Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра прикладної математики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Криптографічні алгоритми
та їх ефективність
(тема)

Виконав:
здобувач _2_ року навчання, групи _ПМм-23-1 _
Удовенко Д.Ю.
(прізвище, ініціали)
Спеціальність
113 Прикладна математика
(код і повна назва спеціальності)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма
Прикладна математика
(повна назва освітньої програми)
Керівник асист. Луханін В.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ПМ _____ Сидоров М.В.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет _інформаційно-аналітичних технологій та менеджменту
Кафедра _прикладної математики
Рівень вищої освіти _другий (магістерський)
Спеціальність _113 Прикладна математика
(код і повна назва)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма _Прикладна математика
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри ПМ _____
(підпис)
“ _25_ ” _листопада_ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві _Удовенку Дмитру Юрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи _Криптографічні алгоритми та їх ефективність

затверджена наказом по університету від 22 листопада 2024 р. № 1223 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 6 січня 2025 р.

3. Вихідні дані до роботи _математичні моделі криптографічних алгоритмів

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз предметної області
2. Вибір і обґрунтування методу розв’язання
3. Програмна реалізація
4. Результати обчислювального експерименту
5. Аналіз можливих застосувань

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

1. Актуальність теми роботи
2. Постановка задачі
3. Аналіз предметної області
4. Метод чисельного аналізу
5. Результати обчислювального експерименту

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Підбір та вивчення технічної літератури за темою роботи	25 листопада – 1 грудня 2024 р.	виконано
2	Вибір та обґрунтування методу	2 – 8 грудня 2024 р.	виконано
3	Розробка алгоритму і програми	9 – 22 грудня 2024 р.	виконано
4	Проведення аналітичних досліджень та розрахунків	23 – 29 грудня 2024 р.	виконано
5	Робота над текстом пояснювальної записки	30 грудня 2024 р. – 9 січня 2025 р.	виконано
6	Представлення роботи на рецензію в ЕК	10 січня 2025 р.	виконано

Дата видачі завдання _25_ листопада 2024 р.

Здобувач _____ _____
(підпис)

Керівник роботи _____ асист. Луханін В.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 51 с., 1 табл., 5 рис., 1 дод., 18 джерел.

КРИПТОГРАФІЧНИЙ АЛГОРИТМ, ПРОТОКОЛ, PYTHON.

Об’єкт дослідження – ефективність криптографічних алгоритмів RSA, AES та ECC у задачах за-хисту даних. Це включає різні характеристики ефективності алгоритмів, зокрема їх продуктивність, ресурсомісткість, швидкість обробки даних, енергоспоживання та стійкість до атак, а також оптимальні умови їх використання.

Мета роботи – аналіз ефективності криптографічних алгоритмів RSA, AES, ECC та їх програмна реалізація.

Методи дослідження – у кваліфікаційній роботі використовуються математичний апарат визначення теоретичної стійкості алгоритмів та експериментальне моделювання для перевірки стійкості до практичних атак і оцінки продуктивності.

У даній роботі було розглянуто криптографічні алгоритми AES, RSA та ECC, побудовано математичні моделі, проведено їх теоретичний аналіз та програмно реалізовано для аналізу часу виконання алгоритмів в залежності від розміру повідомлення та ключа а також споживання пам’яті.

ABSTRACT

Introductory note: 51 pages, 1 tables, 5 figures, 1 appendixes, 18 sources.

CRYPTOGRAPHIC ALGORITHM, PROTOCOL, PYTHON.

Object of research – is the effectiveness of the cryptographic algorithms RSA, AES and ECC in data protection tasks. This includes various characteristics of the effectiveness of the algorithms, in particular their productivity, resource intensity, data processing speed, energy consumption and resistance to attacks, as well as optimal conditions for their use.

Purpose of work – is to analyze the effectiveness of the cryptographic algorithms RSA, AES, ECC and their software implementation..

Methods of research – the qualification work uses the mathematical apparatus for determining the theoretical stability of the algorithms and experimental modeling to verify the resistance to practical attacks and evaluate the performance.

In this work, the cryptographic algorithms AES, RSA and ECC were considered, mathematical models were built, their theoretical analysis was carried out and software implementation was carried out to analyze the execution time of the algorithms depending on the message and key size, as well as memory consumption.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, ОДИНИЦЬ І ТЕРМІНІВ

SSH – протокол безпечного віддаленого доступу до серверів і мережевих пристрої;

TLS – протокол забезпечення безпеки передачі даних через мережу;

SSL – попередник протоколу TLS, який також забезпечує шифрування даних;

OSI – еталонна модель взаємодії відкритих систем, яка визначає рівні роботи мережі;

HTTP – протокол передачі гіпертексту для обміну даними в Інтернеті;

HTTPS – захищений протокол передачі гіпертексту з використанням TLS або SSL;

TCP – протокол управління передачею, що забезпечує надійний обмін даними між пристроями у мережі;

IP – протокол Інтернету, що використовується для адресації та маршрутизації пакетів даних у мережі.

ВСТУП

Актуальність теми. Актуальність зумовлена зростанням обсягу даних та потребою в їх захисті, що потребує ефективних криптографічних алгоритмів для безпечної передачі і зберігання інформації, особливо в умовах цифровізації персональних, фінансових і конфіденційних даних. Постійний розвиток обчислювальних потужностей збільшує можливості для впровадження складних алгоритмів, але також посилює ризики злому, що вимагає вибору найбільш ефективних і стійких до атак алгоритмів. Крім того, актуальність питання підкріплюється необхідністю оптимізації енергоспоживання для мобільних, IoT і вбудованих систем, які обмежені в ресурсах, і де використання менш ресурсомістких алгоритмів, таких як ECC, є важливим. Прискорений розвиток квантових обчислень також ставить під загрозу традиційні алгоритми, такі як RSA, підвищуючи актуальність дослідження альтернатив, стійких до квантових атак.

Мета і завдання кваліфікаційної роботи. Метою кваліфікаційної роботи є провести аналіз ефективності криптографічних алгоритмів RSA, AES, ECC та їх програмна реалізація. Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести огляд і аналіз сучасного стану задачі «аналіз ефективності криптографічних алгоритмів RSA, AES, ECC»;

– провести аналіз застосування криптографічних алгоритмів в інформаційних системах;

– побудувати математичну модель алгоритмів RSA, AES, ECC;

– ознайомитися з особливостями мови програмування Python та використання різних модулів;

– застосувати Python для розв’язання задачі та скласти відповідний алгоритм;

– програмно реалізувати алгоритми на основі побудованої математичної моделі;

– провести тестування та зробити аналіз результатів.

Об'єктом дослідження є ефективність криптографічних алгоритмів RSA, AES та ECC у задачах за-хисту даних. Це включає різні характеристики ефективності алгоритмів, зокрема їх продуктивність, ресурсомісткість, швидкість обробки даних, енергоспоживання та стійкість до атак, а також оптимальні умови їх використання.

Предметом дослідження є основні характеристики криптографічних алгоритмів RSA, AES та ECC, за якими оцінюються параметри та показники ефективності.

Методи дослідження. У кваліфікаційній роботі використовуються математичний апарат визначення теоретичної стійкості алгоритмів та експериментальне моделювання для перевірки стійкості до практичних атак і оцінки продуктивності.

Публікації. Результати, отримані у кваліфікаційній роботі, представлені на 13-й Міжнародній науково-технічній конференції «Інформаційні системи та технології ІСТ-2024» (м. Харків, 26-28 листопада 2024 р.) [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Місце криптографії в сучасному світі

Криптографія – це метод захисту інформації шляхом використання закодованих алгоритмів, хешів та підписів. Інформація може знаходитися на етапі зберігання (наприклад, файл на жорсткому диску), передачі (наприклад, електронний зв'язок між двома або декількома сторонами) або використання (при застосуванні для обчислень). Криптографія має чотири основні цілі:

а) конфіденційність – надає доступ до інформації лише авторизованим користувачам;

б) цілісність – гарантує, що з інформацією не проводилися маніпуляції;

в) справжність – підтверджує справжність інформації чи особистість користувача;

г) забезпечення неможливості відмови – позбавляє можливості заперечувати колишні зобов'язання чи дії.

