Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»
Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка

до кваліфікаційної роботи
ОС Магістр

на тему: Дослідження методів тестування продуктивності та масштабованості в хмарних системах

за освітньою програмою: 12 Інженерія програмного забезпечення
зі спеціальності: 121 Інженерія програмного забезпечення

Виконала: студентка групи ПЗ 2422: /Христина МАКАРОВА/

Керівник: ____ /Тетяна ГРИШЕЧКІНА/

Нормоконтролер: /Світлана ВОЛКОВА/

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Дніпро – 2026 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»
Department «Computer information technology»

Explanatory note
to the qualification work
OS Master

on the topic: Research on Methods for Testing Performance and Scalability in Cloud Systems

according to educational curriculum 12 Software engineering
in the Speciality: 121 Software engineering

Done by the student of the group PZ 2422: ____ /Khrystyna MAKAROVA/

Scientific Supervisor: ____ /Tetyana HRYSHECHKINA/

Normative controller: /Svitlana VOLKOVA/

Dnipro – 2026

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: Комп'ютерні технології і системи
Кафедра: Комп'ютерні інформаційні технології
Рівень вищої освіти: магістр
Освітня програма: 12 Інженерія програмного забезпечення
Спеціальність: 121 Інженерія програмного забезпечення
(шифр та назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
(підпис) (Ім'я ПРІЗВИЩЕ)
Дата _____

З А В Д А Н Н Я

на кваліфікаційну роботу для здобуття ступеня магістра
(ступінь вищої освіти)

студентці: Макаровій Христині Євгенівни
(Прізвище Ім'я По батькові)

1. Тема роботи: Дослідження алгоритмів шифрування даних з використанням еліптичних кривих

Керівник роботи: Гришечкіна Тетяна Сергіївна
(Прізвище Ім'я По батькові, науковий ступінь, вчене звання)

затверджені наказом від «02» жовтня 2025 р. № 1401 ст

2. Строк подання студентом роботи: 11.01.2026 р.

3. Вихідні дані до роботи: алгоритми криптографії, шифрування, метрики дослідження

4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):

4.1 Аналітична частина: дослідити процеси алгоритмів шифрування сучасного стану

4.2 Наукове дослідження: обгрунтувати метрики дослідження

4.3 Розробка: виконати розробку алгоритмів шифрування, протестувати та налагодити алгоритми

4.4 Ефективність алгоритмів: дослідити ефективність за метриками

5. Перелік графічного матеріалу (з точним зазначення обов'язкових креслень): схеми проєктування, блок-схеми роботи алгоритмів, графічне представлення результатів розробки алгоритмів, слайди презентації

КАЛЕНДАРНИЙ ПЛАН

№ пор.	Назва етапів кваліфікаційної роботи	Термін виконання розділів роботи	При-м ітка
1	Описати вступну частину кваліфікаційної роботи	10.10.2025	
2	Виконати аналіз літературних джерел із шифрування даних з використанням еліптичних кривих	01.11.2025	
3	Виконати дослідження еліптичної криптографії в шифруванні даних та сфери її застосування	12.11.2025	30%
4	Обґрунтувати вибір напряму дослідження	18.11.2025	
5	Виконати вибір метрик оцінки ефективності та обґрунтувати їх	27.11.2025	
6	Провести експерименти, реалізувати тестові сценарії та підтримку працездатності алгоритмів	10.12.2025	
7	Виконати проєктування архітектури системи шляхом виконання розробки алгоритмів шифрування Чандрасекхара та Діффі-Геллмана	15.12.2025	50%
8	Виконати тестування алгоритмів, використовуючи обрані методи та налагодити програмні алгоритми	23.12.2025	
9	Виконати дослідження ефективності алгоритмів за метриками	29.12.2025	
10	Надати загальні висновки та рекомендації щодо дослідження	03.01.2025	
11	Оформлення пояснювальної записки та тез доповідей	07.01.2025	
12	Розробка демонстраційної презентації роботи	09.01.2025	100%
13	Подання кваліфікаційної роботи до кафедри	16.01.2025	
14	Захист кваліфікаційної роботи на засідання Екзаменаційної комісії	20.01.2026	

Дата видачі завдання «11» вересня 2025 р.

Керівник дипломної роботи _____ Тетяна ГРИШЕЧКІНА
(підпис) (ПІБ)

Студентка _____ Христина МАКАРОВА
(підпис) (ПІБ)

РЕФЕРАТ

Об’єктом дослідження є методи захисту інформації від несанкціонованого доступу.

Предметом дослідження є алгоритми шифрування на основі еліптичних кривих, їх ефективність і безпека.

Метою роботи є дослідження алгоритмів шифрування даних, засновані на еліптичних кривих, з метою виявлення їх переваг та недоліків у порівнянні з традиційними криптографічними методами, а також визначення сфер їх ефективного застосування.

Методами дослідження є метрики оцінки криптографічної стійкості, час шифрування та дешифрування, час генерування ключів, енергоефективність і витрати на зберігання та передачу даних.

Результати та їх новизна: виконані алгоритми Чандрасекхара та Діффі-Геллмана, які надають якісне шифрування даних, а також надають розуміння, який алгоритм необхідно використовувати у яких ситуаціях.

Пояснювальна записка складається зі вступу, 4 розділів, висновків, бібліографічного списку та 7 додатків:

− у вступі описується суть дослідження та її актуальність. Складається із 2 сторінок;

− у першому розділі описується дослідження сучасного стану шифрування даних з використанням еліптичних кривих. Складається із 29 сторінок;

− у другому розділі описуються загальні методи проведення наукового дослідження. Складається із 11 сторінок;

− у третьому розділі описується розробка інструментальних засобів для дослідження алгоритмів шифрування даних з використанням еліптичних кривих. Складається із __ сторінок;

− у четвертому розділі описується дослідження ефективності алгоритмів за метриками. Складається із 29 сторінок;

− у висновках викладені загальні рекомендації та подальший розвиток дослідження. Складається із 3 сторінок;

− додатки містять технічне завдання, специфікацію, текст програми, опис програми, інструкції для користувачів і програмістів.

Таблиць – 13, рисунків – 22, бібліографія – 83 джерел.

Ключові слова: алгоритми шифрування даних, алгоритм Чандрасекхара, еліптичні криві, криптографічна стійкість, протокол Діффі-Геллмана, час шифрування та дешифрування.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАК, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

AES – Advanced Encryption Standard

DES – Data Encryption Standard

DSA – Digital Signature Algorithm, алгоритм цифрового підпису

ECC – Elliptic Curve Cryptography, криптографія на основі еліптичних кривих

ECDH – Elliptic curve Diffie-Hellman, еліптична крива для обміну ключами Діффі-Геллмана

ECDLP – Elliptic Curve Discrete Logarithm Problem, задача дискретного логарифмування в групі точок еліптичної кривої

ECDSA – Elliptic Curve Digital Signature Algorithm, еліптична крива для цифрових підписів

HTTPS – HyperText Transfer Protocol Secure

IoT – Інтернет речі

MITM-атака – Man-In-The-Middle attack

PGP – Pretty Good Privacy

RSA – криптографічний алгоритм Rivest, Shamir та Adleman

SHA – Secure Hash Algorithm Version

SSL – Secure Sockets Layer

TLS – Transport Layer Security

ЕЦП – електронний цифровий підпис

IT – інформаційні технології

ВСТУП

У сучасному світі безпека передачі інформації постає однією з ключових задач у різних областях, включаючи фінансові системи, державне керування, медицину, промисловість тощо. Із розвитком цифрових технологій зростає й кількість загроз, пов’язаних із несанкціонованим доступом до даних. Криптографія є основним інструментом, що використовується для забезпечення безпеки даних та алгоритми шифрування займають центральне місце в забезпеченні конфіденційності, цілісності та автентичності. Одним із найбільш перспективних направлень сучасної криптографії є алгоритми, засновані на еліптичних кривих (ECC). На відміну від традиційних методів шифрування, таких як алгоритм Rivest, Shamir та Adleman (RSA), алгоритми на еліптичних кривих надають більш високий ступінь безпеки при менших витратах на обчислювальні ресурси, що робить їх актуальними для використання в ресурсозалежних системах, таких як пристроях Інтернет речей (IoT) [1-5].

В умовах стрімкого зросту об’ємів передавання інформації та збільшення кількості кібератак використання ефективних методів шифрування стає все більше актуальною задачею. Алгоритми шифрування, що засновані на еліптичних кривих, забезпечують високий рівень безпеки при невеликих ключах, що робить їх особливо привабливими для мобільних пристроїв та інших обмежених за обчислюванням задачах.

Актуальність дослідження полягає в необхідності поглибленого вивчення алгоритмів шифрування, а також розробці та оптимізації їх практичного застосування в умовах реальних загроз.

Об’єктом дослідження є методи захисту інформації від несанкціонованого доступу.

Предметом дослідження є алгоритми шифрування на основі еліптичних кривих, їх ефективність і безпека.

Метою роботи є дослідження алгоритмів шифрування даних, засновані на еліптичних кривих, з метою виявлення їх переваг та недоліків у порівнянні з традиційними криптографічними методами, а також визначення сфер їх ефективного застосування.

Дослідження включає до себе виконання наступних задач:

- аналіз літературних джерел шифрування даних з використанням еліптичних кривих;
- дослідження еліптичної криптографії в шифрування даних;
- аналіз сфери застосування шифрування даних еліптичними кривими;
- аналіз криптографічних методів захисту інформації;
- обґрунтування вибору напрямку дослідження;
- опис метрик оцінки ефективності;
- проведення експериментів та реалізація тестових сценаріїв;
- підтримка працездатності алгоритмів;
- виконання розробки та тестування алгоритмів;
- дослідження ефективності алгоритмів за метриками.

Методами дослідження є метрики оцінки криптографічної стійкості, час шифрування та дешифрування, час генерування ключів, енергоефективність і витрати на зберігання та передачу даних.

Науковою новизною є поглиблене дослідження застосування еліптичних кривих для шифрування даних в умовах обмежених обчислювальних ресурсів. Було запропоновано два алгоритми та їх порівняльна оцінка, яка надає розуміння до підходу оцінювання алгоритмів за метриками та подальшого розвитку криптографії.

Практичним значенням дослідження є використання запропонованих алгоритмів для високоефективних та безпечних систем шифрування даних для різного роду застосунків, включаючи мобільні пристрої, IoT та систем з обмеженими ресурсами.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ СУЧАСНОГО СТАНУ ШИФРУВАННЯ ДАНИХ З ВИКОРИСТАННЯМ ЕЛІПТИЧНИХ КРИВИХ

1.1 Аналіз літературних джерел із шифрування даних з використанням еліптичних кривих

Еліптичні криві залишаються важливою частиною сучасних криптографічних систем. Нові дослідження продовжують розвивати теоретичні аспекти ECC, фокусуючись на більш безпечних та ефективних реалізаціях. У роботах [6-10] можна ознайомитися з поточною ситуацією ECC в умовах можливих квантових атак та узагальнюють перспективи для постквантових рішень. Автори приділяють увагу тому, як еліптичні криві можуть використовуватися в комбінації з іншими схемами для захисту від погроз із сторони квантових комп'ютерів. Також у роботах обговорюється останні дослідження в області ECC, включаючи нові еліптичні криві з покращеними характеристиками безпеки. Особливу увагу приділяється кривим, стійким до атак на побічні канали.

Останнім часом з'явилися нові дослідження, які фокусуються на порівнянні ECC з іншими криптографічними методами, такими як квантозахищені криптосистеми. У роботах [11-16] висвітлюють ряд наступних аспектів:

- продуктивність та безпека ECC та класичних алгоритмів, як RSA;
- новітні постквантові рішення;
- переваги ECC у мобільних пристроях та застосунках IoT;
- підкреслення загроз зі сторони квантових атак;
- способи використання ECC у гібридних схемах, що можуть забезпечити додатковий захист майбутніх квантових комп'ютерів тощо.

ECC продовжує розширятися в різних областях, таких як фінанси, блокчейн та IoT. В останні роки дослідники фокусувалися на застосування ECC в умовах, де важливі компактність і безпека. У дослідженнях [17-21] автори розглядають, як алгоритми на основі еліптичних кривих застосовуються для створення цифрових

підписів та захисту транзакцій у таких системах як Bitcoin та Ethereum. Також виконується огляд ефективних схем захисту пристроїв IoT з використанням ECC, де головною задачею є мінімізація споживання ресурсів при забезпеченні високого рівня безпеки.

Із збільшенням погроз зі сторони квантових комп'ютерів і розвитком атак на побічні канали, нові дослідження фокусуються на покращення безпеки існуючих ECC-систем. У тому числі, розроблюються міри захисту від аналізу часу виконання операцій та споживання енергії. У роботі [22] досліджують можливі квантові атаки на ECC з аналізом того, наскільки вразливі схеми та які міри можна застосувати для підвизення їх безпеки. А в науковій роботі [23] автор описує сучасні атаки на побічні канали (наприклад, аналіз споживання енергії та часу виконання операцій) та пропонує покращені методи захисту ECC від них.

Сучасні ж дослідження направлені на розробку квантостійких схем, які можуть інтегрувати ECC з іншими постквантовими методами для захисту від квантових атак. Наприклад, у роботі [24] досліджуються гібридні криптографічні системи, що поєднують ECC з ізогенієвим криптографічним підходом, який є стійким до квантових атак. Ці методи можуть стати основою для наступного покоління ECC. А от у дослідженні [25] уже наводять плавний перехід від класичних схем на основі еліптичних кривих до повністю квантостійким схемам. Автори припускають, що гібридні підходи з використанням ECC будуть грати ключову роль у перехідний період.

Шифрування на основі еліптичних кривих продовжує демонструвати свою важливість у сучасному світі. Це можна побачити з опису вищевказаних досліджень. В останні роки дослідження ECC акцентуються на наступних ключових направленнях:

- підвищення безпеки в умовах квантових обчислень;
- покращення стійкості до атак на побічні канали.

Розвиток гібридних схем, які поєднують ECC з іншими постквантовими методами, є перспективним направленням для майбутніх систем шифрування.

1.2 Дослідження еліптичної криптографії в шифруванні даних

ECC є методом шифрування даних, заснованим на математичних властивостях еліптичних кривих. Основною перевагою є забезпечення високої безпеки при менших ключів у порівнянні з традиційними криптосистемами, такими як RSA або Digital Signature Algorithm (DSA). Така перевага робить еліптичну криптографію особливо привабливою для застосування в мобільних застосунках і системах з обмеженими обчислювальними ресурсами [26].

Математичною основою є еліптичні криві, які є набором точок, що задовольняють рівняння наступного виду [27]:

$$y^2 = x^2 + a \times x + b, \tag{1.1}$$

де a і b – константи, що визначають форму кривої.

Такі криві мають особливу математичну структуру, яка дозволяє ефективно реалізовувати криптографічні операції. Однією із ключових операцій є складання двох точок на еліптичній кривій. Така операція лежить в основі криптографічних алгоритмів, таких як створення ключів і шифрування [27].

ECC використовується для створення наступних пар ключів [27-28]:

- закритого типу, який є випадковим числом;
- відкритого типу, який є точкою на еліптичній кривій, отримана в результаті перемноження базової точки кривої на закритий ключ.

Безпека ECC заснована на складності знаходження закритого ключа, коли відома тільки відкрита та базова точки. Така задача є проблемою дискретного логарифму на еліптичних кривих [28-29].

Як уже було зазначено, перевагою ECC є забезпечення рівня безпеки при використанні значно менших ключів, у порівнянні з іншими криптосистемами. Наприклад, для забезпечення 128-бітної безпеки RSA необхідний ключ довжиною

3072 біти, у той час як для ECC достатньо ключа в 256 біт. Така перевага робить ECC ефективним для застосування в пристроях з обмеженими обчислювальними ресурсами та низькою пропускною здатністю мережі [29].

1.2.1 Математичні основи еліптичних кривих

Еліптичні криві мають важливі математичні властивості, які використовуються для вирішення задач, пов’язаних з факторизацією чисел, криптографічними протоколами та дослідженням діофантових рівнянь. Еліптична крива над полем K (кінцеве поле або поле речових чисел) визначається як множина рішень рівняння виду (1.1). При цьому повинна виконуватися наступна умова дискримінанту [30-31]:

$$\Delta = -\ 16\times(4\times a^3 + 27\times b^2)\neq 0. \qquad (1.2)$$

Така умова (1.2) необхідна для уникнення точок перегину або самоперетину. Приклад побудованої еліптичної кривої за рівнянням (1.1) наведено на рисунку 1.1.

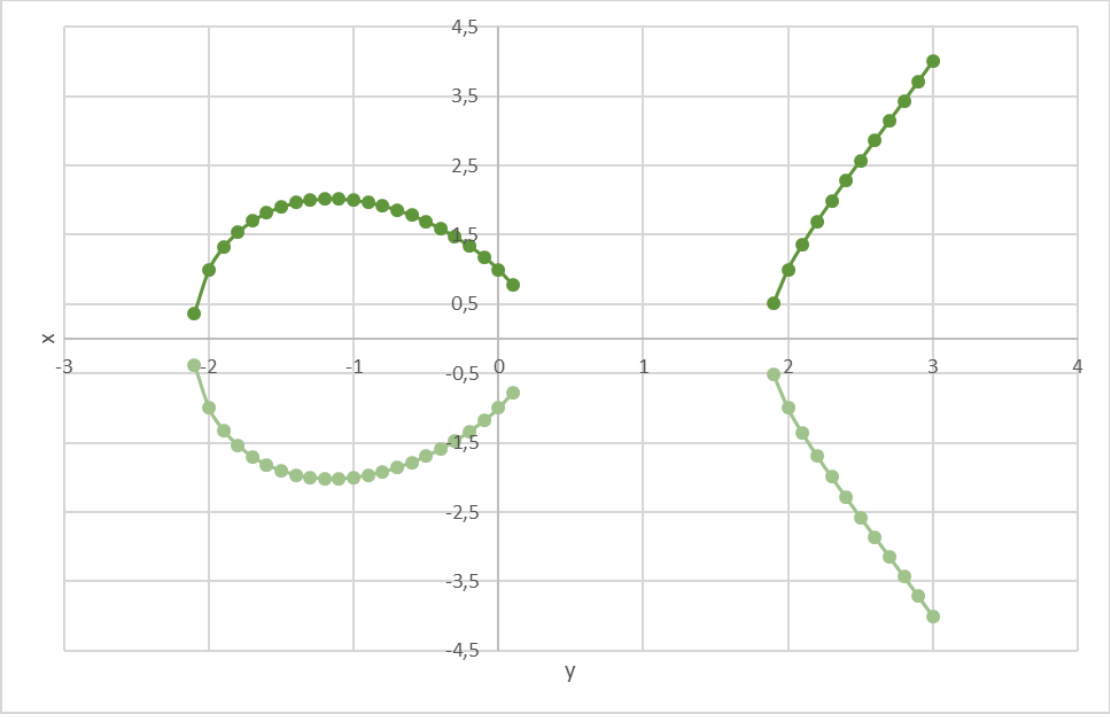

Рисунок 1.1 – Приклад побудованої еліптичної кривої за рівнянням (1.1)

Одним із унікальних властивостей еліптичної кривої є множина її точок, яка утворює абелеву групу з операцією складання. Точками кривої є пари (x, y), які задовольняють рівняння та спеціальна точка на безкінечність O, яка слугує нейтральним елементом у цій групі. Операція складання точок на еліптичній кривій можу бути визначена геометрично. Якщо задані дві точки, то їх сума є третьою точкою перетину прямої, що проходить через ті дві точки з кривою [32]:

$$P = \left(x_1, y_1\right),\ Q = \left(x_2, y_2\right) \Rightarrow P + Q = \left(x_3, y_3\right). \tag{1.3}$$

Якщо точки P та Q співпадають, тобто дорівнюють одна одній, на еліптичній кривій може бути, використовується дотична до кривої в точці P для визначення їх суми. Складання точок (1.3) на еліптичній кривій може бути виражено алгебраїчно наступним чином [32-33]:

1. Якщо $(x_1 \neq x_2)$, то нахил прямої, що з'єднує P та Q, повинен дорівнювати:

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}. \tag{1.4}$$

2. Якщо $P = Q$, нахил дотичної до кривої в точці P обчислюється наступним чином:

$$\lambda = \frac{3 \times x_1^2 + a}{2 \times y_1}. \tag{1.5}$$

3. Нові координати суми $P + Q = (x_3, y_3)$ обчислюються наступним чином:

$$\{x_3 = \lambda^2 - x_1 - x_2; \qquad y_3 = \lambda \times (x_1 - x_3) - y_1. \tag{1.6}$$

Еліптичні криві також розглядаються над кінцевими полями F_q, де q – ступінь простого числа. У даному випадку множина точок еліптичної кривої скінченна і її структура стає особливо корисною для застосування в криптографії. Наприклад, еліптичні криві застосовується в криптографії завдяки їх груповим властивостям. Проблема еліптичних кривих (ECDLP) вважається обчислювально складною. Знаючи точку P та її кратну точку Q [33]:

$$Q = k \times P, \tag{1.7}$$

де k – невідомий множник.

Така властивість використовується для створення криптографічних протоколів, таких як [33]:

- еліптична крива для цифрових підписів (ECDSA);
- еліптична крива для обміну ключами Діффі-Геллмана (ECDH).

У еліптичної кривої є такі поняття як J-інваріант та теорема Хассе. Їх класифікують як властивості кривої. Інваріантом J (модульним інваріантом) є класифікація еліптичних кривих над полем і не змінюється при певних трансформаціях кривих, що обчислюється наступним чином [34]:

$$j(E) = 1728 \times \frac{4 \times a^3}{4 \times a^3 + 27 \times b^2}, \tag{1.8}$$

де a і b є коефіцієнтами з рівняння еліптичної кривої (1.1).

J-інваріант є важливим параметром, який дозволяє розрізнювати криві за їх модульною структурою [34].

Теорема Хассе оцінює кількість точок на еліптичній кривій над кінцевим полем F_q. Теорема стверджує, що число точок N на кривій задовольняє наступній нерівності [34]:

$$|N - (q + 1)| \leq 2 \times \sqrt{q}. \quad (1.9)$$

Результат (1.9) показує, що кількість точок на еліптичній кривій завжди близько до $q + 1$ з деякими відхиленнями, не перевищуючих $2\sqrt{q}$. Теорема Хассе є основою для аналізу еліптичних кривих над кінцевими полями, що важливо для криптографічних застосунків [34].

Еліптичні криві – важливий математичний об’єкт, що застосовується в різних областях, включаючи криптографію та теорію чисел. Їх унікальні властивості, такі як наявність групової структури та складність обчислення дискретних логарифмів, роблять їх особливо корисними для безпеки криптографічних алгоритмів.

1.2.2 Еліптична криптографія в межах України

Еліптична криптографія в Україні набирає все більші оберти, а особливо в контексті сучасних викликів, таких як кібербезпека, захист персональних даних та цифрова трансформація. В умовах постійного зросту числа онлайн-погроз та необхідності захисту інформації на всіх рівнях (державному, корпоративному, особистому тощо), ECC стала важливим інструментом завдяки своїй ефективності та компактним розмірам ключів [35-36].

Основними аспектами використання та розвитку еліптичної криптографії в Україні є [36-38]:

1) використання ECC у державному секторі для захисту конфіденційності даних та забезпечення безпеки комунікацій, що актуально для систем, пов’язаних з критичною інфраструктурою, керуванням даними громадян та урядових ресурсів:

− електронні документи та підписи, що впорядковані на законодавчому рівні та еліптична криптографія активно розповсюджена у використанні електронних цифрових підписах (ЕЦП), що дозволяє скоротити час перевірки автентичності підпису та підвищити рівень безпеки;

− у рамках платформи електронних послуг Дія, яка включає цифрові та електронні документи, використовуються сучасні криптографічні методи для забезпечення безпеки даних та автентифікації користувачів;

2) застосування комерційними та фінансовими структурами еліптичної криптографії для захисту транзакцій, комунікацій та даних користувачів:

− активне застосування ECC у державних банках для шифрування даних, забезпечуючи безпеку онлайн-транзакцій та захист облікових записів кліжнтів;

− використання криптозахисту в криптовалюті, яка активно використовується в Україні;

3) українські університети та дослідницькі центри активно займаються розвитком та вивченням сучасних криптографічних методів як в теоретичних аспектах, так і у вирішенні проблем дискретного логарифму на еліптичних кривих у практиці:

− українські вищі навчальні заклади (наприклад, Харківський національний університет радіоелектроніки (ХНУРЕ) або Київський політехнічний інститут (КПІ) та навчальні інститути проводять дослідження в області застосування ECC у захисті інформаційних систем і мереж;

− Україна розроблює та адаптує міжнародні криптографічні стандарти з урахуванням місцевих особливостей та вимог;

4) через інтеграцію в міжнародні організації та угоди, виконується активна адаптація стандартів безпеки під міжнародний рівень через рекомендативний характер таких технік у Європейському Союзі, де використовується еліптична криптографія:

− використання сучасних методів шифрування Transport Layer Security (TLS) і HyperText Transfer Protocol Secure (HTTPS), які підтримують ECC, що дозволяє забезпечити надійне шифрування даних між клієнтами та серверами;

− алгоритми ECDSA та ECDH стали стандартами для багатьох українських компаній та організацій, що працюють з міжнародними партнерами.

У таблиці 1.1 наведене порівняння основних аспектів використання ЕСС в Україні.

Таблиця 1.1 – Порівняння аспектів використання еліптичної криптографії в Україні

Сектор	Основне застосування ЕСС	Приклади організацій	Переваги використання ЕСС	Виклики та обмеження
Державний	ЕЦП, захист даних, автентифікація користувачів	Платформа Дія, податкові служби	Висока безпека, компактність ключів	Необхідність розвитку інфраструктури
Фінансовий	Шифрування транзакцій, захист користувальницьких даних, онлайн-банкінг	ПриватБанк, ОщадБанк	Зменшення розміру ключів, швидке виконання операцій	Перехід від застарілих методів шифрування
Навчальний	Дослідження в області кібербезпеки, захист інформаційних систем	ХНУРЕ, КПІ	Розробка нових рішень на основі ЕСС, навчання спеціалістів	Нестача спеціалістів в області криптографії
Міжнародний	Відповідність міжнародним вимогам безпеки, використання Інтернет-протоколів	Протоколи TLS або HTTPS	Відповідність міжнародним стандартам безпеки	Перехід на сучасні стандарти, законодавче регулювання

Еліптична криптографія в Україні представляє собою важливу складову сучасних систем безпеки, забезпечуючи захист даних як на державному рівні, так і в приватних компаніях. В умовах прискореної цифровізації та зросту кіберпогроз ЕСС

надає потужний та ефективний інструмент для створення надійних рішень, відповідних міжнародним стандартам.

