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 2. ПРОТОКОЛ ОБМІНУ ІНФОРМАЦІЄЮ»

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

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

Рецензент: д. т. н., проф., зав. кафедри ПЗ
_____Олександр РОМАНЮК
«16» грудня 2025 р.

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. завідувача кафедри ЗІ,
д. т. н., проф.
Володимир ЛУЖЕЦЬКИЙ
«24» 09 2025 року

З А В Д А Н Н Я
НА КОМПЛЕКСНУ МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ
СТУДЕНТУ
Володимиру КОЗИРІ

1. Тема роботи: «Програмний засіб захищеного передавання інформації. Частина 2. Протокол обміну інформацією», 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:
   - Протокол захищеного обміну інформацією – клієнт-клієнт.
   - Протокол угоди про ключі – протокол Діффі-Геллмана.
   - Метод ущільнення ключа – на основі кватерніонів.
   - Метод нормування кватерніона.
   - Метод побудови поворотної матриці – на основі кватерніонів.
   - Розрядність елементів поворотної матриці – 16 біт.
4. Зміст текстової частини: Вступ. 1. Аналіз інформаційних джерел. 2. Розробка протоколу обміну інформацією 3. Програмна реалізація протоколу обміну інформацією. 4. Економічна частина. Висновки. Список використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу: схема отримання відкритого ключа користувачами, схема взаємної автентифікації сторін А та В, процедура формування секретного ключа, ущільнення секретного ключа, обчислення поворотної матриці, архітектура програмного засобу захищеного передавання інформації, алгоритм взаємної автентифікації та встановлення спільного сеансового ключа, алгоритм узгодження параметрів та перевірки ортогональності поворотної матриці, результати обміну повідомленнями між клієнтами, 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	18.12.25
2	Володимир ЛУЖЕЦЬКИЙ, д. т. н., проф., в. о. зав. кафедри ЗІ	25.09.2025	18.12.25
3	Володимир ЛУЖЕЦЬКИЙ, д. т. н., проф., в. о. зав. кафедри ЗІ	25.09.2025	18.12.25
4	Олександр ЛЕСЬКО, зав. каф. ЕПВМ, к. е. н., доц.	25.09.2025	18.12.25

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 протоколів обміну інформацією та моделей узгодження ключів	23.10.2025 – 26.10.2025	
5	Аналіз та формування вимог до протоколу обміну інформацією	27.10.2025 – 02.11.2025	
6	Розробка протоколу обміну інформацією та механізму взаємної автентифікації	03.11.2025 – 10.11.2025	
7	Програмна реалізація та тестування протоколу обміну інформацією	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 _____ Володимир КОЗИРА

Керівник комплексної магістерської кваліфікаційної роботи _____ Володимир ЛУЖЕЦЬКИЙ

АНОТАЦІЯ

УДК 004.056.55

Козира В. Програмний засіб захищеного передавання інформації. Частина 2. Протокол обміну інформацією. Комплексна магістерська кваліфікаційна робота зі спеціальності 125 – Кібербезпека та захист інформації, освітня програма – Безпека інформаційних і комунікаційних систем. Вінниця: ВНТУ, 2025. 98 с.

Укр. мовою. Бібліогр.: 29 назв; рис. 33; табл.: 17.

Комплексна магістерська кваліфікаційна робота присвячена розробці протоколу обміну інформацією та програмного засобу для встановлення автентифікованого захищеного з’єднання між користувачами. У роботі проведено аналіз сучасних протоколів обміну інформацією, моделей узгодження ключів і транспортних протоколів у системах захищеного передавання даних.

Запропоновано протокол клієнт-клієнт, у якому автентифікація сторін та узгодження сеансового ключа Діффі-Геллмана поєднані в єдиному криптографічному циклі з використанням цифрових підписів. Отриманий спільний секрет ущільнюється та використовується для формування ортогональної поворотної матриці на основі кватерніонів, що забезпечує зв’язок із методом автентифікованого шифрування першої частини комплексної магістерської кваліфікаційної роботи.

Розроблено програмний засіб, який реалізує запропонований протокол, проведено його тестування стійкості до атак типу MITM та виконано економічне обґрунтування доцільності впровадження.

Ілюстративна частина складається з 10 плакатів.

Ключові слова: протокол обміну інформацією, взаємна автентифікація, Діффі–Геллман, сеансовий ключ, кватерніони, кібербезпека, захищене передавання даних.

ABSTRACT

Kozyra V. Software tool for secure information transfer. Part 2. Information exchange protocol. Comprehensive master's thesis in the field of 125 – Cybersecurity and Information Protection, educational programme – Security of Information and Communication Systems. Vinnytsia: VNTU, 2025. 98 p.

In Ukrainian. Bibliography: 29 titles; figs. 33; tables: 17.

This comprehensive master's thesis is devoted to the development of an information exchange protocol and software tool for establishing an authenticated secure connection between users. The thesis analyses modern information exchange protocols, key agreement models, and transport protocols in secure data transmission systems.

A client-client protocol is proposed in which party authentication and Diffie-Hellman session key agreement are combined in a single cryptographic cycle using digital signatures. The resulting shared secret is compressed and used to form an orthogonal rotation matrix based on quaternions, which provides a link to the authenticated encryption method of the first part of the comprehensive master's qualification work.

A software tool has been developed that implements the proposed protocol, its resistance to MITM attacks has been tested, and an economic justification for its implementation has been provided.

The illustrative part consists of 10 posters.

Keywords: information exchange protocol, mutual authentication, Diffie–Hellman, session key, quaternions, cybersecurity, secure data transmission.

ЗМІСТ

ВСТУП

Актуальність дослідження зумовлена зростанням обсягів передавання даних і потребою в протоколах, що забезпечують конфіденційність, цілісність і автентичність під час встановлення захищеного зв’язку. Традиційне розділення автентифікації та узгодження ключів збільшує затримки й створює додаткові ризики.

Хоч сучасні протоколи – Діффі-Геллман, X3DH, Double Ratchet – гарантують високий рівень безпеки, вони часто є ресурсомісткими або потребують складної інфраструктури. Тому актуальною є розробка компактніших і швидкодіючих механізмів встановлення захищеного каналу.

Запропонований протокол поєднує автентифікацію та узгодження ключів Діффі-Геллмана в один процес, що зменшує затримки й підвищує стійкість до MITM-атак. Отриманий спільний секрет ущільнюється та використовується для побудови ортогональної поворотної матриці в методі автентифікованого шифрування з першої частини комплексної магістерської кваліфікаційної роботи, забезпечуючи комплексний захист даних.

Значний внесок у розвиток протоколів обміну ключами, потокових алгоритмів та механізмів автентифікації зробили такі вчені, як Вітфілд Діффі, Мартін Геллман, Тахер Ель-Гамаль, Даніел Дж. Бернштейн [1-3]

Об’єктом дослідження є процес встановлення автентифікованого зв’язку між користувачами програмного засобу захищеного передавання інформації.

Предметом дослідження є протокол автентифікації користувачів та протокол узгодження ключів Діффі-Геллмана.

Метою дослідження є пришвидшення процесу встановлення автентифікованого сеансу зв’язку між користувачами шляхом суміщення етапів автентифікації та узгодження параметрів Діффі-Геллмана в єдиному криптографічному протоколі [4].

Для досягнення поставленої мети необхідно виконати такі завдання:

– проаналізувати протоколи сучасних месенджерів та моделі узгодження ключів;

– розробити суміщений протокол автентифікації та узгодження ключів Діффі–Геллмана;

– розробити метод ущільнення секретного ключа;

– розробити метод побудови секретного ключа у вигляді ортогональної поворотної матриці;

– реалізувати програмний модуль, що забезпечує захищений обмін інформацією, та провести його тестування.

Наукова новизна роботи полягає у наступному:

– запропоновано протокол, у якому автентифікація користувачів та узгодження ключів Діффі-Геллмана виконуються в одному інформаційному циклі, що зменшує кількість повідомлень і підвищує стійкість до MITM-атак;

– удосконалено механізм формування сеансового ключа шляхом його ущільнення та подальшого використання в побудові ортогональної поворотної матриці;

Практична цінність роботи полягає в тому, що розроблено та програмно реалізовано протокол захищеного обміну інформацією з поєднаною автентифікацією користувачів і узгодженням сеансового ключа Діффі-Геллмана, який може бути інтегрований у програмні засоби захищеного передавання даних.

Основні результати комплексної магістерської кваліфікаційної роботи обговорювались на міжнародній науково-практичній конференції «DIGITAL TRANSFORMATION: STRENGTHENING THE CYBERSECURITY CAPACITIES IN THE MODERN WORLD» (Польща, Краків) [5] та міжнародній науково-практичній Інтернет-конференції «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ (МН-2026)» (Україна, Вінниця) [6].

1 АНАЛІЗ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1.1 Протоколи обміну інформацією в месенджерах

У сучасному цифровому середовищі месенджери стали одним із основних засобів комунікації – як для особистого, так і для професійного спілкування. Вони дозволяють миттєво обмінюватись текстовими повідомленнями, голосовими й відеодзвінками, файлами, фото, відео, геоданими тощо.

Серед месенджерів, що користуються популярністю серед людей особливе місце займають наступні:

– Telegram;

– WhatsApp;

– Signal;

– Viber;

– Facebook Messenger.

У різних сучасних месенджерах реалізовано власні протоколи обміну інформацією, які відрізняються між собою структурою узгодження параметрів, способом формування спільного секрету, механізмами побудови сеансових ключів та порядком автентифікації даних.

Хоча всі системи мають спільну мету – встановити захищений канал обміну даними між двома учасниками, порядок виконання дій, залучені проміжні ключі та механізми оновлення криптографічного стану відрізняються залежно від архітектури конкретного месенджера.

Ці відмінності є критичними, оскільки саме протокол обміну інформацією визначає рівень стійкості системи до атак типу MITM (Man-In-The-Middle) та здатність забезпечувати стійкість ключів при компрометації одного з сеансів.

В Telegram шифрування реалізовано на основі MTProto 2.0 (cloud chats). У такій моделі встановлення секрету виконується між клієнтом і сервером, а не напряму між двома абонентами. Сервер виконує роль активного елемента протоколу, забезпечуючи розподіл AuthKey, який обчислюється через процедуру

обміну параметрами з використанням елементів Діффі-Геллмана. Після цього весь трафік між клієнтом та сервером шифрується symm-key на основі AuthKey. Прямого end-to-end A→B ключового матеріалу Telegram у звичайних чатах не формує.

Послідовність інформаційних кроків у процесі встановлення захищеної cloud-сесії в Telegram представлено в табл. 1.1 [7].

Таблиця 1.1 – Основні етапи формування AuthKey у Telegram

Етап	Напрямок	Опис дії
1	A→Server	Надсилання запиту PQ
2	Server→A	Передача p,g та public параметрів
3	A→Server	Передача g^a mod p
4	Server→A	Передача g^b mod p
5	A,Server	Обчислення спільного AuthKey
6	A↔Server	Шифрування MTProto-повідомлень симетричним ключем

WhatsApp застосовує Signal Protocol, що включає використання pre-keys, які завчасно зберігаються на сервері. Завдяки цьому відправник може ініціювати встановлення спільного секрету навіть коли одержувач офлайн. Первинна фаза відповідає X3DH, після чого протокол переходить до Double Ratchet з постійною ротацією ключів для кожного повідомлення.

Послідовність інформаційних кроків для встановлення сеансового стану WhatsApp представлено в табл. 1.2 [8].

Таблиця 1.2 – Основні етапи X3DH/Double Ratchet у WhatsApp

Етап	Напрямок	Опис дії
1	B→Server	Публікація pre-keys
2	A→Server	Запит набору pre-keys
3	A↔B	Обчислення X3DH (identity + signed + one-time)
4	A,B	Формування Root Key
5	A,B	Старт Double Ratchet
6	A→B	Передача AEAD повідомлення
7	B	Перевірка MAC та оновлення Ratchet-стану

Signal визначає еталонну модель, яка складається з X3DH та Double Ratchet. X3DH забезпечує встановлення початкового спільного секрету навіть без синхронного контакту, а Double Ratchet забезпечує forward secrecy та post-compromise security шляхом ротації ключів для кожного повідомлення.

Основні етапи встановлення сеансу у Signal наведено у табл. 1.3 [9].

Таблиця 1.3 – Основні етапи встановлення сеансу у Signal

Етап	Напрямок	Опис дії
1	B→Server	Публікація pre-key bundle
2	A→Server	Отримання pre-key bundle
3	A↔B	Обчислення X3DH секрету
4	A,B	Формування Root Key
5	A,B	Ініціалізація Double Ratchet
6	A↔B	AEAD-передача з оновленням ключів

Viber використовує схему E2E, проте без окремої моделі розширених pre-keys як у Signal. Документ протоколу Viber описує формування гібридного криптографічного стану на основі DH-обміну з подальшим формуванням трьох ключів (message key, MAC key, IV). Передача даних виконується у зашифрованому вигляді, а одержувач виконує перевірку цілісності та автентичності кожного отриманого повідомлення.

Основні етапи встановлення сесії у Viber наведено у табл. 1.4 [10].

Таблиця 1.4 – Основні етапи встановлення сесії у Viber

Етап	Напрямок	Опис дії
1	A→B	Надсилання ініціаційного повідомлення
2	A↔B	DH обмін відкритими значеннями
3	A,B	Формування сесійних ключів (3-key set)
4	A→B	Передача зашифрованих даних
5	B	Перевірка цілісності та автентичності

Оновлена версія Facebook Messenger впроваджує повний режим end-to-end encryption для приватних чатів. Ключовий матеріал формується на основі протоколу X3DH-типу (аналогічна модель до Signal), після чого застосовується

механізм ротації ключів для подальшого обміну. Повідомлення супроводжуються перевірочними MAC значеннями.

Основні етапи обміну у Facebook Messenger представлено у табл. 1.5 [11].

Таблиця 1.5 – Основні етапи обміну у Facebook Messenger

Етап	Напрямок	Опис дії
1	B→Server	Публікація pre-keys
2	A→Server	Отримання pre-keys
3	A↔B	Встановлення спільного секрету
4	A,B	Формування сесійних ключів
5	A→B	Передача зашифрованих повідомлень
6	A↔B	Подальший обмін з ротацією ключів

Отже, сучасні месенджери реалізують різні моделі встановлення спільного секрету. Telegram застосовує модель «клієнт–сервер», де AuthKey формується між клієнтом і сервером, а не між двома користувачами, тому cloud-чати не є повноцінними end-to-end.

Натомість WhatsApp, Signal та оновлений Messenger використовують Signal-підхід: формування початкового секрету через X3DH із застосуванням pre-keys та подальшу ротацію ключів за Double Ratchet, що забезпечує forward secrecy. Viber реалізує E2E-шифрування без розширеного pre-key механізму, однак також формує сесійні ключі через DH-обмін.

Таким чином, еволюція йде у напрямку асинхронності та частішого оновлення ключів, що підвищує стійкість систем до ретроспективного криптоаналізу та компрометації секретного матеріалу.

1.2 Протоколи розподілу ключів

У криптографії розподіл відкритих і закритих ключів між відправником і одержувачем є дуже монотонним завданням.

Розповсюдження ключів має вирішальне значення, оскільки безпека всієї системи залежить від того, наскільки добре ключі спільно використовуються та захищені.

Розподіл ключів – це процес безпечного обміну криптографічними ключами між сторонами, що беруть участь у зв’язку, що гарантує збереження ключів у таємниці та неможливість їх перехоплення або зміни неавторизованими третіми особами [12].

Розподіл ключів відіграє значну роль як у симетричній, так і в асиметричній криптографії та забезпечує доступність ключів для зашифрування та розшифрування, зберігаючи при цьому цілісність та конфіденційність зв’язку.

У симетричній криптографії і відправник, і одержувач використовують один і той самий ключ як для зашифрування, так і для розшифрування.

Це означає, що обидві сторони повинні мати доступ до одного й того ж секретного ключа, що створює проблеми із забезпеченням безпечного розповсюдження ключа.

Схему алгоритму симетричного шифрування зображено на рис. 1.1.

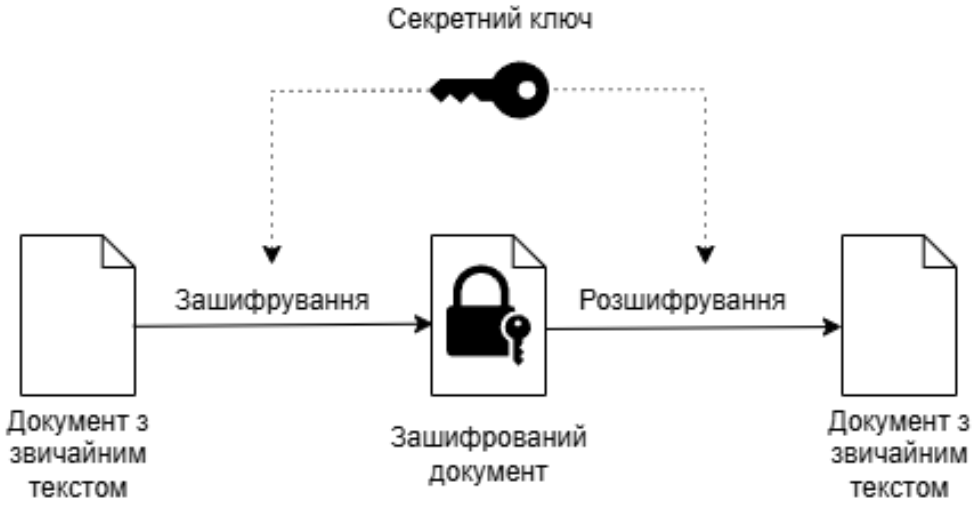


Рисунок 1.1 – Алгоритм симетричного шифрування

Розподіл симетричних ключів є критичною фазою встановлення захищеного зв’язку – саме на цьому етапі забезпечується, щоб тільки довірені сторони мали доступ до ключа. Через це при розподілі ключів часто виникають низка практичних і організаційних проблем:

Спільне використання секрету. Ключ має бути безпечно переданий відправнику та одержувачу, перш ніж може відбутися будь-яке зашифроване

спілкування. Якщо зловмисник перехопить ключ під час розповсюдження, весь зв’язок може бути скомпрометований.

– Керування ключами. У системах з багатьма учасниками розподіл унікального ключа для кожної пари взаємодіючих сторін може стати складним.

Для забезпечення безпеки симетричного шифрування ключ повинен бути переданий сторонам надійним способом. Існує кілька протоколів розподілу симетричних ключів:

– Ручний розподіл;

– Інфраструктура відкритих ключів (PKI, Public Key Infrastructure);

– Обмін ключами за Діффі-Гелманом;

– Квантовий розподіл ключів (QKD, Quantum Key Distribution) [8].

Ручний розподіл полягає у тому, що ключ доставляється фізично або вручну обмінюється між сторонами (наприклад, через довіреного кур’єра, особисту зустріч або захищений офлайн-канал).

Хоча PKI переважно використовується в асиметричній криптографії, вона також може використовуватися для розповсюдження симетричних ключів. Система використовує пари публічних та закритих ключів , які дозволяють сторонам безпечно обмінюватися симетричними ключами.

Наприклад, симетричний ключ може бути зашифрований за допомогою відкритого ключа одержувача та безпечно надісланий [12].

Процес PKI заключається у наступному:

1) Сторона А шифрує симетричний ключ, використовуючи відкритий ключ Сторони Б.

2) Сторона В розшифровує симетричний ключ, використовуючи свій закритий ключ.

Діффі-Гелман – це протокол обміну ключами, який дозволяє двом сторонам безпечно використовувати симетричний ключ через незахищений канал без необхідності попередніх спільних секретів.

Процес розповсюдження ключів Діффі-Геллмана складається з таких етапів:

1) Обидві сторони погоджуються щодо публічної бази та модуля.

2) Кожна сторона генерує закритий ключ та обчислює відповідний відкритий ключ.

3) Відкриті ключі обмінюються, і кожна сторона поєднує свій закритий ключ з відкритим ключем іншої сторони для обчислення спільного симетричного ключа.

QKD – це передовий метод, який використовує квантову механіку для безпечного розподілу симетричних ключів. Безпека QKD базується на принципах квантової суперпозиції та заплутаності, що гарантує виявлення будь-якої спроби підслуховування обміну ключами [12].

Кожний з вищезазначених методів розподілу симетричних ключів має свої переваги та недоліки. Основні переваги та недоліки протоколів розподілу симетричних ключів представлено у табл. 1.6.

Таблиця 1.6 – Переваги та недоліки протоколів розподілу симетричних ключів

Назва протоколу	Переваги	Недоліки
Ручний розподіл	Безпечно, якщо ключ транспортується фізично без перехоплення	Непрактичний для великомасштабних систем та створює логістичні проблеми
PKI	Безпечний обмін ключами навіть через незахищений канал	Накладні витрати через використання асиметричного шифрування для обміну симетричними ключами
Діффі-Гелман	Не вимагає від сторін попередньої зустрічі чи обміну будь-якими секретними ключами	Вразливий до атак типу «людина посередині» (MITM, Man-In-The-Middle), якщо не проведено автентифікацію.
QKD	Теоретично захищений від прослуховування	Потребує спеціалізованого обладнання та ще не отримав широкого поширення

В асиметричній криптографії кожен учасник має пару ключів: відкритий ключ (який можна використовувати відкрито) та закритий ключ (який

зберігається в таємниці). Основне використання асиметричної криптографії полягає в безпечному розподілі симетричних ключів, що дозволяє обом сторонам безпечно обмінюватися інформацією без попереднього використання секретного ключа [12].

Схему алгоритму асиметричного шифрування наведено на рис. 1.2.

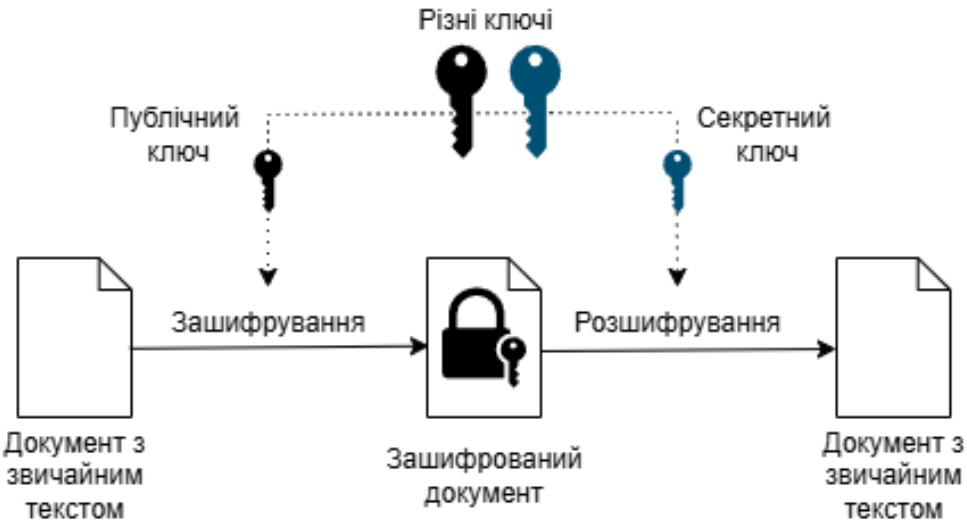


Рисунок 1.2 – Алгоритм асиметричного шифрування

Хоча асиметричний розподіл ключів значно спрощує обмін секретною інформацією, його використання також пов’язане з низкою викликів. Основні проблеми, що можуть виникати під час цього процесу, включають:

Довіра до відкритого ключа. Одержувач повинен бути впевненим, що отриманий ним відкритий ключ є легітимним і не підроблений зловмисником. Цю проблему вирішує використання цифрових сертифікатів, які може перевірити довірений центр сертифікації (CA, Certification Authority).

