пов’язаних даних можна виявити під час розшифрування.

Існує кілька режимів AEAD. Деякі з них:

– GCM (Galois/Counter Mode);

– CCM (Counter with CBC-MAC);

– OCB (Offset Codebook Mode);

– EAX (Encrypt-then-Authenticate-then-Translate).

Кожен із вищезазначених режимів AEAD забезпечує конфіденційність, цілісність та автентичність даних. Усі вони широко використовуються в засобах безпечного зв’язку, де безпека даних є критично важливим питанням.

Режими GCM та CCM належать до класу AEAD-алгоритмів, у яких шифрування реалізується на основі режиму лічильника (CTR), а автентифікація забезпечується окремим механізмом формування тега.

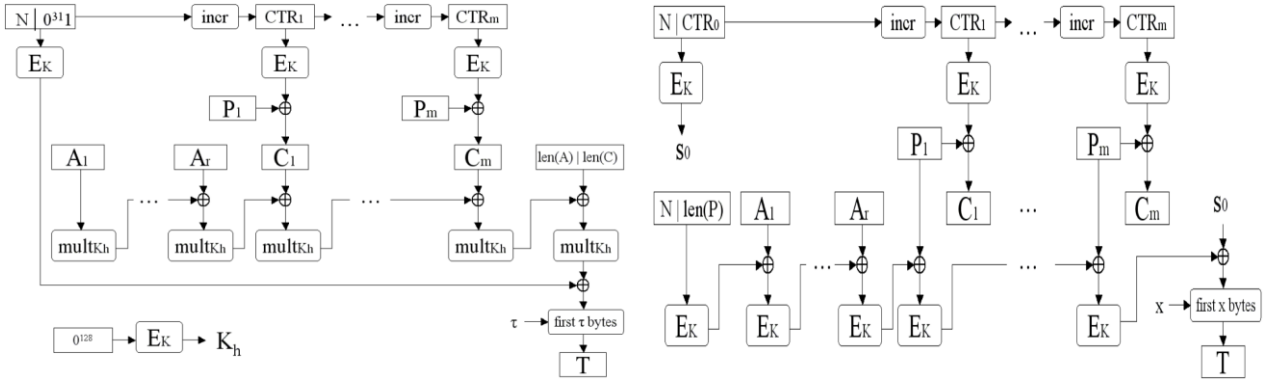
У режимі GCM шифрування виконується шляхом генерації потокової гами за допомогою блокового шифру (зазвичай AES) у режимі CTR з подальшим накладанням її на відкритий текст операцією XOR.

Контроль цілісності та автентичності реалізується за допомогою функції GHASH, що базується на універсальному гешуванні у полі Галуа $GF(2^{128})$. Такий підхід забезпечує високу продуктивність і підтримку паралельної обробки, що зробило GCM одним із найбільш поширених режимів у протоколах TLS, IPsec та VPN-системах. Водночас для збереження криптографічної стійкості режим вимагає суворого дотримання унікальності значення nonce [13, 14, 15].

Режим CCM, на відміну від GCM, використовує алгоритм CBC-MAC для формування коду автентифікації та режим CTR для шифрування. У цьому випадку тег автентифікації формується на основі відкритого тексту та пов’язаних даних, після чого виконується шифрування повідомлення.

Такий підхід є простішим з точки зору реалізації та формалізації стандарту, однак не підтримує ефективну паралельну обробку та має нижчу продуктивність порівняно з GCM [13, 14, 15].

Схематичне представлення відомих підходів до автентифікованого шифрування, а саме GCM та CCM наведено на рис. 1.9.



а) б)

Рисунок 1.9 – Відомі підходи до автентифікованого шифрування: а – GCM, б – CCM

Режими OCB та EAX реалізують альтернативні підходи до побудови автентифікованого шифрування, у яких особлива увага приділяється інтеграції процесів шифрування та автентифікації.

Режим OCB забезпечує шифрування та автентифікацію практично за вартістю одного виклику блокового шифру на блок даних. Його робота ґрунтується на використанні зсувів (offsets), що генеруються для кожного блоку повідомлення та комбінуються з відкритим текстом перед шифруванням .

У результаті процес формування шифротексту та тега автентифікації відбувається в межах єдиного обчислювального механізму. OCB підтримує повну паралельність і характеризується дуже високою продуктивністю, проте його практичне застосування тривалий час стримувалося ліцензійними обмеженнями [13, 14, 15].

Режим EAX реалізує класичну концепцію Encrypt-then-Authenticate, поєднуючи режим лічильника (CTR) із MAC-алгоритмом на основі CBC-MAC. На відміну від GCM та OCB, EAX не потребує використання арифметики поля Галуа або спеціалізованих математичних операцій, що спрощує його програмну реалізацію.

Хоча за швидкодією EAX поступається GCM та OCB, він відзначається високою надійністю та зручністю коректного впровадження в програмних системах загального призначення.

Схематичне представлення відомих підходів до автентифікованого шифрування, зокрема OCB та EAX наведено на рис. 1.10.

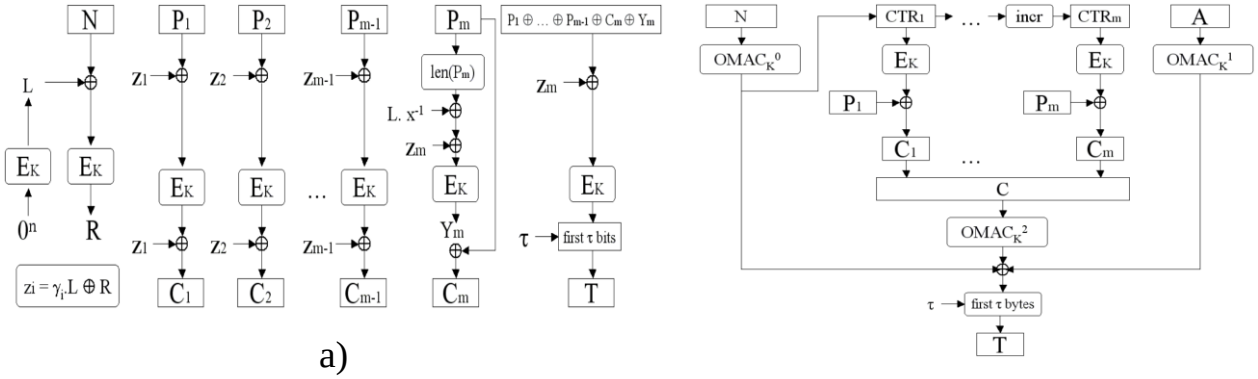


Рисунок 1.10 – Відомі підходи до автентифікованого шифрування: а – OCB, б – EAX

Кожен із наведених вище методів має власні переваги та недоліки, що проявляються у способі реалізації автентифікації, можливості паралельної обробки, рівні продуктивності, а також доступності в програмному чи апаратному забезпеченні. З метою узагальнення та наочного представлення їхніх особливостей нижче наведено порівняльну табл. 1.2. основних характеристик зазначених режимів [13, 14, 15].

Таблиця 1.2 – Порівняльні характеристики методів автентифікованого шифрування

Аспект	GCM	CCM	OCB	EAX
Тип шифрування	Потокове (CTR)	Потокове (CTR)	Блокове з офсетами	Потокове (CTR)
Формування тега автентифікації	GHASH ($GF(2^{128})$)	CBC-MAC	Інтегровано з шифруванням	CBC-MAC
Автентифікація	Над шифротекстом та AAD	Над відкритим текстом та AAD	Над шифротекстом та AAD	Над шифро-текстом та AAD
Паралельна обробка	Підтримується	Не підтримується	Повністю підтримується	Частково
Продуктивність	Висока	Середня	Дуже висока	Середня

Продовження табл. 1.2

Аспект	GCM	CCM	OCB	EAX
Універсальне гешування	Так	Ні	Так	Ні
Вимоги до вирівнювання	Потрібне	Потрібне	Не потрібне	Не потрібне
Потокова (онлайн) обробка	Так	Ні	Так	Так
Складність реалізації	Середня	Низька	Висока	Середня

У сучасних системах безпеки методи автентифікованого шифрування використовуються для забезпечення одночасно конфіденційності і цілісності даних, а саме інтеграція в мережеві протоколи, VPN-сервіси та програмне забезпечення різного призначення [13, 14, 15]. Основні характеристики та особливості використання методів автентифікованого шифрування наведено у порівняльній табл. 1.3.

Таблиця 1.3 – Характеристики та застосування методів автентифікованого шифрування

Алгоритм	Основні сфери застосування	Особливості використання	Ліцензія / доступність
GCM	TLS 1.2/1.3, IPsec, VPN, Ethernet MACsec	Висока швидкодія, суворі вимоги до унікальності nonce	Відкритий стандарт
CCM	Wi-Fi (WPA2), Bluetooth Low Energy	Простота реалізації, обмежена продуктивність	Відкритий стандарт
OCB	Високопродуктивні системи, бездротові мережі	Мінімальні обчислювальні витрати	Ліцензійні обмеження
EAX	Програмні системи загального призначення	Надійність, простота реалізації	Відкритий стандарт

Підсумовуючи, методи автентифікованого шифрування забезпечують комплексний підхід до захисту даних, поєднуючи конфіденційність, цілісність та автентичність у межах єдиного криптографічного механізму.

Режими AEAD, зокрема GCM, CCM, OCB та EAX, дозволяють ефективно інтегрувати процеси шифрування й автентифікації у сучасні мережеві протоколи, VPN-сервіси та програмні засоби захищеного передавання інформації, мінімізуючи ризики атак повторного відтворення та модифікації даних [13, 14, 15].

Згрупований аналіз показує, що режими GCM і CCM базуються на використанні режиму лічильника для шифрування, проте відрізняються механізмами автентифікації та рівнем продуктивності, тоді як режими OCB і EAX реалізують альтернативні підходи до інтеграції автентифікації з шифруванням.

Відмінності у можливості паралельної обробки, обчислювальній складності та вимогах до реалізації визначають практичну доцільність застосування кожного режиму в конкретних умовах [13, 14, 15].

У результаті AEAD-алгоритми залишаються базовим інструментом побудови надійних систем криптографічного захисту та теоретичною основою для розробки більш ефективних методів автентифікованого шифрування.

1.5 Кватерніони та поворотні матриці

Кватерніони – це гіперкомплексні числа рангу 4, які мають дві частини – скалярну та векторну, який є звичайним вектором у тривимірному просторі R^3. Кватерніон q визначається за формулою [5, 16]:

$$q = w + xi + yj + zk, \tag{1.1}$$

де w, x, y, z – дійсні коефіцієнти кватерніона q;

i, j, k – уявні одиниці з такими властивостями: $i^2 = j^2 = k^2 = ijk = -1$, $ij = -ji = k, jk = -kj = i, ki = -ik = k^2 = j$.

Кватерніон також можна записати як транспонований вектор або як композицію скалярної частини w та векторної частини $\vec{v}$:

$$q = [w, x, y, z]^T \text{ або } q = [w, \vec{v}] = [w(x\ y\ z)]. \quad (1.2)$$

Сума двох кватерніонів q_1, q_2 визначається шляхом додавання відповідних компонентів цих кватерніонів, тобто так само, як і для комплексних чисел:

$$q_1 + q_2 = (w_1 + w_2) + (x_1 + x_2)i + (y_1 + y_2)j + (z_1 + z_2)k. \quad (1.3)$$

Добуток двох кватерніонів є складнішим через антикомутативність уявних одиниць цих кватерніонів під час процесу множення. Саме тому поле кватерніонів вважається антикомутативним полем [5, 17]. Добуток двох кватерніонів q_1, q_2 складається зі скалярних та векторних добутків:

$$q_1 \cdot q_2 = [w_1 w_2 - \vec{v}_1 \cdot \vec{v}_2, w_1 \vec{v}_2 + w_2 \vec{v}_1 + \vec{v}_1 \times \vec{v}_2]. \quad (1.4)$$

Крім того, важливо визначити інші властивості кватерніонів: спряжений q^*, норма кватерніона $||q||$ та обернений кватерніон q^{-1} кватерніона q:

$$q^* = w - xi - yj - zk \qquad \big|\big|q\big|\big| = \sqrt{w^2 + x^2 + y^2 + z^2}\,, \quad (1.5)$$

$$q^{-1} = \frac{q^*}{||q||^2} = \frac{w - xi - yj - zk}{w^2 + x^2 + y^2 + z^2}. \quad (1.6)$$

Важливо зазначити, що у випадку одиничного кватерніона, для якого норма дорівнює 1, існує таке співвідношення: $q^{-1} = q^*$.

Існує багато способів представлення обертань об’єктів у тривимірному просторі. Один з них – це використання кватерніонів. Можливість використання представлення Ейлера також існує; однак кватерніони, завдяки своїм унікальним властивостям, стали набагато популярнішими [5].

Щоб отримати обертання кватерніона, нам потрібно мати кватерніон, навколо якого ми будемо обертати інший кватерніон. Якщо розглядати обертаний кватерніон як вектор даних у тривимірному просторі, то ми зможемо реалізувати ідею шифрування кватерніонів.

Розглянемо два кватерніони $q = [w, x, y, z]^T$ та $P = [0, a, b, c]^T$, де вектор $[a, b, c]^T$, який представляє векторну частину кватерніона P з нульовою скалярною частиною, зберігатиме інформацію про фрагмент даних, навколо якого ми хочемо обертати кватерніон q. Отриманий кватерніон P_{rot} буде просторовим відображенням обертаного вектора даних $[a, b, c]^T$. Обертання кватерніона записується як:

$$P_{rot} = q \cdot P \cdot q^{-1}. \tag{1.7}$$

Наразі існує два способи реалізації кватерніонного шифрування. Якщо у нас є інструмент, який може обробляти обчислення кватерніонів, можна реалізувати шифрування відповідно до формули (1.7) – метод кватерніонів [5].

Альтернативний метод, тобто матричний метод, вводить матрицю обертання, яка дозволяє реалізувати кватерніонне шифрування за допомогою множення матриць. Крім того, шифрування таким чином можна розглядати як кватерніонну варіацію шифру Хілла.

Щоб отримати обернутий кватерніон P_{rot} з формули (1.7), використовуючи формули 1.4, 1.5, 1.6, можна визначити матрицю обертання:

$$P_{rot} = \mathbf{\Gamma}(q) \cdot \mathrm{P}\ , \tag{1.8}$$

де $\mathbf{\Gamma}(q)$ – матриця обертання, розрахована на основі векторної частини кватерніона P_{rot}, що визначається формулою (1.7).

Отримана матриця обертання дає нам можливість реалізувати обертання кватерніона за допомогою формули (1.7) без необхідності використання додаткових інструментів для обчислення кватерніонів. Матриця обертання безпосередньо пов'язана з кватерніоном q з якого його було створено [5].

Також можливо створювати матриці обертання вищого порядку, що додатково покращує безпеку зашифрованих даних. Ідея процесу полягає в групуванні кожного стовпця матриці обертання та обробці її елементів як коефіцієнтів наступних кватерніонів.

Процес створення кватерніонів першого порядку з початкової матриці обертання наведено на рис. 1.11.

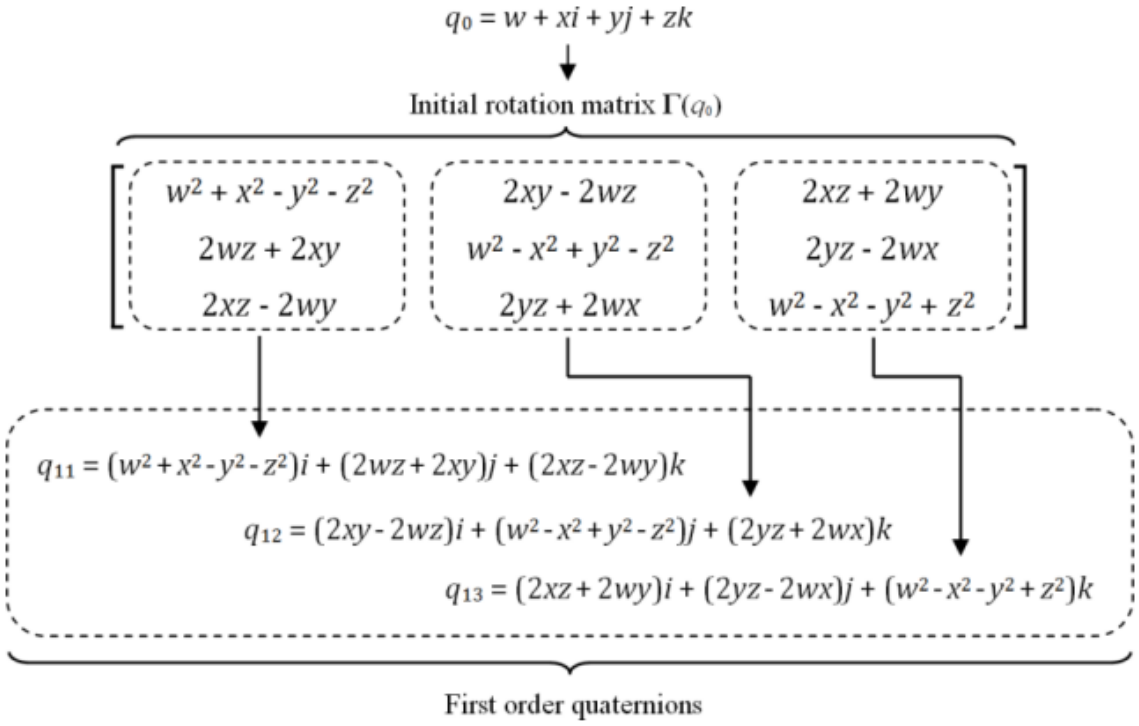


Рисунок 1.11 – Процес створення кватерніонів першого порядку з початкової матриці обертання

Легко помітити, що чим вищий порядок обертання, тим більше кватерніонів цього порядку буде отримано, з яких потім можна визначити інші матриці обертання [5].

Якщо ми визначимо n як порядок обертання, тоді кількість отриманої матриці обертання для відповідного порядку n дорівнюватиме 3^n. Процес є ітеративним, що означає, що для отримання матриць обертання вищого порядку нам спочатку потрібно визначити матриці нижчих порядків [5].