Відносно інформаційних систем застосовуються інші категорії:

а) надійність – гарантія того, що система поводиться в нормальному і позаштатному режимах так, як заплановано;

б) точність – гарантія точного і повного виконання всіх команд;

в) контроль доступу – гарантія того, що різні групи осіб мають різний доступ до інформаційних об'єктів, і ці обмеження доступу постійно виконуються;

г) контрольованість – гарантія того, що в будь-який момент може бути проведена повноцінна перевірка будь-якого компонента програмного комплексу;

д) контроль ідентифікації – гарантія того, що клієнт, підключений в даний момент до системи, є саме тим, за кого себе видає;

е) стійкість до навмисних збоїв – гарантія того, що при навмисному внесенні помилок в межах заздалегідь обговорених норм система буде вести себе так, як обумовлено заздалегідь.

Криптографія використовує деякі низькорівневі криптографічні алгоритми для досягнення однієї або кількох із цих цілей інформаційної безпеки. Серед цих інструментів – алгоритми шифрування, алгоритми цифрового підпису, алгоритми хешування та інші функції. У цьому розділі будуть описані деякі з найпоширеніших низькорівневих криптографічних алгоритмів.

Шифрування сягає корінням у передачу конфіденційної інформації між військовими та політичними фігурами. Повідомлення можна зашифрувати так, щоб вони виглядали для всіх, окрім передбачуваного одержувача, як випадковий текст.

Сьогодні початкові технології шифрування ретельно зламані. Вони зламані настільки, що використовуються лише у розділах загадок у деяких газетах. На щастя, у цій галузі здійснено значні кроки у бік підвищення безпеки, а алгоритми, що використовуються сьогодні, засновані на ретельному аналізі та математичних методах.

У міру підвищення безпеки галузь криптографії стала охоплювати ширше коло цілей безпеки. Це автентифікація повідомлень, цілісність даних, безпечні обчислення та інші цілі.

Криптографія покладена у фундамент сучасного суспільства. На ній засновані незліченні інтернет-програми, що працюють за протоколом безпечної передачі гіпертексту (HTTPS), безпечний текстовий і голосовий зв’язок і навіть цифрові валюти.

Алгоритм шифрування – це процедура, яка перетворює повідомлення у форматі неформатованого тексту на зашифрований текст. Сучасні алгоритми використовують складні математичні обчислення та один або кілька ключів шифрування. Завдяки цьому можна відносно легко зашифрувати повідомлення, але неможливо розшифрувати його, не знаючи ключів.

Залежно від того, як діють ключі, технології шифрування поділяються на дві категорії: симетричні та асиметричні.

Алгоритми шифрування з симетричним ключем використовують одні й ті самі криптографічні ключі для шифрування простого тексту і розшифровки за-

шифрованого. У разі використання симетричного шифрування всі одержувачі повідомлення повинні мати доступ до спільного ключа.

Асиметрична (з відкритим ключем) криптографія охоплює широке коло алгоритмів. Вони ґрунтуються на математичних завданнях, які відносно легко вирішити в одному напрямку та складно у протилежному.

Одним із найвідоміших прикладів таких завдань є проблема розкладання числа на множники: для чітко підібраних простих чисел p і q ми можемо швидко обчислити добуток $N = p \cdot q$, але, якщо дано лише N, то відновити p і q дуже складно.

Широко використовується такий криптографічний алгоритм із відкритим ключем на основі завдання розкладання на множники, як функція Рівеста-Шаміра-Адлемана (RSA). У поєднанні з відповідною схемою заповнення можна використовувати RSA у багатьох цілях, зокрема асиметричного шифрування.

Схеми цифрових підписів – це тип криптографії з відкритим ключем, що гарантує цілісність, справжність та забезпечення неможливості відмови.

Процес підписання можна сприймати як шифрування файлу за допомогою приватного ключа. Особа, яка підписує цифровий документ, наприклад файл або фрагмент коду, використовує для створення підпису свій приватний ключ.

Цей підпис є унікальним для пари документ-приватний ключ і може прикріплюватися до документа та перевірятися з використанням відкритого ключа особи, яка ставить підпис. Двома поширеними алгоритмами цифрового підпису є RSA з ймовірнісною схемою підпису (RSA-PSS) та алгоритм цифрового підпису (DSA).

Еліптична криптографія (ECC) – це технологія криптографії з відкритим ключем, що базується на математичній теорії еліптичних кривих.

Найбільшою перевагою ECC є те, що вона може забезпечити рівень безпеки, подібний до більш традиційних технологій, з меншими ключами та швидшою роботою. Завдяки своїй ефективності ECC добре підходить для використання у пристроях із відносно низькою обчислювальною потужністю, наприклад, для мобільних телефонів.

ECC можна використовувати для ефективного обміну ключами за допомогою варіанта протоколу Діффі-Хеллмана (ECDH) на еліптичних кривих або для цифрових підписів з використанням алгоритму цифрових підписів на еліптичних кривих (ECDSA). Завдяки своїй швидкості та гнучкості ECC широко використовується в багатьох інтернет-додатках.

Криптографічна хеш-функція – це інструмент для перетворення довільних даних на «відбиток» фіксованої довжини. Хеш-функції створюються таким чином, щоб було складно знайти два різні набори вхідних даних, що дають той самий відбиток, і щоб було складно знайти повідомлення, відбиток якого збігається з фіксованим значенням.

На відміну від схем шифрування, схем підписів та MAC, хеш-функції не мають ключа. Хто завгодно може обчислити хеш для даного вхідного значення, і хеш-функція завжди буде генерувати те саме вихідне значення для одного і того ж вхідного значення.

Хеш-функції є важливим конструктивним елементом великих криптографічних алгоритмів та протоколів. Це алгоритми цифрових підписів, алгоритми виділених MAC, протоколи аутентифікації та сховище паролів.

Криптовалюта – це цифрова валюта, під час якої транзакції підтверджуються і записи ведуться децентралізованою системою, а чи не централізованим органом. Криптовалюта є прикладом практичного застосування криптографії.

Криптовалюта використовує безліч низькорівневих криптографічних алгоритмів для створення довіреної та надійної платформи. Криптовалюта використовує багато концепцій, що згадуються у цьому розділі: еліптичну криптографію, цифрові підписи, хеш-функції та інші концепції. У сукупності ці алгоритми забезпечують довіру та відповідальність без централізованого органу.

Представлені на даний момент інструменти дозволяють застосовувати шифрування при зберіганні даних і при їх передачі. Традиційно дані розшифровувалися перед використанням у обчисленнях. Криптографічні обчислення заповнили цю прогалину, надавши інструменти для роботи безпосередньо з даними, захищеними криптографією.

Термін «криптографічні обчислення» охоплює широкий діапазон технологій, у тому числі безпечні багатосторонні обчислення, гомоморфне шифрування та шифрування з можливістю пошуку. Незважаючи на відмінності в подробицях реалізації, ці технології забезпечують криптографічну безпеку даних з можливістю проводити обчислення з використанням захищених даних, зберігаючи при цьому їх конфіденційність.

1.2 Використання криптографії в інформаційних системах

1.2.1 Протоколи SSL та TLS і їх використання на веб-сервері

Останнім часом все більше уваги привертають протоколи TLS та SSL. Вони набувають популярності разом з поширенням цифрових сертифікатів. З’явилися навіть організації, які надають такі сертифікати безкоштовно всім охочим, щоб забезпечити шифрування даних між веб-сторінками та браузерами користувачів. Це важливо для безпеки в мережі, оскільки дозволяє запобігти несанкціонованому доступу до даних, що передаються від сервера до клієнта і назад.

SSL (Secure Socket Layer) – це протокол, який забезпечує захист сокетів. TLS (Transport Layer Security) – протокол, що забезпечує безпеку на транспортному рівні. Спочатку була розроблена технологія SSL, а пізніше на основі її специфікацій з’явився TLS, створений на основі SSL 3.0 компанією Netscape Communications. Обидва ці протоколи мають спільну мету: гарантувати захищену передачу даних між двома пристроями в мережі. Їх використовують для захисту даних на різних сервісах, таких як веб-сайти, електронна пошта, обмін повідомленнями та інших.

Захист переданих даних досягається за рахунок шифрування та автентифікації інформації. Загалом, TLS і SSL працюють схожим чином, і різниці в них небагато. TLS можна вважати «наступником» SSL, але обидва протоколи можуть працювати одночасно на одному сервері. Це дозволяє підтримувати робо-

ту як з новими пристроями та браузерами, так і зі старішими, які підтримують тільки SSL.