Масштабованість . У великомасштабних системах керування відкритими ключами багатьох користувачів може стати складним.

Для забезпечення надійності асиметричної криптографії необхідно використовувати спеціальні механізми обміну та перевірки відкритих ключів [12].

Існують різні протоколи асиметричного розподілу ключів:

– PKI;

– Павутиння довіри (WOT);

– Цифрові підписи.

PKI – це фреймворк, який використовує цифрові сертифікати для перевірки автентичності відкритих ключів. CA підписує цифрові сертифікати, пов’язуючи особу користувача з його відкритим ключем.

Процес PKI в асиметричній криптографії полягає у наступному:

1) Відправник отримує відкритий ключ одержувача від довіреного центру сертифікації.

2) Відправник шифрує повідомлення за допомогою відкритого ключа одержувача.

3) Одержувач розшифровує повідомлення, використовуючи свій закритий ключ.

Мережа довіри (WOT, Web Of Trust) є альтернативою PKI, де довіра до відкритих ключів встановлюється через прямі або непрямі зв’язки між користувачами. У WOT користувачі підписують відкриті ключі один одного для встановлення довіри [12].

Цифрові підписи використовують асиметричне шифрування для забезпечення автентичності повідомлень. Хоча цифрові підписи часто використовуються для перевірки цілісності даних, вони також можуть відігравати певну роль у розподілі ключів.

Процес використання цифрових підписів складається з таких етапів:

1) Відправник підписує повідомлення, використовуючи свій закритий ключ.

2) Одержувач може перевірити підпис за допомогою відкритого ключа відправника.

Кожний з вищезазначених методів розподілу асиметричних ключів має свої переваги та недоліки. Основні переваги та недоліки протоколів розподілу асиметричних ключів наведено табл. 1.7.

Таблиця 1.7 – Переваги та недоліки протоколів розподілу асиметричних ключів

Назва протоколу	Переваги	Недоліки
PKI	Безпечний розподіл ключів з перевіркою особи	Спирається на довірений центр сертифікації, а також є накладні витрати на керування сертифікатами
WOT	Децентралізований та гнучкіший, ніж PKI	Може стати громіздким зі зростанням кількості учасників, а також йому бракує централізованого органу управління, який забезпечує PKI
Цифрові підписи	Забезпечує цілісність, автентичність та неможливість заперечення	Не використовується безпосередньо для обміну ключами, але може допомогти перевірити легітимність відкритих ключів

На практиці сучасні криптографічні системи часто використовують гібридний розподіл ключів , де асиметрична криптографія використовується для безпечного обміну симетричним ключем, а потім симетричне шифрування використовується для фактичної передачі даних.

Це поєднує переваги обох типів криптографії: ефективність симетричного шифрування та можливості безпечного обміну ключами асиметричного шифрування [12].

Процес розподілу гібридних ключів складається з наступних етапів:

1) Обмін ключами . Сторони використовують асиметричну криптографічну систему (наприклад, RSA або Діффі-Гелман) для безпечного обміну симетричним ключем.

2) Зашифрування. Після обміну ключами фактичне спілкування шифрується за допомогою симетричного алгоритму шифрування, такого як AES.

3) Розшифрування. Отримувач розшифровує дані за допомогою спільного симетричного ключа.

Розподіл ключів є фундаментальним аспектом криптографічних систем, що забезпечує безпечний обмін необхідними ключами та їх використання для зашифрування та розшифрування.

Незалежно від того, чи використовується симетрична чи асиметрична криптографія, чи гібридний підхід, метод розподілу ключів повинен забезпечувати конфіденційність, автентичність та цілісність [12].

На основі проведеного аналізу протоколів розподілу ключів можна зробити висновок, що кожен із розглянутих протоколів має свої переваги, недоліки та сферу застосування.

Серед розглянутих протоколів, протокол Діффі-Геллмана займає особливе місце, оскільки поєднує у собі високу криптографічну стійкість і практичність реалізації. Він дозволяє двом сторонам безпечно сформувати спільний симетричний ключ через відкритий (незахищений) канал зв’язку, не потребуючи попереднього обміну секретами або використання централізованих довірених служб. Безпека протоколу ґрунтується на складності задачі обчислення дискретного логарифма, що робить його стійким до більшості сучасних криптографічних атак.

Завдяки своїм властивостям – простоті реалізації, високому рівню безпеки, відсутності необхідності в попередньому розподілі секретних даних та сумісності з іншими криптографічними алгоритмами — протокол Діффі–Геллмана став стандартом у багатьох сучасних системах безпечного обміну даними (зокрема, TLS, SSH, IPsec) [12].

У межах роботи для реалізації протоколу обміну інформацією було обрано саме протокол угоди про ключі Діффі–Геллмана, оскільки він забезпечує оптимальне співвідношення між рівнем безпеки, ефективністю та універсальністю застосування. Використання цього протоколу дозволяє створити надійний механізм формування спільного секретного ключа між сторонами без необхідності передачі його відкритими каналами, що повністю відповідає вимогам до розроблюваного програмного засобу для захищеного передавання інформації.

1.3 Протокол угоди про ключі Діффі-Геллмана

Протокол угоди про ключі Діффі–Геллмана – це метод обміну криптографічними ключами. Один з перших практичних прикладів узгодження ключа, що дозволяє двом учасникам, що не мають жодних попередніх даних один про одного, отримати спільний секретний ключ із використанням незахищеного каналу зв'язку. Цей ключ можна використати для шифрування наступних сеансів зв'язку, що використовують шифр з симетричним ключем.

Алгоритм обміну ключами необхідний у комунікації та криптографії з кількох причин:

1) Це дозволяє двом або більше сторонам узгодити секретний ключ, не розкриваючи його потенційним зловмисникам , який потім використовується для шифрування та дешифрування, що життєво важливо для збереження конфіденційності комунікації.

2) Збереження цілісності даних також було серйозною проблемою в цифровому зв'язку, де дані завжди вразливі до спотворення під час передачі. Алгоритм обміну ключами допомагає зберегти цілісність переданих даних, запобігаючи несанкціонованій зміні або підробці даних під час передачі.

3) Алгоритм обміну ключами сприяє автентифікації сторін, що взаємодіють, перевіряє, ким вони себе видають, тим самим підвищуючи ризик атаки MITM [13].

Таким чином, поряд із шифруванням даних для збереження конфіденційності зв'язку, також був потрібен алгоритм обміну ключами для підтримки цілісності та авторизованого доступу до інформації.

Безпека обміну ключами Діффі-Геллмана спирається на математичні властивості модульного піднесення до степеня та задачі дискретного логарифмування.

Модульне піднесення до степеня – це процес піднесення числа до степеня та отримання лишку від ділення на модуль. В алгоритмі Діффі-Геллмана

модульне піднесення до степеня використовується для обчислення A та B, якими потім обмінюються сторони.

З іншого боку, задача дискретного логарифмування – це завдання знаходження степеня за основою, модулем та результатом модульного піднесення до степеня. Безпека обміну ключами Діффі-Геллмана базується на припущенні, що задачу дискретного логарифмування неможливо вирішити обчислювально, що ускладнює для зловмисника обчислення спільного секретного ключа [13].

Протокол неавтентифікаційного розподілу ключів Діффі-Геллмана полягає у наступному.

Відкриті параметри: просте p і генератор a групи $\mathbf{Z}_p (2 \leq a \leq p-2)$.

1) Учасник A вибирає випадкове число $x (1 < x \leq p-2)$, що зберігається в секреті, і посилає учаснику B таке повідомлення:

$$A \rightarrow B: a^x \ mod \ p \tag{1.1}$$

2) Учасник B вибирає випадкове число $(1 < y \leq p-2)$, що зберігається в секреті, і посилає учаснику A таке повідомлення:

$$A \leftarrow B: a^y \ mod \ p \tag{1.2}$$

3) Учасник A одержує $a^y \ mod \ p$ й обчислює сеансовий ключ:

$$K = (a^y \ mod \ p)^x \ mod \ p = a^{xy} \ mod \ p \tag{1.3}$$

Учасник B одержує $a^x \ mod \ p$ й обчислює сеансовий ключ:

$$K = (a^x \ mod \ p)^y \ mod \ p = a^{xy} \ mod \ p \tag{1.4}$$

Переваги обміну ключами Діффі-Геллмана включають:

1) Пряма секретність – протокол дозволяє сторонам генерувати новий спільний секретний ключ для кожного сеансу зв’язку, гарантуючи, що компрометація одного ключа не вплине на безпеку минулих або майбутніх сеансів.

2) Масштабованість – обмін ключами Діффі-Геллмана добре масштабується залежно від кількості учасників, оскільки кожній стороні потрібно виконати лише невелику кількість степенів для обчислення спільного секретного ключа.

3) Відсутність попереднього спілкування – протокол не вимагає жодного попереднього спілкування чи обміну інформацією між сторонами, що робить його придатним для використання в ситуаціях, коли встановлення попередньої довіри є складним [14].

Обмеження обміну ключами Діффі-Геллмана включають:

1) Вразливість до атак MITM – протокол не забезпечує автентифікацію сторін, що робить його вразливим до атак «людина посередині», коли зловмисник може видати себе за одну або обидві сторони та перехопити або змінити зв’язок. Щоб зменшити цей ризик, обмін ключами Діффі-Геллмана часто поєднується з цифровими підписами або іншими механізмами автентифікації.

2) Обчислювальні витрати – обмін ключами Діффі-Геллмана передбачає модульне піднесення до степеня, що може бути обчислювально ресурсоємним, особливо для великих простих чисел. Однак це обмеження можна вирішити, використовуючи ефективні алгоритми для модульного піднесення до степеня або реалізуючи протокол з криптографією еліптичних кривих , яка вимагає менших розмірів ключів для еквівалентної безпеки.

3) Відсутність шифрування або захисту цілісності даних – протокол надає лише метод для встановлення спільного секретного ключа; він не пропонує шифрування даних або захисту цілісності. Для захисту зв’язку спільний секретний ключ має використовуватися разом із симетричним алгоритмом шифрування та кодом автентифікації повідомлення (MAC, Message Authentication Code) або автентифікованим шифруванням [14].

Алгоритм Діффі-Геллмана широко використовується в різних реальних програмах для встановлення безпечних каналів зв’язку між сторонами. Деякі поширені застосування включають:

1) Безпека транспортного рівня (TLS, Transport Layer Security) – як ключовий компонент протоколу TLS, обмін ключами Діффі-Геллмана використовується для встановлення спільного секретного ключа для безпечного зв’язку між веб-браузерами та серверами, захищаючи конфіденційні дані, такі як облікові дані для входу, платіжна інформація та особисті дані.

2) Безпечна оболонка (SSH, Secure Shell) – обмін ключами Діффі-Геллмана використовується в протоколі SSH для забезпечення безпечного віддаленого доступу та керування комп’ютерними системами через незахищену мережу.

3) Віртуальні приватні мережі (VPN, Virtual Private Network) – у VPN, що використовують протокол IPsec, обмін ключами Діффі-Геллмана використовується під час процесу обміну ключами Інтернету (IKE, Internet Key Exchange) для встановлення спільного секретного ключа для захисту передачі даних між кінцевими точками VPN.

4) Програми миттєвого обміну повідомленнями та голосового зв’язку через IP (VoIP, Voice over Internet Protocol) – обмін ключами Діффі-Геллмана використовується в різних програмах миттєвого обміну повідомленнями та VoIP, таких як Signal та WhatsApp, для встановлення наскрізного шифрування, захищаючи конфіденційність повідомлень та дзвінків.

5) Шифрування електронної пошти – такі протоколи, як Pretty Good Privacy (PGP) та Secure/Multipurpose Internet Mail Extensions (S/MIME), можуть використовувати обмін ключами Діффі-Геллмана для безпечного обміну симетричними ключами для шифрування та дешифрування повідомлень електронної пошти.

Існують різні варіації алгоритму Діффі-Геллмана, зокрема:

1) Еліптична крива Діффі-Геллмана (ECDH). Цей варіант використовує криптографію еліптичної кривої, яка пропонує еквівалентну безпеку з меншими

розмірами ключів, зменшуючи обчислювальні вимоги та покращуючи продуктивність.

2) Анонімний протокол Діффі-Геллмана. Цей варіант не забезпечує автентифікацію, що робить протокол вразливим до атак MITM.

3) Статичний алгоритм Діффі-Геллмана. У цьому варіанті принаймні одна сторона використовує фіксований відкритий ключ, який не забезпечує прямої секретності.

4) Ефемерний Діффі-Геллмана полягає у тому, що обидві сторони генерують тимчасові відкриті ключі для кожного сеансу, забезпечуючи пряму секретність, яка гарантує, що скомпрометований довгостроковий ключ не вплине на ключі попередніх сеансів.

5) Потрійний ключ Діффі-Геллмана. Цей протокол поєднує ефемерний ключ Діффі-Геллмана з додатковою парою ключів для забезпечення взаємної автентифікації та прямої секретності.

6) ElGamal. Це схема шифрування з відкритим ключем, заснована на обміні ключами Діффі-Геллмана, що дозволяє безпечне шифрування та дешифрування повідомлень [13, 14].

1.4 Порівняльний аналіз протоколів TCP та UDP у мережах передачі даних

Десятки мільйонів пристроїв у всьому світі зв’язуються за допомогою комп’ютерних мереж. І кожну комп’ютерну мережу організовано за певними стандартами. Будь-які пристрої взаємодіють за загальноприйнятою моделлю OSI. Дана модель визначає взаємодію різних мережевих пристроїв на семи рівнях – Media (до них відносяться фізичний, канальний та мережевий) та Host – (транспортний, сеансовий, уявлення та прикладний). TCP та UDP відносяться до найпоширеніших мережевих протоколів транспортного рівня [15].

UDP (User Datagram Protocol) – протокол, що забезпечує передачу даних (датаграм) без попереднього створення з’єднання між хостами. При відправленні

датаграм немає впевненості в існуванні одержувача і його готовності до обміну. Мережевий протокол UDP не забезпечує також упорядкування датаграм при отриманні. Він використовується додатками, для яких істотне значення має час доставки, коли немає можливості чекати затримані або запитувати втрачені пакети, наприклад, в системах реального часу. Датаграми можуть доставлятися не в заданому порядку, дублюватися або зовсім не доставлятися. Тому протокол UDP називають «ненадійним протоколом датаграм» [15].

Додатки, що використовують протокол UDP, не чутливі до втрат даних, порушення порядку отримання датаграм і дублювання. При цьому вони можуть використовувати механізми забезпечення надійності на прикладному рівні.

Протокол передачі даних TCP (Transmission Control Protocol) – протокол, що забезпечує надійну доставку пакетів даних, він забезпечує встановлення з'єднання між двома хостами методом «рукостискання», після якого може здійснюватися обмін даними.

Перед початком передачі пакетів через TCP з'єднання встановлюється сесія з одержувачем, в рамках якої потім здійснюється передача даних. Це дозволяє переконатися в тому, що одержувач існує і готовий приймати дані. Після завершення передачі сесія закривається, одержувач повідомляється про те, що даних більше не буде, а відправник повідомляється про те, що одержувач повідомлений [15].

Кожен пакет при обміні має свій порядковий номер. TCP автоматично впорядковує пакети, використовуючи порядковий номер, і передає після склеювання на рівень додатків. Після відправлення декількох пакетів очікується підтвердження і порядковий номер наступного пакета. Якщо підтвердження не отримано, відправлення повторюється, якщо спроби не увінчалися успіхом, сесія розривається. Кількість пакетів даних, на які буде запитуватися підтвердження, залежить від надійності мережі. Якщо дані втрачаються, то підтвердження автоматично запитується частіше. Це називається механізмом ковзаючого вікна (sliding window), завдяки якому TCP може працювати з мережами, незалежно від рівня їх надійності [15].

Застосування TCP доцільне там, де неприпустима втрата даних, наприклад, при авторизації, а також при передачі шифрованої інформації.

Порівняльні характеристики протоколів транспортного рівня описано у табл. 1.8.

Таблиця 1.8 – Порівняльні характеристики протоколів транспортного рівня

Властивість/протокол	TCP	UDP
З’єднання	TCP перед відправленням даних попередньо встановлює з’єднання з отримувачем.	Попереднє з’єднання не встановлюється.
Надійність	Надійний. TCP забезпечує підтвердження отримання, повторну передачу і тайм-аут між повідомленнями. У разі невдачі здійснюються спроби доставити повідомлення повторно. Якщо воно загубиться під час доставки, отримувач автоматично запитує відсутню частину.	Ненадійний. Під час відправлення даних невідомо, чи будуть вони доставлені отримувачу. Відсутнє підтвердження отримання, втрачені дейтаграми повторно не передаються.
Упорядкованість	Якщо дані отримані не в тому порядку, вони зберігаються у буфері до того часу, поки дані не будуть «зклеєні» у необхідному порядку, після чого вони будуть передані застосунку-отримувачу.	Порядок доставки повідомлень не відомий і може бути порушений.
Перевантаження	TCP контролює перевантаження.	Під час обміну між застосунками можливі перевантаження, захист від яких повинен реалізовуватися на рівні застосунку.

Виходячи з наведених порівняльних характеристик, для розробки програмного засобу захищеного передавання інформації доцільним є використання протоколу TCP, оскільки він забезпечує гарантовану доставку даних, контроль упорядкування пакетів та механізми повторної передачі у разі втрати інформації [15].

Ці властивості є критично важливими для систем, що працюють із конфіденційними або зашифрованими повідомленнями, де неприпустимими є втрата даних, їх дублювання чи порушення порядку отримання. Надійність, підтвердження доставки та керування перевантаженнями роблять TCP оптимальним вибором для забезпечення цілісності та безпечності інформаційного обміну.

1.5 Висновки до розділу

У результаті проведеного аналізу інформаційних джерел встановлено, що сучасні системи захищеного цифрового обміну повідомленнями ґрунтуються на поєднанні протоколів узгодження та розподілу ключів, механізмів наскрізного шифрування та вибору транспортного протоколу. Саме модель формування спільного секрету й організація криптографічного стану визначають стійкість системи до атак типу «людина посередині», здатність протистояти компрометації ключів та забезпечувати довготривалу конфіденційність даних.

Аналіз протоколів обміну інформацією у месенджерах показав, що Telegram використовує серверно-орієнтовану модель, у якій AuthKey формується між клієнтом і сервером, а хмарні чати не є повноцінними наскрізно захищеними. Натомість WhatsApp, Signal та оновлений Facebook Messenger застосовують Signal-підхід з використанням pre-keys, X3DH та Double Ratchet, що забезпечує асинхронність встановлення сеансу, forward secrecy та стійкість до компрометації ключового матеріалу. Viber також реалізує E2E-шифрування через DH-обмін і формування набору сеансових ключів, але без розвиненої pre-key інфраструктури. Загальна тенденція еволюції таких протоколів полягає у

переході від статичних сесій до частого оновлення ключів для кожного повідомлення.

Дослідження протоколів розподілу ключів показало, що на практиці домінує гібридна модель, де асиметрична криптографія використовується для початкового узгодження секрету або передачі симетричного ключа, а подальший обмін даними здійснюється із застосуванням симетричних алгоритмів шифрування. У цьому контексті протокол угоди про ключі Діффі–Геллмана відіграє ключову роль, оскільки дозволяє сторонам сформувати спільний секрет без попередньо спільних даних та без передачі секрету каналом зв’язку, а його стійкість ґрунтується на складності задачі дискретного логарифмування. Саме тому класичний та ефемерний Діффі–Гелман, у тому числі на основі еліптичних кривих, широко застосовуються в TLS, SSH, IPsec та в протоколах типу X3DH, що використовуються в E2E-месенджерах.

Проведений порівняльний аналіз транспортних протоколів TCP та UDP засвідчив, що для систем захищеного обміну інформацією критичними є гарантована доставка даних, збереження порядку пакетів та наявність механізмів повторної передачі. Цим вимогам відповідає протокол TCP, який забезпечує встановлення з’єднання, контроль перевантажень, підтвердження отримання та відновлення втрачених пакетів, тоді як UDP орієнтований на швидкість і допускає втрати та порушення порядку доставки, що є неприйнятним для роботи з конфіденційними або зашифрованими повідомленнями.

Отже, обрана в межах даної роботи модель побудови захищеного каналу – використання протоколу угоди про ключі Діффі-Геллмана для формування спільного симетричного ключа у поєднанні та застосуванням протоколу TCP на транспортному рівні – є обґрунтованою. Таке поєднання забезпечує математично стійкий механізм узгодження ключів, відсутність необхідності в попередньому розподілі секретів, а також надійну, впорядковану та контрольовану за помилками доставку зашифрованих повідомлень, що повністю відповідає вимогам до розроблюваного програмного засобу захищеного передавання інформації.

2 РОЗРОБКА ПРОТОКОЛУ ОБМІНУ ІНФОРМАЦІЄЮ

2.1 Узагальнений протокол обміну інформацією та механізм взаємної автентифікації

Узагальнений протокол обміну інформацією є центральною складовою розробленої системи захищеної передачі даних та визначає послідовність дій між двома користувачами (A та B) під час встановлення захищеного сеансу зв'язку.

Реалізація безпечного встановлення сеансу зв'язку у розробленій системі передбачає послідовність операцій, спрямованих на автентифікацію сторін, узгодження криптографічних параметрів та формування спільного секретного ключа.

Ключовою ідеєю є використання сервера, який не бере участі у шифруванні чи обчисленні секретних величин, але виконує роль розподіленої довіреної служби атестації ключів (PKDI, Public Key Directory Infrastructure).

На етапі створення клієнтів відбувається формування їхніх унікальних ідентифікаторів та ключової інформації. Кожен клієнт генерує власний ідентифікатор, який використовується для адресації в межах системи. Ідентифікатор може зчитуватися із локального конфігураційного файлу або генеруватися випадково під час першого запуску. Значення ID_A та ID_B, зберігаються локально на стороні клієнтів в оперативній пам'яті, а також можуть бути записані у файл для подальшого використання [13, 14, 16].

Після встановлення ідентифікатора клієнт формує криптографічний ключ підпису на основі алгоритму Ed25519. Якщо відповідний приватний ключ присутній у файловій системі, він завантажується з локального PEM-файлу.

У випадку його відсутності створюється нова пара ключів, причому приватний ключ зберігається на диску, а публічний ключ обчислюється безпосередньо з нього та зберігається в оперативній пам'яті. Таким чином, приватні ключі є персональними для кожного користувача та ніколи не

покидають його робочого середовища. Схему розміщення даних про клієнта наведено на рис. 2.1

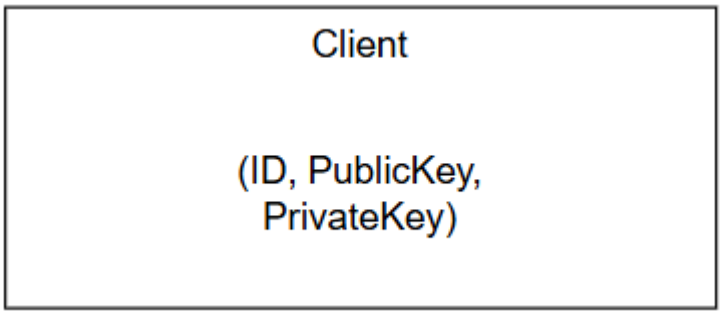


Рисунок 2.1 – Схема розміщення даних про клієнта

Після завершення етапу ініціалізації клієнти переходять до встановлення мережевого з'єднання зі сервером з метою реєстрації та пошуку цільового співрозмовника [16].

