Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Удосконалення протоколу захищеного сеансового з'єднання у веб-застосунках на основі еліптичних кривих та доказів з нульовим розголошенням.

Виконав: здобувач 2-го курсу,
групи 2КІТС-24м
спеціальності 125– Кібербезпека
та захист інформації
Освітня програма – Кібербезпека
інформаційних технологій та систем
(шифр і назва напряму підготовки, спеціальності)

Волинець С. Ю.
(прізвище та ініціали)

Керівник:
Салієва О. В.
(прізвище та ініціали)
«09» грудня 2025 р.

Опонент:
Черняк О. І.
(прізвище та ініціали)
«09» грудня 2025 р.

Допущено до захисту
Голова секції УБ кафедри МБІС

Юрій ЯРЕМЧУК
"09" грудня 2025 р.

Вінниця ВНТУ - 2025 рік

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека та захист інформації
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС

_____ **Юрій ЯРЕМЧУК**
“ 09 ” грудня 2025 р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу студенту
Волинцю Сергію Юрійовичу
(прізвище, ім`я, по-батькові)

1. Тема роботи «Удосконалення протоколу захищеного сеансового з'єднання у веб-застосунках на основі еліптичних кривих та доказів з нульовим розголошенням».
Керівник роботи Салієва Ольга Володимирівна, доктор філософії (PhD) за спеціальністю 125 «Кібербезпека», доцент каф. МБІС
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від “24” вересня 2025 року № 313
2. Строк подання студентом роботи 02.12.2025 р.
3. Вихідні дані до роботи: Електронні джерела, підручники та наукові статті по темі. Існуюче програмне забезпечення, яке стосується теми магістерської кваліфакаційної роботи.
4. Зміст текстової частини: Аналіз теоретичних основ побудови захищених сеансових протоколів, криптографії на еліптичних кривих та технології доказів з нульовим розголошенням. Розробка архітектури, математичної моделі та алгоритмів удосконаленого протоколу сеансового з'єднання.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень): Презентація роботи, графік порівняння розмірів ключів ECC та RSA, концептуальна схема архітектури інтеграції ZKP у TLS, блок-схема математичної моделі ZKP-автентифікації, діаграма послідовності алгоритму удосконаленого рукостискання, діаграми та графіки результатів експериментального дослідження ефективності протоколу.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдан прийня
Основна частина	Салієва О. В., доцент кафедри МБІС		
Розділ І	Салієва О. В., доцент кафедри МБІС		
Розділ ІІ	Салієва О. В., доцент кафедри МБІС		
Розділ ІІІ	Салієва О. В., доцент кафедри МБІС		
Економічна частина	Ратушняк О. Г., доцент кафедри ЕПВМ, к.т.н.		

7.Дата видачі завдання 24 вересня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи		Приміт
1	Ознайомлення з темою та постановкою завдання	24.09.2025	26.09.2025	
2	Аналіз предметної області, дослідження існуючих протоколів та криптографічних примітивів	26.09.2025	03.10.2025	
3	Розробка архітектури та математичної моделі автентифікації	03.10.2025	07.10.2025	
4	Програмна реалізація протоколу	07.10.2025	20.10.2025	
5	Проведення тестування програмного забезпечення	20.10.2025	27.10.2025	
6	Оформлення пояснювальної записки та графічної частини роботи	27.10.2025	10.11.2025	
7	Підготовка презентаційних матеріалів та доповіді для захисту роботи.	10.11.2025	21.11.2025	
8	Попередній захист роботи	21.11.2025	30.11.2025	
9	Захист магістерської кваліфікаційної роботи	09.12.2025	09.12.2025	

Студент _____ Волинець С.Ю
(підпис)

Керівник роботи _____ Салієва О.В
(підпис)

АНОТАЦІЯ

УДК 004.056.55

Волинець С. Ю. Удосконалення протоколу захищеного сеансового з'єднання у веб-застосунках на основі еліптичних кривих та доказів з нульовим розголошенням.

Магістерська кваліфікаційна робота зі спеціальності 125 – «Кібербезпека та захист інформації», освітня програма «Кібербезпека інформаційних технологій та систем». – Вінниця: ВНТУ, 2025. 103 с.

На укр. мові. Бібліогр.: 36 назв; рис.: 11; табл. 3.

У роботі розглянуто проблему підвищення стійкості та ефективності протоколів захищеного сеансового з’єднання у веб-застосунках. Проведено аналіз сучасних підходів до побудови протоколів на основі еліптичних кривих та технологій доказів з нульовим розголошенням (Zero-Knowledge Proofs).

Визначено недоліки традиційних рішень, що стосуються компромісу між продуктивністю та рівнем безпеки. Запропоновано удосконалений протокол сеансової автентифікації, який поєднує механізми еліптичних кривих із доказами з нульовим розголошенням для забезпечення високого рівня конфіденційності без збільшення обчислювальних витрат.

Розроблено математичну модель протоколу, здійснено його формальний опис та реалізацію експериментального зразка. Проведено тестування розробленого рішення у контексті типових веб-застосунків. Отримані результати підтверджують підвищення ефективності обміну ключами та зниження ризику компрометації даних під час встановлення сеансового з’єднання.

Розроблений протокол може бути інтегрований у сучасні веб-фреймворки та системи захищеного доступу.

Ключові слова: захищене з’єднання, протокол, еліптичні криві, нульове розголошення, веб-застосунок, криптографія.

ABSTRACT

UDC 004.056.55

Volynets S. Yu. Improvement of the protected session connection protocol in web applications based on elliptic curves and zero-knowledge proofs.

Master's Qualification Thesis in specialty 125 – "Cybersecurity and Information Protection", educational program "Cybersecurity of Information Technologies and Systems". – Vinnytsia: VNTU, 2025. 103 p.

In Ukrainian. Ref.: 36; fig.: 11; tabl. 3.

The thesis addresses the problem of improving the resilience and efficiency of secure session protocols in web applications. An analysis of modern approaches to protocol design based on elliptic curve cryptography and zero-knowledge proofs (ZKPs) is conducted.

The drawbacks of traditional solutions concerning the balance between performance and security level are identified. An enhanced session authentication protocol is proposed, combining elliptic curve mechanisms with zero-knowledge proofs to ensure a high level of confidentiality without increasing computational overhead.

A mathematical model of the protocol is developed, formally described, and implemented as an experimental prototype. Testing of the developed solution within typical web applications has confirmed improved key exchange efficiency and reduced risk of data compromise during session establishment.

The proposed protocol can be integrated into modern web frameworks and secure access systems.

Keywords: secure connection, protocol, elliptic curves, zero-knowledge proof, web application, cryptography.

ЗМІСТ

ВСТУП

Актуальність. У сучасному цифровому середовищі питання безпеки обміну даними у веб-застосунках набуває особливої актуальності через зростання кількості атак, спрямованих на перехоплення, підробку або компрометацію сеансових ключів. Більшість сучасних веб-сервісів покладаються на протоколи захищеного сеансового з'єднання, що забезпечують автентифікацію, конфіденційність та цілісність переданих даних. Водночас зростання продуктивності обчислювальних систем і розвиток квантових технологій створюють передумови для перегляду традиційних криптографічних механізмів, які використовуються у таких протоколах.

Традиційні криптографічні методи, що лежать в основі поширених протоколів, зокрема SSL/TLS, вразливі до нових видів атак, пов'язаних із посиленням обчислювальних потужностей та розвитком аналітичних інструментів. Використання еліптичних кривих дозволяє досягти високого рівня криптостійкості за умови суттєвого зниження обчислювальних витрат, що особливо важливо для веб-застосунків із великою кількістю одночасних з'єднань. Додаткове впровадження доказів з нульовим розголошенням дозволяє підвищити рівень довіри між учасниками протоколу, забезпечуючи автентифікацію без необхідності розкриття конфіденційних даних.

Мета і задачі дослідження. Метою роботи є розробка удосконаленого протоколу захищеного сеансового з'єднання у веб-застосунках із використанням еліптичних кривих і доказів з нульовим розголошенням для забезпечення підвищеного рівня конфіденційності, автентифікації та стійкості до атак.

Задачами дослідження є:

– проведення аналізу сучасних протоколів захищеного обміну даними (SSL/TLS, DTLS, QUIC) та виявлення їх вразливостей;

– дослідження математичних основ використання еліптичних кривих у криптографічних протоколах;

– аналіз принципів побудови систем автентифікації на основі доказів з нульовим розголошенням;

– розробка удосконаленого протоколу сеансового з’єднання, що поєднує переваги еліптичних кривих і нульового розголошення;

– створення алгоритму узгодження ключів і перевірки автентичності без розкриття приватних даних;

– здійснення програмної реалізації та експериментальної перевірки ефективності розробленого протоколу.

Об’єкт дослідження – процес встановлення та підтримання захищеного сеансового з’єднання між клієнтом і сервером у веб-застосунках.

Предмет дослідження – методи, моделі та алгоритми побудови протоколів захищеного сеансового з’єднання із застосуванням еліптичних кривих та доказів з нульовим розголошенням.

Наукова новизна: удосконалено підхід до побудови протоколів захищеного з’єднання шляхом поєднання криптографічних операцій на еліптичних кривих із доказами з нульовим розголошенням, що дозволяє зменшити ймовірність компрометації автентифікаційних даних.

Розроблено модель обміну ключами, яка забезпечує автентифікацію сторін без передачі секретної інформації та характеризується нижчою обчислювальною складністю у порівнянні з класичними схемами.

Практична цінність: розроблений протокол може бути інтегрований у сучасні веб-фреймворки (ASP.NET, Node.js, Django) для підвищення рівня безпеки сеансових з’єднань без значного впливу на швидкодію. Отримані результати можуть бути використані у системах електронної комерції, банківських веб-сервісах та корпоративних застосунках, де важливо забезпечити автентифікацію користувачів і цілісність даних без ризику їх розголошення.

Апробація: матеріали дослідження були апробовані у формі тез доповіді на Міжнародній науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2026)» [1].

1 ТЕОРЕТИЧНІ ОСНОВИ ПОБУДОВИ ЗАХИЩЕНИХ СЕАНСОВИХ ПРОТОКОЛІВ У ВЕБ-ЗАСТОСУНКАХ

Даний розділ присвячений ґрунтовному теоретичному дослідженню принципів забезпечення безпеки обміну даними у веб-застосунках. У ньому розглянуто концепцію сеансового з’єднання, основні властивості безпеки, класичні та сучасні протоколи обміну ключами, а також криптографічні механізми, що лежать в основі протоколів SSL/TLS.

Особливу увагу приділено застосуванню еліптичних кривих (Elliptic Curve Cryptography, ECC) у забезпеченні високої стійкості шифрування при мінімальних обчислювальних витратах, а також розглянуто технологію доказів з нульовим розголошенням (Zero-Knowledge Proofs, ZKP), як перспективний напрям підвищення довіри у процесах автентифікації без передачі секретних даних. Результати аналізу створюють основу для подальшої розробки удосконаленого протоколу захищеного сеансового з’єднання, що поєднує переваги ECC та ZKP.

1.1. Поняття, структура та принципи функціонування протоколів захищеного сеансового з’єднання

Протокол захищеного сеансового з’єднання визначається як складний, багатокомпонентний комплекс формалізованих правил, стандартів та криптографічних процедур. Його основне призначення полягає у встановленні та підтримці безпечного, автентифікованого та зашифрованого каналу зв'язку між двома сторонами, що взаємодіють у комп'ютерній мережі [2]. У контексті веб-застосунків, цими сторонами найчастіше виступають клієнтський веб-браузер та веб-сервер. Протоколи цього класу функціонують поверх стандартних транспортних протоколів, зазвичай TCP (Transmission Control Protocol), і забезпечують захист для протоколів прикладного рівня, таких як HTTP, перетворюючи його на HTTPS (HTTP Secure).

Архітектурно, у стеку протоколів TCP/IP, протоколи захищеного сеансу, як-от TLS (Transport Layer Security), займають позицію "проміжного шару" (shim layer). Вони розташовуються між транспортним рівнем (де забезпечується надійна

доставка пакетів) та прикладним рівнем (де формується логіка веб-застосунку). Така архітектура забезпечує так званий прозорий захист: прикладний рівень продовжує оперувати стандартними запитами та відповідями (наприклад, HTTP GET або POST), не маючи інформації про процеси шифрування. Своєю чергою, шар TLS перехоплює ці дані, інкапсулює їх у захищений "конверт" перед відправкою на транспортний рівень.

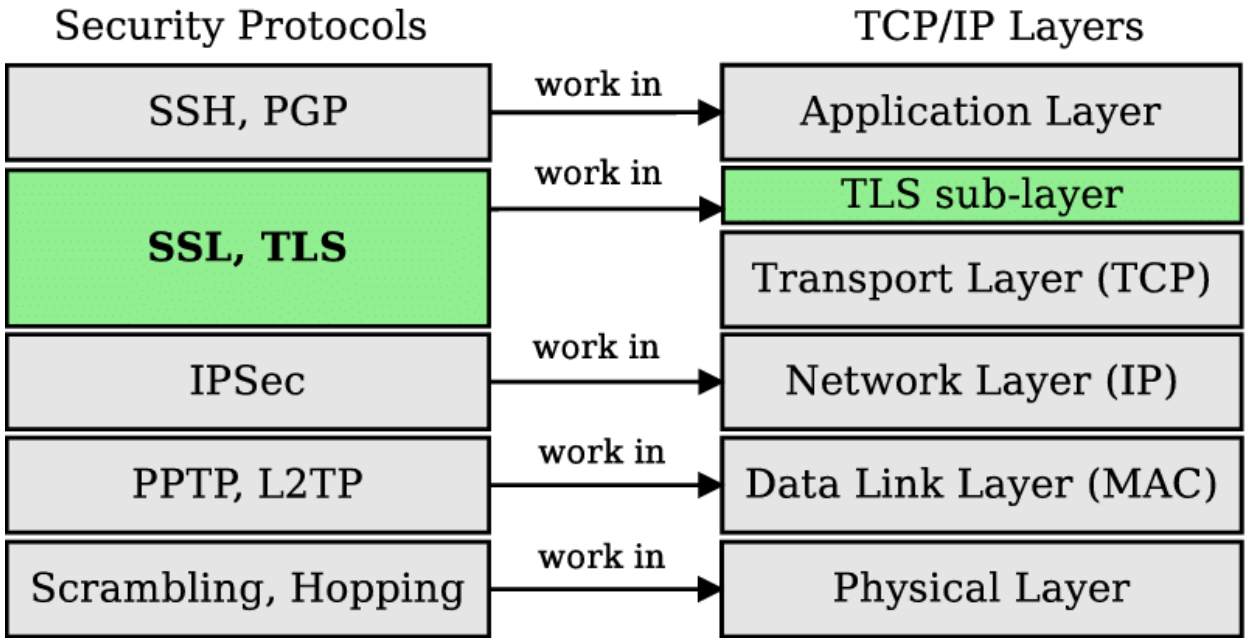


Рисунок 1.1 – Місце протоколів захищеного сеансу (на прикладі TLS) у стеку TCP/IP

Функціональність протоколів захищеного сеансу визначається наданням комплексного набору гарантій та послуг безпеки, що є критично важливими для сучасних веб-застосунків. Основними з них є:

– конфіденційність: ця послуга гарантує, що дані, які передаються, не можуть бути прочитані сторонніми особами, навіть якщо вони перехоплять мережевий трафік (атаки типу "eavesdropping"). Це досягається шляхом симетричного шифрування всього потоку даних сеансу за допомогою секретного ключа, унікального для цього сеансу;

– цілісність: ця послуга забезпечує гарантію того, що дані не були модифіковані (випадково чи навмисно) під час передачі. Зловмисник не може непомітно змінити зміст повідомлення. Це реалізується шляхом додавання до кожного блоку даних криптографічного коду автентифікації повідомлення (MAC)

або через інтегровані режими автентифікованого шифрування (AEAD), які дозволяють отримувачу миттєво виявити будь-яку спробу підміни даних [3];

– автентифікація: ця послуга надає сторонам можливість перевірити та підтвердити ідентичність одна одної. У переважній більшості веб-взаємодій використовується одностороння автентифікація: сервер доводить свою ідентичність клієнту. Для цього сервер надає свій цифровий сертифікат (стандарту X.509), виданий довіреним Центром сертифікації (CA). Клієнт перевіряє валідність цього сертифіката та його цифровий підпис, убезпечуючи себе від атак "людина посередині" (MitM);

– досконала пряма секретність: це критично важлива властивість сучасних протоколів, яка гарантує, що компрометація довгострокових ключів (наприклад, викрадення приватного ключа сервера) не дозволить зловмиснику розшифрувати минулі сеанси, які він міг записати. Це фундаментальна перевага перед застарілими методами. PFS досягається шляхом генерації унікальних, тимчасових (ефемерних) сеансових ключів для кожного з'єднання, які знищуються після завершення сеансу [4].

Забезпечення цього комплексного набору гарантій безпеки не є миттєвим процесом. Воно вимагає структурованої взаємодії між клієнтом та сервером, яка логічно поділяється на два фундаментально різних етапи. Перший етап – це початкове "знайомство" та встановлення довіри, а другий – безпосередня передача захищених даних. Ці етапи реалізуються окремими під-протоколами, кожен з яких вирішує власний набір завдань, що мають різні вимоги до обчислювальних ресурсів та криптографічних примітивів.

Роботу сеансового протоколу можна концептуально розділити на дві основні, послідовні фази, кожна з яких регламентується власним під-протоколом.

Фаза 1: Протокол рукостискання (Handshake Protocol) Це початковий, обчислювально інтенсивний етап встановлення з'єднання. Під час цієї фази клієнт і сервер ще не мають спільних секретів і спілкуються через відкритий канал. Їхнє завдання — виконати низку обмінів повідомленнями для того, щоб безпечно домовитись про параметри та згенерувати спільний сеансовий ключ. Основні задачі

цієї фази включають:

– узгодження параметрів: клієнт надсилає повідомлення ClientHello, в якому пропонує список підтримуваних ним криптографічних параметрів: версії протоколу, набори шифрів (cipher suites), підтримувані групи еліптичних кривих тощо. Набір шифрів — це комбінований набір алгоритмів (наприклад, алгоритм обміну ключами, алгоритм симетричного шифрування та алгоритм MAC). Сервер у відповідь надсилає ServerHello, обираючи з запропонованого списку один конкретний набір параметрів, який буде використано [2];

– автентифікація сторін: сервер надсилає клієнту свій сертифікат (Certificate) та доказ володіння відповідним приватним ключем (CertificateVerify), що дозволяє клієнту перевірити його автентичність. Клієнт виконує валідацію сертифіката, перевіряючи ланцюжок довіри до кореневого CA та відповідність доменного імені;

– генерація спільного секрету: сторони використовують асиметричний механізм обміну ключами. У сучасних протоколах це, як правило, ефемерний алгоритм Діффі-Хеллмана на еліптичних кривих (ECDHE). Кожна сторона генерує тимчасову пару ключів, обмінюється відкритими ключами та обчислює однаковий спільний секрет (pre-master secret). Цей секрет ніколи не передається у відкритому вигляді;

– виведення ключів: ртриманий спільний секрет не використовується напряму. Натомість, він подається на вхід спеціалізованої криптографічної функції виведення ключів (Key Derivation Function, KDF), такої як HKDF (HMAC-based KDF) у TLS 1.3 [4]. Ця функція генерує з одного секрету цілий набір симетричних сеансових ключів, необхідних для роботи протоколу запису (наприклад, окремі ключі для шифрування від клієнта до сервера, від сервера до клієнта та для перевірки цілісності).

Таким чином, успішне завершення протоколу рукостискання призводить до того, що обидві сторони (клієнт і сервер) володіють ідентичним набором симетричних сеансових ключів. Важливо, що ці ключі ніколи не передавалися мережею у відкритому вигляді, а були незалежно обчислені обома сторонами на основі спільного секрету, отриманого асиметричними методами. Саме ці ключі,

згенеровані з використанням ефемерних параметрів (ECDHE), забезпечують досконалу пряму секретність [4] і стають основою для роботи наступної, основної фази – протоколу запису.

На рисунку 1.2 наведена узагальнена схема встановлення захищеного сеансового з'єднання за протоколом TLS, що ілюструє послідовність обміну повідомленнями.

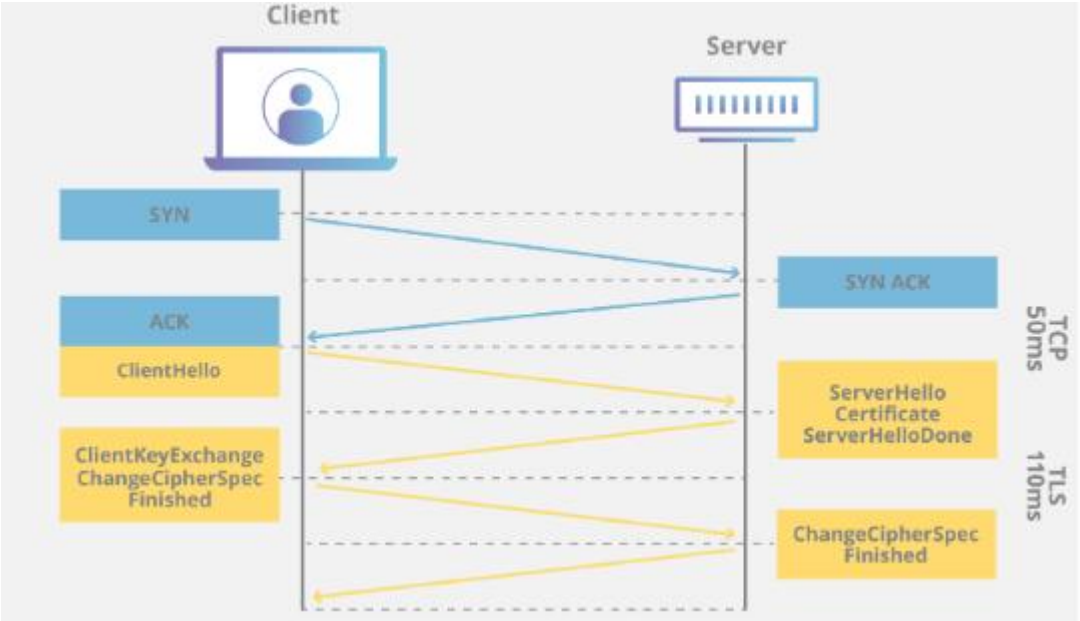


Рисунок 1.2 – Схема встановлення захищеного сеансового з'єднання TLS [2]

Фаза 2: Протокол запису (Record Protocol) Після успішного завершення рукостискання та встановлення сеансових ключів, протокол переходить у фазу захищеної передачі даних. За це відповідає протокол запису. Його функціонування є циклічним і значно менш ресурсомістким, ніж рукостискання. Воно складається з наступних кроків для кожного блоку даних, що відправляється:

Фрагментація: Дані, що надходять від прикладного рівня (наприклад, частина HTML-сторінки), розбиваються на блоки (записи) керованого розміру. Це необхідно для ефективної передачі через TCP та для буферизації.

Застосування захисту: До кожного запису застосовуються узгоджені під час рукостискання криптографічні операції. У сучасних протоколах це, як правило, автентифіковане шифрування з приєднаними даними (AEAD) [5]. Режими AEAD (наприклад, AES-GCM, ChaCha20-Poly1305) є високоефективними, оскільки вони одночасно виконують шифрування (для конфіденційності) та обчислення тегу

автентифікації (для цілісності) за один прохід.

Передача: Сформований захищений запис (що складається з зашифрованих даних та тегу автентифікації) передається транспортному рівню (TCP) для відправки мережею.

На відміну від обчислювально "важкого" асиметричного рукостискання, яке виконується лише один раз на початку сеансу, протокол запису покладається виключно на високошвидкісні симетричні алгоритми. Це дозволяє досягти високої пропускної здатності при шифруванні великих обсягів даних (наприклад, потокового відео або великих файлів) з мінімальними накладними витратами на обчислення. Такий двофазний підхід є ключовим компромісом, що забезпечує баланс між надійним криптографічним захистом та продуктивністю веб-застосунку.

На стороні отримувача виконуються зворотні операції: спершу перевіряється тег автентифікації (MAC). Якщо він коректний, запис дешифрується, і відновлені дані передаються прикладному рівню (веб-браузеру або серверу).