Хронологія виникнення цих протоколів виглядає ось так:

а) SSL 1.0 – не було опубліковано;

б) SSL 2.0 – 1995 рік;

в) SSL 3.0 – 1996 рік;

г) TLS 1.0 – 1999 рік;

д) TLS 1.1 – 2006 рік;

е) TLS 1.2 – 2008 рік;

є) TLS 1.3 – 2018 рік.

Як вже зазначалося, принцип роботи SSL і TLS є схожим. Вони створюють зашифрований канал поверх протоколу TCP/IP, через який передаються дані за допомогою прикладних протоколів, таких як HTTP, FTP тощо. Наочно це можна представити у вигляді рисунку 1.1.

Рисунок 1.1 – Протокол SSL/TLS

Прикладний протокол передається в «оболонці» TLS/SSL, а цей шар, у свою чергу, працює поверх TCP/IP. Таким чином, дані, що передаються через прикладний протокол, проходять через TCP/IP у зашифрованому вигляді. Лише пристрій, який встановив з'єднання, може розшифрувати передану інформацію, і для будь-кого іншого пакети не матимуть сенсу без відповідного ключа для дешифрування.

Процес встановлення захищеного з’єднання складається з кількох етапів.

а) Клієнт звертається до сервера для встановлення захищеного підключення за протоколом TLS або SSL. Це може здійснюватися або через підключення до порту, спеціально призначеного для SSL/TLS (наприклад, порт 443), або шляхом запиту на захищене з’єднання після звичайного підключення.

б) Після з’єднання клієнт надсилає серверу перелік доступних алгоритмів шифрування. Сервер порівнює їх зі своїм набором підтримуваних алгоритмів і вибирає найбільш захищений варіант, після чого повідомляє клієнту про свій вибір.

в) Сервер передає клієнту свій цифровий сертифікат, підписаний центром сертифікації, а також відкритий ключ.

г) Клієнт може перевірити дійсність сертифіката, зв’язавшись з центром сертифікації, який його видав. Однак це не є обов’язковим, оскільки операційні системи часто вже містять кореневі сертифікати, за допомогою яких браузери можуть автоматично перевіряти підписи серверних сертифікатів.

д) Для захищеного з’єднання створюється сеансовий ключ наступним чином:

– клієнт генерує випадкову цифрову послідовність;

– клієнт шифрує її за допомогою відкритого ключа сервера і надсилає на сервер;

– сервер дешифрує послідовність за допомогою свого приватного ключа.

Оскільки для шифрування використовується асиметричний алгоритм, лише сервер, маючи приватний ключ, здатен дешифрувати дані. У цьому методі застосовуються два ключі: відкритий для шифрування і приватний для дешифрування. Дешифрувати повідомлення, зашифроване відкритим ключем, можна лише за допомогою відповідного приватного ключа.

Отже, встановлюється захищене з’єднання, в якому дані шифруються та дешифруються, доки з’єднання не буде розірвано. Як було згадано раніше, кореневий сертифікат належить авторизаційному центру і підтверджує достовірність підписаного сертифіката, що гарантує належність сертифіката до відпові-

дного сервісу. У сертифікаті міститься інформація про сервер, якому його видано, а також термін його дії. Після завершення терміну дії сертифікат вважається недійсним.

Щоб отримати підписаний серверний сертифікат, необхідно створити запит на підпис (Certificate Sign Request, CSR) і надіслати його до авторизаційного центру. Центр поверне підписаний сертифікат, який можна встановити на сервер. Спочатку генерується ключ для шифрування, а потім на його основі формується CSR-файл.

Клієнтський сертифікат може бути створений як для пристроїв, так і для користувачів. Зазвичай, такі сертифікати застосовуються при двосторонній верифікації, коли клієнт підтверджує справжність сервера, а сервер у відповідь підтверджує справжність клієнта. Цей процес називається двосторонньою автентифікацією (mutual authentication) і дозволяє значно підвищити рівень безпеки порівняно з односторонньою автентифікацією. Крім того, двостороння автентифікація може замінити використання логіна та пароля.

1.2.2 Протокол SSH

Secure Shell (SSH) – це популярний транспортний протокол, який забезпечує захищене з’єднання між клієнтом і сервером. Він слугує основою для безпечного доступу до інфраструктури. Нижче описано процес «рукостискання», який здійснюється перед створенням захищеного каналу та шифруванням всього трафіку між клієнтом і сервером.

Процес рукостискання розпочинається з обміну версіями, де кожна зі сторін надсилає іншій рядок із номером своєї версії. Хоча цей етап не включає спеціальних дій, слід зазначити, що більшість сучасних клієнтів і серверів підтримують лише SSH 2.0 через недоліки дизайну версії 1.0.

Під час обміну ключами (відомого як KEX) сторони обмінюються загальнодоступними даними, на основі яких обчислюється спільний секрет, що вико-

ристовується обома сторонами для захисту з’єднання. Цей секрет не може бути розкритий на основі публічної інформації.

Обмін ключами починається з того, що клієнт і сервер надсилають один одному повідомлення SSH_MSG_KEX_INIT, в якому міститься список підтримуваних криптографічних алгоритмів у порядку пріоритету. Ці алгоритми забезпечують основні механізми для обміну ключами та подальшого повного шифрування переданих даних.

Оскільки обидві сторони застосовують однаковий алгоритм для вибору криптографічних примітивів із підтримуваного списку, після ініціалізації можна відразу переходити до обміну ключами. Наприклад, у протоколі еліптичних кривих Діффі-Хеллмана (Elliptic Curve Diffie-Hellman, ECDH) процес починається з того, що клієнт створює тимчасову пару ключів (закритий ключ і пов’язаний із ним відкритий ключ) і надсилає серверу свій відкритий ключ у повідомленні SSH_MSG_KEX_ECDH_INIT як це було зображено на рисунку 1.2.

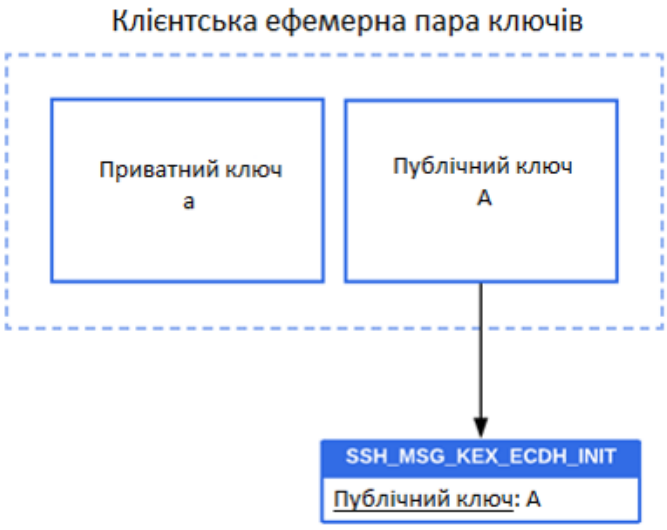

Рисунок 1.2 – Генерація повідомлення ініціалізації обміну ключами

Важливо зазначити, що ця пара ключів є тимчасовою: вона використовується виключно для обміну ключами та знищується після завершення процедури. Це значно ускладнює можливі атаки, коли зловмисник намагається пасивно

перехопити зашифрований трафік, сподіваючись у майбутньому отримати закритий ключ. Неможливо викрасти те, що вже видалено. Така властивість називається прямою секретністю (forward secrecy).

Сервер чекає на повідомлення SSH_MSG_KEX_ECDH_INIT, а після його отримання генерує власну тимчасову пару ключів. Використовуючи відкритий ключ клієнта та власну пару ключів, сервер обчислює спільний секрет K.

Далі сервер формує хеш обміну, який можна назвати H, і підписує його, створюючи підписаний хеш HS. Цей хеш обміну та його підпис виконують кілька важливих функцій:

– оскільки хеш обміну містить загальний секрет, він підтверджує, що обидві сторони успішно згенерували цей спільний секрет;

– цикл підпису та перевірки хеша і підпису обміну дозволяє клієнту переконатися, що сервер дійсно володіє приватним ключем хоста, підтверджуючи, що клієнт підключається саме до потрібного сервера (за умови, що клієнт довіряє відповідному відкритому ключу);

– підписуючи хеш замість вихідних даних, значно зменшується обсяг підписаних даних, що пришвидшує процес рукостискання.