На початку встановлення з'єднання користувач А надсилає запит серверу, зазначаючи власний ідентифікатор, ідентифікатор адресата та свій публічний ключ:

$$A \rightarrow Server: (ID_A, ID_B, PubKey_A). \quad (2.1)$$

Після отримання запиту сервер повертає користувачу А відповідні відкриті криптографічні параметри:

$$Server \rightarrow A: (PubKey_B, p, g), \quad (2.2)$$

де $PubKey_B$ – відкритий ключ користувача В.

Схему отримання відкритого ключа користувача В користувачем А наведено на рис. 2.2.

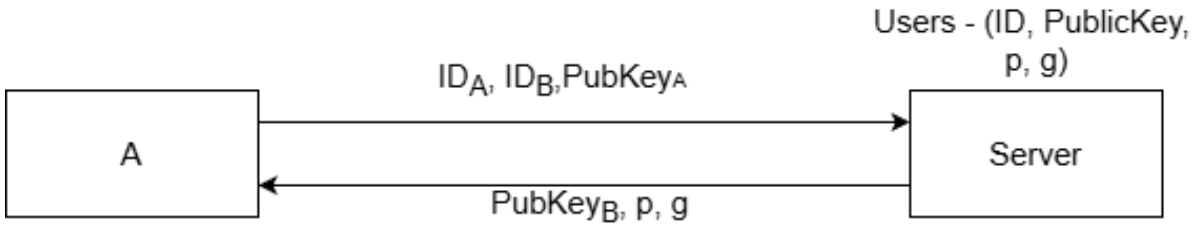


Рисунок 2.2. – Схема отримання відкритого ключа користувача B користувачем A

Аналогічна процедура виконується на стороні B. Користувач B надсилає серверу запит:

$$B \rightarrow Server: (ID_B, ID_A, PubKey_B). \quad (2.3)$$

У відповідь сервер надсилає набір параметрів:

$$Server \rightarrow B: (PubKey_A, p, g), \quad (2.4)$$

де $PubKey_A$ – відкритий ключ користувача A.

Схему отримання відкритого ключа користувача B користувачем A наведено на рис. 2.3.

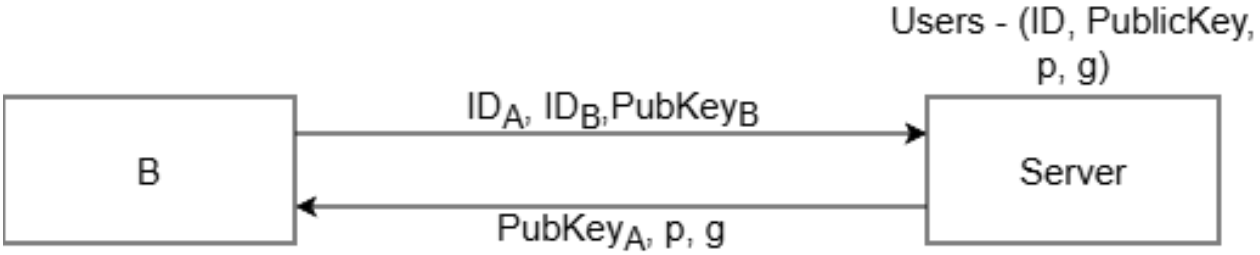


Рисунок 2.3. – Схема отримання відкритого ключа користувача A користувачем B

Після завершення взаємодії з сервером обидві сторони володіють: $(PubKey_A, PubKey_B, p, g)$ та можуть виконувати обмін ключами у режимі peer-to-peer, без залучення сервера [13, 14, 16].

Для підтвердження справжності сторін у протоколі додатково використовується механізм nonce-значень R_A та R_B, що гарантують захист від повторного відтворення.

Схему взаємної автентифікації сторін A та B наведено на рис. 2.4.

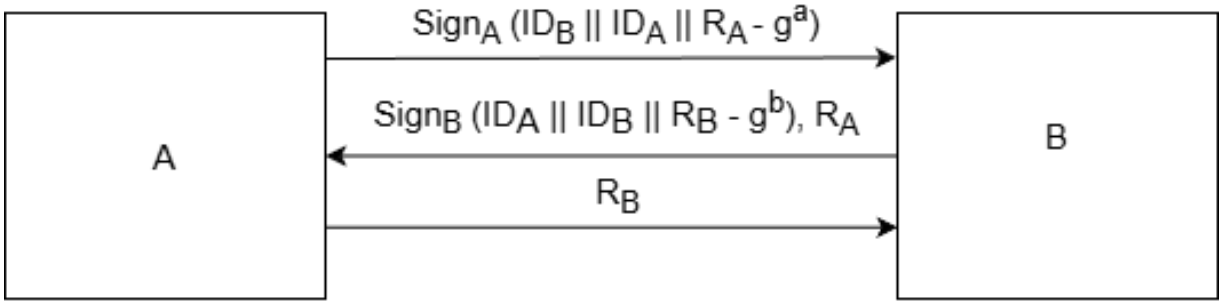


Рисунок 2.4 – Схема взаємної автентифікації сторін A та B

Після отримання відкритих ключів, користувач A переходить до першого автентифікаційного кроку. Сторона A формує повідомлення, що містить її власний ідентифікатор, ідентифікатор отримувача, згенерований nonce R_A та елемент g^{a}. Цей набір даних підписується приватним ключем користувача A [13, 14, 16]. У підпис вставлено всі критичні параметри: ID_A, ID_B, R_A, а також саму частину Діффі-Геллмана A_{pub}, яка обчислюється за формулою:

$$A_{pub} = g^a \ mod \ p, \tag{2.5}$$

де a – приватний випадковий параметр користувача A.

Завдяки цьому користувач B може переконатися не лише в тому, що повідомлення надійшло саме від користувача A, а й у тому, що його цілісність не була порушена.

Сторона B, отримавши це повідомлення, перевіряє цифровий підпис за допомогою відкритого ключа користувача A, який було отримано раніше через сервер. Якщо підпис коректний і всі вкладені ідентифікатори відповідають очікуваним значенням, користувач B переходить до формування відповіді [13,

14, 16]. У відповідь користувач B надсилає користувачу A повідомлення, яке містить власний nonce R_B, елемент B_{pub}, що обчислюється за формулою:

$$B_{pub} = g^b \ mod \ p, \tag{2.6}$$

а також повертає отримане від користувача A значення R_A як підтвердження того, що повідомлення не було змінено.

Це повідомлення також підписується приватним ключем користувача B, що дає можливість користувачу A впевнитися у справжності відправника під час перевірки.

Після отримання відповіді користувач A зчитує значення R'_A з повідомлення від користувача B та порівнює його з власним R_A. Якщо значення збігаються, користувач A переконується, що саме користувач B отримав і обробив початкове повідомлення. У відповідь користувач A надсилає користувачу B значення R_B, яке було вкладене у друге повідомлення. Сторона B, отримавши це значення, перевіряє $R'_B = R_B$?. Якщо перевірка успішна, користувач B завершає автентифікацію [13, 14, 16].

Таким чином, обидві сторони отримують взаємні підтвердження:

– Користувач A підтверджує особу користувача B завдяки перевірці підпису сторони B та відповідності R'_A.

– Користувач B підтверджує особу користувача A завдяки перевірці підпису сторони A та відповідності R'_B.

Жоден з цих етапів не може бути підмінений або змінений зловмисником, оскільки всі повідомлення підписуються приватними ключами сторін, а параметри Діффі-Геллмана включено в підпис. Це усуває можливість атаки MITM, яка є найбільш типовою загрозою для неавтентифікованого Діффі-Геллмана.

Особливістю запропонованого протоколу є те, що підписані значення R_A, R_B, g^aта g^b одночасно виконують дві функції:

– автентифікують кожну зі сторін;

– гарантують коректну передачу параметрів Діффі-Геллмана.

Тобто процеси автентифікації та узгодження сеансового ключа не виконуються окремо, як у класичних схемах, а об’єднані в один спільний механізм. Це суттєво знижує кількість мережевих повідомлень, спрощує реалізацію та забезпечує цілісність усіх ключових параметрів, що використовуються у подальшому шифруванні.

Отримавши відкритий компонент партнера, кожна сторона незалежно обчислює однаковий спільний секрет:

$$K = (B_{pub})^a \ mod \ p = (A_{pub})^b \ mod \ p, \tag{2.7}$$

де K – спільне секретне значення, що не передається мережею.

У результаті виконання всіх вищенаведених кроків сторони А та В незалежно обчислюють однакове значення спільного секрету $K = g^{\{ab\}}$, яке надалі використовується у методі шифрування на основі ортогональної поворотної матриці $\mathbf{\Gamma}(\hat{q})$, сформованої з використанням кватерніонної нормалізації. Таким чином завершується ініціалізація захищеного каналу зв’язку [17, 18].

Після цього секрет перетворюється у внутрішній робочий 64-бітний ключ:

$$K_0 = Compress\ (K). \tag{2.8}$$

Функція Compress() виконує операцію зменшення розрядності з використанням кватерніонного перетворення.

Після встановлення ключа відбувається передавання зашифрованого повідомлення:

$$A \rightarrow B: E_{K_0}(\mathbf{M}) || \ MAC, \tag{2.9}$$

де $E_{K_0}(M)$ – результат зашифрування повідомлення M з використанням ключа K_0.

MAC – код автентифікації повідомлення М.

Отримувач виконує розшифрування та повторне обчислення коду автентифікації MAC'. Якщо $MAC' = MAC$, то повідомлення не було змінено, інакше наявні певні зміни [18, 19].

2.2 Реалізація протоколу Діффі-Геллмана

Реалізація протоколу Діффі–Геллмана у розробленій системі забезпечує узгодження спільного секретного ключа між двома сторонами – користувачами А та В через незахищений канал зв'язку без попереднього обміну секретною інформацією. Основу протоколу становлять операції модульного піднесення до степеня в мультиплікативній групі залишків за простим модулем, що гарантує криптографічну стійкість на основі складності задачі дискретного логарифмування [13, 14].

Обидві сторони домовляються про використання спільних відкритих параметрів:

$$p - \text{велике просте число}, g - \text{первісний корінь (генератор)у } Z_p^*, \qquad (2.10)$$

де $2 \leq g \leq p - 2$.

Вибір параметра p здійснюється з використанням тесту простоти Мілера–Рабіна, що гарантує з високою ймовірністю його простоту [13, 14]. Для цього p формується як непарне випадкове число заданої довжини (наприклад, 1024 біт) і перевіряється за умовами:

$$p - 1 = 2^s t, \;\; t - \text{непарне}, s \geq 1, \qquad (2.11)$$

після чого перевіряється виконання критерію:

$$a^t \equiv \pm 1 \ (mod\ p) \text{ або } a^{2^r t} \equiv -1 \ (mod\ p) \tag{2.12}$$

для кількох баз $a \in [2, p-2]$. Якщо хоча б один тест не виконується, число відкидається.

Користувач А генерує випадкове приватне число:

$$x_A \in [2, p-2], \tag{2.13}$$

використовуючи криптографічно стійкий генератор випадкових бітів, що відповідає стандарту ISO/IEC 18031 [20], який забезпечує непередбачуваність і рівномірний розподіл значень. Відповідний відкритий ключ обчислюється за формулою:

$$y_A = g^{x_A} \ mod\ p. \tag{2.14}$$

Аналогічно користувач В генерує приватний параметр

$$x_B \in [2, p-2], \tag{2.15}$$

і обчислює відкритий ключ:

$$y_B = g^{x_B} \ mod\ p \tag{2.16}$$

Користувач А передає y_A користувачу В, а користувач В надсилає y_B користувачу А через відкритий канал. Незважаючи на відкритість переданих значень, неможливо відновити приватні ключі x_A, x_B, оскільки для цього потрібно розв’язати задачу дискретного логарифмування:

$$x = \log_g y \ (mod\ p), \tag{2.17}$$

яка є обчислювально нездійсненною для великих p [13,14].

Після обміну відкритими ключами обидві сторони незалежно обчислюють спільний секретний ключ K:

$$K_A = (y_B)^{x_A} \, mod \, p = g^{x_A x_B} \, mod \, p, \qquad (2.18)$$
$$K_B = (y_A)^{x_B} \, mod \, p = g^{x_A x_B} \, mod \, p.$$

Таким чином, обидві сторони отримують однакове значення:

$$K = g^{x_A x_B} \, mod \, p. \qquad (2.19)$$

яке ніколи не передається по каналу зв'язку [13,14].

Для подальшого використання в системі автентифікованого шифрування спільний секрет K виступає джерелом для формування 1024-бітного ключа K_s, який потім ущільнюється до 64-бітного значення K_0.

2.3 Процедура генерування секретного ключа

Процес формування секретного ключа є початковим етапом. Його метою є створення 64-бітного ключа K_0 з початкового 1024-бітного секретного ключа, який отримано за протоколом Діффі-Геллмана.

Початкове 1024-бітне число K ділиться на чотири рівні частини по 256 біт кожна:

$$K = (K_1, K_2, K_3, K_4), \;\; K_i \in \; [0, 2^{256} - 1]. \qquad (2.20)$$

Кожна частина далі поділяється на чотири 64-бітні компоненти:

$$K_1 = (a_1, b_1, c_1, d_1), K_2 = (a_2, b_2, c_2, d_2), K_3 = (a_3, b_3, c_3, d_3), \qquad (2.21)$$
$$K_4 = (a_4, b_4, c_4, d_4),$$

де $(a_i, b_i, c_i, d_i) \in [0, 2^{64} - 1]$.

Далі кожна частина подається як кватерніон [17]:

$$\begin{aligned} K_1 &= a_1 + b_1 i + c_1 j + d_1 k, \\ K_2 &= a_2 + b_2 i + c_2 j + d_2 k, \\ K_3 &= a_3 + b_3 i + c_3 j + d_3 k, \\ K_4 &= a_4 + b_4 i + c_4 j + d_4 k. \end{aligned} \tag{2.22}$$

Далі виконується послідовне множення кватерніонів:

$$D_1 = K_1 K_2 = \begin{cases} a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2 = w_1, \\ a_1 b_2 + b_1 a_2 + c_1 d_2 - d_1 c_2 = z_1, \\ a_1 c_2 - b_1 d_2 + c_1 a_2 + d_1 b_2 = x_1, \\ a_1 d_2 + b_1 c_2 - c_1 b_2 + d_1 a_2 = k_1. \end{cases}$$

$$D_2 = K_3 K_4 = \begin{cases} a_3 a_4 - b_3 b_4 - c_3 c_4 - d_3 d_4 = w_2, \\ a_3 b_4 + b_3 a_4 + c_3 d_4 - d_3 c_4 = z_2, \\ a_3 c_4 - b_3 d_4 + c_3 a_4 + d_3 b_4 = x_2, \\ a_3 d_4 + b_3 c_4 - c_3 b_4 + d_3 a_4 = k_2. \end{cases} \tag{2.23}$$

$$Q = D_1 D_2 = \begin{cases} w_1 w_2 - z_1 z_2 - x_1 x_2 - k_1 k_2 = q_1 \\ w_1 z_2 + z_1 w_2 + x_1 k_2 - k_1 x_2 = q_2 \\ w_1 x_2 - z_1 k_2 + x_1 w_2 + k_1 z_2 = q_3 \\ w_1 k_2 + z_1 x_2 - x_1 z_2 + k_1 w_2. = q_4 \end{cases}$$

Усі операції виконуються за модулем 2^{64} [21].

$$K_0 = (q_1 + q_2 + q_3 + q_4) \; mod \; 2^{64} \tag{2.24}$$

Таким чином, початковий 1024-бітний ключ згортається до 64-бітного представлення.

Процедура ущільнення має такі властивості:

– K_0 є геш-значенням для коду K.

– Нелінійність – кватерніонне множення створює складну залежність між вхідними частинами, що унеможливлює лінійне відновлення початкових компонент.

– Незворотність – через геш-значення.

– Ентропійність – усі 1024 біти початкового ключа впливають на кінцеве значення K_0, забезпечуючи повне змішування K_0.

2.4 Обчислення поворотної матриці

Після отримання 64-бітного ключа K_0, що є результатом багатоступеневої процедури згортання початкового 1024-бітного секретного значення, виконується його перетворення у кватерніон, на основі якого формується поворотна матриця.

Цей етап має важливе значення, оскільки саме кватерніонна форма представлення ключа забезпечує можливість побудови ортогональної поворотної матриці, що використовується для реалізації першої стадії шифрування – блочного матричного перетворення.

Отримане 64-бітне число K_0 розбивається на чотири 16-бітні складові:

$$K_0 = (w, x, y, z), \tag{2.25}$$

де $w, x, y, z \in [0, 2^{16})$.

Кожна з цих компонент визначає параметри обертання у чотиривимірному просторі, а їх поєднання утворює кватерніон:

$$q = w + xi + yj + zk, \tag{2.26}$$

що задовольняє класичні правила множення базових елементів:

$$i^2 = j^2 = k^2 = ijk = -1 \tag{2.27}$$

Таким чином, кватерніон q є узагальненням поняття комплексного числа та забезпечує компактне і зручне математичне представлення просторових обертань.

Норма кватерніона визначається як сума квадратів його компонент за модулем обраного простого числа $p = 65537$ [22]:

$$N = (w^2 + x^2 + y^2 + z^2)\ mod\ p \tag{2.28}$$

Норма характеризує «довжину» кватерніона у чотиривимірному просторі. Для побудови ортогональної поворотної матриці необхідно, щоб кватерніон був одиничним, тобто мав норму, рівну 1. Якщо ця умова не виконується, потрібно виконати нормування.

Нормування полягає у приведенні довжини кватерніона до одиниці. Для цього вводиться масштабувальний коефіцієнт t, який задовольняє рівняння:

$$t^2 \equiv N^{-1}\ (mod\ p), \tag{2.29}$$

де N^{-1} – мультиплікативно обернене до N число в полі Z_p.

Нормований кватерніон обчислюється за формулою:

$$\hat{q} = t \cdot q\ mod\ p, \tag{2.30}$$

після чого забезпечується властивість:

$$||\hat{q}||^2 \equiv 1\ (mod\ p), \tag{2.31}$$

тобто $\hat{q}$ є одиничним нормованим кватерніоном [22].

У випадку, якщо N не є квадратичним лишком у полі Z_p, тобто рівняння для t немає розв’язку, виконується мінімальна корекція однієї з компонент (зазвичай z) на невелике значення δ до тих пір, поки оновлена норма N' не стане

квадратичним лишком. Це гарантує існування квадратного кореня та можливість коректної нормалізації.

Такий підхід мінімізує спотворення початкових даних і зберігає стійкість алгоритму.

Після нормалізації кватерніона на його основі формується ортогональна поворотна матриця $\mathbf{\Gamma}(\hat{q})$, елементи якої визначаються за стандартною кватерніонною формулою [23]:

$$\mathbf{\Gamma}(\hat{q}) = \begin{bmatrix} 1-2(y^2+z^2) & 2(xy-wz) & 2(xz+wy) \\ 2(xy+wz) & 1-2(x^2+z^2) & 2(yz-wx) \\ 2(xz-wy) & 2(yz+wx) & 1-2(x^2+y^2) \end{bmatrix} \quad (2.32)$$

Ця поворотна матриця описує тривимірне обертання, яке відповідає напрямку та величині, заданим кватерніоном $\hat{q}$. Її головною властивістю є ортогональність:

$$\Gamma(\hat{q})^{\mathrm{T}}\Gamma(\hat{q}) \equiv I \ (mod\ p), \quad (2.33)$$

де I – одинична матриця. Це означає, що обернена матриця для розшифрування є транспонованою для матриці обертання.

Використання поворотної матриці як блочного шифру забезпечує рівномірне розподілення впливу кожного біта ключа на елементи зашифрованого блоку. Завдяки цьому досягається висока дифузія та стійкість до статистичних атак. Крім того, ортогональність гарантує, що обернення процесу не призводить до накопичення похибок при багатократних обчисленнях у полі $\mathbf{Z}_p$.

Таким чином, поворотна матриця $\mathbf{\Gamma}(\hat{q})$ виступає ядром блочного шифру, яке зв’язує алгебраїчні властивості кватерніонів із практичними вимогами до безпеки криптографічних систем [23].

2.5 Стандарти генерування випадкових чисел

У сучасній криптографії та інформаційній безпеці генерування випадкових чисел є фундаментальною складовою захищених обчислень. Від якості та непередбачуваності випадкових чисел залежить криптостійкість алгоритмів шифрування, надійність ключів, автентичність цифрових підписів і загальна безпека інформаційного обміну.

Для забезпечення високого рівня безпеки створено низку міжнародних і національних стандартів, що регламентують методи генерування випадкових і псевдовипадкових чисел, визначають вимоги до джерел ентропії, критерії статистичної оцінки випадковості та процедури валідації генераторів. Використання таких стандартів забезпечує криптографічну стійкість систем, що реалізують цифрові підписи, шифрування даних і протоколи обміну ключами.

До найвідоміших належать:

– NIST SP 800-90A/B/C – стандарти, що описують архітектуру детермінованих генераторів випадкових чисел (DRBG) та вимоги до ентропійних джерел.

– ISO/IEC 18031 – міжнародний стандарт, який визначає принципи, вимоги та алгоритмічні засоби генерування випадкових і псевдовипадкових чисел у криптографічних додатках.

– FIPS 140-3 і FIPS 186-5 – стандарти, що визначають вимоги до криптографічних модулів і процесів генерування ключів для алгоритмів цифрового підпису.

– RFC 4086 – рекомендації IETF щодо безпечної генерування випадкових чисел у мережевих протоколах.

Для протоколу обміну ключами Діффі–Геллмана було обрано міжнародний стандарт ISO/IEC 18031, який визначає вимоги до криптографічних генераторів випадкових бітів (RBG, Random Bit Generators).

Стандарт ISO/IEC 18031 [20] описує принципи побудови, функціонування та оцінки генераторів випадкових бітів (ГВБ), що застосовуються у

криптографічних системах. Його метою є забезпечення криптографічної надійності – тобто неможливості передбачити вихідні значення навіть за умови часткового розкриття внутрішнього стану генератора.

Документ визначає дві основні категорії генераторів:

1) Фізичні генератори випадкових чисел (TRNG, True Random Number Generators) – ґрунтуються на апаратних джерелах ентропії, таких як електричний шум, теплові флуктуації або квантові процеси.

2) Детерміновані генератори випадкових чисел (DRBG, Deterministic Random Bit Generators) – створюють послідовності бітів на основі криптографічних алгоритмів (наприклад, AES, SHA-2, HMAC, CTR), використовуючи початкове насіння (seed) з високою ентропією.

Генератор випадкових чисел відповідно до ISO/IEC 18031 має три основні компоненти:

1) Джерело ентропії – забезпечує отримання непередбачуваних фізичних подій для ініціалізації генератора.

2) Механізм обробки ентропії – перетворює «сирі» випадкові дані в однорідну, рівномірно розподілену послідовність бітів.

3) Детермінований криптографічний механізм – створює довгі криптографічно стійкі послідовності бітів із короткого насіння, підтримуючи властивості непередбачуваності та незалежності результатів [20].

Відповідно до стандарту ISO/IEC 18031, генератор повинен відповідати таким вимогам:

– Непередбачуваність. Знання попередніх або наступних вихідних бітів не дозволяє обчислити поточні значення.