Оскільки матриця обертання має розмір 3×3, можна розширити вектор даних для оптимізації множення матриць. Замість вектора даних P, розмір якого дорівнює 3×1, ми вводимо матрицю даних B розміром 3×3:

$$\text{Б}' = -\mathbf{\Gamma}(q) - \text{Б}. \tag{1.9}$$

У цьому випадку ми отримали можливість шифрувати більші обсяги даних, зберігаючи при цьому правило відповідного кватерніонного шифрування. Однак, якщо ми вирішимо використовувати метод кватерніонів замість матричного методу, нам потрібно розширити векторні коефіцієнти кватерніона P, щоб отримати новий кватерніон B, яке буде кватерніонним представленням матриці даних B:

$$\text{P} = \begin{bmatrix} a \\ b \\ c \end{bmatrix},$$

$$\mathbf{\Gamma}(q) = \begin{bmatrix} w^2 + x^2 - y^2 - z^2 & 2xy - 2wz & 2xz + 2wy \\ 2wz + 2xy & w^2 - x^2 + y^2 - z^2 & 2yz - 2wx \\ 2xz - 2wy & 2yz + 2wx & w^2 - x^2 - y^2 + z^2 \end{bmatrix}, \tag{1.10}$$

$$\text{P} = \left[0, \begin{pmatrix} a \\ b \\ c \end{pmatrix}\right] \rightarrow \quad \text{B} = [0, \begin{pmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{pmatrix}].$$

Шифрування для кватерніонного методу обробляється відповідно до формули (1.7).

Кватерніони забезпечують ефективне й безпечне представлення обертань у тривимірному просторі, що дозволяє використовувати їх для побудови надійних методів шифрування на основі матриць обертання [5].

1.6 Висновки до розділу

У результаті аналізу інформаційних ресурсів встановлено, що сучасні месенджери, зокрема Telegram, Viber, WhatsApp, Facebook Messenger та Signal, підтримують механізми захисту обміну даними на основі наскрізного шифрування, що спрямоване на забезпечення конфіденційності та цілісності переданої інформації.

Протокол Signal, який використовується як базова криптографічна основа в низці популярних рішень, застосовує криптографічні примітиви високого рівня безпеки (AES-256, HMAC-SHA256 та Curve25519), що забезпечують надійне узгодження ключів і автентифікацію повідомлень у практичних системах зв’язку.

Було розглянуто концепцію автентифікованого шифрування та показано обмеження неавтентифікованих схем, у яких відсутня перевірка цілісності шифротексту. Продемонстровано, що такі підходи можуть бути вразливими до модифікації зашифрованих даних і атак повторного відтворення, оскільки розшифрування виконується без перевірки справжності повідомлення.

У цьому контексті автентифіковане шифрування, зокрема підходи Encrypt-then-MAC, Encrypt-and-MAC і MAC-then-Encrypt, забезпечує одночасно конфіденційність, цілісність та автентичність повідомлень за рахунок використання MAC-коду (тега автентифікації).

Окрему увагу приділено режимам роботи блокових шифрів (ECB, CBC, CFB, OFB, CTR), які визначають спосіб оброблення послідовності блоків даних. Показано, що попри простоту реалізації деякі режими (зокрема ECB) мають суттєві обмеження з точки зору криптографічної стійкості, тоді як інші (CBC, CFB, OFB, CTR) забезпечують покращені властивості за рахунок використання векторів ініціалізації, ланцюгування блоків або лічильника.

Це підкреслює важливість правильного вибору режиму та коректного керування параметрами шифрування (ключ, IV, лічильник) для побудови захищених систем.

На основі огляду методів автентифікованого шифрування з пов’язаними даними (AEAD) — режимів GCM, CCM, OCB та EAX — показано їхні переваги порівняно з традиційними схемами «шифрування + окрема автентифікація». AEAD забезпечує єдину операцію шифрування й автентифікації та дозволяє включати пов’язані дані (адреси, порти, порядкові номери тощо), що не шифруються, але перевіряються на цілісність, мінімізуючи ймовірність помилок реалізації.

Згрупований аналіз показує, що режими GCM і CCM базуються на використанні режиму лічильника для шифрування, проте відрізняються механізмами автентифікації та рівнем продуктивності, тоді як режими OCB і EAX реалізують альтернативні підходи до інтеграції автентифікації з шифруванням. Відмінності у можливості паралельної обробки, обчислювальній складності та вимогах до реалізації визначають практичну доцільність застосування кожного режиму в конкретних умовах.

Окремо досліджено математичну основу використання кватерніонів та поворотних матриць у криптографії. Розглянуто основні алгебраїчні властивості кватерніонів та показано, як на їх основі будуються ортогональні матриці обертання, що реалізують перетворення векторів у тривимірному просторі.

Проаналізовано можливість формування матриць обертання вищих порядків, ітеративне одержання нових матриць та розширення вектора даних до матриці для одночасного шифрування більшої кількості елементів. Такий підхід забезпечує сильну дифузію, коли незначна зміна у вихідних даних або параметрах обертання приводить до істотних змін у всіх компонентах зашифрованого блоку.

Таким чином, у першому розділі сформовано теоретичну базу для подальшої розробки методу автентифікованого шифрування: від аналізу сучасних протоколів та режимів автентифікованого шифрування на основі симетричних алгоритмів до вивчення кватерніонних перетворень і поворотних матриць як альтернативної математичної основи шифрування.

2 РОЗРОБКА МЕТОДУ АВТЕНТИФІКОВАНОГО ШИФРУВАННЯ

2.1 Метод автентифікованого шифрування

2.1.1 Метод зашифрування

Пропонується метод автентифікованого шифрування, який ґрунтується на комбінації двох криптографічних підходів – матричного шифрування та шифрування гамуванням [5]. В його основі лежить використання кватерніонів як математичного інструменту для побудови ортогональних поворотних матриць, які виступають у ролі блокового шифрувального механізму. Усі обчислення виконуються у полі $\boldsymbol{F}_p$.

Особливість методу полягає у тому, що використовується не лише секретний ключ, а й секретна поворотна матриця $\mathbf{\Gamma}(q)$, яка сформована на його основі.

Ортогональна поворотна матриця $\mathbf{\Gamma}(q)$ застосовується для перетворення першого блоку повідомлення. Таким чином забезпечується початкова дифузія – тобто рівномірне розподілення впливу кожного біта ключа та відкритого тексту на результат шифрування [18]. Секретний ключ використовується для формування гами і MAC-коду.

Решта повідомлення зашифровується шифруванням гамуванням, де гама формується на основі початкового ключа та блоків відкритого повідомлення з використанням спеціальної нелінійної функції f_{64}, що виконує 64-бітне модульне змішування результатів. Це дозволяє створити криптографічний потік, який має властивість сильного лавинного ефекту: мінімальна зміна вхідного символу чи ключа призводить до суттєвих змін у вихідному шифротексті [19].

Така функція також забезпечує формування геш-значення повідомлення, що зашифровується, яке використовується для автентифікації повідомлення.

Процес зашифрування передбчає виконання таких етапів: множення поворотної матриці на блок відкритого повідомлення, генерування гами, накладання гами на повідомлення. При цьому завершальним етапом формування гами є MAC-код [20] .

Схему, що відображає дії, виконувані під час зашифрування, наведено на рис. 2.1.

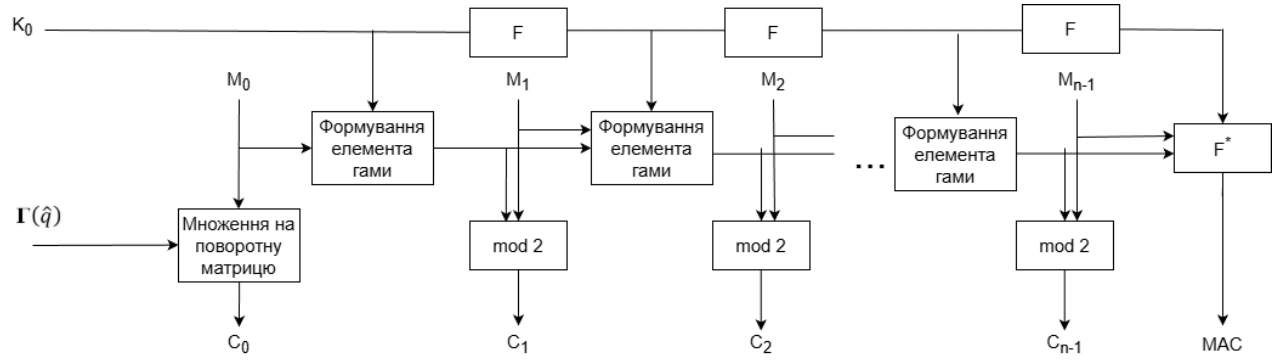


Рисунок 2.1 – Схема процесу зашифрування

Таким чином, метод реалізує комбіновану структуру, у якій перший блок даних обробляється у просторі ортогональних перетворень, а наступні – потоковим генератором.

2.1.2 Метод розшифрування

Процес розшифрування є оберненим до процесу зашифрування і виконується за аналогічними етапами, але в зворотному порядку. Метою розшифрування є відновлення вихідного повідомлення з шифротексту та перевірка його автентичності за допомогою MAC-коду.

Як і під час зашифрування, процес розпочинається з використання спільного секретного ключа K_0 та поворотної матриці $\mathbf{\Gamma}(\hat{q})$, яка формується на основі цього ключа.

Перший блок повідомлення відновлюється за допомогою множення на транспоновану поворотну матрицю $\mathbf{\Gamma}^{\mathrm{T}}(\hat{q})$, а подальші блоки – шляхом послідовного формування елементів гами та виконання операцій XOR.

На рис. 2.2 наведено схему процесу розшифрування, що відображає порядок виконання дій.

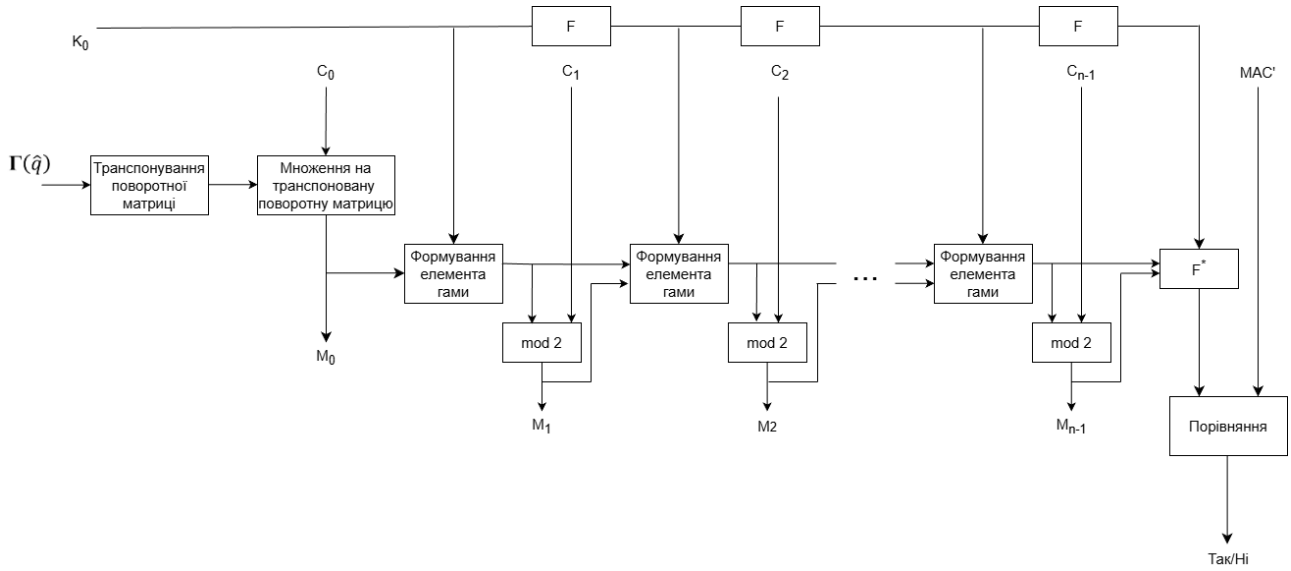


Рисунок 2.2. – Схема процесу розшифрування

Відновлення початкового блоку повідомлення відбувається шляхом множення на транспоновану поворотну матрицю $\mathbf{\Gamma}^{\mathrm{T}}(\hat{q})$ [21]:

$$\mathbf{M}_0 = (\,\mathbf{\Gamma}^{\mathrm{T}}(\hat{q}) \cdot \mathbf{C}_0)\; mod\; 65537, \tag{2.1}$$

де $\mathbf{C}_0$ – початковий зашифрований блок;

$\mathbf{M}_0$ – відновлений блок відкритого повідомлення.

Далі, починаючи з першого блоку, здійснюється послідовне формування елементів гами, ідентичне до процесу зашифрування. Для цього використовуються попередній блок повідомлення та попереднє значення гами:

$$g_i = f_{64}(M_{i-1}, g_{i-1}, F_{i-1}) = ((\Big(\Big(((M_{i-1} \cdot g_{i-1})_L + (M_{i-1} \cdot g_{i-1})_R)\; mod\;\; 2^{64}\Big) \cdot F_{i-1}\Big)_L + \Big(\Big(((M_{i-1} \cdot g_{i-1})_L + (M_{i-1} \cdot g_{i-1})_R)\; mod\;\; 2^{64}\Big) \cdot F_{i-1}\Big)_R)\; mod\;\; 2^{64}. \tag{2.2}$$

Усі операції виконуються за модулем 2^{64}, що гарантує повторюваність гами при однаковому ключі K_0.

Після формування послідовності гами виконується зворотне перетворення шифротексту. Для кожного блоку шифротексту C_i відновлюється відповідний блок відкритого повідомлення за формулою:

$$M_i = C_i \oplus g_{i-1}, \quad i = 0, 1, 2, \dots, n-1. \tag{2.3}$$

Завдяки властивості симетричності операції XOR цей процес є ідентичним до зашифрування, що забезпечує простоту та ефективність реалізації.

Завершальним етапом розшифрування є обчислення MAC-коду для отриманого повідомлення:

$$MAC' = f_{64}(M_{n-1}, g_{n-1}, F_{n-1}). \tag{2.4}$$

Отримане значення MAC' порівнюється з переданим MAC-кодом. Якщо значення збігаються, повідомлення вважається автентичним (Так). У випадку якщо MAC-коди не збігаються, робиться висновок про наявність змін в зашифрованому повідомленні (Ні).

Поєднання множення на ортогональну поворотну матрицю та шифрування гамуванням під час розшифрування забезпечує не лише відновлення даних, але й перевірку їхньої автентичності.

2.2 Процес множення поворотної матриці на блок відкритого повідомлення

Поворотна матриця має такий вигляд:

$$\mathbf{\Gamma}(\hat{q}) = \begin{bmatrix} 1-2(y^2+z^2) & 2(xy-wz) & 2(xz+wy) \\ 2(xy+wz) & 1-2(x^2+z^2) & 2(yz-wx) \\ 2(xz-wy) & 2(yz+wx) & 1-2(x^2+y^2) \end{bmatrix}, \tag{2.5}$$

де кожен елемент матриці розглядається в полі $\mathbf{Z}_p$ для $p = 2^{16} + 1 = 65537$.

Для того, щоб помножити цю матрицю розміром 3×3 на матрицю повідомлення, потрібно, щоб вона також мала розмір 3×3:

$$\mathbf{M}_0 = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}, m_i \in [0, 2^{16}). \tag{2.6}$$

Ця матриця також має бути утворена з чисел, що є елементами поля $\mathbf{Z}_p$. Оскільки модулю p відповідає двобайтний двійковий код, то для створення матриці $\mathbf{M}_0$ потрібно 9 двобайтних чисел, тобто 18 байтів разом.

Якщо довжина повідомлення менша за 18 байтів, застосовується паддинг згідно зі стандартом ISO/IEC 7816-4. Паддинг гарантує, що усі блоки матриці заповнені повністю, а процес розшифрування дозволяє однозначно відновити початкове повідомлення без втрати даних [22].

Після формування блоку відкритого повідомлення виконується множення поворотної матриці $\mathbf{\Gamma}(\hat{q})$ на матрицю $\mathbf{M}_0$ [21]:

$$\mathbf{C}_0 = (\mathbf{\Gamma}(\hat{q}) \cdot \mathbf{M}_0) \; mod \; 65537, \tag{2.7}$$

де $\mathbf{C}_0$ – зашифрована матриця (блок шифротексту), а всі операції виконуються в полі $\mathbf{Z}_p$.

Множення виконується за правилом матричної алгебри:

$$c_{ij} = \sum_{k=1}^{3} \Gamma_{ik}(\hat{q}) \cdot m_{kj} \; (mod \; 65537), \quad i, j = 0,1,2. \tag{2.8}$$

Результатом є матриця $\mathbf{C}_0$, що містить зашифровані елементи першого блоку повідомлення [21].

Оскільки $\mathbf{\Gamma}(\hat{q})$ є ортогональною поворотною матрицею, її множення з будь-яким вектором або матрицею здійснює перетворення, еквівалентне обертанню у тривимірному просторі, тобто забезпечує зміну розташування компонент без втрати інформації.

Після виконання матричного множення зашифрована матриця $\mathbf{C}_0$ перетворюється у послідовність байтів для подальшої обробки або передавання.

Кожен елемент c_{ij} інтерпретується як 16-бітне слово, яке зберігається у форматі «молодший байт – старший байт». Таким чином формується 18-байтний блок шифротексту, який разом із подальшими потоковими блоками утворює повне зашифроване повідомлення.

Описаний процес множення матриці обертання на блок відкритого повідомлення забезпечує першу фазу шифрування, що створює сильну початкову дифузію. Після цієї операції навіть мінімальна зміна одного біта у відкритому тексті або ключі приводить до значних змін у всіх елементах зашифрованої матриці $\mathbf{C}_0$.

Це формує основу для подальшого шифрування гамуванням, у якому гама генерується за допомогою функції f_{64}, що використовує попереднє значення гами, поточний блок даних та стан 64-бітного регістру зсуву з лінійним зворотним зв’язком (LFSR, Linear-feedback shift register), доповнену перевірками на нульові значення та їх корекцією [23].

Фінальне значення гами використовується як MAC-код, який забезпечує автентичність переданих даних.