Хеш обміну створюється шляхом обчислення хеша (SHA256, SHA384 або SHA512, залежно від обраного алгоритму обміну ключами) на основі таких даних:

а) спеціальний рядок, що містить версії клієнта і сервера, повідомлення клієнта SSH_MSG_KEXINIT, а також повідомлення сервера SSH_MSG_KEXINIT;

б) відкритий ключ (або сертифікат) сервера, позначений як HPub. Цей ключ, пов’язаний із закритим ключем HPriv, зазвичай створюється під час запуску процесу, а не при кожному рукостисканні;

в) відкритий ключ клієнта A;

г) відкритий ключ сервера B;

д) загальний секрет K.

На основі цієї інформації сервер може згенерувати повідомлення

SSH_MSG_KEX_ECDH_REPLY, яке містить тимчасовий відкритий ключ сервера B, відкритий ключ хоста сервера HPub та підпис хеша обміну HS.

Як тільки клієнт отримує SSH_MSG_KEX_ECDH_REPLY від сервера, він має всі необхідні дані для обчислення спільного секрету K і хеша обміну H. На завершальному етапі клієнт отримує відкритий ключ хоста або сертифікат із SSH_MSG_KEX_ECDH_REPLY та перевіряє підпис хеша обміну HS, що підтверджує право сервера на володіння закритим ключем хоста. Щоб уникнути атак «людина посередині» (MitM), після перевірки підпису клієнт перевіряє відкритий ключ хоста або сертифікат у своїй локальній базі відомих хостів; якщо ключ або сертифікат не виявляється довіреним, з’єднання переривається.

SSH-клієнт може запропонувати додати цей ключ хоста до локальної бази відомих хостів, що в OpenSSH зберігається зазвичай у файлі ~/.ssh/known_hosts.

На цьому етапі обидві сторони узгодили криптографічні примітиви, обмінялися секретами та створили ключовий матеріал для обраних методів шифрування. Тепер між клієнтом і сервером встановлюється захищений канал, який гарантує конфіденційність і цілісність даних.

Таким чином, процес рукостискання SSH дозволяє налаштувати захищене з’єднання між клієнтом і сервером.

1.3 Змістовна та формальна постановка задачі

1.3.1 Змістовна постановка задачі

Проаналізувати найбільш популярні криптографічні алгоритми та їх використання в інформаційних системах, побудувати їх математичні моделі, оцінити ефективність та надійність, а також програмно реалізувати для більш детального дослідження.

1.3.2 Формальна постановка задачі

Поставлена задача полягає в аналізі ефективності криптографічних алгоритмів RSA, AES та ECC за кількома ключовими параметрами: швидкодія, використання ресурсів, масштабованість та здатність протистояти сучасним атакам. Цей аналіз дозволяє визначити найкращі умови для використання кожного з алгоритмів у відповідних системах та сценаріях захисту даних. Формальна постановка задачі аналізу криптографічних алгоритмів RSA, AES та ECC може бути описана наступним чином:

а) вхідні дані та вимоги:

1) параметри алгоритмів: математичні параметри, які використовуються для роботи RSA, AES та ECC;

2) тестові дані: спеціально підготовлені вхідні та вихідні дані для перевірки стійкості до різних типів атак;

3) типи атак: перелік атак, для яких алгоритми будуть аналізуватися (атаки на факторизацію та атаки на обчислення приватного ключа, атаки диференціального та лінійного криптоаналізу, атаки на дискретний логарифм для еліптичної кривої);

4) безпека: мінімальний рівень криптостійкості, який алгоритм повинен забезпечувати для захисту даних;

5) продуктивність: час, необхідний для виконання операцій шифрування, розшифрування та генерації ключів;

б) задача аналізу криптографічних алгоритмів RSA, AES та ECC може бути сформульована як задача оптимізації:

1) задача визначення мінімальної обчислювальної складності для зламу RSA, AES та ECC;

2) задача оцінки швидкодії: мінімізація часу обчислень для операцій шифрування/розшифрування та генерації ключів;

3) задача оцінки криптостійкості: для кожного алгоритму визначити поріг складності, при якому він перестає забезпечувати належний рівень захисту;

в) критерії оцінки:

1) складність факторизації довгих чисел;

2) необхідна кількість операцій для атаки грубої сили на ключ;

3) складність розв’язання задачі дискретного логарифма на обраній еліптичній кривій;

4) оцінка швидкості виконання алгоритмів шифрування/розшифрування та генерації ключів в умовах з різними обсягами вхідних даних;

г) методи аналізу:

1) математичний апарат визначення теоретичної стійкості алгоритмів;

2) експериментальне моделювання для перевірки стійкості до практичних атак і оцінки продуктивності.

Очікуваним результатом аналізу є визначення умов, за яких кожен з алгоритмів демонструє оптимальну ефективність та рекомендації щодо застосування алгоритмів у відповідних умовах і середовищах, враховуючи продуктивність, безпеку та обчислювальну складність.

1.4 Постановка задач дослідження

Виходячи з проведеного аналізу предметної області, можна дійти до висновку, що задача програмної реалізації криптографічних алгоритмів RSA, ECC та AES буде розв’язуватися за допомогою мови програмування Python. Це популярна мова для реалізації криптографічних алгоритмів завдяки зрозумілому синтаксису, що спрощує читання і підтримку коду. Велика кількість криптографічних бібліотек, як-от PyCryptodome, cryptography та hashlib, забезпечують доступ до популярних алгоритмів шифрування і хешування, що робить розробку безпечнішою та зручнішою. Python також дозволяє швидко створювати прототипи і легко інтегрувати їх у більші проєкти, що суттєво прискорює розробку, особливо на етапі тестування. Завдяки підтримці об’єктно-орієнтованого програмування, криптографічні алгоритми можна легко організувати в класи для

повторного використання коду. Популярність мови та велика спільнота розробників також сприяють активній підтримці, доступу до ресурсів і швидкому вирішенню можливих питань. Усе це робить Python оптимальним вибором для реалізації криптографічних алгоритмів, особливо коли важливі зручність і швидкість розробки.

Отже метою кваліфікаційної роботи є аналіз та програмна реалізація криптографічних алгоритмів RSA, ECC та AES з використанням мови Python для поглиблення знань та вдосконалення практичних навичок у сфері аналізу та застосування криптографічних алгоритмів для забезпечення інформаційної безпеки, а також підвищення рівня компетентності у використанні сучасних методів та інструментів для дослідження ефективності алгоритмів RSA, AES та ECC.

Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести аналіз застосування криптографічних алгоритмів в інформаційних системах;

– побудувати математичну модель алгоритмів RSA, AES, ECC;

– ознайомитися з особливостями мови програмування Python та використання різних модулів;

– застосувати Python для розв’язання задачі та скласти відповідний алгоритм;

– програмно реалізувати алгоритми на основі побудованої математичної моделі;

– провести тестування та зробити аналіз результатів.

2 ВИБІР ТА ОБҐРУНТУВАННЯ МЕТОДУ РОЗВ'ЯЗАННЯ

2.1 Побудова математичної моделі криптографічних алгоритмів

RSA – один із найпопулярніших асиметричних алгоритмів. Його математична модель виглядає так:

а) обираємо два великі прості числа p і q;

б) обчислюємо їх добуток $N = p \cdot q$, що є модулем для шифрування і дешифрування;

в) вибираємо публічний ключ e, такий що e і $(p-1)(q-1)$ – взаємно прості;

г) Обчислюємо приватний ключ d, такий що $e \cdot d \equiv 1\left(\mathrm{mod}(p-1)(q-1)\right)$;

д) для шифрування повідомлення M, яке менше за N, обчислюємо:

$$C = M^e \bmod N; \tag{2.1}$$

е) для дешифрування за допомогою приватного ключа обчислюємо:

$$M = C^d \bmod N. \tag{2.2}$$

AES – це симетричний блоковий шифр, що працює з блоками даних фіксованого розміру (128 біт) і використовує ключі розміру 128, 192 або 256 біт. Основними операціями AES є перестановка байтів, заміна байтів і побітові операції XOR:

а) вхідні дані – блок відкритого тексту PPP довжиною 128 біт. Ключ шифрування KKK довжиною 128, 192 або 256 біт;