– Стійкість до компрометації. Навіть у разі витоку внутрішнього стану генератора попередні вихідні дані не повинні бути відновлені.

– Перевірюваність. Генератор має проходити статистичні тести випадковості (наприклад, NIST SP 800-22 або Diehard tests).

– Самовідновлення. Генератор повинен мати здатність до оновлення насіння для підвищення ентропії в процесі роботи.

– Криптографічна узгодженість. Застосовувані механізми повинні відповідати алгоритмам, визнаним безпечними на міжнародному рівні (AES, HMAC, SHA-3 тощо) [20].

Згідно з ISO/IEC 18031, генератори випадкових чисел застосовуються для:

– створення ключів шифрування та ініціалізаційних векторів (IV);

– формування приватних ключів цифрових підписів;

– генерування сеансових ключів у протоколах обміну інформацією;

– створення псевдовипадкових послідовностей для автентифікаційних механізмів;

– формування одноразових паролів (OTP), PIN-кодів і токенів доступу.

Крім того, ISO/IEC 18031 передбачає, що криптографічно безпечний генератор має підтримувати періодичне оновлення внутрішнього стану для запобігання накопиченню корельованих результатів та зниженню рівня ентропії.

Переваги використання стандарту ISO/IEC 18031 полягають у наступному:

– забезпечує уніфікований підхід до побудови криптографічно стійких генераторів;

– підвищує довіру до безпеки реалізованих систем;

– дозволяє проходити міжнародну сертифікацію криптографічних рішень;

– забезпечує сумісність із іншими стандартами, такими як ISO/IEC 19790 (вимоги до криптографічних модулів) та ISO/IEC 29192 (легковагова криптографія) [20].

2.6 Тест простоти Мілера-Рабіна

Для ефективного та надійного визначення простоти чисел широко застосовується тест Мілера-Рабіна – ймовірнісний алгоритм, який дозволяє перевіряти, чи є число простим, з високою ймовірністю. Цей тест використовується не лише для загальної генерування простих чисел, а й безпосередньо в таких криптографічних протоколах, як протокол Діффі-

Геллмана, де прості числа великого порядку слугують основою для обчислення ключів обміну та забезпечують стійкість шифрування до атак [24].

Мала теорема Ферма стверджує, що якщо m – просте число, то $a^{m-1} \equiv 1(mod\ m)$ для будь-якої основи a, яка є взаємно простою з р. Цю умову a^{p-1} можна використовувати як тест на простоту. Якщо ця умова виконується, то ми кажемо, що m – ймовірне просте число з основою a. На жаль, існують складені числа, m такі що $a^{p-1} \equiv 1(mod\ m)$ для всіх a чисел , які є взаємно простими з m (такі складені числа m називаються числами Кармайкла). Найменше число Кармайкла дорівнює $561 = 17 \times 33$. Застосовуючи Малу теорему Ферма до числа Кармайкла m, ми можемо сплутати його з простим числом [24].

Тест Мілера-Рабіна є набагато кращим інструментом для ідентифікації чисел Кармайкла. Він є більш ефективним ймовірнісним тестом, в якому використовується критерій, в кінцевому рахунку, оснований на факті, що для простого модуля квадратними коренями з одиниці є лише числа ± 1, а для складеного непарного модуля $n = uv$, $(uv) = 1$, число таких коренів більше двох.

Нехай n – непарне натуральне число. Тоді можна записати:

$$n - 1 = 2^s t, \tag{2.34}$$

де t – непарне і $s \geq 1$.

Якщо число n – непарне натуральне число. Тоді можна записати:

$$a^{n-1} \equiv 1(mod\ n), \tag{2.35}$$

при $(a,n) = 1$.

Тому квадратні корені з одиниці мають вигляд:

$$a^{(n-1)/2} = \pm 1(mod\ n), \tag{2.36}$$

де показник рівний $2^{s-1}t$ [24].

Це означає, що в послідовності $a^t, a^{2t}, \dots, a^{2^{s-1}t}$ лишків за простим модулем, які є послідовними квадратами числа a^t, або з'явиться$-1(mod\ n)$, або всі ці лишки порівнянні з одиницею, тобто $a^t = 1(mod\ n)$. Зазначимо, що при простому *n* лівіше $-1(mod\ n)$ можуть розташовуватися лише лишки не рівні $\pm 1(mod\ n)$.

Якщо *n* – складене, то можливі й інші варіанти, оскільки в цьому випадку крім ± 1 існують інші корені з одиниці за модулем *n*.

Заснований на даному зауваженні тест Мілера-Рабіна полягає в тому, що:

1) псевдовипадково вибираємо залишок $a \in \{2, \dots, n-1\}$ і перевіряємо умову $(a, n) = 1$. Якщо умова не виконана, значить, n – складене і робота закінчена;

2) обчислюємо $a^t = mod\ n$. Якщо $a^t = \pm 1\ mod\ n$, то не виключено, що число *n* – просте і необхідно перейти на початок, щоб повторити тест для іншої основи;

3) обчислюємо послідовно лишки чисел $a^{2t}, \dots, a^{2^{s-1}t}$,за модулем *n*, поки не з'явиться (-1) , або не вичерпається список;

4) якщо (-1) знайдено в списку, то не виключено, що число n – просте і необхідно перейти на початок, щоб повторити тест для іншої основи;

5) якщо жодне число із списку не порівнянно з (-1), то число *n* – складене і необхідно закінчити роботу [24].

Як і для інших імовірнісних тестів, існують складені числа *n*, які, для відповідних основ *a*, проходять даний тест.

Назвемо число $n = 2^s t + 1$, де $s \geq 1,\ t$ – непарне, сильним псевдопростим за основою $a \neq 1\ mod\ n$, (*a*,*n*) = 1, якщо виконується одна з двох умов: $a^t = \pm 1\ mod\ n$, або в послідовності $a^{2t}, \dots, a^{2^{s-1}t}$ існує число, порівнянне з –1 за модулем *n*.

Виявляється, можна показати, що для будь-якого *a*, (*a*,*n*) = 1, існує нескінченно багато сильних псевдопростих чисел *n* за основою *a*.

Приклади: a = 7, n = 25; a = 5, n = 781.

Можна довести такі основні властивості сильних псевдопростих чисел:

1) число n, сильне псевдопросте за основою a, є ейлеровим псевдопростим за тією ж основою;

2) якщо непарне складене число n є сильним псевдопростим за основою a, то загальна кількість основ, за якою це число є сильним псевдопростим, не перевищує $(n-1)/4$.

Тому можна стверджувати, що при повторенні випробувань тесту Мілера-Рабіна k раз ймовірність невідбракування складеного числа $\leq (1/4)^k$.

Крім того, виявляється, кількість повторень тесту, достатню для практичних додатків, можна обмежити величиною $2log_2^2 n$.

Простоту невеликих простих чисел можна довести, використовуючи декілька раніше вказаних основ [24].

2.7 Висновки до розділу

У результаті було побудовано узагальнений протокол обміну інформацією, який забезпечує повний цикл захищеної взаємодії між двома сторонами: від ініціалізації клієнтів і розподілу відкритих ключів до взаємної автентифікації, узгодження спільного секрету та передавання зашифрованих і автентифікованих повідомлень.

Сервер виконує роль PKDI і не бере участі у формуванні спільного секрету, що зменшує наслідки його можливої компрометації.

Ключовою особливістю є суміщення автентифікації та узгодження параметрів протоколу Діффі-Геллмана в єдиному обмінному циклі. Цифрові підписи на основі Ed25519 та nonce-значення R_A, R_B, включені до підписаних повідомлень разом з g^a та g^b, забезпечують взаємну автентифікацію та захист від атак повтору і MITM, не збільшуючи кількість кроків протоколу.

Реалізація протоколу Діффі-Геллмана базується на використанні великого простого модуля p і генератора g групи $\mathbf{Z}_p^*$. Простота p перевіряється тестом

Мілера–Рабіна, а приватні показники формуються криптографічно стійким генератором відповідно до стандарту ISO/IEC 18031.

Це забезпечує необхідну непередбачуваність та криптостійкість етапу узгодження спільного секрету.

Отриманий 1024-бітний спільний секрет згортається до 64-бітного робочого ключа K_0 за допомогою кватерніонного множення. Така процедура створює нелінійне, незворотне перетворення з повним змішуванням усіх бітів початкового значення, унаслідок чого K_0 можна розглядати як геш-образ для секрету K.

На основі ключа K_0 формується нормований кватерніон $\hat{q}$, з якого будується ортогональна поворотна матриця $\mathbf{\Gamma}(\hat{q})$ над полем за модулем $p = 65537$.

Вона використовується як ядро блочного шифру на першому етапі обробки повідомлення та забезпечує високу дифузію й зворотність перетворень, оскільки обернена матриця дорівнює транспонованій.

У розділі також було обґрунтовано вибір стандартів і алгоритмів для генерування випадкових чисел (ISO/IEC 18031) і перевірки простоти (Мілер–Рабін), що узгоджує протокол із сучасними міжнародними вимогами до криптографічних систем.

Таким чином, розроблений протокол обміну інформацією поєднує розподіл відкритих ключів через сервер-каталог, взаємну автентифікацію сторін, узгодження спільного секрету за Діффі–Геллманом та кватерніонне ущільнення ключа з побудовою поворотної матриці.

Це створює цілісну основу для подальшої програмної реалізації захищеного каналу передавання інформації.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОТОКОЛУ ОБМІНУ ІНФОРМАЦІЄЮ

3.1 Обґрунтування вибору мови програмування та середовища розробки

Процес вибору мови програмування є одним із найважливіших етапів створення будь-якого програмного забезпечення. Від цього рішення залежить ефективність реалізації алгоритмів, швидкість розробки, подальше масштабування та підтримка проєкту. Кожна мова має власну філософію, синтаксис, набір інструментів і сферу застосування, тому обґрунтований вибір базується на відповідності між можливостями мови та вимогами конкретного проєкту. У цьому контексті мова Python посідає провідне місце завдяки своїй універсальності, простоті та потужному інструментарію [25].

Python є високорівневою, інтерпретованою та об’єктно-орієнтованою мовою програмування. Її синтаксис наближений до природної мови, що значно полегшує процес навчання і розробки. Простота кодування сприяє зменшенню кількості помилок, підвищенню швидкості написання програм та полегшенню супроводу програмного забезпечення. Це робить Python не лише зручним для новачків, а й ефективним для професійних розробників, які прагнуть швидкої реалізації складних проєктів.

Однією з ключових переваг Python є висока швидкість розробки. Завдяки підтримці об’єктно-орієнтованої парадигми (ООР), розробники можуть створювати багаторазово використовувані класи й модулі, що значно скорочує час розробки складних систем. Крім того, інтерактивне середовище REPL (Read–Evaluate–Print Loop) дозволяє тестувати фрагменти коду в реальному часі, що робить Python зручним інструментом для експериментів та прототипування [25].

Ще однією перевагою є автоматичне керування пам’яттю та висока продуктивність. Python має вбудований механізм збору «сміття» (garbage collector), який ефективно очищує невикористані області пам’яті. Завдяки компіляції в байткод і використанню середовища PyPy, виконання програм досягає високої швидкості при збереженні гнучкості інтерпретованої мови.

Python також відомий своєю кросплатформеністю, адже один і той самий код може виконуватись без змін на Windows, macOS і Linux. Це дозволяє створювати універсальні рішення, які легко адаптуються під різні операційні системи без додаткових витрат на переписування програмного коду.

Особливої уваги заслуговує наявність потужних бібліотек і фреймворків, які охоплюють практично всі сфери розробки – від веб-програмування (Django, Flask) до машинного навчання й штучного інтелекту (TensorFlow, PyTorch, Scikit-learn), аналізу даних (NumPy, Pandas), кібербезпеки (Scapy, PyCrypto) та розробки ігор (Pygame, Godot). Така широта застосування робить Python універсальною мовою, придатною для реалізації як наукових експериментів, так і промислових систем [25].

Ще одним вагомим аргументом на користь Python є величезна спільнота розробників, яка постійно розвиває екосистему мови, створює нові бібліотеки, фреймворки та освітні ресурси. Завдяки цьому Python має стабільну підтримку, регулярні оновлення та велику кількість навчальних матеріалів, що забезпечує надійність і довгострокову перспективу використання.

Крім технічних переваг, Python є економічно вигідним рішенням. Мова є повністю відкритою (open-source) і безкоштовною для використання навіть у комерційних цілях. Зменшення кількості рядків коду, швидкість розробки та повторне використання компонентів сприяють скороченню витрат на створення та підтримку програмного забезпечення [25].

У контексті розробки криптографічних систем та кіберзахисних інструментів вибір Python є логічно обґрунтованим. Його багаті математичні бібліотеки, простота обробки великих чисел, підтримка роботи з матрицями та структурами даних роблять його ефективним інструментом для реалізації алгоритмів шифрування, аналізу трафіку та побудови моделей безпеки.

Отже, вибір Python як основної мови програмування обумовлений поєднанням таких чинників, як простота використання, висока швидкість розробки, універсальність, потужна екосистема бібліотек, активна спільнота та економічна доцільність. Усі ці переваги роблять Python оптимальним вибором

для створення сучасних програмних продуктів у галузі інформаційної безпеки та криптографії [25].

Для ефективної розробки програмного забезпечення на мові Python важливо обрати середовище, яке поєднує зручність, швидкодію та широкий набір інструментів. Найбільш оптимальним рішенням у цьому контексті є Visual Studio Code (VS Code) — легке, безкоштовне й кросплатформне середовище розробки, створене компанією Microsoft.

VS Code поєднує простоту текстового редактора з функціональністю повноцінної IDE. Завдяки відкритому вихідному коду і ліцензії MIT він доступний для безкоштовного використання у навчальних, наукових та комерційних проєктах. Однією з головних переваг цього середовища є його висока продуктивність і мінімальні вимоги до системних ресурсів – VS Code швидко запускається, працює стабільно навіть на слабких пристроях і не перевантажує систему, що особливо важливо при розробці великих програмних рішень [26].

Середовище має потужну систему розширень, які дозволяють адаптувати VS Code під конкретні потреби розробника. Серед них — офіційний модуль Python Extension, що забезпечує підтримку автодоповнення коду, підсвічування синтаксису, інтегроване налагодження, тестування, форматування та роботу з середовищами Anaconda чи Jupyter Notebook. Завдяки цьому VS Code стає гнучким інструментом як для початківців, так і для досвідчених розробників.

Особливу увагу заслуговує функція IntelliSense, яка надає інтелектуальні підказки щодо змінних, функцій, класів та методів. Це значно підвищує ефективність програмування, знижує кількість помилок і прискорює процес написання коду.

Вбудована підтримка Git є ще однією важливою перевагою VS Code. Вона дозволяє виконувати всі операції з контролю версій — коміти, злиття, створення гілок, надсилання змін у віддалений репозиторій — безпосередньо з інтерфейсу середовища. Це робить спільну роботу над проєктом простішою й зручнішою [26].

Крім того, VS Code є повністю кросплатформним: він підтримує роботу на Windows, macOS та Linux, що дозволяє розробникам працювати у звичному середовищі незалежно від операційної системи.

Таким чином, вибір Visual Studio Code як основного середовища розробки для Python зумовлений його легкістю, безкоштовністю, високою швидкодією, широкими можливостями налаштування, інтеграцією з Git та потужною системою розширень. Усе це робить його одним із найкращих інструментів для створення сучасних і масштабованих програмних рішень [26].

3.2 Архітектура програмного засобу захищеного передавання інформації

Розроблена архітектура програмного засобу базується на модульному підході, що забезпечує логічну структуризацію компонентів, гнучкість у розширенні функціоналу та простоту супроводу. Основна ідея полягає в розділенні відповідальності між рівнями системи, де кожен модуль виконує власні функції - від взаємодії з користувачем до обробки та захисту даних під час їх передавання.

Архітектура складається з чотирьох взаємопов’язаних рівнів:

1) GUI-модуль (інтерфейс користувача) відповідає за відображення повідомлень, введення тексту та взаємодію користувача із системою. Завдяки використанню бібліотеки PySide6 забезпечується сучасний графічний інтерфейс і асинхронна передача даних у фоновому режимі. GUI є початковою точкою комунікації між користувачем і програмою.

2) Мережевий модуль (net) реалізує логіку встановлення з’єднань, обмін службовими повідомленнями та підтримання сесії зв’язку. На цьому рівні відбувається передача ідентифікаторів користувачів, синхронізація параметрів, обмін публічними ключами підпису Ed25519 та ретрансляція повідомлень протоколу взаємної автентифікації. Мережевий модуль визначає роль кожного клієнта в процесі рукостискання, забезпечує обмін криптографічними параметрами та здійснює маршрутизацію зашифрованих повідомлень до

адресата. Після завершення встановлення сесії він передає дані криптографічному модулю для подальшої обробки.

3) Криптографічний модуль (crypto) виконує основні операції безпеки. У ньому реалізовано механізми формування та перевірки підписів, узгодження параметрів протоколу Діффі-Геллмана, генерацію ефемерних ключів, встановлення спільного секрету, побудову ортогональної матриці та виконання процесів шифрування й розшифрування повідомлень. Важливо, що обмін ключовими параметрами супроводжується взаємною автентифікацією сторін, під час якої публічні ключі підпису використовуються для підтвердження достовірності відправника, а структура повідомлень містить ідентифікатори клієнтів та криптографічні підписи. Це унеможливлює підміну та втручання у процес обміну ключами, забезпечуючи стійкість протоколу до атак типу MITM.

4) Relay Server виконує роль проміжної ланки для пересилання даних між клієнтами. Він приймає підключення, ретранслює повідомлення та не здійснює зберігання інформації, що унеможливлює витік конфіденційних даних. Такий підхід гарантує мінімальну залежність клієнтів від сервера й підвищує загальну безпечність системи.

Загальну архітектуру програмного засобу подано на рис. 3.1.

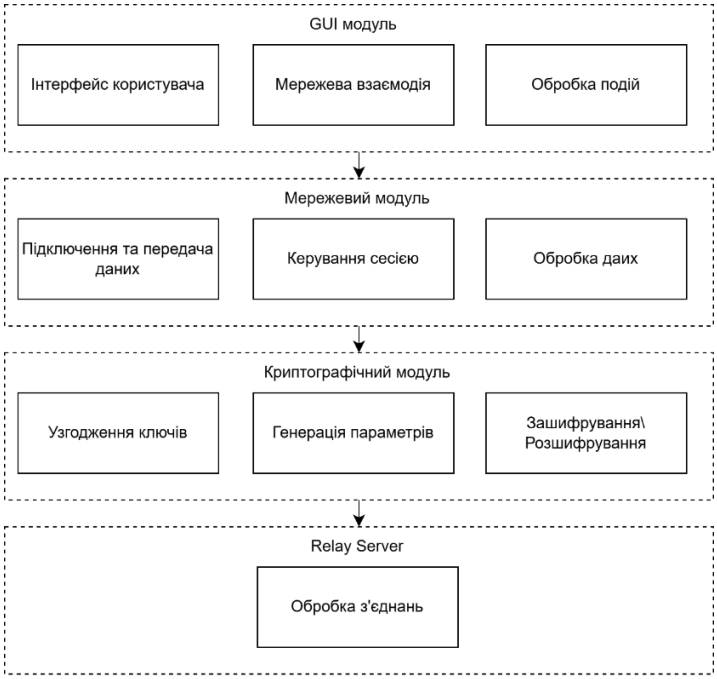


Рисунок 3.1 – Архітектура програмного засобу захищеного передавання інформації

Обмін повідомленнями відбувається послідовно: користувач створює повідомлення через графічний інтерфейс, далі мережевий модуль передає його до криптографічного блоку, де здійснюється шифрування на основі попередньо сформованого спільного ключа. Зашифровані дані надсилаються через реле-сервер іншому клієнту, який виконує зворотну операцію — розшифрування та відображення повідомлення на екрані. Важливо підкреслити, що перед початком обміну даними клієнти проходять етап узгодження параметрів та взаємної автентифікації, у рамках якого формується спільний секрет та підтверджується достовірність сторін.

Запропонована архітектура відзначається високим рівнем модульності, що дозволяє незалежно оновлювати або замінювати компоненти, додавати нові методи шифрування та адаптувати систему під різні сценарії використання. Така структура забезпечує надійність роботи, масштабованість і зручність подальшого розвитку програмного засобу, а інтеграція механізмів взаємної автентифікації суттєво підвищує загальний рівень безпеки та захисту від активних атак на процес встановлення сесії.

3.3 Розробка алгоритму протоколу обміну інформацією

Протокол обміну інформацією реалізовано у вигляді послідовності етапів, кожен із яких виконує чітко визначену функцію та забезпечує безпеку, коректність і узгодженість параметрів між сторонами. Загальна логіка функціонування системи базується на двох ключових процедурах: взаємній автентифікації та встановленні спільного сеансового ключа і перевірці коректності сформованих параметрів та матриці обертання, що використовується під час шифрування даних.

На першому етапі клієнти виконують ініціалізацію параметрів та криптографічних структур, необхідних для встановлення захищеної сесії. Кожен із учасників генерує або завантажує довготривалі ключі підпису, формує параметри протоколу Діффі-Геллмана та ініціює обмін службовими повідомленнями. На цьому етапі здійснюється передача ідентифікаторів та

публічних ключів підпису між сторонами через реле-сервер, який виконує роль каналу доставки, не здійснюючи криптографічних операцій та не маючи доступу до приватних ключів.

Після узгодження базових параметрів сторони обмінюються ефемерними ключами, згенерованими лише для поточного сеансу. Кожен клієнт підписує свої параметри приватним ключем, а отримана інформація перевіряється на коректність та автентичність, що унеможливлює їх підміну.

На завершальному етапі сторони підтверджують немодифікованість даних, виявляючи можливі атаки типу «MITM». Після успішних перевірок обчислюється спільний секрет, і клієнти переходять до подальших обчислень. Алгоритм взаємної автентифікації та встановлення ключа показано на рис. 3.2.