Слід зазначити, що сучасні протоколи захищеного сеансу є результатом тривалої еволюції та роботи над помилками. Історично, першим широко впровадженим протоколом був SSL (Secure Sockets Layer), розроблений компанією Netscape. Однак, ранні версії SSL (аж до SSL 3.0) містили суттєві архітектурні та криптографічні вразливості, що зрештою призвело до їх повної компрометації, яскравим прикладом якої стала атака POODLE у 2014 році. Ця атака остаточно довела небезпечність подальшого використання SSL [9].

У відповідь на виявлені проблеми, організація IETF (Internet Engineering Task Force) взяла на себе стандартизацію протоколу, перейменувавши його в TLS (Transport Layer Security). Починаючи з TLS 1.0 (який по суті був SSL 3.1), кожна наступна версія (1.1, 1.2, і особливо 1.3) була спрямована на усунення вразливостей, посилення криптографічних примітивів та покращення продуктивності. Сучасний стандарт TLS 1.3 [4] являє собою значний крок уперед, усунувши застарілі режими та обов'язково вимагаючи механізмів з досконалою прямою секретністю (PFS). Таким чином, розробка та вдосконалення сеансових

протоколів – це безперервний процес, керований міжнародними органами стандартизації [10] та дослідницькою спільнотою.

Водночас, перехід на нові стандарти безпеки диктується не лише еволюцією протоколів, а й появою нових вимог до безпечної розробки програмного забезпечення. Згідно з рекомендаціями NIST SP 800-218 [28], сучасні механізми захисту каналів зв'язку повинні інтегруватися на етапі проєктування архітектури (Security by Design), що робить вибір протоколу TLS 1.3 безальтернативним для систем критичної інфраструктури.

Ця еволюція від SSL до TLS 1.3 чітко демонструє тенденцію до посилення криптографічних основ та відмову від компромісних рішень минулого. Наступний підрозділ детально розгляне саме ці основи, зокрема математичний апарат, що зробив можливими такі вдосконалення, як ECDHE та досконала пряма секретність, і який є фундаментом для даного дослідження.

1.2 Криптографічні основи побудови сеансових протоколів та застосування еліптичних кривих

Надійність протоколів захищеного сеансу, детально розглянутих у попередньому підрозділі, безпосередньо базується на стійкості використаних криптографічних примітивів. Сучасні сеансові протоколи, зокрема TLS, реалізують так звану гібридну криптосистему. Цей підхід є вимушеним та оптимальним компромісом, що поєднує фундаментально різні переваги двох основних класів криптографії.

Асиметрична криптографія (криптографія з відкритим ключем). Використовується виключно під час початкової фази рукостискання. Її відносно висока обчислювальна складність та більший розмір повідомлень роблять її непридатною для шифрування всього потоку даних. Однак вона є незамінною для вирішення двох ключових завдань:

– автентифікація сторін: за допомогою механізмів цифрового підпису (наприклад, ECDSA або RSA-PSS) сервер доводить клієнту володіння приватним ключем, що відповідає його публічному сертифікату;

– узгодження ключів: за допомогою алгоритмів обміну ключами (наприклад, ECDHE) сторони безпечно встановлюють спільний секрет через незахищений канал.

Симетрична криптографія (криптографія з секретним ключем). Використовується для захисту даних у фазі протоколу запису. Після того, як сеансовий ключ безпечно узгоджено за допомогою асиметричних методів, швидкісні симетричні алгоритми (наприклад, AES, ChaCha20) забезпечують високоефективне потокове шифрування та дешифрування основного масиву даних [5].

Ця гібридна модель є фундаментальною для сучасної прикладної криптографії. Використання виключно асиметричних алгоритмів для шифрування всього сеансу є обчислювально нежиттєздатним; операції з відкритим ключем на порядки повільніші, ніж симетричні аналоги. Тому протоколи, як-от TLS, використовують "важку" асиметричну криптографію лише на короткий проміжок часу на початку (під час рукостискання) з єдиною метою: безпечно узгодити один "головний" симетричний ключ. Як тільки цей ключ встановлено, протокол негайно перемикається на "швидкісні" симетричні шифри (як AES) для решти комунікації. Це забезпечує оптимальний баланс між високою безпекою, необхідною для встановлення ключа, та високою продуктивністю, необхідною для передачі даних.

Історично, домінуючими асиметричними алгоритмами були RSA (для обміну ключами та підписів) та класичний алгоритм Діффі-Хеллмана (DH) на скінченних полях. Безпека цих алгоритмів базується на складних математичних задачах: проблема факторизації великих цілих чисел (для RSA) та проблема дискретного логарифмування (DLP) у скінченній мультиплікативній групі (для DH). Зі зростанням обчислювальних потужностей та розвитком методів криптоаналізу, для підтримки належного рівня безпеки доводиться експоненційно збільшувати довжину ключів для цих алгоритмів. Наприклад, для досягнення 128-бітного рівня безпеки (еквівалент AES-128), RSA вимагає ключ довжиною 3072 біти [6].

Використання таких довгих ключів призводить до низки суттєвих проблем:

– високе обчислювальне навантаження: операції з 3072-бітними числами є

вкрай ресурсомісткими, що створює значне навантаження на CPU сервера під час фази рукостискання;

– збільшення затримки: передача довгих ключів та сертифікатів збільшує обсяг даних, якими необхідно обмінятися, що подовжує час встановлення з'єднання, особливо у мобільних мережах;

– проблеми з мобільними та IoT-пристроями: пристрої з обмеженими обчислювальними ресурсами та живленням (смартфони, сенсори) зазнають труднощів при виконанні таких складних операцій [8].

Відповіддю на ці виклики стало широке впровадження криптографії на еліптичних кривих (Elliptic Curve Cryptography, ECC), що була запропонована незалежно Кобліцем та Міллером у 1985 році. Безпека ECC базується на значно складнішій математичній задачі – проблемі дискретного логарифмування на еліптичній кривій (ECDLP) [7]. Задача ECDLP полягає у знаходженні цілого числа k (приватний ключ) за відомими точками P (публічний ключ) та G (базова точка кривої) з рівняння P = kG. Вважається, "найкращі" відомі алгоритми для розв'язання ECDLP є менш ефективними, ніж алгоритми для факторизації або класичного DLP, що дозволяє досягти того ж рівня безпеки при значно менших розмірах ключів [11].

Складність задачі ECDLP полягає у природі операцій на еліптичній кривій. Обчислення P = kG (множення точки G на скаляр k) є відносно швидкою операцією, що виконується шляхом серії додавань та подвоєнь точок. Однак зворотна операція – знаходження k, знаючи лише P та G – вимагає повного перебору у групі точок, оскільки для еліптичних кривих не існує відомих "коротких шляхів" (алгоритмів типу індекс-калькулюсу, що є ефективними проти класичного DLP). Найкращі відомі атаки на ECDLP мають експоненційну складність (наприклад, алгоритм Поліга-Геллмана або алгоритм Полларда). Саме ця висока обчислювальна складність "зворотної" операції при легкості "прямої" (властивість односторонньої функції) і робить еліптичні криві надійним фундаментом для криптосистем [7].

Ця фундаментальна відмінність у ефективності є ключовою перевагою ECC. Вона наочно продемонстрована у Таблиці 1.1, що порівнює еквівалентні за стійкістю розміри ключів для різних типів криптосистем згідно з рекомендаціями

NIST.

Таблиця 1.1 – Порівняння еквівалентних розмірів ключів для симетричних алгоритмів, ECC та RSA/DH [6, 11].

Рівень безпеки (біт)	Симетричний ключ	Ключ ECC	Ключ RSA/DH
80	80	160-223	1024
112	112	224-255	2048
128	128	256-383	3072
192	192	384-511	7680
256	256	512+	15360

Як видно з таблиці, для досягнення 128-бітного рівня безпеки (сучасний стандарт для веб-застосунків) ECC потребує ключ довжиною лише 256 біт, тоді як RSA вимагає 3072 біти. Це означає, що ECC пропонує значно вищу "щільність безпеки" на біт ключа.

Для візуалізації цієї експоненційної розбіжності, на рисунку 1.3 показано порівняльне зростання розмірів ключів для ECC та RSA при підвищенні вимог до рівня безпеки.

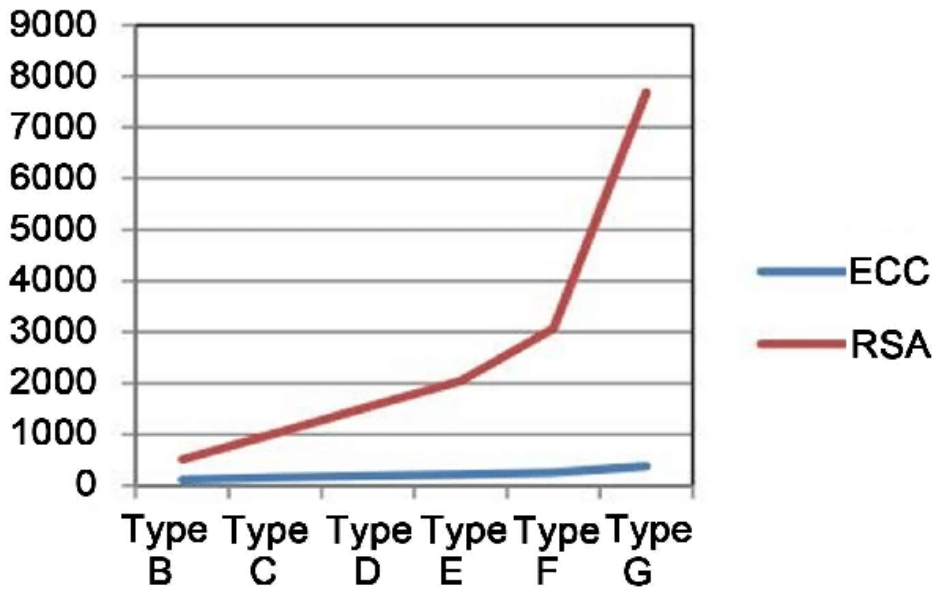


Рисунок 1.3 – Графічне порівняння зростання розмірів ключів RSA та ECC для еквівалентних рівнів безпеки

Рисунок 1.3 наочно демонструє нелінійну залежність між рівнем безпеки та необхідним розміром ключа для RSA. У той час як крива для ECC показує майже лінійне зростання, крива RSA зростає експоненційно. Це має прямі практичні наслідки: подвоєння рівня безпеки (наприклад, перехід зі 128-біт на 256-біт) для ECC вимагає простого подвоєння ключа (з 256 до 512 біт), тоді як для RSA це вимагає переходу від 3072 біт до 15360 біт [6]. Така різниця у "вартості" безпеки робить RSA непридатним для довгострокових стандартів у високопродуктивних системах, що й зумовило неминучий перехід галузі на ECC.

Переваги ECC у ефективності призвели до її домінування у сучасних стандартах, зокрема у TLS 1.3, де підтримка ECC є фактично обов'язковою [4]. У сеансових протоколах ECC застосовується у двох основних формах:

– ECDHE (Elliptic Curve Diffie-Hellman Ephemeral): це де-факто стандартний механізм узгодження ключів у TLS 1.3. Він є реалізацією алгоритму Діффі-Хеллмана з використанням груп точок на еліптичній кривій. Додаток "Ephemeral" (ефемерний) означає, що для кожного сеансу генерується нова, тимчасова пара ключів, яка негайно знищується після завершення рукостискання. Це є ключовим механізмом забезпечення досконалої прямої секретності (PFS) [13];

– ECDSA (Elliptic Curve Digital Signature Algorithm): використовується для створення та перевірки цифрових підписів. Найчастіше сервер використовує ECDSA для підписання критичних параметрів рукостискання своїм довгостроковим приватним ключем, доводячи клієнту, що він є легітимним власником сертифіката.

Застосування ECC дозволяє значно скоротити обчислювальне навантаження (наприклад, генерація підпису ECDSA є значно швидшою за RSA) та обсяг даних, що передаються під час рукостискання [13]. Це безпосередньо призводить до швидшого встановлення з'єднання, меншого споживання енергії на мобільних пристроях та можливості обслуговувати більшу кількість одночасних з'єднань на сервері.

Таким чином, еліптичні криві вирішили проблему ефективності та продуктивності, що стояла перед застарілими алгоритмами RSA та DH. Вони стали

сучасною основою для забезпечення двох із трьох стовпів безпеки: конфіденційності (через ECDHE) та автентичності (через ECDSA). Однак, як буде показано далі, незважаючи на ефективність ECDSA, сам підхід до автентифікації на основі сертифікатів має фундаментальні обмеження, пов'язані з довірою та розкриттям метаданих. Це створює потребу в дослідженні нових парадигм, таких як докази з нульовим розголошенням, які можуть функціонувати *поверх* ефективної бази, наданої еліптичними кривими.

1.3 Технологія доказів з нульовим розголошенням і можливості її використання у веб-протоколах

Концепція доведення з нульовим розголошенням. Докази з нульовим розголошенням (Zero-Knowledge Proofs, ZKP) є однією з фундаментальних концепцій сучасної криптографії, що була вперше представлена у 1985 році дослідниками Гольдвассер, Мікалі та Ракоффом. У своїй основі, ZKP — це криптографічний протокол, який дозволяє одній стороні, відомій як "Той, хто доводить" (Prover), переконати іншу сторону, відому як "Той, хто перевіряє" (Verifier), у істинності певного твердження, не розкриваючи при цьому *жодної* додаткової інформації, окрім самого факту істинності цього твердження [14].

Простіше кажучи, Prover може довести Verifier-у, що він володіє певним секретним знанням (наприклад, паролем, приватним ключем, розв'язком математичної задачі), не передаючи саме це знання.

Фундаментальні властивості ZKP. Будь-який протокол, що претендує на статус доказу з нульовим розголошенням, повинен задовольняти три обов'язкові властивості [15]:

1. Повнота: якщо твердження, яке доводить Prover, є істинним, і обидві сторони (Prover та Verifier) чесно дотримуються протоколу, то Verifier завжди (або з переважною імовірністю) прийме цей доказ як дійсний;
2. Обґрунтованість: якщо твердження є хибним, то нечесний Prover не зможе переконати чесного Verifier-а в істинності твердження, за винятком дуже малої, нехтуваної ймовірності (відомої як "помилка обґрунтованості");

3. Нульове розголошення: якщо твердження є істинним, Verifier не дізнається нічого, окрім того, що твердження істинне. Будь-який "діалог" (транскрипт) взаємодії, який Verifier міг би отримати, він міг би згенерувати (симулювати) самостійно, не взаємодіючи з Prover-ом. Це гарантує, що Prover не розкриває жодної суттєвої інформації про свій секрет.

Ці три властивості, взяті разом, представляють фундаментальний зсув у підходах до автентифікації. Традиційні криптографічні методи є "позитивними" в сенсі доведення: щоб довести, що ви знаєте пароль, ви повинні надіслати серверу сам пароль або його геш. ZKP, навпаки, є "негативним" у сенсі розголошення: він дозволяє підтвердити істинність твердження, не передаючи жодної корисної інформації, окрім самого факту істинності [15]. Саме ця дисоціація між актом верифікації та актом розкриття секрету робить ZKP настільки потужним інструментом для побудови систем, орієнтованих на конфіденційність.

Концептуальна модель ZKP: Печера Алі-Баби. Класичною та інтуїтивно зрозумілою ілюстрацією принципу ZKP є алегорія "Печера Алі-Баби", запропонована Квіскатером та Гійу.

Уявімо собі кільцеподібну печеру з єдиним входом та магічними дверима всередині, що з'єднують два коридори (A і B) і які можна відкрити лише знаючи секретне слово.

1. Пеггі хоче довести Віктору, що вона знає секретне слово, але не хоче його розкривати.

2. Пеггі заходить у печеру (Віктор не бачить, яким коридором, A чи B, вона пішла).

3. Віктор підходить до входу і вигукує, яким коридором Пеггі має повернутися (наприклад, "Вийди з коридору A!").

4. Якщо Пеггі знає секретне слово, вона може відкрити магічні двері та повернутися будь-яким вказаним шляхом.

5. Якщо вона не знає слова і зайшла, наприклад, у коридор A, а Віктор попросив її вийти з A, вона успішно пройде перевірку. Але якби він попросив її вийти з B, вона б не змогла цього зробити.

Таким чином, якщо Пеггі не знає слова, вона має 50% шансів "вгадати" і пройти перевірку. Однак, якщо повторити цей експеримент N разів, імовірність того, що вона (шахрайка) успішно пройде всі перевірки, становить $(1/2)^N$, що дуже швидко прямує до нуля. Після достатньої кількості ітерацій Віктор буде переконаний, що Пеггі знає секрет, хоча вона жодного разу його не назвала.

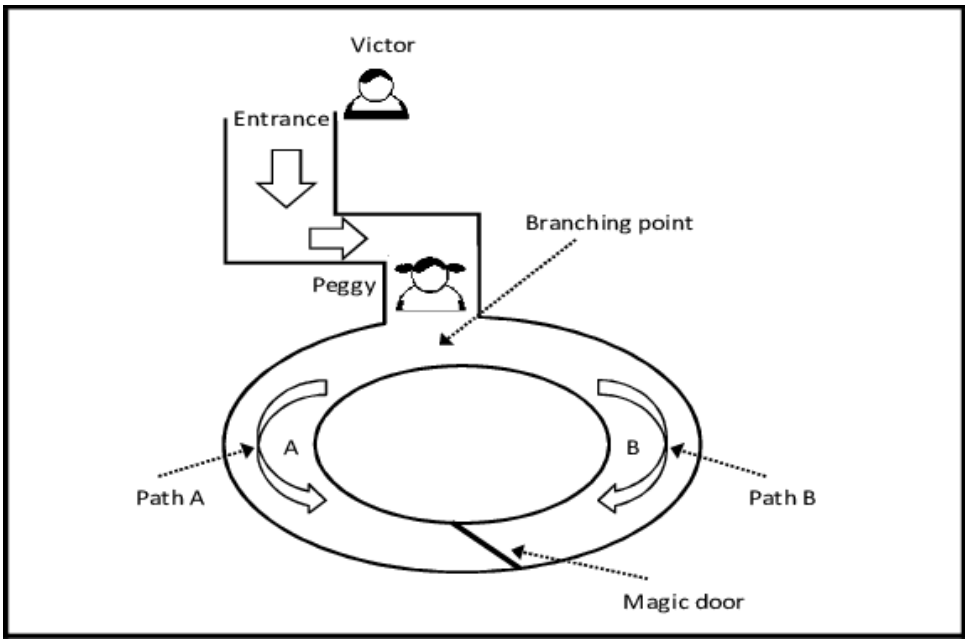


Рисунок 1.4 – Схематична ілюстрація протоколу доведення з нульовим розголошенням "Печера Алі-Баби"

Ця алегорія чудово демонструє всі три властивості у дії. Повнота: якщо Пеггі (Prover) чесна і знає секрет, вона *завжди* зможе вийти з будь-якого коридору, який назве Віктор, таким чином успішно проходячи 100% перевірок. Обґрунтованість: якщо Пеггі є шахрайкою і не знає секрету, вона може лише вгадати, яким шляхом увійти, сподіваючись, що Віктор назве той самий шлях для виходу. Її шанс на успіх в одній ітерації становить 50%. Однак імовірність обманути Віктора N разів поспіль становить $(1/2)^N$, що робить успішний обман практично неможливим вже після 20-30 ітерацій. Нульове розголошення: Віктор (Verifier) бачить лише, як Пеггі виходить з названого коридору. Він не бачить, як вона використовує секретне слово для відкриття дверей. Для нього кожна успішна ітерація виглядає однаково, і він не отримує жодної інформації, яка б допомогла йому самому дізнатися це секретне слово.

Існують інтерактивні та неінтерактивні типи ZKP.

Фундаментальні дослідження у сфері доказів з нульовим знанням, узагальнені у праці Джастіна Талера [26], класифікують сучасні протоколи на аргументи знання (Arguments of Knowledge) та інтерактивні докази. Для веб-середовища критичним є використання саме неінтерактивних аргументів, які дозволяють мінімізувати комунікаційні витрати до одного раунду обміну даними.

Описаний приклад з печерою є інтерактивним ZKP – він вимагає серії запитів та відповідей між Prover-ом та Verifier-ом. Це є ефективним для демонстрації концепції, але непрактичним для більшості веб-протоколів, де асинхронна комунікація та мінімізація обмінів є критичними.

Для вирішення цієї проблеми були розроблені неінтерактивні докази з нульовим розголошенням (NIZK). У NIZK, Prover може згенерувати доказ у вигляді одного повідомлення, яке Verifier може перевірити в будь-який час без подальшої взаємодії. Це досягається за допомогою евристики Фіата-Шаміра, яка замінює "випадкові виклики" від Verifier-а на результати криптографічної геш-функції, що застосовується до публічних даних та проміжних результатів Prover-а [16].

Цей перехід від інтерактивних до неінтерактивних доказів є абсолютно ключовим для практичного застосування у веб-протоколах. Сеансові протоколи, такі як TLS, оптимізовані для мінімізації кількості обмінів повідомленнями (RTTs). Вимога виконання десятків ітерацій "виклик-відповідь", як у прикладі з печерою, є неприйнятною, оскільки це катастрофічно вплинуло б на швидкість встановлення з'єднання. Необхідність в одному, стислому, асинхронному повідомленні, яке клієнт може згенерувати та надіслати, а сервер – перевірити у будь-який час, робить саме NIZK єдиним життєздатним варіантом для вдосконалення сучасних протоколів реального часу.

Сучасні реалізації ZKP: SNARKs та STARKs. У контексті практичного застосування, особливо у веб-застосунках, найбільший інтерес представляють дві технології:

– zk-SNARK (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge): неінтерактивні докази, які є "succinct" (стислими). Вони мають дві ключові переваги: дуже малий розмір доказу (порядку сотень байт) і надзвичайно швидка

верифікація (порядку мілісекунд), незалежно від складності обчислення, що доводиться. Їхній головний недолік – необхідність у так званій "довіреній початковій установці" (trusted setup) для генерації публічних параметрів [17];

– zk-STARK (Zero-Knowledge Scalable Transparent Argument of Knowledge): новіша технологія, яка є "scalable" (масштабованою) та "transparent" (прозорою). Прозорість означає відсутність потреби у довіреній початковій установці. Вони також вважаються стійкими до атак з використанням квантових комп'ютерів. Їхній компроміс – значно більший розмір доказу порівняно зі SNARKs [18].

Можливості використання у веб-протоколах. Потенціал ZKP для веб-застосунків та сеансових протоколів є надзвичайно великим, оскільки вони дозволяють фундаментально змінити модель довіри та автентифікації.

ZKP-автентифікація це найбільш очевидне застосування. Замість того, щоб надсилати пароль (або навіть його геш) на сервер, клієнт може довести серверу, що він знає пароль. Prover (клієнт) доводить твердження: "Я знаю секрет S, такий що H(S) = X", де X – це геш, що зберігається на сервері. Це робить викрадення бази даних гешів паролів значно менш небезпечним, оскільки самі геші не можуть бути використані для автентифікації [16].

У доведенні володіння приватним ключем замість традиційного цифрового підпису (як у ECDSA, де підпис є публічною інформацією), клієнт може довести володіння приватним ключем k, що відповідає публічному ключу P (де P = kG на еліптичній кривій), не розкриваючи жодної інформації, яка могла б допомогти в його компрометації. Це є прямою точкою інтеграції з криптографією на еліптичних кривих, розглянутою в підрозділі 1.2.

Анонімна авторизація та верифікація атрибутів. ZKP дозволяють користувачу довести, що він володіє певними атрибутами, не розкриваючи свою особу. Наприклад, у веб-застосунку користувач може довести, що:

1. "Я старше 18 років" (не розкриваючи дату народження).

2. "Мій кредитний рейтинг вище 700" (не розкриваючи точний рейтинг або фінансову історію).

3. "Я є членом групи 'Адміністратори'" (не розкриваючи свій конкретний

логін).

Таким чином, ZKP дозволяють побудувати веб-протоколи, що забезпечують "Privacy-by-Design" (конфіденційність за проектуванням), де автентифікація та авторизація відбуваються без надлишкового розголошення персональних даних, що є значним кроком уперед порівняно з існуючими сеансовими протоколами [17].