б) ключ K розширюється для створення множини підключів $K_0, K_1, \ldots, K_n$, де n – кількість раундів шифрування (10 для AES-128, 12 для AES-192, і 14 для AES-256);

в) шифрування – шифрування складається з кількох раундів. Кожен раунд виконує наступні операції:

1) AddRoundKey – виконання побітової операції XOR між блоком

відкритого тексту P і підключем K_i:

$$P' = P \oplus K_i; \tag{2.3}$$

2) SubBytes – заміна кожного байта блоку P' за допомогою таблиці заміщення (S-Box):

$$P'' = SubBytes\left(P'\right); \tag{2.4}$$

3) ShiftRows – зсув рядків у матриці байтів P'':

$$P''' = ShiftRows\left(P''\right); \tag{2.5}$$

4) MixColumns – перетворення стовпців блоку P''' з використанням операцій над елементами поля $GF\left(2^8\right)$:

$$P'''' = MixColumns\left(P'''\right); \tag{2.6}$$

г) остаточний раунд (без MixColumns) – після останнього раунду отримуємо зашифрований блок C:

$$C = AES\left(P, K\right), \tag{2.7}$$

де AES – основна формула шифрування AES;

P, K, C – блок відкритого тексту, ключ, шифротекст відповідно;

д) дешифрування – тут виконується зворотний процес: заміна S-Box, зворотний зсув рядків, зворотне перетворення стовпців і застосування підключів.

ECC базується на використанні еліптичних кривих над скінченними по-

лями. Безпека ECC ґрунтується на складності обчислення дискретного логарифма на еліптичних кривих.

1. Еліптична крива задається рівнянням:

$$y^2 = x^3 + ax + b \bmod p, \tag{2.8}$$

де a, b – параметри кривої;

p – просте число (скінченне поле $GF(P)$).

2. Точки на кривій: множина всіх точок (x, y), що задовольняють рівняння кривої, разом з точкою на нескінченності, утворюють групу з операцією додавання точок.

3. Операції на кривій та додавання точок. Якщо є дві точки $P = (x_1, y_1)$ і $Q = (x_2, y_2)$, то сума $R = P + Q$ визначається як нова точка на кривій. Формули для додавання залежать від координат точок. Множення точки на скаляр: Операція множення точки на скаляр d – це додавання точки PPP до себе d разів:

$$Q = d \cdot P. \tag{2.9}$$

4. Приватний ключ d – випадкове число в діапазоні від 1 до $n-1$, де n – порядок кривої. Публічний ключ Q – точка на кривій, обчислена як:

$$Q = d \cdot G, \tag{2.10}$$

де G – фіксована початкова точка (генератор групи).

5. Шифрування (ECC ElGamal). Відправник обирає випадкове число k і обчислює пару точок:

$$C_1 = k \cdot G, \tag{2.11}$$

$$C_2 = M + k \cdot Q, \tag{2.12}$$

де M – точка, що представляє повідомлення.

6. Дешифрування – одержувач використовує свій приватний ключ d, щоб обчислити:

$$M = C_2 - d \cdot C_1. \tag{2.13}$$

Це дозволяє відновити точку M, яка представляє зашифроване повідомлення.

2.2 Порівняльний аналіз алгоритмів

Для більшої зручності наведемо результати порівняння з теоретичних відомостей у вигляді таблиці 2.1.

Таблиця 2.1 – Порівняльний аналіз алгоритмів

Алгоритм	AES (Advanced Encryption Standard)	ECC (Elliptic Curve Cryptography)	RSA (Rivest-Shamir-Adleman)
Тип алгоритму	Симетричний (блоковий шифр)	Асиметричний (шифрування на основі еліптичних кривих)	Асиметричний (шифрування на основі факторизації чисел)
Призначення	Шифрування великих обсягів даних (конфіденційність)	Захист ключів і цифрові підписи (конфіденційність, автентифікація)	Захист ключів і цифрові підписи (конфіденційність, автентифікація)

Кінець таблиці 2.1

Алгоритм	AES (Advanced Encryption Standard)	ECC (Elliptic Curve Cryptography)	RSA (Rivest-Shamir-Adleman)
Розмір ключа	128, 192 або 256 біт	160-512 біт (залежно від рівня безпеки)	1024, 2048, 4096 біт і більше
Математична основа	Перестановки, підстановки, побітові операції, матричні операції	Дискретний логарифм на еліптичних кривих	Обчислювальна складність факторизації великих чисел
Швидкість роботи	Дуже швидкий для шифрування великих блоків даних (особливо на апаратному рівні)	Дуже ефективний для шифрування невеликих даних і створення цифрових підписів	Повільний порівняно з ECC і AES, особливо при великих ключах
Розмір блоку	128 біт	Не фіксований (залежить від поля і кривої)	Зазвичай дорівнює довжині ключа (1024 біт або більше)
Ефективність	Висока ефективність для шифрування великих обсягів даних	Менший розмір ключа для забезпечення того ж рівня безпеки, що й RSA	Низька ефективність при великих ключах (чим більший ключ, тим повільніший алгоритм)
Безпека	Висока безпека при правильній реалізації (наприклад, з ключем 256 біт AES дуже стійкий)	Висока безпека при меншій довжині ключа, ніж RSA (напр., ECC з 256-бітним ключем так само безпечний, як RSA з 3072-бітним ключем)	Висока безпека, але великі ключі роблять його повільним і менш ефективним у сучасних умовах

2.3 Алгоритм подальшого аналізу

Аналіз криптографічних алгоритмів та програмна реалізація вимагає досить складного підходу та ретельного планування, щоб забезпечити якісний результат. Ось кроки, які пропонуються для розв’язання цієї задачі:

Визначення цілей аналізу. Аналіз криптографічних алгоритмів є важливою частиною забезпечення інформаційної безпеки та проводиться з метою оцінки надійності методів шифрування та їх здатності протистояти потенційним атакам. До цілей аналізу входить оцінка продуктивності алгоритму: швидкості його роботи, ресурсозатратності та ефективності при обробці великих обсягів даних. Це особливо важливо для реальних систем, де алгоритми мають забезпечувати безпеку без значного впливу на продуктивність.

Вивчення особливостей мови програмування Python та його модулів. Python є одним із найпопулярніших мов програмування для реалізації криптографічних алгоритмів завдяки своїй читабельності та великій кількості бібліотек для криптографії. З його допомогою можна реалізовувати як прості криптографічні операції, так і складні алгоритми захисту даних, використовувані в сучасних застосунках. PyCryptodome є однією з найбільш широко використовуваних бібліотек для криптографічних операцій у Python. Вона забезпечує реалізацію багатьох криптографічних алгоритмів, таких як AES, RSA, ECC та інші.

Тестування та відладка. Після програмної реалізації необхідно виконати тестування та відладку для того, щоб переконатися у її коректному функціонуванні та відсутності помилок.

Ці кроки можуть бути розширені та деталізовані залежно від вимог до аналізу. Детальний аналіз ефективності криптографічних алгоритмів є складним та тривалим процесом, який вимагає знань та досвіду у галузі програмування, математики, теорії кіберзахисту інформації та інтернет протоколів.

Висновки за розділом 2

Під час виконання цього розділу були виконані наступні завдання:

– проведений огляд і аналіз сучасного стану задачі «аналіз ефективності криптографічних алгоритмів RSA, AES, ECC»;

– проведений аналіз застосування криптографічних алгоритмів в інформаційних системах;

– побудувано математичну модель алгоритмів RSA, AES, ECC.