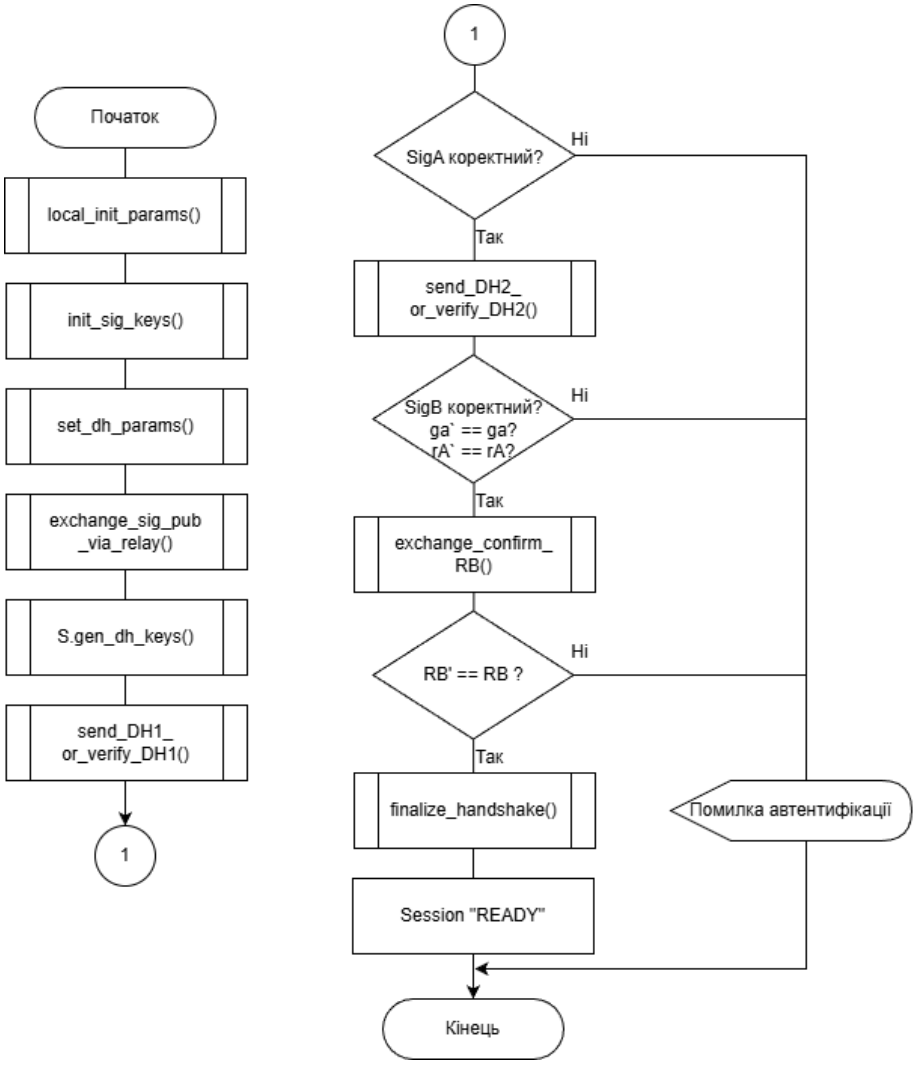


Рисунок 3.2 – Алгоритм взаємної автентифікації та встановлення спільного сеансового ключа

Після завершення автентифікації та обчислення спільного секрету сторони формують стислий ключовий матеріал, який використовується для побудови нормалізованої структури, на основі якої генерується ортогональна матриця повороту. Формування матриці здійснюється незалежно на кожній зі сторін, тому важливим етапом є перевірка узгодженості отриманих параметрів між учасниками обміну.

На цьому етапі здійснюється перевірка ортогональності сформованої матриці, що дозволяє визначити відповідність її властивостей очікуваній структурі та виявити помилки обчислення або маніпуляції з даними. Якщо порівняння результатів обчислень обох сторін є успішним, система переходить у стан готовності до шифрування повідомлень. У протилежному випадку механізм генерує повідомлення про помилку узгодження параметрів, що призводить до повторного виконання процедури або завершення сеансу.

Узагальнену логіку узгодження параметрів та перевірки ортогональності наведено на рис. 3.3.

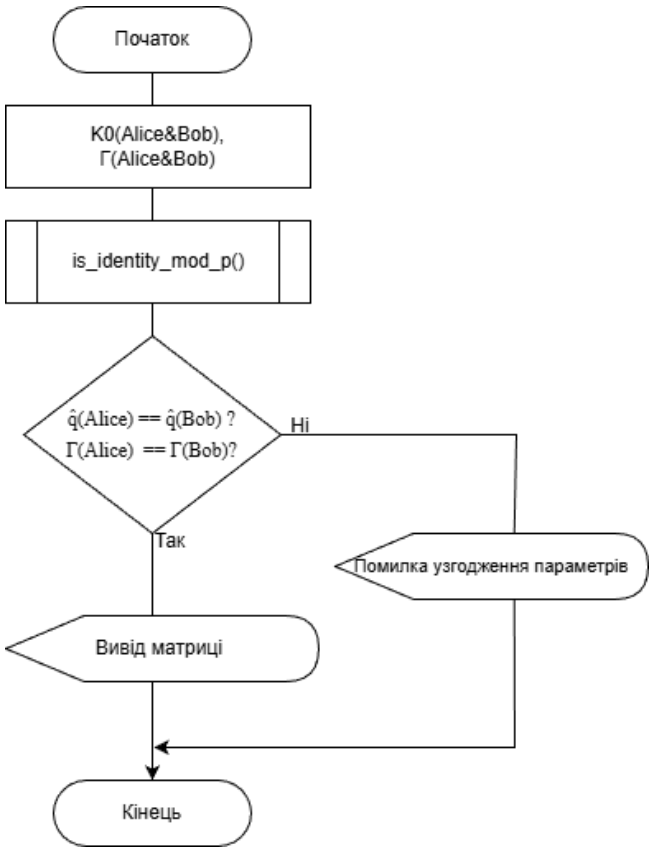


Рисунок 3.3 – Алгоритм узгодження параметрів та перевірки ортогональності поворотної матриці

Після успішного узгодження параметрів і підтвердження коректності матриці система переходить у стан готовності до обміну зашифрованими даними. На цьому етапі сформована матриця використовується для виконання операцій шифрування та розшифрування, що забезпечує цілісність та конфіденційність інформації, яку обмінюють сторони.

Таким чином, запропонований алгоритм поєднує криптографічні механізми автентифікації та обмін параметрами, механізм захисту від активних атак та математичні процедури перевірки коректності обчислень. Це забезпечує високу надійність обміну, захищеність від підміни та відтворення повідомлень, а також узгодженість параметрів на всіх етапах формування сесії.

3.4 Програмна реалізація

Після розробки протоколу обміну інформацією, який забезпечує формування захищеного каналу між двома сторонами на основі алгоритму Діффі-Геллмана та кватерніонної поворотної матриці, було створено його програмну реалізацію у вигляді модульної системи. Програмний засіб складається з чотирьох головних модулів: relay_server.py, net_client.py, secure_session.py та gui.py [27, 28].

Кожен з них виконує окрему функціональну роль у побудові безпечного з’єднання, починаючи від транспортного рівня та завершуючи графічним інтерфейсом користувача. Додатково на рівні прикладного інтерфейсу використовується модуль lobby.py, який реалізує «лобі» для вибору співрозмовника перед встановленням захищеного чату.

Розробка системи проводилася з урахуванням принципів асинхронності, модульності та криптографічної незалежності. Серверна частина функціонує як нейтральний ретранслятор, клієнтська – як ініціатор та обробник обміну, а модуль захищеної сесії реалізує математичну частину формування ключа та шифрування.

Для забезпечення взаємної автентифікації сторін у протоколі додатково використано довготривалі ключі підпису Ed25519, на основі яких підписуються

параметри обміну Діффі–Геллмана. Це дозволяє не лише сформувати спільний секрет, а й підтвердити, що всі параметри були згенеровані саме легітимними користувачами.

Relay-сервер виконує функцію транспортного вузла, який приймає вхідні з'єднання від клієнтів та передає повідомлення між ними. Його архітектура реалізована на основі бібліотеки asyncio, що дозволяє організувати неблокуючу багатоклієнтську комунікацію. На відміну від класичних серверів, релей не зберігає жодних даних користувачів та не здійснює розшифрування повідомлень – він лише ретранслює JSON-пакети.

Додатково на сервері підтримується «директорія» публічних ключів підпису користувачів і карта статусів (idle / busy), що використовуються виключно для маршрутизації та організації лобі, але не впливають на криптографічну стійкість протоколу.

На початку визначаються основні параметри мережі та створюються структури даних для зберігання стану клієнтів:

```
HOST = os.getenv("HOST", "26.228.177.167")
PORT = int(os.getenv("PORT", "8765"))
class Relay:
    def __init__(self):
        self.directory: Dict[str, str] = {}  # ID → Ed25519 pub
        self.clients: Dict[str, ClientState] = {}
        self.status: Dict[str, str] = {}     # "idle" / "busy"
```

Реалізація основної логіки взаємодії з клієнтами відбувається у функції handle. Ця функція створює обробник для кожного нового підключення, читає вхідні рядки, інтерпретує їх як JSON-пакети та виконує необхідні дії залежно від типу повідомлення.

При отриманні службового пакета hello сервер реєструє клієнта в директорії, зберігає його публічний ключ підпису та визначає роль у сесії (лідер або послідовник) на основі порівняння ідентифікаторів. Одразу після цього клієнту повертається відповідь hello_ok з вказанням ролі та, за наявності, публічного ключа підпису обраного співрозмовника:

```
if t == "hello":
    st.id = obj["id"]
    st.peer_id = obj.get("peer")
    self.directory[st.id] = obj["sig_pub"]
    self.clients[st.id] = st
    self.status.setdefault(st.id, "idle")
    peer_pub = self.directory.get(st.peer_id)
    role = "leader" if st.peer_id and st.id < st.peer_id else "follower"
    await st.send({"type": "hello_ok", "role": role, "peer_sig_pub":
peer_pub})
```

У подальшій роботі relay-сервер прозоро пересилає між клієнтами службові пакети протоколу рукостискання (dh1, dh2, confirm) та зашифровані повідомлення типу msg, а також обслуговує запити лобі (list, invite, invite_reply, status). Ретельна організація передачі повідомлень дозволяє уникати колізій між одночасними підключеннями. Сервер не втручається у структуру переданих даних і не зберігає їх у пам’яті, що підвищує рівень конфіденційності. Такий підхід також дає можливість легко масштабувати систему – достатньо лише змінити IP або порт без модифікації внутрішньої логіки.

Модуль net_client.py реалізує роботу клієнта, який під’єднується до relay-сервера, визначає свою роль у сеансі (лідер чи послідовник), обмінюється службовими даними, виконує узгодження параметрів Діффі–Геллмана та зашифровує повідомлення перед відправленням. Кожен клієнт використовує спеціальні асинхронні функції, що дозволяє підтримувати одночасно кілька операцій без блокування основного циклу. При створенні клієнта зчитуються ідентифікатори MY_ID і PEER_ID, завантажується або генерується ключ підпису Ed25519, а також створюється об’єкт сесії, відповідальний за криптографічні операції.

Після створення об’єкта клієнта викликається функція connect, яка встановлює TCP-з’єднання з сервером, надсилає перше службове повідомлення типу hello з ідентифікаторами сторін та публічним ключем підпису і запускає фоновий цикл обробки вхідних даних:

```
async def connect(self):
```

```
    self.reader, self.writer = await asyncio.open_connection(self.host,
self.port)
    print(f"[{MY_ID}] connected to relay {self.host}:{self.port}")
    await self._send({
        "type": "hello",
        "id": MY_ID,
        "peer": PEER_ID,
        "sig_pub": ed_pub_to_b64(self.sig_pub),})
    asyncio.create_task(self._recv_loop())
```

У наведеному фрагменті програма одразу після підключення до relay-сервера передає службову інформацію, необхідну для подальшого рукостискання: власний ідентифікатор, бажаного співрозмовника та публічний ключ підпису. Паралельно запускається асинхронний цикл _recv_loop, який у режимі реального часу обробляє всі вхідні пакети протоколу:

```
async def _recv_loop(self):
    while True:
        line = await self.reader.readline()
        if not line:
            break
        obj = json.loads(line.decode("utf-8"))
        await self._handle(obj)
```

Особливістю оновленої реалізації є інтеграція механізмів взаємної автентифікації безпосередньо у клієнтський модуль. Після отримання від сервера відповіді hello_ok клієнт створює об’єкт SecureSession, передаючи йому власний приватний ключ підпису та публічний ключ підпису піра. Далі, залежно від ролі (лідер або послідовник), запускається виконання кроків протоколу: лідер формує та надсилає пакет dh1, послідовник перевіряє його підпис, у відповідь надсилає dh2, після чого лідер перевіряє підпис і надсилає підтвердження confirm. Фрагмент обробки цих станів наведено нижче:

```
if t == "hello_ok":
    self.role = obj["role"]
    peer_pub_b64 = obj.get("peer_sig_pub")
    if peer_pub_b64:
        self.peer_sig_pub = ed_pub_from_b64(peer_pub_b64)
```

```
    self.sess = SecureSession(MY_ID, PEER_ID, self.sig_priv,
self.peer_sig_pub)
    if self.role == "leader" and self.peer_sig_pub and not
self._dh1_sent:
        ga_b64, ra_b64, sig_b64 = self.sess.sign_dh1()
        await self._send({"type": "dh1", "to": PEER_ID, "ga": ga_b64,
"r": ra_b64, "sig": sig_b64})
     ...
```

Таким чином, клієнтська частина організовує не лише обмін службовими даними та узгодження параметрів, а й повноцінну взаємну автентифікацію сторін на основі цифрових підписів. Після завершення рукостискання сесія переходить у стан READY, після чого метод send_plain шифрує текст за допомогою матричного шифру та відправляє його у вигляді JSON-структури з полями n (nonce) і c (шифротекст).

Криптографічна частина реалізована у модулі secure_session.py, який забезпечує створення спільного ключа, його ущільнення, перетворення в кватерніон і побудову матриці обертання Γ(q̂). Основу становить клас SecureSession, який інкапсулює всі операції – від генерування параметрів до готових функцій зашифрування і розшифрування. На початку визначено допоміжні структури та функції для роботи з ключами підпису Ed25519, публічними ключами та кодуванням у форматі Base64:

```
def load_or_create_ed25519(path_priv: str):
    if os.path.exists(path_priv):
        with open(path_priv, "rb") as f:
            priv = serialization.load_pem_private_key(f.read(),
password=None)
        return priv, priv.public_key()
    priv = Ed25519PrivateKey.generate()
    ...
    return priv, priv.public_key()
```

Далі у конструкторі SecureSession створюються ефемерні ключі X25519 для поточного сеансу, а також випадкові значення r_my і r_peer, які використовуються як соль у процедурі отримання 1024-бітного ключового матеріалу з використанням HKDF. Для захисту від атак повторного відтворення

(replay) сесія запам’ятовує локально відправлені значення ga та r, щоб у наступних кроках перевірити їх узгодженість:

```
class SecureSession:
    def __init__(...):
        self.ec_priv = X25519PrivateKey.generate()
        self.ec_pub = self.ec_priv.public_key()
        self.r_my = secrets.token_bytes(16)
        ...
        self.ga_local_b64: Optional[str] = None
        self.r_local_b64: Optional[str] = None
```

Механізм взаємної автентифікації реалізовано через пари функцій sign_dh1 / verify_dh1 та sign_dh2 / verify_dh2. У sign_dh1 ініціатор формує публічний ключ ga, обчислює контекст підпису, який включає ідентифікатори обох сторін та випадкове значення r_my, і підписує його власним приватним ключем Ed25519:

```
def sign_dh1(self) -> Tuple[str, str, str]:
    ga = self.ec_pub.public_bytes(...)
    ctx = self._ctx_dh1(self.my_id.encode(), self.peer_id.encode(), ga,
self.r_my)
    sig = self.sig_priv.sign(ctx)
    ga_b64 = _b64e(ga)
    ra_b64 = _b64e(self.r_my)
    self.ga_local_b64 = ga_b64
    self.r_local_b64 = ra_b64
    return ga_b64, ra_b64, _b64e(sig)
```

На стороні отримувача функція verify_dh1 відновлює контекст, перевіряє підпис та зберігає публічний ключ X25519 і випадкове значення піра. Аналогічно, під час другого кроку протоколу функція sign_dh2 формує контекст, який містить як власні, так і віддзеркалені параметри співрозмовника, а verify_dh2 виконує подвійний контроль: перевіряє підпис і порівнює значення ga та ra з тими, що були відправлені у DH1, блокуючи атаки типу replay.

Фіналізація рукостискання реалізована у функції finalize. Вона обчислює спільний секрет за допомогою операції X25519PrivateKey.exchange, застосовує HKDF для отримання 1024-бітного ключового матеріалу, після чого виконується

ущільнення до 64 біт функцією compress_1024_to_64. На основі отриманого значення формується кватерніон, нормалізований за допомогою функції normalize_quaternion_from_k0, та будується ортогональна поворотна матриця $\mathbf{\Gamma}(\hat{q})$:

```
def finalize(self, initiator: bool):
    shared = self.ec_priv.exchange(self.peer_ec_pub)
    salt = (self.r_my + self.r_peer) if initiator else (self.r_peer +
self.r_my)
    k1024_bytes = HKDF(...).derive(shared)
    K0_64 = compress_1024_to_64(int.from_bytes(k1024_bytes, "big"))
    qhat = normalize_quaternion_from_k0(K0_64)
    Gamma = gamma_from_quaternion(qhat[0], qhat[1], qhat[2], qhat[3])
    self.keys = SessionKeys(k1024_bytes, K0_64, Gamma)
    self.ready = True
```

На основі цих параметрів реалізовано обгортки aead_encrypt та aead_decrypt, які використовують розроблений матричний шифр для шифрування та розшифрування текстових повідомлень. Таким чином, криптографічний ключ стає геометричним оператором, що використовується при зашифруванні даних у подальших етапах.

Для зручності користувача створено графічний інтерфейс у модулі gui.py. Його завдання – відображення всіх повідомлень, що проходять через систему, а також забезпечення введення нових даних. Робота інтерфейсу організована у вигляді окремого потоку, який виконує асинхронні мережеві операції без блокування основного вікна. Клас ClientLoopThread створює власний цикл asyncio, піднімає об’єкт NetClient та забезпечує thread-safe виклики до методів відправки:

```
class ClientLoopThread(threading.Thread):
    def run(self):
        self.loop = asyncio.new_event_loop()
        asyncio.set_event_loop(self.loop)
        self.client = NetClient(self.host, self.port,
on_plain_rx=self.on_plain_rx)
        self.loop.run_until_complete(self._main())
```

Передача текстових повідомлень у мережу виконується через функцію send_text, яка використовує механізм asyncio.run_coroutine_threadsafe для безпечного звернення до мережевого клієнта з графічного середовища:

```
def send_text(self, text: str):
    if not text or not self.loop or not self.client:
        return
    fut = asyncio.run_coroutine_threadsafe(self.client.send_plain(text),
self.loop)
    fut.result(timeout=10)
```

Окремий модуль lobby.py реалізує вікно лобі, у якому користувач може переглядати список онлайн-клієнтів, надсилати запити на встановлення чату та приймати або відхиляти інвайти. Клас RelayAPI інкапсулює мінімальний клієнт до relay-сервера для операцій list, invite, invite_reply та status, а у випадку підтвердження запиту запускає процес із графічним чат-клієнтом. Таким чином, рівень автентифікації та шифрування залишається повністю прозорим для користувача, який працює лише з простим інтерфейсом вибору співрозмовника та обміну повідомленнями.

Розроблений програмний засіб забезпечує повний цикл встановлення захищеного з’єднання між двома клієнтами. Relay-сервер виконує роль нейтрального посередника, гарантуючи передачу повідомлень без втручання у їхній зміст, але зберігаючи мінімальну службову інформацію про публічні ключі підпису. Клієнтська частина організовує послідовність обміну службовими даними, виконує взаємну автентифікацію та всі етапи узгодження параметрів.

Модуль захищеної сесії реалізує формування ключа за алгоритмом Діффі–Геллмана, його стиснення та перетворення в ортогональну поворотну матрицю $Г(\hat{q})$, що створює унікальний криптографічний простір для кожної сесії. Графічний інтерфейс (лобі та чат) забезпечує зручний спосіб візуалізації процесу шифрованого обміну, приховуючи від користувача всі деталі взаємної автентифікації та роботи криптографічних алгоритмів.

3.5 Тестування програмного засобу

3.5.1 Тестування коректності роботи застосунку

Метою тестування є перевірка повного циклу роботи системи: від запуску relay-сервера до встановлення зашифрованого каналу зв'язку між двома клієнтами, а також відображення переданих повідомлень у графічному інтерфейсі.

Першим кроком є запуск модуля relay_server.py, який створює асинхронний TCP-сервер на локальному хості 127.0.0.1:8765. Сервер очікує підключень клієнтів і забезпечує маршрутизацію повідомлень між ними. Приклад виведення повідомлення під час кожного нового підключення наведено на рис. 3.4.

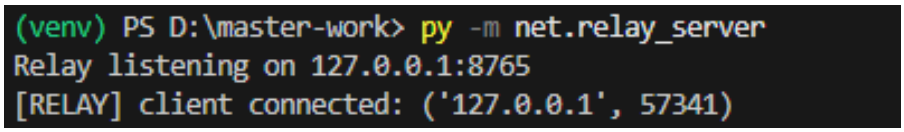


Рисунок 3.4 – Повідомлення про нове підключення

Якщо вже є інший активний клієнт, сервер негайно передає новому клієнту останнє отримане службове повідомлення hello, як показано на рис. 3.5., забезпечуючи синхронізацію етапу рукостискання.

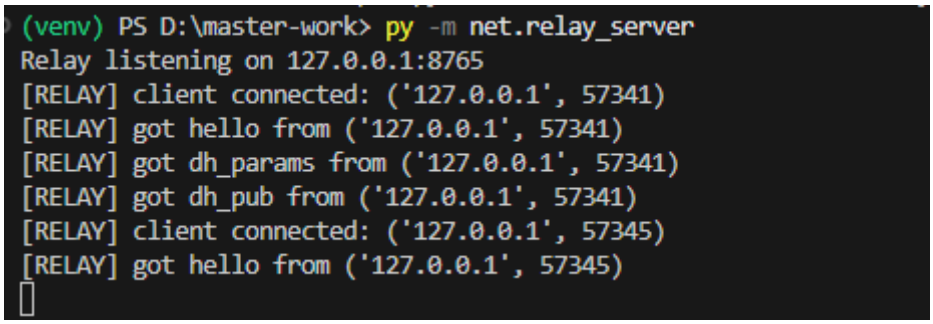


Рисунок 3.5 – Результати синхронізації етапу рукостискання

Далі запускаються два екземпляри gui.py, які створюють вікна чату, як показано на рис 3.6.

Кожен клієнт ініціалізує об'єкт ClientLoopThread, який у фоновому потоці запускає цикл подій asyncio і створює мережевий клієнт NetClient.

Після запуску на екрані з’являється інтерфейс з полем введення повідомлення, кнопкою Send та областю історії повідомлень.

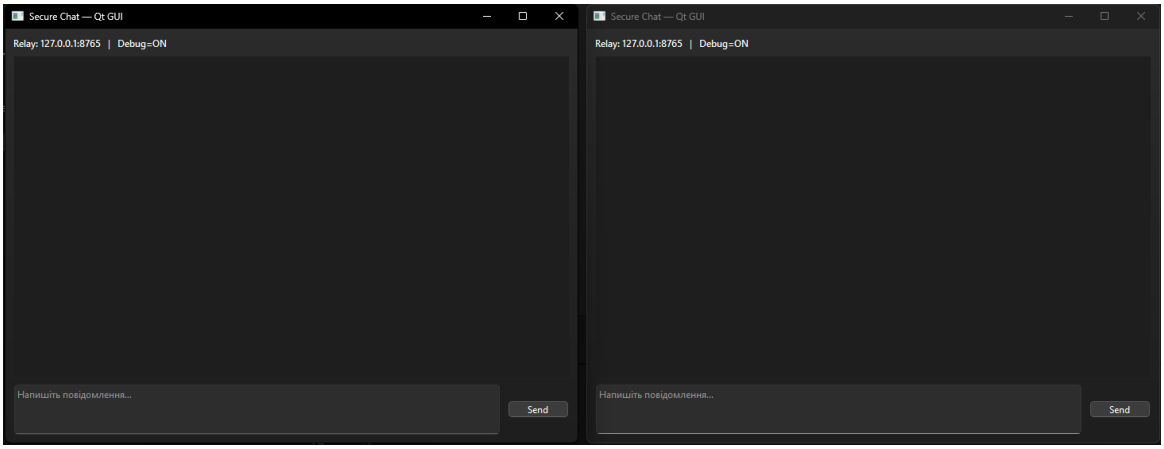
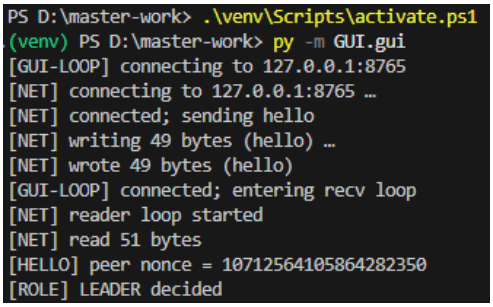
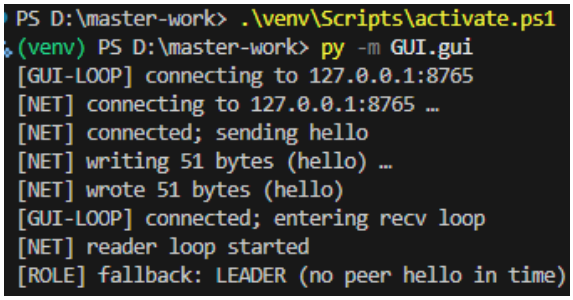


Рисунок 3.6 – Створені екземпляри GUI

Після встановлення TCP-з’єднання клієнт надсилає службове повідомлення типу “hello”, яке містить випадковий 64-бітний nonce – ідентифікатор сеансу. Relay-сервер приймає цей пакет, зберігає його у змінній last_hello_payload, і пересилає іншому клієнтові. Отримавши nonce від пари, кожен клієнт порівнює значення – клієнт із меншим nonce визначається як лідер, який надалі ініціює параметри Діффі-Геллмана, як зображено на рис. 3.7.