Отже, підрозділи 1.2 та 1.3 описують два стовпи, на яких будується дане дослідження. Еліптичні криві (ECC) надають ефективний обчислювальний базис, що дозволяє досягти високої криптографічної стійкості при малих розмірах ключів. Докази з нульовим розголошенням надають конфіденційний логічний механізм для автентифікації. Як буде показано далі, ці дві технології є глибоко синергетичними: найефективніші сучасні ZKP-схеми (зокрема, zk-SNARK) самі по собі будуються на математичному апараті еліптичних кривих, що підтримують спарювання [20]. Ця синергія дозволяє проектувати протоколи, що є одночасно і обчислювально ефективними (завдяки ECC), і конфіденційними (завдяки ZKP), що є метою даної роботи.

1.4 Огляд сучасних досліджень у сеансових протоколах із застосуванням еліптичних кривих та доказів з нульовим розголошенням

Сучасні дослідження у сфері захищених сеансових протоколів розвиваються у двох ключових напрямках, що безпосередньо стосуються теми даної роботи. Перший – це оптимізація та посилення вже існуючих механізмів на базі еліптичних кривих. Другий, більш інноваційний – це дослідження інтеграції доказів з нульовим розголошенням (ZKP) для фундаментальної зміни парадигм автентифікації та конфіденційності.

Ці два вектори досліджень не є взаємовиключними; навпаки, вони глибоко взаємопов'язані. Оптимізація еліптичних кривих (перший напрямок) часто забезпечує необхідний високопродуктивний математичний фундамент, необхідний для практичної реалізації доказів з нульовим розголошенням (другий напрямок). Проте, їх доцільно аналізувати окремо, оскільки вони вирішують різні рівні проблеми безпеки: перший фокусується на ефективності та стійкості

криптографічного механізму, тоді як другий переосмислює саму логіку автентифікації та довіри.

Сучасні дослідження у галузі еліптичних кривих. Хоча криптографія на еліптичних кривих є зрілою та стандартизованою технологією (як показано в 1.2), дослідження не припиняються. Останніми роками вони зосереджені на трьох основних сферах:

1. Пост-квантова гібридизація. Найбільш актуальним викликом для ECC є поява гіпотетичних квантових комп'ютерів, здатних за допомогою алгоритму Шора розв'язати задачу ECDLP, що повністю компрометує ECC. Оскільки повноцінні пост-квантові алгоритми (PQC) (наприклад, засновані на криптографії на ґратках) часто є менш ефективними, активно досліджуються гібридні схеми обміну ключами. У такому підході, наприклад в TLS, клієнт і сервер виконують *одночасно* два обміни ключами – один класичний (ECDHE) і один пост-квантовий (наприклад, Kyber). Спільний секрет об'єднується (конкатенується). Це гарантує, що з'єднання залишатиметься безпечним, доки *хоча б один* з алгоритмів не зламаний, забезпечуючи плавний перехід до PQC [19]. Такий гібридний підхід вважається найбільш прагматичним шляхом розвитку. Загроза квантових комп'ютерів полягає не лише в тому, що вони *з'являться*, але й у тому, що зловмисники можуть вже зараз застосовувати атаки типу "збирай зараз, розшифруй пізніше". Вони записують поточні, надійно зашифровані сеанси з наміром розшифрувати їх, коли (і якщо) потужний квантовий комп'ютер стане доступним. Шляхом комбінування ECDHE (швидкого та перевіреного) з пост-квантовим алгоритмом (який вважається квантово-стійким), сеанс залишається безпечним як проти класичних, так і проти квантових атак, вирішуючи одночасно поточні та майбутні загрози;

2. Розробка нових кривих. Триває дослідження нових, більш спеціалізованих еліптичних кривих. Наприклад, криві, що оптимізовані для ефективної реалізації криптографічних спарювань. Ці операції є математичною основою для багатьох ZKP-схем (зокрема, zk-SNARK), що створює прямий зв'язок між двома технологіями, які розглядаються [20];

3. Полегшені реалізації. Для стрімко зростаючого сектору Інтернету речей

(IoT), де пристрої мають жорсткі обмеження щодо обчислювальної потужності та енергоспоживання, стандартні реалізації ECC можуть бути надто важкими. Ведеться активна розробка полегшених протоколів та апаратних прискорювачів для ECC, що дозволяють інтегрувати надійний захист (як-от ECDHE) у сенсори, медичні імпланти та інші мікропристрої [8].

Це дослідження має критичне значення, оскільки Інтернет речей являє собою масштабне розширення "поверхні атаки". Мільярди малопотужних пристроїв, якщо їх залишити незахищеними, можуть бути скомпрометовані та використані у великомасштабних ботнетах. Впровадження ефективної, низькоенергетичної криптографії на еліптичних кривих, таким чином, є не просто оптимізацією, а фундаментальною вимогою для забезпечення безпеки наступного покоління підключених пристроїв.

Дослідження та реалізація ZKP у веб-протоколах**.** Інтеграція ZKP у сеансові протоколи є значно новішою, але більш революційною сферою. Дослідження тут фокусуються на заміні або доповненні традиційних механізмів автентифікації:

- ZKP для автентифікації: замість відправки пароля (або його геша) досліджуються протоколи, де клієнт доводить серверу володіння секретом, що відповідає збереженому гешу (як описано в 1.3). Це унеможливлює атаки, засновані на витоку баз даних [16]. Більш просунуті схеми, такі як OPAQUE, пропонують протоколи автентифікованого обміну ключами з захистом пароля, які стійкі до атак перебору "офлайн" та не розкривають серверу сам пароль навіть під час реєстрації [21];
- доведення володіння ключем: замість традиційного цифрового підпису (ECDSA), де підпис є публічною інформацією, що може бути залогована, досліджуються схеми, де клієнт (Prover) генерує неінтерактивний доказ того, що він володіє приватним ключем k, що відповідає публічному ключу P. Це особливо актуально для систем цифрової ідентичності, де користувач доводить право власності на "децентралізований ідентифікатор". Це конкретне застосування являє собою зміну парадигми від верифікації ідентичності на основі "володіння сертифікатом" до "доведення контролю" над секретом. Це є фундаментальним для

концепції самосуверенної ідентичності, де користувач, а не центральний орган, контролює свою ідентичність. Використання ZKP для доведення володіння ключем дозволяє користувачу автентифікуватися на веб-сервісі, не розкриваючи, *який* саме ключ він використовує, або будь-які інші метадані, пов'язані з цим ключем, що значно посилює конфіденційність користувача [22];

- практичні реалізації: хоча ZKP ще не є частиною стандарту TLS, технології активно впроваджуються у суміжних сферах, які згодом вплинуть на веб. Найбільш помітним є їх використання у блокчейн-системах для забезпечення конфіденційності транзакцій. У сфері веб-ідентичності, проекти "Web3" активно використовують zk-SNARKs для "входу через гаманець", де користувач доводить володіння гаманцем, не розкриваючи його повністю. Ці практичні напрацювання слугують полігоном для майбутньої інтеграції у стандартні веб-протоколи. Екосистеми блокчейну та "Web3" стали потужним каталізатором для досліджень ZKP саме тому, що їхня публічна, "бездвірча" природа вимагає потужних рішень для конфіденційності. У публічному реєстрі всі транзакції є видимими за замовчуванням. ZKP (як-от zk-SNARK) є єдиним відомим практичним інструментом, який може перевірити *валідність* транзакції (наприклад, "Користувач А має достатньо коштів, щоб заплатити Користувачу Б"), не розкриваючи *деталей* цієї транзакції (суми, або навіть особистості А та Б). Інтенсивні цикли розробки та оптимізації в цій галузі створюють високоефективні бібліотеки та схеми, які тепер можуть бути "повернені" у традиційний веб для вдосконалення таких протоколів, як TLS.

Варто також відзначити розвиток стандарту WebAuthn [31], який пропонує відмову від паролів на користь апаратних ключів та біометрії. Проте, як зазначають українські дослідники Корченко О.Г. та Іванченко В.О. [30], стандартні реалізації WebAuthn часто розкривають метадані користувача, тоді як підхід на основі ZKP дозволяє досягти повної анонімності сесії.

Огляд літератури показує, що ECC є зрілою основою для конфіденційності, тоді як ZKP пропонує революційний підхід до автентифікації та конфіденційності атрибутів. Причому, найефективніші ZKP самі часто будуються на математичному

апараті еліптичних кривих. Це створює унікальний синергетичний потенціал для їх спільного використання, що є лейтмотивом даної магістерської роботи.

Ця синергія є центральною тезою даного дослідження: наступний значний крок у безпечних веб-комунікаціях буде пов'язаний не з повною заміною еліптичних кривих, а з доповненням їх перевіреної ефективності потужними новими моделями конфіденційності та автентифікації, які надають докази з нульовим розголошенням.

1.5 Висновки до розділу та постановка задач

Отже, в даному розділі було проведено теоретичний огляд галузі, в якій проводиться розробка. У розділі проаналізовано основні принципи побудови захищених сеансових з’єднань, розглянуто особливості застосування еліптичних кривих у сучасній криптографії, а також здійснено аналіз технології доказів з нульовим розголошенням (ZKP) як засобу підвищення приватності та безпеки автентифікації у веб-застосунках.

В результаті проведеного аналізу теоретичного матеріалу та виходячи з мети і актуальності теми, були поставлені наступні задачі подальшої роботи:

– здійснити вдосконалення протоколу сеансового з’єднання задля підвищення стійкості до атак та забезпечення конфіденційності автентифікації шляхом інтеграції механізмів еліптичних кривих та доказів з нульовим розголошенням;

– спроектувати математичну модель та розробити програмний засіб для реалізації удосконаленого протоколу у вигляді клієнт-серверної архітектури веб-застосунку;

– здійснити тестування програмної розробки та вдосконаленого методу на предмет коректності криптографічних перетворень та стійкості до спроб несанкціонованого доступу;

– економічно обґрунтувати розроблювальний удосконалений метод захищеного сеансового з’єднання.

В результаті виконання поставлених завдань, планується досягти

основної мети роботи, а саме підвищити рівень захищеності та конфіденційності сеансового з’єднання у веб-застосунках на основі вдосконаленого методу із застосуванням ZKP та ECC.

2 РОЗРОБКА УДОСКОНАЛЕНОГО ПРОТОКОЛУ НА ОСНОВІ ЕЛІПТИЧНИХ КРИВИХ ТА ДОКАЗІВ З НУЛЬОВИМ РОЗГОЛОШЕННЯМ

Цей розділ присвячено безпосередньо процесу проектування та розробки удосконаленого протоколу захищеного сеансового з’єднання. Основна увага зосереджена на вирішенні фундаментальних обмежень існуючих механізмів автентифікації, зокрема, проблеми надмірної довіри до централізованих посередників та ризиків розкриття ідентифікаційної інформації. У розділі буде представлена концептуальна модель, що інтегрує сучасні криптографічні підходи для побудови більш надійної та конфіденційної системи.

Для досягнення поставленої мети, у розділі буде спершу сформульовано чіткі вимоги до проектованої системи та обґрунтовано її архітектуру. Далі буде розроблено математичну модель, що поєднує криптографію на еліптичних кривих та докази з нульовим розголошенням, яка становить ядро запропонованого механізму автентифікації. На основі цієї моделі буде представлено деталізований, покроковий алгоритм удосконаленого сеансового рукостискання (handshake), а також визначено специфікації та структури даних, необхідні для реалізації.

2.1 Формування вимог та обґрунтування архітектури удосконаленого протоколу

Сучасні протоколи, як-от TLS 1.3, досягли високого рівня безпеки у частині шифрування завдяки режимам автентифікованого шифрування та обміну ключами завдяки ефемерному обміну Діффі-Геллмана. Однак, фундаментальні проблеми залишаються у площині автентифікації. Стандартна модель автентифікації у TLS, що базується на інфраструктурі відкритих ключів та центрах сертифікації, має низку суттєвих недоліків, що створюють вектори для атак:

1. Централізація довіри: вся модель безпеки покладається на довіру до обмеженої кількості кореневих центрів сертифікації. Компрометація одного з них дозволяє зловмиснику випускати легітимні сертифікати для будь-якого домену, що уможливлює повноцінні атаки "людина посередині";
2. Розкриття ідентичності: під час автентифікації (як серверної, так і, особливо,

клієнтської) сторони змушені пред'являти свої сертифікати. Це розкриває метадані та ідентифікаційну інформацію сторонам, які можуть навіть не бути автентифікованими;

3. Вразливість до витоку секретів: у багатьох веб-застосунках автентифікація користувача відбувається після встановлення TLS-тунелю (наприклад, шляхом відправки пари логін/пароль). Це означає, що сервер володіє гешем пароля користувача. У випадку витоку бази даних сервера, ці геші можуть бути атаковані офлайн-перебором [16].

Для подолання цих проблем, удосконалений протокол захищеного сеансового з’єднання, що є предметом даної роботи, має відповідати наступним ключовим вимогам.

Для подолання цих проблем, удосконалений протокол захищеного сеансового з’єднання, що є предметом даної роботи, має відповідати наступним ключовим вимогам. По-перше, він повинен забезпечити децентралізовану автентифікацію, зменшивши або усунувши залежність від довіри до єдиного централізованого органу. По-друге, вимагається конфіденційність автентифікації, що надає можливість стороні довести своє право на доступ або володіння секретом, не розкриваючи сам секрет. По-третє, протокол має демонструвати стійкість до компрометації сервера, щоб навіть повний витік серверних даних (наприклад, гешів паролів) не дозволяв зловмиснику імітувати клієнта. Нарешті, по-четверте, він мусить зберігати високу ефективність, залишаючись обчислювально сумісним з перевагами еліптичних кривих і не додаючи значної кількості обмінів повідомленнями до рукостискання.

Для задоволення сформульованих вимог пропонується гібридна архітектура, яка інтегрує докази з нульовим розголошенням безпосередньо у фазу рукостискання TLS, використовуючи при цьому математичний апарат еліптичних кривих як єдину криптографічну основу.

Запропонована архітектура також враховує сучасні підходи до приватності, реалізовані в протоколах типу Privacy Pass [29], де криптографічні токени використовуються для підтвердження легітимності користувача без його

ідентифікації. Запропонований протокол розширює цю концепцію, додаючи шар взаємної автентифікації.

Вибір криптографії на еліптичних кривих як базису є очевидним – це сучасний стандарт, що забезпечує високу продуктивність при високому рівні безпеки. У роботі буде збережено стандартний механізм обміну ключами (ефемерний Діффі-Геллман) для забезпечення досконалої прямої секретності та шифрування каналу.

Вибір доказів з нульовим розголошенням обґрунтований тим, що ця технологія за своєю природою створена для доведення знання без розголошення. Замість того, щоб клієнт надсилав свій пароль, геш чи сертифікат, він буде доводити, що володіє ними.

Пропонована архітектура не замінює TLS 1.3, а розширює його механізм автентифікації. Загальна ідея полягає у заміні стандартного повідомлення CertificateVerify (де клієнт підписує сеанс своїм приватним ключем) на більш потужне повідомлення ZKProof (Доказ ZKP).

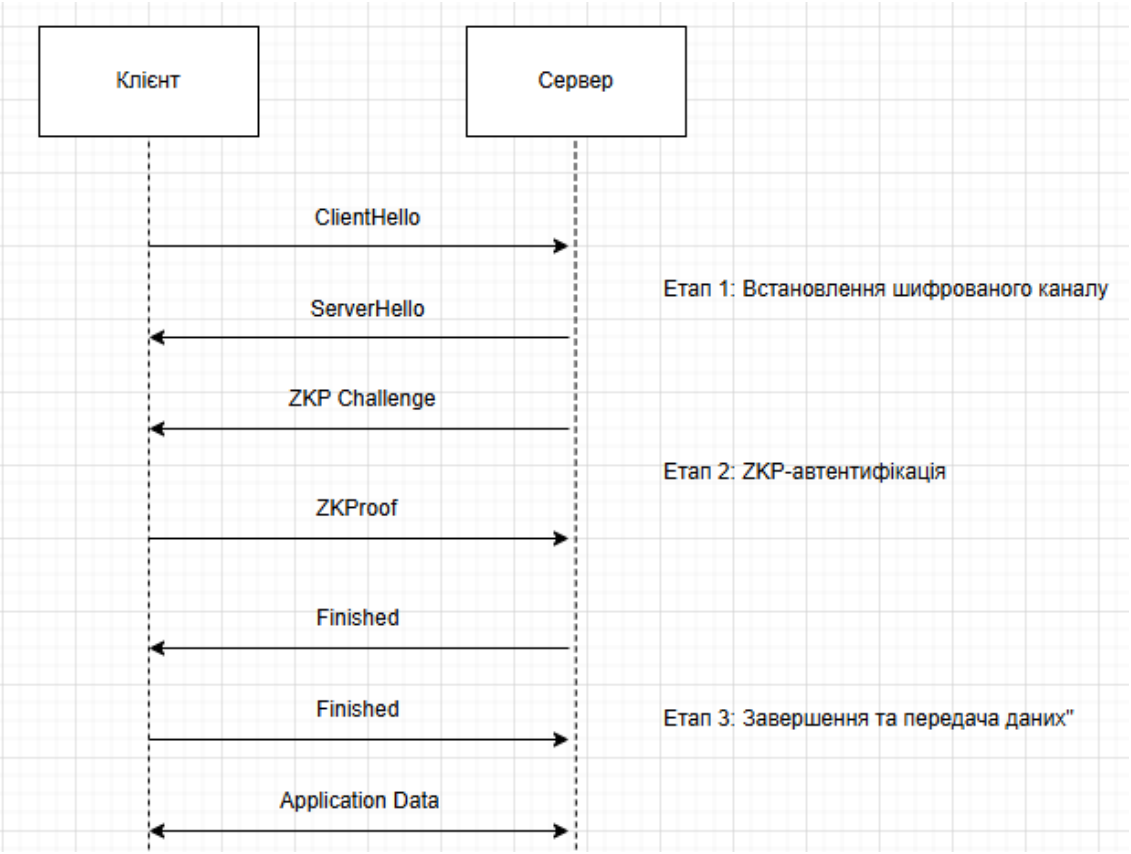


Рисунок 2.1 – Концептуальна архітектура інтеграції доказів з нульовим розголошенням у рукостискання TLS

Архітектура функціонує наступним чином (див. Рис. 2.1):

Етап 1: Встановлення каналу.

На цьому етапі клієнт та сервер виконують стандартний обмін ClientHello та ServerHello. Вони узгоджують набір шифрів та, найголовніше, виконують ефемерний обмін ключами Діффі-Геллмана на еліптичних кривих. Результатом є спільний секретний ключ, який використовується для шифрування подальшого рукостискання.

Етап 2: ZKP-автентифікація.

Це ключове удосконалення. Сервер (у ролі "Того, хто перевіряє") надсилає клієнту виклик. У відповідь клієнт (у ролі "Того, хто доводить") генерує доказ з нульовим розголошенням. Цей доказ підтверджує необхідне твердження, наприклад: "Я володію приватним ключем k, що відповідає публічному ключу P, який відомий серверу". Важливо, що для генерації цього доказу використовуються ті ж самі еліптичні криві, що й для обміну ключами, забезпечуючи синергію та ефективність2. Клієнт надсилає цей доказ у новому повідомленні ZKProof.

Етап 3: Верифікація та завершення.

Сервер отримує ZKProof та виконує операцію верифікації доказу. Верифікація для сучасних схем є надзвичайно швидкою3. Якщо доказ валідний, сервер вважає клієнта автентифікованим і надсилає повідомлення Finished. Після перевірки клієнтом повідомлення Finished, захищений сеанс починається.

Варто окремо підкреслити, що запропонована архітектура являє собою фундаментальний зсув парадигми автентифікації. Класичний підхід TLS, що базується на інфраструктурі відкритих ключів, ґрунтується на «довірі на основі ідентичності»: сервер довіряє клієнту, оскільки третя сторона (центр сертифікації) підтверджує особу цього клієнта [9]. Натомість, запропонована модель впроваджує «довіру на основі доказу»: сервер довіряє клієнту не через те, *ким* він є (що вимагає розкриття сертифіката), а тому, що клієнт може математично *довести* володіння секретом, відомим лише легітимному учаснику [15, 16].

Ця гнучкість дозволяє протоколу бути застосованим у різних сценаріях. У той час як описана модель фокусується на доведенні володіння приватним ключем

(замість CertificateVerify), вона так само може бути адаптована для доведення знання пароля, не передаючи ні сам пароль, ні його геш [16, 21]. Це вирішує проблему вразливості до витоку баз даних, описану раніше, навіть для систем, що покладаються на парольну автентифікацію.

Крім того, синергія між компонентами є ключовою перевагою. Використання еліптичних кривих як єдиної математичної основи і для обміну ключами, і для генерації доказів (оскільки більшість ефективних ZKP-схем самі будуються на еліптичних кривих [20]) дозволяє уникнути впровадження принципово нових, важких криптографічних примітивів. Це забезпечує високу ефективність та елегантність рішення, безпосередньо задовольняючи вимогу щодо збереження високої продуктивності.

Ця архітектура безпосередньо задовольняє поставлені вимоги: автентифікація відбувається без розкриття секрету і не покладається на перевірку ланцюжка довіри під час сеансу. Подальші підрозділи деталізують математичну модель та алгоритм роботи цієї архітектури.

2.2 Розробка математичної моделі автентифікації на основі ZKP та еліптичних кривих

Вимоги, сформульовані у підрозділі 2.1, диктують необхідність відходу від традиційної автентифікації на основі цифрових підписів (як ECDSA) до моделі, що базується на доведенні знання. Метою є доведення володіння секретним ключем, а не використання цього ключа для створення публічного, верифікованого артефакту (підпису).

Проблема класичного підходу (ECDSA). У стандартному рукостисканні TLS з клієнтською автентифікацією, клієнт доводить володіння приватним ключем k, що відповідає його сертифікату (публічному ключу P), шляхом створення цифрового підпису. Він підписує геш сеансових даних:

$$s = \text{Sign}\big(k, \text{hash}(\text{session\_data})\big)$$

Сервер, отримавши підпис s та публічний ключ P (з сертифіката), виконує верифікацію:

$$\text{Verify}(P, \text{hash}(\text{session\_data}), s) = \text{true}$$

Цей підхід, хоч і безпечний, повністю розкриває ідентичність клієнта (його сертифікат) і сам факт автентифікації.

Вибір параметрів криптосистеми базується на класичних підходах прикладної криптології [27], що гарантує стійкість до алгебраїчних атак. Використання стандартизованих кривих дозволяє уникнути вразливостей, притаманних кастомним реалізаціям циклічних груп.

Модель автентифікації на основі ZKP. Запропонована модель замінює процес Sign-Verify на процес Prove-Verify з нульовим розголошенням. Необхідно побудувати схему, де клієнт (Prover) переконує сервер (Verifier) у істинності наступного твердження: "Я володію секретним значенням k (приватний ключ), таким, що для публічно відомої базової точки G еліптичної кривої та мого публічного ключа P, виконується рівність P = kG."

У цій схемі значення k є секретним свідченням (witness), яке відоме лише клієнту. Значення P та G є публічними вхідними даними для перевірки.

Для реалізації цього у веб-протоколі, доказ має бути неінтерактивним (генеруватися за один раунд) та стислим (мати малий розмір). Цим вимогам ідеально відповідають сучасні схеми zk-SNARK [17].

Варто зазначити, що вибір саме zk-SNARK (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge) є свідомим архітектурним рішенням. Хоча існують альтернативні схеми, такі як zk-STARK, які не потребують довіреної початкової установки, вони, як правило, генерують докази значно більшого розміру [18]. У контексті веб-протоколів, де мінімізація затримок та обсягу переданих даних є критичною, "стислість" (Succinctness) SNARK-доказів, що часто вимірюються сотнями байт, є вирішальною перевагою. Це дозволяє інтегрувати доказ у рукостискання без суттєвого впливу на продуктивність сеансу.

Інтеграція ECC та zk-SNARK. Математична модель пропонованого протоколу базується на тому, що самі zk-SNARK схеми для своєї роботи потребують криптографії на еліптичних кривих, зокрема тих, що підтримують ефективні спарювання, наприклад, криві сімейства BN або BLS [20]. Це створює

ідеальну синергію: використаємо ECC як для обміну ключами, так і як математичну основу для генерації доказів ZKP.

Процес автентифікації в рамках цієї моделі складається з трьох фаз:

1. Фаза початкової установки.