Побудовані математичні моделі криптографічних алгоритмів будуть використовуватися у подальшій програмній реалізації.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Мова програмування Python

Python — це високорівнева мова програмування загального призначення, відома своєю простотою у використанні та читанні. Вона була створена Гвідо ван Россумом і вперше випущена у 1991 році. Ґвідо працював у Центрі математики й інформатики в Нідерландах і розробив Python як спадкоємця мови ABC, усуваючи її недоліки та додаючи нові можливості. Назва "Python" походить від комедійного шоу "Monty Python’s Flying Circus". Python став дуже популярним завдяки своїй синтаксичній простоті, потужній функціональності та широкому застосуванню в різних галузях. Мова була розроблена таким чином, щоб код був легко читабельним. Завдяки цьому програмісти можуть швидко навчатися і писати якісний код. Наприклад, відсутність фігурних дужок і використання відступів замість них роблять код структурованим і зрозумілим. Також Python є інтерпретованою мовою, що означає, що код виконується рядок за рядком без необхідності компіляції в машинний код перед виконанням. Це дозволяє швидко тестувати та розробляти програми, але також робить його дещо повільнішим у порівнянні з компільованими мовами, як C або C++. Присутня підтримка динамічної типізації, що означає, що розробникам не потрібно явно вказувати тип змінної під час її оголошення. Змінні можуть змінювати типи під час виконання, що спрощує розробку, але може викликати помилки в деяких випадках. Python використовує збирач сміття для управління пам’яттю, автоматично очищаючи непотрібні об’єкти. Це зменшує кількість помилок, пов’язаних з управлінням пам’яттю, з якими часто стикаються розробники на інших мовах а також має вбудовану підтримку модулів і пакетів, що дозволяє організовувати код у зручні блоки. Існує величезна кількість бібліотек і модулів, які допомагають вирішувати специфічні завдання, від обробки даних до машинного навчання.

3.2 Python для реалізації криптографічних алгоритмів

Мова програмування Python є однією з найбільш популярних і потужних мов для розробки програмного забезпечення, і її застосування в області криптографії не є виключенням. Вона забезпечує зручний і інтуїтивно зрозумілий синтаксис, що робить її ідеальним вибором для реалізації криптографічних алгоритмів, таких як AES (Advanced Encryption Standard), RSA (Rivest-Shamir-Adleman) та ECC (Elliptic Curve Cryptography). Ці алгоритми використовуються для забезпечення безпеки інформації в різноманітних системах — від захисту даних до цифрових підписів. Python має низку переваг, що роблять її ідеальною платформою для їх порівняння та реалізації. Однією з головних причин, чому Python є гарним вибором для криптографічних задач, є його простота і зрозумілість. Завдяки чистому синтаксису програмісти можуть зосередитись на логіці алгоритмів, а не на складностях мови. Це особливо важливо при розробці криптографічних алгоритмів, де точність і зрозумілість коду мають критичне значення для забезпечення коректної роботи системи. Мова дозволяє реалізувати складні криптографічні операції в кілька рядків коду, що значно скорочує час розробки. Також Python має кілька потужних бібліотек, які забезпечують широкі можливості для реалізації криптографічних алгоритмів. Наприклад, бібліотека PyCryptodome дозволяє легко працювати з симетричними алгоритмами шифрування, такими як AES. Для асиметричних алгоритмів, таких як RSA, є бібліотека cryptography, що підтримує шифрування, підписання і верифікацію за допомогою RSA. Для реалізації ECC є бібліотека ecdsa, яка дозволяє використовувати еліптичні криві для шифрування та цифрових підписів. Ці бібліотеки не лише спрощують реалізацію, але й оптимізують код, дозволяючи фокусуватися на основних завданнях, замість того щоб вручну впроваджувати складні математичні операції. Мова є стандартом у галузі обробки даних, штучного інтелекту та науки про дані. Завдяки цьому Python активно використовується для тестування криптографічних алгоритмів в академічних і наукових дослідженнях. Багато дослідників у галузі криптографії використовують його для тесту-

вання і порівняння різних алгоритмів шифрування, оскільки він дозволяє швидко експериментувати з різними параметрами алгоритмів і перевіряти їх ефективність. Динамічна типізація дозволяє розробникам швидко розробляти прототипи алгоритмів без необхідності зберігати чітке визначення типів даних. Це корисно для криптографії, де часто використовуються складні структури даних, такі як великі числа або масиви бітів. Крім того, Python підтримує кросплатформеність, що дозволяє легко переносити криптографічні реалізації між різними операційними системами без значних змін у коді. Мова активно підтримує сучасні криптографічні стандарти, що є важливим для реалізації алгоритмів шифрування, таких як AES, RSA та ECC і підтримує як старі, так і нові стандарти, зокрема алгоритм AES у режимах ECB, CBC, GCM та інші. Також Python забезпечує підтримку сучасних підходів до реалізації ECC, що дозволяє забезпечити високу безпеку з мінімальними витратами на обчислення.

3.3 Програмна реалізація

Програма реалізації криптографічних алгоритмів AES, RSA та ECC створена в середовищі Google Collab за допомогою мови програмування Python. Метою програми є порівняння цих алгоритмів за ефективністю та можливістю їх застосування для захисту інформації. Програма здійснює шифрування і розшифрування повідомлень за допомогою трьох різних криптографічних підходів, дозволяючи порівняти їх продуктивність і безпеку.

Програма використовує популярні бібліотеки Python для криптографії:

а) PyCryptodome для реалізації алгоритму AES (симетричне шифрування);

б) cryptography для реалізації RSA (асиметричне шифрування);

в) ecdsa для роботи з ECC (криптографія на основі еліптичних кривих).

Опис функціональності програми:

а) Шифрування та розшифрування за допомогою AES: Програма генерує випадковий ключ для AES і ініціалізаційний вектор (IV). Вона виконує шифру-

вання повідомлення в режимі CBC (Cipher Block Chaining) та потім розшифровує його, повертаючи результат. Усі операції шифрування і розшифрування зберігаються у змінних та виводяться на екран разом з часом виконання кожної операції;

б) Шифрування та розшифрування за допомогою RSA: Для алгоритму RSA програма генерує пару ключів (публічний і приватний) та використовує їх для шифрування і розшифрування повідомлення. Публічний ключ використовується для шифрування, а приватний – для розшифрування. Час виконання кожної операції RSA також виводиться для порівняння;

в) Операції на еліптичних кривих за допомогою ECC: Алгоритм ECC реалізує операцію складання точок на еліптичній кривій, що дозволяє ефективно виконувати криптографічні операції з коротшими ключами в порівнянні з RSA. Програма реалізує операцію на кривій NIST P-256, що є стандартом для багатьох криптографічних протоколів.

Основні етапи роботи програми:

а) вхідні дані: програма приймає на вхід просте повідомлення для шифрування. Це повідомлення може бути будь-яким текстом, наприклад, "Hello";

б) шифрування: використовуються три різні алгоритми (AES, RSA, ECC), кожен з яких застосовується до вхідного повідомлення для отримання зашифрованого тексту;

в) розшифрування: після шифрування програма автоматично виконує розшифрування за допомогою відповідного алгоритму та виводить результат;

г) час виконання: для кожного алгоритму вимірюється час, необхідний для шифрування і розшифрування повідомлення. Це дозволяє порівняти ефективність різних криптографічних підходів;

д) виведення результатів: програма виводить на екран зашифровані повідомлення, розшифровані результати, а також час виконання для кожного алгоритму. Це дає змогу порівняти ефективність AES, RSA і ECC.

Принцип роботи програми:

а) AES: включає генерацію випадкового ключа і ініціалізаційного векто-

ру, використання режиму CBC для шифрування даних і розшифрування з відновленням початкового повідомлення;

б) RSA: програма генерує пару ключів за допомогою алгоритму RSA, шифрує та розшифровує повідомлення за допомогою публічного і приватного ключів;

в) ECC: програма реалізує основні операції на еліптичних кривих, зокрема складання точок для симуляції операцій, схожих на підписування та перевірку підписів у протоколах безпеки.

Оцінка результатів:

а) час виконання: порівняння часу виконання кожного з алгоритмів дозволяє визначити, який з них є найбільш ефективним для конкретних задач. Зокрема, AES, як правило, працює швидше за RSA та ECC через свою симетричну природу;

б) безпека: всі три алгоритми мають різні рівні безпеки. RSA забезпечує сильний захист при великих розмірах ключа, але потребує більше часу на обчислення. ECC є більш ефективним з точки зору використання пам’яті і часу при порівнянні з RSA, при тому ж рівні безпеки;

в) використання в реальних системах: AES зазвичай використовується для шифрування даних на рівні файлів і мереж, RSA для цифрових підписів і протоколів аутентифікації, а ECC – для систем з обмеженими ресурсами, таких як мобільні пристрої та IoT.

Висновки за розділом 3

Програма дозволяє провести порівняння трьох основних криптографічних алгоритмів з точки зору їх продуктивності та безпеки. Використання Python дозволяє швидко реалізувати ці алгоритми та порівняти їх ефективність у реальних умовах. Це корисно для розробки систем безпеки, де потрібно вибрати найбільш оптимальний алгоритм шифрування залежно від вимог.