а)

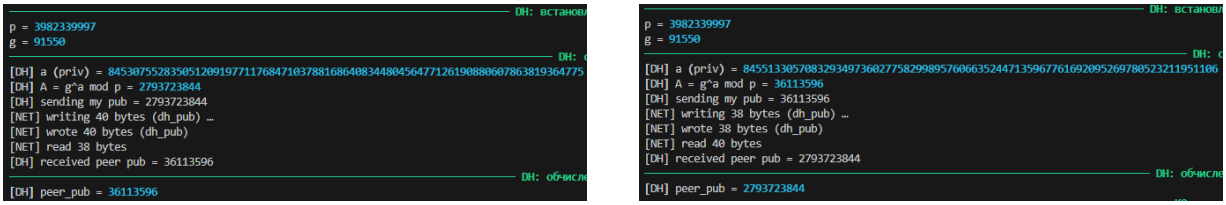


б)

Рисунок 3.7. – Результати завершення рукостикання: а – лідер, б – не лідер

Користувач A генерує пару параметрів (p, g) за допомогою функції local_init_params() із модуля secure_session.py. Ці значення передаються іншому клієнту через сервер у повідомленні типу “dh_params”. Після прийняття параметрів обидва клієнти створюють власні пари ключів, обмінюються відкритими ключами “dh_pub”, та переходять до формування спільного секрету.

Приклад обміну параметрами Діффі-Гелманна між клієнтами наведено на рис. 3.8.



а) б)

Рисунок 3.8. – Результат обміну параметрів Діффі-Геллмана: а – користувач А, б –користувач В

Після отримання відкритого ключа від пари кожен клієнт обчислює спільну таємницю.

Отримане число приводиться до 1024-бітного вигляду через функцію KDF, ущільнюється до 64-бітного ключа K_0 та використовується для формування нормованого кватерніона. Приклад даної операції наведено на рис. 3.9.

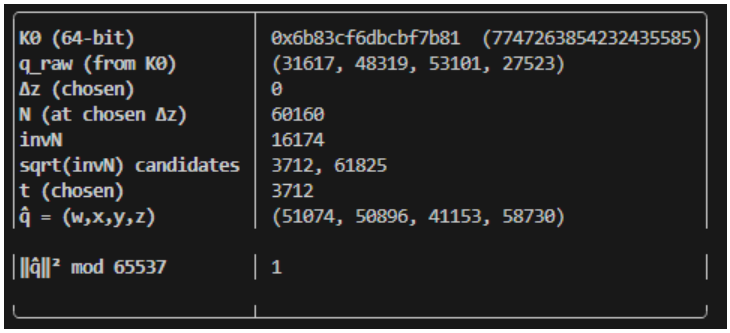


Рисунок 3.9 – Результат формування нормованого кватерніона на основі спільного секрету K_0

На його основі будується ортогональна поворотна матриця $\mathbf{\Gamma}(\widehat{q})$, яка слугує матрицею шифрування потоку. Після успішного створення матриці $\mathbf{\Gamma}(\widehat{q})$, що зображена на рис. 3.10, клієнти переходять у стан READY, що сигналізує готовність до безпечного обміну повідомленнями.

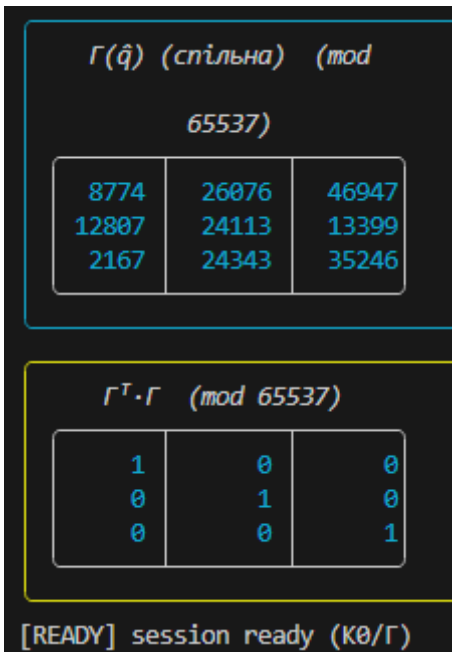


Рисунок 3.10 – Ортогональна поворотна матриця $\mathbf{\Gamma}(\widehat{q})$

Після встановлення сеансу користувач вводить текст у вікні чату. Повідомлення шифрується методом, що представлений у першій частині роботи і передається через relay-сервер іншому клієнтові у форматі JSON-пакета “msg”.

Отримувач розшифровує вміст функцією sess.decrypt_text() і миттєво відображає його у вікні чату з часовою міткою. Приклад обміну повідомленнями двох клієнтів наведено на рис. 3.11.

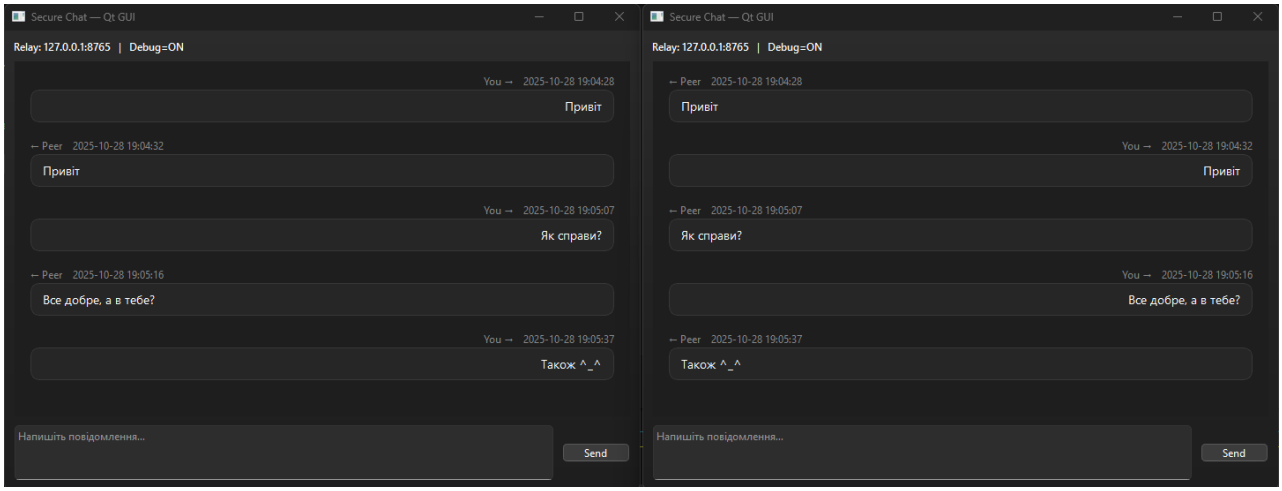


Рисунок 3.11 – Результат обміну повідомленнями між клієнтами

Після завершення обміну повідомленнями користувач може закрити вікно клієнтського застосунку. При цьому клієнт надсилає сигнал завершення,

припиняє роботу фонових потоків і закриває з'єднання із сервером. Це гарантує коректне завершення сеансу без зависань чи втрати даних при наступному запуску програми. Сервер, у свою чергу, фіксує відключення клієнта та переходить у режим очікування нових підключень.

У результаті тестування підтверджено правильність реалізації всіх етапів роботи програмного засобу – від запуску relay-сервера до встановлення зашифрованого каналу зв'язку між двома клієнтами. Передача повідомлень здійснювалася коректно, дані успішно зашифрувалися та розшифровувалися, а обмін параметрами Діффі–Геллмана відбувався без помилок. Програма показала стабільну роботу, що свідчить про правильність реалізації алгоритму та надійність створеного протоколу обміну інформацією.

3.5.2 Тестування стійкості протоколу обміну інформацією

Для перевірки стійкості протоколу обміну інформацією було виконано серію з семи атак MITM, у яких зловмисник перехоплював та модифікував службові повідомлення протоколу, або блокував їх доставку. Для кожної атаки аналізувалися журнали зловмисника та клієнтів, а ключовим індикатором стійкості була неможливість встановити захищену сесію та доставити повідомлення.

Перша атака flip_ga – спотворення g^a у dh1, полягала у тому, що атакер перехоплював перше DH-повідомлення та змінював у ньому параметр g^a, намагаючись порушити узгодження спільного секрету або спричинити помилку верифікації підпису. На рис. 3.12 показано момент підміни значення g^a перед передаванням клієнту B.

Рисунок 3.12 – Фрагмент журналу зловмисника під час атаки flip_ga

Після отримання модифікованого пакета клієнт виконав перевірку підпису та виявив його недійсність, що видно на рис. 3.13.

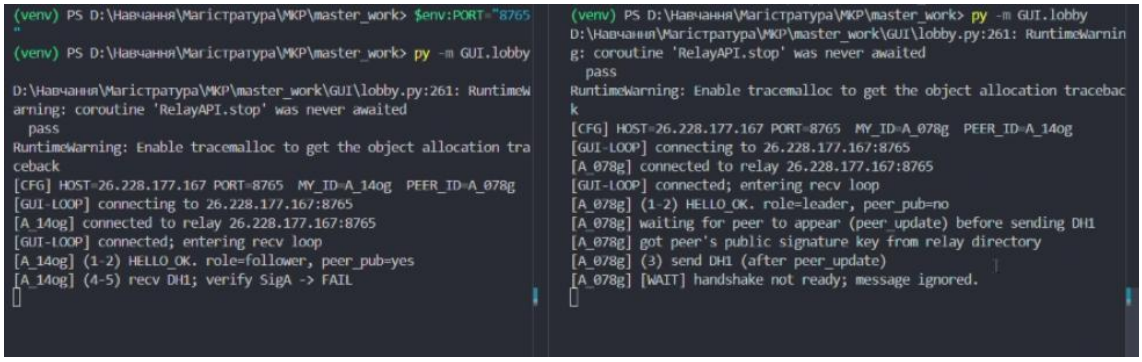


Рисунок 3.13 – Фрагмент консолі клієнтів під час атаки flip_ga

Рукостискання було зупинене, сесія не сформована, повідомлення не дійшло, отже протокол стійко блокує втручання у DH-параметри.

Друга атака flip_r – спотворення r у DH1 заключалася у тому, що зловмисник змінював nonce R_A, який також входить до контексту підпису. На рис. 3.14 наведено епізод підміни поля r.

Рисунок 3.14 – Фрагмент журналу зловмисника під час атаки flip_r

Оскільки цифровий підпис Ed25519 прив’язаний до всіх байтових значень DH1, клієнт B негайно виявив невідповідність підпису, що демонструє рис. 3.15.

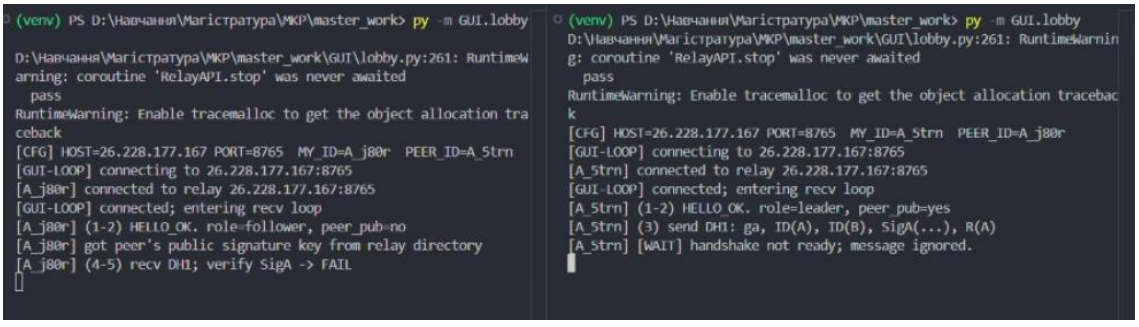


Рисунок 3.15 – Фрагмент консолі клієнтів під час атаки flip_r

Як наслідок, встановлення сеансу було скасовано, а повідомлення не дійшло, що підтверджує стійкість протоколу до спотворення nonce-значень.

Наступна атака flip_rb – спотворення rb у confirm спрямова на завершальний етап рукостискання: зловмисник змінює nonce R_B у повідомленні confirm, намагаючись порушити завершення узгодження ключів. На рис. 3.16 видно модифікацію поля rb.

Рисунок 3.16 – Фрагмент журналу зловмисника під час атаки flip_rb

Клієнт B, отримавши confirm, порівняв отримане значення з локально збереженим та виявив розбіжність, що відображено на рис. 3.17.

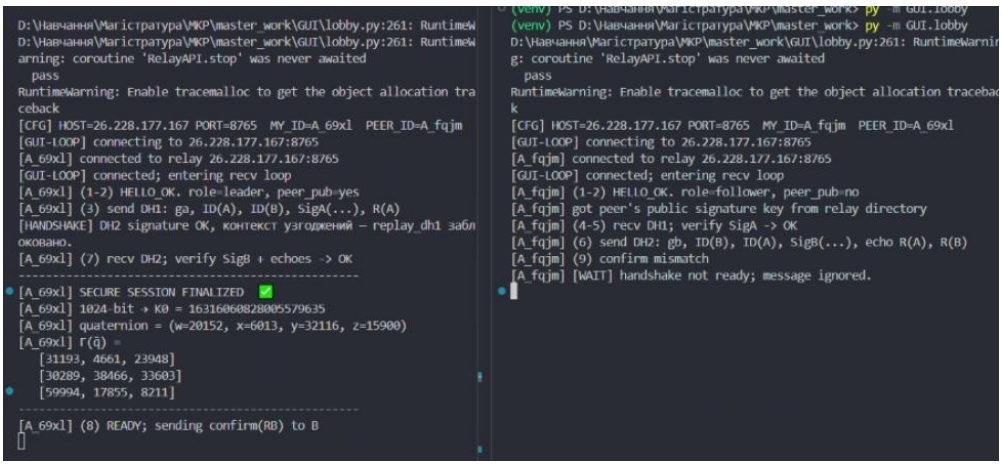


Рисунок 3.17 – Фрагмент консолі клієнтів під час атаки flip_rb

Через це сесію не було завершено, шифрування не активувалося, і повідомлення не дійшло, що свідчить про коректну перевірку фінальної узгодженості nonce.

Атака corrupt_sig – модифікація цифрового підпису полягає у тому, що зловмисник намагається змінити цифровий підпис Ed25519 у DH1 або DH2,

залишаючи інші поля незмінними, щоб перевірити, чи зможе протокол виявити пошкоджений підпис. На рис. 3.18 наведено момент підміни поля sig.

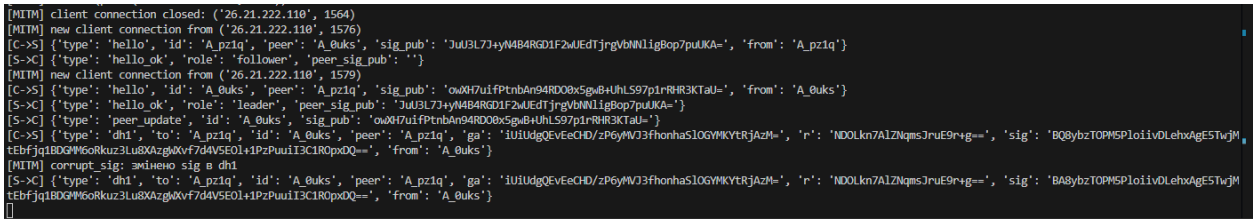


Рисунок 3.18 – Фрагмент журналу зловмисника під час атаки corrupt_sig

Після отримання пакета клієнт виконав перевірку й одразу відхилив повідомлення, що видно на рис. 3.19.

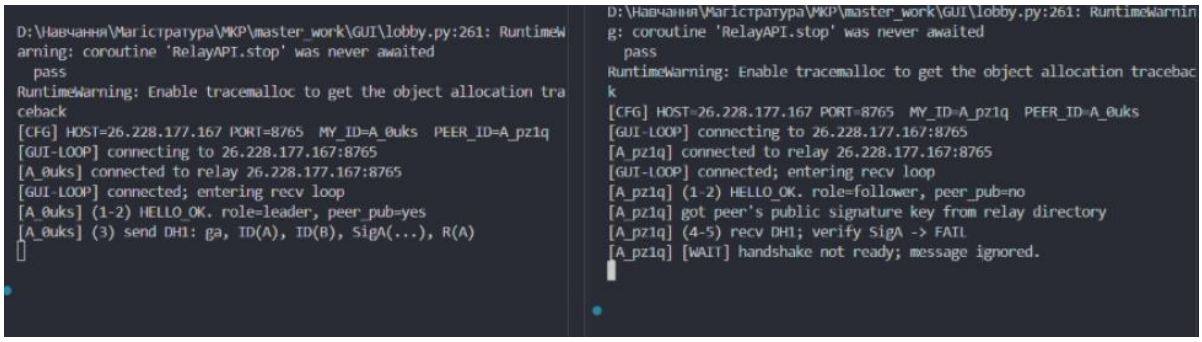


Рисунок 3.19 – Фрагмент консолі клієнтів під час атаки corrupt_sig

Оскільки підробити підпис без приватного ключа неможливо, протокол впевнено заблокував атаку, і повідомлення не дійшло.

Атака replay_dh1 – повторне використання попереднього DH1 передбачала повторну відправку раніше перехопленого DH1 у новій сесії.

Під час першого запуску атаки атакер здійснив перехоплення початкового DH1 та зареєстрував його у своїй консолі, що показано на рис. 3.20.

Рисунок 3.20 – Фрагмент журналу зловмисника під час першого з’єднання атаки replay_dh1

Оскільки протокол ще не мав підстав підозрювати повторне використання пакета, а значення DH-параметрів були коректними, клієнтська сторона сприйняла раунд рукостискання як звичайний. На рис. 3.21 видно, що у консолі клієнтів не виникло жодних помилок, а обмін повідомленнями відбувся успішно.

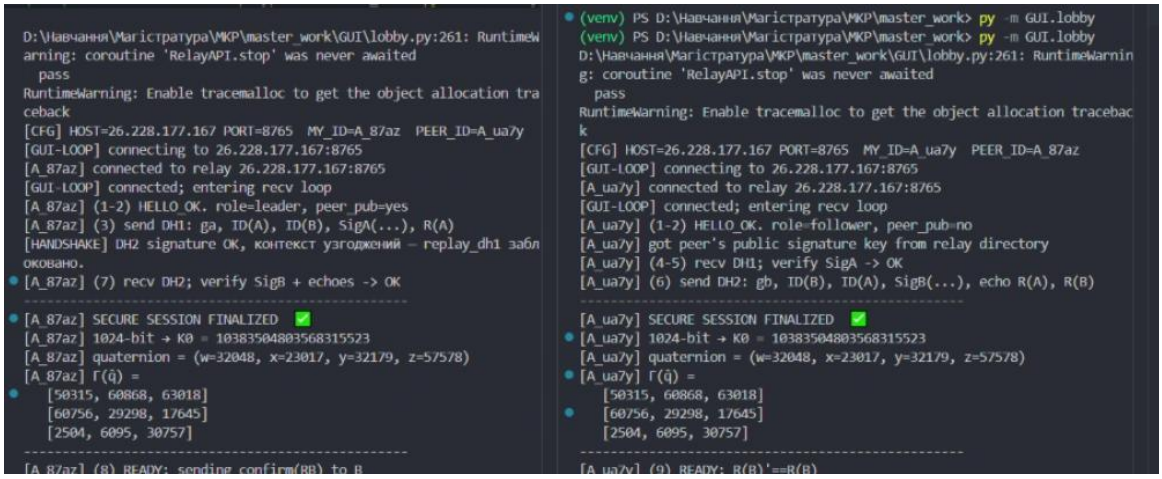


Рисунок 3.21 – Фрагмент консолі клієнтів під час першого з'єднання атаки replay_dh1

На цьому етапі протокол не міг виявити атаку, адже підміна ще не була виконана – зловмисник лише зберіг пакет DH1 для подальшого використання.

У другій сесії зловмисник повторно відправив збережене DH1-повідомлення, і це відображено на рис. 3.22.

Рисунок 3.22 – Фрагмент журналу зловмисника під час повторного з'єднання атаки replay_dh1

На відміну від першого запуску, цього разу клієнтська сторона вже мала локально збережені значення g^a та R_A, які були відправлені у попередній DH1. Коли протокол отримав DH2 із підробленим або невідповідним контекстом, відбулася перевірка узгодженості. На рис. 3.23 видно, що клієнт зафіксував

невідповідність параметрів та згенерував помилку, що чітко вказує на replay-атаку.

Рисунок 3.23 – Фрагмент консолі клієнтів під час повторного з’єднання атаки replay_dh1

Це призвело до негайного припинення рукостискання: сесія не була завершена, режим шифрування не активувався, а повідомлення не дійшло, що підтверджує ефективність механізму захисту протоколу від повторного використання пакетів.

Атака drop_confirm – блокування confirm, складається з того, що зловмисник не спотворює дані, а повністю перехоплює і не пересилає повідомлення confirm, необхідне для переходу сторін у стан READY. На рис. 3.24 видно відсутність пересланого пакета.

Рисунок 3.24 – Фрагмент журналу зловмисника під час атаки drop_confirm

На рис. 3.25 зафіксовано, що клієнти залишаються у режимі очікування завершення рукостискання, оскільки жодна зі сторін не отримує підтверджувального повідомлення confirm, необхідного для переходу у стан READY. Консоль демонструє, що протокол не може завершити фінальний етап

встановлення сесії, тому сторони продовжують очікувати відповідь, не переходячи до етапу обміну зашифрованими даними.