1.2.3 Аналіз переваг і недоліків еліптичної криптографії

Еліптична криптографія отримала широке розповсюдження через високу ефективність при відносно невеликій довжині ключів, що робить її привабливою для використання в різних областях [39-40].

На рисунку 1.2 наведені основні переваги та недоліки ECC.

Рисунок 1.2 – Схема з перевагами та недоліками ECC

За наведеною схемою можна скласти таблицю, яка наводить більш детальний опис переваг і недоліків, а також наочно показує відмінності [41-42].

Переваги і недоліки ECC наведені в таблиці 1.2.

Таблиця 1.2 – Переваги та недоліки еліптичної криптографії

№ з/п	Переваги	Недоліки
1	2	3
1	Менший розмір ключів. Використання ключів меншого розміру, аніж у традиційних методах для досягнення аналогічного рівня безпеки	Складність реалізації. Збільшення помилок при розробці, які можуть призвести до злому систем
2	Висока швидкість. Операції на еліптичних кривих, такі як генерація ключів і підпис повідомлень, виконуються швидше, аніж в інших криптосистемах з аналогічним рівнем безпеки	Чутливість до атак за побічними каналами. Схильність до атак на основі аналізу електромагнітного випромінювання або часових характеристик
3	Ефективне використання ресурсів. Через меншу обчислювальність ресурсів, ECC підходить для пристроїв з обмеженою потужністю	Патентні обмеження. Присутність застарілих патентів, що іноді збільшує вартість упровадження ECC
4	Більш висока безпека на біт. За рахунок використання математичних принципів забезпечує більш високий рівень безпеки на біт ключа в порівнянні з традиційними алгоритмами	Менший досвід використання та дослідження. Існує менше академічних досліджень та аудитів у порівнянні з іншими криптосистемами, що залишає можливість для несподіваних атак

Продовження таблиці 1.2

1	2	3
5	Менший об’єм даних. Використання ECC знижує кількість переданих і збережених даних за рахунок менших розмірів ключів в цифрових підписів, що зменшує навантаження на мережу та прискорює процеси шифрування та дешифрування даних	Уразливість до квантових атак. Алгоритм Шора, реалізований на квантовому комп’ютері, може ефективно вирішувати задачі факторизації та обчислення дискретного логарифму на еліптичних кривих, що робить традиційні алгоритми ECC небезпечними у випадку появи потужних квантових комп’ютерів

6	Широка підтримка та стандартизація. Підтримка більшістю сучасних криптографічних бібліотек, стандартів	Залежність від обраної кривої. Деякі криві можуть бути скомпрометовані або мати недостатню криптографічну стійкість, що потребує ретельного вибору та стандартизації допустимих кривих

Еліптична криптографія пропонує багато переваг, особливо для систем з обмеженими ресурсами таких як мобільні пристрої та IoT завдяки компактним розмірам ключів та ефективним обчисленням. Однак, складність її реалізації та вразливість до побічних каналів потребують додаткових мір обережності. У той же час, із розвитком квантових технологій, використання ECC, як й інших криптографічних систем, у майбутньому може потребувати заміни на постквантові криптосистеми.

1.3 Аналіз сфери застосування шифрування даних еліптичними кривими

Шифрування даних за допомогою ECC користується великим попитом у багатьох сферах життя. Основними напрямками є інформаційні технології (IT), фінанси, Інтернет речі, державні установи, цифрові посвідчення тощо [43-45].

У таблиці 1.3 наведені основні напрямки сфер застосування еліптичних кривих у шифруванні даних.

Таблиця 1.3 – Сфери застосування шифрування даних еліптичними кривими

Сфера застосування	Опис сфери застосування
1	2
Сектор інформаційних технологій	
Веббезпека	Використання еліптичних кривих у шифруванні в протоколах безпеки HTTPS, Secure Sockets Layer (SSL), TLS тощо. Сайти застосовують ECC для захисту передачу даних між користувачами та серверами. Наприклад, сучасні бравзери та вебсистеми підтримують сертифікати SSL з еліптичними кривими з мінімальною затримкою та максимальним захистом
Електронні підписи	Алгоритми цифрових підписів на основі ECC, наприклад ECDSA, застосовуються для забезпечення автентичності та цілісності повідомлень і документів. Такий захист необхідний для електронної комерції, юридичних документів та інших транзакцій
Захист мобільних пристроїв	ECC широко використовується в мобільних застосунках та операційних системах для захисту даних, зо передаються через мобільну мережу, включаючи комунікації в месенджерах, платіжних системах та біометричну інформацію

Продовження таблиці 1.3

1	2
Сектор інформаційних технологій	
Хмарні технології	Провайдери хмарних сервісів застосовують еліптичні криві для шифрування даних на серверах під час передачі інформації між сховищем і клієнтом, забезпечуючи безпеку конфіденційних даних у хмарних сховищах та обчислень
Сектор фінансовий	

Платіжні системи	ECC використовується для захисту транзакцій в онлайн-банкінгу та мобільних платежів, забезпечуючи автентифікацію користувачів та шифрування даних карт
Криптовалюта	У блокчейн-системах шифрування на кліптичних кривих застосовується для створення та керування приватними та публічними ключами та підпису транзакцій. ECC забезпечує безпеку операцій та анонімність власників криптогаманців
Сектор IoT	
Захист малопотужних пристроїв	Через низькі обчислювальні вимоги ECC підходить для шифрування даних на IoT-пристроях, таких як розумні годинники, датчики, камери та інші системи з обмеженими ресурсами, що дозволяє реалізувати безпечну автентифікацію пристроїв та шифрування їх даних
Безпека мережі	Використання еліптичних кривих для шифрування в мережах IoT для забезпечення з'єднання між пристроями та серверами, що захищає від атак та витоків даних
Електронні паспорти	Захист інформації про громадян та запобігати підробку документів таких як цифрові посвідчення особистості

Продовження таблиці 1.3

1	2
Системи голосування	У системах електронного голосування ECC допомагає захистити конфіденційність і цілісність даних, а також запобігає підміну результатів голосування
Сектор воєнних і державних систем	
Захист конфіденційних даних	Воєнні та державні системи, що оброблюють секретні дані, використовують ECC для забезпечення безпеки при передачі та зберігання інформації. Важливим аспектом є здатність ECC забезпечити високий рівень захисту при обмежених обчислених потужностях
Шифрування зв'язку	У захищених системах зв'язку для автентифікації та шифрування трафіку застосовуються алгоритми на основі еліптичних кривих

Шифрування на еліптичних кривих стало невід'ємною частиною інформаційної безпеки, а особливо в сфері IT завдяки продуктивності, енергоефективності та надійності. Воно використовується у всіх ключових направленнях: від веббезпеки та мобільних застосунків до фінансових систем та IoT.

1.4 Аналіз криптографічних методів захисту інформації

1.4.1 Класифікація криптографічних методів захисту інформації

Сучасні криптографічні методи захисту даних є ключовою частиною системи технічних засобів інформаційної безпеки. Ці методи можна розділити на такі категорії як методи гешування, шифрування, ЕЦП, які мають свої методи для їх реалізації та різні сфери застосування [46-48].

Вони наведені в таблиці 1.4.

Таблиця 1.4 – Категорії криптографічних методів захисту інформації

Категорія	Опис категорії	Приклади
Методи гешування	Застосовуються для перевірки цілісності даних без необхідності читання їх вмісту. Геш-функції перетворюють довільний об’єм даних у фіксоване значення (геш). Навіть мінімальні зміни у вихідних даних призводять до істотних змін гешу, що робить такі методи ефективними для виявлення маніпуляцій з інформацією.	Secure Hash Algorithm Version 2 (SHA-256) використовуються активно в перевірці цілісності файлів
Методи шифрування	Процес перетворення вихідних даних у зашифровану форму, яка стає нечитабельною без відповідного ключа. Шифрування буває симетричним та асиметричним. У сучасних умовах симетричне шифрування часто використовується для захисту великих об’ємів даних, а асиметричне – для безпеки обміну ключами	Advanced Encryption Standard (AES), Data Encryption Standard (DES)
Методи електронного цифрового підпису	Забезпечують гарантії автентичності та цілісності даних, а також підтверджують авторство. ЕЦП формується з використанням асиметричних алгоритмів шифрування та засновується на закритому ключі відправника та гешування. Отримувач може перевірити підпис за допомогою відкритого ключа, тим самим засвідчуючись у тому, що дані не були змінеими та походять саме від указаного відправника	У правових та фінансових системах для підтвердження транзакцій, підписання договорів і документів

Основні принципи шифрування та створення цифрових підписів включають перестановки та зсуви вихідного тесту, гамування (накладення одного тесту на інший), підстановки (заміна одних символів іншими) або їх комбінації. Класифікація криптографічних методів інформації включає більш деталізовані підкатегорії, які допомагають вирішувати різні задачі безпеки [48].

1.4.2 Протокол Діффі-Геллмана

Алгоритм Діффі-Геллмана є криптографічним протоколом, який дозволяє двом сторонам, що не мають попередніх загальних даних, безпечно обмінюватися секретними ключем. Ключ може використовуватися для подальшого симетричного шифрування. Обмін ключами з використанням еліптичних кривих може бути виконаний наступним чином [49-50]:

- загальний вибір параметрів;
- генерація закритих ключів;
- обчислення відкритих ключів;
- обчислення загального секретного ключа.

Отож, дві сторони (наприклад, Учасник 1 та Учасник 2) домовляються про ва числа: просте число p та примітивний корінь g (основа, яка зазвичай невелике ціле число), який менше за p. Учасник 1 обирає випадкове приватне число a (закритий ключ Учасника 1), яке він зберігає в таємниці. Учасник 2 виконує аналогічну дію для свого числа b. Учасник 1 обчислює своє публічне значення та надсилає його Учаснику 2. Учасник 2 виконує аналогічну дію. Обчислення публічного значення виконуються за наступними формулами відповідно до учасників [51-53]:

$$\{A = g^a \times mod\, p,\ B = g^b \times mod\, p. \tag{1.10}$$

Учасники, отримавши значення A та B один від одного, обчислюють загальні секрети s за наступними формулами [51-53]:

$$\{s = B^a \times mod\, p,\ s = A^b \times mod\, p. \tag{1.11}$$

Обидва результати будуть однаковими через наступну рівність [51-53]:

$$s = \left(g^b\right)^a \times mod\, p = \left(g^b\right)^a \times mod\, p. \tag{1.12}$$

Таким чином, обидва учасника мають однаковий секрет s, який може бути використаний для шифрування повідомлень. Безпека алгоритму заснована на складності обчислення дискретного логарифму. Якщо злочинець переходить значення A та B, йому буде складно відновити закриті ключі a або b без обчислень дискретного логарифму, що є дуже складною задачею для великих чисел. Даний алгоритм використовується в різних криптографічних протоколах для встановлення захищених з'єднань, таких як протоколи TLS/SSL для веббезпеки [51-53].

1.4.3 Схема Ель-Гамаля

Алгоритм Ель-Гамаля – криптографічна система з відкритим ключем, заснована на складності задачі дискретного логарифмування. Даний алгоритм використовується як для шифрування даних, так і для цифрового підпису. Основними етапами роботи алгоритму наступні [54-56]:

- генерація ключів;
- шифрування;
- дешифрування.

Схема Ель-Гамаля заснована на задачі зведення в ступінь. Алгоритм використовує наступні параметри [54-55]:

– p – велике просте число (модуль), яке публічно відомо;

– g – примітивний корінь за модулем p, також відомий;

– x – випадкове число, приватний ключ, обирається з діапазону $[1;\ p-2]$, параметри якого $(p,\ g,\ x)$;

– y – публічний ключ, в якому параметри $(p,\ g,\ y)$ його складають та виконується нерівність:

$$y = g^x \times mod\ p. \tag{1.13}$$

Складність відновлення закритого ключа за відкритим каналом пов'язана із задачею дискретного логарифмування. Припустимо, що злочинцеві відомий відкритий ключ $(p,\ g,\ y)$ за нерівністю (1.13). Задля відновлення x із цього виразу (1.13) необхідно обчислити дискретний логарифм [55]:

$$x = y \times mod\ p. \tag{1.14}$$

Дана задача за складністю ідентична до задачі факторизації. Не існує ефективних поліноміальних алгоритмів для обчислення числа x. Однак, існують субекспоненціальні алгоритми та алгоритми для квантового комп'ютера, які, можливо, здатні вирішити дану задачу [55].

Наступним етапом в алгоритмі Ель-Гамаля є шифрування. Для його здійснення необхідно обрати випадкове число k, що знаходиться в межах $[1;\ p-2]$ та обчислити числа a та b за наступною формулою [55-56]:

$$\{a = g^k \times mod\ p, \qquad b = y^k \times M \times mod\ p, \tag{1.15}$$

де M – повідомлення.

Числа a та b утворюють підпис. Для систем з відкритими ключами підпис повідомлення не передається за каналом зв'язку, так як її розмір великий. Попередньо обчислюється геш-функція h від повідомлення, а підпис – уже від неї [56]:

$$h = H(M). \tag{1.16}$$

Геш-функція має фіксовані розміри, за рахунок чого ЕЦП буде достатньо невеликого об'єму. Далі відбувається передача повідомлення та підпису обчисленої геш-функції. У випадку щ шифруванням обирається випадковий ключ – сесійний ключ S. Він зашифровується за схемою, а закритий текст отримується шляхом шифрування повідомлення M на обраному випадковому ключі. Важливим аспектом є той, що при кожному шифруванні отримується новий результат. Тобто, при генеруванні числа k, будуть різні числа a та b на виході [56].

При дешифруванні фактично відбувається зворотній процес. Воно відбувається за формулою [57-58]:

$$M = \frac{b}{a} \times mod\ p, \tag{1.17}$$

де M – символ повідомлення;

a, b – числа, обрані при шифруванні;

p – просте число.

Відновлення повідомлення без знання ключа відбувається таким чином, що злочинець знає відкритий ключ (p, g, y) та числа a та b, що були переданими за каналом зв'язку. Для відновлення необхідно знайти значення повідомлення та випадкове число k. При чому, для обчислення останнього необхідний дискретний логарифм. Формули для знаходження наступні [58]:

$$M = b \times y^{-k} \times mod\, p, \tag{1.18}$$

$$k = a \times mod\, p. \tag{1.19}$$

Схема Ель-Гамаля має високий рівень безпеки, але його використання пов’язано з вимогами до великих ключів для забезпечення захисту від сучасних обчислювальних атак.

1.4.4 Криптографічна програма PGP

Pretty Good Privacy (PGP) є програмним забезпеченням для щифрування та захисту даних. Основною метою є забезпечення безпеки передачі даних за незахищеними каналами зв’язку, таким як електронна пошта. Програма використовує метод гібридного шифрування, що поєднує симетричні та асиметричні криптографічні алгоритми, що забезпечує високий ступінь безпеки при передачі даних. PGP використовує принцип двох взаємопов’язаних ключів – відкритого та закритого. Відкритий ключ знаходиться у відкритому доступі та передається користувачам, з яким можна обмінюватися конфіденційною інформацією, а до закритого доступ є тільки у користувача [59-60].

Алгоритмом роботи програмного забезпечення PGP є [61]:

- виконання стиснення тексту, що збільшує швидкість та надійність шифрування;
- генерування сесійного ключа, який є випадковим числом;
- шифрування інформації за допомогою сесійного ключа;
- шифрування сесійного ключа за допомогою відкритого ключа;
- виконується передача повідомлення, яке містить зашифровану інформацію та зашифрований ключ;
- програма отримує зашифровану інформацію;

− виконується дешифрування сесійного ключа за допомогою закритого ключа;

− розшифровується інформація з використанням сесійного ключа.

Ключі, які використовуються в PGP зберігаються на комп’ютері користувача та при їх втраті він не зможе працювати. Перевагою даної програми є застосування геш-функції, яка аналізує змінення інформації. З її допомогою та закритого ключа інформація підписується ЕЦП, а при отриманні інформації отримувача PGP перевіряє підпис [62-63].

1.4.5 Алгоритм цифрового підпису

Алгоритм ЕЦП України заснований на криптографічних методах та керується відповідними законодавчими актами та законами. Шифрування виконується асиметрично та використовується пара ключів: закритий і відкритий [64-66].

В алгоритму є основні кроки [67-68]:

1) користувач генерує пару ключів: приватний (зберігається тільки у користувача) та публічний, зазвичай, за допомогою криптографічних засобів, таких як державні стандарти або інші документи;

2) перед підписанням даних (документа або повідомлення), інформація проходить через геш-функцію, щоб отримати унікальне цифрове представлення даних (геш-суму), що допомагає захистити дані та перевіряє їх цілісність;

3) отриманий геш шифрується з використанням закритого ключа користувача (ЕЦП) та завдяки асиметричності криптографії, тільки цей закритий ключ може створювати такий підпис, а перевірка можлива за допомогою відкритого ключа;

4) підписані дані (документ та підпис) передаються отримувачу; підпис може передаватися як у складі самого документу, так і окремо від нього;

5) підпис вважається дійсним, якщо результат збігається з розрахованим геш-кодом;

6) для перевірки автентичності ключа використовується сертифікат, що виданий акредитованим центром сертифікації ключів, який підтверджує відповідно публічного ключа даному користувачеві.

Схематично алгоритм наведений на рисунку 1.3.

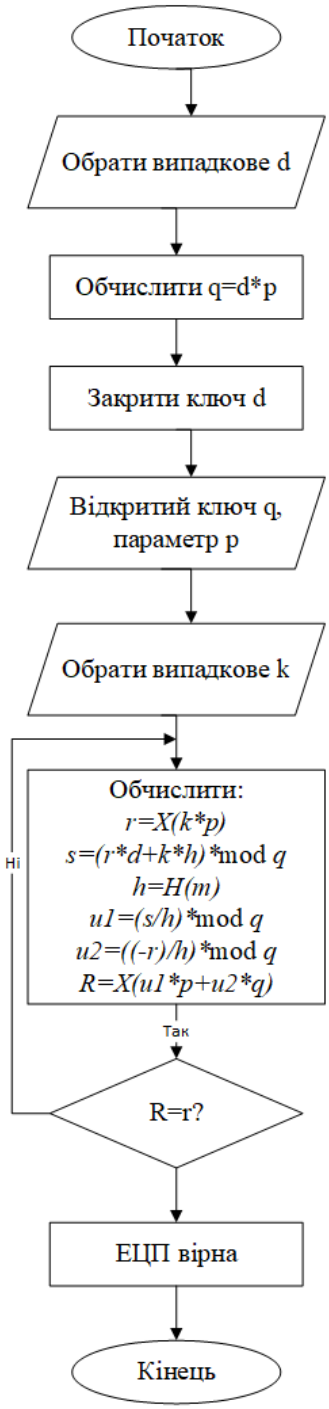

Рисунок 1.3 – Схема алгоритму електронного цифрового підпису [67-68]

ЕЦП активно використовується в електронному документообігу, електронних державних послугах, банківській системі та інших сферах (див. табл. 1.3). Цифрові підписи мають багато переваг, які спрощують певні процеси життєдіяльності та обміну інформації.

1.4.6 Алгоритм Чандрасекхара

Алгоритм Чандрасекхара є асиметричним методом шифрування, що використовує еліптичні криві для забезпечення конфіденційності інформації. Спочатку обирається рівняння $E(a, b)$ еліптичної кривої (1.1) над кінцевим полем F_p, де p – просте число. Необхідно впевнитися, що крива була невирадженою за нерівністю [69-70]:

$$4a^3 + 27b^2 \neq 0. \tag{1.20}$$

Далі відбувається генерація точок та таблиці символів. Знаходяться всі точки, що належать кривій і складається таблиця N, в якій до кожної точки присвоюється символ. Дана таблиця відома тільки тим учасникам, які обмінюються інформацією. Учасники (наприклад, Учасник 1 та Учасник 2) обирають випадкові числа a і b, точки A і B, що належать до $E(a, b)$ та обчислюють відкриті ключі за формулами [70-71]:

$$\{A_1 = a \times C + A,\ A_2 = A \times a^2; \tag{1.21}$$

$$\{B_1 = b \times C + B,\ B_2 = B \times b^2. \tag{1.22}$$

Далі обидва учасники обчислюють загальні ключі [70-71]:

$$A_b = A_2 \times a; \tag{1.23}$$

$$B_a = B_2 \times b. \tag{1.24}$$

Для шифрування інформації Учасника 2 кожен символ повідомлення M кодується за такими діями [72-73]:

1) обирається випадкове число γ, яке менше за просте число p;

2) обчислюється перше значення зашифрованого повідомлення за формулою:

$$E_1 = \gamma \times C_1, \tag{1.25}$$

де C_1 – заздалегідь відома всім учасникам точка на еліптичній кривій;

3) обчислюється друге значення зашифрованого повідомлення за формулою:

$$E_2 = M + B_2 + A_1 + A_2 - \gamma, \tag{1.26}$$

де M – символ повідомлення, перетворений в точку на еліптичній кривій за допомогою таблиці N.

Пара (E_1, E_2) представляє зашифроване повідомлення.

Дешифрування інформації Учасником 1 відбувається за такими діями [72-73]:

1) Учасник 1 обчислює загальний ключ за допомогою формули:

$$Ba = a \times B_2, \tag{1.27}$$

де a – закритий ключ Учасника 1.

2) Для кожного отриманого символу E_2, Учасник 1 обчислює вихідне повідомлення M за наступною формулою:

$$M = E_2 - a \times E_1 + A_1 + B_a, \tag{1.28}$$

де $a \times E_1$ – операція, що дозволяє врахувати випадкове число γ.

Далі отримана точка M перетворюється зворотно в символ із використанням таблиці N.

Етапи алгоритму наведені на рисунку 1.4.

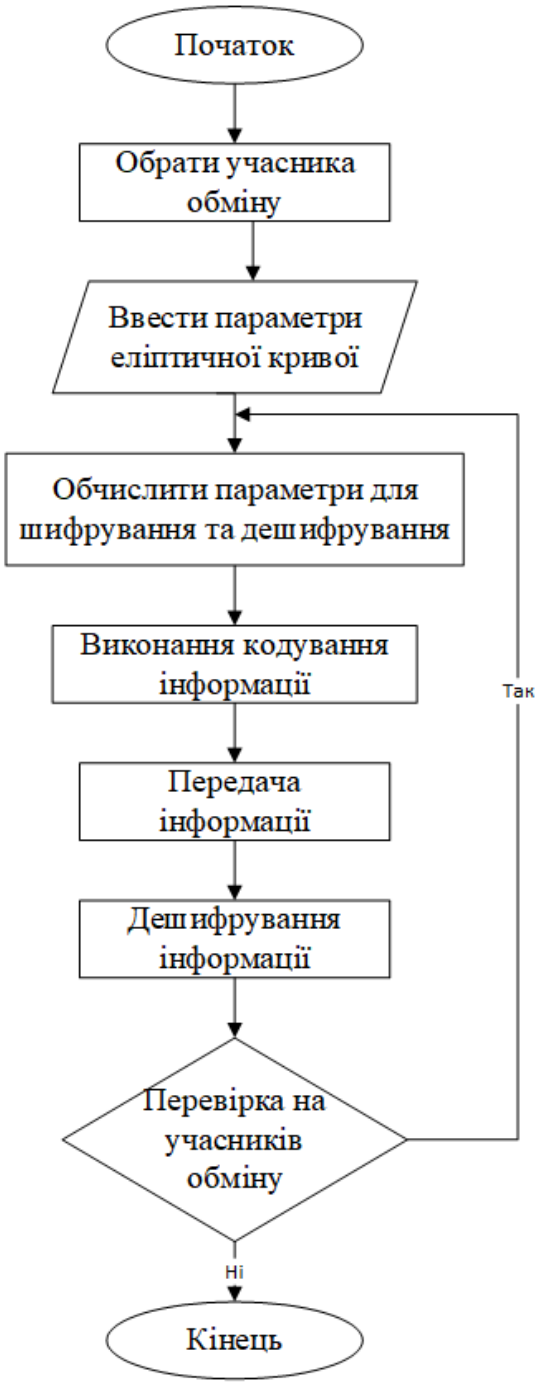

Рисунок 1.4 – Схема алгоритму Чандрасекхара [73]

Безпека шифрування на основі еліптичних кривих залежить від значення p (порядку кінцевого поля) та випадково обраної точки C на еліптичній кривій. Елаптичні параметри повинні бути ретельно підібрані для забезпечення захисту від атак.

1.4.7 Дискретне логарифмування на еліптичних кривих

ECDLP є однією з фундаментальних проблем у криптографії на еліптичних кривих, від вирішення якої залежить безпека багатьох криптографічних систем. Нехай E – еліптична крива, що визначена кінцевим полем F_p. На цій кривій задана точка P (генератор групи точок) порядку n та деяка точка Q, що належить групі точок кривій. Задача дискретного логарифму полягає в знаходженні цілого числа k (логарифму), такого що виражається за формулою (1.7). Основна складність ECDLP полягає в тому, що для еліптичних кривих не існує ефективних алгоритмів для обчислення дискретного логарифму, як це можливо для інших груп. Це робить криптографічні системи на еліптичних кривих стійкими до атак методом перебору або ділення на невеликі множники [74-76].

Існує багато методів вирішення задач дискретного логарифмування. Основні наведені в таблиці 1.5.