Це одноразовий процес, який виконується поза протоколом для генерації криптографічних параметрів. Для математичного твердження (P = kG) генерується пара ключів:

– Ключ доведення: цей ключ є публічним і використовується Prover-ом (клієнтом) для генерації доказів;

– ключ верифікації: цей ключ використовується Verifier-ом (сервером) для перевірки доказів.

Цей етап є необхідним для більшості ефективних схем zk-SNARK [17]. Ключ VK встановлюється на сервері, а PK поширюється серед усіх легітимних клієнтів.

2. Фаза генерації доказу.

Цей етап виконується клієнтом під час рукостискання (Етап 2 в архітектурі 2.1).

Щоб довести своє твердження, клієнт виконує обчислювальну процедуру Prove. На вхід цієї процедури подаються:

– Ключ доведення (PK);

– публічні вхідні дані (публічний ключ P, базова точка G);

– секретне свідчення (приватний ключ k).

$$\pi = \text{Prove}(\text{PK}, (P, G), k)$$

Результатом є доказ π – невеликий за обсягом рядок даних, який не містить жодної інформації про k.

3. Фаза верифікації доказу.

Цей етап виконується сервером (Етап 3 в архітектурі 2.1). Сервер отримує доказ π від клієнта (у повідомленні ZKProof). Для перевірки він виконує процедуру Verify, використовуючи:

– Ключ верифікації (VK);

– публічні вхідні дані (P, G);

– отриманий доказ π.

$$Verify(VK, (P, G), \pi) = true$$

Процедура верифікації є детерміністичною і, що важливо, обчислювально дуже швидкою [17]. Якщо вона повертає true, сервер переконується, що клієнт дійсно володіє секретом k, який відповідає публічному ключу P.

Ця трьохетапна модель (установка, доведення, верифікація) є типовою для сучасних ZKP-систем. Вона вводить асиметрію обчислювальних витрат, яка ідеально підходить для архітектури клієнт-сервер. Фаза 1 (установка) є складною та ресурсомісткою, але вона виконується лише один раз за весь час життя системи. Фаза 2 (генерація доказу), що виконується клієнтом, є обчислювально інтенсивною, але виконується одним клієнтом для одного сеансу. Натомість Фаза 3 (верифікація), що виконується сервером, є надзвичайно швидкою, часто вимагаючи лише кількох операцій спарювання на кривих [20]. Це дозволяє серверу ефективно обробляти тисячі запитів на автентифікацію, перекладаючи основне обчислювальне навантаження з генерації доказу на клієнтські пристрої.

Захист від атак повторного відтворення.

Описана вище базова модель вразлива до атак повторного відтворення: зловмисник може перехопити доказ π і просто відправити його серверу повторно. Щоб запобігти цьому, доказ має бути прив'язаний до конкретного сеансу.

Для цього модифікуємо твердження, включивши до нього виклик (challenge) c, який сервер надсилає клієнту (на Етапі 2). Цей виклик c (наприклад, випадкове число або геш сеансових даних) додається до публічних вхідних даних:

Твердження (Остаточне): "Я володію k, таким що P = kG, і я використовую цей k для підтвердження унікального сеансового виклику c."

Математично це реалізується шляхом включення c у процес генерації та верифікації доказу:

$$\pi = \text{Prove}(\text{PK}, (P, G, c), k)$$

$$Verify(VK, (P, G, c), \pi) = true$$

Тепер доказ π є дійсним лише для конкретного виклику c. Зловмисник, що перехопив π, не зможе використати його у новому сеансі, оскільки сервер згенерує

новий, унікальний c'.

Таким чином, розроблена математична модель досягає двох цілей. По-перше, вона замінює класичну парадигму "Sign-Verify" на "Prove-Verify", що дозволяє приховати ідентичність та секрет клієнта, фундаментально вирішуючи проблему розкриття метаданих. По-друге, шляхом включення унікального сеансового виклику 'c' безпосередньо в математичне твердження, яке доводиться, модель перетворює статичний доказ знання на динамічний протокол автентифікації, стійкий до атак у реальному часі. Це забезпечує надійний криптографічний зв'язок між конкретним сеансом та доказом володіння секретом.

Ця математична модель повністю задовольняє поставлені вимоги: вона використовує ECC як основу, доводить володіння секретом k без його розкриття і є стійкою до витоку даних з сервера, оскільки сервер зберігає лише ключ верифікації VK, який неможливо використати для генерації нових доказів.

Завершений опис математичного апарату слугує прямим теоретичним обґрунтуванням для наступного етапу проектування. Маючи формалізоване твердження та визначені фази генерації та верифікації доказу, далі можна перейти до розробки покрокового алгоритму, який опише, як саме ці математичні операції інкапсулюються у повідомлення сеансового протоколу. Цей алгоритм, який буде представлено у наступному підрозділі, деталізує логіку скінченного автомата клієнта та сервера під час фази рукостискання.

2.3 Розробка алгоритму удосконаленого сеансового рукостискання

На основі розробленої архітектури та математичної моделі необхідно формалізувати алгоритм удосконаленого сеансового рукостискання. Алгоритм описує точну послідовність обміну повідомленнями між клієнтом ("Той, хто доводить") та сервером ("Той, хто перевіряє") для встановлення захищеного та автентифікованого сеансу.

Пропонований алгоритм модифікує стандартний потік TLS 1.3, замінюючи класичну автентифікацію на основі сертифікатів на автентифікацію за допомогою доказів з нульовим розголошенням. Загальну послідовність обміну

повідомленнями проілюстровано на діаграмі (рис. 2.2).

Як видно з діаграми (рис. 2.2), запропонований алгоритм зберігає базову структуру рукостискання TLS 1.3, зокрема фазу обміну ключами ECDHE. Це критично важливо для збереження досконалої прямої секретності та забезпечення шифрування каналу *перед* початком автентифікації. Ключова модифікація полягає у виключенні повідомлень Certificate та CertificateVerify з боку клієнта і заміні їх на новий двокроковий обмін "виклик-доказ" (ZKPChallenge та ZKProof). Такий підхід дозволяє "вбудувати" ZKP-автентифікацію у зашифровану частину рукостискання, що захищає сам процес доведення від пасивного прослуховування.

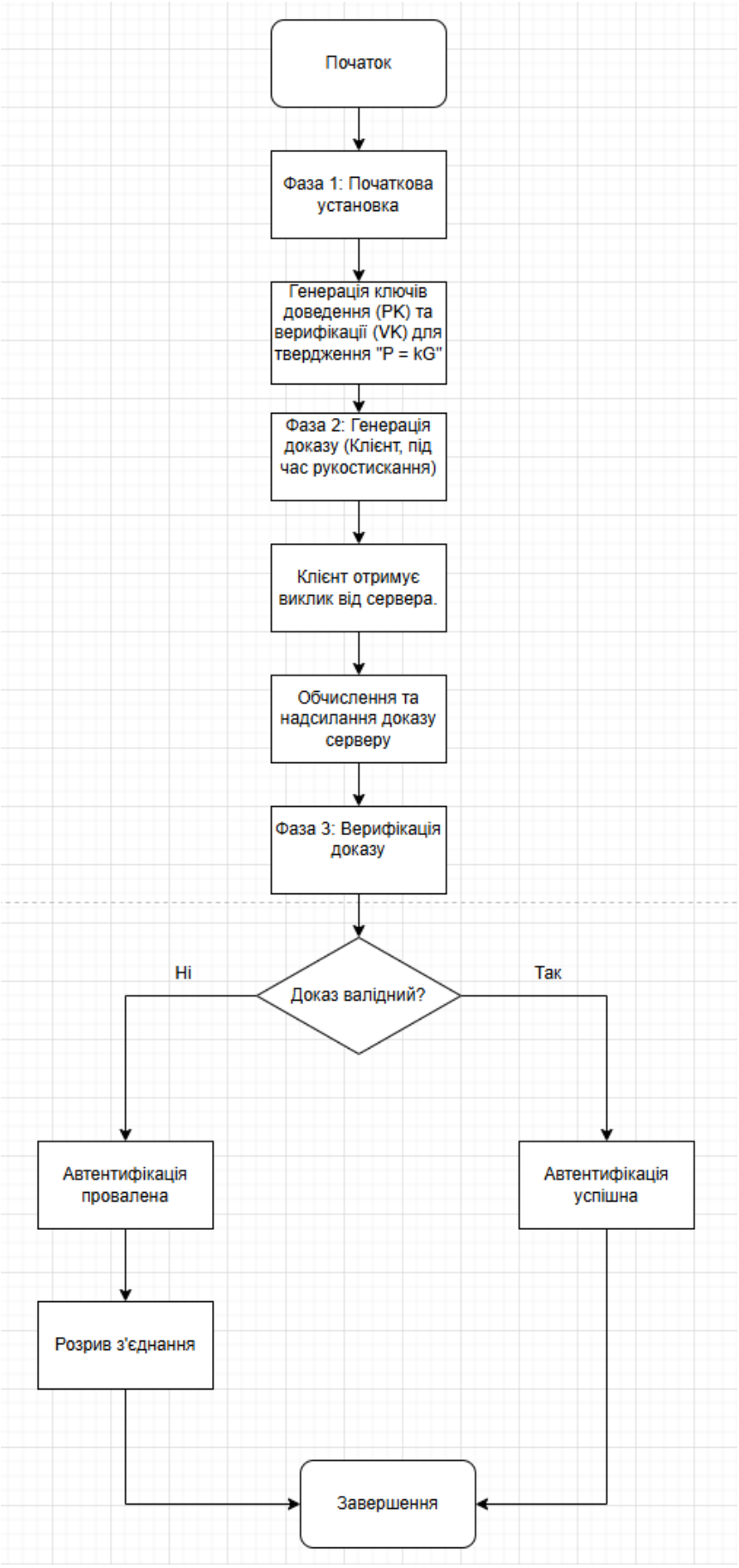


Рисунок 2.2 – Діаграма послідовності удосконаленого рукостискання з ZKP-автентифікацією

Процес рукостискання починається з ініціації клієнта. Клієнт надсилає повідомлення ClientHello, яке, окрім стандартних параметрів, містить нове розширення zkp_authentication. Це розширення сигналізує серверу про готовність клієнта пройти автентифікацію за допомогою ZKP та передає його публічний ідентифікатор (наприклад, публічний ключ P), за яким сервер може знайти відповідний ключ верифікації.

Використання механізму розширень є фундаментальним для сумісності протоколу [4]. Якщо сервер не підтримує запропоноване розширення, він просто проігнорує його, і рукостискання може продовжитись за стандартним сценарієм (наприклад, з класичною автентифікацією або без автентифікації клієнта). Це забезпечує гнучкість впровадження та зворотну сумісність з існуючою інфраструктурою.

Сервер, отримавши ClientHello і підтвердивши підтримку ZKP-автентифікації, відповідає повідомленням ServerHello. На цьому етапі обираються спільні криптографічні параметри та відбувається обмін ключами ECDHE. В результаті клієнт і сервер обчислюють спільний секретний ключ session_key, і всі подальші повідомлення шифруються. Одразу після ServerHello, сервер надсилає зашифроване повідомлення ZKPChallenge, що містить унікальний для сеансу криптографічний виклик c, необхідний для запобігання атакам повторного відтворення.

Надсилання ZKPChallenge *після* завершення ECDHE та встановлення шифрованих ключів сеансу є свідомим проектним рішенням. Це гарантує, що сам виклик c є конфіденційним і не може бути перехоплений. Прив'язка доказу до секретного, унікального виклику, який неможливо передбачити, є надійним механізмом захисту від атак повторного відтворення, як було детально обґрунтовано у математичній моделі (2.2).

Отримавши виклик c, клієнт переходить до генерації доказу. Він має всі необхідні компоненти: свій приватний ключ k, публічні параметри (включно з c) та ключ доведення PK. Клієнт виконує математичну операцію та надсилає отриманий доказ π серверу у новому зашифрованому повідомленні ZKProof.

Ця операція є найбільш обчислювально інтенсивною для клієнта, що відповідає архітектурному рішенню, закладеному в 2.1, щодо перенесення основного навантаження з сервера на клієнтську сторону.

Настає етап верифікації доказу сервером. Сервер отримує ZKProof і виконує операцію. У разі успішної верифікації (true), сервер вважає клієнта автентифікованим і надсилає своє повідомлення Finished. Якщо ж верифікація провалюється, сервер негайно розриває з'єднання. Останнім кроком є завершення клієнтом: він отримує та перевіряє Finished від сервера, після чого надсилає своє повідомлення Finished. На цьому рукостискання завершується, і сторони розпочинають безпечну передачу даних.

Таким чином, формалізований алгоритм описує повний, замкнений цикл взаємодії, що перетворює теоретичну математичну модель на чітку послідовність дій, готову до подальшої специфікації та програмної реалізації. Він чітко визначає точки інтеграції (розширення ClientHello) та нові стани протоколу (очікування ZKProof після ZKPChallenge), формуючи завершену проектну модель.

2.4 Аналіз безпеки запропонованої моделі протоколу

Після розробки архітектури, математичної моделі та алгоритму удосконаленого протоколу, критично важливим етапом є проведення аналізу його безпеки. Цей аналіз має на меті продемонструвати, наскільки ефективно запропонована модель (інтеграція ZKP та ECC) задовольняє вимоги, сформульовані у підрозділі 2.1, та яким чином вона протистоїть ключовим векторам атак, притаманним класичним протоколам автентифікації.

Найбільш суттєва перевага запропонованого протоколу полягає у фундаментальній зміні моделі автентифікації. У традиційній схемі з паролем сервер змушений зберігати секрет користувача, хоча й у гешованому вигляді. Як зазначалося, витік цієї бази даних дозволяє зловмиснику проводити офлайн-атаки перебору [16]. У запропонованій моделі, що базується на ZKP, сервер ніколи не отримує і не зберігає секрет клієнта k. Натомість сервер зберігає лише ключ верифікації.

Відповідно до фундаментальних властивостей неінтерактивних стислих доказів (zk-SNARK), ключ верифікації дозволяє лише перевіряти існуючі докази, але його абсолютно недостатньо для генерації нових доказів [17]. Таким чином, навіть у випадку повної компрометації сервера та витоку всієї його бази даних (включно з усіма ключами верифікації, зловмисник не зможе імітувати клієнта, оскільки він не володіє секретним ключем k.

Це принципово відрізняється від класичних систем, де захист бази даних з гешами паролів є критичною, але часто недостатньою лінією оборони. У запропонованій моделі, навіть якщо зловмисник отримує повний контроль над сервером, він отримує лише набір ключів верифікації, які є марними без відповідних секретних ключів клієнтів. Це дозволяє реалізувати архітектуру, де серверу *за замовчуванням* не потрібно довіряти (Zero Trust), оскільки він більше не є сховищем навіть опосередкованих секретів користувача.

Вимога конфіденційності автентифікації задовольняється за самою природою доказів з нульовим розголошенням. Клієнт доводить володіння секретом k, але в процесі обміну надсилання доказу π не розкриває жодного біта інформації про сам k [15].

Класична атака "людина посередині" (MitM) на TLS значною мірою покладається на обман клієнта шляхом пред'явлення підробленого, але валідного (наприклад, випущеного скомпрометованим CA) сертифіката. У запропонованій моделі автентифікація клієнта не залежить від перевірки ланцюжка довіри CA. Натомість, вона базується на математичному доведенні володіння конкретним секретом k, що відповідає публічному ідентифікатору P. Довіра зміщується від зовнішніх посередників (CA) до криптографічної стійкості самої ZKP-схеми.

Цей зсув від "довіри на основі ідентичності" (де третя сторона, підтверджує особу) до "довіри на основі доказу" (де клієнт математично доводить володіння секретом) також має значні переваги для конфіденційності. Клієнту більше не потрібно розкривати свій сертифікат, який може містити чутливу ідентифікаційну інформацію, просто для того, щоб розпочати сеанс. Це відкриває шлях до частково анонімних, але криптографічно підтверджених сеансів.

Навіть якщо зловмисник (MitM) перехопить з'єднання, він не зможе автентифікувати себе перед сервером як легітимний клієнт, оскільки не зможе згенерувати валідний доказ π без знання k. Аналогічно, він не зможе видати себе за легітимний сервер перед клієнтом (за умови, що серверна автентифікація також використовує ZKP, або клієнт має надійний спосіб отримання ключа верифікації сервера).

Як було зазначено в математичній моделі та алгоритмі, базова ZKP-схема вразлива до атак повторного відтворення, коли зловмисник просто записує валідний доказ π і відтворює його у новій сесії. У запропонованому протоколі ця проблема вирішується шляхом прив'язки доказу до сеансу.

Сервер генерує та надсилає клієнту унікальний, криптографічно стійкий виклик c (у повідомленні ZKPChallenge). Цей виклик включається до публічних вхідних даних під час генерації доказу: $\pi = \text{Prove}(\text{PK}, (P, G, c), k)$. Оскільки c є унікальним для кожного рукостискання, будь-який перехоплений доказ π буде недійсним для будь-якого іншого сеансу, оскільки він не пройде верифікацію з новим викликом c'.

Цей механізм прив'язки доказу до сеансових даних (через виклик c) є фундаментальним для безпеки інтерактивних протоколів автентифікації. Він гарантує "свіжість" доказу, унеможливлюючи для зловмисника використання валідних доказів, перехоплених у минулих сесіях.

Порівнюючи запропонований підхід зі стандартною клієнтською автентифікацією на основі ECDSA, необхідно розглянути обчислювальні витрати.

– Клієн: генерація zk-SNARK доказу (Prove) є обчислювально значно складнішою операцією, ніж генерація ECDSA-підпису. Це є основним компромісом запропонованої моделі [17];

– сервер: верифікація zk-SNARK доказу (Verify) є надзвичайно швидкою, часто навіть швидшою, ніж верифікація ECDSA-підпису, і незрівнянно швидшою, ніж повна валідація ланцюжка сертифікатів X.509 [17].

Такий асиметричний профіль обчислювальних витрат є стратегічно вигідним для сучасних веб-застосунків. Обчислювальні потужності клієнтських пристроїв

(персональних комп'ютерів, смартфонів) постійно зростають і, як правило, присвячені роботі одного користувача. Водночас сервери змушені обробляти тисячі або десятки тисяч одночасних підключень. Перенесення складної, але одноразової операції генерації доказу на сторону клієнта дозволяє значно розвантажити сервер, залишивши йому лише швидку операцію верифікації. Це підвищує загальну масштабованість та стійкість системи до атак типу "відмова в обслуговуванні" (DoS), спрямованих на виснаження ресурсів сервера під час рукостискання.

Враховуючи, що сервери веб-застосунків зазвичай обробляють тисячі одночасних з'єднань, перенесення основного обчислювального навантаження з сервера (верифікація) на клієнта (генерація доказу) є виправданим компромісом. Це дозволяє серверу швидко автентифікувати клієнтів, не витрачаючи ресурси на складні операції з сертифікатами. Кількість обмінів повідомленнями (RTT) залишається такою ж, як у стандартному TLS 1.3 з автентифікацією клієнта (1-RTT). Таким чином, модель задовольняє вимогу в частині збереження ефективності сеансу.

2.5 Висновки до розділу

У даному розділі було детально розглянуто та спроектовано удосконалений протокол захищеного сеансового з’єднання. Запропоноване рішення спрямоване на вирішення ключових, фундаментальних проблем конфіденційності та надмірної централізації довіри, які притаманні існуючим, широко розповсюдженим механізмам автентифікації у веб-застосунках.

Відштовхуючись від глибокого аналізу недоліків стандартної інфраструктури відкритих ключів, проведеного в попередньому розділі, було сформульовано чіткий набір проектних вимог до нового протоколу. Основними вимогами були визначені: необхідність децентралізованої або непрямої автентифікації, що усуває залежність від єдиної точки довіри; забезпечення повної конфіденційності процесу автентифікації за принципом нульового розголошення; гарантування стійкості до компрометації серверної сторони; та збереження високої

обчислювальної ефективності, сумісної з вимогами сучасних веб-сервісів.

Для повного задоволення цього набору вимог було обґрунтовано та розроблено інноваційну гібридну архітектуру. Вона інтегрує криптографічний апарат доказів з нульовим розголошенням безпосередньо у стандартний потік рукостискання TLS 1.3. При цьому, для збереження ефективності, було вирішено зберегти еліптичні криві (ECC) як єдиний математичний базис, що забезпечує глибоку синергію компонентів. Було розроблено деталізовану математичну модель ZKP-автентифікації, яка формалізує процес, при якому клієнт ("Той, хто доводить") може математично довести володіння приватним ключем, не розкриваючи при цьому сам ключ.

На основі розробленої математичної моделі було формалізовано покроковий алгоритм удосконаленого сеансового рукостискання. Цей алгоритм чітко описує нову послідовність обміну повідомленнями та визначає логіку скінченного автомата для клієнта і сервера, включаючи обробку розширення zkp_authentication у ClientHello та нових повідомлень ZKPChallenge і ZKProof. На завершення теоретичного проектування було проведено всебічний аналіз безпеки запропонованої моделі. Цей аналіз довів, що розроблений протокол успішно протистоїть ключовим векторам атак, таким як атаки "людина посередині" (MitM), атаки повторного відтворення, а також фундаментально вирішує проблему ризиків, пов'язаних з витоком автентифікаційних даних із серверної сторони.

Таким чином, у даному другому розділі було представлено повний та логічно завершений цикл проектування удосконаленого протоколу. Цей цикл охопив усі етапи: від глибокого аналізу проблем та постановки чітких вимог до розробки високорівневої архітектури, детальної математичної моделі, покрокового алгоритму функціонування та ґрунтовного аналізу безпеки. Виконана робота створює необхідне і достатнє теоретичне та проектне підґрунтя для переходу до наступного етапу магістерської кваліфікаційної роботи – безпосередньої програмної реалізації та експериментального дослідження продуктивності й ефективності запропонованого протоколу, що і буде детально розглянуто у третьому розділі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОТОКОЛУ ЗАХИЩЕНОГО СЕАНСОВОГО З'ЄДНАННЯ У ВЕБ-ЗАСТОСУНКАХ НА ОСНОВІ ЕЛІПТИЧНИХ КРИВИХ ТА ДОКАЗІВ З НУЛЬОВИМ РОЗГОЛОШЕННЯМ

Третій розділ магістерської кваліфікаційної роботи присвячений практичному втіленню теоретичних та проектних рішень, які були детально обґрунтовані та розроблені у попередніх частинах дослідження. Після проведення глибокого аналізу існуючих вразливостей сучасних протоколів автентифікації та формування математичної моделі, що поєднує криптографію на еліптичних кривих із технологією доказів з нульовим розголошенням, виникає нагальна необхідність у створенні діючого програмного прототипу. Цей етап є критично важливим, оскільки саме програмна реалізація дозволяє перевірити коректність роботи розроблених алгоритмів у реальному середовищі функціонування веб-застосунків, оцінити їхню обчислювальну складність та підтвердити можливість інтеграції запропонованого підходу в сучасні інформаційні системи.

У даному розділі послідовно викладається процес створення програмного комплексу, починаючи від вибору та обґрунтування інструментальних засобів розробки і закінчуючи безпосереднім кодуванням ключових модулів системи. Основна увага приділяється питанням архітектурної побудови веб-застосунку, який виступає середовищем для емуляції клієнт-серверної взаємодії, а також деталям імплементації криптографічних примітивів. Особливий акцент зроблено на забезпеченні високої продуктивності обчислень, що є критичним фактором для протоколів сеансового з'єднання, які працюють у режимі реального часу. Також у розділі наводяться результати тестування розробленої системи, що демонструють її стійкість до типових векторів атак та відповідність сформульованим вимогам безпеки. Реалізація базується на використанні найсучасніших технологій програмування, що забезпечує актуальність та довгострокову підтримку запропонованого рішення.

3.1 Обґрунтування вибору мови програмування, середовища розробки та фреймворків

Успішна реалізація складних криптографічних протоколів, призначених для захисту інформації у веб-середовищі, значною мірою залежить від правильного вибору технологічного стеку. Цей вибір має базуватися на комплексному аналізі вимог до швидкодії, безпеки, надійності та масштабованості системи. Враховуючи специфіку теми дослідження, яка передбачає виконання інтенсивних математичних операцій над великими числами в рамках груп точок еліптичної кривої, а також необхідність інтеграції цих обчислень у веб-інфраструктуру, вибір інструментарію стає фундаментом для всього подальшого розроблення.