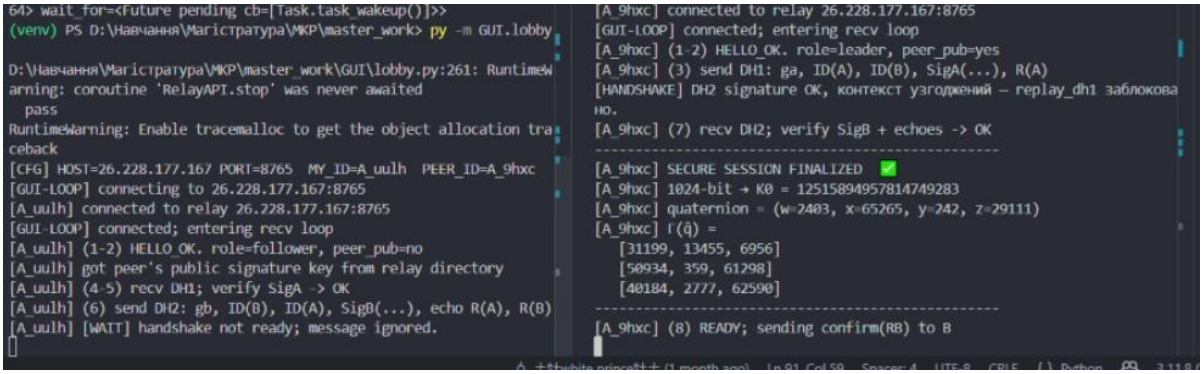


Рисунок 3.25 – Фрагмент консолі клієнтів під час атаки drop_confirm

Оскільки фінальний крок протоколу не було виконано, сесія не активувалася, і повідомлення не дійшло, що підтверджує правильну реакцію протоколу на втрату критичних службових повідомлень.

Остання Атака drop_msg – блокування зашифрованих повідомлень передбачала перехоплення та непередавання вже зашифрованих повідомлень msg після встановлення ключів. На рис. 3.26 показано, що шифротекст блокується атакером.

Рисунок 3.26 – Фрагмент журналу зловмисника під час атаки drop_confirm

На рис. 3.27 наведено фрагмент журналів клієнтів, де відображено, що отримувач не приймає жодного шифротексту. Клієнтська сторона коректно очікує надходження даних, однак через блокування пакета зловмисником передача фактично не відбувається.

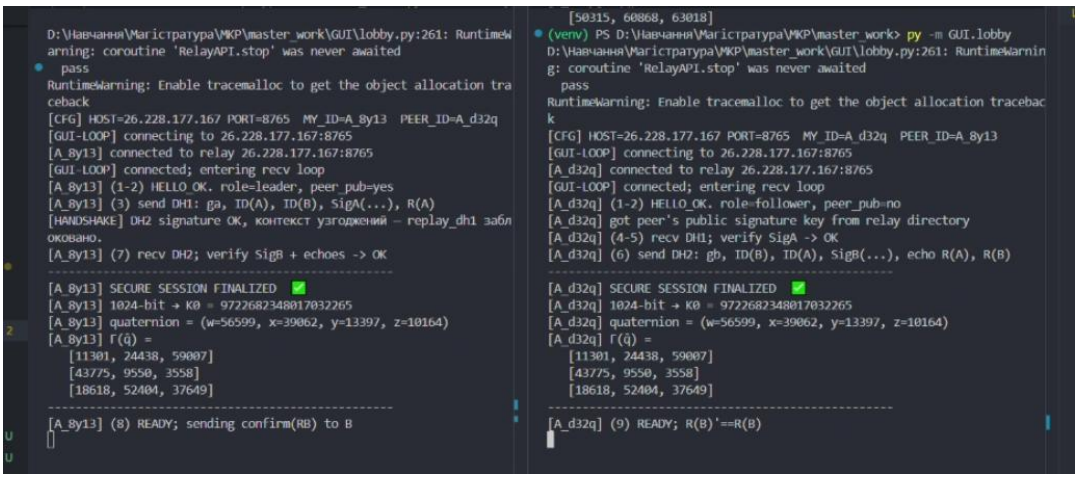


Рисунок 3.27 – Фрагмент консолі клієнтів під час атаки drop_confirm

Хоча зловмисник не може підробити або розшифрувати дані, він може лише перервати доставку. Як наслідок, повідомлення не дійшло, протокол залишився криптографічно захищеним, і жодної компрометації ключів не відбулося.

3.6 Висновки до розділу

У результаті було розроблено та програмно реалізовано повноцінний програмний засіб, що втілює запропонований протокол обміну інформацією та забезпечує формування захищеного каналу передачі даних між двома клієнтами. Використання мови Python та середовища Visual Studio Code дало змогу організувати швидку, модульну й керовану розробку, що є важливим для криптографічних систем підвищеної складності та подальшої підтримки коду.

Архітектура системи побудована за модульним принципом і включає GUI-модуль, мережевий клієнт, криптографічний модуль та relay-сервер, а також прикладний модуль lobby. Такий поділ функцій забезпечив чітке розмежування відповідальності: relay-сервер виступає нейтральним ретранслятором без доступу до закритих ключів і відкритого тексту, мережевий клієнт відповідає за протокольну логіку, криптографічний модуль – за формування спільного секрету, ущільнення ключа та матричне шифрування, а графічний інтерфейс надає користувачу зручний доступ до функцій системи.

Алгоритм протоколу обміну інформацією реалізовано з урахуванням взаємної автентифікації сторін і узгодження спільного сеансового ключа. У програмній реалізації використано поєднання довготривалих ключів підпису Ed25519, ефемерного узгодження ключів (X25519/Діффі-Геллман) та подальшої обробки спільного секрету з отриманням 1024-бітного ключового матеріалу, який ущільнюється до 64-бітного ключа K_0 і використовується для формування нормованого кватерніона та ортогональної поворотної матриці $\mathbf{\Gamma}(\hat{q})$. Це забезпечує узгодженість із теоретичною частиною роботи та створює криптографічно стійкий простір сесії.

Функціональне тестування підтвердило коректність роботи всіх модулів: встановлення з'єднання, обмін службовими повідомленнями, формування спільного секрету, побудову матриці обертання та виконання операцій зашифрування й розшифрування. Передача повідомлень між клієнтами відбувалася без помилок, а relay-сервер продемонстрував коректну роботу в режимі прозорого ретранслятора без зберігання конфіденційних даних.

Окремо було проведено тестування стійкості протоколу до атак типу MITM, у межах якого змодельовано різні сценарії спотворення, повторного використання та блокування службових і зашифрованих повідомлень. У всіх випадках спроби модифікації параметрів рукостискання або підписів призводили до виявлення помилок і зриву встановлення сесії, а у випадку блокування пакетів протокол коректно не переходив у стан готовності до обміну даними. Це підтверджує криптографічну стійкість реалізованого протоколу.

Таким чином, розроблений програмний засіб повністю реалізує запропонований протокол обміну інформацією, поєднуючи практичну реалізацію мережевої взаємодії, криптографічні механізми автентифікації та узгодження ключів, а також матричне шифрування на основі кватерніонної поворотної матриці. Отримані результати свідчать про життєздатність та ефективність підходу і створюють основу для подальшого розширення функціональності та інтеграції системи у реальні сценарії захищеного обміну даними.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Оскільки результати МКР «Програмний засіб захищеного передавання інформації. Частина 2. Протокол обміну інформацією» мають потенціал комерціалізації та можуть бути інтегровані в програмні продукти підприємств і сторонніх розробників, необхідно провести комерційний аудит розробки.

Його мета – оцінити науково-технічну цінність програмного засобу, перспективи його виходу на ринок та підготувати економічне обґрунтування для потенційного інвестора [29].

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

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

Середньоарифметична сума балів розраховується за формулою:

$$\text{СБ} = \frac{\sum_{i=1}^{3} \text{СБ}_i}{3} \tag{4.1}$$

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

Критерії	Бали		
	Бабачук О. Д.	Атаманюк О. В.	Малахов В. 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	3	4

Продовження табл. 4.1

Критерії	Бали		
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	41	42
Середньоарифметична сума балів $СБ_c$	41		

Отже, результати розрахунків свідчать, що науково-технічний рівень і комерційний потенціал розробленого протоколу обміну інформацією є високими (табл. 4.2). Це зумовлено вдосконаленими механізмами захищеної взаємодії, інтеграцією взаємної автентифікації та оптимізованим формуванням сеансових ключів і контролю цілісності [29].

Такі рішення підвищують стійкість до атак типу «людина посередині», зменшують затримки під час встановлення сеансу та підвищують загальну ефективність порівняно з традиційними методами. Впровадження протоколу може вимагати певної адаптації під конкретні системи та додаткових тестувань, проте ці витрати є помірними та легко оптимізуються завдяки модульній структурі та програмній реалізації.

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

У межах даної магістерської роботи необхідно провести аналіз конкурентоспроможності розробленого протоколу обміну інформацією. Його рівень визначається шляхом порівняння технічних характеристик, криптографічних властивостей та економічних показників упровадження з існуючими рішеннями у сфері захищеної комунікації.

Оцінювання охоплює аналіз стійкості автентифікації сторін, ефективності встановлення сеансових ключів, продуктивності обміну та вартості інтеграції у практичні системи. Як базовий аналог для порівняння обрано Signal Protocol – один із найпоширеніших та найнадійніших сучасних протоколів захищеної передачі даних [29].

4.1.1 Розрахунок одиничних параметричних індексів

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

$$q_i = \frac{P_i}{P_{\text{баз}i}}, \tag{4.2}$$

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

P_i – значення i-го параметра розробки;

$P_{\text{баз}i}$ – 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 [29].

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	3	0,6	0,05
Функціональність	5-бальна шкала	5	5	1,0	0,05
Безпека	5-бальна шкала	5	4	0,8	0,25

Продовження табл. 4.3

Параметр	Одиниця виміру	Аналог	Нова розробка	Індекс зміни значення параметра	Коефіцієнт вагомості
Масштабованість	5-бальна шкала	4	5	1,25	0,4
Економічні					
Вартість інтеграції	грн	10 000	7 000	0,7	0,3
Вартість підтримки	грн	5 000	4 000	0,8	0,3
Час впровадження	години	14	10	0,71	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:

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

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

q_i – одиничний параметричний показник i-го параметра;

α_i – вагомість i-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:

$$I_{\text{ТП}} = 1{,}25 \cdot 0{,}25 + 0{,}6 \cdot 0{,}05 + 1 \cdot 0{,}05 + 0{,}8 \cdot 0{,}25 + 1{,}25 \cdot 0{,}4 = 1{,}093$$

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

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

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:

$$I_{\text{ЕП}} = 0{,}7 \cdot 0{,}3 + 0{,}8 \cdot 0{,}3 + 0{,}71 \cdot 0{,}4 = 0{,}734$$

4.1.3 Розрахунок інтегрального показника

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

$$K_{\text{ІНТ}} = I_{\text{НП}} \cdot \frac{I_{\text{ТП}}}{I_{\text{ЕП}}},$$
$$K_{\text{ІНТ}} = 1 \cdot \frac{1{,}093}{0{,}734} = 1{,}49. \quad (4.5)$$

На основі інтегрального показника $K_{\text{ІНТ}} = 1{,}49$ можна зробити висновок, що розроблений протокол є конкурентоспроможним і може бути виведений на ринок. Такий результат отримано завдяки вищим технічним показникам розробки та суттєво нижчим економічним витратам порівняно з аналогом [29].

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 [29].

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.

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
Аналітик з інформаційної безпеки	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 [29].

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

$t_{ЗМ}$ – тривалість зміни, год.

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

$$З_{р} = 90 \cdot 14 = 1260 \text{ грн.}$$

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 протоколів захищеного обміну інформацією	14	2	1,1	90,00	1 260,00
Проєктування протоколу обміну інформацією	25	7	2,2	180,00	4 500,00
Розробка архітектури програмної реалізації протоколу	4	3	1,35	110,45	441,80
Побудова алгоритмів виконання процедур протоколу	8	3	1,35	110,45	883,63
Програмна реалізація клієнтської та серверної частин протоколу	50	6	2,0	163,63	8 181,50
Оптимізація обміну повідомленнями в протоколі	39	5	1,7	139,09	5 424,51
Тестування протоколу обміну інформацією	40	3	1,35	101,25	4 045,00
Тестування стійкості протоколу до атак	17	6	2,0	163,63	2 781,71
Виправлення виявлених помилок у реалізації протоколу	18	3	1,35	110,45	1 998,10
Збірка застосунку та підготовка демонстраційного стенду протоколу	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) [29].

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 [29].

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.

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

Продовження табл. 4.7

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
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 [29].

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 Dell	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 (Ве) 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 [29].

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 [29].

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

$$I_в = (З_о + З_р) \cdot \frac{H_{ів}}{100\%}, \quad (4.15)$$

$$I_{\text{в}} = (29772{,}68 + 29598{,}06) \cdot \frac{50\%}{100\%} = 29685{,}37.$$

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:

$$\text{В}_{\text{нзв}} = (\text{З}_{\text{о}} + \text{З}_{\text{р}}) \cdot \frac{H_{\text{зв}}}{100\%}, \tag{4.16}$$

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

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

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

$$\text{В}_{\text{заг}} = \text{З}_{\text{о}} + \text{З}_{\text{р}} + \text{З}_{\text{дод}} + \text{З}_{\text{н}} + \text{М} + \text{К}_{\text{в}} + \text{В}_{\text{спец}} + \text{В}_{\text{прг}} + \text{А}_{\text{обл}} + \text{В}_{\text{е}} + \text{В}_{\text{св}} + \text{В}_{\text{сп}} + \text{І}_{\text{в}} + \text{В}_{\text{нзв}}, \tag{4.17}$$

$$\text{В}_{\text{заг}} = 29772{,}68 + 29598{,}06 + 7124{,}48 + 14628{,}88 + 334{,}4 + 3906{,}65 + 2249{,}99 + 47{,}06 + 29685{,}37 + 59370{,}74 = 176718{,}31.$$

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

$$\text{ЗВ} = \frac{\text{В}_{\text{заг}}}{\eta}, \tag{4.18}$$

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 [29].

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

У даній МКР розроблено спеціалізоване програмне забезпечення для захищеного передавання інформації, яке може бути впроваджене у виробничу та інформаційну інфраструктуру підприємства [29]. Потенційне зростання чистого прибутку підприємства (ΔПі) у роки, коли очікується позитивний ефект від впровадження та комерціалізації розробки, визначається за відповідною формулою:

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

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. Розрахунок відбувається за формулою [29]:

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

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.

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

Далі потрібно розрахувати величину початкових інвестицій PV, які замовник має вкласти для здійснення науково-технічної розробки. Для цього можна використати формулу [29]:

$$PV = k_{\text{розр}} \cdot \text{ЗВ}, \quad (4.21)$$

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:

$$E_{\text{абс}} = \text{ПП} - PV, \quad (4.22)$$

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 [29]. Внутрішня економічна дохідність інвестицій $E_{\text{в}}$, які можуть бути вкладені замовником у впровадження науково-технічної розробки, розраховується за формулою:

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

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

$$T_{\text{ок}} = \frac{1}{E_{\text{в}}}, \quad (4.24)$$

де $E_{\text{в}}$ – внутрішня економічна дохідність вкладених інвестицій.

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

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

4.4 Висновки до розділу

У межах економічної частини було проведено комплексний аналіз науково-технічного рівня, комерційного потенціалу та економічної доцільності впровадження розробленого протоколу обміну інформацією.

Експертне оцінювання, здійснене фахівцями ТОВ «ДАТАДЕФЕНДР» за визначеними критеріями, дало середній бал 41, що вказує на високий рівень технологічної зрілості розробки та значні перспективи її подальшої комерціалізації.

Порівняльний аналіз конкурентоспроможності, виконаний відносно Signal Protocol, засвідчив переваги запропонованого рішення у частині технічних параметрів, оптимізації процесів встановлення сеансових ключів, підвищеної стійкості до атак типу «людина посередині» та зменшення затримок при встановленні захищених сесій. Інтегральний показник конкурентоспроможності ($K_{\text{ІНТ}} = 1{,}49$) підтвердив, що протокол є ринково перспективним і може бути ефективно інтегрований у сучасні системи захищеної комунікації.

Детальний розрахунок витрат на виконання науково-дослідної роботи дозволив визначити загальну собівартість розробки на рівні 252,45 тис. грн. Оцінка очікуваного економічного ефекту впровадження протоколу показала суттєву фінансову вигоду: приведена вартість зростання чистого прибутку підприємства становить 902,83 тис. грн, а абсолютний економічний ефект ($E_{\text{абс}}$) – 397,92 тис. грн.

Важливим показником є внутрішня економічна дохідність інвестицій ($E_{\text{в}} = 0{,}78$), що перевищує середні значення для ІТ-проєктів, а короткий період окупності ($T_{\text{ок}} = 1{,}28$) підтверджує високу інвестиційну привабливість розробки.

Таким чином, запропонований протокол обміну інформацією є не лише технічно та науково обґрунтованим, але й економічно ефективним і доцільним для практичного впровадження, забезпечуючи підвищення рівня інформаційної безпеки та швидку фінансову віддачу.

ВИСНОВКИ

У процесі виконання комплексної магістерської кваліфікаційної роботи було досягнуто поставленої мети шляхом розроблення суміщеного протоколу автентифікації користувачів та узгодження ключів Діффі-Геллмана, а також створення програмного модуля, що забезпечує захищений обмін інформацією між клієнтами.

Проведені дослідження підтвердили ефективність запропонованого підходу та його відповідність сучасним вимогам до швидкодії, надійності та стійкості протоколів безпечного зв’язку.

На основі аналізу сучасних протоколів обміну інформацією було встановлено, що традиційні механізми встановлення захищеного каналу виконують автентифікацію й узгодження ключів окремо. Це збільшує кількість мережевих повідомлень, підвищує затримки та створює додаткові можливості для атак, зокрема MITM.

Дослідження існуючих рішень, таких як Діффі-Геллман, X3DH та Double Ratchet, показало, що вони мають високу криптографічну стійкість, проте часто вимагають значних обчислювальних ресурсів або використання розвиненої інфраструктури попередньо згенерованих ключів.

Це довело необхідність створення більш компактного, швидкодіючого та універсального протоколу, здатного виконувати необхідні криптографічні процедури в єдиному інтегрованому циклі.

У роботі було запропоновано протокол, який об’єднує автентифікацію користувачів та узгодження ключів Діффі–Геллмана в один процес. Такий підхід дозволив скоротити кількість інформаційних кроків, зменшити затримку встановлення захищеного з’єднання та підвищити стійкість до атак перехоплення.

Верифікація критичних параметрів за допомогою цифрових підписів Ed25519 забезпечила надійну взаємну автентифікацію без участі сторонніх

служб. Використання nonce-значень гарантувало захист від повторного відтворення повідомлень.

Після встановлення спільного секрету було виконано його ущільнення за допомогою кватерніонного перетворення, що дозволило перетворити 1024-бітне значення, отримане з Діффі-Геллмана, у придатний для подальшої обробки 64-бітний ключ із збереженням високої ентропії.

На його основі формується ортогональна поворотна матриця, що застосовується в методі автентифікованого шифрування, розробленому у першій частині комплексної магістерської кваліфікаційної роботи. Завдяки поєднанню матричного перетворення першого блоку з нелінійним генератором гами для наступних блоків забезпечується подвійний рівень дифузії та цілісності, що робить інтегровану систему придатною для використання в реальних криптографічних протоколах, месенджерах та IoT-пристроях.

Розроблений протокол було реалізовано у вигляді програмного модуля, який охоплює всі етапи встановлення захищеного зв’язку: генерування ключових параметрів, взаємну автентифікацію, узгодження секрету, ущільнення ключа, формування матриці $\mathbf{\Gamma}(\hat{q})$ та обмін зашифрованими повідомленнями.

Проведене тестування підтвердило коректність реалізації, стійкість до базових типів атак та узгодженість роботи всіх компонентів. Усі сценарії взаємодії показали стабільний, передбачуваний і безпечний результат.

Отже, в результаті виконання комплексної магістерської кваліфікаційної роботи було створено ефективний протокол встановлення автентифікованого зв’язку, розроблено механізм ущільнення секретного ключа та інтегровано його з методом автентифікованого шифрування.

Робота підтвердила доцільність запропонованого підходу та його подальшу перспективність у використанні в системах захищеного передавання інформації.

Розроблене рішення може бути розширене у напрямках оптимізації продуктивності, удосконалення захисту від активних атак та інтеграції у масштабовані багатокористувацькі архітектури.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Diffie W., Hellman M. *New directions in cryptography*. IEEE Trans. Inf. Theory. 1976; 22(6): 644–654.

2. ElGamal T. A. *A public key cryptosystem and a signature scheme based on discrete logarithms*. IEEE Trans. Inf. Theory. 1985; 31(4): 469-472.

3. Bernstein DJ, Lange T, editors. *Post-Quantum Cryptography*. Springer; 2009.

4. 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.

5. 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.

6. 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.

7. MTProto Mobile Protocol. URL: https://core.telegram.org/mtproto (accessed: 20.09.2025).

8. Everything You Need to Know about WhatsApp End-to-End Encryption. URL:https://www.gupshup.ai/resources/blog/whatsapp-end-to-end-encryption /#:~: text=WhatsApp%20employs%20the%20Signal%20Protocol,to%20ensure%20confidentiality%20and%20authenticity (accessed: 20.09.2025).

9. The X3DH Key Agreement Protocol. URL: https://signal.org/docs/specifications/x3dh/ (accessed: 22.09.2025).

10. Viber Encryption Overview. URL: https://www.viber.com/app/uploads/viber-encryption-overview.pdf (accessed: 22.09.2025).

11. Messenger End-to-End Encryption Overview. URL: https://engineering.fb.com/wp-content/uploads/2023/12/MessengerEnd-to-EndEncryptionOverview_12-6-2023.pdf (accessed: 25.09.2025).

12. Key Distribution in Cryptography. URL: https://awjunaid.com/cryptography/key-distribution-in-cryptography/ (accessed: 28.09.2025).

13. Diffie-Hellman Key Exchange Algorithm. URL: https://www.1kosmos.com/security-glossary/diffie-hellman-key-exchange-algorithm/ (accessed: 28.09.2025).

14. Diffie-Hellman Key Exchange (DHKE) Algorithm. URL: https://dev.to/techlabma/diffie-hellman-key-exchange-dhke-algorithm-505b (accessed: 02.10.2025).

15. Differences between TCP and UDP. – URL: https://www.geeksforgeeks.org/digital-logic/linear-feedback-shift-registers-lfsr/ (accessed: 07.10.2025).

16. What is mutual authentication? | Two-way authentication – URL: https://www.cloudflare.com/ru-ru/learning/access-management/what-is-mutual-authentication/ (accessed: 12.10.2025).

17. Conrad K. Quaternion Algebras. URL: https://kconrad.math.uconn.edu/blurbs/ringtheory/quaternionalg.pdf (accessed: 19.10.2025).

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

19. What Is A Message Authentication Code? URL: https://www.fortinet.com/uk/resources/cyberglossary/message-authentication-code (accessed: 26.10.2025).

20. ISO/IEC 18031:2025. Information technology – Security techniques – Random bit generation.

21. Viro O. Lecture 5. Quaternions. URL: https://www.math.stonybrook.edu/~oleg/courses/mat150-spr16/lecture-5.pdf (accessed: 03.11.2025).

22. Кватерніони. URL: https://stud.com.ua/156188/informatika/kvater nioni (accessed: 10.11.2025).

23. Rotation Matrices and Orthogonal Matrices. URL: https://math.libretexts.org/Bookshelves/Differential_Equations/Applied_Linear_Algebra_and_Differential_Equations_(Chasnov)/02%3A_II._Linear_Algebra/01%3A_Matrices/1.04%3A_Rotation_Matrices_and_Orthogonal_Matrices (accessed: 12.11.2025).

24. Тест Рабіна-Міллера і сильні псевдопрості числа. URL: https://studfile.net/preview/5367441/page:7/ (accessed: 15.11.2025).