Таблиця 1.5 – Методи вирішення дискретного логарифмування на еліптичних кривих

Метод	Опис методу
1	2
Метод повного перебору	Простий спосіб вирішення – перебір всіх можливих значень k поки не знайдеться значення, для якого виконується нерівність (1.7). Цей метод неефективний, так як складність лінійна $O(n)$

Продовження таблиці 1.5

1	2
«Крок малюка крок велетня»	Даний метод використовує принцип розбиття логарифма на два етапи: виконання малих кроків і великих кроків. Такий спосіб скорочує складність задачі до $O(\sqrt{n})$, однак потребує значних об’ємів пам’яті

Алгоритм Полларда	Стохастичний метод, який має складність $O(\sqrt{n})$ та ємнісною складністю $O(\sqrt{\log log q})$ та потребує менший об'єм пам'яті в порівнянні з попереднім методом. Принцип заснований на використанні імовірнісних обчислень та випадкових ходів
Метод Кангару Полларда	Метод оптимізований для випадку, коли відомо, що дискретний логарифм знаходиться в деякому відомому діапазоні. Складність вирішення $O(\sqrt{n})$
Алгоритми на базі решіткових методів	Включає підходи Лагаріаса та Оделла та інші методи, що використовують решіткові структури для скорочення складності дискретного логарифмування в залежності від визначених властивостей групи еліптичної кривої
Алгоритм зустрічі на випадковому дереві	Універсальний алгоритм, що має однакову часову та ємнісну складність $O(\sqrt{q \times \log log q})$, але припускає розпаралелювання для довільного числа паралельно працюючих процесів

За таблицею 1.5 можна побачити основні методи вирішення задачі дискретного логарифмування на еліптичних кривих. Однак, існують проблеми, які можуть виникнути при вирішенні. Вони поділяються на такі категорії, як обчислювальної складності, уразливості криптографічних систем, а також пов'язані з вибором еліптичної кривої та характеристиками використовуваного поля. Наприклад, експоненціальна складність полягає в тому, що незважаючи на існування таких алгоритмів, як метод Полларда або «Крок малюка – крок велетня», обчислення дискретного логарифму на еліптичних кривих залишається задачею з експоненціальною складністю. Тобто, складність задачі збільшується, а ресурси, необхідні для вирішення стають дуже великими. Вибір суперсингулярних кривих є поганим рішенням через уразливість до атак з використанням алгоритмів, що засновані на застосуванні кінцевих полів та перетворенні задачі дискретного логарифму на еліптичній кривій у задачу дискретного логарифму в більш просту групу. Також поганим вибором є криві з низьким порядком, тому що тоді можна вирішити задачу ECDLP більш ефективними методами. Додатковою проблемою може бути оптимізація алгоритмів в швидкість їх обчислення. Вони обумовлені складними обчисленнями

складання точок та скалярного помноження, а також потреба в оптимізації задач, таких як редукція за модулем тощо. Іншими проблемами є атаки бакових каналів, на конкретні реалізації, перехід до квантової криптографії, уразливість через структуру поля або нерівномірне розподілення точок тощо [76-78].

Як можна побачити, задача дискретного логарифмування на еліптичних кривих є основою криптографічної стійкості сучасних систем, але вона має ряд недоліків і проблем, пов’язаних як з обчислювальними складнощами, так і з потенційними вразливостями, включаючи квантові атаки та вибір невірних параметрів еліптичної кривої.

Висновки до розділу 1

У ході аналізу сучасних методів шифрування даних з використанням еліптичних кривих було встановлено, що це перспективне направлення, яке забезпечує високий ступінь безпеки при відносно малих розмірів ключів. Такий аспект робить ECC ефективною для використання в умовах обмежених обчислювальних ресурсів, таких як мобільні пристрої та IoT. Математичні основи грають ключову роль у безпеці криптографічних алгоритмів. Таким чином, шифрування даних з використанням ECC є важливим елементом сучасних систем, що забезпечує високий рівень безпеки в різних сферах застосування.

РОЗДІЛ 2 ЗАГАЛЬНІ МЕТОДИ ПРОВЕДЕННЯ НАУКОВОГО ДОСЛІДЖЕННЯ

2.1 Обґрунтування вибору напряму дослідження

Із зростанням об’ємів пам’яті та зростанням числа кібератак необхідність забезпечення безпеки інформації стає критично важливою для сучасних систем. Одним із найбільш ефективних направлень у криптографії є використання еліптичних кривих. Вони забезпечують високий рівень захисту при відносно малому розмірі ключів і високої продуктивності. У рамках даного дослідження обрані два алгоритми, заснованих на еліптичних кривих.

Алгоритм Чандрасекхара є асиметричним алгоритмом шифрування, який комбінує можливості еліптичних кривих та методи шифрування на основі публічних і приватних ключів. Схема роботи алгоритму Чандрасекхара наведена на рисунку 2.1 [79].

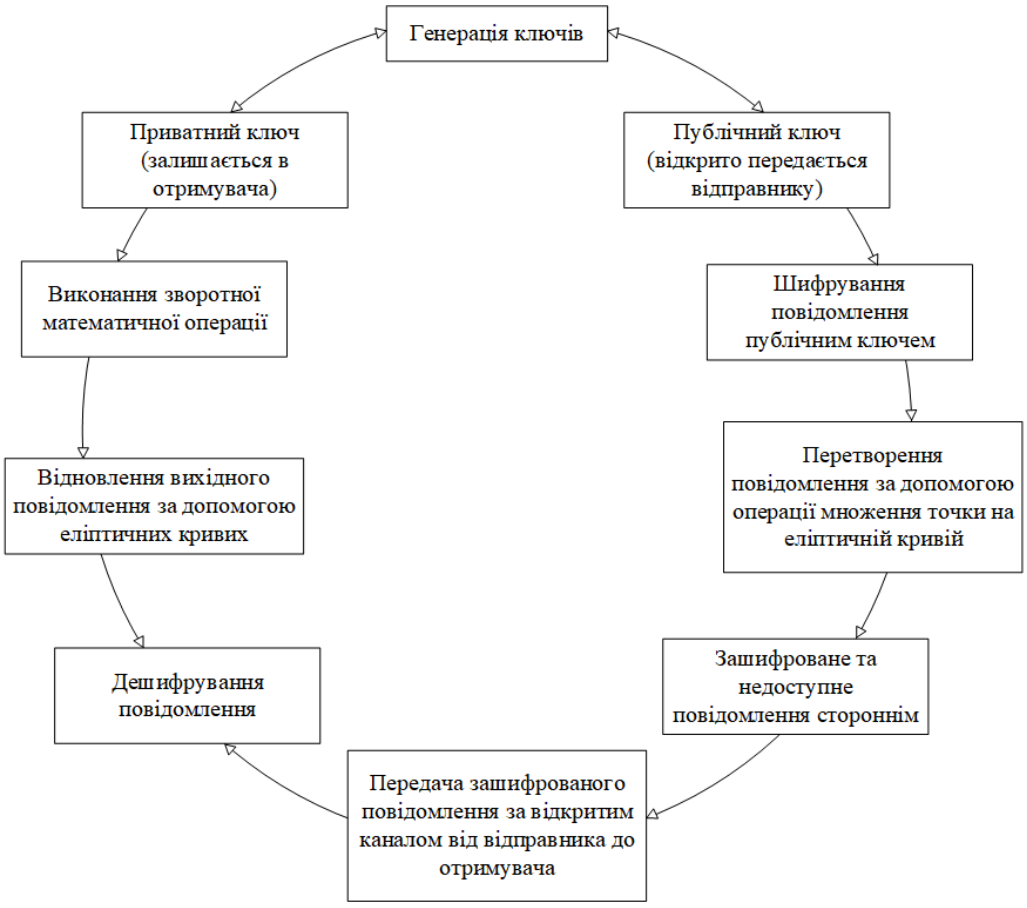

Рисунок 2.1 – Схема роботи алгоритму Чандрасекхара

Алгоритм Чандрасекхара здатний забезпечити високий ступінь безпеки за рахунок складності обчислення дискретного логарифму на еліптичних кривих. Однією з особливостей алгоритму є можливість обирати параметри кривої, що дозволяє адаптувати його під різні умови застосування, забезпечуючи баланс між швидкістю виконання операцій і рівнем безпеки [79-80].

Протокол Діффі-Геллмана є класичним механізмом обміну ключами, що забезпечує безпечну передачу криптографічного матеріалу між двома сторонами за відкритим каналом. Використання еліптичних кривих у даному протоколі робить процес більш ефективним, аніж при застосування традиційних криптографічних методів. Схема роботи алгоритму Діффі-Геллмана наведена на рисунку 2.2 [79-80].

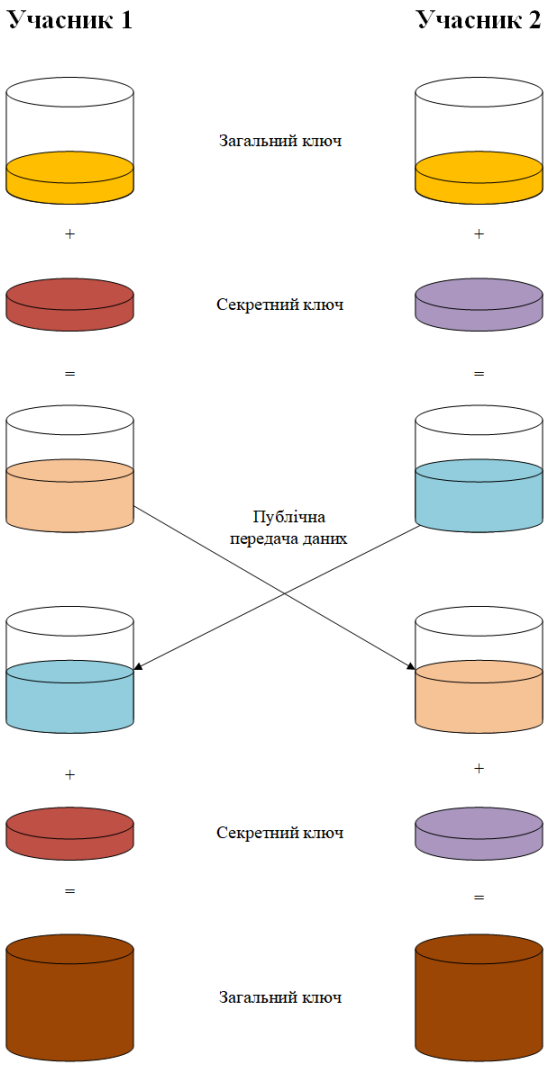

Рисунок 2.2 – Схема роботи алгоритму протоколу Діффі-Геллмана

ECDH отримав широке розповсюдження в сучасних стандартах безпеки, включаючи протоколи SSL/TLS, що підкреслює його практичну значимість і необхідність дослідження [80].

Обрані алгоритми для дослідження мають високу практичну цінність, оскільки вони можуть бути використані в різних областях: від захисту даних в Інтернет-комунікаціях до безпечного обміну даними IoT-пристроїв. Проведення порівняльного аналізу алгоритмів допоможе визначити найбільш ефективне рішення для конкретних задач шифрування. Таким чином, тема дослідження актуальна як з наукової, так і з практичної точок зору, а вибір алгоритмів дозволяє провести

комплексне дослідження сучасних методів шифрування даних на основі еліптичних кривих.

2.2 Вибір метрик оцінки ефективності

2.2.1 Криптографічна стійкість

Криптографічна стійкість алгоритмів на еліптичних кривих напряму пов’язана зі складністю задачі обчислення дискретного логарифму на еліптичних кривих. Чим складніше знаходження дискретного логарифму, тим вища стійкість. Для знаходження криптографічної стійкості необхідно використати формулу [81]:

$$n \approx 2^{\frac{k}{2}}, \tag{2.1}$$

де n – кількість операцій, необхідних для взлому;

k – довжина ключа (у бітах).

Для еліптичних кривих, в яких k дорівнює 256 біт забезпечує стійкість, аналогічну RSA-2048. Для обраних алгоритмів стійкість повинна бути аналогічною при однаковій довжині ключа [81].

2.2.2 Час шифрування та дешифрування

Час виконання операції шифрування та дешифрування, зазвичай, залежить від числа точок на еліптичній кривій та складності їх арифметики. Для їх обчислення необхідно використовувати формулу [81]:

$$T_{enc/dec} = T_{point\_mult} \times N, \tag{2.2}$$

де $T_{enc/dec}$ – загальний час шифрування або дешифрування;

T_{point_mult} – час однієї операції множення точки на кривій;

N – кількість операцій на кривій (зазвичай множення).

Помноження точки на еліптичній кривій вважається основною витратною операцією. В алгоритмі Чандрасекхара можуть використовуватися додаткові операції, що можуть збільшити загальний час [81].

2.2.3 Час генерування ключів

Процес генерації ключів в алгоритмах на еліптичних кривих надає вибір випадкового числа d та обчислення відкритого ключа як [82]:

$$Q = d \times G, \tag{2.3}$$

де G – базова точка кривої;

Q – відкритий ключ.

Час генерації ключів знаходиться за наступною формулою [82]:

$$T_{key\_gen} = T_{point\_mult}. \tag{2.4}$$

Час однієї операції T_{point_mult} залежить від потужності обладнання та характеристик кривої. Для обраних алгоритмів цей час буде приблизно однаковим при схожих умовах [82].

2.2.4 Енергоефективність

Енергоефективність алгоритму пов’язана з кількістю операцій, виконуваних пристроєм і споживаною енергією за одиницю часу. Енергоефективність можна виразити наступним чином [83]:

$$E = P \times T, \tag{2.5}$$

де E – загальне енергоспоживання;

P – споживана потужність процесору під час обчислень;

T – час виконання криптографічної операції.

Для оцінки ефективності алгоритмів часто використовується енергія на біт, що виражається таким чином [83]:

$$EpB = \frac{E}{k}, \tag{2.6}$$

де k – довжина шифрованого повідомлення.

Алгоритми на еліптичних кривих часто потребують менше енергії в порівнянні з RSA та іншими алгоритмами з довгими ключами [83].

2.2.5 Витрати на зберігання та передачу даних

Дана метрика показує, які кількість даних необхідна для зберігання ключів і виконання обміну в мережі. Еліптичні криві дозволяють використовувати більш короткі ключі, що зменшують об’єм даних, необхідних для зберігання та передачі. Об’єм пам’яті, необхідний для зберігання ключа, можна виразити наступним чином [83]:

$$S_{key} = k_{key} \times n_{key}, \tag{2.7}$$

де S_{key} – об’єм пам’яті для зберігання ключів;

k_{key} – довжина одного ключа (у бітах);

n_{key} – кількість ключів, які необхідні зберігати.

Для алгоритмів на еліптичних кривих значення k_{key} зазвичай менше, аніж у інших алгоритмів із аналогічною стійкістю, що знижує витрати на зберігання та передачу даних [83].

2.3 Проведення експериментів та реалізація тестових сценаріїв

Для проведення експериментів для кожного з алгоритмів можна спиратися на вищеописані метрики, за якими й виконаються тестові сценарії. Описані експерименти наведені в таблиці 2.1.

Таблиця 2.1 – Експерименти та тестові сценарії алгоритмів шифрування даних

Експеримент	Опис для алгоритму Чандрасекхара	Опис для протоколу Діффі-Геллмана
Оцінка криптографічної стійкості	Запуск атаки для підбору ключа, фіксуючи число спроб та час, витрачений на злом	Запуск Man-In-The-Middle attack (MITM-атака)
Вимір часу шифрування та дешифрування або обміну ключами	Вимір часу, витраченого на шифрування та дешифрування різних об’ємів даних	Вимір часу, необхідний для завершення обміну ключами
Зміна часу генерації ключів	Запуск генерації ключів для різних розмірів ключів	Запуск генерації ключів для різних розмірів ключів
Оцінка енергоефективності	Запис споживаної енергії під час виконання операції шифрування та дешифрування	Запуск протоколу на тестовому пристрої та запис споживаної енергії під час обміну ключами
Зміна витрат на зберігання	Вимір об’єму пам’яті, що займає ключами та проміжним результатом	Вимір об’єму пам’яті, що займає ключами та проміжним результатом
Оцінка витрат на передачу даних	Вимір об’єму даних, що передаються при шифруванні	Запуск обміну ключами

Описані експерименти тестових сценаріїв дозволять провести порівняльну оцінку та аналіз алгоритмів Чандрасекхара та протоколу Діффі-Геллмана за метриками ефективності та безпеки.

2.4 Підтримка працездатності алгоритмів

Підтримка працездатності алгоритмів включає декілька аспектів, які забезпечують їх ефективну функціонування та безпеку. У таблиці 2.2 наведені всі ключові моменти для алгоритму Чандрасекхара.

Таблиця 2.2 – Опис аспектів підтримки працездатності алгоритму Чандрасекхара

Аспект працездатності	Опис аспекту
1	2
Ключова інфраструктура	
Генерація ключів	Використання надійних генераторів випадкових чисел для створення ключів. Вибір точки на еліптичній кривій повинен забезпечувати максимальну стійкість до атак
Кодування ключів	Для передачі ключів необхідно використовувати методи кодування, такі як Base64 або Hex для забезпечення сумісності з різними системами
Алгоритмічна стійкість	
Складність обчислень	Стійкість алгоритму залежить від складності математичних задач, таких як проблема логарифму на еліптичних кривих, що робить його важким для рішення за допомогою грубої сили
Захист від атак	Регулярні оновлення математичних моделей та використання нових методів шифрування для підвищення стійкості до атак

Продовження таблиці 2.2

1	2
Енергоефективність	
Оптимізація обчислень	Використання ефективності алгоритмів для операцій з еліптичними кривими, таких як алгоритми подвійного та потрійного скалярного помноження
Мобільні та вбудовані системи	Розробка версій алгоритму, адаптованих для використання в системах з обмеженими ресурсами
Підтримка стандартів	
Дотримання стандартів	Сертифікації алгоритму у відповідності з міжнародними стандартами для генерації випадкових чисел та керуванням ключами
Тестування та верифікація	
Тестування на стійкість	Використання тестів для оцінки криптографічної стійкості, таких як тести на випадковість та навантажувальні тести
Аудит безпеки	Проведення регулярних аудитів коду та архітектури для виявлення вразливостей

У таблиці 2.3 наведені всі ключові моменти для алгоритму Діффі-Геллмана. Вони включають у себе такі аспекти як:

- ключова інфраструктура;
- алгоритмічна стійкість;
- енергоефективність;
- підтримка стандартів;
- тестування.

Таблиця 2.3 – Опис аспектів підтримки працездатності протоколу Діффі-Геллмана

Аспект працездатності	Опис аспекту
Ключова інфраструктура	
Генерація ключів	Генерація великих простих чисел та відповідних примітивних коренів, які забезпечують безпеку обміну
Обмін ключами	Забезпечення безпеки передачі відкритих ключів між учасниками для запобігання атак типу MITM
Алгоритмічна стійкість	
Складність обчислень	Протокол заснований на складності дискретного логарифмування, що робить його стійким до атак з використанням алгоритмів грубої сили
Захист від атак	Використання додаткових методів, таких як автентифікація учасників для захисту від атак MITM
Енергоефективність	
Оптимізація обчислень	Скорочення числа операцій при виконанні протоколу, щоб мінімізувати час відгуку, особливо в мобільних пристроїв
Адаптація до ресурсів	Розробка алгоритмів, які можуть ефективно працювати на пристроях з обмеженою обчислювальною потужністю
Підтримка стандартів	
Дотримання стандартів	Участь у стандартизації протоколу через організації для забезпечення його сумісності та безпеки
Тестування та верифікація	
Тестування на стійкість	Проведення навантажувального тестування для визначення меж продуктивності протоколу
Аудит безпеки	Аналіз уразливостей у реальних умовах експлуатації

Такі міри можуть підтримати працездатність і безпеку як алгоритму, так і протоколу, що вкрай важливо для ефективного використання криптографічних систем.

Висновки до розділу 2

За розділом були розглянуті аспекти проведення наукового дослідження в сфері криптографії та шифрування даних. Було обґрунтовано актуальність обраної теми та алгоритмів, яка підкреслює важливість безпеки інформації в сучасному світі. Визначені

метрики для оцінки ефективності алгоритмів шифрування ECC. Вони дозвлять оцінити не лише безпечність, а й швидкість алгоритмів у реальних умовах. Опис експериментів і тестових сценаріїв дозволив отримати емпіричні дані, які в подальшому нададуть відповідні результати для відповідних висновків. Таким чином, дане дослідження надає фундамент для подальшої розробки алгоритму Чандрасекхара та протоколу Діффі-Геллмана.

РОЗДІЛ 3 РОЗРОБКА ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ДЛЯ ДОСЛІДЖЕННЯ АЛГОРИТМІВ ШИФРУВАННЯ ДАНИХ З ВИКОРИСТАННЯМ ЕЛІПТИЧНИХ КРИВИХ

3.1 Проєктування архітектури системи

3.1.1 Обґрунтування інструментарію розробки

Для роботи з криптографічними алгоритмами шифрування та дешифрування необхідно використовувати певні бібліотеки. Їх опис наведений у таблиці 3.1.

Таблиця 3.1 – Бібліотеки Python для роботи з криптографічними алгоритмами

Назва бібліотеки	Обґрунтування використання бібліотеки
1	2
os	Робота з операційною системою, включаючи генерацію випадкових чисел (у даному випадку, створення ключів). Надання функціоналу для роботи з файловою системою, наприклад, із зберіганням і завантаженням ключів або конфігурації
ecdsa	Надання інструментів для роботи з алгоритмами, заснованих на еліптичних кривих. Вона містить функції для генерації ключів, підпису та перевірки ключів, що робить її ключовим компонентом при реалізації шифрування на еліптичних кривих
hashlib	Використовується для створення гешів, що важливо для перевірки цілісності даних і створення міток повідомлень у процесі шифрування та підпису. Геш-функції також використовуються для формування ключів із паролів або інших секретів

Продовження таблиці 3.1

1	2
Cryptodome. Cipher	Реалізація різних симетричних шифрів, що може використовуватися для шифрування даних після обміну через алгоритми на еліптичних кривих

Cryptodome.Util.Padding	Надає утиліти для додавання та видалення доповнень до даних перед шифруванням та після дешифрування. Забезпечує коректну обробку даних у блочних шифрах, довжина яких не кратна розміру блоку
pycryptodome	Реалізація різних криптографічних алгоритмів, як симетричних, так і асиметричних. Використання бібліотеки для генерації ключів, шифрування та дешифрування даних і створення цифрових підписів. Містить підтримку алгоритмів на еліптичних кривих, що дозволяє інтегрувати гібридні схеми шифрування

Використання мови програмування Python та обраних бібліотек надають потужний інструментарій для розробки алгоритмів шифрування та еліптичних кривих. Вони надають усе необхідне для забезпечення безпеки даних, перевірки їх цілісності та ефективної роботи з криптографією.

3.1.2 Технологічна платформа

В якості мови програмування був обраний Python. Він має ряд наступним переваг:

- швидке прототипування та розробка складних алгоритмів, таких як шифрування на еліптичних кривих;
- спрощення реалізації криптографічних алгоритмів за допомогою використання широкої підтримки бібліотек;
- розробка та запуск криптографічних алгоритмів на різних операційних системах без необхідності внесення змін у код;
- множина ресурсів і документації, що спрощує розробку різного роду алгоритмів.

3.1.3 Програмна реалізація алгоритму Чандрасекхара

Алгоритм Чандрасекхара було розроблено в поєднанні алгоритму на основі еліптичних кривих для генерації та обміну ключів і симетричного шифрування AES для

захисту повідомлень. Такий підхід забезпечує високий рівень безпеки через ряд причин:

– еліптичні криві використовуються для створення пари відкритого (для шифрування) та закритого (для дешифрування) ключів;

– симетричне шифрування AES застосовується для захисту повідомлення після створення симетричного ключа, отриманого із загальної точки на еліптичній кривій, який отримано з геш-функції SHA-256, використовується для шифрування та дешифрування повідомлень, що дозволяє зберігати конфіденційність даних;

– закритий ключ створюється випадковим чином, а на його основі обчислюється відкритий, тому для кожного сеансу шифрування використовується унікальний тимчасовий ключ, який запобігає повторні атаки.

Фрагмент коду наведений на рисунку 3.1.

```
# Генерація ключів
def generate_keys():  1 usage
    private_key = int.from_bytes(os.urandom(32),  byteorder: 'big') % field_order
    public_key = private_key * G
    return private_key, public_key

# Деривація ключа
def key_derivation(secret_point):  2 usages
    return sha256(str(secret_point).encode()).digest()
```

Рисунок 3.1 – Фрагмент програмного коду алгоритму Чандрасекхара

Повна реалізація алгоритму наведена в додатку В.

В алгоритмі класи відсутні, бо вони не відіграють важливої ролі в розробці, але є функції для його реалізації які описані в таблиці 3.2.

Таблиця 3.2 – Функції програмної реалізації алгоритму Чандрасекхара

Функція	Опис	Параметри	Значення, що повертаються
1	2	3	4

get_curve_ parameters()	Збирає параметри a і b для еліптичної кривої від користувача, перевіряючи, чи є крива виродженою	–	Кортеж (a, b), які є коефіцієнтами еліптичної кривої
generate_keys()	Генерує закритий і відкритий ключі	–	Кортеж закритого та відкритого ключів (private_key, public_key)
key_derivation (secret_point)	Деривація ключа на основі загальної точки, використовуючи SHA-256	Точка на еліптичній кривій secret_point	Симетричний ключ (байтовий рядок)

Продовження таблиці 3.2

1	2	3	4
encrypt_message_aes (message, derived_key)	Шифрує повідомлення за допомогою AES	Повідомлення для шифрування message. Симетричний ключ derived_key	Зашифроване повідомлення (байтовий рядок), що містить IV
decrypt_message_aes (encrypted_message, derived_key)	Розшифровує повідомлення, зашифроване AES	Зашифроване повідомлення encrypted_message. Симетричний ключ derived_key	Дешифроване повідомлення (рядок)
encrypt(message, public_key)	Основна функція для шифрування повідомлення, використовуючи відкритий ключ	Повідомлення для шифрування message. Відкритий ключ public_key	Кортеж зашифрованого повідомлення та тимчасовий рядок (encrypted_message, temp_point)

decrypt (encrypted_message, private_key, temp_point)	Основна функція для дешифрування повідомлення	Зашифроване повідомлення encrypted_message. Закритий ключ private_key. Тимчасова точка temp_point	Дешифроване повідомлення (рядок)

Взаємодію функцій можна зобразити на діаграмі взаємодій функцій у вигляді діаграми послідовності, яка наведена на рисунку 3.2.