2.3 Генерування гами на основі множення кватерніонів

Використання кватерніонної арифметики є одним із можливих підходів для генерування гами в межах запропонованого методу автентифікованого шифрування. У цьому випадку процес генерування гами базується на операціях кватерніонного множення, що забезпечує підвищену нелінійність та складнішу просторову взаємодію між блоками даних.

Для кожного кроку i визначається кватерніон $Q_{M_{i-1}}$, сформований із попереднього блоку повідомлення M_{i-1}, та кватерніон $Q_{g_{i-1}}$, утворений із попереднього елемента гами g_{i-1}. Обидва кватерніони мають структуру [24]:

$$Q = a + bi + cj + dk, \tag{2.9}$$

де $a, b, c, d \in [0, 2^{64})$.

Процес генерування гами описується рівнянням:

$$Q_{g_i} = Q_{M_{i-1}} \cdot Q_{g_{i-1}}, \tag{2.10}$$

де множення здійснюється за формулою:

$$\begin{cases} a_i = a_M a_g - b_M b_g - c_M c_g - d_M d_g \\ b_i = a_M b_g + b_M a_g + c_M d_g - d_M c_g \\ c_i = a_M c_g - b_M d_g + c_M a_g + d_M b_g \\ d_i = a_M d_g + b_M c_g - c_M b_g + d_M a_g \end{cases}, \tag{2.11}$$

усі операції виконуються по модулю 2^{64}.

Результатом множення є новий кватерніон Q_{g_i}, який визначає наступний елемент гами [24]:

$$g_i = (a_i + b_i + c_i + d_i) \bmod 2^{64}. \tag{2.12}$$

Таким чином, кожне нове значення гами залежить від чотирьох компонент попередніх кватерніонів, що створює високу ступінь взаємопов’язаності між послідовними блоками. Навіть незначна зміна одного біта у M_{i-1} або g_{i-1} спричиняє суттєві зміни у всіх складових Q_{g_i}.

Формування MAC-коду може виконуватися аналогічно – через кватерніонне множення останнього блоку повідомлення Q_{M_n} на попередній кватерніон гами $Q_{g_{n-1}}$:

$$Q_{MAC} = Q_{M_n} \cdot Q_{g_{n-1}},$$
$$MAC = (a_{MAC} + b_{MAC} + c_{MAC} + d_{MAC}) \; mod \; 2^{64}. \quad (2.13)$$

Такий підхід забезпечує єдність процесів шифрування та автентифікації на основі єдиної кватерніонної структури, що підвищує узгодженість математичної моделі алгоритму [24].

Однак, незважаючи на переваги підвищеної нелінійності, реалізація кватерніонного множення потребує більших обчислювальних витрат, що може знижувати швидкодію системи при обробці великих обсягів даних у реальному часі.

У даній роботі для досягнення балансу між криптографічною стійкістю та обчислювальною ефективністю було обрано генерування гами на основі множення цілих чисел, яка забезпечує високу швидкість обчислень при збереженні достатнього рівня лавинного ефекту та захисту від аналітичних атак.

2.4 Генерування гами на основі множення цілих чисел

Етап генерування гами є центральною частиною шифрування гамуванням розробленого методу автентифікованого шифрування. Він відповідає за створення псевдовипадкової послідовності значень, які накладаються на відкритий текст для його шифрування.

Формування гами виконується за формулою:

$$g_i = f_{64}(M_{i-1}, g_{i-1}, F_{i-1}) = (\Big(\Big(((M_{i-1} \cdot g_{i-1})_L + (M_{i-1} \cdot g_{i-1})_R) \bmod 2^{64}\Big) \cdot F_{i-1}\Big)_L + \Big(\Big(((M_{i-1} \cdot g_{i-1})_L + (M_{i-1} \cdot g_{i-1})_R) \bmod 2^{64}\Big) \cdot F_{i-1}\Big)_R) \bmod 2^{64}. \quad (2.14)$$

де $i = 0, 1, 2, \dots, n-1$;

M_{i-1} – $(i-1)$-й блок відкритого повідомлення;

g_{i-1} – $(i-1)$-й елемент гами;

F_{i-1} – стан 64-бітного LFSR, оновлений після кожного блоку;

$(\ \ )_L$ – старші розряди добутку (64 біт);

$(\ \ )_R$ – молодші розряди добутку (64 біт).

При формуванні елементів гами можуть виникнути ситуації, коли елемент гами дорівнює нулю, що призведе до формування усіх елементів гами з таким же значенням. Для усунення цього, пропонується аналізувати блок повідомлення і елемент гами на наявність нульового значення. Якщо маємо нульове значення, то воно замінюється на попередні значення розгорнутого ключа:

$$M_{i-1} = \begin{cases} M_{i-1}, \text{якщо } M_{i-1} \neq 0, \\ K_{i-1}, \text{якщо } M_{i-1} = 0,' \end{cases}$$
$$g_{i-1} = \begin{cases} g_{i-1}, \text{якщо } g_{i-1} \neq 0, \\ K_{i-2}, \text{якщо } g_{i-1} = 0,' \end{cases} \quad (2.15)$$

де K_{i-1} та K_{i-2} – відповідно попередні значення розгорнутого ключа, отримані шляхом ітерацій LFSR, ініціалізованого значення K_0.

Функція f_{64} забезпечує сильний лавинний ефект: навіть мінімальна зміна одного біта в M_{i-1}, g_{i-1} або в LFSR викликає суттєві зміни у результаті. Це робить генератор гами стійким до статистичних атак та атак із частково відомим або вибраним відкритим текстом [23].

Генерування гами продовжується до моменту завершення шифрування всіх блоків повідомлення. Після цього формується MAC за формулою:

$$MAC = f_{64}(M_{n-1}, g_{n-1}, F_{n-1}) = ((((M_{n-1} \cdot g_{n-1})_L + (M_{n-1} \cdot g_{n-1})_R) \bmod 2^{64}) \cdot F_{n-1})_L + (((M_{n-1} \cdot g_{n-1})_L + (M_{n-1} \cdot g_{n-1})_R) \bmod 2^{64}) \cdot F_{i-1})_R) \bmod 2^{64}. \quad (2.16)$$

MAC є геш-значенням повідомлення, що залежить від самого повідомлення та від секретного ключа і служить для перевірки автентичності повідомлення на приймальній стороні.

Таким чином, процес генерування гами інтегрує механізм шифрування та автентифікації в єдину систему. Такий підхід унеможливлює незалежне підроблення блоків, оскільки будь-яке локальне втручання у структуру шифротексту спричиняє суттєві зміни у подальших значеннях гами та порушення цілісності MAC-коду [20].

2.5 Процес накладання гами на повідомлення

Після генерування гами виконується безпосереднє зашифрування відкритого повідомлення шляхом накладання гами на блоки повідомлення. На цьому етапі реалізується принцип шифрування гамуванням, у якому кожен елемент гами виконує роль псевдовипадкового ключа для відповідного блоку повідомлення.

Процес зашифрування виконується за формулою:

$$C_i = M_i \oplus g_{i-1}\ ,\ i = 0, 1, 2, \dots, n-1, \quad (2.17)$$

де M_i – 64-бітний блок відкритого повідомлення;

g_{i-1} – елемент гами, сформований на попередньому етапі;

C_i – i-й блок шифротексту.

Це дозволяє забезпечити просту та ефективну реалізацію алгоритму як у програмному, так і в апаратному вигляді. Для кожного нового блоку повідомлення використовується нове значення гами, обчислене за допомогою функції f_{64}.

Процес зашифрування завершується формуванням MAC, який забезпечує перевірку цілісності всього повідомлення. При отриманні повідомлення приймаюча сторона, маючи спільний ключ K_0, відтворює ту саму послідовність гами та виконує обчислення MAC-коду. Якщо отримане значення збігається з переданим, повідомлення вважається автентичним [20].

Таким чином, процес накладання гами поєднує у собі функції шифрування гамуванням та контролю цілісності переданого повідомлення.

2.6 Оцінка складності реалізації алгоритму

Нехай повідомлення, що підлягає шифруванню складається з n блоків. Алгоритм зашифрування передбачає виконання таких дій: множення поворотної матриці $\mathbf{\Gamma}(\hat{q})$ на матрицю $\mathbf{M}_0$, формування елементів гами g_i, накладання елементів гами на блоки повідомлення. Виходячи з цього маємо таку складність алгоритму:

$$S_{\text{зашиф.}} = S_{\text{множ.}} + S_{\text{роз.кл.}} + S_{\text{гам.}} + S_{\text{наклад.}}, \tag{2.18}$$

де $S_{\text{множ.}}$ – складність множення матриць. $S_{\text{множ.}}$= 54 оп.;

$S_{\text{роз.кл.}}$ – складність розгортання ключа. $S_{\text{роз.кл.}}$= $3(n+1)$ оп.;

$S_{\text{гам.}}$ – складність формування елементів гами. $S_{\text{гам.}}$ = $8(n+1)$ оп.;

$S_{\text{наклад.}}$ – складність накладання елементів гами на блоки повідомлення, $S_{\text{наклад.}}$= $(n-1)$ оп..

Загальна кількість операцій дорівнюватиме:

$$S_{\text{зашиф.}} = (12n + 64)\text{оп.} \tag{2.19}$$

Процедура розшифрування передбачає виконання таких самих дій як і зашифрування, а також додаткової операції порівняння MAC-кодів.

В табл. 2.1 наведено оцінки складності для відомих AE/AEAD-алгоритмів і запропонованого. Всі значення наведено як середня кількість **ос**новних арифметичних операцій на один 64-бітний блок**.**

Таблиця 2.1 – Оцінка складності

Алгоритм	Кількість операцій на блок	Тип операцій
AES-GCM	≈ 150–180	128-бітні XOR, множення у GF(2^{128}), 10 раундів AES
AES-CCM	≈ 200–220	10 AES-раундів × 2 проходи
AES-EAX	≈ 190–210	AES-операції у двох проходах
AES-OCB	≈ 120–140	AES-раунди, XOR
Пропонований метод	≈ 12 операції/блок (після ініціалізації)	множення, LFSR, XOR, додавання mod 2^{64}

Аналіз оцінок, наведених у таблиці, показує, що алгоритмічна складність реалізації запропонованого методу у 10-18 разів менша, порівняно з відомими AEAD-алгоритмами. Це є наслідком складності реалізації кожного раунду і певної кількості раундів.

2.7 Висновки до розділу

У результаті було розроблено метод автентифікованого шифрування, який поєднує переваги матричного шифрування та шифрування гамуванням у єдиній криптографічній схемі.

Запропонований підхід забезпечує не лише конфіденційність, але й автентичність переданих даних, що робить його придатним для застосування у сучасних системах захисту інформації.

Основою підходу є ортогональна поворотна матриця, сформована з кватерніонного секретного ключа. Її застосування до першого блоку забезпечує

сильну дифузію: зміна одного біта ключа чи відкритого тексту впливає на весь зашифрований блок, підвищуючи стійкість до аналітичних атак.

Наступні блоки шифруються потоковим методом за допомогою гами, що генерується нелінійною функцією f_{64}, яка враховує попереднє значення гами, попередній блок повідомлення та стан 64-бітного LFSR.

Такий механізм створює лавинний ефект і ускладнює статистичний аналіз. Окремий спосіб обробки нульових блоків запобігає формуванню вироджених послідовностей гами.

MAC-код інтегровано безпосередньо у процес зашифрування: він формується тією ж f_{64}, тому залежить від усього повідомлення та ключа. Будь-яка зміна шифротексту порушує коректність MAC-коду, що забезпечує контроль цілісності.

Було розглянуто два підходи до генерування гами – на основі кватерніонів та 64-бітних чисел. Перший дає вищу нелінійність, але є надто повільним; тому обрано ефективнішу цілочисельну реалізацію, що зберігає необхідні криптографічні властивості.

Оцінка складності показала, що обробка повідомлення з n блоків описується формулою $S_{\text{зашиф.}} = (12n + 64)$ операцій, що у 10–18 разів менше за витрати типовими AE/AEAD-алгоритмами (120–220 операцій на блок). Це забезпечує високу швидкодію та придатність для продуктивних систем.

Алгоритм симетричний відносно зашифрування й розшифрування: застосовується транспонована поворотна матриця, відтворюється гама та перевіряється MAC-код. Така структура спрощує реалізацію і гарантує як конфіденційність, так і автентичність даних.

Отже, запропонований метод поєднує математичну стійкість кватерніонних перетворень, ефективність потокового шифрування та вбудований механізм автентифікації, що робить його перспективним для сучасних систем захисту інформації. Після теоретичного опису було виконано його програмну реалізацію.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДУ АВТЕНТИФІКОВАНОГО ШИФРУВАННЯ

3.1 Обґрунтування вибору мови програмування та середовища розробки

Вибір мови програмування – це критично важливий етап у плануванні та реалізації будь-якого програмного продукту чи проєкту. На цьому етапі необхідно детально проаналізувати вимоги та специфіку проєкту, щоб визначити найбільш відповідну мову для його успішної реалізації.

Сьогодні доступний широкий спектр мов програмування, кожна з яких має свою унікальну структуру (синтаксис, семантика), цільові області застосування, а також власні сильні та слабкі сторони. Хоча конкретний вибір залежить від поставлених завдань, Python часто виділяється серед інших завдяки своєму набору властивостей.

Python – одна з найпопулярніших та найшвидше зростаючих мов програмування. По суті, це інтерпретована, високорівнева, універсальна та об’єктно-орієнтована мова сценаріїв, що означає наступне:

1) Інтерпретований – інтерпретатор обробляє вихідний файл під час виконання, він зчитує рядки коду один за одним і виконує те, що йому кажуть.

2) Високорівневий – Python спирається на легкі для читання структури, які пізніше перетворюються на низькорівневу мову, тобто вихідний код, що виконується на центральному процесорі (CPU) комп’ютера.

3) Загального призначення – Python можна використовувати майже для всього. Він застосовний майже в кожній галузі для різноманітних завдань.

4) Об’єктно-орієнтований – ця парадигма програмування забезпечує загальну орієнтацію на сценарії та потужне структурування коду.

Python добре підходить для всіх форм програмування, що сприяє швидкому зростанню його бази користувачів [25].

Мова має багато переваг:

1) Простота. Python має зрозумілий і лаконічний синтаксис, тому його легко вивчати, читати, ділитися кодом і підтримувати його.

2) Потужний набір інструментів. Мова має повнофункціональні IDE з перевіркою синтаксису, налагоджувачами та браузерами коду, а також велику кількість сторонніх бібліотек і фреймворків, що спрощують розробку.

3) Швидкість розробки. Завдяки динамічній природі та можливості повторного використання компонентів Python забезпечує швидке створення програмного забезпечення й скорочує час виведення продукту на ринок.

4) Гнучкість. Python підтримує об’єктно-орієнтований і процедурний підходи, має різні типи даних і дозволяє ефективно проводити дослідницький аналіз даних.

5) Портативність. Програми на Python працюють у будь-якій сучасній операційній системі (Linux, Windows, macOS, UNIX) без додаткового налаштування.

6) Сильна спільнота. Велика кількість користувачів і розробників постійно розширює екосистему Python, створюючи нові бібліотеки та забезпечуючи потужну підтримку [25].

Для розробки на Python важливо використовувати відповідне середовище розробки.

Visual Studio Code (VS Code) вважається одним із найкращих середовищ для розробки на Python завдяки поєднанню легкості, швидкодії та потужного функціоналу. Це безкоштовний, кросплатформний і високо налаштовуваний редактор, який має усі необхідні можливості для ефективної роботи з Python-кодом [26].

По-перше, VS Code забезпечує відмінну підтримку Python. Завдяки офіційному розширенню Python користувач отримує синтаксичне підсвічування, інтелектуальні підказки (IntelliSense), автодоповнення коду, автоматичне форматування за допомогою інструментів Black або autopep8, а також лінтинг для виявлення помилок у реальному часі. Це значно спрощує написання чистого та якісного коду.

По-друге, налаштування VS Code дуже просте і швидке. Для початку достатньо встановити VS Code, Python і розширення Python з маркетплейсу. Уже після цього можна одразу запускати скрипти, налаштовувати середовище виконання та працювати над проектами без складних конфігурацій.

Ще однією перевагою є вбудований термінал, який дозволяє запускати Python-програми, керувати віртуальними середовищами (venv), встановлювати пакети через pip і відлагоджувати код – усе без виходу з редактора.

Крім того, VS Code має потужну систему налагодження. Вона дозволяє встановлювати точки зупину, крокове виконувати код, переглядати значення змінних і виразів, що робить процес відлагодження інтуїтивним і ефективним.

Великим плюсом є ринок розширень – тисячі додаткових модулів, які розширюють функціонал VS Code. Для Python особливо корисними є розширення Jupyter (для роботи з ноутбуками), Code Runner (для швидкого запуску скриптів) та Pylance (для покращеного автодоповнення).

Отже, VS Code – це ідеальний вибір як для початківців, так і для досвідчених Python-розробників. Він поєднує простоту текстового редактора з можливостями повноцінного IDE: швидкий, зручний, гнучкий і налаштований на максимальну продуктивність.

Після вибору мови програмування та середовища розробки, необхідно розробити алгоритм роботи методу автентифікованого шифрування, щоб успішно виконати його програмну реалізацію [26].

3.2 Архітектура програмного засобу захищеного передавання інформації

Перед реалізацією програмного засобу було розроблено його архітектуру, яка повинна забезпечити надійність, гнучкість та розширюваність. Основними вимогами до архітектури є модульність, масштабованість, відсутність постійного зберігання даних, а також наявність чіткої взаємодії між компонентами без використання глобальних змінних. Додатковою вимогою стало включення у криптографічне ядро внутрішнього регістру зсуву, який

використовується як окремий елемент генерування гами та формування MAC-коду. Завдяки цьому засіб можна легко інтегрувати у більш складні системи або вдосконалювати в майбутньому.

Архітектура побудована за модульним принципом і складається з двох основних частин – модуля зашифрування та модуля розшифрування. Кожен із модулів має власну структуру, внутрішні функції та етапи обробки даних. Модулі об’єднані спільним механізмом керування ключами, гамуванням та MAC-кодом.

У межах цього механізму використовується 64-бітний LFSR, стан якого ініціалізується від ущільненого ключа K_0 і далі змінюється під час обробки кожного блоку даних. Узагальнену архітектуру програмного засобу для захищеного передавання інформації подано на рис. 3.1.