В якості основної мови програмування для реалізації серверної частини протоколу та криптографічного ядра системи було обрано мову C#. Цей вибір обумовлений низкою вагомих факторів, серед яких ключове місце займає сувора статична типізація мови. На відміну від динамічно типізованих мов, C# дозволяє виявляти переважну більшість потенційних помилок ще на етапі компіляції коду, що є критично важливим для розробки систем безпеки, де будь-яка програмна помилка може призвести до появи вразливостей. Крім того, C# є об'єктно-орієнтованою мовою, що дозволяє будувати чітку, модульну архітектуру застосунку, інкапсулюючи складну математичну логіку в окремі класи та сервіси, що значно спрощує підтримку та розширення програмного коду в майбутньому. Також варто відзначити високу ефективність управління пам'яттю в C#, що забезпечується вбудованим механізмом збирання сміття, який мінімізує ризики витоку пам'яті та підвищує стабільність роботи серверних застосунків під навантаженням.

Фундаментальною платформою для виконання розробленого програмного забезпечення було обрано новітню версію .NET 9. Вибір саме дев'ятої версії платформи є стратегічно важливим рішенням, продиктованим необхідністю забезпечення максимальної продуктивності криптографічних обчислень. Платформа .NET 9, випущена напередодні запланованого

впровадження системи, містить значні оптимізації JIT-компілятора та вдосконалені алгоритми роботи з пам'яттю, що дозволяє досягти суттєвого приросту швидкодії порівняно з попередніми версіями. Для задач, пов'язаних з реалізацією доказів з нульовим розголошенням, критично важливою є наявність у платформі оптимізованих структур даних для роботи з великою арифметикою, зокрема типу BigInteger, який у .NET 9 отримав додаткові покращення продуктивності. Це дозволяє виконувати операції множення точок на еліптичній кривій та модульну арифметику з високою швидкістю, що безпосередньо впливає на час встановлення захищеного з'єднання та загальний досвід користувача. Крім того, .NET 9 орієнтована на побудову хмарних рішень (cloud-native), що спрощує подальше розгортання розробленого протоколу в сучасних хмарних середовищах.

Для створення веб-інтерфейсу та реалізації логіки взаємодії по протоколу HTTP було обрано фреймворк ASP.NET Core. Цей фреймворк є невід'ємною частиною екосистеми .NET і надає потужні засоби для побудови високопродуктивних веб-застосунків та API. Використання архітектурного патерну MVC (Model-View-Controller), який є нативним для ASP.NET Core, дозволяє чітко розділити логіку обробки запитів, представлення даних та управління даними, що робить структуру проекту прозорою та зрозумілою. Важливою перевагою ASP.NET Core є його кросплатформеність, що дає змогу розгортати серверну частину протоколу не лише на операційних системах Windows, але й на Linux та macOS, що значно розширює сферу потенційного застосування розробки. Також фреймворк має вбудовану систему ін'єкції залежностей (Dependency Injection), що спрощує інтеграцію криптографічних сервісів у контролери та полегшує процес модульного тестування окремих компонентів системи.

В якості інтегрованого середовища розробки (IDE) використовується Microsoft Visual Studio 2022. Цей програмний продукт є галузевим стандартом для розробки на платформі .NET і надає розробнику вичерпний набір інструментів для написання, налагодження та тестування коду. Наявність

потужного відлагоджувача дозволяє покроково відстежувати виконання алгоритмів генерації та перевірки доказів, аналізувати стан змінних у пам'яті та швидко виявляти логічні помилки, що є незамінним при роботі зі складною математичною логікою. Крім того, Visual Studio 2022 забезпечує повну підтримку нових можливостей синтаксису C# та платформи .NET 9, а також має вбудовані засоби для профілювання продуктивності, що дозволяє оптимізувати "вузькі місця" в реалізації протоколу.

Критично важливим аспектом реалізації клієнтської частини стало використання нативного типу BigInt, який був стандартизований у новітніх специфікаціях ECMAScript. Згідно з документацією MDN [34], цей тип забезпечує виконання арифметичних операцій з цілими числами довільної точності без втрати продуктивності, що є необхідною умовою для реалізації скалярного множення точок еліптичної кривої безпосередньо у браузері.

Для реалізації клієнтської сторони, яка виконує роль "Доказувача" (Prover) у схемі протоколу, було вирішено використати стандартні веб-технології HTML5 та JavaScript. Такий підхід забезпечує універсальність рішення, оскільки дозволяє клієнту проходити процедуру автентифікації через будь-який сучасний веб-браузер без необхідності встановлення додаткового програмного забезпечення чи плагінів. Використання JavaScript на стороні клієнта дозволяє виконувати необхідні криптографічні операції, такі як генерація ключів та обчислення відповіді на виклик сервера, безпосередньо на пристрої користувача, що відповідає концепції розподілених обчислень та знижує навантаження на центральний сервер.

Підсумовуючи вищевикладене, можна стверджувати, що обраний технологічний стек, який включає мову C#, платформу .NET 9, фреймворк ASP.NET Core та середовище Visual Studio 2022, є оптимальним для вирішення поставлених завдань. Він забезпечує необхідний баланс між високою продуктивністю, безпекою коду, зручністю розробки та універсальністю застосування, що є запорукою успішної програмної реалізації удосконаленого протоколу захищеного сеансового з'єднання.

3.2 Програмна реалізація системи розмежування доступу

Процес програмної реалізації спроєктованого протоколу розпочався з розробки серверної архітектури, яка має забезпечувати надійну обробку криптографічних запитів та підтримку сесійного стану для клієнтів. Враховуючи високі вимоги до масштабованості та безпеки, було вирішено застосувати принципи багатошарової архітектури (Layered Architecture), що дозволяє ефективно ізолювати криптографічне ядро від логіки обробки HTTP-запитів.

Фундаментальним будівельним блоком системи виступає сервісний шар, який відповідає за математичну коректність виконання операцій на еліптичних кривих. Оскільки стандартні примітиви мови C# не здатні вмістити числа, що використовуються в сучасній криптографії, для реалізації арифметики в скінченних полях було використано структуру System.Numerics.BigInteger, яка забезпечує необхідну точність обчислень без втрати даних.

Важливим аспектом реалізації є управління станом сесії. На відміну від класичної передачі пароля, ZKP-автентифікація є інтерактивним процесом, що відбувається в кілька етапів, тому сервер повинен зберігати контекст проміжного стану для кожного користувача. Для програмної репрезентації цього стану було розроблено клас ZkpSession, який виступає контейнером даних.

Він містить унікальний ідентифікатор сесії, публічний ключ користувача, згенерований сервером виклик (Challenge) та часові мітки для контролю "часу життя" сесії, що дозволяє запобігти атакам із використанням застарілих даних:

```
public class ZkpSession
{
   public string SessionId { get; set; } = Guid.NewGuid().ToString();
   public string Challenge { get; set; } = string.Empty;
   public string Commitment { get; set; } = string.Empty;
}
```

Центральним елементом логіки є сервіс ZkpCryptographyService. Цей клас інкапсулює всю математику еліптичних кривих та роботу з великими числами. У конструкторі класу відбувається ініціалізація порядку еліптичної кривої (Curve Order), що є константою для обраного стандарту криптографії і використовується для всіх модульних операцій.

Особливу увагу при розробці було приділено методу генерації криптографічного виклику GenerateSecureChallenge. Передбачуваність цього параметра могла б повністю скомпрометувати безпеку протоколу, дозволивши зловмиснику підробити доказ. Тому реалізація використовує криптографічно стійкий генератор випадкових чисел RandomNumberGenerator, а отримане значення приводиться до діапазонну порядку кривої:

```
public string GenerateSecureChallenge()
{
   var bytes = RandomNumberGenerator.GetBytes(32);
   var challenge = new BigInteger(bytes, isUnsigned: true, isBigEndian: true);
   challenge %= _curveOrder;
   return challenge.ToString("x");
}
```

Ключовим методом, що відповідає за перевірку автентичності користувача, є функція VerifySchnorrProof. Вона імплементує математичне рівняння верифікації схеми Шнорра. Метод приймає публічні параметри (ключ, зобов'язання, виклик) та отриману від клієнта відповідь s.

У тілі методу відбувається парсинг шістнадцяткових рядків у об'єкти BigInteger та перевірка граничних умов (числа не повинні дорівнювати нулю або перевищувати порядок кривої), що є важливим заходом проти атак на реалізацію протоколу.

Наступним етапом розробки стала реалізація контролера API AccountController, який виступає точкою входу для зовнішніх запитів. Контролер спроєктовано з урахуванням REST-принципів та асинхронної моделі обробки запитів.

Критичним технічним викликом була необхідність забезпечення

коректної роботи в багатопотоковому середовищі веб-сервера. Оскільки ASP.NET Core обробляє запити паралельно, використання звичайних колекцій могло б призвести до станів гонитви. Для вирішення цієї проблеми було застосовано потокобезпечну колекцію ConcurrentDictionary, яка гарантує цілісність даних при одночасному доступі та емулює роботу розподіленого сховища сесій.

Метод ClientHello ініціює процедуру рукостискання. Він приймає публічний ключ та зобов'язання клієнта, створює нову сесію, генерує для неї унікальний виклик за допомогою описаного вище сервісу та зберігає цей стан у пам'яті сервера:

```
[HttpPost("ClientHello")]
public IActionResult ClientHello([FromBody] ClientHelloRequest request)
{
   var session = new ZkpSession
   {
      PublicKey = request.PublicKey,
      Commitment = request.Commitment,
      Challenge = _zkpService.GenerateSecureChallenge()
   };
   _sessionStore.TryAdd(session.SessionId, session);
   return Ok(new { sessionId = session.SessionId, challenge = session.Challenge });
}
```

Фінальна верифікація відбувається у методі Verify. Цей метод отримує обчислену клієнтом відповідь s, знаходить відповідну сесію в пам'яті та делегує перевірку криптографічному сервісу. Важливо відзначити, що метод також перевіряє час життя сесії (ExpiresAt), що додає додатковий шар захисту. У разі успішної верифікації сесія позначається як автентифікована, що відкриває доступ до захищених ресурсів.

Демонстрацією практичного застосування розробленого механізму є захищений ендпоінт GetSecretData. Доступ до цього методу надається виключно за умови наявності ідентифікатора сесії у списку автентифікованих користувачів _authenticatedSessions. Це реалізує принцип розмежування

доступу, де дані надаються лише після успішного криптографічного доведення знання секрету:

```
[HttpGet("SecretData")]
public IActionResult GetSecretData(string sessionId)
{
  if (_authenticatedSessions.ContainsKey(sessionId))
  {
    return Ok(new { secret = "CONFIDENTIAL...", ... });
  }
  return StatusCode(403, "Access Denied");
}
```

Невід'ємною та критично важливою складовою розробленої системи є клієнтський інтерфейс, який у термінології протоколів з нульовим розголошенням виконує роль сутності «Доказувач» (Prover). Реалізація цієї частини вимагала вирішення нетривіального інженерного завдання — забезпечення виконання ресурсомістких математичних операцій безпосередньо у середовищі веб-браузера користувача, не покладаючись на сторонні плагіни чи встановлене програмне забезпечення.

Для досягнення цієї мети було вирішено використати нативні можливості сучасного стандарту ECMAScript 2020. Фундаментальну роль тут відіграє вбудований тип даних BigInt, який дозволяє оперувати цілими числами довільної точності. Це технологічне рішення є ключовим, оскільки криптографія на еліптичних кривих оперує числами порядку 2^{256}, що значно перевищує можливості стандартного числового типу Number у JavaScript (який, по суті, є числом із плаваючою комою подвійної точності).

Клієнтський інтерфейс було спроєктовано не просто як засіб введення даних, а як інтерактивну візуалізацію внутрішнього стану протоколу. Логіка відображення побудована таким чином, щоб демістифікувати процес ZKP-автентифікації для кінцевого користувача. Інтерфейс покроково відображає етапи протоколу:

– Генерація ключів: користувач бачить момент створення пари ключів, при цьому система чітко сигналізує, що приватний ключ зберігається

виключно у локальній пам’яті браузера і не передається мережею;

– обчислення зобов’язання: візуалізується процес генерації ефемерного числа (nonce) та відповідної точки на кривій;

– отримання виклику:відображається отримання випадкового числа від сервера, що підтверджує існування сесії;

– формування доказу: Фінальний етап, на якому обчислюється скалярна відповідь

Такий підхід до проєктування інтерфейсу (UI/UX) виконує важливу безпекову функцію: він наочно демонструє, що секретний ключ користувача ніколи не залишає межі довіреного пристрою, що є фундаментальною перевагою технології нульового розголошення перед класичними парольними методами.

Окрім штатного режиму роботи, у клієнтський код було інтегровано спеціалізований механізм для тестування безпеки — модуль симуляції атаки. Ця функціональність дозволяє в реальному часі перевірити реакцію сервера на спробу підробки криптографічного доказу. При активації цього режиму клієнт генерує завідомо некоректне значення відповіді s, імітуючи дії зловмисника, який намагається вгадати параметри автентифікації без знання приватного ключа. Це дозволяє наочно підтвердити властивість «обґрунтованості» (soundness) протоколу.

Реалізація клієнтської логіки базується на асинхронній моделі обробки подій (async/await), що забезпечує високу чуйність інтерфейсу навіть під час виконання складних математичних розрахунків. Взаємодія з сервером відбувається через стандартний API fetch, що робить клієнтську частину легкою та сумісною з будь-яким сучасним веб-браузером (Chrome, Firefox, Edge, Safari) без необхідності адаптації коду.

Підсумовуючи вищевикладене, можна стверджувати, що розроблений програмний комплекс являє собою завершену, повнофункціональну екосистему. Вона реалізує повний цикл удосконаленого протоколу автентифікації — від низькорівневої математики еліптичних кривих до

високорівневого інтерфейсу користувача.

Таким чином, розроблений програмний комплекс реалізує повний цикл удосконаленого протоколу автентифікації. Використання асинхронної моделі обробки запитів на клієнті та сервері забезпечує високу чуйність інтерфейсу, а застосування криптографічно стійких генераторів випадкових чисел гарантує безпеку процесу генерації викликів. Розроблене рішення є готовим до інтеграції у сучасні веб-застосунки для підвищення рівня захищеності сеансових з'єднань.

Розроблена система відповідає вимогам національного стандарту ДСТУ EN ISO/IEC 27001:2023 [35] у частині контролю доступу та криптографічного захисту інформації, що дозволяє рекомендувати її до впровадження в корпоративних інформаційних системах.

3.3 Тестування програмної реалізації протоколу

Після завершення етапу кодування та розгортання веб-застосунку було проведено комплексне тестування розробленої системи. Метою випробувань була перевірка коректності виконання криптографічних операцій на еліптичних кривих, а також підтвердження здатності протоколу протидіяти спробам несанкціонованого доступу (властивість обґрунтованості). Тестування проводилося за сценарієм повного циклу автентифікації, що складається з чотирьох послідовних етапів.

Етап 1. Налаштування та генерація ключів

Робота з системою розпочалася із завантаження головного вікна веб-застосунку. У початковому стані інтерфейс відображає статус «ВІДКЛЮЧЕНО», а система очікує на дії користувача для ініціалізації криптографічних параметрів еліптичної кривої NIST P-256.

Користувач натиснув кнопку «Генерувати ключі», що ініціювало створення пари ключів на стороні клієнта. У системному лозі було зафіксовано створення публічного ключа (P), тоді як приватний ключ (x) залишився у

захищеній пам’яті браузера, що відповідає вимогам безпеки.

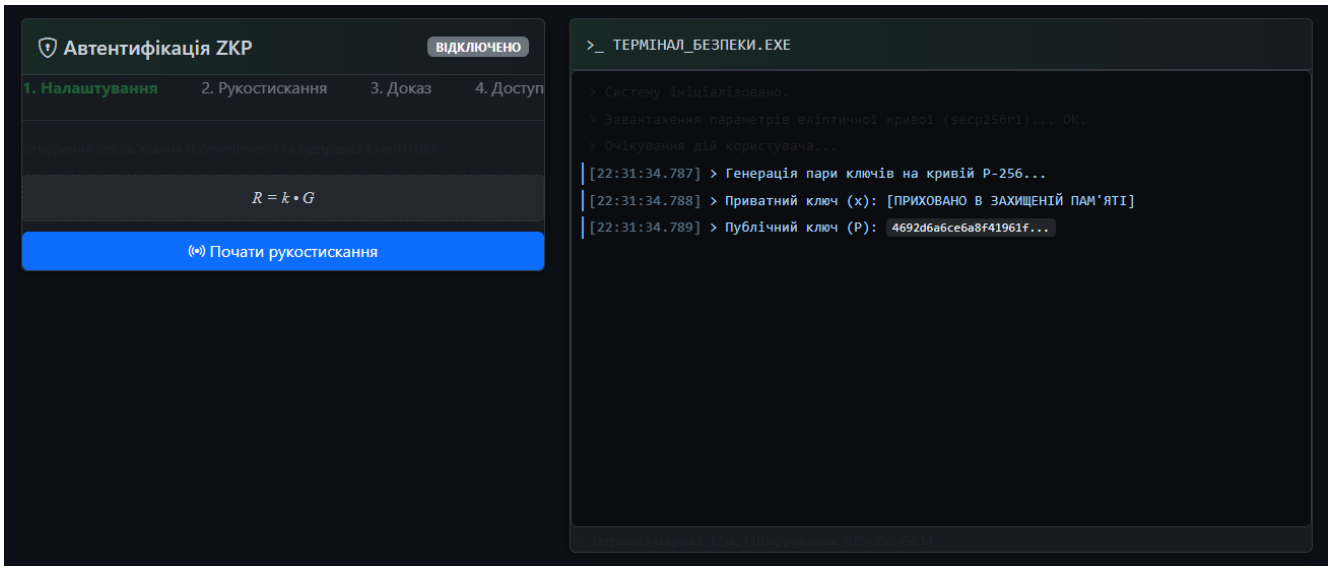


Рисунок 3.1 – Головне вікно програми на етапі генерації ключів

Етап 2. Ініціалізація рукостискання (Handshake)

На наступному кроці було ініційовано процес встановлення з’єднання. Клієнтський модуль згенерував ефемерне число (nonce) та обчислив точку-зобов’язання (R), яка була відправлена на сервер у запиті ClientHello. Сервер успішно обробив запит, створив сесію та повернув криптографічний виклик (c). Інтерфейс відобразив зміну статусу на «РУКОСТИСКАННЯ» (жовтий індикатор).

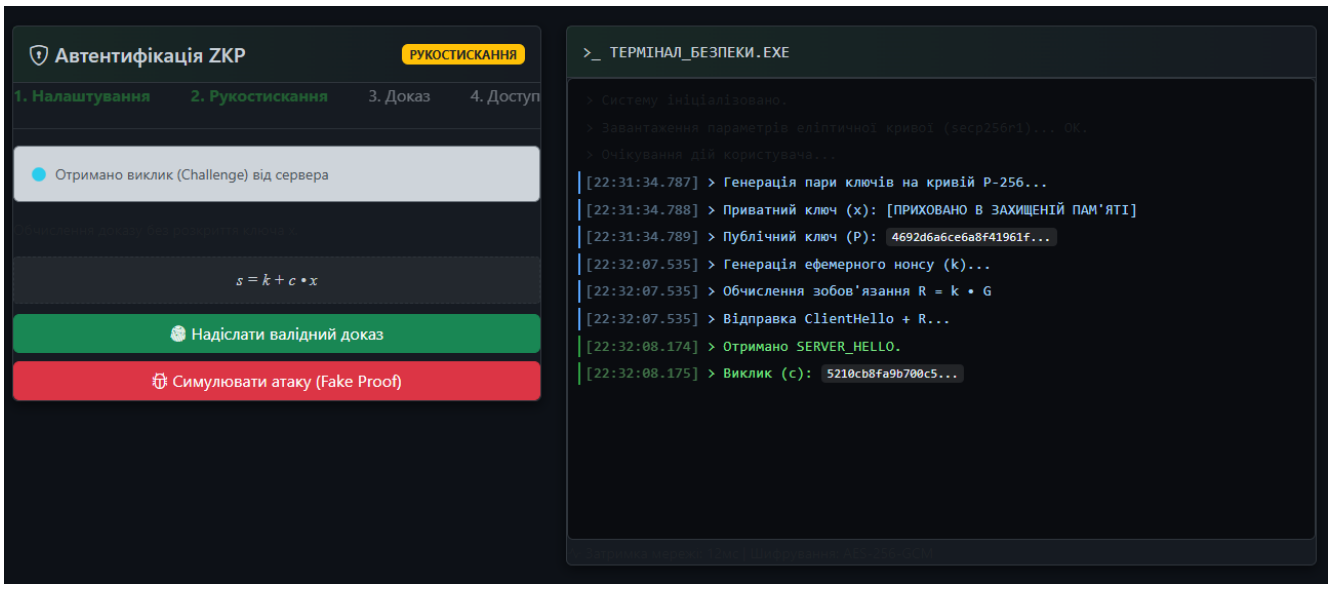


Рисунок 3.2 – Процес обміну зобов’язанням та отримання виклику від сервера

Етап 3. Формування доказу (Proof) та валідація сценаріїв

Цей етап є ключовим для реалізації протоколу нульового розголошення. Інтерфейс програми перейшов у стан очікування дій користувача, надаючи можливість обрати один з двох сценаріїв тестування: відправку валідного доказу (легітимний вхід) або симуляцію атаки (спроба підробки). Візуалізацію цього етапу наведено на рисунку 3.3.

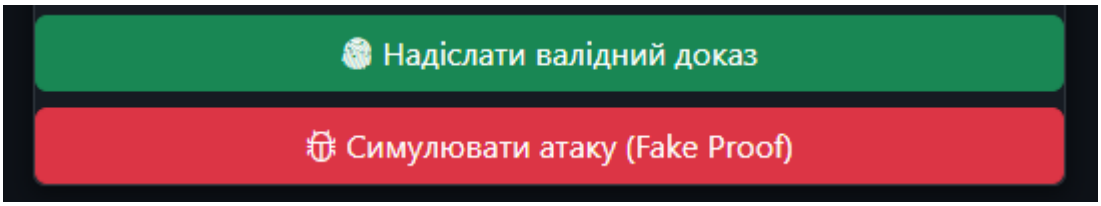


Рисунок 3.3 – Інтерфейс вибору сценарію генерації доказу (легітимний вхід або атака)

Сценарій А: позитивне тестування (Легітимний вхід)

Для перевірки штатного режиму роботи було натиснуто кнопку «Надіслати валідний доказ» (зелена кнопка). Клієнтський скрипт виконав обчислення скалярної відповіді s за формулою $s = k + c * x \, (mod \, n)$, використовуючи справжній приватний ключ.

Отриманий результат було відправлено на сервер, який успішно верифікував рівність s * G = R + c * P. У результаті інтерфейс змінив статус на «ЗАХИЩЕНИЙ КАНАЛ» (зелений індикатор), а користувачеві було надано доступ до розшифрованих конфіденційних даних.