4 РЕЗУЛЬТАТИ ОБЧИСЛЮВАЛЬНОГО ЕКСПЕРИМЕНТУ ТА ЇХ АНАЛІЗ

4.1 Залежність від розміру повідомлення

Практична оцінка часу виконання криптографічних алгоритмів залежно від розміру повідомлення є ключовою для оптимізації безпеки й продуктивності в сучасних інформаційних системах. Вона дозволяє визначити, який алгоритм найкраще підходить для певних сценаріїв: AES ефективно працює з великими даними, забезпечуючи високу швидкість, тоді як RSA та ECC доцільні для малих обсягів даних, де важлива асиметрична безпека. Аналіз цієї залежності допомагає приймати рішення щодо вибору алгоритму в реальних умовах, оцінювати ресурси системи, прогнозувати затримки й розраховувати витрати на масштабування. Таким чином, ця оцінка є основою для створення ефективних, швидкодіючих і безпечних систем, які відповідають вимогам користувачів і обмеженням обладнання.

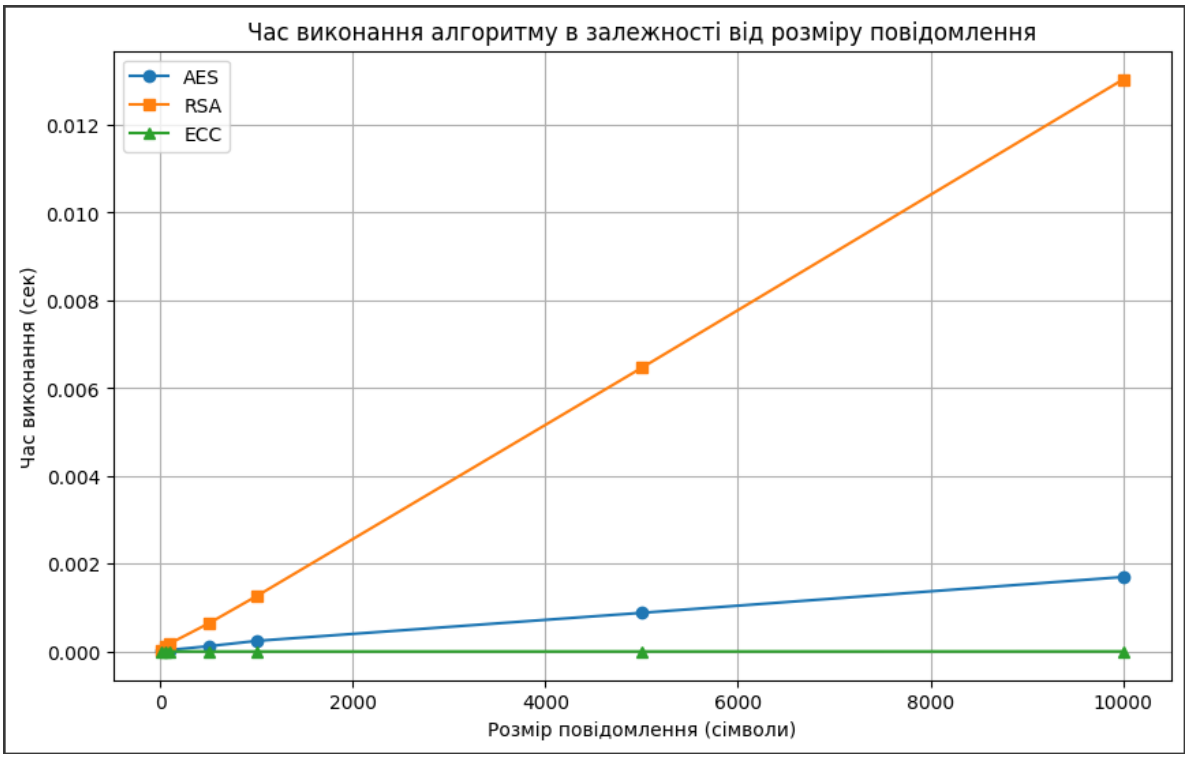

Рисунок 4.1 – Залежність від розміру повідомлення

Алгоритм AES демонструє лінійне збільшення часу виконання зі збільшенням розміру повідомлення. Це очікувано, оскільки AES працює з блоками даних, і час шифрування прямо пропорційний кількості блоків. Алгоритм RSA демонструє значно більший час виконання порівняно з AES, навіть для невеликих повідомлень. Це пов’язано з тим, що RSA оперує з великими числовими обчисленнями, які є дорогими з точки зору продуктивності. Час виконання RSA також зростає з розміром повідомлення. Час виконання ECC залишається практично постійним, незалежно від розміру повідомлення, тому що в цій реалізації виконується симуляція операцій над точками, а не реальне шифрування повідомлень. Це відображає ефективність операцій над еліптичними кривими.

AES добре підходить для великих повідомлень через лінійну масштабованість.

RSA повільніше, оскільки залучає операції з великими числами, але його масштабованість схожа на AES.

ECC виконує базові операції незалежно від розміру повідомлення, що видно на графіку, але його використання для шифрування довгих даних може бути менш ефективним у реальній практиці.

4.2 Залежність від розміру ключа

Практична оцінка часу виконання криптографічних алгоритмів залежно від розміру ключа є важливим аспектом вибору оптимального методу захисту інформації. Розмір ключа безпосередньо впливає на рівень криптостійкості та продуктивність: збільшення ключа підвищує безпеку, але водночас зростають обчислювальні витрати та час виконання. Алгоритми, такі як AES, демонструють майже лінійну залежність часу від розміру ключа, тоді як RSA та ECC мають складніші закономірності через математичну природу їхніх операцій. Такий аналіз дозволяє балансувати між рівнем безпеки та швидкодією, обираючи оптимальний алгоритм і розмір ключа залежно від завдань, обмежень апаратно-

го забезпечення й вимог користувачів.

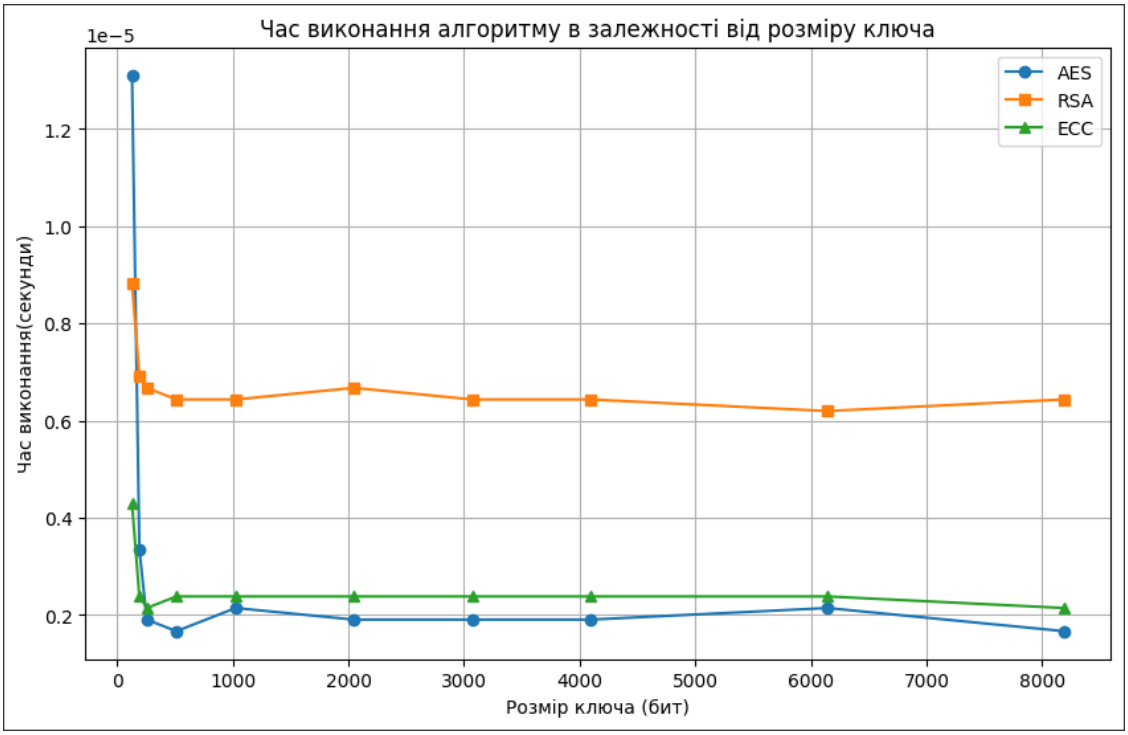

Рисунок 4.2 – Залежність від розміру ключа

Графік ілюструє, як час виконання алгоритмів AES, RSA і ECC залежить від розміру ключа. Алгоритм AES демонструє найменшу залежність від розміру ключа, оскільки час виконання залишається стабільним. Це пов'язано з тим, що AES працює з блоками фіксованого розміру (128 біт) і розмір ключа впливає тільки на кількість раундів шифрування, але не на загальну структуру операцій. Алгоритм RSA демонструє сильнішу залежність від розміру ключа. Час виконання шифрування/дешифрування зростає з ростом розміру ключа, оскільки збільшується складність операцій з великими числами. Для ECC час виконання також збільшується з розміром ключа, хоча зростання є більш поступовим у порівнянні з RSA.