Рисунок 3.1 – Узагальнена архітектура програмного засобу для захищеного передавання інформації

Модуль зашифрування виконує попередню обробку вхідного повідомлення, його поділ на частини та підготовку до подальшого зашифрування. Після цього повідомлення проходить через етап побудови ортогональної поворотної матриці, яка використовується для основного шифрувального перетворення. Паралельно з матричним перетворенням запускається регістр зсуву, що генерує послідовність внутрішніх станів, на основі яких обчислюється значення гами для кожного 64-бітного слова повідомлення.

Паралельно відбувається обчислення послідовності значень, що застосовується для потокового шифрування решти повідомлення. Для кожного слова даних гама оновлюється за допомогою нелінійної функції, яка комбінує попереднє значення гами, поточний стан регістру зсуву та слово повідомлення, що підвищує чутливість шифру до змін у вхідних даних. На завершальному етапі формується MAC, який дозволяє перевірити автентичність повідомлення. MAC-код обчислюється з використанням тієї самої гама-ланцюжка та регістру зсуву, що забезпечує єдиний криптографічний контекст для шифрування й автентифікації.

Модуль розшифрування виконує зворотні дії, відновлюючи початкове повідомлення з шифротексту. На першому етапі здійснюється обернене перетворення поворотної матриці, що дозволяє отримати початковий блок даних. Далі на основі того самого ключа K_0 повторно ініціалізується регістр зсуву, після чого кроки його оновлення синхронно відтворюються для кожного блоку, що дозволяє послідовно відновити гамову послідовність.

Далі проводиться розшифрування решти повідомлення шляхом зворотних обчислень. Паралельно з розшифруванням повторно обчислюється MAC-код; отримані дані проходять порівняння розрахованого MAC-коду з переданим значенням, що забезпечує підтвердження правильності та цілісності повідомлення. У випадку невідповідності MAC-кодів повідомлення вважається зміненим або пошкодженим.

Між двома модулями існує чітка взаємодія, яка дозволяє забезпечити повний цикл перетворення даних – від початкового тексту до зашифрованого повідомлення та назад.

Стан регістру зсуву не передається по каналу зв’язку і не зберігається у постійній пам’яті: він повністю відновлюється на стороні розшифрування з використанням спільного ключа K_0 та структури повідомлення, що виключає можливість доступу до внутрішніх параметрів гами ззовні. Усі операції виконуються в оперативній пам’яті, що виключає можливість доступу до конфіденційних даних ззовні.

Запропонована архітектура повністю відповідає висунутим вимогам. Вона має модульну побудову, не використовує постійних сховищ, підтримує можливість доповнення новими алгоритмами шифрування та може бути використана як частина більшої системи захисту інформації.

Включення регістру зсуву як окремого компоненту генерування гами дозволяє підвищити криптографічну стійкість, розширити простір станів системи та ускладнити проведення криптоаналітичних атак.

На основі побудованої архітектури перейдено безпосередньо до розробки програмного засобу, який реалізує описані принципи роботи та взаємодії модулів.

3.3 Розробка алгоритму модуля шифрування

Алгоритм роботи застосунку варто розглядати частинами – за основними функціональними модулями. У межах даної розробки ключове місце займають два модулі – модуль зашифрування та модуль розшифрування, які забезпечують повний цикл обробки даних від відкритого тексту до отримання зашифрованого повідомлення і назад.

Взаємодія модулів побудована таким чином, що обидві операції виконуються у потоковому режимі, з використанням матричних перетворень та динамічно змінюваних параметрів гами.

Узагальнений алгоритм роботи функції зашифрування наведено на рис. 3.2.

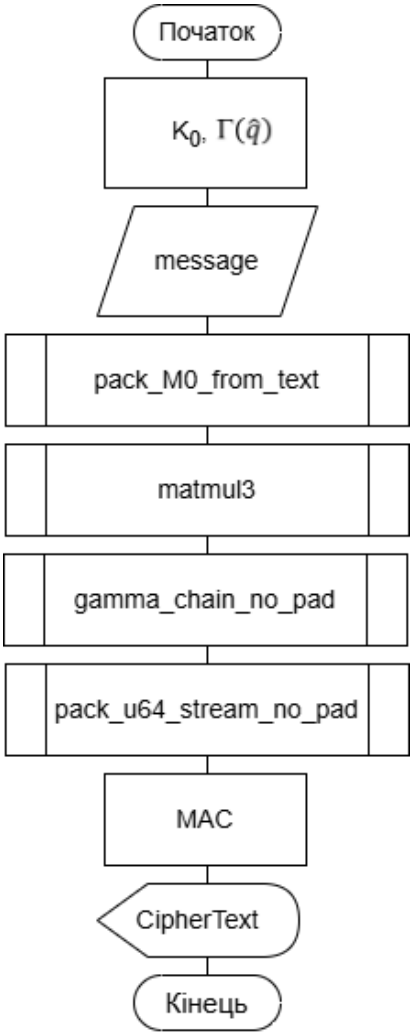


Рисунок 3.2 – Узагальнений алгоритм роботи функції encrypt

Модуль зашифрування розпочинає роботу з отримання основних параметрів, що включають ущільнений секретний ключ та ортогональну поворотну матрицю. На їх основі формується первинний стан системи, а також ініціалізується регістр зсуву, який використовується для подальшого генерування гами під час потокового шифрування. Вхідне повідомлення перетворюється у послідовність байтів, після чого формується початковий матричний блок. На його основі виконується перетворення з використанням поворотної матриці, що дозволяє отримати перший зашифрований фрагмент і забезпечити випадковий розподіл даних на ранньому етапі.

Після формування першого блоку виконуються послідовні операції потокового шифрування. Вхідні дані розбиваються на фіксовані частини, для кожної з яких обчислюється гама залежно від поточного стану регістру зсуву. Особливістю алгоритму є те, що гамування відбувається без застосування доповнення та з динамічним оновленням параметрів на кожному кроці, що забезпечує високий рівень випадковості та захист від атак, заснованих на аналізі структури вихідних даних.

Паралельно з накладанням гами генерується MAC-код, який залежить як від зашифрованого вмісту, так і від поточного стану регістру. Це дозволяє інтегрувати контроль автентичності в сам процес зашифрування, без необхідності виконання окремої обчислювальної операції після завершення основних перетворень.

Результатом роботи модуля зашифрування є сформований шифротекст, що складається із зашифрованого первинного блоку, потокового зашифрованого представлення повідомлення та MAC-коду. Модуль розшифрування реалізує зворотну послідовність операцій. Узагальнений алгоритм роботи функції розшифрування наведено на рис. 3.3.

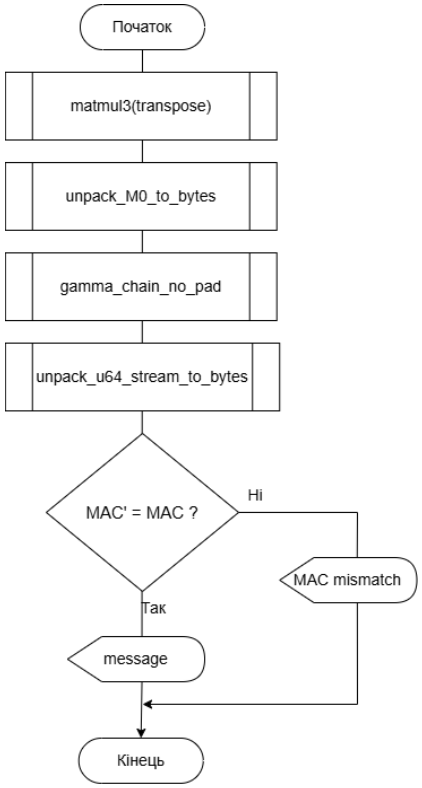


Рисунок 3.3 – Узагальнений алгоритм роботи функції decrypt

На першому етапі обробляється початковий блок, який відновлюється шляхом застосування транспонованої поворотної матриці. Це дозволяє отримати вихідні дані, які використовуються для подальшого відтворення структури повідомлення.

Далі здійснюється потокове розшифрування, що повністю дублює логіку зашифрування: регістр зсуву ініціалізується тим самим ключовим матеріалом, поступово оновлюється та використовується для генерування гами, яка накладається на шифротекст для відновлення вихідних байтів. Під час розшифрування паралельно формується нове значення MAC-коду, що обчислюється за тим самим принципом, що й під час зашифрування.

Після завершення обробки даних отримане значення порівнюється з MAC, що міститься в повідомленні. У випадку відповідності розшифровані дані вважаються автентичними і повертаються користувачеві. Якщо ж значення не збігаються, система повідомляє про порушення цілісності або про спробу підміни даних, що робить неможливим подальше використання результату.

Таким чином, розроблений алгоритм реалізує повний цикл зашифрування та розшифрування даних із вбудованою перевіркою їхньої цілісності. Використання матричних перетворень, потокового режиму обробки та регістру зсуву дозволяє поєднати високу швидкість функціонування з криптографічною стійкістю.

Інтегрований MAC-код гарантує, що будь-яке спотворення або підміна даних буде виявлена, а система не перейде до реконструкції повідомлення, що забезпечує додатковий рівень захисту у процесі обміну інформацією.

3.4 Програмна реалізація

Після розробки методу автентифікованого шифрування, який ґрунтується на поєднанні матричного шифрування на основі алгебри кватерніонів та шифрування гамуванням, було створено його програмну реалізацію у вигляді модуля matrix_stream_cipher.py. Модуль реалізує послідовне перетворення

текстового повідомлення у зашифрований вигляд з використанням матричних операцій та потокової гами без додаткових блоків вирівнювання.

Розробка програмного забезпечення починається з оголошення базових констант, які визначають математичні межі обчислень. У даному модулі використовується простір залишків за простим числом $p = 65537$, що забезпечує коректну роботу операцій у полі модульної арифметики. Для зручності додатково визначено маски для обмеження результатів до 16- та 64-бітових значень.

Такі обмеження особливо важливі для запобігання переповненню при множенні чисел великої розрядності та гарантують відтворюваність результатів незалежно від апаратної архітектури. Окремо задається 64-бітний поліном зворотного зв'язку LFSR_POLY_64, який використовується для роботи регістра зсуву та формування двох потоків керуючих значень F і K.

Одним із ключових етапів алгоритму є формування початкової матриці $\mathbf{M}_0$, яка складається з перших 18 байтів вхідного повідомлення. Це дозволяє отримати початкове значення для подальших обчислень. Реалізація цієї частини подана у функції pack_M0_from_text:

```
def pack_M0_from_text(msg_utf8: bytes) -> Tuple[List[List[int]], bytes, bytes]:
    first = msg_utf8[:18]
    if len(first) < 18:
        first = first + b"\x00" * (18 - len(first))
    rest = msg_utf8[18:]
    words = [int.from_bytes(first[i:i+2], "little") % P for i in range(0, 18, 2)]
    M0 = [
        [words[0], words[1], words[2]],
        [words[3], words[4], words[5]],
        [words[6], words[7], words[8]],
    ]
    return M0, rest, first
```

Функція виконує читання перших 18 байтів повідомлення, при потребі доповнюючи їх нулями до повного розміру. Далі дані перетворюються у дев'ять

16-бітових чисел, які утворюють матрицю $\mathbf{M_0}$ розміром 3×3. Таке представлення дозволяє застосувати подальше матричне множення для створення першого блоку шифротексту. Решта байтів повідомлення (rest) зберігається для шифрування гамуванням.

Зворотне перетворення для розшифрування зашифрованого повідомлення здійснює функція unpack_M0_to_bytes, яка переводить матрицю назад у байти. Це забезпечує можливість повного відновлення оригінальних даних під час розшифрування.

Після створення матриці $\mathbf{M_0}$ решта повідомлення обробляється у вигляді послідовності 64-бітових слів. На цьому етапі не застосовується стандартний паддінг – замість цього останній неповний блок просто доповнюється нулями всередині. Це забезпечує збереження структури даних та зменшує обсяг надлишкової інформації. Функція, що виконує цю операцію, має вигляд:

```
def pack_u64_stream_no_pad(data: bytes) -> List[int]:
    if not data:
        return []
    res: List[int] = []
    for i in range(0, len(data), 8):
        chunk = data[i:i+8]
        if len(chunk) < 8:
            chunk += b"\x00" * (8 - len(chunk))
        res.append(int.from_bytes(chunk, "little"))
    return res
```

У результаті формується список 64-бітових чисел, готових для подальшої обробки у режимі шифрування гамуванням. Цей підхід усуває потребу у спеціальних схемах вирівнювання, характерних для класичних блокових алгоритмів.

Для реалізації матричної частини шифрування використано дві допоміжні функції – matmul3 та transpose3, які виконують множення та транспонування 3×3 матриць у полі за модулем p:

```
def matmul3(A: List[List[int]], B: List[List[int]]) -> List[List[int]]:
    C = [[0,0,0],[0,0,0],[0,0,0]]
```

```
for i in range(3):
    for j in range(3):
        s = 0
        for k in range(3):
            s = (s + (A[i][k] % P) * (B[k][j] % P)) % P
        C[i][j] = s
return C
```

Ця функція виконує матричне множення з поелементним зведенням до модуля p. Завдяки цьому забезпечується стабільність результатів і запобігається переповненню при великих значеннях елементів. Обернена операція, тобто відновлення початкової матриці під час розшифрування, реалізована функцією transpose3, оскільки ортогональна матриця $\mathbf{\Gamma}(\hat{q})$ має властивість, за якої її обернена матриця збігається з транспонованою.

У частині шифрування гамуванням ключову роль відіграє гама, яка будується на основі 64-бітних операцій множення з використанням декількох потоків керуючих значень. Базовою функцією є _f64, що перемножує 64-бітові аргументи, розбиває добуток на старшу та молодшу частини і поєднує їх, забезпечуючи сильну дифузію бітів.

На основі _f64 реалізовано більш складну функцію _f64_triple, яка обчислює нове значення гами з урахуванням поточного слова повідомлення та станів двох незалежних LFSR-потоків *F* і *K*. Таким чином, кожне оновлення гами залежить одразу від попереднього значення, даних повідомлення та внутрішнього стану регістрів зсуву, що істотно ускладнює криптоаналіз.

Формування повної гами та MAC-ланцюга здійснює функція gamma_chain_no_pad. Вона ініціалізує початкове значення g_0 на основі ключа K_0, обробляє слова матриці $\mathbf{M}_0$ і решту повідомлення, паралельно генеруючи послідовність g-значень і фінальний MAC-код. Додатково використовується випадкова сіль, яка передається у вигляді рядка base64 і підмішується у випадку вироджених станів, що запобігає нульовим чи передбачуваним значенням гами. Ця функція також керує LFSR-потоками *F* і *K*, завдяки чому MAC-код і гама

формуються за однаковими правилами як на етапі побудови ланцюга, так і під час потокового шифрування.

Основна функція encrypt реалізує алгоритм зашифрування у повному циклі і в оновленій версії використовує два LFSR-потоки для формування гами та MAC-коду:

```
def encrypt(text: str, K0: int, Gamma: List[List[int]]) -> Ciphertext:
    """
    One-shot encryption with NO padding in gamma.
    """
    msg = text.encode("utf-8")
    M0, rest, m0_bytes_18 = pack_M0_from_text(msg)
    C0 = matmul3(Gamma, M0)
    mac_salt_b64 = generate_salt(8)
    g_final, g_trace = gamma_chain_no_pad(K0, m0_bytes_18, rest,
mac_salt_b64)
    ...
```

Функція спочатку формує матрицю $\mathbf{M_0}$ з перших 18 байтів повідомлення та обчислює матричний блок C_0 множенням на ортогональну матрицю $\mathbf{\Gamma}(\hat{q})$. Далі генерується випадкова сіль для MAC-колу і викликається gamma_chain_no_pad, яка формує повний MAC-код і повертає фінальне значення g_final та трасу проміжних g-значень. Одне з цих проміжних значень (g_prev) використовується як стартова гама для потокового шифрування «хвоста» повідомлення.

Потім решта байтів пакується у 64-бітові слова, а два регістри зсуву F і K ініціалізуються від ключа K_0 та попередньо «прокручуються» на кількість слів, що відповідає матриці $\mathbf{M_0}$. Це необхідно для того, щоб стан LFSR-потоків під час потокового шифрування повністю співпадав з станом, який був використаний у gamma_chain_no_pad під час обчислення MAC.

Далі виконується цикл потокового шифрування: кожне слово x поєднується з поточним значенням гами g_prev через операцію XOR, формуючи елемент шифротексту. Після цього обидва регістри зсуву переходять у наступний стан, і викликається _f64_triple, яка, використовуючи x, попередню гаму та поточні значення F і K, обчислює нове значення гами для наступного

кроку. Завдяки цьому гама залежить не тільки від ключа, але й від реального вмісту повідомлення та станів двох незалежних LFSR-потоків. Підсумковий MAC-код зберігається як молодші 64 біти від g_final і разом з шифротекстом та службовими метаданими повертається у вигляді об’єкта Ciphertext.

Зворотна процедура розшифрування забезпечує повне відновлення вихідного тексту та перевірку автентичності даних. Її реалізовано у функції decrypt:

```
def decrypt(ct: Ciphertext, K0: int, Gamma: List[List[int]]) -> str:
    """
    Decrypt and verify MAC according to the same no-pad, multiplicative
rules.
    """
    # Recover M0 (18 bytes) from C0
    M0_dec = matmul3(transpose3(Gamma), ct.C0)  # Γ^{-1} = Γ^T
(orthogonal)
    m0_bytes_18 = unpack_M0_to_bytes(M0_dec)


    salt_b64 = ct.meta.get("mac_salt_b64") or None
    g_after_m0, trace_m0 = gamma_chain_no_pad(K0, m0_bytes_18, b"",
salt_b64)
    g_prev = trace_m0[-1]
    ...
```

На початковому етапі функція відновлює матрицю $\mathbf{M_0}$ через множення транспонованої поворотної матриці $\mathbf{\Gamma}(\hat{q})^{\mathrm{T}}$на $\mathbf{C_0}$ і перетворення результату назад у 18 байтів. Далі, використовуючи ті самі параметри K_0 і сіль, викликається gamma_chain_no_pad, але лише для початкового блоку; це дозволяє відтворити правильне стартове значення гами після $\mathbf{M_0}$. Регістри зсуву F та K знову ініціалізуються від K_0 і прокручуються на ту ж кількість кроків, що і під час зашифрування.