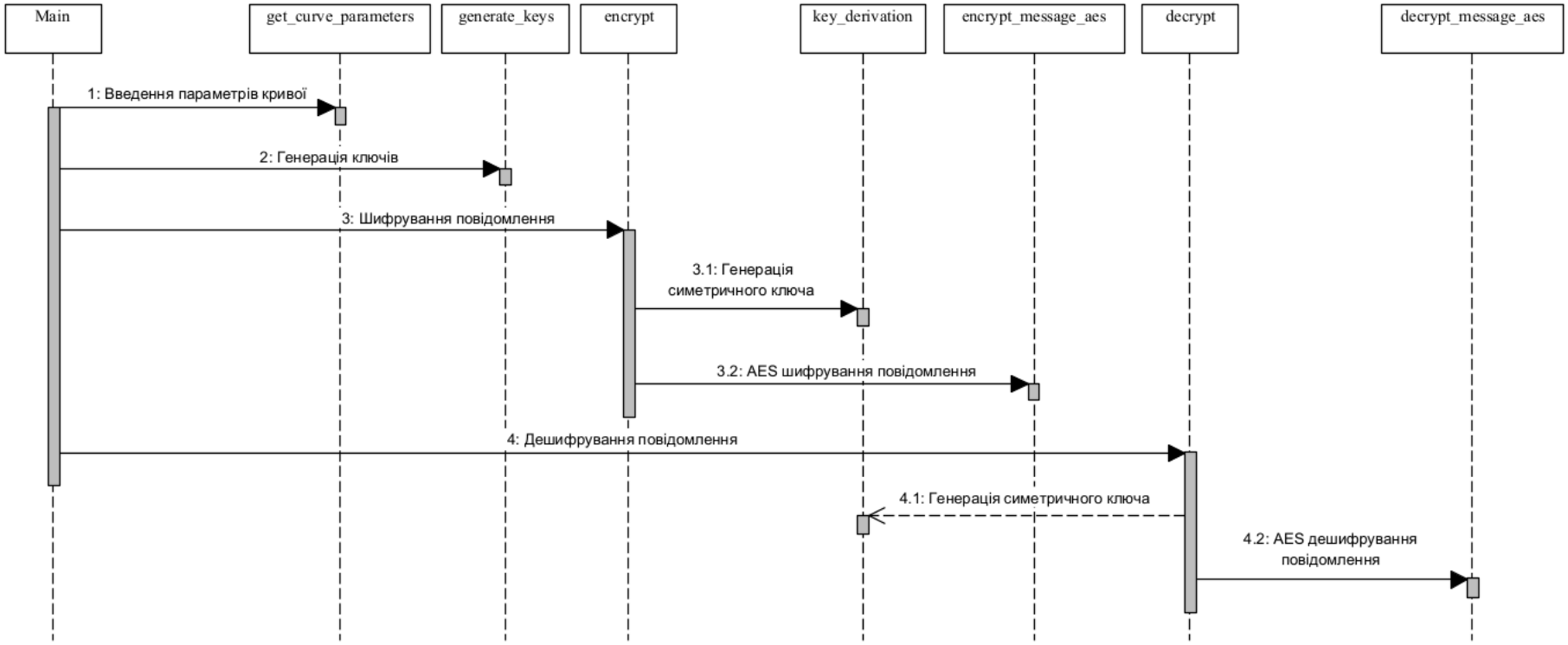

Рисунок 3.2 – Діаграма взаємодій функцій алгоритму Чандресекхара у вигляді діаграми послідовності

За рисунком 3.2 можна наочно побачити та зрозуміти всі процеси взаємодії між функціями програмного коду під час його виконання. Спочатку виконується основний процес, який викликає всі функції, далі виконуються такі функції:

1) отримання параметрів еліптичної кривої;
2) генерація ключів;
3) виконання шифрування, яке викликає дві додаткові функції:
   - отримання симетричного ключа;
   - шифрування повідомлення з використанням AES;
4) виконання дешифрування, яке викликає:
   - отримання симетричного ключа;
   - дешифрування повідомлення з використанням AES.

Основними етапами алгоритму Чандрасекхара за описаними функціями є наступні:

- отримання параметрів кривої шляхом введення користувачем коефіцієнтів a і b, що визначають еліптичну криву;
- виконання перевірки вродженості кривої;
- виконання генерації відкритого та закритого ключів (закритий створюється випадковим чином, відкритий – перемноження закритого ключа та генератору кривої);
- деривація ключа шляхом виконання гешування для створення симетричного ключа на основі загальної точки, отриманої під час шифрування;
- шифрування повідомлення за допомогою симетричного шифрування AES;
- дешифрування повідомлення за допомогою симетричного шифрування AES;
- виводяться всі необхідні дані в результати середовища розробки.

Наочно етапи можна зобразити на діаграмі послідовності. Вона наведена на рисунку 3.3. Результат виконаного алгоритму наведений на рисунку 3.4.

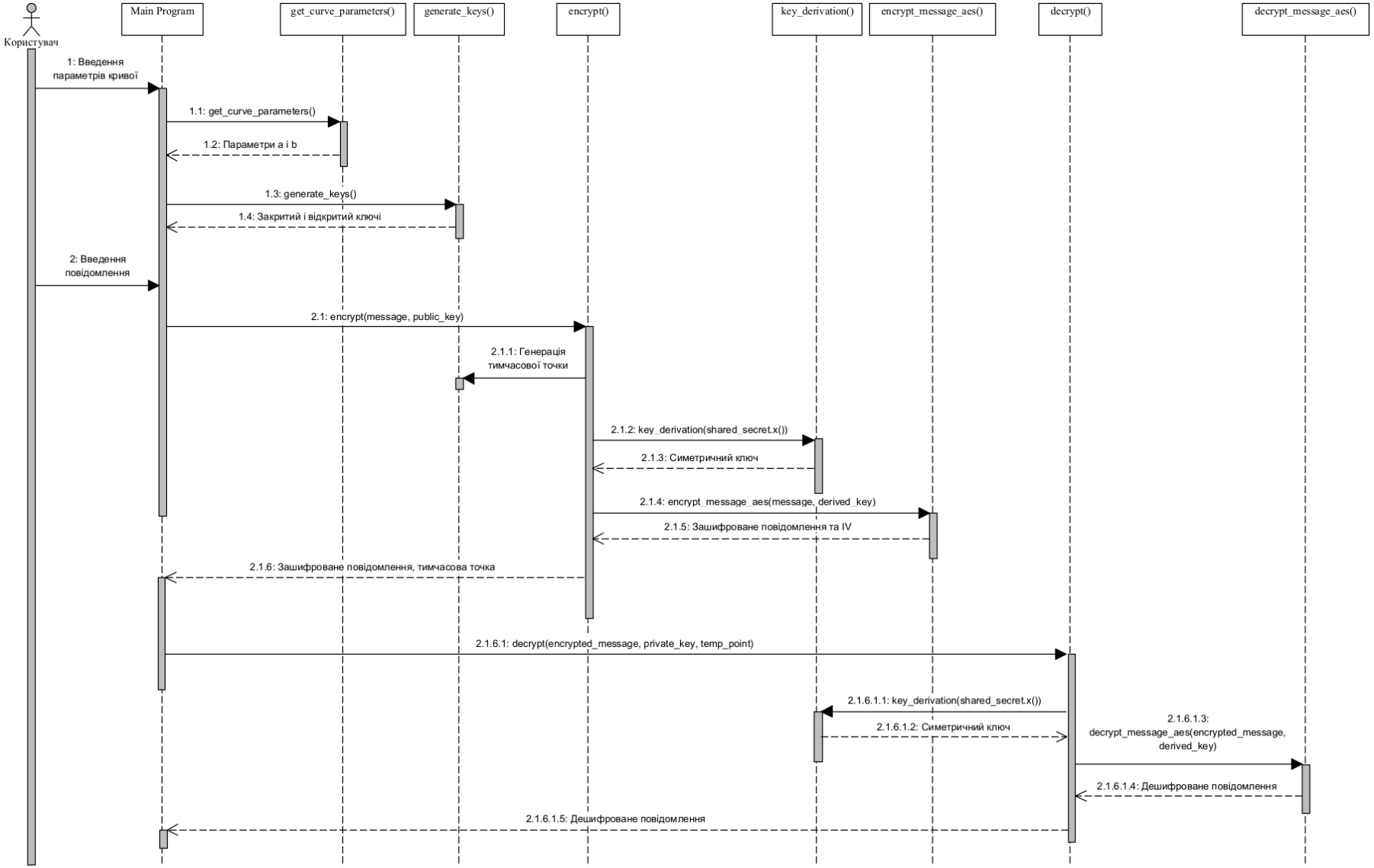

Рисунок 3.3 – Діаграма послідовності алгоритму Чандрасекхара

```
Введіть коефіцієнт a (ціле число): 3
Введіть коефіцієнт b (ціле число): -5
Закритий ключ: 113813900783784369012664335044159068759945967132375863260388908454217994742647
Відкритий ключ: (270148686888453477911354631631693754933823971918096146571707854543473535469 04, 97344660755182856281679896742169762553560492897180022133135089956122907333714)
Введіть повідомлення для шифрування: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt at labore et dolore magna aliqua…
Зашифроване повідомлення: b'\x81\xc8\xf9\x11\x9a\x9b\xa2\x84(\xcf\xdc\xb42\xa3\x0f\xa5S\xad\x1a\xeb\xf7\xe0J\xd1\xf4Si\xbf\xc5F\x16\xf6\x81[\x08a\x8b\xff\x16\xe9S\x19\xdch\xa2'
Дешифроване повідомлення: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt at labore et dolore magna aliqua…
```

Рисунок 3.4 – Результат виконання алгоритму Чандрасекхара

Для реалізації алгоритму Чандрасекхара було використано гібридний спосіб з двома алгоритмами, які підходять для як потужних систем, так і малопотужних. Вони забезпечують високу ефективність безпеки даних за рахунок еліптичних кривих і AES.

3.1.4 Програмна реалізація протоколу Діффі-Геллмана

Алгоритм Діффі-Геллмана заснований на еліптичних кривих та використовується для створення спільного секретного ключа між двома сторонами:

- приватний ключ ніколи не передається та залишається секретним;
- публічний ключ не надає змогу обчислити приватний завдяки складності задачі дискретного логарифму на еліптичних кривих;
- для кожного сеансу шифрування генерується унікальний спільний секрет, який захищає від атак повтору;
- еліптичні криві дозволяють досягти тієї ж безпеки, що й традиційний алгоритм, але з коротшими ключами.

Фрагмент коду наведений на рисунку 3.5.

```
def __add__(self, other):
    # Додавання двох точок на кривій
    if self.x == 0 and self.y == 0:
        return other  # Якщо поточна точка нульова, повертаємо іншу точку
    if other.x == 0 and other.y == 0:
        return self  # Якщо інша точка нульова, повертаємо поточну точку

    if self.x == other.x and self.y != other.y:
        # Якщо x-координати рівні, але y різні, то результат - нульова точка
```

Рисунок 3.5 – Фрагмент програмного коду алгоритму Діффі-Геллмана

Повна реалізація алгоритму наведена в додатку В.

В алгоритмі є як класи, так і методи та функції, які описані в таблиці 3.3.

Таблиця 3.3 – Функції програмної реалізації алгоритму Діффі-Геллмана

Класи, функції	Опис	Параметри	Значення, що повертаються
Point	Клас для представлення точки на еліптичній кривій	Методи __init__(self, x, y) та __add__(self, other), який додає дві точки на кривій	Результат додавання Point(x3, y3)
generate_private_key	Функція, генерує приватний ключ	–	Ціле число в межах від 1 до $p - 1$
generate_public_key	Генерує публічний ключ з використанням приватного	private_key (int) приватний ключ	Об’єкт класу з публічним ключем
encrypt	Шифрує повідомлення	message (str) повідомлення. secret (int) спільний секретний ключ	Зашифроване повідомлення (str)
decrypt	Розшифровує повідомлення	encrypted_message (str) повідомлення. secret (int) спільний секретний ключ	Оригінальне повідомлення (str)

Взаємодію класу та функцій можна зобразити на діаграмі взаємодій функцій у вигляді діаграми послідовності, яка наведена на рисунку 3.6.

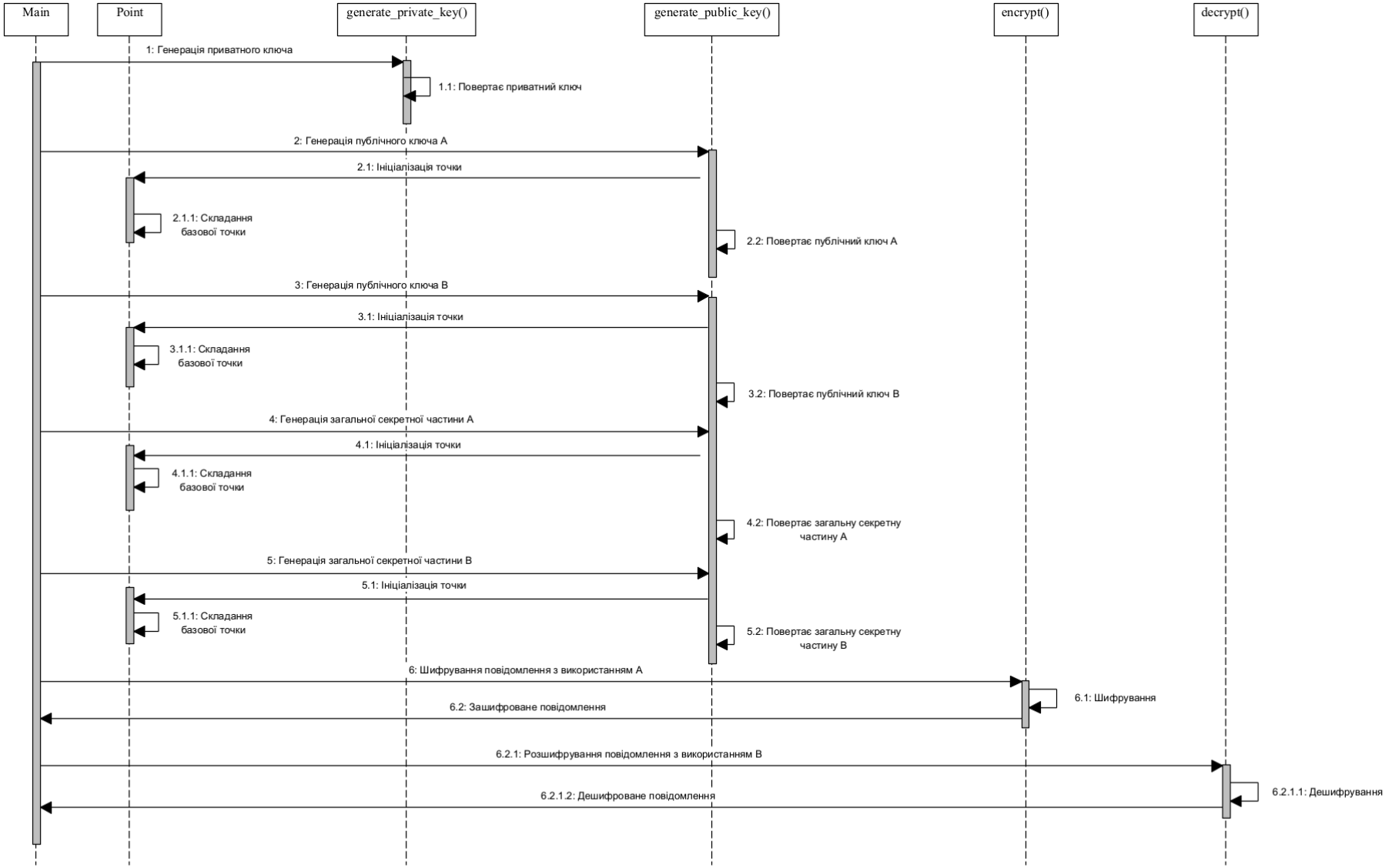

Рисунок 3.6 – Діаграма взаємодій функцій алгоритму Діффі-Геллмана у вигляді діаграми послідовності

За рисунком 3.6 наочно наведені всі процеси взаємодії між функціями програмного коду під час його виконання. Спочатку виконується основний процес, який викликає всі функції, а далі виконуються:

- клас, що надає точку на еліптичній кривій;
- взаємодія з класом через методи для складання точок (в операції генерування публічного ключа та обчислення загального секрету);
- генерація приватного ключа;
- генерація публічного ключа на основі приватного ключа;
- функція шифрування повідомлення з використанням загального секрету;
- функція для дешифрування повідомлення з використанням загального секрету.

За процесами можна описати етапи алгоритму Діффі-Геллмана:

1) запуск програми:
   - введення коефіцієнтів a і b та координати G, простого числа p;
   - створення бази для еліптичної кривої;
2) генерування приватних ключів для обох сторін;
3) використання приватних ключів задля генерування публічних ключів;
4) розрахування загального секрету кожним учасником, використовуючи публічний ключ іншої сторони;
5) шифрування повідомлення;
6) дешифрування повідомлення;
7) виведення ключів, повідомлень користувачам.

За етапами, наведеними вище, можна спроєктувати діаграму послідовності алгоритму Діффі-Геллмана. Вона наведена на рисунку 3.7.

Результат виконання алгоритму Діффі-Геллмана наведений на рисунку 3.8

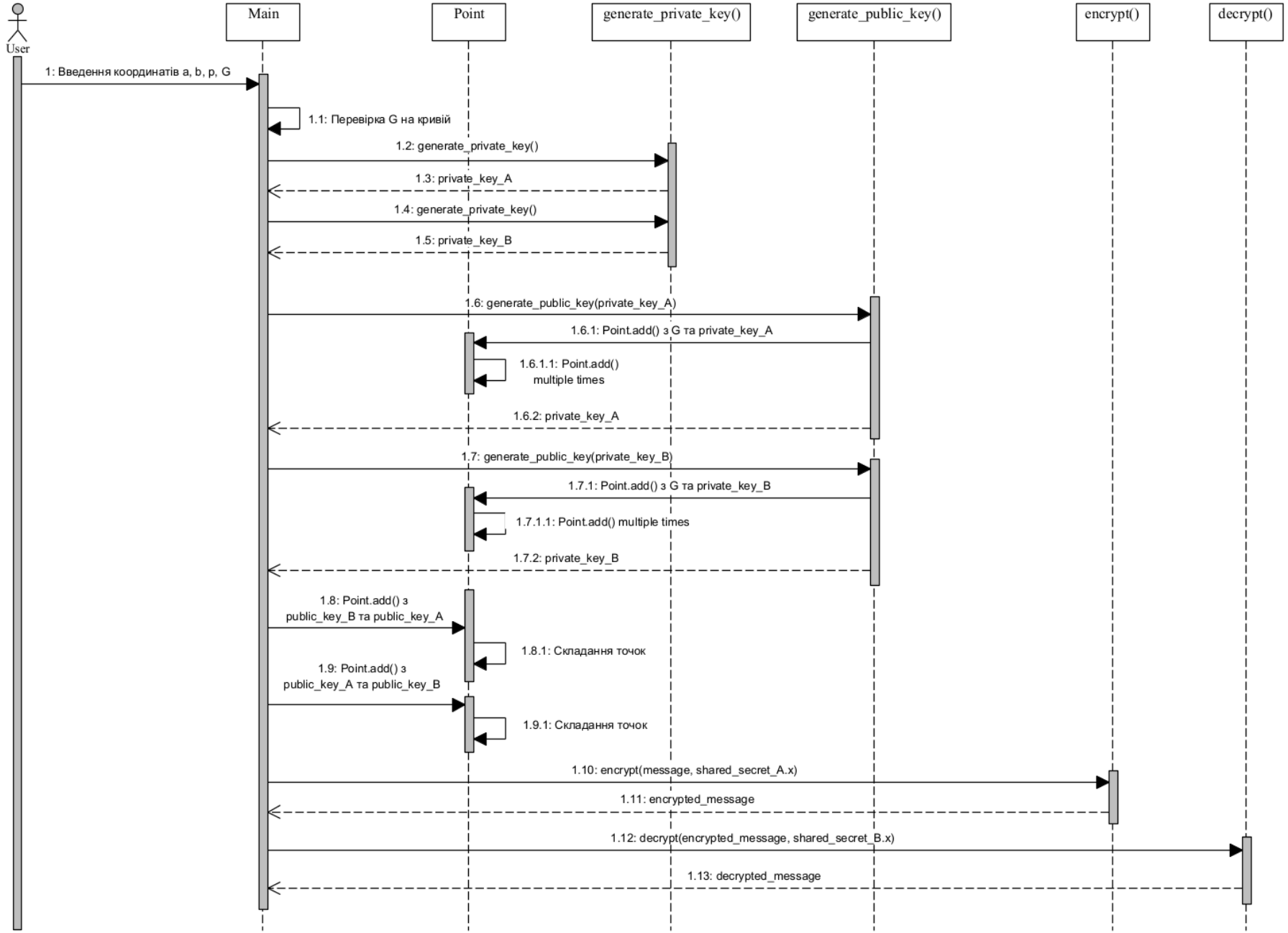

Рисунок 3.7 – Діаграма послідовності алгоритму Діффі-Геллмана

```
Введіть коефіцієнт a (ціле число): 1
Введіть коефіцієнт b (ціле число): -1
Введіть просте число p (ціле число): 17
Введіть координату x базової точки G (ціле число): 4
Введіть координату y базової точки G (ціле число): 4
Введіть повідомлення для шифрування: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt at labore et dolore mag
Коефіцієнти: a = 1, b = -1
Початкова точка (базова точка) G: Point(4, 4)
Приватний ключ A: 14
Публічний ключ A: Point(2, 3)
Приватний ключ B: 11
Публічний ключ B: Point(8, 14)
Зашифроване повідомлення: Twzmu(qx{}u(lwtwz({q|(ium|4(kwv{mk|m|}z(ilqxq{kqvo(mtq|4({ml(lw(mq}{uwl(|muxwz(qvkqlql}v|(i|(tijwzm(m|(lwtwzm(uiovi(itqy}i.
Розшифроване повідомлення: Ruxks&ovy{s&jurux&yoz&gskz2&iutykizkz{x&gjovoyiotm&kroz2&ykj&ju&ko{ysuj&zksvux&otiojoj{tz&gz&rghuxk&kz&juruxk&sgmtg&grow{g,
```

Рисунок 3.8 – Результат виконання алгоритму Діффі-Геллмана

Алгоритм Діффі-Геллмана заснований на еліптичних кривих надає високий ступінь безпеки при менших витратах обчислювальних ресурсів. Його особливості полягають у використанні арифметики на еліптичній кривій, що дозволяє значно зменшити розмір ключів та прискорити операції у порівнянні з традиційним алгоритмом Діффі-Геллмана.

3.2 Тестування алгоритмів

3.2.1 Аналіз методів тестування та відлагодження

Тестування алгоритмів Чандрасекхара та Діффі-Геллмана на еліптичних кривих включають різні методи, що направлені на оцінку їх безпеки, ефективності та продуктивності. Існує безліч методів тестування алгоритмів, але для даного випадку будуть підходити тестування на стійкість на час шифрування та дешифрування та оцінювання витрат даних.

Тестування методом криптографічної стійкості направлений на перевірку стійкості до атак та вразливості. Він включає аналіз потенційних уразливостей та змогу проведення атак, щоб оцінити наскільки алгоритм захищений. Для алгоритму Чандрасекхара краще використовувати стимулювання атаки на повторне використання ключів. Тобто, якщо ключи повторно використовуються для шифрування декількох повідомлень алгоритм може призвести до плинності інформації. Якщо тестування не пройшло, то краще за все буде впровадити використання унікальних ключів для кожної операції та видаляти старі ключи після їх використання. У випадку алгоритму Діффі-Геллмана можно імітувати атаку MITM. Втручання в алгоритм такою атакою означає виконати заміну ключів і перехопити інформацію. Якщо тестування не пройшло, тоді слід використовувати цифрові сертифікати для автентифікації сторін та використовувати додатковий захист.

Перевагами такого тестування є:

- дозволяє виявити слабкі місця, роблячи алгоритм більш стійким до реальних атак;
- допомагає розробникам зрозуміти, які аспекти потребують доопрацювання;
- надає впевненість у безпеці перед застосуванням алгоритму у важливих системах.

Недоліками тестування на стійкість є:

- вимога до значних ресурсів для моделювання складних атак;
- ефективність залежить від класифікації тестувальників та використовуваних інструментів;
- при тестуванні можна не охопити невідомі на той момент вектори атак.

Тестування методом виміру часу шифрування та дешифрування або обміну ключами оцінює продуктивність алгоритму, вимірюючи, скільки часу необхідно для виконання операцій шифрування та дешифрування (алгоритм Чандрасекхара) або обміну ключами (алгоритм Діффі-Геллмана). Для виконання такої перевірки слід виконати заміри часу, що необхідне для шифрування та дешифрування або обміну ключами повідомлень різного розміру та ініціювати сценарій взаємодії двох сторін, вимірюючи час на виконання обчислень для загального секрету. Якщо алгоритм не пройшов перевірку, то можна виконати оптимізацію, використовуючи апаратне прискорення, обрати більш продуктивні еліптичні криві тощо. Перевагами такого тестування є:

- допомагає зрозуміти, наскільки алгоритм підходить для роботи в реальному часі або систем з обмеженими ресурсами;
- надає об’єктивні дані для порівняння з іншими алгоритмами;
- дозволяє оцінити вплив параметрів на продуктивність.

Недоліками тестування є такі:

- результати залежать від обладнання та умов тестування;
- заміри на великих даних можуть потребувати значних ресрусів;
- не завжди враховуються можливості оптимізації.

Тестування методом оцінки витрати при передачі даних оцінює вплив алгоритму на розмір даних, що передаються. Збільшення даних (через метадані або інших елементів) може бути критичним у мережах з низькою пропускною здатністю. Для такого можна зашифрувати дані та виміряти, наскільки збільшився їх об’єм. Оцінити, як це впливає на пропускну здатність мережі або продуктивність системи. Якщо алгоритми не пройшли тестування, тоді можна спробувати зменшити розмір ключів або обрати оптимізовані криві, використовувати стиснення даних перед шифруванням або оптимізувати протоколи передачі.

3.2.2 Тестування алгоритмів методом криптографічної стійкості

Для алгоритму Чандрасекхара було виконано ініційовану атаку з використанням одного ключа декілька разів. Атака була пропущеною, через ненадійність джерела випадкових чисел та ключа, який використовується повторно. Реалізація тестування наведена в додатку В. Результат тестування наведений на рисунку 3.9.

```
Тимчасовий ключ повторився. Атака успішна.
Розшифроване повідомлення 1: Перше повідомлення
Розшифроване повідомлення 2: Друге повідомлення

Process finished with exit code 0
```

Рисунок 3.9 – Результат тестування: пропущена атака криптографічної стійкості алгоритму Чандрасекхара

Для алгоритму Діффі-Геллмана була застосована імітація атаки MITM, яка також пройшла. Можливо вона виникла через перехоплення ключа одного користувача іншим та його заміною. Реалізація тестування наведена в додатку В. Результат тестування наведений на рисунку 3.10.