25. Why Python Is the Best Programming Language? URL: https://stepmediasoftware.com/blog/why-python-is-the-best-programming-language/ (accessed: 24.11.2025).

26. VS Code vs. Pycharm: The Best IDE for Python. URL: https://geekflare.com/dev/vs-code-vs-pycharm/ (accessed: 26.11.2025).

27. PyQt6 – Comprehensive Python Bindings for Qt v6. URL: https://pypi.org/project/PyQt6/ (accessed: 27.11.2025).

28. asyncio (obsolete backport). URL: https://pypi.org/project/asyncio/ (accessed: 29.11.2025).

29. 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

Додаток А. ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Програмний засіб захищеного передавання інформації. Частина 2. Протокол обміну інформацією.

Автор роботи: Козира Володимир Андрійович
Тип роботи: магістерська кваліфікаційна робота
Підрозділ кафедра захисту інформації ФІТКІ , група 1 БС-24м

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism **3, 56%**

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., доцент _____ Олеся ВОЙТОВИЧ

Особа, відповідальна за перевірку _____ Валентина КАПЛУН

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Володимир ЛУЖЕЦЬКИЙ

Здобувач _____ Володимир КОЗИРА

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 2. ПРОТОКОЛ ОБМІНУ ІНФОРМАЦІЄЮ

Виконав: студент 2 курсу групи 1БС-24м
спеціальності 125 Кібербезпека
_____Володимир КОЗИРА

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

СХЕМА ОТРИМАННЯ ВІДКРИТОГО КЛЮЧА КОРИСТУВАЧАМИ

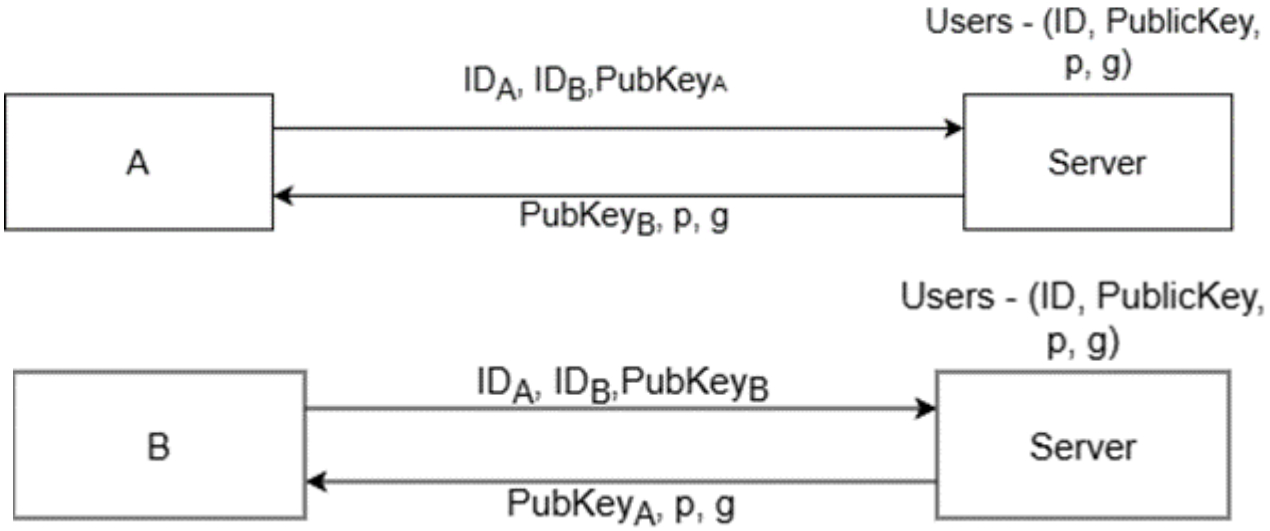
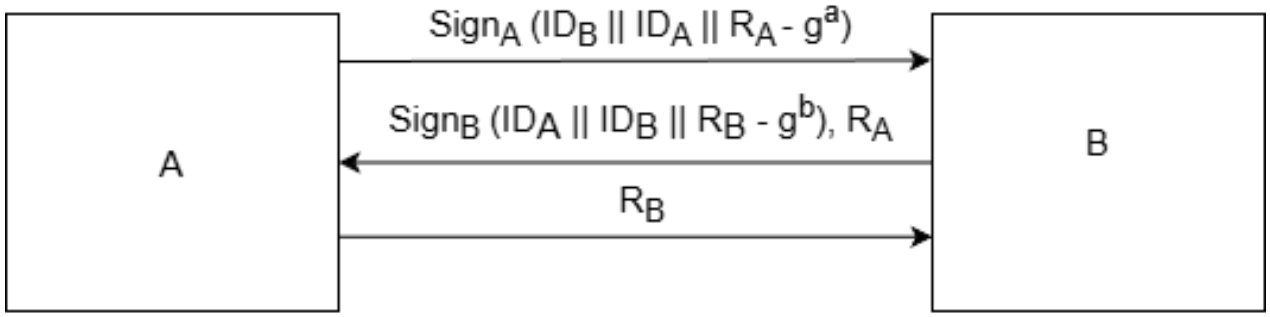


СХЕМА ВЗАЄМНОЇ АВТЕНТИФІКАЦІЇ СТОРІН A ТА B



ПРОЦЕДУРА ФОРМУВАННЯ СЕКРЕТНОГО КЛЮЧА

$$K = (K_1, K_2, K_3, K_4), \;\; K_i \in \; [0, 2^{256} - 1]$$

$$K_1 = (a_1, b_1, c_1, d_1), K_2 = (a_2, b_2, c_2, d_2), K_3 = (a_3, b_3, c_3, d_3), K_4 = (a_4, b_4, c_4, d_4)$$

$$K_1 = a_1 + b_1 i + c_1 j + d_1 k,$$
$$K_2 = a_2 + b_2 i + c_2 j + d_2 k,$$
$$K_3 = a_3 + b_3 i + c_3 j + d_3 k,$$
$$K_4 = a_4 + b_4 i + c_4 j + d_4 k$$

$$D_1 = K_1 K_2 = \begin{cases} a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2 = w_1, \\ a_1 b_2 + b_1 a_2 + c_1 d_2 - d_1 c_2 = z_1, \\ a_1 c_2 - b_1 d_2 + c_1 a_2 + d_1 b_2 = x_1, \\ a_1 d_2 + b_1 c_2 - c_1 b_2 + d_1 a_2 = k_1. \end{cases}$$

$$D_2 = K_3 K_4 = \begin{cases} a_3 a_4 - b_3 b_4 - c_3 c_4 - d_3 d_4 = w_2, \\ a_3 b_4 + b_3 a_4 + c_3 d_4 - d_3 c_4 = z_2, \\ a_3 c_4 - b_3 d_4 + c_3 a_4 + d_3 b_4 = x_2, \\ a_3 d_4 + b_3 c_4 - c_3 b_4 + d_3 a_4 = k_2. \end{cases}$$

$$Q = D_1 D_2 = \begin{cases} w_1 w_2 - z_1 z_2 - x_1 x_2 - k_1 k_2 = q_1 \\ w_1 z_2 + z_1 w_2 + x_1 k_2 - k_1 x_2 = q_2 \\ w_1 x_2 - z_1 k_2 + x_1 w_2 + k_1 z_2 = q_3 \\ w_1 k_2 + z_1 x_2 - x_1 z_2 + k_1 w_2. = q_4 \end{cases}$$

УЩІЛЬНЕННЯ СЕКРЕТНОГО КЛЮЧА

$$K_0 = (q_1 + q_2 + q_3 + q_4)\ mod\ 2^{64}$$

Процедура ущільнення має такі властивості:

– K_0 є геш-значенням для коду K.

– Нелінійність – кватерніонне множення створює складну залежність між вхідними частинами, що унеможливлює лінійне відновлення початкових компонент.

– Незворотність – через геш-значення.

– Ентропійність – усі 1024 біти початкового ключа впливають на кінцеве значення K_0, забезпечуючи повне змішування K_0.

ОБЧИСЛЕННЯ ПОВОРОТНОЇ МАТРИЦІ

$$K_0 = (w, x, y, z)$$

$$N = (w^2 + x^2 + y^2 + z^2)\ mod\ p$$

$$t^2 \equiv N^{-1}\ (mod\ p)$$

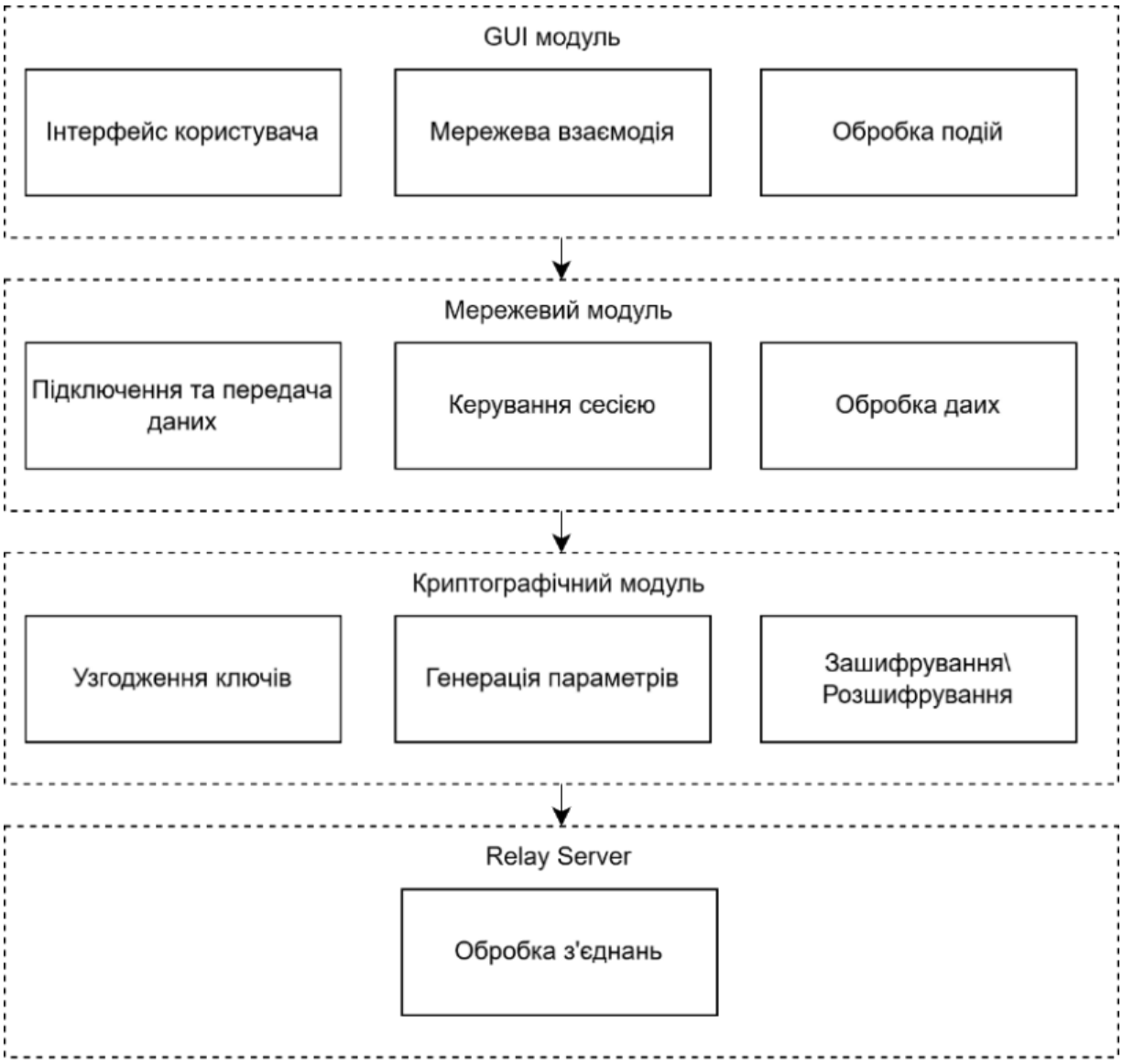
$$\left|\left|\hat{q}\right|\right|^2 \equiv 1\ (mod\ p)$$

$$\hat{q} = t \cdot q\ mod\ p$$

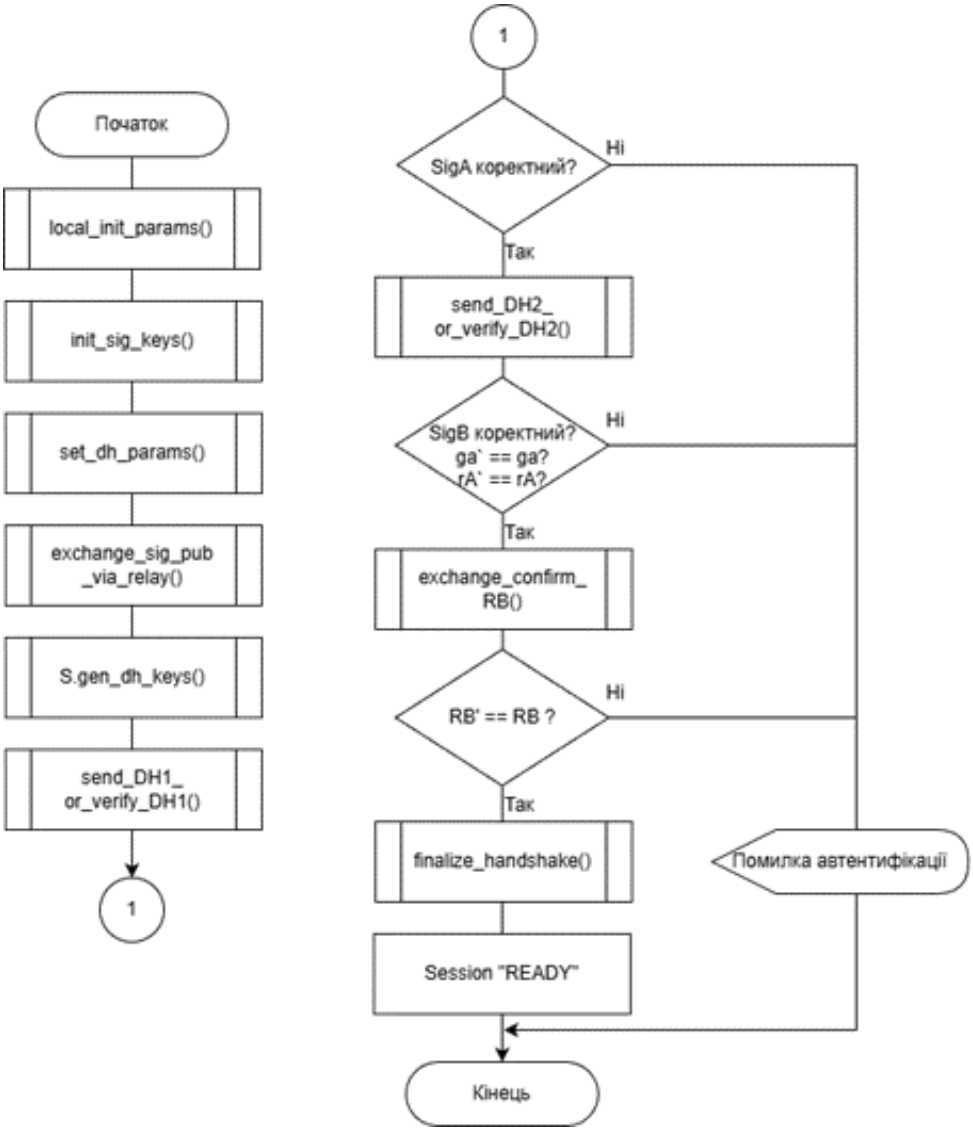
$$\mathbf{\Gamma}(\hat{q}) = \begin{bmatrix} 1 - 2(y^2 + z^2) & 2(xy - wz) & 2(xz + wy) \\ 2(xy + wz) & 1 - 2(x^2 + z^2) & 2(yz - wx) \\ 2(xz - wy) & 2(yz + wx) & 1 - 2(x^2 + y^2) \end{bmatrix}$$

$$\mathbf{\Gamma}(\hat{q})^{\mathrm{T}}\ \mathbf{\Gamma}(\hat{q}) \equiv I\ (mod\ p)$$

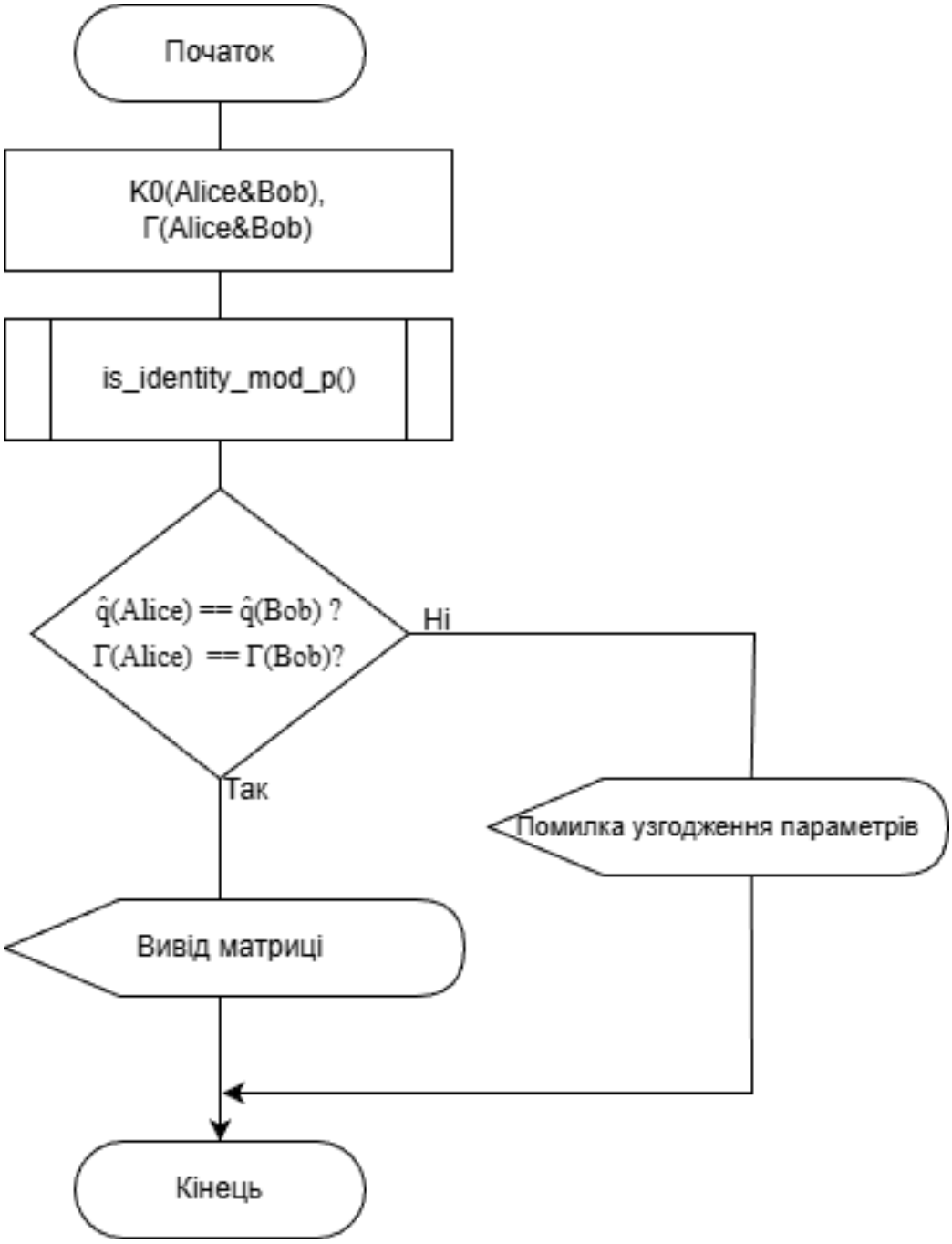
АРХІТЕКТУРА ПРОГРАМНОГО ЗАСОБУ ЗАХИЩЕНОГО ПЕРЕДАВАННЯ ІНФОРМАЦІЇ



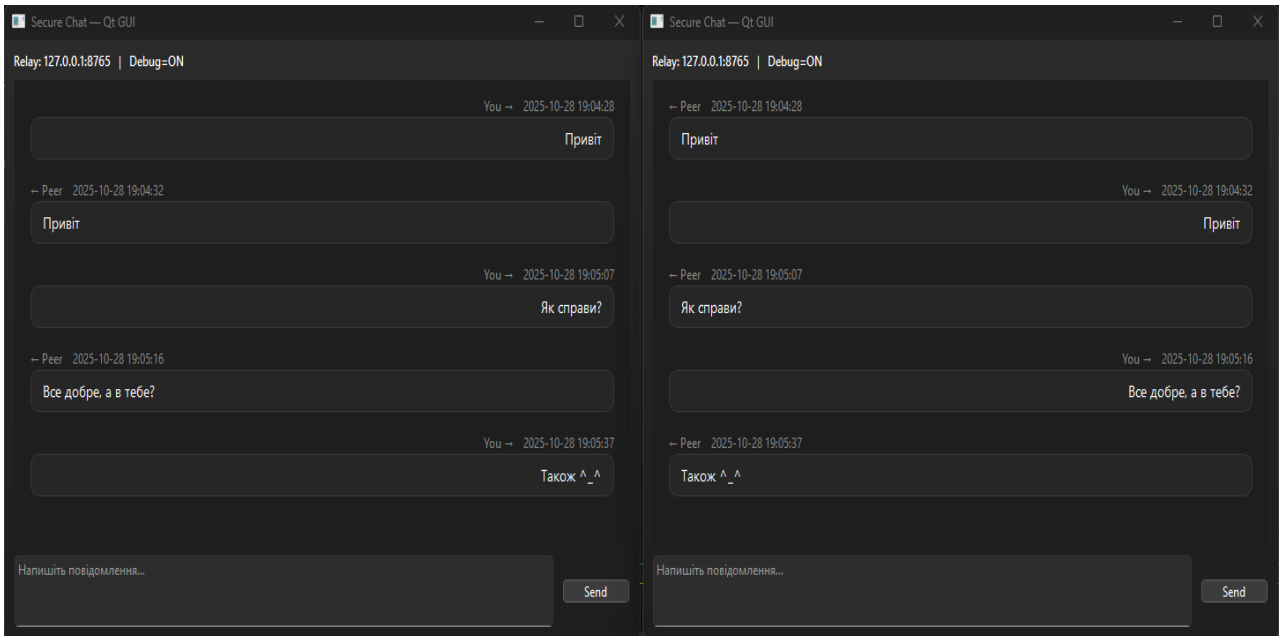
АЛГОРИТМ ВЗАЄМНОЇ АВТЕНТИФІКАЦІЇ ТА ВСТАНОВЛЕННЯ СПІЛЬНОГО СЕАНСОВОГО КЛЮЧА



АЛГОРИТМ УЗГОДЖЕННЯ ПАРАМЕТРІВ ТА ПЕРЕВІРКИ ОРТОГОНАЛЬНОСТІ ПОВОРОТНОЇ МАТРИЦІ



РЕЗУЛЬТАТИ ОБМІНУ ПОВІДОМЛЕННЯМИ МІЖ КЛІЄНТАМИ



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	3	4
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	41	42
Середньоарифметична сума балів $СБ_c$	41		

$$K_{\text{ІНТ}} = 1 \cdot \frac{1{,}093}{0{,}734} = 1{,}49$$

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

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