Після цього виконується цикл зворотного потокового перетворення: для кожного елемента шифротексту відновлюється слово x шляхом XOR із поточним значенням гами, а потім, аналогічно до шифрування, оновлюються стани LFSR-потоків та нове значення гами за допомогою _f64_triple. Отримані 64-бітові слова перетворюються назад у байти. На завершальному етапі для всієї послідовності

(блок $\mathbf{M}_0$ плюс решта байтів) знову обчислюється MAC-ланцюг, і отримане значення порівнюється з MAC-кодом, що зберігається у структурі Ciphertext. Якщо значення не збігаються, викидається виняток про помилку автентифікації; у разі успіху відновлений байтовий потік обрізається до початкової довжини повідомлення і декодується у текст.

Отже, розроблений програмний модуль реалізує повний цикл роботи методу автентифікованого шифрування, поєднуючи матричні перетворення, двопоточне LFSR-кероване гамування та MAC-перевірку в єдину цілісну структуру.

Такий підхід забезпечує модульність і прозорість логіки обчислень, усуває необхідність застосування додаткових схем вирівнювання та підвищує загальну ефективність алгоритму завдяки використанню 64-бітових операцій.

Впровадження механізму контролю автентичності за допомогою MAC, який залежить не лише від повідомлення, а й від внутрішнього стану LFSR-потоків, гарантує виявлення будь-яких спотворень даних і підвищує криптографічну стійкість системи загалом.

3.5 Тестування програмного засобу

3.5.1 Тестування коректності роботи програмного засобу

Метою тестування є перевірка правильності роботи модуля автентифікованого шифрування, а саме – забезпечення коректного перетворення початкового повідомлення у зашифрований вигляд і відновлення його після розшифрування.

У процесі тестування етапи генерування та ущільнення ключа були опущені, поворотна ортогональна матриця $\mathbf{\Gamma}(\hat{q})$ уже отримана на основі спільного секрету з другої частини роботи. Таким чином, випробування зосереджується виключно на функціональності методу автентифікованого шифрування, описаного у другому розділі роботи.

Першим кроком виконується ініціалізація тестового середовища. Програма імпортує необхідні модулі та очікує на введення повідомлення. Вхідне повідомлення передається до функцій encrypt() і decrypt(), що реалізують відповідно процеси зашифрування та розшифрування даних.

Після запуску програма проводить обробку вхідного тексту. На початковому етапі відбувається перетворення перших байтів повідомлення у базову матрицю $\mathbf{M}_0$, яка є початковим блоком шифрування. Далі ця матриця множиться на поворотну матрицю $\mathbf{\Gamma}(\hat{q})$ за модулем $p = 65537$, у результаті чого формується перший блок шифротексту $\mathbf{C}_0$, як показано на рисунку 3.4.

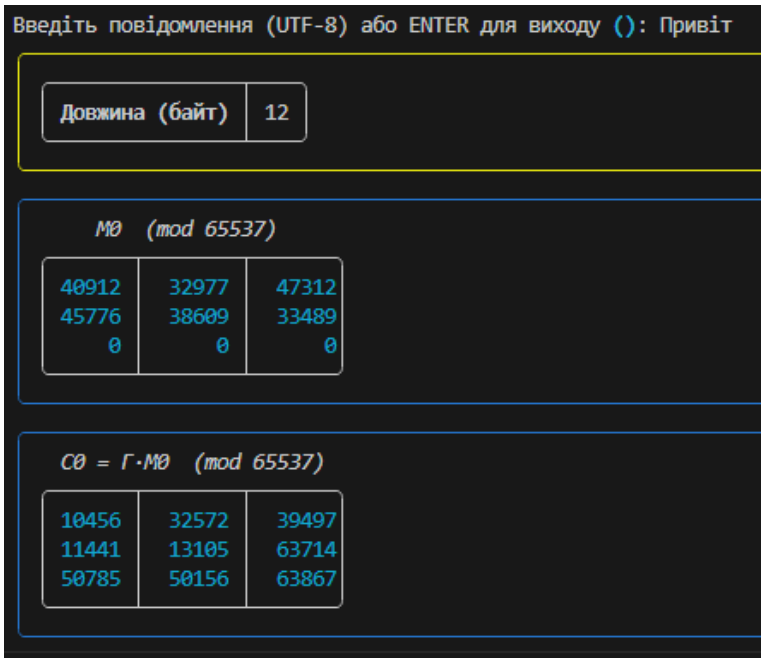


Рисунок 3.4 – Перший блок шифротексту

Наступним кроком програма переходить до шифрування гамуванням решти повідомлення. Для цього формується гама. На основі кожного наступного значення гами виконується побітова операція XOR між потоком гами та байтами повідомлення, утворюючи послідовність зашифрованих блоків. Процес шифрування гамуванням наведено на рисунку 3.5.

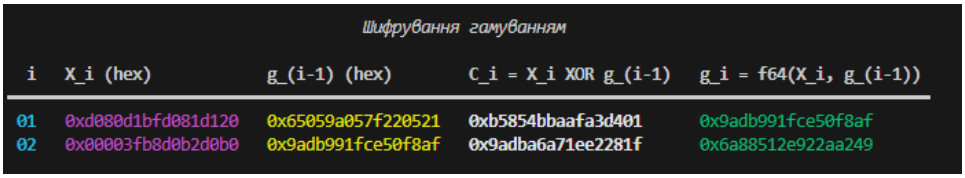


Рисунок 3.5 – Процес шифрування гамуванням

Після завершення шифрування гамуванням генерується контрольне значення MAC, що показано на рис. 3.6., яке використовується для перевірки автентичності під час розшифрування.

```
MAC = final g  =  0x6a88512e922aa249  (7676474825315623497)
```

Рисунок 3.6 – Обчислене значення MAC-коду

Це значення додається до шифротексту, формуючи кінцевий результат, що складається з трьох компонентів: початкового блоку $\mathbf{C}_0$, шифротексту та MAC-коду.

Під час розшифрування програма виконує зворотні дії. Отриманий перший блок $\mathbf{C}_0$ множиться на транспоновану матрицю $\mathbf{\Gamma}^{\mathrm{T}}(\hat{q})$, завдяки чому відновлюється початковий блок $\mathbf{M}_0$. Потім для решти повідомлення формується аналогічна послідовність гами, а побітова операція XOR повертає вихідний текст.

Результати розшифрування та порівняння MAC наведені на рисунку 3.7.

MAC порівняння

Сторона	MAC (hex)	MAC (dec)
Alice	0x6a88512e922aa249	7676474825315623497
Bob	0x6a88512e922aa249	7676474825315623497

```
Оригінал:Привіт, як справи?
Розкрито:Привіт, як справи?
```

Рисунок 3.7 – Результат розшифрування та порівняння MAC-кодів

У результаті тестування підтверджено коректність усіх етапів роботи програмного модуля.

3.5.2 Тестування стійкості програмного засобу

Для оцінювання стійкості методу автентифікованого шифрування було проведено статистичне тестування випадковості вихідних бітових послідовностей згідно стандарту NIST SP 800-22 Rev.1 – A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications [27].

Метою дослідження було встановити, чи демонструє сформований шифротекст та гамма-послідовність ознаки криптографічної випадковості, необхідні для стійкості проти статистичних атак.

В якості вхідних даних використовувались 100 незалежних бінарних послідовностей, сформованих алгоритмом шифрування. Для кожної послідовності було виконано повний набір статистичних тестів, а результати оцінювалися за двома критеріями:

– P-value ≥ 0.01, що свідчить про відсутність статистично значущих відхилень від випадковості;

– Прохідність тесту (Proportion) – не менше ніж 96% для вибірки зі 100 послідовностей (або 67% для тестів Random Excursions Variant, де кількість вибірок = 71).

Фрагмент зведеної таблиці результатів наведено на рис. 3.8.

RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES

generator is <nist_input.bin>

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
19	8	7	13	10	8	7	15	7	6	0.055361	96/100	Frequency
7	14	5	10	10	14	11	15	7	7	0.275709	99/100	BlockFrequency
14	13	9	8	9	16	7	6	9	9	0.401199	99/100	CumulativeSums
13	11	9	14	12	4	13	9	4	11	0.249284	97/100	CumulativeSums
14	9	10	9	8	7	10	8	15	10	0.739918	99/100	Runs
8	13	11	13	7	8	10	7	11	12	0.834308	100/100	LongestRun
7	15	14	10	5	12	7	7	10	13	0.304126	100/100	Rank
9	11	13	9	9	11	15	9	9	5	0.678686	100/100	FFT
8	6	11	11	9	14	10	9	15	7	0.595549	100/100	NonOverlappingTemplate
12	9	10	17	11	6	9	8	10	8	0.534146	99/100	NonOverlappingTemplate
11	11	16	9	5	11	9	9	7	12	0.534146	100/100	NonOverlappingTemplate
11	11	8	8	7	6	13	8	13	15	0.514124	99/100	NonOverlappingTemplate
10	9	12	8	12	9	9	11	7	13	0.946308	100/100	NonOverlappingTemplate
7	13	11	11	13	10	8	8	8	11	0.897763	99/100	NonOverlappingTemplate
5	17	9	7	10	8	10	12	8	14	0.262249	99/100	NonOverlappingTemplate
10	13	9	10	12	9	6	11	12	8	0.911413	99/100	NonOverlappingTemplate
9	12	9	11	13	12	5	10	5	14	0.474986	99/100	NonOverlappingTemplate
15	6	11	11	6	11	13	10	11	6	0.474986	99/100	NonOverlappingTemplate
11	10	6	12	9	8	9	13	13	9	0.867692	97/100	NonOverlappingTemplate
8	8	9	5	10	17	16	10	11	6	0.137282	98/100	NonOverlappingTemplate

Рисунок 3.8 – Фрагмент зведеної таблиці результатів

Тест частотності (Frequency) продемонстрував успішне проходження при значенні пропорції 96/100, що відповідає мінімальному порогу NIST для довірчого інтервалу. Це показує відсутність глобального дисбалансу між кількістю нулів і одиниць.

Блоковий тест частотності (Block Frequency), Cumulative Sums, Runs, Longest Run, Rank, FFT та більшість інших базових тестів показали пропорцію від 97% до 100%, що свідчить про відсутність періодичності, залежностей та лінійності [27].

Тест Non-overlapping Template Matching, що є чутливим до локальних структур та повторюваних шаблонів, у понад 100 викликах продемонстрував пропорцію у межах 97–100%, що вказує на відсутність домінуючих підрядків у вихідній послідовності.

Тести Random Excursions та Random Excursions Variant, які оцінюють поведінку стохастичного процесу в термінах відхилень від нульового стану Марковського ланцюга, показали пропорцію 70/71, 69/71, або вищу, що перевищує мінімально прийнятні значення 67/71. Навіть тест із найменшим результатом (69/71) знаходиться у межах допустимого статистичного інтервалу.

Serial test та Linear Complexity test продемонстрували пропорцію 98–100%, що свідчить про відсутність коротких детермінованих структур, які могли б бути використані при лінійному аналізі.

Найбільш важливим є те, що відсутня системна тенденція до падіння пропорції, а всі p-value знаходяться значно вище критичного рівня $\alpha = 0.01$, що свідчить про стабільність поведінки алгоритму [27].

Згідно зі стандартом, “мінімальна пропорція проходження тестів” для 100 послідовностей має бути не менше 96%, а для тестів групи Random Excursions Variant – не менше 67%. Усі тести виконані в межах цих вимог:

– середній показник пропорції для основних тестів: від 97% до 100%

– мінімальні значення не виходять за межі допустимого інтервалу

– відсутні критичні p-value, що свідчили б про статистичне відхилення

Отже, результати комплексного тестування демонструють, що сформований шифротекст та гама-послідовність мають високий рівень криптографічної випадковості, не містять статистично значущих аномалій, та не проявляють структур, які могли б бути використані для атаки.

Отримані результати дають підстави вважати шифротекст інформаційно-прихованим і непридатним для відновлення початкових даних без знання секретного ключа [27].

3.6 Висновки з розділу

Отже, було виконано програмну реалізацію методу автентифікованого шифрування та проведено його практичне тестування. Реалізацію здійснено мовою Python у середовищі Visual Studio Code, що забезпечило високу швидкість розробки, зручність налагодження, роботу з віртуальними середовищами та гнучку модульну структуру коду.

Побудована архітектура програмного засобу передбачає чіткий поділ на модуль зашифрування та модуль розшифрування, що дозволяє підтримувати симетричність операцій і забезпечує прозорість реалізації.

Обидва модулі об’єднані спільним механізмом керування ключем, генерацією гами та формуванням MAC-коду, при цьому всі операції виконуються в оперативній пам’яті без використання постійних сховищ, а внутрішні стани регістрів зсуву не передаються каналом зв’язку.

У процесі програмної реалізації було створено механізм перетворення перших байтів повідомлення у матрицю $\mathbf{M}_0$ та виконано матричне множення з ортогональною поворотною матрицею $\mathbf{\Gamma}(\hat{q})$, що формує перший блок шифротексту й забезпечує сильну початкову дифузію.

Подальша обробка повідомлення реалізована у вигляді шифрування гамуванням без класичного паддингу: байти пакуються у 64-бітові слова, а гама формується послідовно на основі 64-бітних операцій множення, нелінійної функції f_{64} та двох LFSR-потоків керуючих значень F і K.

У межах цього ж процесу формується MAC-код, який залежить не лише від вмісту повідомлення, а й від внутрішнього стану регістрів зсуву, що забезпечує вбудований механізм перевірки автентичності та розширює простір можливих станів системи.

Проведене функціональне тестування підтвердило коректність усіх етапів роботи програмного модуля: правильне формування блоку $\mathbf{C}_0$, відтворення початкової матриці $\mathbf{M}_0$, коректну роботу потокового шифрування та розшифрування, а також збіг обчисленого MAC-коду з переданим значенням.

Додатково було виконано статистичне тестування стійкості згідно зі стандартом NIST SP 800-22 Rev.1 для 100 бінарних послідовностей, сформованих алгоритмом. Усі основні тести продемонстрували припустимі значення p-value (не нижче 0,01) та пропорції проходження на рівні не менше 96 %, а тести групи Random Excursions Variant — вище 67 %, що свідчить про відсутність статистично значущих аномалій та належний рівень криптографічної випадковості гама-послідовності та шифротексту.

Таким чином, програмна реалізація повністю підтвердила працездатність розробленого методу автентифікованого шифрування та його ефективність у практичних умовах.

Алгоритм демонструє можливість обробки повідомлень довільної довжини без застосування додаткових схем вирівнювання, з використанням 64-бітових операцій, із збереженням високого рівня захисту даних, контролю цілісності та автентичності завдяки інтегрованому MAC-коду та LFSR-керованому механізму гамування.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Оскільки результати МКР «Програмний засіб захищеного передавання інформації» можуть бути комерціалізовані та інтегровані в існуючі програмні рішення, необхідно провести комерційний аудит розробки.

Його мета – оцінити науково-технічний потенціал створеного засобу, визначити доцільність його виходу на ринок та підготувати економічне обґрунтування для інвестора.

У оцінюванні науково-технічного рівня і комерційного потенціалу розробки взяли участь три експерти: Бабачук О. Д., директор ТОВ «ДАТАДЕФЕНДР»; Малахов В. В., заступник директора ТОВ «ДАТАДЕФЕНДР» та Атаманюк О. В. – фінансовий директор ТОВ «ДАТАДЕФЕНДР».

Оцінки було проставлено відповідно до рекомендованих критеріїв, зображених у додатку Б. Результати оцінювання зображено в табл. 4.1 [28].

Середньоарифметична сума балів розраховується за формулою:

$$СБ = \frac{\sum_{i=1}^{3} СБ_i}{3} \quad (4.1)$$

Таблиця 4.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки.

Критерії	Бали		
	Бабачук О. Д.	Малахов В. В.	Атаманюк О. В.
1. Технічна здійсненність концепції	3	3	3
2. Ринкові переваги (наявність аналогів)	2	2	2
3. Ринкові переваги (ціна продукту)	4	4	4
4. Ринкові переваги (технічні властивості)	3	4	3
5. Ринкові переваги (експлуатаційні витрати)	3	4	4
6. Ринкові перспективи (розмір ринку)	4	4	4

Продовження табл. 4.1

Критерії	Бали		
7. Ринкові перспективи (конкуренція)	2	2	2
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	4	4	4
Сума балів	40	42	41
Середньоарифметична сума балів $СБ_c$	41		

Отже, результати розрахунків показують, що науково-технічний рівень і комерційний потенціал розробки є високими (табл. 4.2). Це зумовлено поєднанням підвищеної захищеності та оптимізованих обчислень: поворотна матриця на основі кватерніонів забезпечує сильну дифузію вже в першому блоці, а суміщення формування гами та MAC-коду скорочує додаткові етапи обробки. Такий підхід покращує баланс між швидкодією та стійкістю до підробки [28].

Впровадження може потребувати інтеграційних робіт і сертифікації, однак ці витрати є помірними й легко оптимізуються в існуючих інфраструктурах.

Таблиця 4.2 – Науково-технічні рівні та комерційні потенціали розробки

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

Також необхідно оцінити конкурентоспроможність розробки. Вона визначається за якісними та економічними показниками і передбачає проведення кількох етапів аналізу. Порівняння виконуватиметься зі стандартним режимом автентифікованого шифрування AES-GCM, який широко застосовується в сучасних системах захищеної передачі даних та протоколах транспортного рівня [28].

4.1.1 Розрахунок одиничних параметричних індексів

Якщо збільшення величини параметра свідчить про підвищення якості нової розробки, одиничний параметричний індекс розраховується за формулою:

$$q_i = \frac{P_i}{P_{базі}}, \quad (4.2)$$

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

P_i – значення i-го параметра розробки;

$P_{базі}$ – аналогічний параметр базової розробки-аналога, з якою проводиться порівняння.

Технічні та економічні параметри аналога та нової науково-технічної розробки варто подати у таблиці 4.3.