Такі помилки потребують виправлення та налагодження алгоритмів.

```
Введіть коефіцієнт a: -1
Введіть коефіцієнт b: 1
Введіть просте число p: 17
Введіть координату x базової точки G: 5
Введіть координату y базової точки G: 6
Введіть повідомлення для шифрування: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt at labore
Повідомлення від A до B (перехоплене і змінене E): Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incidi
Зашифроване повідомлення, отримане B: Y|▯rz-v}▯▯z-q|y|▯-▯v▯-nzr▯9-p|{▯rp▯r▯▯▯-nqv}v▯pv{t-ryv▯9-▯rq-q|-rv▯▯z|q-▯rz}|▯-v{pvqvq▯{▯-n▯-yno|▯
```

Рисунок 3.10 – Результат тестування: пропущена атака криптографічної стійкості алгоритму Діффі-Геллмана

3.2.3 Тестування алгоритмів методом виміру часу шифрування та дешифрування або обміну ключами

Тестування методом виміру часу шифрування та дешифрування повідомлення алгоритмів важливо для розуміння продуктивності та масштабованості алгоритму. Необхідність такого методу тестування полягає:

- у виконанні перевірки коректності зашифрованого повідомлення та його захисту;
- у вимірі часу шифрування та дешифрування, що дозволяє оцінити ефективність алгоритму;
- у тестуванні швидкодії, яке є критичним для вибору параметрів для забезпечення балансу між безпекою та продуктивністю;
- в оцінці стійкості до помилок тощо.

Для тестування алгоритму Чандрасекхара був використаний такий алгоритм:

1) завдання вхідних даних одразу в коді, щоб автоматизувати процес;
2) виконання генерації ключів для створення закритого та відкритого;
3) вимір часу перед викликом функції шифрування, що дозволяє обчислити тривалість виконання функції; обчислення відбувається після закінчення шифрування;
4) відлік часу пере викликом функції дешифрування; обчислюється після дешифрованого повідомлення;
5) виконання перевірки дешифрування.

У самому тесті перевіряються наступне:

- оцінка продуктивності алгоритму для шифрування та дешифрування тексту;
- перевірка оригінальності розшифрованому повідомленню;
- гарантування стабільності роботи функцій шифрування та дешифрування.

Реалізація тестування наведена в додатку В. Результат тестування наведений на рисунку 3.11.

```
(.venv) PS C:\Users\     \PycharmProjects\chandrasekhar_elliptic> python test_encryption_time.py
2024-11-28 14:51:48,633 - INFO - Початок тестування шифрування та дешифрування
2024-11-28 14:51:48,634 - INFO - Вхідне повідомлення: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt at labore et dolore magna aliqua?
2024-11-28 14:51:48,634 - INFO - Параметри кривої: a=5, b=3
2024-11-28 14:51:48,641 - INFO - Згенеровано ключі: Закритий ключ=22107241846492242324881879548967281746726806171715051169061776154455916112003, Відкритий ключ=<ecdsa.ellipticcurve
.PointJacobi object at 0x000001E23154DA90>
2024-11-28 14:51:48,642 - INFO - Час шифрування повідомлення: 0.001000 секунд
2024-11-28 14:51:48,642 - INFO - Зашифроване повідомлення: (10484403141634222512062009271034882065708852564881782899148281908128014360932 3, 45132563990141893279560955627304845253073493012231766366893347950395166891480)
2024-11-28 14:51:48,643 - INFO - Час дешифрування повідомлення: 0.001000 секунд
2024-11-28 14:51:48,643 - INFO - Розшифроване повідомлення: 24439500843965521213559807127241644642863242536317420071024259341110428766034
2024-11-28 14:51:48,643 - INFO - Шифрування та дешифрування пройшли успішно!
```

Рисунок 3.11 – Результат тестування алгоритму Чандрасекхара методом виміру часу шифрування та дешифрування

Для алгоритму Діффі-Геллмана було проведено тестування методом виміру часу обміну ключами між двома учасниками. Основним завданням тесту є перевірка правильності та ефективності обміну ключами за певний проміжок часу. Алгоритм тестування наступний:

1) генерування приватних ключів для обох учасників алгоритму;
2) генерація публічних ключів для обох учасників алгоритму;
3) обчислення загального секрету, який має бути однаковим;
4) перевірка співпадіння секретів;
5) вимірювання часу обміну ключами;
6) гешування приватних ключів;
7) виведення публічних ключів.

Реалізація тестування наведена в додатку В. Результат тестування наведений на рисунку 3.12.

```
Час обміну ключами: 0.003998756408691406 сек
Приватний ключ A (хеш): 77ff25b832d2dcf9fac6611626cda817efecb680d43ae8d32bdf0a0189daff57
Приватний ключ B (хеш): 2db81e1ad11c4b8e1329956625db02833a8bb154eb9e8b2c33d0ce31db2eb9af
Публічний ключ A: -----BEGIN PUBLIC KEY-----
MFYwEAYHKoZIzj0CAQYFK4EEAAoDQgAEYG5SmUoY4tu5kwh4n5t7K5RpUjnob2e0
2N1N6iFcWn5+X2h0cHU34dXIuF2eTKH/H7KIXwFdBWdoZJjse35x4w==
-----END PUBLIC KEY-----

Публічний ключ B: -----BEGIN PUBLIC KEY-----
MFYwEAYHKoZIzj0CAQYFK4EEAAoDQgAEH9WLq/tgKqUB3S3Js+Uwbup3zo6oFiMA
cm0DRbUTkVyGjF3+KgmGblTL5PErfVjAVtm4k9pbrMyCNq4i38zS2g==
-----END PUBLIC KEY-----

Розмір спільного секрету A (в байтах): 32
Розмір спільного секрету B (в байтах): 32
Хеш спільного секрету A: 7c2a23110ca41ed9e8f65e93f4d393f98523fe568351f68cfe781330b77401e9
Хеш спільного секрету B: 7c2a23110ca41ed9e8f65e93f4d393f98523fe568351f68cfe781330b77401e9
```

Рисунок 3.12 – Результат тестування алгоритму Діффі-Геллмана методом виміру часу обміну ключами

За тестуваннями можна побачити, що час шифрування та дешифрування і обмін ключами відбувається миттєво. За успішними результатами можна побачити, що алгоритми є ефективними в криптографічному процесі.

3.2.4 Тестування алгоритмів методом оцінки витрат на передачу даних

Для тестування оцінки втрати передачі даних в алгоритмі Чандрасекхара було обрано використовувати перевірку гешів. Втрата даних поділена на три випадки:

- повідомлення передано без втрат;
- повідомлення передано з мінімальними втратами, менше, ніж один відсоток, що є в межах норми;
- повідомлення передано з великими втратами.

Для оцінювання витрати даних використовується наступна формула:

$$\text{Втрата даних}_{\text{ч}} = \frac{|\text{дешифрований геш} - \text{оригінальний геш}|}{\text{оргигінальний геш}} \times 100, \quad (3.1)$$

де оригінальний геш – геш вихідного повідомлення перед шифруванням;
дешифрований геш – геш відновленого повідомленого після дешифрування.

Різниця за модулем у формулі (3.1) показує наскільки сильно відрізняється відновлення повідомлення від вихідного. Чим більша різниця, тим більша втрата даних, що вказує на різницю.

Реалізація тестування наведена в додатку В. Результат тестування наведений на рисунку 3.13.

```
(.venv) PS C:\Users\     \PycharmProjects\chandrasekhar_elliptic> python test_data_loss.py
Повідомлення передано без втрат даних! Втрата: 0.00%
.
----------------------------------------------------------------------
Ran 1 test in 0.008s
```

Рисунок 3.13 – Результат тестування алгоритму Чандрасекхара методом оцінки втрати переданих даних

Для тестування на оцінку втрати переданих даних в алгоритмі Діффі-Геллмана необхідно слідувати наступному алгоритму:

1) ініціалізувати параметри;
2) виконання шифрування повідомлення;
3) виконання дешифрування повідомлення;
4) виконання оцінки втрати даних за формулою:

$$\text{Втрата даних}_{\text{ДГ}} = \frac{\text{кількість втрачених символів}}{\text{довжина оригінального повідомлення}} \times 100; \quad (3.2)$$

5) перевірка коректності дешифрування;
6) аналіз складу повідомлення в кількості літер, цифр, пробілів тощо;
7) виведення результатів.

Реалізація тестування наведена в додатку В. Результат тестування наведений на рисунку 3.14.

```
Втрату даних не виявлено.
Розшифрування успішне. Оригінал і розшифроване повідомлення збігаються.
Статистика повідомлення:
  Літери: 102 (83.61%)
  Цифри: 0 (0.00%)
  Пробіли: 18 (14.75%)
Час шифрування: 0.000000 сек.
Час дешифрування: 0.000000 сек.
```

Рисунок 3.14 – Результат тестування алгоритму Діффі-Геллмана методом оцінки втрати переданих даних

Отож, тестування оцінки втрати переданих даних в обох випадках пройшли успішно, дані не втрачені.

3.2.5 Відлагодження алгоритмів

Як будо наведено на рисунках 3.9-3.10 атаки на алгоритми відбулися, а тому вони потребують налагодження.

Для алгоритму Чандрасекхара були виконані налагодження за такими факторами:

- використання унікальних нових значень для k для кожного користувача при шифруванні різних повідомлень;
- використання різних ключів для кожного сеансу шифрування;
- перевірка на повторення зашифрованих повідомлень;
- забезпечення унікальності повідомлень;
- удосконалення протоколу шифрування та дешифрування.

Також було покращено результат виводу алгоритму тестування наступним чином:

- виведення даних про згенеровані ключі та параметри кривої для кращого розуміння того, що відбувається в процесі виконання;
- указання, скільки унікальних повідомлень було зашифроване;
- виведення повідомлення про унікальність або ні повідомлень;
- додана інформація про кожну ітерацію шифрування повідомлення та їх загальну кількість.

Реалізація налагодженого тестування наведена в додатку В. Результат тестування наведений на рисунку 3.15.

```
Шифрування повідомлення 5 разів з унікальними тимчасовими ключами:
Шифрування #1: зашифроване повідомлення - b688e17121a606802970f69f184b50990287ea0076d0ec167b447aa7d99babaae8df3decc1efbaa4e9250fc3a859cb3a346889e8b22f2f3e87f3acae0ef0571d270f500a9e
c1ebf67cd37849c0f8f38304847056a219710aca9014aefbb557886bd67a753a50460c9a42f651f6a0d325ea8c7eedec3139ebf1fd1d7a77417e234d938f1309c0d02fcee4b1ea061b63fe
Шифрування #2: зашифроване повідомлення - e629caf4cd16c906d12bab9c80af926a080d4f6c8ed98ed3a5b8a98b403040e6b0c0667c19af9072f4629f8268e02f07f2e4a9d64e2f7475d9da222ffb0f6fd9369f540390
111fde4a1a58c0e0483a12b2b1a60917ded185bea54c1eddc49b5233f0793021fe55d449dc01f8306d55c66255c910aa54e83b2ce4f7a468bfbc404e5a176a6cfbca840a03b13dba48f16a
Шифрування #3: зашифроване повідомлення - 9c230f8281019838838cd2fbbf50c4ce21f191e3b1ef788c057df0d23110143b2d3e222d43af03c6457ad45e8f2a52c7cee7bb39d82c395bd4e100b0f4d0bcf238e542f8dd
2be5a39415377f85d560b279a0ba7236d71d758b1e2d507acdc56e8ed4916a85c63f3bce844cfe908e8a45c61153f66763c675c0526e9444363cd2032f3149708bb06289d2ac373c1e50f1
Шифрування #4: зашифроване повідомлення - d5afe07909de92a63a9089aef032cd560a482aec70a824350c6c115318b402aa209aecb9486ef05d885a21a5831dccb948cd01ec3a8347408e2982c2d54d87518e2cd0d349
d44bb39aa0f4b2f2acb8e8a886a2c06a6e08a223295306a96e4c6a33b4b5f530fe44b7f3c2025a83769c248634d88ea754be7a6a1a2a81b6d0763b12080bba16daf6c9283b1effbd5e2396
Шифрування #5: зашифроване повідомлення - d9c902b667b14c4967b63ec1078415fe6ab9f0f010cbcbe42d462f3f71e8174c7c0b853f9fdb10985f3a061e7cd90f91d26683e20f3e4c2c9010d76f5bb217a6495d38a0e5
1db998531180bd815a84f4ea8f38f4171a0754c9e8a506dfacf3fc23122de8d4b35f7c572a14695982515b4caaa92c5f30e646f8b73be70c43a2dc83e1072ead62bb40cd42bb01da481f6f
Атака повторного використання ключа не виявлена. Всі зашифровані повідомлення унікальні.
Кількість унікальних зашифрованих повідомлень: 5
```

Рисунок 3.15 – Результат налагодженого тестування на криптографічну стійкість алгоритму Чандрасекхара

Для тестування криптографічної стійкості алгоритму Діффі-Геллмана необхідно було виправити наступні помилки:

- використання глобальних змінних без їх перевірки;
- шифрування та дешифрування без перевірки;
- використання псевдовипадкових чисел для ключів;
- необроблені винятки, що призводять до помилок;
- проблеми з точністю при великих числах.

Для поліпшення коду були виправлені помилки такими змінами:

1) додано використання RSA для підпису та перевірки підпису публічних ключів;

2) функції sign_public_key та verify_signature використовують RSA для підпису публічних ключів, що є захистом для підробки ключів;

3) покращена генерація приватних і публічних ключів для еліптичних кривих з використанням базової точки, яка перевіряється;

4) додана перевірка підпису публічних ключів за допомогою RSA, що робить неможливими проведення атаки;

5) після успішної перевірки підпису продовжується генерація спільного секрету та обмін повідомленнями;

6) додана перевірка шифрування та дешифрування;

7) використані небезпечні випадкові числа для генерації приватних ключів;

8) використання певної координати для шифрування.

Реалізація налагодженого тестування наведена в додатку В. Результат тестування наведений на рисунку 3.16.

```
Підписи публічних ключів дійсні. Атака MITM не вдалася!
A і B обмінялися публічними ключами.
Введіть повідомлення для шифрування: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt at labore et dolore ma
Коефіцієнти: a = -1, b = 1
Початкова точка (базова точка) G: Point(5, 6)
Приватний ключ A: 10
Публічний ключ A: Point(13, 3)
Підпис публічного ключа A: b'\x07\x89\xdb\x13g\xd4\xe8P\xfc0\xa6\xff\xd7"\x18\x14sbx1SI6\xb0\xc38\xbf\x1c\xe7\x92\xb4"n\t$/\xe70\x1e\x17\x97\x94,\xcf
Приватний ключ B: 13
Публічний ключ B: Point(5, 11)
Підпис публічного ключа B: b'\xaf~:lq\xed)X\xd4\xc8\xd8+D\xf8\t}\x1cI\xfb\xecV\xd6\x13\xa3\x17S\xbe8\xcf\xf0\xc6[E\xe5\x1d\xcb\xc7\xf7:\x02\x16\xaf\n
Зашифроване повідомлення: Qtwjr%nuxzr%itqtw%xny%frjy1%htsxjhyjyzw%finunxhnsl%jqny1%xji%it%jnzxrti%yjrutw%nshninizsy%fy%qfgtwj%jy%itqtwj%rflsf%fqnvzf
Розшифроване повідомлення: Dgj]e0ahkme0\gdgj0kal0Ye]l$0[gfk][l]lmj0Y\ahak[af_0]dal$0k]\0\g0]amkeg\0l]ehgj0af[a\a\mfl0Yl0dYZgj]0]l0\gdgj]0eY_fY0YdaimY
```

Рисунок 3.16 – Результат налагодженого тестування на криптографічну стійкість алгоритму Діффі-Геллмана

Висновки до розділу 3

За розділом були розглянуті ключові аспекти проєктування інструментальних засобів для реалізації алгоритмів шифрування з використанням еліптичних кривих. Було обрано ефективний інструментарій для розробки, який забезпечує стабільну роботу системи з урахуванням технологічної платформи, яка оптимально підходить для реалізації криптографічних алгоритмів. Виконана розробка алгоритмів Чандрасекхара та Діффі-Геллмана, які дозволяють забезпечити правильну взаємодію компонентів системи. Було проведене тестування за обраними методами, які показали стійкість алгоритмів до атак. Було виконано деякі налагодження, які додали більшого захисту алгоритмам.

РОЗДІЛ 4 ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ АЛГОРИТМІВ ЗА МЕТРИКАМИ

4.1 Результати проведених експериментів

За проведеними результатами можна стверджувати, що алгоритми Чандрасекхара та Діффі-Геллмана є стійкими криптографічними алгоритмами з використанням еліптичних кривих. З описаних сценаріїв (табл. 2.1) були обрані ті, що значно можуть вплинути на роботу алгоритмів, якщо не врахувати їх особливості.

Результати експериментів наведені в таблиці 4.1.

Таблиця 4.1 – Порівняння результатів експериментів криптографічних алгоритмів Чандрасекхара та Діффі-Геллмана на еліптичних кривих

Експеримент	Алгоритм Чандрасекхара	Алгоритм Діффі-Геллмана
Оцінка криптографічної стійкості	Була застосована атака підбору ключів. Спочатку атаку було здійснено, але після налагодження та внесення змін в алгоритм, атака не змогла пройти зовсім	Була застосована атака MITM, яка пройшла. Були виконані зміни в алгоритми, після яких атака не відбувалася
Вимір часу шифрування та дешифрування або обміну ключами	Час шифрування та дешифрування є мінімальним, що є гарним показником для передачі великих обсягів інформації	Час обміну ключами є мінімальним, що означає якісну передачу інформації між користувачами
Оцінка втрат на передачу даних	Втрату даних не було виявлено зовсім	Втрату даних не було виявлено зовсім

Алгоритм Чандрасекхара більш орієнтований на ефективне шифрування великих обсягів даних з фокусом на час шифрування та енергоефективність. Протокол Діффі-Геллмана спрямований на безпечний обмін ключами, але може бути менш ефективним з точки зору швидкості та енергоефективності.

4.2 Аналіз результатів експериментів

За проведеними експериментами та тестуванням алгоритмів Чандрасекхара та Діффі-Геллмана можна сказати, що протокол Діффі-Геллмана може мати менші витрати на передачу даних при умовах, коли обмін ключами відбувається регулярно. Для алгоритму Чандрасекхара передача даних зазвичай зменшується після одного разу обміну.

За результатами проведеного дослідження можна виконати порівняльний аналіз обох алгоритмів задля виконання остаточних висновків, який алгоритм для чого вартий. Порівняння наведене в таблиці 4.2.

Таблиця 4.2 – Порівняльний аналіз криптографічних алгоритмів на еліптичних кривих Чандрасекхара та Діффі-Геллмана

Критерій	Алгоритм Чандрасекхара	Алгоритм Діффі-Геллмана
1	2	3
Тип криптографії	Симетричне шифрування з використання еліптичних кривих	Асиметричний обмін ключами на основі еліптичних кривих
Основна мета	Шифрування повідомлень з використанням приватних і публічних ключів	Встановлення спільного секрету між двома сторонами

Продовження табл. 4.2

1	2	3
Ключова залежність	Використовує спільний секрет для шифрування і дешифрування	Обмін публічними ключами для створення спільного секрету
Тип операцій з точками	Множення точок на еліптичній кривій для генерації ключів	Додавання точок на еліптичній кривій для генерації спільного секрету
Безпека	Висока стійкість до атак при правильній реалізації кривої SECP256k1	Залежить від складності знаходження дискретного логарифма на еліптичних кривих

Шифрування	Симетричне шифрування за допомогою AES після обміну ключами	Не має власного шифрування, використовується лише для обміну ключами
Протокол обміну ключами	Генерація спільного секрету за допомогою еліптичних кривих для подальшого шифрування	Використовує обмін публічними ключами для створення спільного секрету
Часова складність	Час залежить від довжини ключа та операцій множення на кривій	Час залежить від обчислень на еліптичній кривій для створення спільного секрету
Енергоефективність	Залежить від потужності пристрою та оптимізації коду для AES	Може бути менш енергоефективним через потребу в кількох обчисленнях
Використовувані параметри	Параметри кривої SECP256k1 та генерація ключів для шифрування	Параметри визначаються для конкретної еліптичної кривої та точки G

Продовження табл. 4.2

1	2	3
Захист від атак	Захищений від атак перехоплення ключів	Захищений від атак через використання обміну публічними ключами та захист від MITM атак при належному застосуванні

Алгоритм Чандрасекхара використовує симетричне шифрування AES з еліптичними кривими для обміну ключами, де основна мета — забезпечення безпеки при шифруванні повідомлень. Протокол Діффі-Геллмана спрямований на встановлення спільного секрету між двома сторонами, з можливістю шифрування повідомлень лише після обміну публічними ключами.

Висновки до розділу 4

За розділом були виконані узагальнення за виконаною роботою. Було виконано дослідження за виконаними експериментами та порівняльний аналіз між алгоритмами

Чандрасекхара та Діффі-Геллмана. Визначено, що обидва алгоритми є ефективними для криптографічного шифрування.

ЗАГАЛЬНІ ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ

У ході виконання дослідження було успішно вирішено поставлені завдання, що дозволило отримати вагомі як теоретичні так і практичні результати, спрямовані на вдосконалення криптографічних методів захисту інформації з використанням еліптичних кривих.

Був проведений комплексний огляд сучасних методів шифрування даних із використанням еліптичних кривих, що включав вивчення математичних основ, переваг і недоліків таких підходів, а також їх використання в межах України. Аналіз літератури дозволив виявити найбільш перспективні напрямки застосування еліптичної криптографії, зокрема в умовах зростаючих вимог до енергоефективності та безпеки в мобільних і хмарних середовищах.

Проведено оцінку використання еліптичних кривих у таких сферах, які фінанси, державні інформаційні системи, системи IoT та блокчейн. У ході аналізу криптографічних методів розглянуті ключові протоколи та алгоритми, такі як Діффі-Геллмана, Ель-Гамаля, Чандрасекхара тощо. Виокремлені переваги та недоліки кожного, що стало основою для вибору об’єктів подальших експериментів.

Виконано обґрунтування вибору напрямку дослідження та опис метрик ефективності. Вибір еліптичної криптографії як об’єкту дослідження обґрунтовано її високою криптографічною стійкістю, низькими вимогами до обчислювальних ресурсів та актуальністю в умовах сучасних кіберзагроз. Описані метрики для оцінки алгоритмів, що містять криптографічну стійкість, вимір часу шифрування та дешифрування і обміну ключами та витрати на передачу даних.

У процесі розробки алгоритмів були створені ієрархії та взаємодії класів для алгоритмів Чандрасекхара та Діффі-Геллмана, що спрощує їх подальше вдосконалення. Тестування включало оцінку криптографічної стійкості, яке показало високу ефективність алгоритмів після їх налагодження. Тестування виміру часу та можливих втрат надало високі результати одразу: виконання алгоритмів миттєве, а втрата даних

не відбувається. Було встановлено, що алгоритм Чандрасекхара забезпечує краще співвідношення між криптографічною стійкістю та енергоефективністю, а Діффі-Геллмана потребує додаткових механізмів захисту для запобігання MITM-атакам, але забезпечує високу продуктивність у реальному часі. Усі налагодження алгоритмів були виконані якісно та протестовані знову. Атаки на алгоритми не відбувались.

Отож, можна підкреслити, що алгоритм Чандрасекхара доцільно використовувати для систем, що потребують високого рівня безпеки та обмежені в енергоресурсах (наприклад, IoT), а протокол Діффі-Геллмана з додатковою автентифікацією підходить для систем, де важлива швидкість обміну даними. У перспективі можна рекомендувати адаптувати алгоритми для роботи у квантово-стійких середовищ та розробити апаратні прискорювачі для еліптичної криптографії, що знизить затрати на її впровадження у великомасштабних системах.

Отримані результати створюють підґрунтя для подальшого розвитку сучасних методів криптографії, спрямованих на забезпечення безпеки та ефективності у різних галузях, зокрема в умовах зростання глобальних кіберзагроз.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Єсін В. І. Криптографічні методи в кібербезпеці : робоча програма навч. дисципліни / В. І. Єсін. – Харків : Харків. нац. ун-т ім. В.Н. Каразіна, 2020. – 21 с.

2. М. К. Еліптичні криві / Купріков М. // Математика, що нас оточує: минуле, сучасне, майбутнє : Зб. наук. пр. VII Всеукр. конф. курсантів та студентів, Львів. – Львів, 2020. – С. 84–87.

3. Павло Я. Основні відомості про еліптичні криві, які використовують в еліптичній криптографії / Янковий Павло // Математика, що нас оточує: минуле, сучасне, майбутнє : Зб. наук. пр. VII Всеукр. конф. курсантів та студентів, Львів. – Львів, 2020. – С. 88–91.

4. Наталія Щ. Криптографія на еліптичних кривих та її практичне застосування / Щур Наталія. – 2-ге вид. – [Б. м. : б. в.], 2023. – 121 с.

5. Фізулі-кизи С. С. Огляд та порівняння методів захисту інформації на основі еліптичних кривих / Саміра Султанова Фізулі-кизи // Інформаційні моделюючі технології, системи та комплекси : Зб. тез IV Міжнар. науково-практ. конф., Черкаси, 25 трав. 2023 р. – 26 трав. 2024 р. – Черкаси, 2023. – С. 76–79.

6. Post Quantum Cryptography: A survey of Past and Future / Kokare Priyanka N [et al.] // JOUR. – 2024. – P. 5–20.

7. Bao J. Research on the Security of Elliptic Curve Cryptography / Jiaxu Bao // Social Sciences and Economic Development : 2022 7th International Conference. – [S. l.], 2022.

8. Щур Н. Криптографія на еліптичних кривих та її практичне застосування / Наталія Щур, Олександра Покотило, Єлизавета Байлюк // Кібербезпека: освіта, наука, техніка. – Т. 1, № 21. – С. 48–64.

9. Yadav A. A Comparative Study of Elliptic curve and Hyperelliptic Curve Cryptography Methods and an Overview of Their Applications / Anu Yadav, Prabha Sharma,

Yogita Gigras // Intelligent Systems for Cybersecurity : 2024 International Conference, Gurugram. – [S. l.], 2024. – P. 1–6.

10. Development of a Data Transmission Security System on a Mobile Smart Trash Bin Using an Elliptic Curve Cryptography Algorithm / Fachroni Arbi Murad [et al.] // Information and Communications Technology : 2023 6th International Conference, Yogyakarta. – [S. l.], 2023. – P. 527–532.