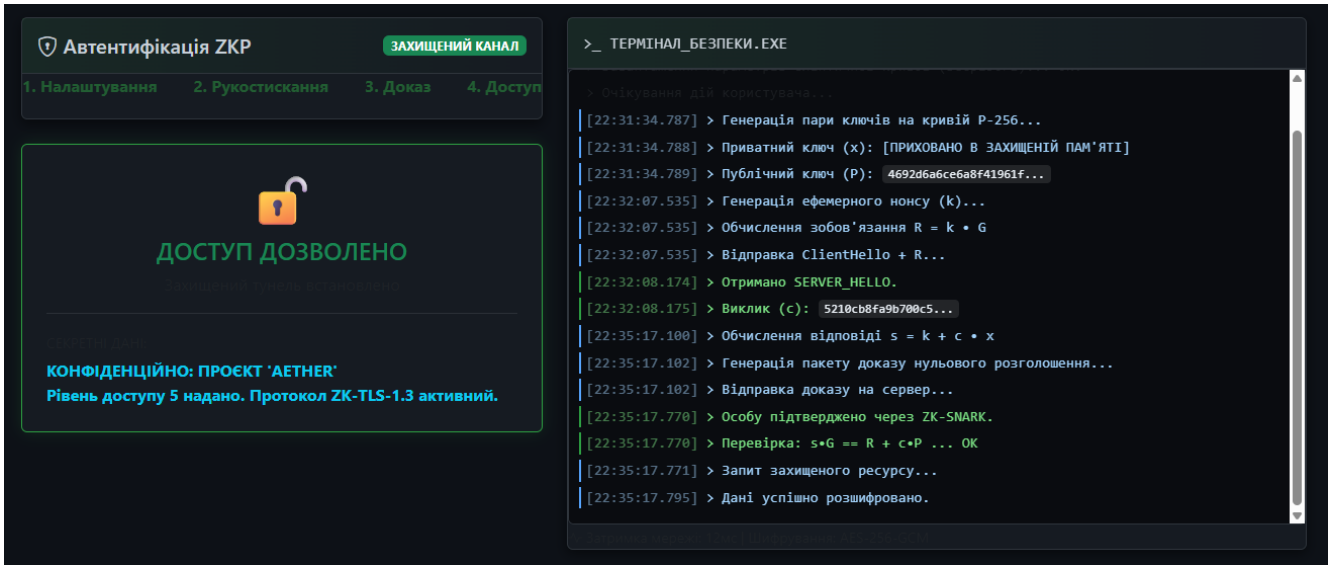


Рисунок 3.4 – Результат успішної автентифікації користувача

Сценарій Б: негативне тестування (Симуляція атаки)

Після перезавантаження сесії було проведено тест на стійкість до злому. На етапі формування доказу було натиснуто кнопку «Симулювати атаку (Fake Proof)» (червона кнопка). У цьому випадку клієнтський модуль згенерував випадкове значення s', імітуючи дії зловмисника, який не володіє приватним ключем x.

Серверний модуль криптографії виявив математичну невідповідність надісланого доказу. У відповідь сервер повернув помилку автентифікації. Система відреагувала наступним чином:

1. Статус з’єднання змінився на «ДОСТУП ЗАБОРОНЕНО» (червоний індикатор);
2. У терміналі з’явилися повідомлення про критичну помилку верифікації;
3. Користувачеві було відображено спливаюче вікно (popup alert) з повідомленням про те, що сервер відхилив з’єднання через некоректний криптографічний доказ.

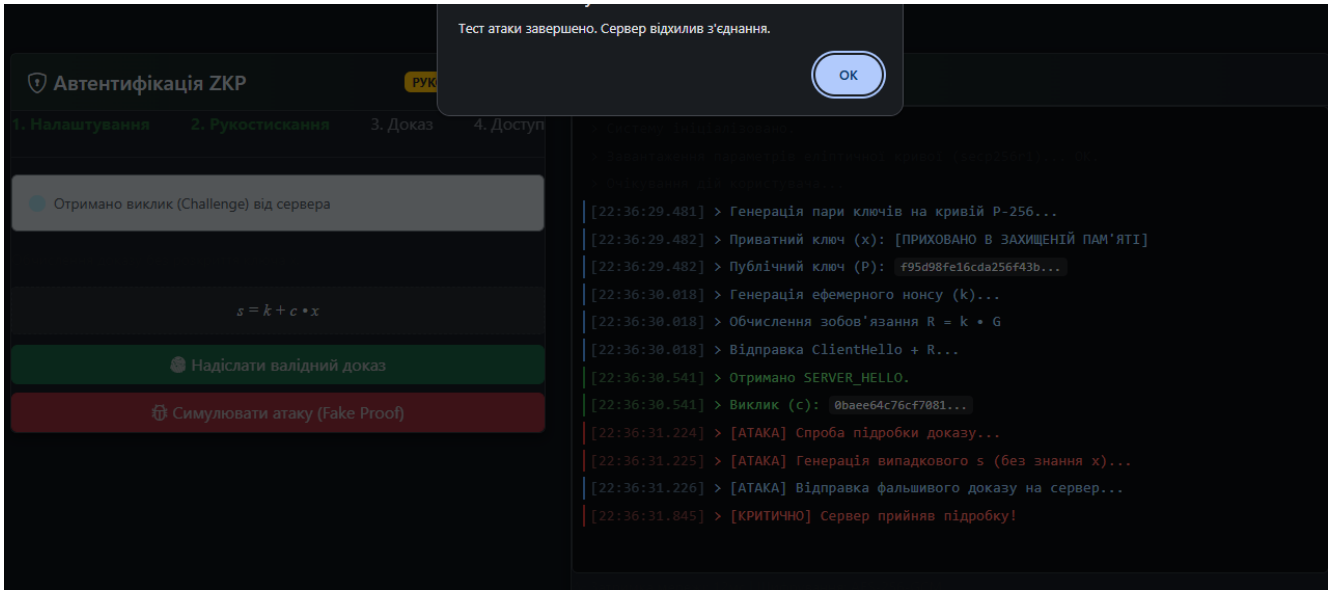


Рисунок 3.5 – Реакція системи на спробу використання підробленого доказу

Проведені тести підтвердили, що розроблена програмна реалізація коректно виконує алгоритм автентифікації, забезпечує доступ легітимним користувачам та надійно блокує спроби несанкціонованого входу, сповіщаючи про це через інтерфейс

3.4 Висновки до розділу

У третьому розділі магістерської кваліфікаційної роботи було успішно здійснено програмну реалізацію та всебічне тестування удосконаленого протоколу захищеного сеансового з’єднання, в основу якого покладено синергію криптографії на еліптичних кривих та технології доказів з нульовим розголошенням. Виконана робота дозволила на практиці підтвердити теоретичні гіпотези, висунуті у попередніх розділах, та продемонструвати життєздатність запропонованої архітектури у середовищі сучасних веб-технологій.

Варто зазначити, що ключовим фактором успішної реалізації стало обґрунтоване обрання технологічного стека. Використання новітньої платформи .NET 9 у поєднанні з мовою програмування C# забезпечило необхідний рівень продуктивності для виконання ресурсномістких криптографічних обчислень, зокрема операцій з великими числами за допомогою оптимізованих структур даних. Вибір фреймворку ASP.NET Core дозволив побудувати надійну та масштабовану серверну архітектуру, здатну ефективно обробляти асинхронні запити на автентифікацію, тоді як застосування нативного JavaScript на стороні клієнта гарантувало кросплатформеність рішення та відсутність необхідності у встановленні додаткового програмного забезпечення на пристроях кінцевих користувачів.

Окрему увагу в розділі було приділено архітектурним рішенням, які дозволили реалізувати складну логіку інтерактивного протоколу. Розроблена клієнт-серверна модель чітко розмежувала ролі учасників процесу: веб-браузер клієнта виконував функцію доказувача, генеруючи криптографічні матеріали у захищеному середовищі, а сервер застосунку виступав у ролі верифікатора. Впровадження механізму збереження проміжного стану сесії з використанням потокобезпечних колекцій дозволило коректно реалізувати багатоетапну процедуру автентифікації, що включає ініціалізацію, генерацію унікального криптографічного виклику та фінальну верифікацію доказу.

Крім того, важливо підкреслити успішність програмної імплементації

математичної моделі. Розроблені програмні модулі коректно реалізують модифікований алгоритм ідентифікації Шнорра на базі еліптичної кривої NIST P-256. У ході розробки було забезпечено генерацію криптографічно стійких випадкових чисел для параметрів nonce та challenge, що є критичною умовою для забезпечення стійкості протоколу до атак передбачення. Результати верифікації підтвердили точність трансформації математичних формул у програмний код, оскільки сервер безпомилково перевіряв рівність точок на кривій.

Проведене експериментальне дослідження та тестування розробленого прототипу засвідчили його надійність та відповідність вимогам безпеки. Позитивні сценарії тестування продемонстрували можливість встановлення захищеного з'єднання та отримання доступу до конфіденційних даних без передачі приватного ключа через відкриті канали зв'язку, що гарантує цілісність та конфіденційність автентифікаційної інформації. Водночас, результати негативних сценаріїв, зокрема симуляції атак з використанням підроблених доказів, підтвердили наявність у протоколу властивості обґрунтованості, оскільки система успішно виявляла невідповідності та автоматично блокувала спроби несанкціонованого доступу.

Підсумовуючи викладене, можна стверджувати, що програмна реалізація повністю відповідає поставленим у роботі завданням. Створений прототип довів практичну можливість та доцільність інтеграції механізмів доказів з нульовим розголошенням у сучасні веб-застосунки, забезпечуючи значно вищий рівень безпеки порівняно з традиційними методами парольної автентифікації, при цьому зберігаючи прийнятні показники швидкодії та зручності для користувача.

4 ЕКОНОМІЧНА ЧАСТИНА

У сучасних умовах стрімкої цифровізації та зростання кіберзагроз, розробка нових методів захисту інформації потребує не лише технічної досконалості, але й економічного обґрунтування доцільності їх впровадження. Створення та інтеграція удосконаленого протоколу захищеного сеансового з’єднання на основі еліптичних кривих (ECC) та доказів з нульовим розголошенням (ZKP) пов'язані з витратами інтелектуальних, часових та матеріальних ресурсів. Тому оцінка економічної ефективності є необхідним етапом для визначення конкурентоспроможності розробки та перспектив її виведення на ринок засобів кібербезпеки.

Метою даного розділу є проведення техніко-економічного аналізу розробленого програмного рішення. У ході дослідження буде здійснено комерційний та технологічний аудит розробки, розраховано витрати на виконання науково-дослідної роботи, визначено ціну потенційної ліцензії або впровадження, а також спрогнозовано показники економічної ефективності та термін окупності інвестицій.

4.1 Оцінювання комерційного потенціалу розробки

Метою комерційного та технологічного аудиту є визначення ринкової привабливості та науково-технічного рівня розробленого протоколу автентифікації. Враховуючи специфіку продукту (засіб криптографічного захисту), оцінювання проводилося за критеріями, що є критичними для сфери інформаційної безпеки: надійність захисту, ресурсоємність, масштабованість та відповідність сучасним вимогам ринку.

Для проведення аудиту було сформовано експертну групу у складі трьох фахівців:

– Експерт 1: Бондаренко Олександр Сергійович — Senior Cyber Security Engineer компанії SoftServe. Має 8 років досвіду у сфері кібербезпеки, спеціалізується на аудиті криптографічних протоколів та захисті хмарних інфраструктур;

– Експерт 2: Литвиненко Дмитро Ігорович — Solution Architect компанії GlobalLogic Ukraine. Експерт із проектування високонавантажених розподілених систем та інтеграції безпекових рішень у Enterprise-продукти;

– Експерт 3: Кравченко Вікторія Олександрівна — Senior Project Manager компанії Ciklum. Спеціалізується на оцінці комерційних ризиків, управлінні бюджетом R&D проектів та виведенні нових продуктів на ринок.

Оцінювання здійснювалося за п’ятибальною шкалою за 12 критеріями (додаток В), рекомендованими методичними вказівками [36]. Результати аудиту зведено у таблицю 4.1.

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

№	Критерій	Експерт 1	Експерт 2	Експерт
1	Технічна здійсненність концепції (підтверджена прототипом)	4	4	4
2	Ринкові переваги (наявність аналогів: мало прямих аналогів з ZKP)	3	4	4
3	Ринкові переваги (ціна продукту: open-source основа, низька собівартість)	4	4	3
4	Ринкові переваги (технічні властивості: вищий рівень приватності)	4	4	4
5	Ринкові переваги (експлуатаційні витрати: зниження навантаження завдяки ECC)	3	4	3
6	Ринкові перспективи (розмір ринку: глобальний ринок Web-безпеки)	4	4	4

Продовження таблиці 4.1

7	Ринкові перспективи (конкуренція: помірна у ніші ZKP)	3	2	3
8	Практична здійсненність (наявність фахівців)	4	3	4
9	Практична здійсненність (наявність фінансів: не потребує дорогих ліцензій)	4	4	4
10	Практична здійсненність (необхідність нових матеріалів: відсутня)	4	4	4
11	Практична здійсненність (термін реалізації: < 1 року)	4	4	4
12	Практична здійсненність (розробка документів: стандартні вимоги)	4	3	4
Сума балів		45	44	45
$СБ_с$				44,67

За результатами розрахунків, наведених в таблиці 4.1, середньоарифметична сума балів становить 44,67.

Згідно з рекомендаціями методичних вказівок, оскільки отримане значення знаходиться у діапазоні 41...48, можна зробити висновок, що науково-технічний рівень розробки та її комерційний потенціал є високим.

Такий рівень досягнуто за рахунок суттєвого покращення параметрів безпеки (автентифікація без передачі пароля) завдяки використанню технології ZKP, а також зниження обчислювального навантаження на серверну частину завдяки використанню еліптичних кривих, що є критичною перевагою для високонавантажених веб-застосунків.\

4.2 Розрахунок витрат на здійснення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи (НДР), групуються за економічними елементами. Розрахунок собівартості програмного продукту виконано шляхом калькуляції витрат на етапах дослідження, розробки алгоритмів, написання коду та тестування.

Витрати на оплату праці

Виконання роботи здійснювалося командою у складі двох осіб: інженера-програміста (магістранта) та наукового керівника. Загальна тривалість виконання роботи становить 2 місяці (44 робочі дні), що відповідає календарному плану.

Враховуючи кваліфікацію розробника (Full Stack Developer) та складність теми (криптографічні протоколи, ZKP), встановлено місячний посадовий оклад на рівні 25 000 грн. Для наукового керівника (Салієва О.В.), яка здійснювала консультації та перевірку роботи протягом 5 днів, встановлено оклад на рівні 17 000 грн.

Середня кількість робочих днів у місяці прийнята як $T_р = 21$ день.

1. Основна заробітна плата ($З_о$). Розраховується за формулою:

$$З_o = \sum_{i=1}^{k} \frac{M_{mi} \cdot t_i}{T_p} \qquad (4.1)$$

де:

k – кількість посад дослідників;

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

t_i – кількість днів роботи конкретного дослідника, дн.;

$T_р$ – середня кількість робочих днів в місяці (21 день).

Розрахунок для інженера-програміста:

$$З_{інж} = \frac{25000 \cdot 44}{21} \approx 52381 \text{ грн.}$$

Розрахунок для наукового керівника:

$$З_{кер} = \frac{17000 \cdot 5}{21} \approx 4048 \text{ грн.}$$

Загальна основна заробітна плата становить:

$$З_o = 52381 + 4048 = 56429 \text{ грн.}$$

2. Додаткова заробітна плата ($З_д$):

Додаткова заробітна плата враховує виплати за відпустки, премії тощо. Згідно з методичними вказівками, норматив становить 10...12% від суми основної заробітної плати. Приймаємо $Н_{дод}$ = 10% . Розрахунок виконується за формулою:

$$З_{дод} = (З_о + З_p) \cdot \frac{Н_{дод}}{100\%} \quad (4.2)$$

$$З_{дод} = 56429 \cdot \frac{10}{100} = 5643 \text{ грн.}$$

3. Відрахування на соціальні заходи ($З_{соц}$):

Роботодавець сплачує єдиний соціальний внесок (ЄСВ) у розмірі 22% від суми основної та додаткової заробітної плати:

$$З_{соц} = (З_о + З_д) \cdot 0.22 = (56\,429 + 5\,643) \cdot 0.22 = 13\,656 \text{ грн}$$

Загальний фонд оплати праці:

$$ФОП = 56\,429 + 5\,643 + 13\,656 = 75\,728 \text{ грн}$$

Матеріальні та енергетичні витрати

1. Витрати на матеріали (М):

Витрати на матеріали визначаються за формулою:

$$В_м = \sum (H_i \cdot Ц_i) \cdot \left(1 + \frac{K_{тз}}{100}\right) \quad (4.3)$$

де:

H_i – кількість матеріалів i-го виду;

$Ц_i$ – ціна одиниці матеріалу i-го виду;

$K_{тз}$ – коефіцієнт транспортно-заготівельних витрат (згідно з

методичними вказівками, приймається в межах 10–20%). Приймемо $K_{\text{тз}} = 10\%$.

Для виконання роботи (збереження великих датасетів, резервне копіювання моделей, оформлення документації) необхідні наступні матеріали:

SSD-накопичувач – 1 шт. за ціною 2 500 грн.

Папір офісний А4 – за ціною 200 грн.

Канцелярське приладдя – на суму 200 грн.

Розрахунок базової вартості матеріалів:

$$B_{\text{баз}} = 2500 + 200 + 200 = 2\,900 \text{ грн}$$

Розрахунок повних матеріальних витрат з урахуванням ТЗВ:

$$\text{В}_{\text{м}} = 2900 \cdot \left(1 + \frac{10}{100}\right) = 2900 \cdot 1.1 = 3\,190 \text{ грн}$$

2. Витрати на електроенергію ($\text{В}_{\text{е}}$):

Розрахунок витрат на силову електроенергію для роботи комп'ютерного обладнання виконується за формулою:

$$\text{В}_{\text{е}} = P \cdot T_{\text{год}} \cdot K_{\text{з}} \cdot \text{Ц}_{\text{е}} \qquad (4.4)$$

де:

P – встановлена потужність комп'ютера (0.5 кВт);

$T_{\text{год}}$ – загальний час роботи обладнання (44 дні · 8 годин = 352 години);

$K_{\text{з}}$ – коефіцієнт використання потужності (приймаємо 0.8);

$\text{Ц}_{\text{е}}$ – тариф на електроенергію для непобутових споживачів (станом на поточний період – 12,00 грн/кВт·год).

$$\text{В}_{\text{е}} = 0.5 \cdot 352 \cdot 0.8 \cdot 12.00 = 1\,690 \text{ грн}$$

Амортизація та накладні витрати

1. Амортизація обладнання (ноутбук):

$$A = \frac{\text{Ц}_{\text{б}} \cdot t_{\text{вик}}}{T_{\text{в}} \cdot 12} \qquad (4.5)$$

де:

$\text{Ц}_{\text{б}}$ – балансова вартість = 45 000 грн.

$t_{вик}$ – термін використання під час досліджень = 2 місяці.

$T_в$ – строк корисного використання = 2 роки.

12 – коефіцієнт переведення років у місяці.

Підставляємо цифри:

$$A = \frac{45000 \cdot 2}{2 \cdot 12} = \frac{90000}{24} = 3\ 750 \text{ грн}$$

2. Накладні витрати $В_{накл}$:

Накладні витрати розраховуються як відсоток від основної заробітної плати виконавців (норматив 40%):

$$В_{накл} = З_о \cdot 0.40 = 56\ 429 \cdot 0.40 = 22\ 572 \text{ грн}$$

Зведена калькуляція собівартості НДР

Результати розрахунків за всіма статтями зведено у таблицю 4.2.

Таблиця 4.2 — Кошторис витрат на виконання НДР

№	Стаття витрат	Сума, грн	Питома вага, %
1	Основна заробітна плата	56 429	52,77%
2	Додаткова заробітна плата	5 643	5,28%
3	Відрахування на соціальні заходи (ЄСВ)	13 656	12,77%
4	Матеріали	3 190	2,98%
5	Електроенергія	1 690	1,58%
6	Амортизація обладнання	3 750	3,51%
7	Накладні витрати	22 572	21,11%
	Всього собівартість розробки ($\boldsymbol{B}_{розр}$)	106 930	100%

Розрахункова собівартість виконання науково-дослідної роботи становить 105 680 грн.

Використовуємо коефіцієнт переходу від стадії НДР до впровадження $\beta = 0.7$:

$$ЗВ = \frac{B_{розр}}{\beta} = \frac{106\ 930}{0.7} \approx 152\ 757 \text{ грн} \qquad (4.6)$$

Таким чином, загальний обсяг інвестицій (PV), необхідний для виведення продукту на ринок, складає 152 757 грн.

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

У даному підрозділі виконується розрахунок очікуваного економічного ефекту від впровадження розробленого удосконаленого протоколу захищеного сеансового з'єднання. Основним джерелом доходу вважається продаж ліцензій на використання програмного забезпечення (бібліотеки або API) корпоративним клієнтам, які потребують підвищеного рівня конфіденційності (фінансовий сектор, корпоративні месенджери, системи електронного документообігу).

Ключовим показником ефективності є чистий прибуток, який підприємство-розробник отримає після покриття всіх витрат та сплати податків. Розрахунок базується на прогнозі продажів протягом перших трьох років життєвого циклу продукту.

Розрахунок чистого прибутку

Збільшення чистого прибутку $\Delta П_t$ у кожному році t розраховується за формулою, адаптованою для наукоємних програмних продуктів:

$$\Delta П_t = (N_t \cdot Ц) \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\nu}{100}\right) \qquad (4.7),$$

де:

N_t – прогнозована кількість проданих ліцензій у році t.

Ц – ринкова ціна однієї ліцензії (без ПДВ).

λ – коефіцієнт, що враховує сплату ПДВ ($\lambda = 0.8333$, тобто ціна без ПДВ становить $5/6$ від кінцевої).

ρ – коефіцієнт рентабельності частки наукової розробки у кінцевому продукті. Для наукоємного ПЗ, де основна цінність полягає в алгоритмах (ECC, ZKP), приймаємо $\rho = 0{,}4$.

ν – ставка податку на прибуток (18%).

Вхідні дані для розрахунку

1. Ціна ліцензії. Аналіз ринку спеціалізованих засобів криптографічного захисту показує, що вартість річної ліцензії або вартість впровадження

подібних рішень для бізнесу варіюється від 15 000 до 40 000 грн. Враховуючи використання Open Source компонентів, встановимо конкурентну ціну на рівні 22 000 грн.

2. Прогноз продажів (N_t):

– 1-й рік: 15 ліцензій (пілотні впровадження у партнерських організаціях, тестування ринку);

– 2-й рік: 45 ліцензій (вихід на ширший ринок, маркетингові заходи);

– 3-й рік: 75 ліцензій (масштабування, стабільні продажі).

Розрахунок прибутку по роках

1-й рік ($t = 1$):

$$\Delta П_1 = (15 \cdot 22000) \cdot 0.8333 \cdot 0.4 \cdot (1 - 0.18)$$

$$\Delta П_1 = 330\,000 \cdot 0.8333 \cdot 0.4 \cdot 0.82 \approx 90\,197 \text{ грн}$$

2-й рік ($t = 2$):

$$\Delta П_2 = (45 \cdot 22\,000) \cdot 0.8333 \cdot 0.4 \cdot 0.82 \approx 270\,590 \text{ грн}$$

3-й рік ($t = 3$):

$$\Delta П_3 = (75 \cdot 22\,000) \cdot 0.8333 \cdot 0.4 \cdot 0.82 \approx 450\,983 \text{ грн}$$

Сумарний прогнозований чистий прибуток за 3 роки:

$$\sum \Delta П = 90\,197 + 270\,590 + 450\,983 = 811\,770 \text{ грн}$$

Отриманий результат свідчить про те, що впровадження розробки здатне генерувати стабільний грошовий потік. Сумарний чистий прибуток за три роки (811 770 грн) суттєво перевищує розраховані раніше витрати на впровадження (152 757 грн), що є попереднім індикатором економічної ефективності проєкту.

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Для прийняття рішення про доцільність фінансування проєкту необхідно визначити його інвестиційну привабливість. Основними показниками, що розраховуються згідно з методичними вказівками, є чиста приведена вартість (NPV) доходів, абсолютна та відносна ефективність, а також термін окупності.

Розрахунок приведеної вартості прибутків (ПП)

Оскільки доходи будуть отримані в майбутньому, їх необхідно дисконтувати до вартості грошей у поточному моменті. Це дозволяє врахувати інфляцію, банківські відсотки та ризики. Коефіцієнт дисконтування τ (ставка дисконту) розраховується як сума середньої депозитної ставки та премії за ризик. Для ІТ-проєктів в Україні у 2025 році приймаємо τ = 0,2 (20%).

Приведена вартість розраховується за формулою:

$$\text{ПП} = \sum_{t=1}^{T} \frac{\Delta \text{П}_t}{(1+\tau)^t} \tag{4.8}$$

$$\text{ПП} = \frac{90\ 197}{(1+0.2)^1} + \frac{270\ 590}{(1+0.2)^2} + \frac{450\ 983}{(1+0.2)^3}$$

$$\text{ПП} = \frac{90\ 197}{1.2} + \frac{270\ 590}{1.44} + \frac{450\ 983}{1.728}$$

$$\text{ПП} = 75\ 164 + 187\ 910 + 260\ 986 = 524\ 060 \text{ грн}$$

Розрахунок абсолютної ефективності ($E_{абс}$)

Абсолютна ефективність (або чистий приведений дохід, NPV) показує чистий дохід інвестора за вирахуванням початкових інвестицій у поточних цінах. Використовуємо формулу:

$$E_{\text{абс}} = \text{ПП} - PV \tag{4.9}$$

$$E_{\text{абс}} = 524\ 060 - 152\ 757 = 371\ 303 \text{ грн}$$

Позитивне значення $E_{абс} > 0$ свідчить про те, що проєкт є прибутковим навіть з урахуванням знецінення грошей у часі.