Таблиця 4.3 – Технічні та економічні параметри аналога нової науково-технічної розробки

Параметр	Одиниця виміру	Аналог	Нова розробка	Індекс зміни значення параметра	Коефіцієнт вагомості
Технічні					
Продуктивність	5-бальна шкала	4	5	1,25	0,25
Сумісність	5-бальна шкала	5	4	0,8	0,05
Функціональність	5-бальна шкала	5	5	1	0,05
Безпека	5-бальна шкала	5	5	1	0,25
Масштабованість	5-бальна шкала	4	5	1,25	0,4
Економічні					
Вартість інтеграції	грн	9000	7000	0,78	0,3
Вартість підтримки	грн	4500	3500	0,78	0,3
Час впровадження	години	12	9	0,75	0,4

4.1.2 Розрахунок групових параметричних індексів

Значення групового параметричного індексу за технічними параметрами визначається з урахуванням вагомості (частки) кожного параметра за формулою:

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

де $I_{\text{ТП}}$ – груповий параметричний індекс за технічними показниками (порівняно з аналогом);

q_i – одиничний параметричний показник *i*-го параметра;

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

n – кількість технічних параметрів, за якими оцінюється конкурентоспроможність.

Тобто розрахунок групового параметричного індексу за технічними параметрами матиме такий вигляд:

$$I_{\text{ТП}} = 1{,}25 \cdot 0{,}25 + 0{,}8 \cdot 0{,}05 + 1 \cdot 0{,}05 + 1 \cdot 0{,}25 + 1{,}25 \cdot 0{,}4 = 1{,}15.$$

Груповий параметричний індекс за економічними параметрами (за ціною споживання) розраховується за формулою:

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

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

q_i – економічний параметр *i*-го виду;

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

m – кількість економічних параметрів, за якими здійснюється оцінювання.

Тобто розрахунок групового параметричного індексу за економічними параметрами матиме вигляд:

$$I_{\text{ЕП}} = 0{,}78 \cdot 0{,}3 + 0{,}78 \cdot 0{,}3 + 0{,}75 \cdot 0{,}4 = 0{,}768.$$

4.1.3 Розрахунок інтегрального показника

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розраховують інтегральний показник конкурентоспроможності за формулою:

$$K_{\text{ІНТ}} = I_{\text{НП}} \cdot \frac{I_{\text{ТП}}}{I_{\text{ЕП}}}, \tag{4.5}$$

$$K_{\text{ІНТ}} = 1 \cdot \frac{1{,}15}{0{,}768} = 1{,}5.$$

На основі інтегрального показника ($K_{\text{ІНТ}} = 1{,}5$) можна зробити висновок, що розробка є перспективною та може бути виведена на ринок. Таке значення досягнуте завдяки вищим технічним показникам та суттєво нижчим економічним витратам порівняно з аналогами [28].

4.2 Розрахунок витрат на здійснення науково-дослідної роботи

Витрати на здійснення науково-дослідної роботи групуються за такими статтями:

- витрати на оплату праці;
- відрахування на соціальні заходи;
- матеріали;
- паливо та енергія для науково-виробничих цілей;
- програмне забезпечення для наукових (експериментальних) робіт;
- інші витрати;
- накладні (загальновиробничі) витрати.

Варто розглянути кожну статтю окремо [28].

4.2.1 Витрати на оплату праці

Для початку, варто розрахувати основну заробітну плату дослідників. Для цього необхідно використати формулу:

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

де k – кількість посад дослідників, залучених до процесу досліджень;

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

t_i – кількість днів роботи конкретного дослідника, дн.;

T_p – середня кількість робочих днів в місяці, T_p = 22 дні.

Для зручності варто винести результати розрахунків у таблицю 4.4 [28].

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Кількість днів роботи	Витрати на заробітну плату, грн
Керівник проекту	25000	1136,36	5	5681,8
Криптограф	20000	909,09	7	6363,63
Розробник програмного модулю	20000	909,09	15	13636,35
Тестувальник	18000	818,18	5	4090,9
Всього				29772,68

Основна заробітна плата робітників розраховується за формулою:

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

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

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

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

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{ЗМ}}, \quad (4.8)$$

де M_M – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), грн – наразі M_M = 8000,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати [28].

T_p – середня кількість робочих днів в місяці, приблизно T_p = 22 дні;

$t_{\text{ЗМ}}$ – тривалість зміни, год.

$$C_i = \frac{8000 \cdot 1{,}1 \cdot 1{,}8}{22 \cdot 8} = 90 \text{ грн.},$$
$$З_{\text{р}} = 90 \cdot 14 = 1260 \text{ грн.}$$

Для зручності всі витрати варто винести в таблицю 4.5.

Таблиця 4.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд Роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Дослідження наявних методів автентифікованого шифрування	14	2	1,1	90	1260
Розробка методу автентифікованого шифрування	25	7	2,2	180	4500
Розробка архітектури програмного застосунку	4	3	1,35	110,45	441,8
Побудова алгоритмів виконання програмного застосунку	8	3	1,35	110,45	883,63
Програмна реалізація застосунку	50	6	2,0	163,63	8181,5
Оптимізація застосунку	39	5	1,7	139,09	5424,51
Тестування застосунку	40	3	1,35	101,25	4045

Продовження табл. 4.5

Найменування робіт	Тривалість роботи, год	Розряд Роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Тестування криптографічної стійкості алгоритму	17	6	2,0	163,63	2781,71
Виправлення виявлених помилок	18	3	1,35	110,45	1998,1
Збірка застосунку	1	1	1,0	81,81	81,81
Всього					29598,06

Додаткова заробітна плата робітників розраховується за формулою:

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

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Нехай це буде 12%.

$$З_{дод} = (29772,68 + 29598,06) \cdot \frac{12\%}{100\%} = 7124,48.$$

4.2.2 Відрахування на соціальні заходи

До статті «Відрахування на соціальні заходи» належать відрахування внеску на загальнообов’язкове державне соціальне страхування та для здійснення заходів щодо соціального захисту населення (ЄСВ – єдиний соціальний внесок) [28].

Нарахування на заробітну плату розраховується як 22% від суми основної та додаткової заробітної плати за формулою:

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

де $Н_{зп}$ – норма нарахування на заробітну плату, тобто $Н_{зп}$ = 22%.

$$З_н = (29772{,}68 + 29598{,}06 + 7124{,}48) \cdot \frac{22\%}{100\%} = 14628{,}88.$$

4.2.3 Сировина та матеріали

Всі процеси відбуваються у електронному форматі, працівники мають власні інструменти для досліджень та розробки, та все ж є необхідність у записниках [28].

Витрати на матеріали у вартісному вираженні розраховуються окремо для кожного виду матеріалів за формулою:

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

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

n – кількість видів матеріалів;

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

K_j – коефіцієнт транспортних витрат, (K_j= 1,1);

B_j – маса відходів j-го найменування, кг;

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

Результати всіх розрахунків зображено у табл. 4.6.

Таблиця 4.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 шт., грн.	Норма витрат, шт	Величина відходів, шт	Ціна відходів, грн/шт	Вартість витраченого матеріалу, грн
Блокнот формату А5	54	4	0	0	237,6
Ручка кулькова, чорна	22	4	0	0	96,8
Всього					334,4

Отже, загальна кількість витрат на матеріали – 334,4 грн.

4.2.4 Програмне забезпечення для наукових (експериментальних) робіт

Балансова вартість програмного забезпечення розраховується за формулою:

$$В_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_j, \quad (4.12)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу цього виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_j – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_j = 1{,}1$);

k – кількість найменувань програмних засобів.

Отримані результати винесені у таблицю 4.7 [28].

Таблиця 4.7 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
IDE VS Code (Extensions)	2	350	770,0
Trello Standard	4	207,5	899,8
Postman Enterprise	1	2033,5	2236,85
Всього			3906,65

Отже, всього витрати на програмне забезпечення становлять 3906,65 грн.

4.2.5 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо можуть бути розраховані з використанням прямолінійного методу амортизації за формулою:

$$А_{обл} = \frac{Ц_б}{Т_в} \cdot \frac{t_{вик}}{12}, \quad (4.13)$$

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

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

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

Результати розрахунків варто винести в таблицю 4.8 [28].

Таблиця 4.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Кількість, шт	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Приміщення	1	100000	20	1	416,66
Ноутбук Asus	4	11000	2	1	1833,33
	Всього				2249,99

Отже, сума амортизації – 2249,99 грн.

4.2.6 Палива та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($В_e$) можна розрахувати за формулою:

$$В_e = \sum_{i=1}^{n} W_{yi} \cdot t_i \cdot Ц_e \cdot \frac{K_{вэ}}{\eta_i}, \quad (4.14)$$

де W_{yi} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн, $Ц_e$ = 10,5 грн;

$K_{вэ}$ – коефіцієнт, що враховує використання потужності, $K_{вэ} < 1$, $K_{вэ} = 0{,}38$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$, $\eta_i = 0{,}9$.

Результати проведених розрахунків занесено до таблиці 4.9.

Таблиця 4.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Ноутбук Asus №1	0,025	40	4,33
Ноутбук Asus №2	0,04	56	9,93
Ноутбук Asus №3	0,045	120	23,94
Ноутбук Asus №4	0,050	40	8,86
Всього			47,06

Отже, витрати на електроенергію становлять 47,06 грн.

4.2.7 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками [28].

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_{\text{в}} = (З_{\text{о}} + З_{\text{р}}) \cdot \frac{H_{\text{ів}}}{100\%},$$
$$I_{\text{в}} = (29772{,}68 + 29598{,}06) \cdot \frac{50\%}{100\%} = 29685{,}37. \quad (4.15)$$

Отже, інші витрати становлять 29685,37 грн.

4.2.8 Накладні (загальновиробничі) витрати

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

$$В_{\text{нзв}} = (З_{\text{о}} + З_{\text{р}}) \cdot \frac{H_{\text{зв}}}{100\%}, \quad (4.16)$$

де $H_{\text{зв}}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$В_{нзв} = (29772{,}68 + 29598{,}06) \cdot \frac{100\%}{100\%} = 59370{,}74.$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$\begin{gathered} В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + \\ В_{св} + В_{сп} + І_в + В_{нзв}, \\ В_{заг} = 29772{,}68 + 29598{,}06 + 7124{,}48 + 14628{,}88 + 334{,}4 + \\ 3906{,}65 + 2249{,}99 + 47{,}06 + 29685{,}37 + 59370{,}74 = 176718{,}31. \end{gathered} \quad (4.17)$$

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (4.18)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи. Оскільки наукова робота знаходиться на стадії розробки промислового зразка, то $\eta = 0{,}7$.

$$ЗВ = \frac{176718{,}31}{0{,}7} = 252454{,}73.$$

Отже, загальновиробничі витрати становлять $252454{,}73$ грн.

4.3 Розрахунок економічної ефективності науково-технічної розробки від її впровадження безпосередньо замовником

У цій МКР розроблено спеціалізоване програмне забезпечення для захищеного передавання інформації, яке може бути впроваджене в інфраструктуру підприємства [28]. Можливе збільшення чистого прибутку

підприємства ΔПі для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою:

$$\Delta П_i = (\pm \Delta П_я \cdot N + П_я \cdot \Delta N_i), \quad (4.19)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження на підприємстві результатів науково-технічної розробки в аналізованому році;

N – основний кількісний показник, який визначає обсяг діяльності підприємства у році до впровадження результатів нової науково-технічної розробки;

$П_я$ – основний якісний показник, який визначає результати діяльності підприємства у кожному із років після впровадження науково-технічної розробки;

ΔN – зміна основного кількісного показника діяльності підприємства в результаті впровадження науково-технічної розробки в аналізованому році.

$$\Delta П_1 = 0{,}2 \cdot 800 + 2{,}5 \cdot 50 = 285 \text{ тис. грн.}$$
$$\Delta П_2 = \Delta П_3 = \Delta П_4 = \Delta П_5 = 0{,}12 \cdot 800 + 2{,}5 \cdot 50 = 223{,}4 \text{ тис. грн.}$$

Далі потрібно розрахувати приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати замовник від можливого впровадження науково-технічної розробки на власному підприємстві [28]. Розрахунок відбувається за формулою:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.20)$$

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

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

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0{,}1$;

t – період часу (в роках) від моменту початку впровадження науковотехнічної розробки до моменту отримання підприємством збільшеної величини чистого прибутку в аналізованому році.

$$\text{ПП} = \frac{285}{(1+0{,}1)^1} + \frac{223{,}4}{(1+0{,}1)^2} + \frac{223{,}4}{(1+0{,}1)^3} + \frac{223{,}4}{(1+0{,}1)^4} + \frac{223{,}4}{(1+0{,}1)^5} = 902{,}83 \text{ тис. грн.}$$

Далі потрібно розрахувати величину початкових інвестицій PV, які замовник має вкласти для здійснення науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{\text{розр}} \cdot \text{ЗВ}, \tag{4.21}$$

де розр k – коефіцієнт, що враховує витрати розробника (замовника) на впровадження науково-технічної розробки. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{\text{розр}} = 2 \ldots 5$, але може бути і більшим. У випадку даного дослідження та розробки $k_{\text{розр}} = 2$;

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

$$PV = 2 \cdot 252454{,}73 = 504909{,}46 \text{ грн.}$$

Абсолютний економічний ефект $E_{\text{абс}}$ або чистий приведений дохід (NPV, Net Present Value) для розробника (замовника) від можливого впровадження науково-технічної розробки можна розрахувати за формулою:

$$E_{\text{абс}} = \text{ПП} - PV, \tag{4.22}$$

де ПП – приведена вартість збільшення всіх чистих прибутків від можливого впровадження науково-технічної розробки, грн;

PV – теперішня вартість початкових інвестицій, грн.

$$E_{\text{абс}} = 902{,}83 - 504{,}909 = 397{,}920 \text{ тис. грн.}$$

Для остаточного прийняття рішення необхідно розрахувати внутрішню економічну дохідність $E_{\text{в}}$ або показник внутрішньої норми дохідності (IRR, Internal Rate of Return) вкладених замовником коштів [28]. Внутрішня економічна дохідність інвестицій $E_{\text{в}}$, які можуть бути вкладені замовником у впровадження науково-технічної розробки, розраховується за формулою:

$$E_{\text{в}} = \sqrt[\text{Т}_{\text{ж}}]{1 + \frac{E_{\text{абс}}}{PV}} - 1, \tag{4.23}$$

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

PV – теперішня вартість початкових інвестицій, грн;

$\text{Т}_{\text{ж}}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

$$E_{\text{в}} = \sqrt[1]{1 + \frac{397{,}920}{504{,}909}} - 1 = 0{,}78.$$

Далі розраховується період окупності інвестицій $T_{ок}$ (DPP, Discounted Payback Period), які можуть бути вкладені розробником (замовником) у впровадження та комерціалізацію науково-технічної розробки. Розрахунок відбувається за формулою:

$$T_{ок} = \frac{1}{E_в}, \tag{4.24}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = \frac{1}{0,78} = 1,28.$$

Оскільки $T_{ок} < 3$ років, можна зробити висновок, що впровадження науково-технічної розробки замовником є економічно ефективним [28].

4.4 Висновки до розділу

У межах економічної частини було комплексно проаналізовано науково-технічний рівень, комерційний потенціал та економічну доцільність впровадження розробленого програмного засобу захищеного передавання інформації.

Оцінювання, проведене трьома експертами ТОВ «ДАТАДЕФЕНДР» за визначеними критеріями, показало середній бал 41, що свідчить про високий рівень технологічної готовності та значні перспективи подальшої комерціалізації.

Порівняльний аналіз конкурентоспроможності продемонстрував, що запропонований метод автентифікованого шифрування має вагомі технічні переваги, зокрема кращу масштабованість та продуктивність, а також нижчі витрати впровадлення та підтримки порівняно зі стандартним AES-GCM.

Отриманий інтегральний показник $K_{\text{ІНТ}} = 1{,}5$ підтверджує, що розробка є конкурентоспроможною на ринку та може бути успішно інтегрована в існуючі системи захищеного зв’язку.

Детальний розрахунок витрат на виконання науково-дослідної роботи дав змогу визначити загальну собівартість розробки – 252,45 тис. грн, а оцінка економічної ефективності впровадження довела її фінансову доцільність.

Розрахована приведена вартість очікуваного приросту чистого прибутку підприємства становить 902,83 тис. грн, що забезпечує позитивний абсолютний економічний ефект у розмірі 397,92 тис. грн.

Особливо важливим показником є період окупності інвестицій, який склав лише $T_{\text{ок}} = 1{,}28$ року. Такий результат свідчить про високий рівень привабливості проєкту для потенційних інвесторів, оскільки інвестиції повертаються значно швидше за типовий нормативний трирічний період.

Крім того, внутрішня норма дохідності $E_{\text{в}} = 0{,}78$ перевищує середні ринкові показники, що додатково підтверджує економічну вигідність впровадлення.

Узагальнюючи результати розрахунків та аналізу, можна стверджувати, що розробка програмного засобу захищеного передавання інформації є технічно обґрунтованою, економічно ефективною, конкурентоспроможною та комерційно перспективною. Її впровадження підвищує рівень інформаційної безпеки підприємства та забезпечує позитивну фінансову віддачу в короткі строки.

ВИСНОВКИ

У процесі виконання комплексної магістерської кваліфікаційної роботи було досягнуто поставленої мети шляхом виконання всіх запланованих дослідницьких, аналітичних, математичних та програмних завдань . У роботі було розроблено метод автентифікованого шифрування, який поєднує матричне перетворення на основі кватерніонів із потоковим шифруванням та механізмом формування MAC-коду, а також підтверджено його коректність, стійкість та можливість практичного застосування у сучасних системах захищеного передавання інформації.

Для досягнення поставленої мети було проведено аналіз існуючих методів автентифікованого шифрування та популярних режимів блокових шифрів, що їх реалізують. Виконаний огляд дозволив встановити, що більшість відомих AE/AEAD-алгоритмів створені для універсального застосування й не орієнтовані на специфічні сценарії, пов’язані з великими обсягами даних або з вимогами мінімізації обчислювальних витрат.

Як показав аналіз, жоден із класичних методів не забезпечує оптимального поєднання сильної дифузії, низької складності та інтегрованого механізму автентифікації, що підтвердило необхідність розроблення нового комбінованого рішення.

Для створення методу було виконано ґрунтовний математичний опис його складових, включно з побудовою ортогональної матриці на основі кватерніонів та формалізацією потокового механізму генерування гами. Такий підхід дозволив визначити ключові властивості майбутнього шифру, сформувати модель даних та визначити операції, які забезпечують сильну початкову дифузію та однорідність процесу шифрування.