11. A Survey on Cryptography: Comparative Study between RSA vs ECC Algorithms, and RSA vs El-Gamal Algorithms / Fatma Mallouli [et al.] // Cyber Security and Cloud Computing : 2019 6th IEEE International Conference. – [S. l.], 2019.

12. Howe J. How (not) to Design and Implement Post-Quantum Cryptography / James Howe, Thomas Prest, Daniel Apon // PQShield. – Vol. 462. – P. 1–42.

13. Медведєва К. В. Використання нечіткого екстрактора для генерації ключів шифрування на основі параметрів клавіатурного почерку / К. В. Медведєва, Н. Р. Кондратенко // Вінниця, 31 трав. 2022 р. – Вінниця, 2022. – С. 3.

14. Караджян Б. Ю. Методи шифрування інформації за допомогою алгебраїчних фракталів : пояснювальна записка до кваліфікаційної роботи здобувача вищої освіти на другому (магістерському) рівні / Караджян Б. Ю. – Харків, 2023. – 65 с.

15. Голда А. А. Методи та засоби забезпечення інформаційної безпеки в системах інтернет-банкінгу : кваліфікаційна робота на здобуття освітнього ступеня магістр / Голда А. А. – Тернопіль, 2023. – 72 с.

16. Кубайчук О. О. Огляд застосування метаевристичного підходу в криптоаналізі / О. О. Кубайчук // Вісник ХНТУ. – 2023. – Т. 2, № 85. – С. 147–149.

17. Guo H. A Survey on Blockchain Technology and its Security / Huaqun Guo, Xingjie Yu // Journal Pre-proof. – 2021. – P. 1–22.

18. An efficient ECC-based cp-ABE scheme for power IOT / Rui Cheng [et al.] // Processes. – 2021. – Vol. 9, no. 7. – P. 1176.

19. Науменко С. Забезпечення кібербезпеки в Smart-імплантах: роль полегшеної криптографії / С. Науменко, І. Розломій, П. Михайловський // Інформаційна

безпека та комп’ютерні технології : VII Міжнар. науково-практ. конф. – Кропивницький, 2023. – С. 17–18.

20. Приходіна П. Телекомунікаційний пристрій шифрування даних на основі алгоритму Ель Гамаля : робота на здобуття кваліфікаційного ступеня бакалавра / Приходіна П. – Суми, 2023. – 52 с.

21. Александров М. О. Нейромережеве синхронне генерування ключів підвищеної надійності для симетричних систем шифрування : дисертація на здобуття наукового ступеня доктора філософії / Александров Микита Олександрович. – Луцьк, 2023. – 137 с.

22. De Luca S. Cryptographic security: Critical to Europe's digital sovereignty : Briefing / Stefano De Luca, Tristan Marcelin. – [S. l.] : European Parliamentary Research Service, 2024. – 8 p.

23. Oswald E. Enhancing Simple Power-Analysis Attacks on Elliptic Curve Cryptosystems / Elisabeth Oswald // Cryptographic Hardware and Embedded Systems : 4th International Workshop, Redwood Shores, CA, 13–15 August 2020. – [S. l.], 2020.

24. Post-Quantum Cryptography on FPGA Based on Isogenies on Elliptic Curves / Brian Koziel [et al.] // IEEE Transactions on Circuits and Systems I Regular Papers. – 2020. – PP, no. 99. – P. 1–14.

25. Yuan X. Future of Secure Communication: Transitioning from ECC to Post-Quantum Cryptography : book / X. Yuan, C. Liu. – [S. l. : s. n.], 2022. – 85 p.

26. Дичка І. А. Модифікований віконний метод однократного множення точки еліптичної кривої на скаляр у полі GF(p) / І. А. Дичка, М. В. Онай, Т. П. Дрозда // Журнал Радіоелектроніка, інформатика, управління. – 2020. – Т. 2. – С. 95–103.

27. Драченко В. Дослідження розбіжних парабол (еліптичних функцій) / В. Драченко, М. Фурман // Проблеми вищої математичної освіти: виклики сучасності : XLIX Науково-техн. конф. ф-ту інформ. технологій та комп'ютер. інженерії. – Вінниця, 2020. – С. 2.

28. Пчелінцев І. С. Дослідження алгоритмів шифрування для забезпечення кібербезпеки телекомунікаційних систем та мереж : магістерська дисертація на здобуття ступеня магістра / Пчелінцев Ілля Сергійович. – Київ, 2019. – 87 с.

29. Бичова І. В. Особливості криптографічного захисту ділової документації / І. В. Бичова, В. В. Чередниченко // Перспективи управлінської діяльності суб'єктів господарювання в контексті економічної безпеки : матеріали міжнар. форуму з безпеки, Черкаси, 25–27 трав. 2019 р. – [Б. м.], 2019. – С. 214–216.

30. Кулібаба В. Comparative analysis of cryptoprimitives on canonical elliptic curves and Edwards curves / В. Кулібаба // Radiotekhnika. – 2019. – Vol. 198, no. 2019. – P. 203–208.

31. Царук Д. Застосування асиметричної криптографії для безпечного документообігу / Дмитро Царук, Андрій Фесенко // Information, Communication, Society : 12th International Academic Conference, Lviv, 18–20 трав. 2023 р. – Lviv, 2023. – С. 40–41.

32. Мерзлікіна В. І. Дослідження та практична реалізація алгоритмів криптографії та криптоаналізу, що використовують апарат модулярної арифметики : кваліфікаційна робота магістра / Мерзлікіна В. І. – Запоріжжя, 2020. – 79 с.

33. Панчак О. І. Реалізація криптографії на еліптичних кривих в сучасних месенджерах : дипломна робота бакалавра / Панчак Остап Ігорович. – Київ, 2022. – 78 с.

34. Зозуля П. Деякі невирішені задачі математики / П. Зозуля // Математика, що нас оточує: минуле, сучасне, майбутнє : IX Всеукр. конф. курсантів та студентів, Львів. – Львів, 2023. – С. 42–43.

35. Гавриш Б. Криптографія та стеганографія в інформаційній безпеці / Б. Гавриш // Використання сучасних інформаційних технологій в діяльності Національної поліції України : матеріали Всеукр. науково-практ. семінару, Дніпро, 28 листоп. 2019 р. – Дніпро, 2019. – С. 83–90.

36. Грик Ю. Аналіз захисту інформації в системах електроного документообігу / Ю. Грик, З. Сельменська // Стан, досягнення і перспективи інформаційних систем і технологій : XX Всеукр. науково-техн. конф. молодих вчен., аспірантів та студентів, Одеса, 21–22 квіт. 2020 р. – Одеса, 2020. – С. 61–63.

37. Самойлов І. В. Криптографія : навч. посіб. для здобувачів першого (бакалавр.) рів. вищ. освіти / І. В. Самойлов, А. А. Матійко, А. С. Сторчак. – Київ : КПІ ім. Ігоря Сікорського, 2023. – 372 с.

38. Merezhko M. Квантова криптографія: Революційний захист даних у добу квантових обчислень / Maksym Merezhko // Харківський національний університет імені В. Н. Каразіна. – 2023. – С. 2–10.

39. Лунгол О. Оцінка застосування криптографічних алгоритмів в системах автентифікації на основі біометричних даних / О. Лунгол // Наука і техніка сьогодні. – 2024. – Т. 8, № 36. – С. 1089–1090.

40. Аналіз методів забезпечення конфіденційності даних, які передаються з БПЛА / Sergiy Gnatyuk [та ін.] // Кібербезпека: освіта, наука, техніка. – 2022. – № 17. – С. 1445–1450.

Новиков Д. Технології постквантової криптографії / Д. Новиков, В. Полторак // Адаптивні системи автоматичного управління. – 2023. – № 42. – С. 171–183.

41. Глеб В. Ю. Засіб передачі інформації, що зашифрована асиметричними алгоритмами, в комп’ютерних мережах : магістерська дисертація / Глеб Владислав Юрійович. – Київ, 2023. – 69 с.

42. Математичні основи алгебраїчних решіток та їх застосування в квантовій криптології / Андрій Кожухівський [та ін.] // Ukrainian Information Security Research Journal. – 2024. – Т. 26, № 1. – С. 117–129.

43. Шевченко Д. Вплив віртуальної реальності на основні сфери життя людини / Д. Шевченко, М. Прокопович, А. Денисюк // Комп'ютерні ігри і мультимедіа як інноваційний підхід до комунікації – 2024 : Матеріали iv Всеукр. науково – техн. конф.

молодих вчен., аспірантів та студентів, Одеса, 26–27 верес. 2024 р. – Одеса, 2024. – С. 355–357.

44. Кобус О. Еліптичні криві над скінченними полями та їх застосування в сучасних криптографічних системах для підвищення безпеки та ефективності / О. Кобус // Наука і техніка сьогодні. – 2024. – Т. 7, № 35. – С. 1031–1040.

45. Улічев О. Стандартизація еліптичних кривих: аналіз та впровадження в криптографічні протоколи / О. Улічев // Central Ukrainian Scientific Bulletin. Technical Sciences. – 2024. – Т. 9, № 40. – С. 14–26.

46. Михайлишин К. Символіка безпеки: інтеграція криптографії з кібербезпекою для захисту цифрових систем / Катерина Михайлишин, Іван Опірський // Ukrainian Information Security Research Journal. – 2024. – Т. 26, № 1. – С. 147–157.

47. Popp M. I. The Encryption Algorithms / Marharyta Ihorivna Popp // Автоматизація та комп’ютерноінтегровані технології у виробництві та освіті: стан, досягнення, перспективи розвитку : Всеукраїнська науково-практична Інтернет-конференція, Черкаси, 14–20 March 2022. – Черкаси, 2022. – P. 26–28.

48. FIPS 186-5. Digital Signature Standard (DSS). – Replaces FIPS PUB 186-4 ; effective from 2024-02-03. – Official edition. – Gaithersburg : Computer Security Division (Information Technology Laboratory), 2024. – 131 p.

49. Сатура А. Р. Розробка практичної моделі обміну інформації з використанням квантового розподілу ключів шифрування : кваліфікаційна бакалаврська робота / Сатура Андрій Русланович. – Суми, 2022. – 32 с.

50. Станкевіч А. О. Дослідження та реалізація протоколу Діффі-Геллмана на еліптичних кривих : автореф. магістерської дисертації / Станкевіч А. О. ; ЧНУ ім. Петра Могили. – Миколаїв, 2021. – 15 с.

51. Сава Л. М. Аналіз та реалізація криптографічних перетворень для алгоритму ECDH (Elliptic Curve Diffie-Hellman) : кваліфікаційна робота бакалавра / Сава Лука Михайлович. – Тернопіль, 2022. – 78 с.

52. Маньковський С. Python-модель протоколу узгодження секретного ключа у групі з довільної кількості учасників / С. Маньковський, Ю. Матієшин // Інфокомунікаційні технології та електронна інженерія. – 2024. – Т. 4, № 1. – С. 93–101.

53. Савчин Ю. В. Система захисту даних на базі алгоритму Ель-Гамаля : дипломна робота на здобуття ступеня бакалавра / Савчин Юхим Васильович. – Київ, 2019. – 74 с.

54. Клещев М. А. Пристрій захисту інформації на основі алгоритму Ель Гамаля : робота на здобуття кваліфікаційного рівня бакалавра / Клещев М. А. – Суми, 2020. – 48 с.

55. Кріпака І. А. Усічення цифрового підпису для схем типу Ель-Гамаля : дипломна робота на здобуття ступеня бакалавра / Кріпака Ілля Анатолійович. – Київ, 2023. – 52 с.

56. Яковлєв С. В. Методи усічення цифрового підпису для схем типу Ель-Гамаля та ДСТУ4145-2002 / С. В. Яковлєв, І. А. Кріпака // Системи і технології зв’язку, інформатизації та кібербезпеки. – 2024. – № 5. – С. 227–230.

57. Increasing the Level of Security of Internet Things Network Systems due to Encryption of Data on Devices with Limited Computer Systems / Роман Миколайович Черненко [et al.] // Кібербезпека: освіта, наука, техніка. – 2021. – Vol. 3, no. 11. – P. 124–135.

58. Денисенко Д. Г. Програмний модуль захисту інформації на основі OpenPGP : кваліфікаційна робота здобувача вищої освіти освітнього ступеня магістр / Денисенко Дмитро Геннадійович. – Київ, 2020. – 93 с.

59. Лапенко В. О. Криптографічний модуль на базі генератора псевдовипадкових чисел : дипломна робота здобувача вищої освіти освітнього ступеня бакалавр / Лапенко Валентина Олександрівна. – Київ, 2021. – 63 с.

60. Нікітін В. А. Підвищення безпеки системи навчання робототехніці : магістерська дисертація / Нікітін Валерій Андрійович. – Київ, 2019. – 89 с.

61. Миронець І. В. Криптографічні алгоритми та особливості їх використання в блокчейн-системах / Ірина Валеріївна Миронець, Андрій Володимирович Шкребтій // Cybersecurity & Critical Information Infrastructure Protection (CIIP). – 2019. – Т. 25, № 2. – С. 104–109.

62. Горобченко М. О. Дослідження засобів генерації ключів в системах з криптографічним захистом : магістерська дисертація / Горобченко Микита Олегович. – Київ, 2019. – 114 с.

63. Плотніков В. М. Захист даних засобом цифрового підпису / В. М. Плотніков, Ю. В. Борцова // Автоматизація технологічних і бізнес-процесів. – 2019. – Т. 11, № 4. – С. 49–55.

64. Ярмошевич А. М. Система захисту даних на базі методу шифрування RSA : дипломна робота на здобуття ступеня бакалавра / Ярмошевич Андрій Миколайович. – Київ, 2019. – 76 с.

65. Литвиненко Ю. С. Криптоаналіз алгоритму цифрового підпису «Вершина» : дипломна робота на здобуття ступеня бакалавра / Литвиненко Юлія Сергіївна. – Київ, 2022. – 54 с.

66. Олійник М. А. Застосування медового шифрування до схеми цифрового підпису Шнорра / М. А. Олійник // Могилянський математичний журнал. – 2021. – Т. 4, № 3.

67. Баранов К. О. Побудова системи електронного цифрового підпису : магістерська робота / Баранов Костянтин Олександрович. – Київ, 2021. – 36 с.

68. Реалізація криптостійкого алгоритму із простою процедурою шифрування та дешифрування на основі еліптичних кривих / В. В. Завгородній [та ін.] // Таврійський науковий вісник. Серія: Технічні науки. – 2023. – № 3. – С. 13–20.

69. Кадім Рахма Р. М. Моделі та методи побудови операційних вузлів для полів Галуа, які використовуються при криптографічному захисту інформації на основі еліптичних кривих : дисертація / Кадім Рахма Рахма Моххамед. – Львів, 2019. – 190 с.

70. Онай М. В. Класифікація методів дискретного логарифмування на еліптичній кривій / М. В. Онай, Д. Т. Гулько // Вісник Херсонського національного технічного університету. – 2024. – Т. 1, № 88. – С. 264–271.

71. Недзельський Р. В. Генетичний метод пошуку параметрів еліптичних кривих : кваліфікаційна робота до здобуття освітнього ступеня магістр / Недзельський Роман Володимирович. – Тернопіль, 2022. – 86 с.

72. Шарацький О. С. Система попередньої обробки та шифрування даних : дипломна робота на здобуття ступеня бакалавра / Шарацький Олександр Сергійович. – Київ, 2019. – 75 с.

73. Літава В. Г. Моделі та засоби підвищення живучості інформаційно управляючих систем на основі еліптичних кривих : дисертація на здобуття наукового ступеня кандидата технічних наук / Літава Владислав Гжегож. – Тернопіль, 2019. – 139 с.

74. Криптологія : монографія. – Київ : Нац. авіац. ун-т, 2019. – 34 с.

75. Онацький О. В. Криптографічний протокол автентифікації із нульовим розголошенням секрету на еліптичних кривих із застосуванням відкритих ключів і випадкових повідомлень / О. В. Онацький, О. В. Жарова // Цифрові технологій. – 2019. – № 26. – С. 22–28.

76. Щур Н. Прикладна криптологія : метод. рек. до виконання лаборатор. робіт / Н. Щур. – Житомир : Держ. ун-т «Житом. політехніка», 2021. – 104 с.

77. Стабецька Т. А. Методи та засоби синтезу операцій розширеного матричного криптографічного перетворення довільної кількості аргументів : дисертація / Стабецька Тетяна Анатоліївна. – Черкаси, 2019. – 208 с.

78. Могилевич Д. І. Аналіз стану системи технічного обслуговування електронного комунікаційного обладнання мережі спеціального призначення / Д. І. Могилевич, Г. В. Долженко // Системи і технології зв’язку, інформатизації та кібербезпеки. – 2024. – № 5. – С. 101–107.

79. Кавин Я. Методи обробки та аналізу зображень / Я. Кавин, Б. Кавин // Study Of Modern Problems Of Civilization : V International Scientific and Practical Conference, Oslo, 19–23 жовт. 2020 р. – [Б. м.], 2020. – С. 451–453.

80. Янік І. І. Алгоритми шифрування на мобільних пристроях з використанням криптографічних методів : кваліфікаційна робота / Янік Іван Іванович. – Тернопіль, 2023. – 84 с.

81. Метод адаптивного багатошарового захисту інформації на основі стеганографії та криптографії / В. В. Лукічов [та ін.] // Інформаційні технології та комп'ютерна інженерія. – 2023. – № 3. – С. 4–11.

82. Цезарук С. Ю. Метод шифрування даних на основі коригуючих кодів : кваліфікаційна робота / Цезарук С. Ю. – Тернопіль, 2023. – 92 с.

Додаток А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
_____

ПРОГРАММА ДЛЯ ДОСЛІДЖЕННЯ АЛГОРИТМІВ ШИФРУВАННЯ ДАНИХ З ВИКОРИСТАННЯМ ЕЛІПТИЧНИХ КРИВИХ

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.1547-01-ЛЗ

Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
_____

Керівник розробки
_____ Тетяна ГРИШЕЧКІНА
_____

Виконавець
_____ Христина МАКАРОВА
_____

Нормоконтролер
_____ Світлана ВОЛКОВА
_____

2026

МІНСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО

44165850.1547-01

ПРОГРАММА ДОСЛІДЖЕННЯ АЛГОРИТМІВ ШИФРУВАННЯ ДАНИХ З ВИКОРИСТАННЯМ ЕЛІПТИЧНИХ КРИВИХ

Технічне завдання

Аркушів 10

2026

ЗМІСТ

ВСТУП

Технічне завдання розроблено для теми кваліфікаційної роботи «Дослідження алгоритмів шифрування даних з використанням еліптичних кривих». Алгоритми призначені для дослідження, тестування та практичного застосування сучасних криптографічних алгоритмів, зокрема Чандрасекхара та Діффі-Геллмана.

Еліптичні криві є сучасним математичним інструментом, що забезпечує високий рівень захисту інформації з меншими обчислювальними витратами порівняно з традиційними криптографічними методами. У рамках цього проєкту досліджується використання еліптичних кривих для побудови криптографічних систем, що забезпечують безпеку передачі даних.

Необхідність розробки даних алгоритмів обумовлена зростаючими вимогами до інформаційної безпеки в умовах активного розвитку цифрових технологій. Алгоритми призначені для використання в навчальних, дослідницьких і прикладних цілях, зокрема в сфері кібербезпеки, фінансових установах, військових структурах та інших областях, де потрібно забезпечити надійний захист інформації.

Програмне забезпечення дозволяє:

- тестувати роботу алгоритмів шифрування та розшифрування даних;
- аналізувати ефективність алгоритмів у різних умовах;
- демонструвати переваги еліптичних кривих у криптографії.

Розробка спрямована на підвищення розуміння криптографічних процесів і практичного використання алгоритмів, що ґрунтуються на сучасних математичних методах.

1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Розробка програмного забезпечення на тему «Дослідження алгоритмів шифрування даних з використанням еліптичних кривих» здійснюється відповідно до наказу ректора Українського державного університету науки і технологій №1401 від 02.10.2025, яким затверджено перелік тем дипломних проєктів для студентів випускного курсу.

Розробка виконується відповідно до навчального плану 121 Інженерія програмного забезпечення, затвердженого Українським державним університетом науки і технологій. Проєкт є складовою наукової теми – «Дослідження алгоритмів шифрування даних з використанням еліптичних кривих».

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення програмного забезпечення призначене для реалізації, тестування та аналізу двох сучасних криптографічних алгоритмів, що базуються на еліптичних кривих: алгоритму Чандрасекхара та алгоритму Диффі-Геллмана. Основні функції програми:

1) генерація ключів та параметрів на основі еліптичних кривих;
2) шифрування та розшифрування даних;
3) моделювання обміну ключами між користувачами;
4) проведення тестів для оцінки безпеки та продуктивності алгоритмів.

Експлуатаційне призначення програмного забезпечення забезпечує:

− наочну демонстрацію принципів роботи криптографічних алгоритмів для навчальних цілей;
− підвищення ефективності дослідницької діяльності в галузі криптографії;
− автоматизацію процесів тестування криптографічних методів;
− створення основи для подальшої інтеграції еліптичних кривих у практичні системи інформаційної безпеки.

Впровадження цього ПЗ дозволить зменшити витрати часу на дослідження, покращити якість навчання у сфері криптографії, а також розширити можливості для впровадження безпечних методів захисту інформації в різних галузях.

3 ВИМОГИ ДО ПРОГРАМИ АБО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до функціональних характеристик

Програма повинна забезпечувати:

- генерацію криптографічних ключів на основі алгоритмів Чандрасекхара та Диффі-Геллмана;
- шифрування та розшифрування даних з використанням еліптичних кривих;
- обмін ключами між двома користувачами через відкритий канал;
- тестування та аналіз продуктивності алгоритмів;
- зручний графічний або текстовий інтерфейс для взаємодії з користувачем.

3.2 Вимоги до надійності

Програма повинна забезпечувати:

- програма повинна забезпечувати автоматичне збереження даних при виникненні збоїв;
- у разі відмови обладнання повинно забезпечуватись відновлення роботи з мінімальними втратами даних;
- архівна копія вихідного коду повинна зберігатися на зовнішньому носії;
- програма повинна бути захищена від несанкціонованого доступу;
- забезпечення стійкості до помилкових або некоректних вхідних даних.

3.3 Умови експлуатації

Умовами експлуатації є:

- програма повинна функціонувати за температури від +10°C до +35°C і відносної вологості не більше 80%;
- рекомендований вид обслуговування: періодична перевірка працездатності обладнання та оновлення програмного забезпечення;

− персонал, що працює з програмою, повинен мати базові знання з криптографії та навички роботи з ПК.

3.4 Вимоги до складу й параметрів технічних засобів

Мінімальними вимогами до обладнання є:

− процесор з частотою не менше 2 ГГц;

− оперативна пам’ять — не менше 4 ГБ;

− дисковий простір — не менше 500 МБ;

− підтримка операційної системи Windows 10/11 або Linux.

3.5 Вимоги до інформаційної та програмної сумісності

Вимогами до інформаційної та програмної сумісності є:

− програма повинна підтримувати введення даних у текстовому форматі або через файли .txt/.csv;

− вихідні дані мають бути представлені у вигляді виводу в консолі або через командний рядок IDE;

− програма має бути розроблена на мові Python для забезпечення портативності;

− код програми повинен бути задокументованим та забезпечувати легкість подальшої модифікації.

3.6 Вимоги до маркування й упаковки

Вимогами до маркування та упаковки є:

− програмний виріб повинен мати електронну документацію, що містить інструкцію користувача та технічний опис;

− при поставці в друкованій версії упаковка має бути водонепроникною та маркованою відповідно до стандартів.

3.7 Вимоги до транспортування й зберігання

Вимогами до транспортування та зберігання є:

- програмний продукт може бути розповсюджений через цифрові носії або у вигляді файлів для завантаження;
- у разі зберігання на фізичних носіях (USB, CD/DVD) вони повинні бути захищені від впливу високої температури, вологості та магнітних полів;
- термін зберігання на цифрових носіях – не менше 5 років за умови дотримання рекомендацій.

3.8 Спеціальні вимоги

Спеціальними вимогами є:

- програма повинна підтримувати розширення функціоналу для інтеграції нових алгоритмів шифрування;
- можливість адаптації до мобільних платформ або веб-інтерфейсу на запит замовника;
- забезпечення захисту ПЗ від несанкціонованого копіювання шляхом використання ліцензійного ключа.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Складові програмної документації наведені в таблиці А.1

Таблиця А.1 – Складові програмної документації

Складова програмної документації	Опис
Технічне завдання	Визначає мету, функціональні можливості та вимоги до програмного забезпечення
Специфікації	Надає список всіх необхідних документів у відповідності із вказаними номерами
Текст програми	Забезпечує розуміння логіки реалізації для розробників
Опис програми	Документ із детальним описом алгоритмів, структур даних та архітектури
Керівництво користувача	Містить інструкції з використання програми, опис функціоналу та можливих помилок
Керівництво програміста	Містить інструкції із підключення необхідних складових для роботи програмних алгоритмів
Спеціальні вимоги	Документація повинна бути доступна в електронному вигляді, створена українською мовою, та мати англомовну версію для міжнародного використання

5 ТЕХНІКО-ЕКОНОМІЧНІ ПОКАЗНИКИ

Розробка забезпечує економію часу на впровадження криптографічних методів завдяки автоматизації процесів генерації ключів і аналізу даних. Орієнтовна економія витрат у дослідницьких і освітніх установах може становити до 20% завдяки спрощенню навчання і досліджень.

Можливою річною потребою є:

- освітні установи: до 50 копій програмного забезпечення на рік;
- компанії, що займаються інформаційною безпекою: до 20 копій на рік.

Економічними перевагами є:

- нижча вартість розробки порівняно з аналогами завдяки використанню відкритих бібліотек і простих інструментів;
- висока ефективність роботи з великими обсягами даних у порівнянні з традиційними методами шифрування;
- поліпшення умов праці для дослідників через автоматизацію та зменшення ймовірності помилок.

Соціальним значенням програми є:

- сприяє підготовці висококваліфікованих спеціалістів у галузі інформаційної безпеки;
- збільшує рівень захисту інформації в умовах підвищених кіберзагроз.

6 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Стадії та етапи розробки наведені в таблиці А.2.