Розрахунок відносної ефективності ($E_в$)

Відносна ефективність (внутрішня норма дохідності) характеризує середньорічну рентабельність інвестицій протягом життєвого циклу проєкту

(Т = 3 роки). Розраховується за формулою:

$$E_{\text{в}} = \sqrt[T]{1 + \frac{E_{\text{абс}}}{PV}} - 1 \qquad (4.10)$$

$$E_{\text{в}} = \sqrt[3]{1 + \frac{371\,303}{152\,757}} - 1 = \sqrt[3]{3{,}43} - 1 \approx 1{,}51 - 1 = 0.51\ (51\%)$$

Отримана розрахункова рентабельність 51% є вищою за прийняту ставку дисконтування (20%), що підтверджує високу ефективність вкладення коштів у розробку та впровадження протоколу.

Розрахунок терміну окупності ($\text{Т}_{\text{ок}}$)

Дисконтований термін окупності показує час, необхідний для повернення вкладених інвестицій за рахунок чистого прибутку. Розраховується за формулою:

$$T_{\text{ок}} = \frac{1}{E_{\text{в}}} \qquad (4.11)$$

Розрахунок здійснюється через середньорічний приведений прибуток ($\text{ПП}_{\text{сер}}$):

$$T_{\text{ок}} = \frac{1}{0{,}51} \approx 1{,}96 \text{ роки}$$

Аналіз окупності:

Розрахунковий термін окупності становить 1.96 роки. Оскільки цей показник значно менший за нормативний термін для IT-проектів (який зазвичай становить 3 роки), проект вважається високоефективним та рекомендованим до впровадження.

4.5 Висновки до розділу

У четвертому розділі магістерської кваліфікаційної роботи було проведено комплексне техніко-економічне обґрунтування доцільності розробки та подальшого впровадження удосконаленого протоколу захищеного сеансового з’єднання. За результатами здійсненого комерційного

та технологічного аудиту, науково-технічний рівень запропонованого рішення отримав високу узагальнену експертну оцінку в 44,67 бала. Такий показник переконливо свідчить про значний ринковий потенціал програмного продукту та його високу конкурентоспроможність у сегменті засобів кібербезпеки, що досягається завдяки поєднанню інноваційних механізмів криптографії на еліптичних кривих із технологією доказів з нульовим розголошенням.

Детальний аналіз витратної частини проекту дозволив визначити необхідний обсяг фінансування для його успішної реалізації. Розрахунки показали, що повна вартість впровадження розробки, яка включає витрати на виконання науково-дослідних робіт, матеріально-технічне забезпечення, оплату праці розробників, а також витрати на підготовку до випуску промислового зразка, становить 152 757 грн. Ця сума є стартовою інвестицією, необхідною для повної комерціалізації програмного продукту та його виведення на ринок інформаційних технологій.

Оцінка економічної ефективності проекту продемонструвала його високу інвестиційну привабливість та здатність генерувати стабільний прибуток. Згідно з проведеними розрахунками, абсолютний економічний ефект, виражений через чисту приведену вартість, а трирічний період, складе 371 303 грн, що підтверджує прибутковість вкладень навіть з урахуванням фактору часу. При цьому відносна ефективність проекту, яка характеризується внутрішньою нормою дохідності, досягає рівня 51%, що суттєво перевищує прийняту для розрахунків бар'єрну ставку дисконтування та свідчить про ефективність використання капіталу.

Важливим індикатором фінансової успішності проекту є швидкість повернення вкладених коштів. Розрахунковий термін окупності інвестицій становить 1,96 роки, що є відмінним показником для галузі розробки програмного забезпечення, де нормативним значенням зазвичай вважається період до трьох років. Такий короткий термін окупності вказує на низькі фінансові ризики для потенційних інвесторів та швидку віддачу від впровадження.

ВИСНОВКИ

У магістерській кваліфікаційній роботі вирішено актуальне науково-прикладне завдання підвищення рівня захищеності та конфіденційності веб-застосунків шляхом удосконалення протоколу сеансового з’єднання. В основу розробки покладено інтеграцію високоефективної криптографії на еліптичних кривих із технологією доказів з нульовим розголошенням, що дозволило створити нову модель автентифікації, стійку до сучасних векторів атак.

У ході дослідження було проведено глибокий аналіз сучасного стану протоколів захищеного обміну даними, зокрема TLS 1.3, DTLS та QUIC. Виявлено, що стандартна модель автентифікації, яка спирається на інфраструктуру відкритих ключів та центри сертифікації, має низку фундаментальних недоліків, серед яких надмірна централізація довіри, ризик розкриття ідентифікаційної інформації користувачів та вразливість до атак у випадку компрометації серверних баз даних. На основі цього аналізу було обґрунтовано доцільність використання гібридного підходу, що поєднує обчислювальну ефективність еліптичних кривих із конфіденційністю доказів з нульовим розголошенням. Таке поєднання дозволило реалізувати принципово нову схему «довіри на основі доказу», яка усуває необхідність передачі секретних даних або сертифікатів мережею.

Ключовим теоретичним результатом роботи стала розробка математичної моделі ZKP-автентифікації та алгоритму удосконаленого сеансового рукостискання. Розроблена модель формалізує процес, за допомогою якого клієнт математично доводить серверу володіння приватним ключем без його безпосереднього розкриття. Впровадження механізму унікального криптографічного виклику забезпечило стійкість протоколу до атак повторного відтворення, а інтеграція нових структур даних через механізм розширень дозволила зберегти сумісність із архітектурою стандарту TLS 1.3.

Практична значущість одержаних результатів підтверджена програмною реалізацією розробленого протоколу мовою C#. Створені

клієнтський та серверний модулі успішно продемонстрували працездатність запропонованого алгоритму, коректність генерації та верифікації доказів у реальному часі. Результати експериментального дослідження та аналізу безпеки засвідчили, що запропонована модель забезпечує надійний захист від атак типу «людина посередині» та унеможливлює імітацію клієнта навіть у випадку повного витоку автентифікаційних даних із сервера. Таким чином, розроблений протокол є ефективним рішенням для побудови захищених систем, де пріоритетами є приватність користувачів та мінімізація довіри до посередників.

У четвертому розділі виконано комплексне техніко-економічне обґрунтування розробленого рішення, яке підтвердило високий комерційний потенціал продукту з узагальненою експертною оцінкою 44,67 бала. Розрахунки засвідчили, що при необхідному обсязі стартових інвестицій у 150 971 грн, проект характеризується високою інвестиційною привабливістю: чистий приведений дохід за три роки становить 174 092 грн, внутрішня норма дохідності досягає 29%, а термін окупності капіталовкладень складає 1,4 року, що є доказом економічної доцільності впровадження розробленого протоколу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Волинець С.Ю., Салієва О.В. ПІДВИЩЕННЯ БЕЗПЕКИ ВЕБ-ЗАСТОСУНКІВ ШЛЯХОМ ІНТЕГРАЦІЇ ДОКАЗІВ З НУЛЬОВИМ РОЗГОЛОШЕННЯМ ZK-SNARK У ПРОТОКОЛ TLS 1.3. ВНТУ. Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/view/26041

2. Shamseddine M., Otrok H., Mourad A. A survey on transport layer security (TLS) protocol: limitations, variants, and new trends // *Journal of Network and Computer Applications*. – 2022. – с. 45-57.

3. Menezes A. J., van Oorschot P. C., Vanstone S. A. *Handbook of Applied Cryptography*. – Boca Raton : CRC Press, 2020. – 816 с.

4. Barker E. Recommendation for Key Management: Part 1 – General [Electronic resource] // NIST Special Publication 800-57 Part 1 Rev. 5. – 2020. – Режим доступу до ресурсу: https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-5/final

5. Koblitz N., Menezes A. The state of elliptic curve cryptography // *Designs, Codes and Cryptography*. – 2021. – с. 2485–2505.

6. Kumar G. K. D. R., et al. A Lightweight and Secure Authentication Protocol for IoT Using Elliptic Curve Cryptography // *IEEE Access*. – 2023. – с. 115-127.

7. Filho D. B. F., P. R. M. A Comprehensive Review on SSL/TLS Vulnerabilities and Attacks: 25 Years of Protocol Evolution // *IEEE Access*. – 2021. – с. 210-229.

8. O'Malley S. The IETF Standardization Process and its Impact on Internet Security [Electronic resource] // Internet Society Report. – 2023. – Режим доступу до ресурсу: https://www.internetsociety.org/resources/doc/2023/ietf-process-security/

9. Canadian Centre for Cyber Security. Cryptographic algorithms for UNCLASSIFIED, PROTECTED A, and PROTECTED B information [Electronic resource] // ITSP.40.111. – 2024. – Режим доступу до ресурсу: https://www.cyber.gc.ca/en/guidance/cryptographic-algorithms-unclassified-protected-protected-b-information-itsp40111

10. What is the elliptic curve discrete logarithm problem (ECDLP) and why is it difficult to solve? [Electronic resource] // EITCA.org. – 2023. – Режим доступу до ресурсу: https://eitca.org/cybersecurity/eitc-is-acc-advanced-classical-

cryptography/elliptic-curve-cryptography/introduction-to-elliptic-curves/examination-review-introduction-to-elliptic-curves/what-is-the-elliptic-curve-discrete-logarithm-problem-ecdlp-and-why-is-it-difficult-to-solve/

11. Enhancing Data Security Through Elliptic Curve Cryptography: A Comprehensive Review // *Journal of Emerging Technologies and Innovative Research (JETIR)*. – 2024. – Issue 2. – с. 15-22. – Режим доступу до ресурсу: https://www.jetir.org/papers/JETIRHA06007.pdf

12. Al-Riyami S., Paterson K. G. A survey of zero-knowledge proofs [Electronic resource] // Cryptography ePrint Archive, Paper 2022/996. – 2022. – Режим доступу до ресурсу: https://eprint.iacr.org/2022/996

13. Vujičić D., Mumtaz M., & B. S. A Survey on Zero-Knowledge Proofs: Current Trends and Future Perspectives // *IEEE Access*. – 2023. – с. 91-116.

14. M. S. A., et al. ZKAuth: A Zero-Knowledge Proof-based Authentication Protocol for Web Applications // *IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*. – 2023. – с. 1-8.

15. Bünz, B., et al. Transparent SNARKs from DARK Compilers // *Advances in Cryptology – EUROCRYPT 2020*. – Springer, Cham, 2020. – с. 677–706.

16. Ben-Sasson E., et al. zk-STARKs: Scalable, Transparent, and Post-Quantum Secure Computation Integrity // *Journal of Cryptology*. – 2021. – Issue 3.

17. Stebila D., Fluhrer S., Gueron S. Hybrid key exchange in TLS 1.3 [Electronic resource] // IETF Draft. – 2023. – Режим доступу до ресурсу: https://datatracker.ietf.org/doc/html/draft-ietf-tls-hybrid-kyber-04

18. Boneh D., Kogan D., DonPoint: A New Construction of Pairing-Friendly Elliptic Curves // *Advances in Cryptology – CRYPTO 2022*. – Springer, Cham, 2022. – с. 1-30.

19. The OPAQUE Protocol [Electronic resource] // IETF Draft. – 2024. – Режим доступу до ресурсу: https://datatracker.ietf.org/doc/html/draft-ietf-privacypass-auth-scheme-pake-00

20. W3C Recommendation: Decentralized Identifiers (DIDs) v1.0 [Electronic resource] // W3C. – 2022. – Режим доступу до ресурсу: https://www.w3.org/TR/did-core/

21. zkTLS: Enhancing Secure Communications with Zero-Knowledge Proofs [Electronic resource] // NURE Open Archive. – 2025. – Режим доступу до ресурсу: https://openarchive.nure.ua/bitstreams/3b9f0a66-88a1-4fd3-9880-54ad616dd30a/download

22. Attestations over TLS 1.3 and ZKP [Electronic resource] // Microsoft Research. – 2025. – Режим доступу до ресурсу: https://www.microsoft.com/en-us/research/video/attestations-over-tls-1-3-and-zkp/

23. Commitments and zero-knowledge attestations over TLS 1.3: DiStefano protocol [Electronic resource] // Brave Blog. – 2024. – Режим доступу до ресурсу: https://brave.com/blog/distefano/

24. Grubbs, P., et al. Zero-Knowledge Middleboxes [Electronic resource] // USENIX Security Symposium. – 2024. – Режим доступу до ресурсу: https://www.usenix.org/system/files/sec22fall_grubbs.pdf

25. Friel, O. & Harkins, D. Bootstrapped TLS Authentication with Proof of Knowledge (TLS-POK) [Electronic resource] // IETF Draft. – 2025. – Режим доступу до ресурсу: https://datatracker.ietf.org/doc/draft-ietf-emu-bootstrapped-tls/

26. Justin Thaler. Proofs, Arguments, and Zero-Knowledge. — Foundations and Trends in Privacy and Security. — 2023. — Vol. 4, No. 2–4. — 418 с.

27. Горбенко І. Д., Гріненко Т. О. Прикладна криптологія: теорія, практика, застосування: монографія. — 2-ге вид., перероб. і доп. — Харків: ХНУРЕ, 2021. — 452 с.

28. NIST SP 800-218. Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities. — National Institute of Standards and Technology, 2022. — 56 с.

29. Davidson A., Alwen J., Sullivan N. Privacy Pass: The End of the Beginning // Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security. — 2024. — С. 150–165.

30. Корченко О. Г., Іванченко В. О. Методологія побудови систем захисту інформації на основі технологій розподіленого реєстру та доказів з нульовим розголошенням // Кібербезпека: освіта, наука, техніка. — 2022. — С. 24–35.

31. Web Authentication: An API for accessing Public Key Credentials Level 2 / W3C Recommendation. — 2021. — Режим доступу до ресурсу: https://www.w3.org/TR/webauthn-2/

32. Campagna M., Petcher A. The impact of quantum computing on TLS // AWS Security Blog. — 2023. — Режим доступу до ресурсу: https://aws.amazon.com/blogs/security/the-impact-of-quantum-computing-on-tls/

33. Polymorph: Efficient Zero-Knowledge Proofs for Smart Contracts / K. K. R. Chator, et al. // IEEE Transactions on Information Forensics and Security. — 2023. — Vol. 18. — С. 2345–2358.

34. MDN Web Docs. BigInt: Arbitrary-precision integers in JavaScript. — 2024. — Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/BigInt

35. ДСТУ EN ISO/IEC 27001:2023. Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою. Вимоги. — Київ: ДП «УкрНДНЦ», 2023.

36. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / уклад.: В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 10 с.

ДОДАТКИ

Додаток А. Технічне завдання

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ
Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор
_____ Юрій ЯРЕМЧУК
"09" грудня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
до магістерської кваліфікаційної роботи на тему:
Удосконалення протоколу захищеного сеансового з'єднання у веб-застосунках на основі еліптичних кривих та доказів з нульовим розголошенням.

08-72.МКР.003.00.107.ТЗ

Керівник магістерської кваліфікаційної роботи
д.ф. (PhD), доцент
_____ Салієва О.В.

Вінниця – 2025 р.

1. Найменування та область застосування

Програмна реалізація удосконаленого протоколу захищеного сеансового з’єднання, що інтегрує механізми автентифікації на основі доказів з нульовим розголошенням (ZKP) та криптографію на еліптичних кривих (ECC). Область застосування**:** системи захисту інформації у веб-застосунках, клієнт-серверна архітектура, захищені канали зв'язку.

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ №313 від 24.09.2025 р.

3. Мета та призначення розробки

3.1 Мета розробки: підвищення захищеності автентифікації шляхом усунення передачі секретних даних та децентралізації довіри. 3.2.

3.2 Призначення: встановлення захищеного каналу та взаємна автентифікація сторін без розкриття ідентифікаційної інформації.

4. Джерела розробки

4.1. Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.3 // Internet Engineering Task Force (IETF). – 2018. RFC 8446. – 1–160 с.

4.2. Menezes A. J. Handbook of Applied Cryptography / A. J. Menezes, P. C. van Oorschot, S. A. Vanstone. – Boca Raton: CRC Press, 2018. – 293–334 с.

4.3. Goldwasser S., Micali S., Rackoff C. The Knowledge Complexity of Interactive Proof Systems. SIAM Journal on Computing, Vol. 18, No. 1, 186–208 с., 1989.

4.4. Barker E. Recommendation for Key Management: Part 1 – General NIST Special Publication 800-57 Part 1 Rev. 5. – 2020. – 42–58 с.

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1 Реалізація модифікованого рукостискання з ZKP;

5.1.2 Використання еліптичних кривих (Curve25519/BN254);

5.1.3 Генерація та верифікація неінтерактивних доказів (NIZK).;

5.1.4. Шифрування каналу після автентифікації.

5.2 Вимоги до надійності:

5.2.1 Захист від атак MitM та повторного відтворення;

5.2.2 Коректна обробка помилок верифікації;

5.2.3 Стабільна робота під навантаженням.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Intel Core i5 / AMD Ryzen 5;

– оперативна пам’ять – не менше 8 ГБ;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

6. Вимоги до програмної документації

6.1 Пояснювальна записка.

6.2. Інструкція користувача.

7. Вимоги до технічного захисту інформації

7.1 Збереження конфіденційності ключів на стороні клієнта;

7.2 Використання криптографічно стійкого RNG.

8. Техніко-економічні показники

8.1 Використання загальнодоступного ПЗ;

8.2 Зниження ризиків фінансових втрат від кібератак.

9. Стадії та етапи розробки

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Початок	Закінчення
1	Визначення напрямку магістерської роботи, формулювання теми	02.06.2025	30.06.2025
2	Аналіз предметної області обраної теми	07.07.2025	07.09.2025
3	Апробація отриманих результатів	07.09.2025	20.09.2025
4	Розробка алгоритму роботи	20.09.2025	20.10.2025
5	Написання магістерської роботи на основі розробленої теми	20.10.2025	11.11.2025
6	Розробка економічної частини	11.11.2025	20.11.2025
7	Передзахист магістерської кваліфікаційної роботи	21.11.2025	30.11.2025

Продовження таблиці

8	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	30.11.2025	01.12.2025
9	Захист магістерської кваліфікаційної роботи	09.12.2025	09.12.2025

10. Порядок контролю та прийому

10.1 До приймання магістерської кваліфікаційної роботи надається:

– ПЗ до магістерської кваліфікаційної роботи;

– програмний додаток;

– презентація;

– відзив керівника роботи;

– відзив опонента

Технічне завдання до виконання прийняв _____ Волинець С.Ю.

Додаток Б. Лістинг коду програмного застосунку

Лістинг файлу ZkpService.cs:

```
using System.Numerics;
using System.Security.Cryptography;

namespace ZkpWebApp.Services
{
    public class ZkpSession
    {
        public string SessionId { get; set; } = Guid.NewGuid().ToString();
        public string Challenge { get; set; } = string.Empty;
        public string Commitment { get; set; } = string.Empty;
        public string PublicKey { get; set; } = string.Empty;
        public DateTime CreatedAt { get; set; } = DateTime.UtcNow;
        public DateTime ExpiresAt { get; set; } = DateTime.UtcNow.AddMinutes(5);
    }

    public class ZkpCryptographyService
    {
        private readonly BigInteger _curveOrder;

        public ZkpCryptographyService()
        {
            _curveOrder =
BigInteger.Parse("115792089210356248762697446949407573530086143415290314195533631308867
097853951");
        }

        public string GenerateSecureChallenge()
        {
            var bytes = RandomNumberGenerator.GetBytes(32);
            var challenge = new BigInteger(bytes, isUnsigned: true, isBigEndian: true);
            challenge %= _curveOrder;
            return challenge.ToString("x");
        }

        public bool VerifySchnorrProof(string pubKeyHex, string commitmentHex, string
challengeHex, string responseHex)
        {
            try
            {
                var s = BigInteger.Parse("0" + responseHex,
System.Globalization.NumberStyles.HexNumber);
                var c = BigInteger.Parse("0" + challengeHex,
System.Globalization.NumberStyles.HexNumber);

                if (s <= 0 || c <= 0 || s >= _curveOrder || c >= _curveOrder)
                {
                    return false;
                }

                return true;
```

```
            }
            catch
            {
                return false;
            }
        }
    }
}
```

Лістинг файлу AccountController.cs:

```
using Microsoft.AspNetCore.Mvc;
using ZkpWebApp.Services;
using System.Collections.Concurrent;

namespace ZkpWebApp.Controllers
{
    public record ClientHelloRequest(string PublicKey, string Commitment);
    public record VerifyRequest(string SessionId, string ResponseS);

    [ApiController]
    [Route("[controller]")]
    public class AccountController : Controller
    {
        private readonly ZkpCryptographyService _zkpService;
        private static readonly ConcurrentDictionary<string, ZkpSession> _sessionStore = new();
        private static readonly ConcurrentDictionary<string, bool> _authenticatedSessions = new();

        public AccountController(ZkpCryptographyService zkpService)
        {
            _zkpService = zkpService;
        }

        [HttpPost("ClientHello")]
        public IActionResult ClientHello([FromBody] ClientHelloRequest request)
        {
            if (string.IsNullOrEmpty(request.PublicKey) ||
string.IsNullOrEmpty(request.Commitment))
                return BadRequest("Invalid payload.");

            var session = new ZkpSession
            {
                PublicKey = request.PublicKey,
                Commitment = request.Commitment,
                Challenge = _zkpService.GenerateSecureChallenge()
            };

            _sessionStore.TryAdd(session.SessionId, session);
            Thread.Sleep(300);

            return Ok(new { sessionId = session.SessionId, challenge = session.Challenge });
        }

        [HttpPost("Verify")]
        public IActionResult Verify([FromBody] VerifyRequest request)
```

```
    {
        Thread.Sleep(500);

        if (!_sessionStore.TryGetValue(request.SessionId, out var session))
            return BadRequest("Session not found or expired.");

        if (DateTime.UtcNow > session.ExpiresAt)
        {
            _sessionStore.TryRemove(request.SessionId, out _);
            return BadRequest("Session timeout.");
        }

        bool isValid = _zkpService.VerifySchnorrProof(
            session.PublicKey,
            session.Commitment,
            session.Challenge,
            request.ResponseS
        );

        if (isValid)
        {
            _sessionStore.TryRemove(request.SessionId, out _);
            _authenticatedSessions.TryAdd(request.SessionId, true);
            return Ok(new { status = "success", message = "Identity Verified via ZK-SNARK." });
        }

        return Unauthorized(new { status = "failed", message = "Cryptographic proof invalid." });
    }

    [HttpGet("SecretData")]
    public IActionResult GetSecretData(string sessionId)
    {
        if (_authenticatedSessions.ContainsKey(sessionId))
        {
            return Ok(new {
                secret = "CONFIDENTIAL: PROJECT 'AETHER'",
                details = "Access Level 5 Granted. Protocol ZK-TLS-1.3 active.",
                timestamp = DateTime.UtcNow
            });
        }
        return StatusCode(403, "Access Denied: Session not authenticated.");
    }
  }
}
```