Розроблений метод має дворівневу структуру обробки даних: перший блок повідомлення перетворюється за допомогою ортогональної поворотної матриці $\mathbf{\Gamma}(\hat{q})$, тоді як усі наступні блоки шифруються потоковим методом із використанням нелінійного генератора гами та паралельним формуванням

MAC-коду. Завдяки використанню ортогональної поворотної матриці для першого блоку і нелінійного генератора гами для наступних блоків, забезпечується подвійний рівень дифузії й цілісності, що робить розроблений метод придатним для практичного використання у сучасних системах захищеного зв’язку, зокрема у криптопротоколах реального часу, месенджерах та IoT-пристроях.

Після завершення математичного опису метод було реалізовано програмно мовою Python. Побудована архітектура забезпечує модульність, гнучкість та можливість масштабування. Було створено окремі компоненти для формування матриці $\mathbf{\Gamma}(\hat{q})$, обчислення гами, виконання операцій шифрування та розшифрування, а також модуль для формування MAC-коду.

Після програмної реалізації було проведено тестування коректності роботи. Застосування тестів показало правильність побудови матриці, стабільність генератора гами та відсутність помилок у реалізації зворотних операцій. Після цього виконано тестування криптографічної стійкості за стандартом NIST SP 800-22. Результати статистичних тестів підтвердили відсутність закономірностей у шифротекстах та дозволили вважати вихідні послідовності статистично випадковими.

Оцінка продуктивності розробленого методу продемонструвала його придатність для практичного використання. Отримані результати підтвердили, що комбінована структура дає перевагу у швидкодії порівняно з класичними AE/AEAD-рішеннями при збереженні високого рівня криптографічної стійкості. Метод виявився особливо ефективним для сценаріїв, де важливо мінімізувати обчислювальні витрати та забезпечити швидку обробку даних – у мобільних застосунках, реальному часі та ресурсно обмежених системах.

Таким чином, у ході виконання МКР було:

– сформовано математичну модель та описано новий метод автентифікованого шифрування;

– виконано програмну реалізацію всіх компонентів методу;

– проведено тестування коректності, криптографічної стійкості та продуктивності;

– підтверджено ефективність та практичну придатність запропонованого методу.

Отже, у результаті виконання комплексної магістерської кваліфікаційної роботи було створено повноцінний метод автентифікованого шифрування та програмний засіб для його реалізації, доведено його ефективність, коректність і можливість використання у сучасних інформаційних системах.

Розроблений метод має потенціал для подальшого розвитку, зокрема у напрямках оптимізації генератора гами, підвищення продуктивності та інтеграції в складні криптографічні протоколи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Katz J., Lindell Y. Introduction to Modern Cryptography. Third Edition. Taylor and Francis, 2020. 648 p.

2. Rogaway P., Bellare M., Black J. OCB: A block-cipher mode of operation for efficient authenticated encryption. ACM Transactions on Information and System Security (TISSEC). 2003. Vol. 6, no. 3. P. 365-403. (accessed: 12.09.2024).

3. Boneh D., Shoup V., A Graduate Course in Applied Cryptography, 2020. 953 p. URL: https://scholar.google.com/citations?_op=view_citation&hl=en&user=H4cQsHAAAAAJ&cstart=20&pagesize=80&citation_for_view=H4cQsHAAAAAJ:XD-gHx7UXLsC (accessed: 15.09.2024).

4. Black J. Authenticated Encryption. Springer, Boston, MA, 2011.

5. Dzwonkowski M., Rykaczewski R. Quaternion Encryption Method for Image and Video Transmission. Article. September 2013.

6. Методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 125 «Кібербезпека та захист інформації» : освітньо-професійна програма «Безпека інформаційних і комунікаційних систем» / Уклад. : Л. М. Куперштейн, Ю. В. Баришев, В. А. Каплун. – Вінниця : ВНТУ, 2024. – 95 с.

7. Luzhetskyi V., Rohachevskyi D., Kozyra V. Authenticated encryption method. Poland, Krakow. International Scientific and Practical Conference DIGITAL TRANSFORMATION: STRENGTHENING THE CYBERSECURITY CAPACITIES IN THE MODERN WORLD, 2025. 6 p.

8. Лужецький В. А., Рогачевський Д.Б., Козира В.А. Метод захищеного передавання інформації. м. Вінниця / Молодь в науці: дослідження, проблеми, перспективи (МН-2026). Вінниця: ВНТУ 2025. 3 с. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/view/26535.

9. Bhuse V. Review of End-to-End Encryption for Social Media. URL: https://www.researchgate.net/publication/368905447_Review_of_End-to-End_Encryption_for_Social_Media (accessed: 20.09.2025).

10. Greenberg A. Hacker Lexicon: What Is End-to-End Encryption? URL: https://www.wired.com/2014/11/hacker-lexicon-end-to-end-encryption/ (accessed: 22.09.2025).

11. Schneir G. Authenticated Encryption: An Explainer. URL: https://www.ubiqsecurity.com/understanding-authenticated-encryption-an-explainer/ (accessed: 25.09.2025).

12. Block Cipher modes of Operation. URL: https://www.geeksforgeeks.org/ethical-hacking/block-cipher-modes-of-operation/ (accessed: 28.09.2025).

13. Green M. How to choose an Authenticated Encryption mode. URL: https://blog.cryptographyengineering.com/2012/05/19/how-to-choose-authenticated-encryption/ (accessed: 02.10.2025).

14. Svenda P. Basic comparison of modes for authenticated-encryption (IAPM, XCBC, OCB, CCM, EAX, CWC, GCM, PCFB, CS) / P. Svenda. – Brno : Masaryk University, Faculty of Informatics, 2005. – 13 p.

15. Jayasinghe R. What is AEAD (Authenticated Encryption with Associated Data)? URL: https://medium.com/@rushikajayasinghe/what-is-aead-authenticated-encryption-with-associated-data-17a5b2f42404 (accessed: 07.10.2025).

16. Zhang F. Quaternion and Matrices of Quaternions. Linear Algebra and its Applications. Elsevier Science Inc., 1997. P. 21–57.

17. Goldman R. Understanding Quaternions. Graphical Models. 2011. Vol. 73 (2). P. 21–49.

18. Rotation Matrix. URL: https://www.geeksforgeeks.org/maths/rotation-matrix/ (accessed: 12.10.2025).

19. Nonlinear Function. URL: https://www.sciencedirect.com/topics/mathematics/nonlinear-function (accessed: 19.10.2025).

20. How message authentication code works? URL: https://www.geeksforgeeks.org/computer-networks/how-message-authentication-code-works/ (accessed: 26.10.2025).

21. How to Multiply Matrices. URL: https://www.mathsisfun.com/algebra/matrix-multiplying.html (accessed: 03.11.2025).

22. Padding. URL: https://libsodium.gitbook.io/doc/padding (accessed: 10.11.2025).

23. Linear Feedback Shift Registers (LFSR). – URL: https://www.geeksforgeeks.org/digital-logic/linear-feedback-shift-registers-lfsr/ (accessed: 12.11.2025).

24. (Yet another) Introduction to quaternions. URL: https://lisyarus.github.io/blog/posts/introduction-to-quaternions.html (accessed: 15.11.2025)

25. 10 Advantages of Python Over Other Programming Languages. URL: https://www.geeksforgeeks.org/blogs/top-advantages-of-python/ (accessed: 24.11.2025).

26. Joseph E. PyCharm vs VS Code -- which is best for Python in 2024? URL: https://blog.openreplay.com/pycharm-vs-vscode--which-is-best-for-python-in-2024/ (accessed: 29.11.2025).

27. National Institute of Standards and Technology. NIST SP 800-22 Rev. 1. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. Gaithersburg: NIST. 2010. 131 p.

28. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТКИ

88

Додаток А. ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Програмний засіб захищеного передавання інформації. Частина 1. Метод автентифікованого шифрування.

Автор роботи: Рогачевський Дмитро Богданович
Тип роботи: магістерська кваліфікаційна робота
Підрозділ кафедра захисту інформації ФІТКІ , група І БС-24м

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism **1, 39** %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

√ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

В. о. зав. кафедри ЗІ д. т. н., проф. _____ Володимир ЛУЖЕЦЬКИЙ

Гарант освітньої програми «Безпека інформаційних і комунікаційних систем» к. т. н., доц.

_____ Олеся ВОЙТОВИЧ

Особа, відповідальна за перевірку _____ Валентина КАПЛУН

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Володимир ЛУЖЕЦЬКИЙ

Здобувач _____ Дмитро РОГАЧЕВСЬКИЙ

Додаток Б. КРИТЕРІЇ ОЦІНЮВАННЯ НАУКОВО-ТЕХНІЧНОГО РІВНЯ І КОМЕРЦІЙНОГО ПОТЕНЦІАЛУ РОЗРОБКИ ТА БАЛЬНА ОЦІНКА

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

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

ІЛЮСТРАТИВНА ЧАСТИНА

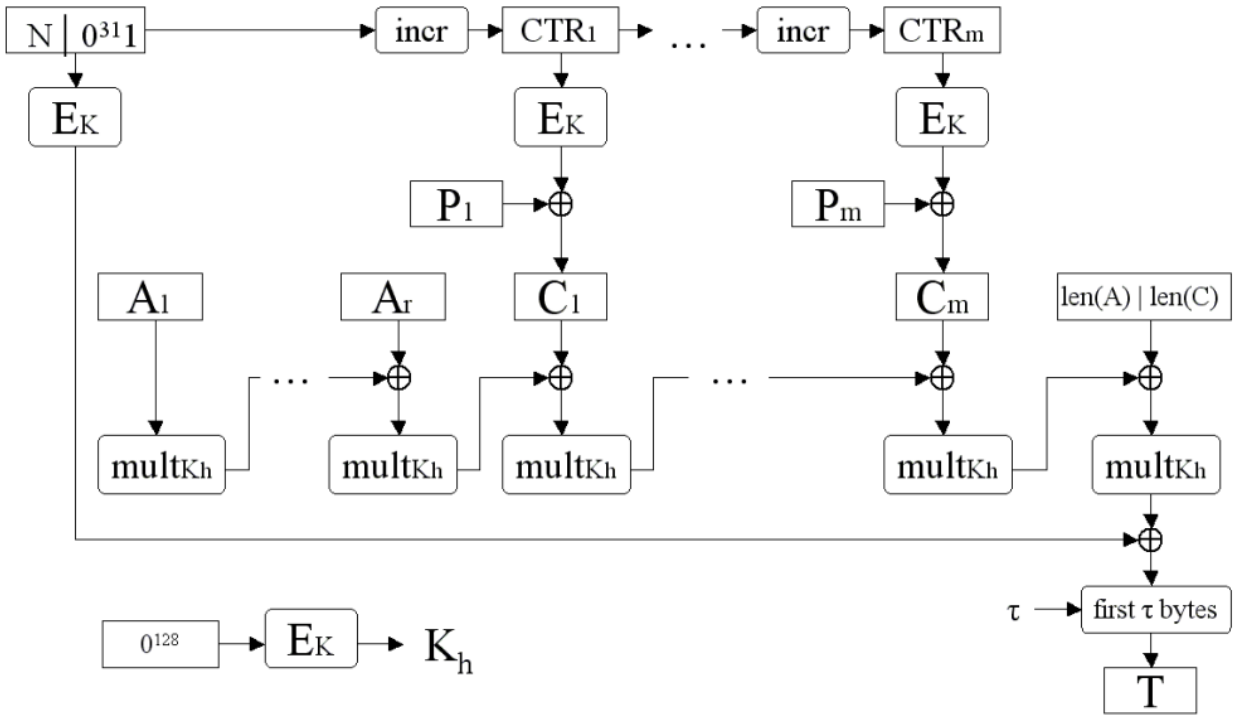
ПРОГРАМНИЙ ЗАСІБ ЗАХИЩЕНОГО ПЕРЕДАВАННЯ ІНФОРМАЦІЇ. ЧАСТИНА 1. МЕТОД АВТЕНТИФІКОВАНОГО ШИФРУВАННЯ

Виконав: студент 2 курсу групи 1БС-24м
спеціальності 125 Кібербезпека та захист інформації
_____Дмитро РОГАЧЕВСЬКИЙ

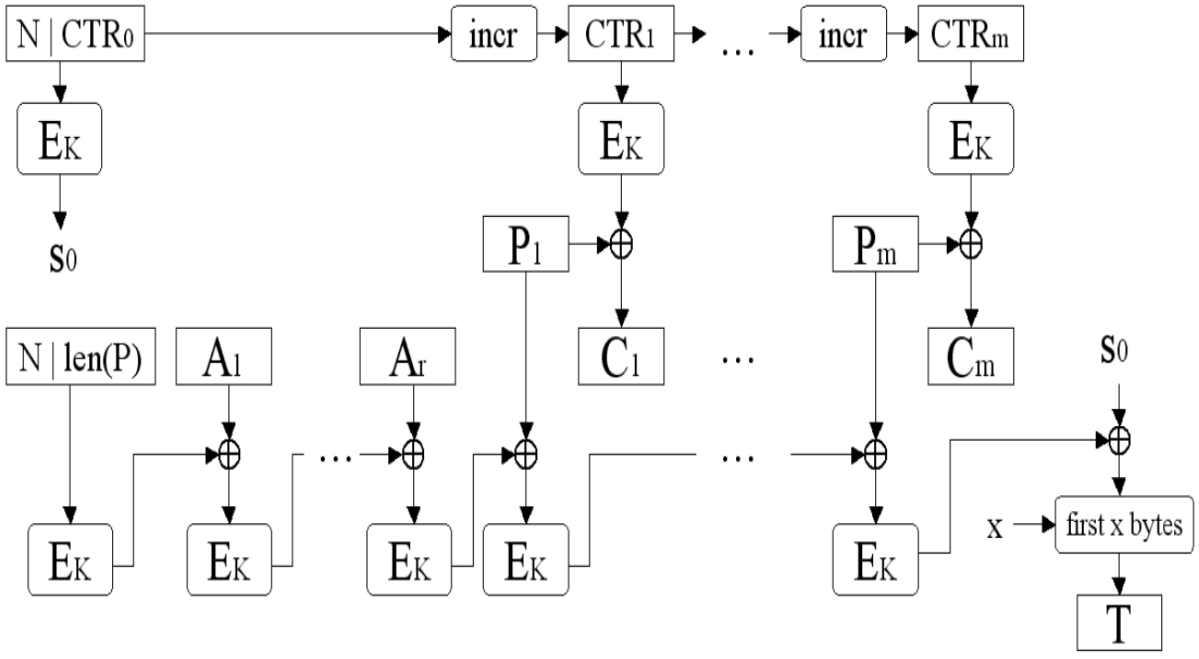
Керівник: В. о. зав. кафедри ЗІ, д. т. н., проф.
_____Володимир ЛУЖЕЦЬКИЙ
«16» 12 2025 р.