Таблиця А.2 – Стадії та етапи розробки

Етапи / стадії	Підетапи	Термін виконання, міс.
Аналітичний етап	Збір і аналіз вимог. Розробка технічного завдання. Затвердження документації	1
Етап проєктування	Розробка архітектури програмного забезпечення. Створення прототипу користувацького інтерфейсу	2
Етап реалізації	Написання коду та інтеграція алгоритмів. Проведення модульного тестування	3
Етап тестування та налагодження	Проведення функціонального тестування. Виправлення помилок і оптимізація	1
Етап впровадження	Передача програми замовнику. Проведення навчання персоналу	1

7 ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ

Види випробувань алгоритмів є наступні:

- функціональні випробування: перевірка роботи основних модулів програми;
- стрес-тестування: оцінка роботи під високими навантаженнями;
- безпекові випробування: тестування стійкості до атак.

Загальними вимогами до приймання є:

- програма повинна повністю відповідати технічному завданню;
- усі знайдені помилки мають бути усунені.

Приймальна комісія складається з:

- голови комісії – керівник проєкту;
- інших членів-представників замовника, викладачі, відповідальні за навчальний процес тощо;

Дослідною експлуатацією є:

- тривалість: 1 місяць;
- мета: перевірка програми в умовах реальної експлуатації та збір відгуків.

Додаток Б

Специфікації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
_____

ПРОГРАММА ДЛЯ ДОСЛІДЖЕННЯ АЛГОРИТМІВ ШИФРУВАННЯ ДАНИХ З ВИКОРИСТАННЯМ ЕЛІПТИЧНИХ КРИВИХ

Специфікації
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.1547-01-ЛЗ

Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
_____

Керівник розробки
_____ Тетяна ГРИШЕЧКІНА
_____

Виконавець
_____ Христина МАКАРОВА
_____

Нормоконтролер
_____ Світлана ВОЛКОВА
_____

2026

МІНСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО

44165850.1547-01

ПРОГРАММА ДОСЛІДЖЕННЯ АЛГОРИТМІВ ШИФРУВАННЯ ДАНИХ З ВИКОРИСТАННЯМ ЕЛІПТИЧНИХ КРИВИХ

Специфікації

Аркушів 3

2026

ЗМІСТ

1 ДОКУМЕНТАЦІЯ

Документація наведена в таблиці Б.1.

Таблиця Б.1 – Специфікація документації

Позначення	Найменування	Примітка
44165850.1547-01-ЛЗ	Лист затвердження	
44165850.1547-12-ЛЗ	Лист затвердження	
44165850.1547-12	Текст програми	
44165850.1547-01-ЛЗ	Лист затвердження	
44165850.1547-13	Опис програми	
44165850.1547-І3-ЛЗ	Лист затвердження	
44165850.1547-І3	Керівництво користувача	
44165850.1547-33-ЛЗ	Лист затвердження	
44165850.1547-33	Керівництво програміста	

2 КОМПЛЕКСИ

Комплекси наведені в таблиці Б.2

Таблиця Б.2 – Специфікації комплексів

Позначення	Найменування	Примітка
44165850.1547-12	Текст програми	
44165850.1547-13	Опис програми	

3 КОМПОНЕНТИ

Компоненти наведені в таблиці Б.3

Таблиця Б.3 – Специфікації компонентів

Позначення	Найменування	Примітка
44165850.1547-12	Текст програми	
44165850.1547-13	Опис програми	

Додаток В

Текст програми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
_____

ПРОГРАММА ДОСЛІДЖЕННЯ АЛГОРИТМІВ ШИФРУВАННЯ ДАНИХ З ВИКОРИСТАННЯМ ЕЛІПТИЧНИХ КРИВИХ

Текст програми
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.1547-12-ЛЗ

Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
_____

Керівник розробки
_____ Тетяна ГРИШЕЧКІНА
_____

Виконавець
_____ Христина МАКАРОВА
_____

Нормоконтролер
_____ Світлана ВОЛКОВА
_____

2026

МІНСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО

44165850.1547-12

ПРОГРАММА ДОСЛІДЖЕННЯ АЛГОРИТМІВ ШИФРУВАННЯ ДАНИХ З ВИКОРИСТАННЯМ ЕЛІПТИЧНИХ КРИВИХ

Текст програми

Аркушів 11

2026

Текст algorithm_chandrasekhar_elliptic.py

```
import os
from ecdsa import SECP256k1, ellipticcurve
from hashlib import sha256
from Cryptodome.Cipher import AES
from Cryptodome.Util.Padding import pad, unpad
# Отримання параметрів кривої
def get_curve_parameters():
    while True:
        try:
            a = int(input("Введіть коефіцієнт a (ціле число): "))
            b = int(input("Введіть коефіцієнт b (ціле число): "))
            # Перевірка на виродження кривої
            if 4 * a**3 + 27 * b**2 == 0:
                print("Помилка: Крива вироджена (дискримінант дорівнює 0). Введіть інші значення.")
            else:
                return a, b
        except ValueError:
            print("Помилка: Введіть цілі числа для коефіцієнтів a та b.")
# Генерація ключів
def generate_keys():
    private_key = int.from_bytes(os.urandom(32), 'big') % field_order
    public_key = private_key * G
    return private_key, public_key
# Деривація ключа
def key_derivation(secret_point):
    return sha256(str(secret_point).encode()).digest()
# Шифрування повідомлення з використанням симетричного ключа
def encrypt_message_aes(message, derived_key):
    cipher = AES.new(derived_key, AES.MODE_CBC)
    ciphertext = cipher.encrypt(pad(message.encode(), AES.block_size))
    return cipher.iv + ciphertext  # додаємо IV для декодування
# Розшифрування повідомлення
def decrypt_message_aes(encrypted_message, derived_key):
    iv = encrypted_message[:16]
    ciphertext = encrypted_message[16:]
    cipher = AES.new(derived_key, AES.MODE_CBC, iv)
    return unpad(cipher.decrypt(ciphertext), AES.block_size).decode()
# Основне шифрування
def encrypt(message, public_key):
    k = int.from_bytes(os.urandom(32), 'big') % field_order
    temp_point = k * G
    shared_secret = k * public_key
    # Використовуємо KDF для створення симетричного ключа
    derived_key = key_derivation(shared_secret.x())
    encrypted_message = encrypt_message_aes(message, derived_key)
    return encrypted_message, temp_point
# Основне розшифрування
def decrypt(encrypted_message, private_key, temp_point):
    shared_secret = private_key * temp_point
    derived_key = key_derivation(shared_secret.x())
    return decrypt_message_aes(encrypted_message, derived_key)
# Приклад використання
if __name__ == "__main__":
    a, b = get_curve_parameters()
    curve = SECP256k1.curve
    G = SECP256k1.generator
```

```
    field_order = G.order()
    private_key, public_key = generate_keys()
    print(f"Закритий ключ: {private_key}")
    print(f"Відкритий ключ: ({public_key.x()}, {public_key.y()})")
    message = input("Введіть повідомлення для шифрування: ")
    # Шифрування повідомлення
    encrypted_message, temp_point = encrypt(message, public_key)
    print(f"Зашифроване повідомлення: {encrypted_message}")
    # Дешифрування повідомлення
    decrypted_message = decrypt(encrypted_message, private_key, temp_point)
    print(f"Дешифроване повідомлення: {decrypted_message}")
```

Текст key_reuse_no.py

```
import os
from ecdsa import SECP256k1
from hashlib import sha256
from Cryptodome.Cipher import AES
from Cryptodome.Util.Padding import pad, unpad
import pytest
# Отримання параметрів кривої
def get_curve_parameters(a=None, b=None):
    if a is None or b is None:
        while True:
            try:
                a = int(input("Введіть коефіцієнт a (ціле число): "))
                b = int(input("Введіть коефіцієнт b (ціле число): "))
                # Перевірка на виродження кривої
                if 4 * a ** 3 + 27 * b ** 2 == 0:
                    print("Помилка: Крива вироджена (дискримінант дорівнює 0). Введіть інші значення.")
                else:
                    return a, b
            except ValueError:
                print("Помилка: Введіть цілі числа для коефіцієнтів a та b.")
    else:
        return a, b
# Генерація ключів
def generate_keys(field_order, G):
    private_key = int.from_bytes(os.urandom(32), 'big') % field_order
    public_key = private_key * G
    return private_key, public_key
# Деривація ключа
def key_derivation(secret_point):
    return sha256(str(secret_point).encode()).digest()
# Шифрування повідомлення з використанням симетричного ключа
def encrypt_message_aes(message, derived_key):
    cipher = AES.new(derived_key, AES.MODE_CBC)
    ciphertext = cipher.encrypt(pad(message.encode(), AES.block_size))
    return cipher.iv + ciphertext  # Додаємо IV для декодування
# Розшифрування повідомлення
def decrypt_message_aes(encrypted_message, derived_key):
    iv = encrypted_message[:16]
    ciphertext = encrypted_message[16:]
    cipher = AES.new(derived_key, AES.MODE_CBC, iv)
    return unpad(cipher.decrypt(ciphertext), AES.block_size).decode()
# Основне шифрування
def encrypt(message, public_key, field_order, G):
```

```
    k = int.from_bytes(os.urandom(32), 'big') % field_order
    temp_point = k * G
    shared_secret = k * public_key
    # Використовуємо KDF для створення симетричного ключа
    derived_key = key_derivation(shared_secret.x())
    encrypted_message = encrypt_message_aes(message, derived_key)
    return encrypted_message, temp_point
# Основне розшифрування
def decrypt(encrypted_message, private_key, temp_point):
    shared_secret = private_key * temp_point
    derived_key = key_derivation(shared_secret.x())
    return decrypt_message_aes(encrypted_message, derived_key)
# Тестування для виявлення атаки повторного використання ключа
def test_key_reuse():
    a, b = 2, 3  # Наприклад, задаємо коефіцієнти a та b
    curve = SECP256k1.curve
    G = SECP256k1.generator
    field_order = G.order()
    # Генерація ключів
    private_key, public_key = generate_keys(field_order, G)
     message = "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt at labore et dolore magna aliqua"
    # Зберігаємо зашифровані повідомлення для порівняння
    encrypted_messages = set()
    print("Шифрування повідомлення 5 разів з унікальними тимчасовими ключами:")
    for i in range(5):  # Шифруємо повідомлення 5 разів
        encrypted_message, temp_point = encrypt(message, public_key, field_order, G)
        encrypted_messages.add(encrypted_message)
        print(f"Шифрування #{i+1}: зашифроване повідомлення - {encrypted_message.hex()}")
    # Перевірка на наявність повторень зашифрованих повідомлень
    if len(encrypted_messages) < 5:
        print("Атака повторного використання ключа виявлена! Декілька зашифрованих повідомлень однакові.")
    else:
        print("Атака повторного використання ключа не виявлена. Всі зашифровані повідомлення унікальні.")
        print(f"Кількість унікальних зашифрованих повідомлень: {len(encrypted_messages)}")
# Приклад використання
if __name__ == "__main__":
    test_key_reuse()
```

Текст test_encryption_time.py

```
import logging
import time
from hashlib import sha256
from ecdsa import SECP256k1
import os
# Налаштування логування
logging.basicConfig(
    level=logging.INFO,  # Рівень логування: INFO
    format="%(asctime)s - %(levelname)s - %(message)s",  # Формат повідомлень
)
# Ключові параметри
def get_curve_parameters(a, b):
    # Перевірка на виродження кривої
    if 4 * a ** 3 + 27 * b ** 2 == 0:
        raise ValueError(
            "Помилка: Крива вироджена (дискримінант дорівнює 0). Будь ласка, введіть інші значення для a та b."
```

```
            )
        return a, b
    # Генерація ключів
    def generate_keys():
        # Закритий ключ: випадкове число
        private_key = int.from_bytes(os.urandom(32), "big") % field_order
        # Відкритий ключ: множення генераторної точки на закритий ключ
        public_key = private_key * G
        return private_key, public_key
    # Процес шифрування
    def encrypt(message, public_key):
        # Перетворюємо повідомлення в хеш і потім в точку
        message_hash = sha256(message.encode()).hexdigest()
        message_point = int(message_hash, 16) % field_order
        # Створюємо тимчасову точку на основі відкритого ключа та випадкового числа
        k = int.from_bytes(os.urandom(32), "big") % field_order
        temp_point = k * G
        # Зашифроване повідомлення
        encrypted_message = (message_point + temp_point.x() % field_order, temp_point.y() % field_order)
        return encrypted_message, k
    # Процес дешифрування
    def decrypt(encrypted_message, private_key, k):
        message_point, temp_y = encrypted_message
        # Обчислюємо тимчасову точку на основі закритого ключа
        temp_point = k * G
        # Дешифруємо повідомлення
        decrypted_message = (message_point - temp_point.x() % field_order) % field_order
        return decrypted_message
    # Тестування
    def test_encryption_decryption_time():
        logging.info("Початок тестування шифрування та дешифрування")
        # Повідомлення
        message = (
            "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt at labore et dolore
magna aliqua?"
        )
        logging.info(f"Вхідне повідомлення: {message}")
        # Параметри еліптичної кривої
        a, b = 5, 3
        get_curve_parameters(a, b)
        logging.info(f"Параметри кривої: a={a}, b={b}")
        # Генерація ключів
        private_key, public_key = generate_keys()
        logging.info(f"Згенеровано ключі: Закритий ключ={private_key}, Відкритий ключ={public_key}")
        # Вимірювання часу шифрування
        start_time = time.time()
        encrypted_message, k = encrypt(message, public_key)
        encryption_time = time.time() - start_time
        logging.info(f"Час шифрування повідомлення: {encryption_time:.6f} секунд")
        logging.info(f"Зашифроване повідомлення: {encrypted_message}")
        # Вимірювання часу дешифрування
        start_time = time.time()
        decrypted_message = decrypt(encrypted_message, private_key, k)
        decryption_time = time.time() - start_time
        logging.info(f"Час дешифрування повідомлення: {decryption_time:.6f} секунд")
        logging.info(f"Розшифроване повідомлення: {decrypted_message}")
        # Перевірка коректності дешифрування
        original_message_hash = int(sha256(message.encode()).hexdigest(), 16) % field_order
        if decrypted_message == original_message_hash:
```

```
            logging.info("Шифрування та дешифрування пройшли успішно!")
        else:
            logging.error("Помилка: Повідомлення не збігається після дешифрування")
    # Переконання, що змінні field_order та G визначені
    curve = SECP256k1.curve
    G = SECP256k1.generator
    field_order = G.order()

    if __name__ == "__main__":
        test_encryption_decryption_time()
```

Текст test_data_loss.py

```
    import unittest
    from hashlib import sha256
    import os
    from ecdsa import SECP256k1
    class TestChandrashekharECC(unittest.TestCase):
        def generate_keys(self):
            curve = SECP256k1.curve
            G = SECP256k1.generator
            field_order = G.order()
            private_key = int.from_bytes(os.urandom(32), 'big') % field_order
            public_key = private_key * G
            return private_key, public_key
        def encrypt(self, message, public_key):
            curve = SECP256k1.curve
            G = SECP256k1.generator
            field_order = G.order()
            message_hash = sha256(message.encode()).hexdigest()
            message_point = int(message_hash, 16) % field_order
            k = int.from_bytes(os.urandom(32), 'big') % field_order
            temp_point = k * G
            encrypted_message = (message_point + temp_point.x() % field_order, temp_point.y() % field_order)
            return encrypted_message, k
        def decrypt(self, encrypted_message, private_key, k):
            curve = SECP256k1.curve
            G = SECP256k1.generator
            field_order = G.order()
            message_point, temp_y = encrypted_message
            temp_point = k * G
            decrypted_message = (message_point - temp_point.x() % field_order) % field_order
            return decrypted_message
        def test_data_integrity(self):
            # Повідомлення для перевірки
              message = "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt at labore et
dolore magna aliqua"
            # Генерація ключів
            private_key, public_key = self.generate_keys()
            # Шифрування та дешифрування повідомлення
            encrypted_message, k = self.encrypt(message, public_key)
            decrypted_message = self.decrypt(encrypted_message, private_key, k)
            # Хеш оригінального повідомлення
            original_message_hash = int(sha256(message.encode()).hexdigest(), 16) % SECP256k1.generator.order()
            # Розрахунок проценту втрат
            loss_percentage = abs(decrypted_message - original_message_hash) / original_message_hash * 100
            # Перевірка цілісності даних (порівняння хешів)
```

```
        if loss_percentage == 0:
            print(f"Повідомлення передано без втрат даних! Втрата: 0.00%")
        elif loss_percentage < 1:  # Потеря данных менее 1% считается нормой
            print(f"Повідомлення передано з мінімальними втратами: ({loss_percentage:.2f}%) - це в межах норми.")
        else:
            print(f"Помилка! Повідомлення передано з великими втратами: ({loss_percentage:.2f}%)!")
if __name__ == "__main__":
    unittest.main()
```

Текст algorithm_dh.py

```
import unittest
from hashlib import sha256
import os
from ecdsa import SECP256k1
class TestChandrashekharECC(unittest.TestCase):
    def generate_keys(self):
        curve = SECP256k1.curve
        G = SECP256k1.generator
        field_order = G.order()
        private_key = int.from_bytes(os.urandom(32), 'big') % field_order
        public_key = private_key * G
        return private_key, public_key
    def encrypt(self, message, public_key):
        curve = SECP256k1.curve
        G = SECP256k1.generator
        field_order = G.order()
        message_hash = sha256(message.encode()).hexdigest()
        message_point = int(message_hash, 16) % field_order
        k = int.from_bytes(os.urandom(32), 'big') % field_order
        temp_point = k * G
        encrypted_message = (message_point + temp_point.x() % field_order, temp_point.y() % field_order)
        return encrypted_message, k
    def decrypt(self, encrypted_message, private_key, k):
        curve = SECP256k1.curve
        G = SECP256k1.generator
        field_order = G.order()
        message_point, temp_y = encrypted_message
        temp_point = k * G
        decrypted_message = (message_point - temp_point.x() % field_order) % field_order
        return decrypted_message
    def test_data_integrity(self):
        # Повідомлення для перевірки
          message = "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt at labore et
dolore magna aliqua"
        # Генерація ключів
        private_key, public_key = self.generate_keys()
        # Шифрування та дешифрування повідомлення
        encrypted_message, k = self.encrypt(message, public_key)
        decrypted_message = self.decrypt(encrypted_message, private_key, k)
        # Хеш оригінального повідомлення
        original_message_hash = int(sha256(message.encode()).hexdigest(), 16) % SECP256k1.generator.order()
        # Розрахунок проценту втрат
        loss_percentage = abs(decrypted_message - original_message_hash) / original_message_hash * 100
        # Перевірка цілісності даних (порівняння хешів)
        if loss_percentage == 0:
            print(f"Повідомлення передано без втрат даних! Втрата: 0.00%")
```

```
        elif loss_percentage < 1:  # Потеря данных менее 1% считается нормой
            print(f"Повідомлення передано з мінімальними втратами: ({loss_percentage:.2f}%) - це в межах норми.")
        else:
            print(f"Помилка! Повідомлення передано з великими втратами: ({loss_percentage:.2f}%)!")
if __name__ == "__main__":
    unittest.main()
```

Текст MITM-atack_no.py

```
import random
import hashlib
from Crypto.PublicKey import RSA
from Crypto.Signature import pkcs1_15
from Crypto.Hash import SHA256
class Point:
    # Клас для представлення точки на еліптичній кривій
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def __add__(self, other):
        if self.x == 0 and self.y == 0:
            return other
        if other.x == 0 and other.y == 0:
            return self
        if self.x == other.x and self.y != other.y:
            return Point(0, 0)
        if self.x == other.x and self.y == other.y:
            m = (3 * self.x ** 2 + a) * pow(2 * self.y, -1, p) % p
        else:
            denominator = other.x - self.x
            if denominator % p == 0:
                raise ValueError("Неможливо знайти обернене значення для даного модуля.")
            m = (other.y - self.y) * pow(denominator, -1, p) % p
        x3 = (m ** 2 - self.x - other.x) % p
        y3 = (m * (self.x - x3) - self.y) % p
        return Point(x3, y3)
    def __repr__(self):
        return f"Point({self.x}, {self.y})"
def generate_private_key():
    return random.randint(1, p - 1)
def generate_public_key(private_key):
    public_key = Point(0, 0)
    for _ in range(private_key):
        public_key = public_key + G if public_key.x != 0 or public_key.y != 0 else G
    return public_key
def encrypt(message, secret):
    return ''.join(chr((ord(char) + secret) % 256) for char in message)
def decrypt(encrypted_message, secret):
    return ''.join(chr((ord(char) - secret) % 256) for char in encrypted_message)
def sign_public_key(public_key, private_rsa_key):
    # Підписати публічний ключ
    public_key_data = f"{public_key.x},{public_key.y}".encode()
    h = SHA256.new(public_key_data)
    signature = pkcs1_15.new(private_rsa_key).sign(h)
    return signature
def verify_signature(public_key, signature, rsa_key):
    # Перевірка підпису
```

```
            public_key_data = f"{public_key.x},{public_key.y}".encode()
            h = SHA256.new(public_key_data)
            try:
                pkcs1_15.new(rsa_key).verify(h, signature)
                return True
            except (ValueError, TypeError):
                return False
        # Введення коефіцієнтів
        a = int(input("Введіть коефіцієнт a: "))
        b = int(input("Введіть коефіцієнт b: "))
        p = int(input("Введіть просте число p: "))
        # Введення координат базової точки G
        while True:
            try:
                x_G = int(input("Введіть координату x базової точки G: "))
                y_G = int(input("Введіть координату y базової точки G: "))
                if (y_G ** 2 % p) != (x_G ** 3 + a * x_G + b) % p:
                    raise ValueError("Точка G не лежить на еліптичній кривій.")
                G = Point(x_G, y_G)
                break
            except ValueError as e:
                print(e)
        # Генерація ключів RSA для підпису
        private_rsa_key = RSA.generate(2048)
        public_rsa_key = private_rsa_key.publickey()
        # Генерація приватних та публічних ключів для A та B
        private_key_A = generate_private_key()
        private_key_B = generate_private_key()
        public_key_A = generate_public_key(private_key_A)
        public_key_B = generate_public_key(private_key_B)
        # Підпис публічних ключів
        signature_A = sign_public_key(public_key_A, private_rsa_key)
        signature_B = sign_public_key(public_key_B, private_rsa_key)
        # Перевірка підпису публічних ключів
        if not verify_signature(public_key_A, signature_A, public_rsa_key):
            print("Підпис публічного ключа A недійсний!")
            exit()
        if not verify_signature(public_key_B, signature_B, public_rsa_key):
            print("Підпис публічного ключа B недійсний!")
            exit()
        # Успішна перевірка підписів
        print("Підписи публічних ключів дійсні. Атака MITM не вдалася!")
        # Обмін публічними ключами
        print("A і B обмінялися публічними ключами.")
        # Загальне секретне значення
        shared_secret_A = generate_public_key(private_key_B)
        shared_secret_B = generate_public_key(private_key_A)
        # Шифрування та розшифрування повідомлення
        message = input("Введіть повідомлення для шифрування: ")
        encrypted_message = encrypt(message, shared_secret_A.x)  # Шифрування з використанням x-координати загального секрету
        decrypted_message = decrypt(encrypted_message, shared_secret_B.x)  # Розшифрування з використанням x-координати
        # Вивід результатів
        print("Коефіцієнти: a = {}, b = {}".format(a, b))
        print("Початкова точка (базова точка) G:", G)
        print("Приватний ключ A:", private_key_A)
        print("Публічний ключ A:", public_key_A)
        print("Підпис публічного ключа A:", signature_A)
```

```
print("Приватний ключ B:", private_key_B)
print("Публічний ключ B:", public_key_B)
print("Підпис публічного ключа B:", signature_B)
print("Зашифроване повідомлення:", encrypted_message)
print("Розшифроване повідомлення:", decrypted_message)
```

Текст test_key_exchange_time.py

```
import time
from cryptography.hazmat.primitives.asymmetric import ec
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives import serialization
def generate_private_key():
    # Генерація приватного ключа
    return ec.generate_private_key(ec.SECP256K1())
def generate_public_key(private_key):
    # Генерація публічного ключа на основі приватного ключа
    return private_key.public_key()
def derive_shared_secret(private_key, peer_public_key):
    # Обчислення спільного секрету на основі приватного ключа та публічного ключа іншої сторони
    shared_secret = private_key.exchange(ec.ECDH(), peer_public_key)
    return shared_secret
def test_key_exchange_time():
    # Генерація приватних та публічних ключів для двох сторін з вимірюванням часу
    start_time = time.time()
    # Генерація приватних ключів для двох учасників
    private_key_A = generate_private_key()
    private_key_B = generate_private_key()
    # Генерація публічних ключів для двох учасників
    public_key_A = generate_public_key(private_key_A)
    public_key_B = generate_public_key(private_key_B)
    # Обчислення спільного секрету для обох сторін
    shared_secret_A = derive_shared_secret(private_key_A, public_key_B)
    shared_secret_B = derive_shared_secret(private_key_B, public_key_A)
    # Перевірка на співпадіння секретів
    assert shared_secret_A == shared_secret_B, "Спільні секрети не співпадають!"
    end_time = time.time()
    key_exchange_time = end_time - start_time
    # Виведення додаткової інформації
    print("Час обміну ключами:", key_exchange_time, "сек")
    # Виведення хешу приватного ключа A
    private_key_A_bytes = private_key_A.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.TraditionalOpenSSL,
        encryption_algorithm=serialization.NoEncryption()
    )
    private_key_A_hash = hashes.Hash(hashes.SHA256())
    private_key_A_hash.update(private_key_A_bytes)
    print("Приватний ключ A (хеш):", private_key_A_hash.finalize().hex())
    # Виведення хешу приватного ключа B
    private_key_B_bytes = private_key_B.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.TraditionalOpenSSL,
        encryption_algorithm=serialization.NoEncryption()
    )
    private_key_B_hash = hashes.Hash(hashes.SHA256())
    private_key_B_hash.update(private_key_B_bytes)
```

```
    print("Приватний ключ B (хеш):", private_key_B_hash.finalize().hex())
    # Виведення публічних ключів
    print("Публічний ключ A:", public_key_A.public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKeyInfo
    ).decode())
    print("Публічний ключ B:", public_key_B.public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKeyInfo
    ).decode())
    # Розмір спільного секрету
    print("Розмір спільного секрету A (в байтах):", len(shared_secret_A))
    print("Розмір спільного секрету B (в байтах):", len(shared_secret_B))
    # Хеші спільного секрету (переводим в байтовую строку перед хешированием)
    shared_secret_A_hash = hashes.Hash(hashes.SHA256())
    shared_secret_A_hash.update(shared_secret_A)
    print("Хеш спільного секрету A:", shared_secret_A_hash.finalize().hex())
    shared_secret_B_hash = hashes.Hash(hashes.SHA256())
    shared_secret_B_hash.update(shared_secret_B)
    print("Хеш спільного секрету B:", shared_secret_B_hash.finalize().hex())
# Запуск тесту
test_key_exchange_time()
```