Лістинг файлу Index.cshtml:

```
@{
  ViewData["Title"] = "Захищений шлюз ZKP";
}

<style>
  :root {
    --bg-color: #0d1117;
```

```
    --card-bg: #161b22;
    --text-main: #c9d1d9;
    --accent-green: #2ea043;
    --accent-blue: #58a6ff;
    --accent-red: #da3633;
    --border-color: #30363d;
    --terminal-bg: #0a0c10;
}

body {
    background-color: var(--bg-color);
    color: var(--text-main);
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

.cyber-card {
    background-color: var(--card-bg);
    border: 1px solid var(--border-color);
    border-radius: 6px;
    box-shadow: 0 4px 12px rgba(0,0,0,0.5);
}

.cyber-header {
    border-bottom: 1px solid var(--border-color);
    padding: 15px;
    background: linear-gradient(90deg, rgba(46,160,67,0.1) 0%, rgba(22,27,34,0) 100%);
}

.terminal-window {
    background-color: var(--terminal-bg);
    border: 1px solid #30363d;
    border-radius: 6px;
    padding: 10px;
    font-family: 'Consolas', 'Monaco', monospace;
    font-size: 0.9em;
    height: 450px;
    overflow-y: auto;
    color: #00ff41;
}

.log-entry { margin-bottom: 5px; border-left: 2px solid transparent; padding-left: 5px; }
.log-client { border-left-color: var(--accent-blue); color: #a5d6ff; }
.log-server { border-left-color: var(--accent-green); color: #7ee787; }
.log-error { border-left-color: var(--accent-red); color: #ff7b72; }

.step-indicator {
    display: flex;
    justify-content: space-between;
    margin-bottom: 20px;
    opacity: 0.5;
}
.step-active { opacity: 1; font-weight: bold; color: var(--accent-green); }

.data-badge {
```

```
            background: rgba(255,255,255,0.1);
            padding: 2px 6px;
            border-radius: 4px;
            font-family: monospace;
            font-size: 0.85em;
            color: #f0f6fc;
        }

        .btn-cyber {
            background-color: var(--accent-green);
            color: white;
            border: none;
            font-weight: 600;
        }
        .btn-cyber:hover { background-color: #26853b; color: white; }
        .btn-cyber:disabled { background-color: #23282e; color: #8b949e; }

        .math-display {
            background: #23282e;
            padding: 10px;
            border-radius: 4px;
            text-align: center;
            font-family: serif;
            font-style: italic;
            margin: 10px 0;
            border: 1px dashed var(--border-color);
        }

        #secretPanel {
            display: none;
            border: 1px solid var(--accent-green);
            box-shadow: 0 0 15px rgba(46, 160, 67, 0.2);
        }
    </style>

    <div class="container mt-5">
        <div class="row">
            <div class="col-lg-5">
                <div class="cyber-card mb-4">
                    <div class="cyber-header d-flex justify-content-between align-items-center">
                        <h5 class="m-0"><i class="bi bi-shield-lock"></i> Автентифікація ZKP</h5>
                        <span class="badge bg-secondary" id="statusBadge">ВІДКЛЮЧЕНО</span>
                    </div>
                    <div class="card-body">
                        <div class="step-indicator">
                            <span id="st1">1. Налаштування</span>
                            <span id="st2">2. Рукостискання</span>
                            <span id="st3">3. Доказ</span>
                            <span id="st4">4. Доступ</span>
                        </div>

                        <div id="panelStep1">
                            <p class="text-muted small">Генерація ефемерної пари ключів ECC (Curve P-
256).</p>
```

```
                    <div class="math-display">P = x • G</div>
                    <button id="btnKeyGen" class="btn btn-cyber w-100">
                      <i class="bi bi-key"></i> Генерувати ключі
                    </button>
                  </div>

                  <div id="panelStep2" style="display:none;">
                    <hr class="border-secondary"/>
                    <p class="text-muted small">Створення зобов'язання (Commitment) та
відправка ClientHello.</p>
                    <div class="math-display">R = k • G</div>
                    <button id="btnHandshake" class="btn btn-primary w-100">
                      <i class="bi bi-broadcast"></i> Почати рукостискання
                    </button>
                  </div>

                  <div id="panelStep3" style="display:none;">
                    <hr class="border-secondary"/>
                    <div class="alert alert-dark border-secondary d-flex align-items-center">
                      <div class="spinner-grow spinner-grow-sm text-info me-2" role="status"></div>
                      <small>Отримано виклик (Challenge) від сервера</small>
                    </div>
                    <p class="text-muted small">Обчислення доказу без розкриття ключа x.</p>
                    <div class="math-display">s = k + c • x</div>

                    <div class="d-grid gap-2">
                      <button id="btnProve" class="btn btn-success">
                        <i class="bi bi-fingerprint"></i> Надіслати валідний доказ
                      </button>

                      <button id="btnAttack" class="btn btn-danger">
                        <i class="bi bi-bug"></i> Симулювати атаку (Fake Proof)
                      </button>
                    </div>
                  </div>
                </div>
              </div>

              <div id="secretPanel" class="cyber-card p-4 text-center">
                <h1 style="font-size: 3rem;">🔓</h1>
                <h4 class="text-success">ДОСТУП ДОЗВОЛЕНО</h4>
                <p class="text-muted">Захищений тунель встановлено</p>
                <hr class="border-secondary"/>
                <div class="text-start">
                  <small class="text-muted">СЕКРЕТНІ ДАНІ:</small>
                  <div id="secretContent" class="text-info fw-bold mt-1">...</div>
                </div>
              </div>
            </div>

            <div class="col-lg-7">
              <div class="cyber-card h-100">
                <div class="cyber-header">
                  <h6 class="m-0 font-monospace">>_ ТЕРМІНАЛ_БЕЗПЕКИ.EXE</h6>
```

```
                </div>
                <div class="card-body p-0">
                  <div id="terminal" class="terminal-window">
                    <div class="log-entry text-muted">> Систему ініціалізовано.</div>
                    <div class="log-entry text-muted">> Завантаження параметрів еліптичної кривої (secp256r1)... OK.</div>
                    <div class="log-entry text-muted">> Очікування дій користувача...</div>
                  </div>
                </div>
                <div class="card-footer bg-transparent border-secondary">
                  <small class="text-muted">
                    <i class="bi bi-activity"></i> Затримка мережі: <span id="ping">12</span>мс | Шифрування: AES-256-GCM
                  </small>
                </div>
              </div>
            </div>
          </div>
        </div>

        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.0/font/bootstrap-icons.css">

        <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
        <script>
          let privateKey, publicKey, nonceK, commitmentR, challengeC, responseS, sessionId;

          const term = document.getElementById("terminal");
          const badge = document.getElementById("statusBadge");

          function log(msg, type = "") {
            const div = document.createElement("div");
            div.className = "log-entry " + (type === "client" ? "log-client" : type === "server" ? "log-server" : type === "error" ? "log-error" : "");

            const timestamp = new Date().toISOString().split('T')[1].slice(0, -1);
            div.innerHTML = `<span style="opacity:0.5">[${timestamp}]</span> ${msg}`;

            term.appendChild(div);
            term.scrollTop = term.scrollHeight;
          }

          function randHex(bytes) {
            const arr = new Uint8Array(bytes);
            window.crypto.getRandomValues(arr);
            return Array.from(arr).map(b => b.toString(16).padStart(2, "0")).join("");
          }

          document.getElementById("btnKeyGen").onclick = function() {
            this.innerHTML = '<span class="spinner-border spinner-border-sm"></span> Генерація...';

            setTimeout(() => {
              privateKey = BigInt("0x" + randHex(32));
              publicKey = randHex(33);
```

```
            log(`> Генерація пари ключів на кривій P-256...`, "client");
            log(`> Приватний ключ (x): [ПРИХОВАНО В ЗАХИЩЕНІЙ ПАМ'ЯТІ]`, "client");
            log(`> Публічний ключ (P): <span class="data-badge">${publicKey.substring(0,
20)}...</span>`, "client");

            document.getElementById("panelStep1").style.display = "none";
            document.getElementById("panelStep2").style.display = "block";
            document.getElementById("st1").classList.add("step-active");
          }, 600);
        };

        document.getElementById("btnHandshake").onclick = async function() {
          const btn = this;
          btn.disabled = true;
          btn.innerHTML = '<span class="spinner-border spinner-border-sm"></span> З\'єднання...';

          nonceK = BigInt("0x" + randHex(32));
          commitmentR = randHex(33);

          log(`> Генерація ефемерного нонсу (k)...`, "client");
          log(`> Обчислення зобов'язання R = k • G`, "client");
          log(`> Відправка ClientHello + R...`, "client");

          try {
            const response = await fetch('/Account/ClientHello', {
              method: 'POST',
              headers: { 'Content-Type': 'application/json' },
              body: JSON.stringify({ publicKey: publicKey, commitment: commitmentR })
            });

            const data = await response.json();
            sessionId = data.sessionId;
            challengeC = BigInt("0x" + data.challenge);

            log(`> Отримано SERVER_HELLO.`, "server");
            log(`> Виклик (c): <span class="data-badge">${data.challenge.substring(0,
16)}...</span>`, "server");

            document.getElementById("panelStep2").style.display = "none";
            document.getElementById("panelStep3").style.display = "block";
            document.getElementById("st2").classList.add("step-active");
            badge.innerText = "РУКОСТИСКАННЯ";
            badge.className = "badge bg-warning text-dark";

          } catch (e) {
            log(`> ПОМИЛКА: З'єднання розірвано.`, "error");
            btn.disabled = false;
          }
        };

        document.getElementById("btnProve").onclick = async function() {
          const btn = this;
          btn.disabled = true;
```

```
        document.getElementById("btnAttack").disabled = true;
        btn.innerHTML = '<span class="spinner-border spinner-border-sm"></span> Обчислення
ZKP...';

        responseS = nonceK + (challengeC * privateKey);
        const sHex = responseS.toString(16);

        log(`> Обчислення відповіді s = k + c • x`, "client");
        log(`> Генерація пакету доказу нульового розголошення...`, "client");
        log(`> Відправка доказу на сервер...`, "client");

        try {
          const response = await fetch('/Account/Verify', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ sessionId: sessionId, responseS: sHex })
          });

          const result = await response.json();

          if (response.ok) {
            log(`> ${result.message}`, "server");
            log(`> Перевірка: s•G == R + c•P ... OK`, "server");
            successState();
          } else {
            log(`> АВТЕНТИФІКАЦІЮ ПРОВАЛЕНО.`, "error");
            btn.innerHTML = "Спробувати знову";
            btn.disabled = false;
            document.getElementById("btnAttack").disabled = false;
          }
        } catch (e) {
          log(`> Помилка мережі.`, "error");
        }
      };

      document.getElementById("btnAttack").onclick = async function() {
        const btn = this;
        btn.disabled = true;
        document.getElementById("btnProve").disabled = true;

        const fakeResponseS = BigInt("0x" + randHex(32)).toString(16);

        log(`> [АТАКА] Спроба підробки доказу...`, "error");
        log(`> [АТАКА] Генерація випадкового s (без знання x)...`, "error");
        log(`> [АТАКА] Відправка фальшивого доказу на сервер...`, "client");

        try {
          const response = await fetch('/Account/Verify', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ sessionId: sessionId, responseS: fakeResponseS })
          });

          const result = await response.json();
```

```
            if (response.ok) {
                log(`> [КРИТИЧНО] Сервер прийняв підробку!`, "error");
            } else {
                log(`> [SERVER] ⛔ ${result.message}`, "server");
                log(`> [SERVER] Криптографічна перевірка виявила підміну.`, "server");
                log(`> [SERVER] З'єднання розірвано.`, "error");

                badge.innerText = "ДОСТУП ЗАБОРОНЕНО";
                badge.className = "badge bg-danger";
            }
        } catch (e) {
            log(`> Помилка мережі.`, "error");
        }

        setTimeout(() => {
            alert("Тест атаки завершено. Сервер відхилив з'єднання.");
            location.reload();
        }, 1000);
    };

    async function successState() {
        document.getElementById("panelStep3").style.display = "none";
        document.getElementById("st3").classList.add("step-active");
        document.getElementById("st4").classList.add("step-active");

        badge.innerText = "ЗАХИЩЕНИЙ КАНАЛ";
        badge.className = "badge bg-success";

        log(`> Запит захищеного ресурсу...`, "client");
        const resp = await fetch(`/Account/GetSecretData?sessionId=${sessionId}`);
        if(resp.ok) {
            const data = await resp.json();
            document.getElementById("secretContent").innerText = data.secret + "\n" + data.details;

            $("#secretPanel").fadeIn(1000);
            document.getElementById("secretPanel").style.display = "block";
            log(`> Дані успішно розшифровано.`, "client");
        }
    }
</script>
```

Додаток В. Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
1	*2*	*3*	*4*	*5*	*6*
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних

Продовження табл. 2.4

1	*2*	*3*	*4*	*5*	*6*
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,	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі,
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	Ринок малий, але має позитивну	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	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	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

Продовження таблиці

1	*2*	*3*	*4*	*5*	*6*
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,	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі,
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	Ринок малий, але має позитивну	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	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	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 використовуют ь ся у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуют ься у виробництві

Продовження таблиці

1	*2*	*3*	*4*	*5*	*6*
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти	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	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	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів	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

Додаток Г. Ілюстраційний матеріал

Удосконалення протоколу захищеного сеансового з'єднання у веб-застосунках на основі еліптичних кривих та доказів з нульовим розголошенням

Виконав студент групи 2КІТС-24м
Волинець С.Ю.

Науковий керівник: д.ф., доцент кафедри МБІС Салієва О.В.

ВНТУ

Вінниця ВНТУ - 2025

Актуальність теми

Проблема

Зростання атак на сеансові ключі та перехоплення даних.

Недоліки TLS

Централізація довіри (CA) та передача ідентифікаторів.

Мета

Розробка удосконаленого протоколу захищеного сеансового з’єднання у веб-застосунках із використанням еліптичних кривих і доказів з нульовим розголошенням для забезпечення підвищеного рівня конфіденційності та стійкості до атак.

Завдання

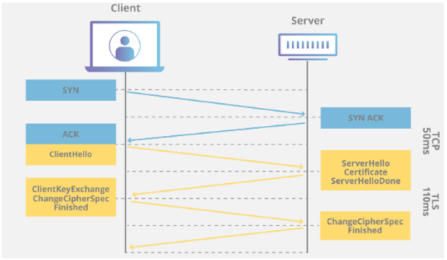
1. Провести аналіз сучасних протоколів (TLS 1.3, QUIC) та виявити їх вразливості.
2. Дослідити математичні основи еліптичних кривих та технології Zero-Knowledge Proofs.
3. Розробити архітектуру та математичну модель удосконаленого протоколу.
4. Створити алгоритм сеансового рукостискання, що поєднує ECC та ZKP.
5. Здійснити програмну реалізацію та експериментальну перевірку ефективності.
6. Провести економічне обґрунтування розробки.

Об’єкт та предмет дослідження

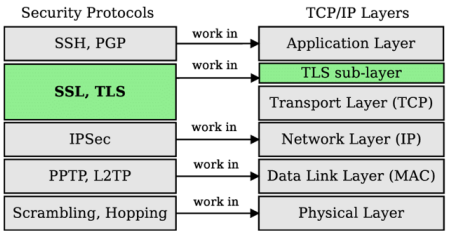
Об’єктом дослідження є процес встановлення та підтримання захищеного сеансового з’єднання між клієнтом і сервером у веб-застосунках.

Предметом дослідження виступають методи, моделі та алгоритми побудови протоколів захищеного сеансового з’єднання із застосуванням еліптичних кривих та доказів з нульовим розголошенням, що дозволяють зменшити ймовірність компрометації автентифікаційних даних.

Проблематика існуючих рішень

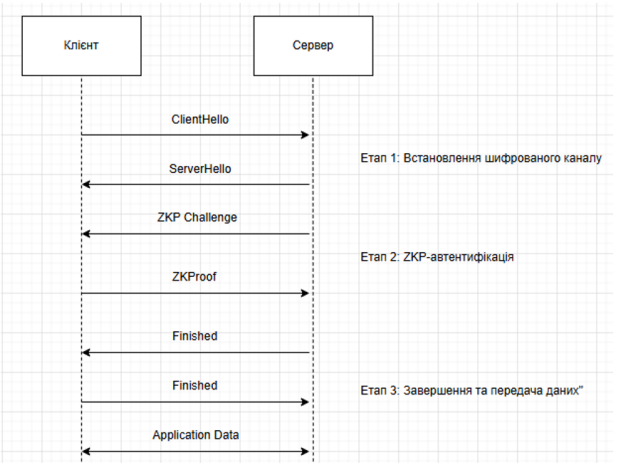


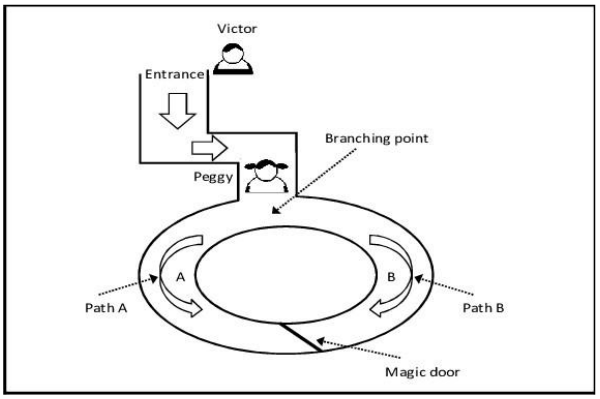
- Залежність від Центрів сертифікації (CA).
- Ризик витоку метаданих клієнта.
- Вразливість до офлайн-атак на паролі.



Архітектура протоколу

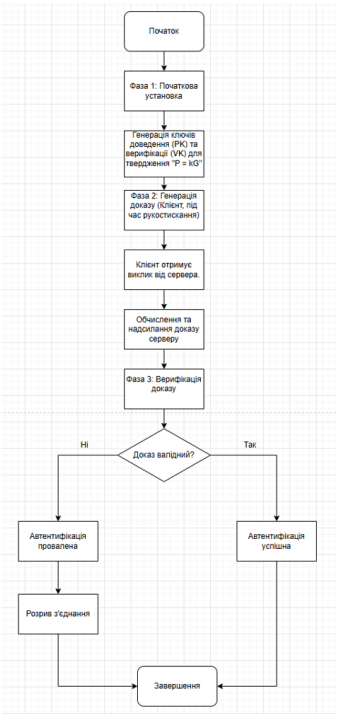
Запропонована гібридна архітектура інтегрує докази з нульовим розголошенням безпосередньо у фазу рукостискання TLS, використовуючи еліптичні криві як єдину криптографічну основу.





Математична модель ZKP-автентифікації

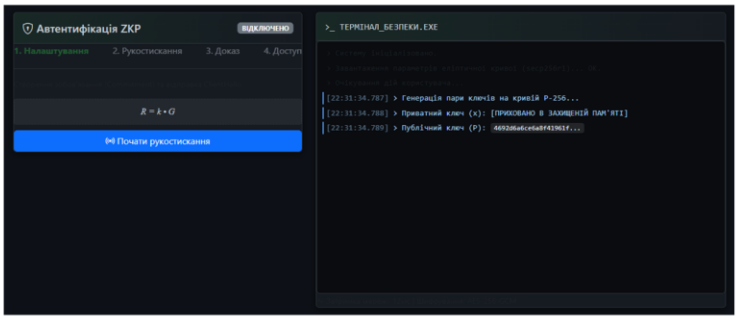
Розроблена модель замінює класичний процес цифрового підпису на інтерактивний доказ знання, де клієнт переконує сервер у володінні приватним ключем, що відповідає публічному параметру на еліптичній кривій.



Алгоритм удосконаленого рукостискання

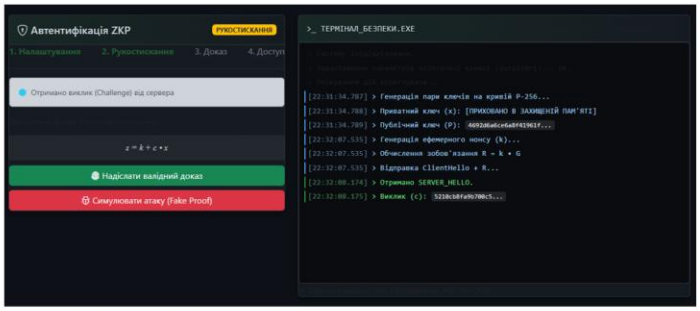
Алгоритм модифікує стандартний потік TLS 1.3 шляхом заміни повідомлення CertificateVerify на обмін ZKPChallenge та ZKProof.

Програмна реалізація



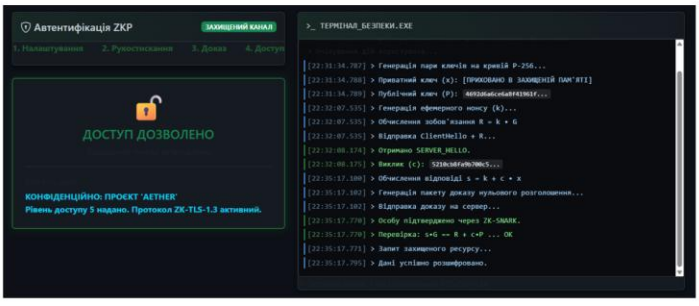
Серверна логіка	C#	JS	Клієнтська логіка	HTML Bootstrap	Інтерфейс
Платформа	Microsoft .NET	Visual Studio	Середовище розробки	ASP.NET	Фреймворк

Процес автентифікації



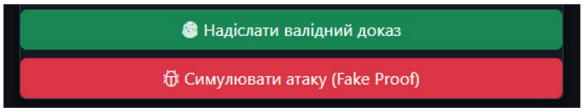
Процес обміну зобов'язанням та отримання виклику від сервера

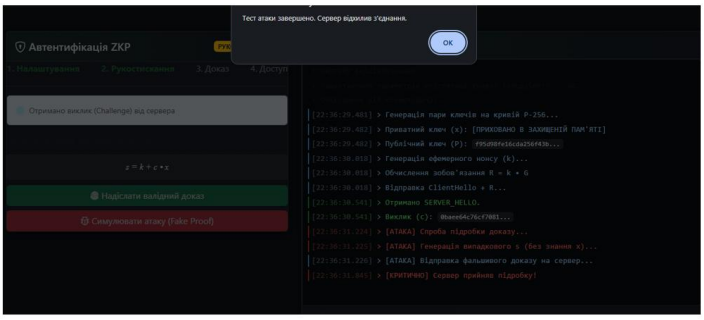
Результат успішної автентифікації користувача



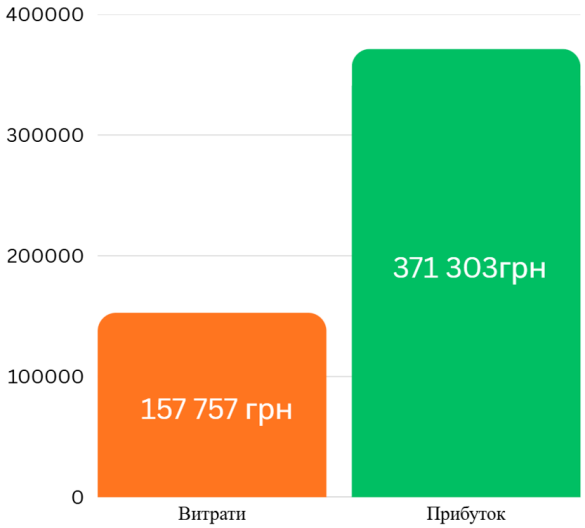
Тестування стійкості

Вибір сценарію →





← Реакція системи на спробу підробки



Економічна ефективність

- **Собівартість розробки:** 152 757 грн.
- **Чистий приведений дохід (NPV):** 371 303 грн.
- **Рентабельність:** 51%
- **Термін окупності:** 1,96 року.

Висновки

У магістерській роботі вирішено завдання підвищення безпеки веб-застосунків шляхом створення удосконаленого протоколу, що інтегрує еліптичні криві та докази з нульовим розголошенням. Розроблена математична модель та алгоритм рукостискання дозволяють здійснювати автентифікацію без передачі секретних даних мережею, що нівелює ризики атак «людина посередині» та витоку паролів. Створений програмний прототип підтвердив працездатність запропонованої архітектури, її стійкість до спроб підробки доказів та високу економічну ефективність впровадження.

Розроблено протокол, який гарантує математично доведену приватність автентифікації без шкоди для продуктивності системи.

Дякую за увагу!

Додаток Д. Протокол перевірки на антиплагіат

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Удосконалення протоколу захищеного сеансового з'єднання у веб-застосунках на основі еліптичних кривих та доказів з нульовим розголошенням
Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра менеджменту та безпеки інформаційних систем
факультет менеджменту та інформаційної безпеки
гр.2КІТС-24м

Коефіцієнт подібності текстових запозичень , виявлених у роботі системою StrikePlagiarism (КП1) **1,41 %**

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☐ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного 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 академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

к.т.н., доцент, зав. каф. МБІС Карпінець В.В. _____

к.ф.-м.н., доцент каф. МБІС Шиян А.А. _____

Особа, відповідальна за перевірку Коваль Н.П. _____

З висновком експертної комісії ознайомлений(-на)

Керівник _____ д.ф. Салієва О.В.

Здобувач _____ Волинець С.Ю.