ВІДОМІ ПІДХОДИ ДО АВТЕНТИФІКОВАНОГО ШИФРУВАННЯ: GCM ТА CCM

GCM

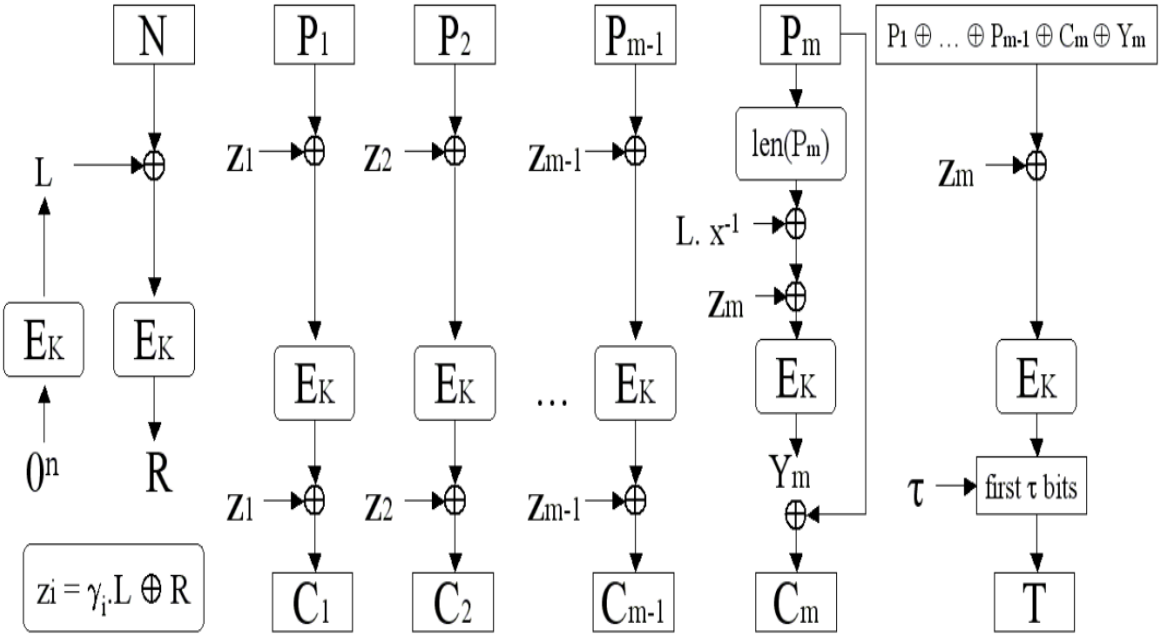


CCM



ВІДОМІ ПІДХОДИ ДО АВТЕНТИФІКОВАНОГО ШИФРУВАННЯ: OCB ТА EAX

OCB



EAX

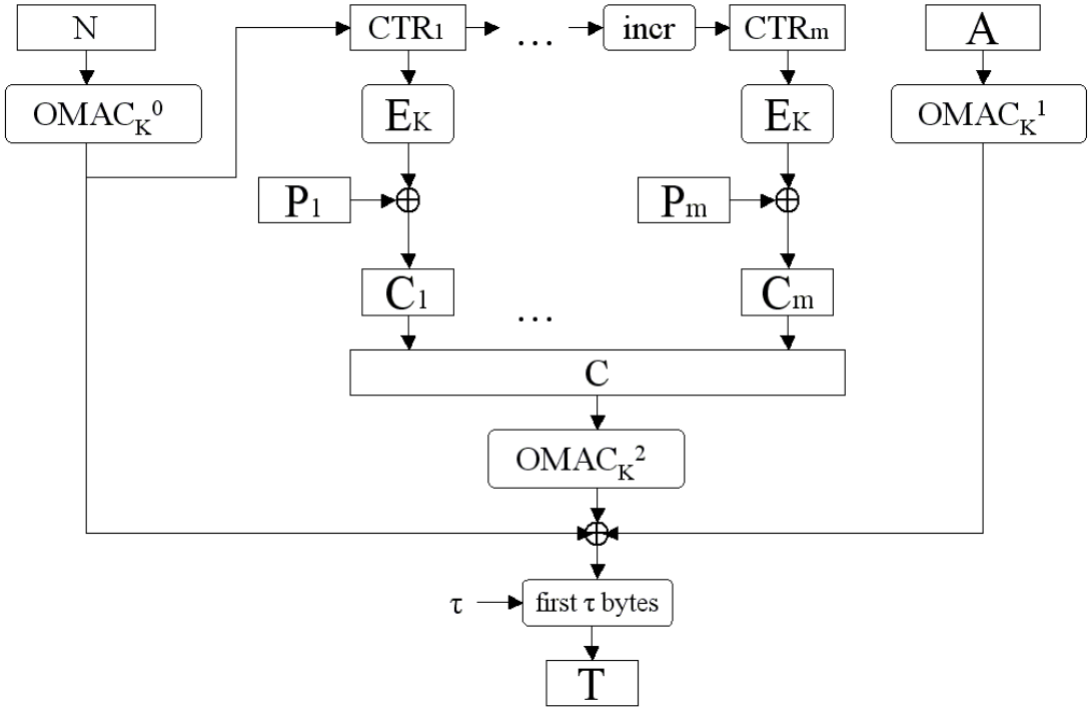
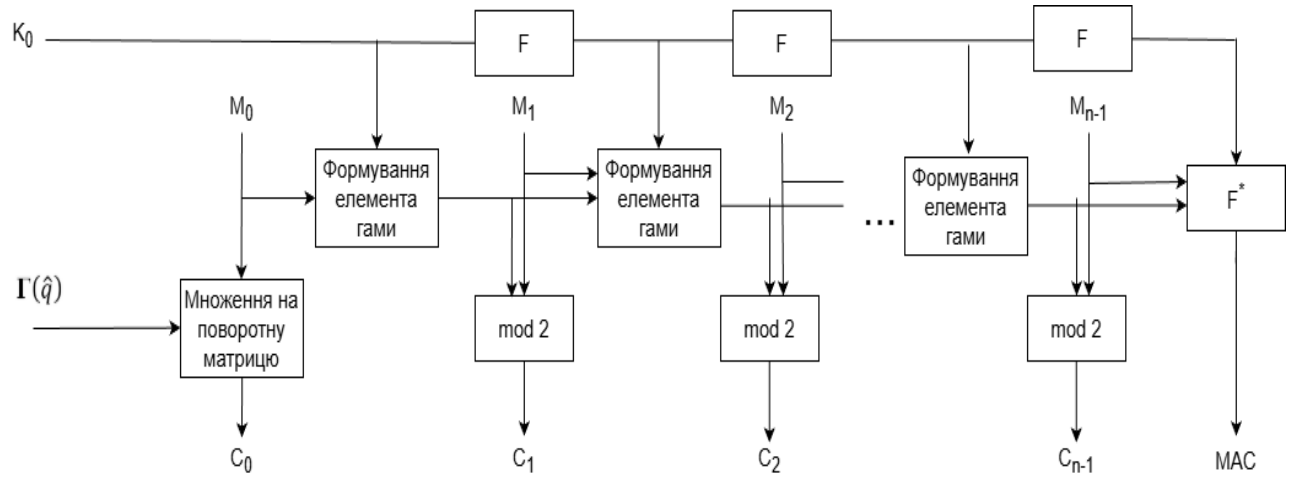


СХЕМА ПРОЦЕСУ ЗАШИФРУВАННЯ



$$\mathbf{\Gamma}(\hat{q}) = \begin{bmatrix} 1-2(y^2+z^2) & 2(xy-wz) & 2(xz+wy) \\ 2(xy+wz) & 1-2(x^2+z^2) & 2(yz-wx) \\ 2(xz-wy) & 2(yz+wx) & 1-2(x^2+y^2) \end{bmatrix}$$

$$\mathbf{M}_0 = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}, m_i \in [0, 2^{16})$$

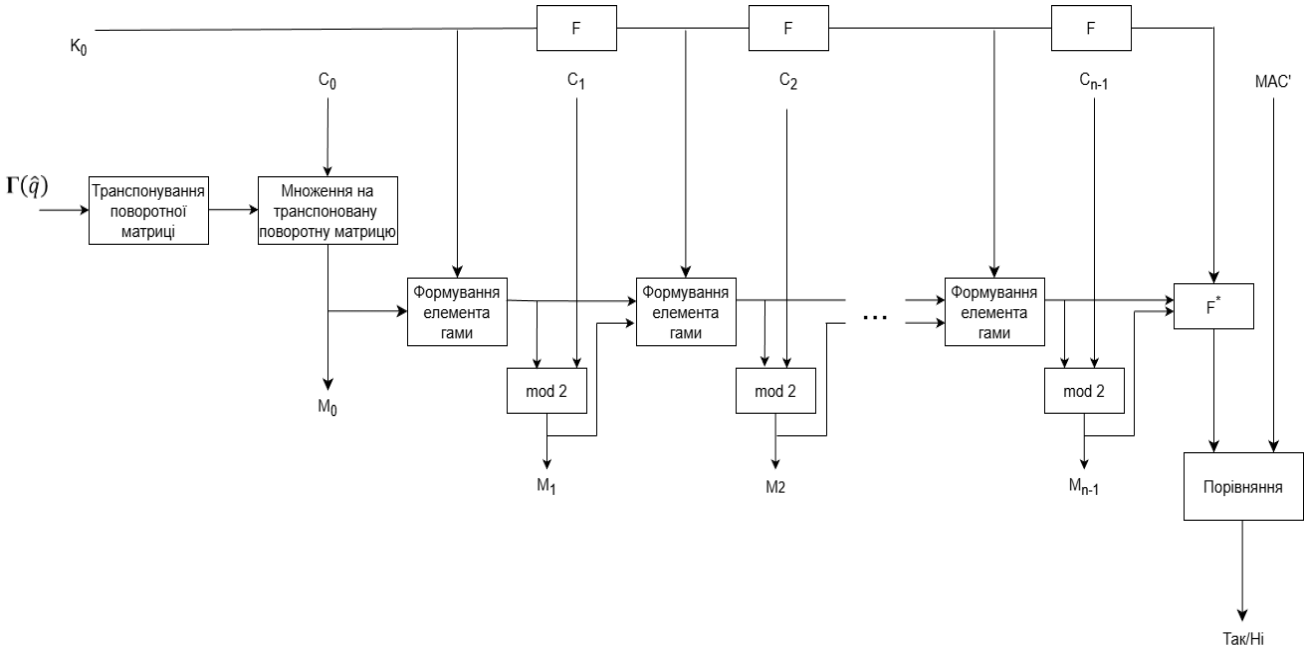
$$\mathbf{C}_0 = (\mathbf{\Gamma}(\hat{q}) \cdot \mathbf{M}_0)\ mod\ 65537$$

$$g_i = f_{64}(M_{i-1}, g_{i-1}, F_{i-1}) = (\Big(\big(((M_{i-1} \cdot g_{i-1})_L + (M_{i-1} \cdot g_{i-1})_R)\ mod\ 2^{64}\big) \cdot F_{i-1}\Big)_L + \Big(\big(((M_{i-1} \cdot g_{i-1})_L + (M_{i-1} \cdot g_{i-1})_R)\ mod\ 2^{64}\big) \cdot F_{i-1}\Big)_R)\ mod\ 2^{64}$$

$$C_i = M_i \oplus g_{i-1}$$

$$MAC = f_{64}(M_{n-1}, g_{n-1}, F_{n-1}) = ((((M_{n-1} \cdot g_{n-1})_L + (M_{n-1} \cdot g_{n-1})_R)\ mod\ 2^{64}) \cdot F_{n-1})_L + (((M_{n-1} \cdot g_{n-1})_L + (M_{n-1} \cdot g_{n-1})_R)\ mod\ 2^{64}) \cdot F_{i-1})_R)\ mod\ 2^{64}.$$

СХЕМА ПРОЦЕСУ РОЗШИФРУВАННЯ



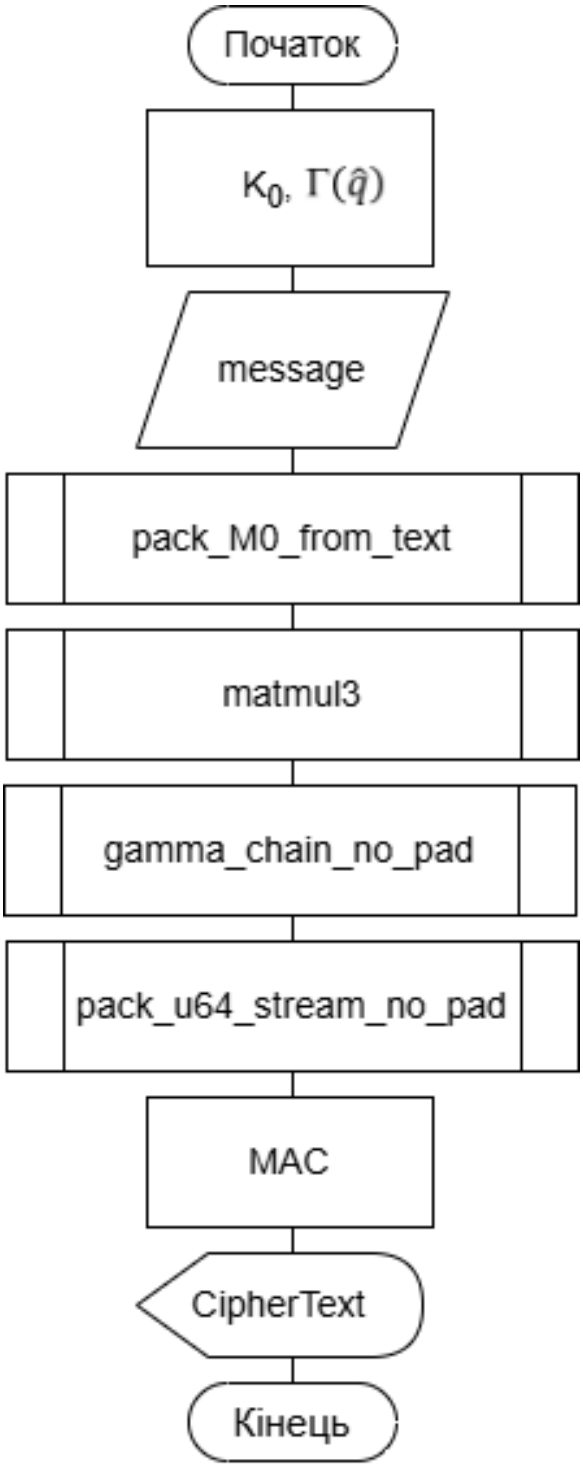
$$\mathbf{M}_0 = (\,\mathbf{\Gamma}^{\mathrm{T}}(\hat{q}) \cdot \mathbf{C}_0)\; mod\; 65537$$

$$M_i = C_i \oplus g_{i-1}$$

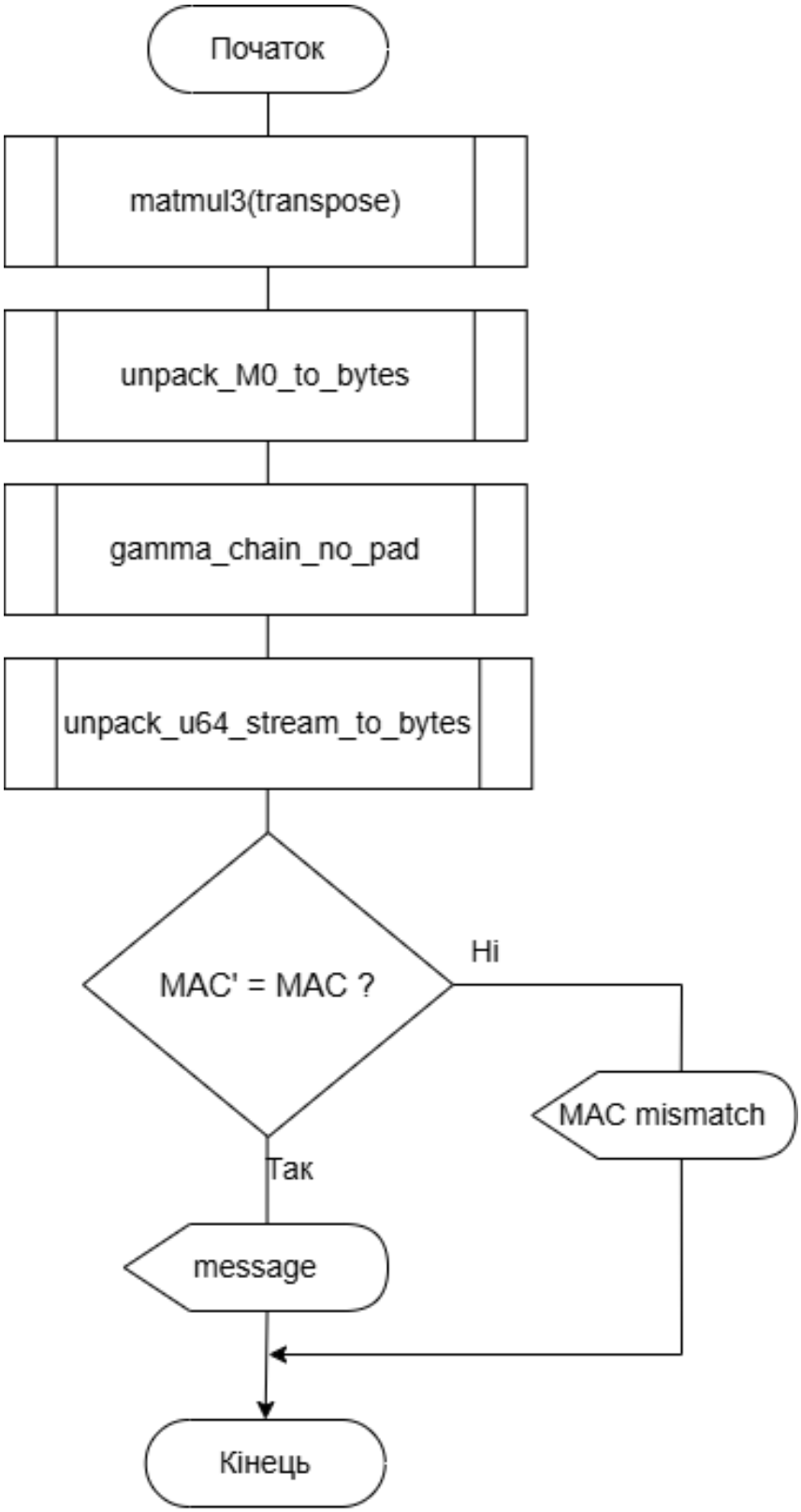
$$MAC' = f_{64}(M_{n-1}, g_{n-1}, F_{n-1})$$

УЗАГАЛЬНЕНА АРХІТЕКТУРА ПРОГРАМНОГО ЗАСОБУ ДЛЯ ЗАХИЩЕНОГО ПЕРЕДАВАННЯ ІНФОРМАЦІЇ

УЗАГАЛЬНЕНИЙ АЛГОРИТМ РОБОТИ ФУНКЦІЇ ENCRYPT



УЗАГАЛЬНЕНИЙ АЛГОРИТМ РОБОТИ ФУНКЦІЇ DECRYPT



ОЦІНКА СКЛАДНОСТІ РЕАЛІЗАЦІЇ ЗАПРОПОНОВАНОГО АЛГОРИТМУ

$$S_{\text{зашиф.}} = S_{\text{множ.}} + S_{\text{роз.кл.}} + S_{\text{гам.}} + S_{\text{наклад.}}$$

де: $S_{\text{множ.}}$ – складність множення матриць. $S_{\text{множ.}}$= 54 оп.;

$S_{\text{роз.кл.}}$ – складність розгортання ключа. $S_{\text{роз.кл.}}$= $3(n+1)$ оп.;

$S_{\text{гам.}}$ – складність формування елементів гами. $S_{\text{гам.}}$ = $8(n+1)$ оп.;

$S_{\text{наклад.}}$ – складність накладання елементів гами на блоки повідомлення, $S_{\text{наклад.}}$= $(n-1)$ оп..

Загальна кількість операцій дорівнюватиме:

$$S_{\text{зашиф.}} = (12n + 64)\text{оп}$$

ПОРІВНЯЛЬНІ ОЦІНКИ СКЛАДНОСТІ РЕАЛІЗАЦІЇ АЛГОРИТМІВ

Алгоритм	Кількість операцій на блок	Тип операцій
AES-GCM	≈ 150–180	128-бітні XOR, множення у $GF(2^{128})$, 10 раундів AES
AES-CCM	≈ 200–220	10 AES-раундів × 2 проходи
AES-EAX	≈ 190–210	AES-операції у двох проходах
AES-OCB	≈ 120–140	AES-раунди, XOR
Пропонований метод	≈ 12 операції/блок (після ініціалізації)	множення, LFSR, XOR, додавання mod 2^{64}

ПОКАЗНИКИ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ

Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки.

Критерії	Бали		
1. Технічна здійсненність концепції	3	3	3
2. Ринкові переваги (наявність аналогів)	2	2	2
3. Ринкові переваги (ціна продукту)	4	4	4
4. Ринкові переваги (технічні властивості)	3	4	3
5. Ринкові переваги (експлуатаційні витрати)	3	4	4
6. Ринкові перспективи (розмір ринку)	4	4	4
7. Ринкові перспективи (конкуренція)	2	2	2
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	4	4	4
Сума балів	40	42	41
Середньоарифметична сума балів $СБ_c$	41		

$$K_{\text{ІНТ}} = 1 \cdot \frac{1{,}15}{0{,}768} = 1{,}5$$

$$\text{ЗВ} = \frac{176718{,}31}{0{,}7} = 252454{,}73$$

$$T_{\text{ок}} = \frac{1}{0{,}78} = 1{,}28$$