Текст test_data_loss.py

```
import random
import time
class Point:
    # Клас для представлення точки на еліптичній кривій
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def __add__(self, other):
        # Додавання двох точок на кривій
        if self.x == 0 and self.y == 0:
            return other  # Якщо поточна точка нульова, повертаємо іншу точку
        if other.x == 0 and other.y == 0:
            return self  # Якщо інша точка нульова, повертаємо поточну точку
        if self.x == other.x and self.y == other.y:
            # Використовуємо формулу подвоєння
            m = (3 * self.x ** 2 + a) * pow(2 * self.y, -1, p) % p
        else:
            # Використовуємо формулу додавання
            m = (other.y - self.y) * pow(other.x - self.x, -1, p) % p
        x3 = (m ** 2 - self.x - other.x) % p
        y3 = (m * (self.x - x3) - self.y) % p
        return Point(x3, y3)
    def __repr__(self):
        return f"Point({self.x}, {self.y})"
def encrypt(message, secret):
    # Шифрування повідомлення
    return ''.join(chr((ord(char) + secret) % 256) for char in message)
def decrypt(encrypted_message, secret):
    # Розшифрування повідомлення
    return ''.join(chr((ord(char) - secret) % 256) for char in encrypted_message)
def evaluate_data_loss(original_message, decrypted_message):
    # Оцінка втрати даних при шифруванні та розшифруванні
```

```
        original_len = len(original_message)
        decrypted_len = len(decrypted_message)
        if original_len != decrypted_len:
            # Якщо довжини різні, виводимо кількість втрачених символів
            loss = abs(original_len - decrypted_len)
            # Обчислюємо втрату даних у відсотках
            loss_percentage = (loss / original_len) * 100
            print(f"Втрачено даних: {loss} символів ({loss_percentage:.2f}%).")
        else:
            print("Втрату даних не виявлено.")
        # Перевірка коректності розшифрування
        if original_message != decrypted_message:
            print("Оригінальне повідомлення не збігається з розшифрованим.")
        else:
            print("Розшифрування успішне. Оригінал і розшифроване повідомлення збігаються.")
    def analyze_message(original_message):
        # Статистика по символах
        letters = sum(c.isalpha() for c in original_message)
        digits = sum(c.isdigit() for c in original_message)
        spaces = sum(c.isspace() for c in original_message)
        total = len(original_message)
        print(f"Статистика повідомлення:")
        print(f"  Літери: {letters} ({(letters / total) * 100:.2f}%)")
        print(f"  Цифри: {digits} ({(digits / total) * 100:.2f}%)")
        print(f"  Пробіли: {spaces} ({(spaces / total) * 100:.2f}%)")
    # Тестування
    if __name__ == "__main__":
        original_message = "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt at
labore et dolore magna aliqua"  # Початкове повідомлення
        secret_key = random.randint(1, 100)  # Генерація випадкового секретного ключа
        # Час шифрування
        start_time = time.time()
        encrypted_message = encrypt(original_message, secret_key)  # Шифрування
        encryption_time = time.time() - start_time
        # Час дешифрування
        start_time = time.time()
        decrypted_message = decrypt(encrypted_message, secret_key)  # Дешифрування
        decryption_time = time.time() - start_time
        # Оцінка втрати даних
        evaluate_data_loss(original_message, decrypted_message)
        # Аналіз повідомлення
        analyze_message(original_message)
        # Часові показники
        print(f"Час шифрування: {encryption_time:.6f} сек.")
        print(f"Час дешифрування: {decryption_time:.6f} сек.")
```

Додаток Г

Опис програми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
_____

ПРОГРАММА ДОСЛІДЖЕННЯ АЛГОРИТМІВ ШИФРУВАННЯ ДАНИХ З ВИКОРИСТАННЯМ ЕЛІПТИЧНИХ КРИВИХ

Опис програми
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.1547-13-ЛЗ

Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
_____

Керівник розробки
_____ Тетяна ГРИШЕЧКІНА
_____

Виконавець
_____ Христина МАКАРОВА
_____

Нормоконтролер
_____ Світлана ВОЛКОВА
_____

2026

МІНСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО

44165850.1547-13

ПРОГРАММА ДОСЛІДЖЕННЯ АЛГОРИТМІВ ШИФРУВАННЯ ДАНИХ З ВИКОРИСТАННЯМ ЕЛІПТИЧНИХ КРИВИХ

Опис програми

Аркушів 7

2026

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ

Програма для реалізації двох алгоритмів шифрування даних із використанням еліптичних кривих: алгоритм Чандрасекхара та алгоритм Діффі-Геллмана. Вона призначена для дослідження ефективності шифрування, генерації ключів і захисту переданих даних.

Програмне забезпечення, необхідне для функціонування програми:

1) операційна система: Windows, Linux, або macOS;
2) мова програмування: Python (версія 3.8 або новіша);
3) додаткові бібліотеки Python:
   - ecdsa – для роботи з еліптичними кривими;
   - hashlib – для генерації хешів;
   - Cryptodome – для реалізації симетричного шифрування (AES) в алгоритмі Чандрасекхара;
   - стандартна бібліотека Python (random, os).

Програма написана мовою програмування Python з використанням її бібліотек для реалізації математичних обчислень, операцій над еліптичними кривими та шифрування даних.

2 ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Програма призначена для шифрування та розшифрування даних із використанням алгоритмів Чандрасекхара та Диффі-Геллмана на основі еліптичних кривих. Основними задачами є наступні:

1) генерація ключів (приватних і публічних);
2) використання еліптичних кривих для обчислення спільних секретів;
3) шифрування повідомлень із використанням симетричних алгоритмів;
4) розшифрування повідомлень.

Функціональними обмеженнями є такі:

− програма залежить від коректності введених параметрів еліптичної кривої (коефіцієнти a, b і модуль p);

− ефективність алгоритмів залежить від розміру поля еліптичної кривої (значення p);

− необхідність використання лише цілих чисел і простих чисел для параметрів кривої.

3 ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Загальний алгоритм програми для обох наявних алгоритмів наступний:

1) ініціалізація параметрів еліптичної кривої та базової точки G;
2) генерація приватних і публічних ключів для сторін;
3) обчислення спільного секретного ключа на основі приватного і публічного ключів;
4) шифрування повідомлення з використанням симетричного ключа;
5) розшифрування повідомлення за допомогою того самого секретного ключа.

Використані методи наступні:

- методи еліптичних кривих: додавання точок, подвоєння точок, обчислення спільного секрету;
- криптографічні методи: генерація ключів, симетричне шифрування (AES), хешування (SHA-256);
- алгоритм Диффі-Геллмана: обмін ключами на основі еліптичних кривих;
- алгоритм Чандрасекхара: використання додаткового шифрування з AES.

Структура програми є такою:

1) функції для роботи з еліптичними кривими:
   - generate_keys() – генерує пару ключів (приватний і публічний);
   - key_derivation() – обчислює симетричний ключ із секретної точки;
2) функції шифрування і розшифрування:
   - encrypt_message_aes() – шифрування повідомлення через AES;
   - decrypt_message_aes() – розшифрування повідомлення через AES;
   - encrypt() і decrypt() – основні функції шифрування і дешифрування;
3) головна програма:
   - уведення параметрів кривої, базової точки та повідомлення;
   - виклик функцій для шифрування/дешифрування повідомлення.

Програма не залежить від зовнішніх програм, але використовує бібліотеки Python для криптографічних операцій.

4 ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Типи електронних обчислювальних машин і пристроїв описані в таблиці Г.1.

Таблиця Г.1 – Використані технічні засоби

Засіб	Опис
Комп’ютер	Персональний комп’ютер або сервери з підтримкою Python 3.8+ (Windows, Linux, macOS)
Пам’ять	Необхідно не менше 1 ГБ оперативної пам’яті для ефективної роботи
Диск	Мінімальний розмір сховища – 10 МБ

5 ВИКЛИК І ЗАВАТАЖЕННЯ

Програма запускається з командного рядка або середовища розробки Python. Команда для запуску є python main.py. Замість main може бути назва файлу, як вказано в додатку В.

Вхідними точками є наступні:

- головна функція: if __name__ == "__main__":;
- усі параметри вводяться користувачем під час виконання.

6 ВХІДНІ ДАНІ

Характер і організація вхідних даних є наступною:

- параметри еліптичної кривої: a, b, модуль p, координати базової точки $G(x, y)$;
- повідомлення для шифрування (текстовий рядок).

Формат і спосіб кодування вхідних даних такий:

- числові параметри вводяться у десятковому форматі;
- повідомлення вводиться у вигляді текстового рядка (UTF-8).

7 ВИХІДНІ ДАНІ

Характер і організація вихідних даних наступні:

- зашифроване повідомлення (байтовий рядок);
- дешифроване повідомлення (текстовий рядок).

Формат і спосіб кодування вихідних даних такий:

- зашифроване повідомлення кодується у байтовому форматі для збереження або передачі;
- дешифроване повідомлення відображається у текстовому вигляді.

Додаток Д

Керівництво користувача

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
_____

ПРОГРАММА ДОСЛІДЖЕННЯ АЛГОРИТМІВ ШИФРУВАННЯ ДАНИХ З ВИКОРИСТАННЯМ ЕЛІПТИЧНИХ КРИВИХ

Керівництво користувача
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.1547-І3-ЛЗ

Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
_____

Керівник розробки
_____ Тетяна ГРИШЕЧКІНА
_____

Виконавець
_____ Христина МАКАРОВА
_____

Нормоконтролер
_____ Світлана ВОЛКОВА
_____

2026

МІНСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО

44165850.1547-І3

ПРОГРАММА ДЛЯ ДОСЛІДЖЕННЯ АЛГОРИТМІВ ШИФРУВАННЯ ДАНИХ З ВИКОРИСТАННЯМ ЕЛІПТИЧНИХ КРИВИХ

Керівництво користувача

Аркушів 7

2026

ЗМІСТ

ВСТУП

Програмні алгоритми призначені для шифрування та розшифрування даних з використанням алгоритмів на основі еліптичних кривих. Він дозволяє здійснювати безпечний обмін повідомленнями між користувачами, використовуючи методи шифрування, що базуються на алгоритмах Чандрасекхара та Діффі-Геллмана.

Програма дозволяє:

- генерувати приватні та публічні ключі для обох користувачів;
- виконувати обмін зашифрованими повідомленнями з використанням еліптичних кривих;
- використовувати алгоритм AES для шифрування даних;
- виконувати шифрування і розшифрування повідомлень із застосуванням методів еліптичних кривих і симетричного шифрування.

Для роботи з програмою користувач повинен мати базові знання з криптографії, розуміти принципи роботи еліптичних кривих і основи симетричного шифрування (AES). Досвід програмування не є необхідним, але буде корисним для налаштування програми і розуміння її роботи.

Перелік експлуатаційної документації, з якою необхідно ознайомитися користувачу:

- керівництво користувача (даний документ);
- документація з налаштування середовища програмування Python;
- інструкція з використання бібліотек ecdsa, Cryptodome для роботи з еліптичними кривими та шифруванням.

1 ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

Програма призначена для забезпечення безпечної передачі даних через канали зв'язку. Вона автоматизує процес шифрування і дешифрування повідомлень, що використовують еліптичні криві для обміну секретами, а також застосовує алгоритм AES для симетричного шифрування.

Умови застосування:

- програма повинна працювати на комп'ютерах з операційною системою Windows, Linux або macOS;
- необхідно мати встановлену середу Python версії 3.8 або вище;
- програма вимагає наявності бібліотек ecdsa, Cryptodome;
- для запуску програми потрібно мати доступ до консольного інтерфейсу Python.

2 ПІДГОТОВКА ДО РОБОТИ

Склад і зміст дистрибутивного носія даних повинен містити:

- основний скрипт main.py;
- залежності (пакети) для Python: ecdsa, Cryptodome;
- документація до програми (включаючи це керівництво).

Порядок завантаження даних і програм:

1) завантажте та розпакуйте дистрибутив у зручне місце;
2) установіть необхідні бібліотеки, як наведено на рисунку Д.1.

```
(.venv) PS C:\Users\     \PycharmProjects\chandrasekhar_elliptic> pip install ecdsa pycryptodome
Requirement already satisfied: ecdsa in c:\users\sasha\pycharmprojects\chandrasekhar_elliptic\.venv\lib\site-packages (0.19.0)
Requirement already satisfied: pycryptodome in c:\users\sasha\pycharmprojects\chandrasekhar_elliptic\.venv\lib\site-packages (3.21.0)
Requirement already satisfied: six>=1.9.0 in c:\users\sasha\pycharmprojects\chandrasekhar_elliptic\.venv\lib\site-packages (from ecdsa) (1.16.0)
(.venv) PS C:\Users\     \PycharmProjects\chandrasekhar_elliptic>
```

Рисунок Д.1 – Встановлення необхідних бібліотек

Порядок перевірки працездатності наступний:

- перевірте наявність всіх необхідних бібліотек;
- запустіть програму за допомогою команди, як показано на рисунку Д.2;
- переконайтеся, що програма коректно виконує шифрування і дешифрування повідомлень.

```
(.venv) PS C:\Users\     \PycharmProjects\chandrasekhar_elliptic> python algorithm.py
Введіть коефіцієнт a (ціле число): 3
Введіть коефіцієнт b (ціле число): 5
Закритий ключ: 111500525825202922462914955917764098886176996950575661164678759342793618494694
Відкритий ключ: (111215782272110852961520266361961590193725607053720169606932036741364625419333, 3
Введіть повідомлення для шифрування: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed
Зашифроване повідомлення: b'\xe9\x88\xf2\xe0\x95)\xc5\xa1E\x9c0\x8d#s\xb2&\xfaV\x18\xc7\x1f\xac9\x
1\x18M\xfa/\xe4\x0bW8\xbf&`\x8f`\xfah\xd3\xd7\x98y\xcf8\\>\xd6\x06\x98\xa7\x1aW\n@\x90\xe436\xb3\x
bd\xe6Sr\xffV\xba\xffc\xb7\x89\xa7\x98\xb4\xf9\x80qx\x9f2\x05\xda2\xe0%n\xd2%\xc1a\xe2^s'
Дешифроване повідомлення: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
(.venv) PS C:\Users\     \PycharmProjects\chandrasekhar_elliptic>
```

Рисунок Д.2 – Запуск програми

3 ОПИС ОПЕРАЦІЙ

Після запуску програми (рис. Д.2) виконуються функції і задачі:

1) генерація ключів:

− користувач вводить параметри еліптичної кривої (коефіцієнти a, b, просте число p та координати базової точки G);

− програма генерує приватні та публічні ключі для обох користувачів;

2) шифрування повідомлення:

− користувач вводить повідомлення для шифрування;

− програма генерує спільний секрет на основі ключів користувачів, використовує AES для шифрування повідомлення;

3) розшифрування повідомлення:

− користувач вводить зашифроване повідомлення;

− програма розшифровує повідомлення за допомогою спільного секрету, використовуючи той самий алгоритм AES.

4 АВАРІЙНІ СИТУАЦІЇ

Дії на випадок недотримання умов виконання технологічного процесу:

1) технічні відмови:

− якщо програма не запускається, перевірте наявність Python і необхідних бібліотек;

− у разі проблем із шифруванням або розшифруванням перевірте коректність введених даних (параметри кривої, координати точки G);

2) відмова носіїв або помилки в даних:

− якщо програма не може завантажити дані або файл, перевірте цілісність файлів;

3) несанкціоноване втручання в дані:

− якщо з'являється повідомлення про помилку під час шифрування або дешифрування, перевірте цілісність введених даних.

5 РЕКОМЕНДАЦІЇ ЩОДО ЗАСВОЄННЯ

Рекомендації щодо засвоєння та експлуатації:

− ознайомтеся з основами криптографії на еліптичних кривих та симетричного шифрування (AES);

− використовуйте програму для практики шифрування і розшифрування повідомлень;

− для запуску програми використовуйте консоль або IDE для Python.

Приклад контрольний:

− запуск програми та введення параметрів для еліптичної кривої: $a = 2, b = 3, p = 17, G = (5, 1)$;

− введення повідомлення для шифрування;

− програма зашифрує повідомлення і розшифрує його назад.

Додаток Е

Керівництво програміста

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
_____

ПРОГРАММА ДЛЯ ДОСЛІДЖЕННЯ АЛГОРИТМІВ ШИФРУВАННЯ ДАНИХ З ВИКОРИСТАННЯМ ЕЛІПТИЧНИХ КРИВИХ

Керівництво програміста
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.1547-33-ЛЗ

Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
_____

Керівник розробки
_____ Тетяна ГРИШЕЧКІНА
_____

Виконавець
_____ Христина МАКАРОВА
_____

Нормоконтролер
_____ Світлана ВОЛКОВА
_____

2026

МІНСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО

44165850.1547-33

ПРОГРАММА ДОСЛІДЖЕННЯ АЛГОРИТМІВ ШИФРУВАННЯ ДАНИХ З ВИКОРИСТАННЯМ ЕЛІПТИЧНИХ КРИВИХ

Керівництво програміста

Аркушів 6

2026

ЗМІСТ

1 ПРИЗНАЧЕННЯ Й УМОВИ ЗАСТОСУВАННЯ ПРОГРАМИ

Програма призначена для шифрування та розшифрування даних за допомогою криптографії на основі еліптичних кривих та симетричного шифрування за допомогою алгоритму AES. Вона автоматизує процес обміну зашифрованими повідомленнями між користувачами, забезпечуючи високий рівень безпеки передачі даних.

Функції програми є наступними:

- генерація ключів для двох користувачів на основі еліптичної кривої;
- шифрування та розшифрування даних за допомогою еліптичних кривих (алгоритм Діффі-Геллмана);
- шифрування і розшифрування повідомлень із використанням симетричного шифрування (AES);
- обробка введених даних (повідомлень) і виведення зашифрованих або розшифрованих повідомлень.

Умови для виконання програми наведені в таблиці Е.1.

Таблиця Е.1 – Мінімальні умови для виконання програми

Умова	Характеристика
Операційна система	Windows, Linux, macOS
Оперативна пам'ять	Не менше 2 ГБ
Програмне забезпечення	Python 3.8 або вище. Бібліотеки ecdsa, Cryptodome для роботи з криптографією
Периферійні пристрої	Для роботи програми не потрібні спеціальні периферійні пристрої, окрім стандартного монітора та клавіатури для вводу даних і перегляду результатів

2 ХАРАКТЕРИСТИКИ ПРОГРАМИ

Основні характеристики програми наведені в таблиці Е.2.

Таблиця Е.2 – Основні характеристики програми

Характеристика	Опис
Часові характеристики	Час генерування публічного та приватного ключа: до однієї секунди (залежно від параметрів еліптичної кривої). Час шифрування та розшифрування повідомлення: до кількох мілісекунд для стандартних повідомлень
Режим роботи	Програма працює в інтерактивному режимі через консоль або термінал. Взаємодія з користувачем відбувається через введення текстових команд та параметрів
Засоби контролю правильності виконання	Програма перевіряє правильність введених даних (наприклад, параметрів еліптичної кривої, повідомлень). Використовуються блоки обробки помилок для виявлення невірних вхідних даних або помилок під час шифрування/розшифрування
Самовідновлюваність	Програма не має механізмів самовідновлення, оскільки є консольною утилітою для однокрокових операцій (шифрування/розшифрування). Однак при виникненні помилок буде виведено відповідне повідомлення для корекції

3 ЗВЕРНЕННЯ ДО ПРОГРАМИ

Програма запускається через консоль або термінал за допомогою команди, яка наведена на рисунку Д.2 додатку Д. При запуску програма очікує введення необхідних параметрів для шифрування або розшифрування. Користувач може вибрати відповідну операцію, що потребує введення:

- для генерації ключів;
- для шифрування повідомлення;
- для розшифрування повідомлення.

Параметри передаються через консоль у вигляді аргументів або запитів, що вимагають вводу (наприклад, параметри еліптичної кривої, текст повідомлення). Кожен користувач повинен ввести публічні та приватні ключі, щоб здійснити шифрування або розшифрування.

4 ВХІДНІ Й ВИХІДНІ ДАНІ

Вхідними даними є параметри еліптичної кривої та повідомлення для шифрування та дешифрування. Всі дані (параметри еліптичної кривої та повідомлення) вводяться в текстовому форматі через консоль.

Зашифроване або розшифроване повідомлення у вигляді тексту. Результат виводиться у текстовому форматі, який може бути скопійований або збережений користувачем.

5 ПОВІДОМЛЕННЯ

Усі повідомлення, що можуть випливати при роботі програми наведені в таблиці Е.3.

Таблиця Е.3 – Повідомлення в алгоритмах

Повідомлення	Текст	Опис ситуації	Дії
Повідомлення про неправильний формат ключа	Невірний формат ключа	Користувач ввів невірний формат для публічного або приватного ключа	Перевірити правильність введеного ключа. Переконатися, що він відповідає вимогам програми
Повідомлення про успішне шифрування	Повідомлення успішно зашифровано	Повідомлення успішно зашифроване	Показати зашифроване повідомлення, що буде передано іншим користувачам для розшифрування
Повідомлення про успішне розшифрування	Повідомлення успішно розшифровано	Повідомлення успішно розшифроване	Показати оригінальний текст повідомлення
Повідомлення про помилку в параметрах еліптичної кривої	Помилка: невірні параметри еліптичної кривої	Введено невірні або некоректні параметри для еліптичної кривої	Перевірити параметри. Переконатися, що вони відповідають математичним вимогам для еліптичної кривої

Регламентованими діями програміста під час початкового налагодження можуть бути:

1) налагодження середовища встановленням мови програмування та бібліотек ecdsa та pycryptodome;

2) перевірити коректність роботи програми шляхами:

- запустити базовий тест для перевірки генерації ключів;
- перевірити шифрування та дешифрування простого тексту;
- виконати перевірку на помилки при введенні некоректних даних;

3) перевірити журнал помилок.

Додаток Ж

Відомість матеріалів кваліфікаційної роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Українського
державного університету
науки і технологій
_____ Анатолій РАДКЕВИЧ
_____

ПРОГРАММА ДЛЯ ДОСЛІДЖЕННЯ АЛГОРИТМІВ ШИФРУВАННЯ ДАНИХ З ВИКОРИСТАННЯМ ЕЛІПТИЧНИХ КРИВИХ

Відомість матеріалів кваліфікаційної роботи
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.1547-90-ЛЗ

Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
_____

Керівник розробки
_____ Тетяна ГРИШЕЧКІНА
_____

Виконавець
_____ Христина МАКАРОВА
_____

Нормоконтролер
_____ Світлана ВОЛКОВА
_____

2026

МІНСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО

44165850.1547-90

ПРОГРАММА ДОСЛІДЖЕННЯ АЛГОРИТМІВ ШИФРУВАННЯ ДАНИХ З ВИКОРИСТАННЯМ ЕЛІПТИЧНИХ КРИВИХ

Відомість матеріалів кваліфікаційної роботи

Аркушів 1

2026


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ НАУКИ І ТЕХНОЛОГІЙ
ABSTRACTS
OF THE XIX INTERNATIONAL CONFERENCE
«MODERN INFORMATION AND COMMUNICATION TECHNOLOGIES ON A TRANSPORT, IN INDUSTRY AND EDUCATION»
18-19, December, 2025
СУЧАСНІ ІНФОРМАЦІЙНІ ТА КОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ НА ТРАНСПОРТІ, В ПРОМИСЛОВОСТІ І ОСВІТІ
ПРИСВЯЧЕНО ПАМ'ЯТІ ПРОФЕСОРА ІГОРЯ ЖУКОВИЦЬКОГО
ТЕЗИ
XIX МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ
18-19 ГРУДНЯ 2025
ДНІПРО
2025

Дослідження та реалізація алгоритму шифрування даних з використанням еліптичних кривих

Макарова Х.Є., Гришечкіна Т.С., Український державний університет науки та технологій, Україна

Еліптичні криві є одним із найбільш актуальних та ефективних математичних інструментів, що застосовуються у сучасній криптографії для забезпечення конфіденційності, цілісності та автентичності даних. В умовах стрімкого розвитку інформаційних технологій та зростання обсягів передаваних даних особливого значення набувають криптографічні алгоритми, які поєднують високий рівень безпеки з оптимальними обчислювальними витратами.

Криптографія на основі еліптичних кривих (Elliptic Curve Cryptography, ECC) забезпечує еквівалентний рівень стійкості порівняно з класичними асиметричними алгоритмами, такими як RSA, але з використанням значно коротших ключів. Це дозволяє зменшити навантаження на пам’ять, прискорити процеси шифрування та дешифрування, а також підвищити ефективність роботи систем, особливо в умовах обмежених ресурсів.

Метою даної роботи є дослідження принципів побудови та практичної реалізації алгоритмів шифрування даних з використанням еліптичних кривих, а також аналіз їхніх переваг і обмежень у сучасних інформаційних системах.

У межах дослідження розглянуто математичні основи еліптичних кривих над скінченними полями, принципи формування відкритих і закритих ключів, а також механізми виконання криптографічних операцій. Особливу увагу приділено таким алгоритмам, як ECDH (Elliptic Curve Diffie–Hellman) для безпечного обміну ключами та ECDSA (Elliptic Curve Digital Signature Algorithm) для створення й перевірки цифрового підпису.

Процес реалізації алгоритмів ECC включає вибір криптографічно стійких параметрів кривої, генерацію ключів, виконання операцій над точками еліптичної кривої та інтеграцію алгоритмів у програмне середовище. Важливим аспектом є забезпечення захисту від побічних атак та використання перевірених стандартів.

Попри високий рівень безпеки, алгоритми на основі еліптичних кривих мають низку практичних викликів, зокрема складність математичної реалізації та залежність надійності від коректного вибору параметрів. Подальший розвиток ECC спрямований на оптимізацію обчислень, підвищення стійкості до нових типів атак та адаптацію до постквантових криптографічних вимог.

Таким чином, використання еліптичних кривих є перспективним напрямом розвитку криптографічних систем і відіграє ключову роль у забезпеченні інформаційної безпеки сучасних цифрових рішень.