Алгоритм AES підходить для застосунків, де потрібна швидка обробка великих обсягів даних, незалежно від розміру ключа.

RSA потребує значних обчислювальних ресурсів при великих ключах, тому частіше використовується для передачі малих обсягів даних, таких як си-

метричні ключі.

ECC, забезпечуючи схожу безпеку на менших ключах, ніж RSA, дозволяє досягати кращого балансу між швидкістю і безпекою.

4.3 Споживання пам’яті

Практична оцінка споживання пам’яті криптографічними алгоритмами є важливим аспектом їх вибору для різних систем, особливо для ресурсозалежних середовищ, таких як вбудовані системи чи мобільні пристрої. Аналіз показує, що алгоритми симетричного шифрування, такі як AES, зазвичай мають менше споживання пам’яті порівняно з асиметричними алгоритмами, такими як RSA та ECC, через меншу складність операцій. Однак ECC, завдяки ефективній математичній основі, часто демонструє оптимальне співвідношення між рівнем безпеки та використанням ресурсів у порівнянні з RSA. Оцінка споживання пам’яті дозволяє вибирати алгоритми, які відповідають апаратним обмеженням і забезпечують необхідний рівень продуктивності та безпеки.

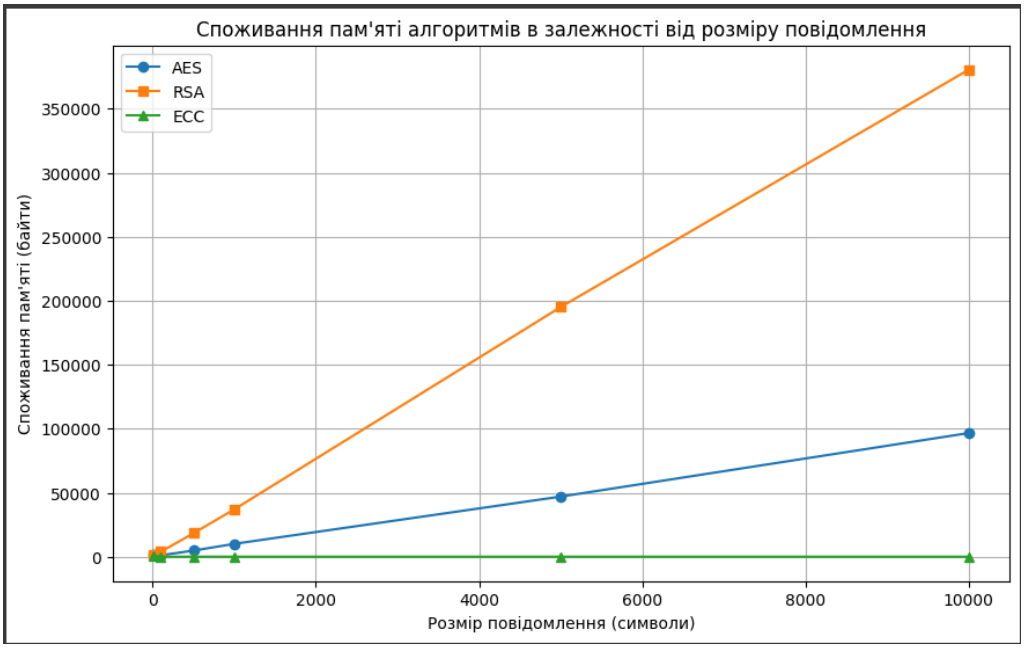

Рисунок 4.3 – Споживання пам’яті

AES має стабільно низьке споживання пам’яті, яке незначно зростає зі збільшенням розміру повідомлення. RSA демонструє значно більше споживання пам’яті порівняно з AES, причому залежність від розміру повідомлення більш виражена. ECC: має стабільне, але помірно високе споживання пам’яті, яке майже не залежить від розміру повідомлення, оскільки криптографічні операції виконуються над фіксованими координатами. Для алгоритмів симетричного шифрування (AES) споживання пам’яті більш оптимальне завдяки їхній структурі. Алгоритми асиметричного шифрування (RSA та ECC) мають більший вплив розміру повідомлення через обчислення великих чисел та виконання математичних операцій (експоненціювання та роботи з еліптичними кривими).

Далі більш детально будемо досліджувати алгоритм ECC.

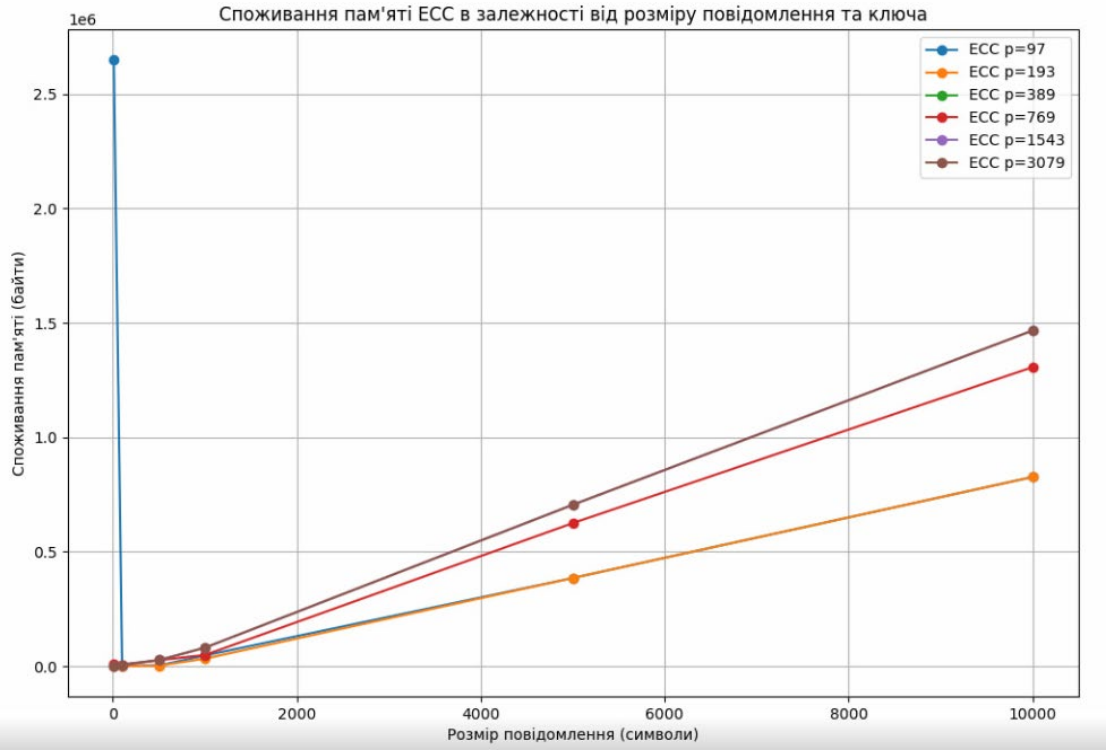

Рисунок 4.4 – Споживання пам’яті

Розмір ключа (модуль p) впливає на обчислення координат точок, що збільшує використання пам’яті. При великих значеннях модуля p обчислення стають більш ресурсоємними. Для деяких значень модуля p могли виникати

випадки, коли інверсія числа модулю не існує (помилка ValueError). Такі випадки враховуються, і для них споживання пам’яті позначене як NaN.

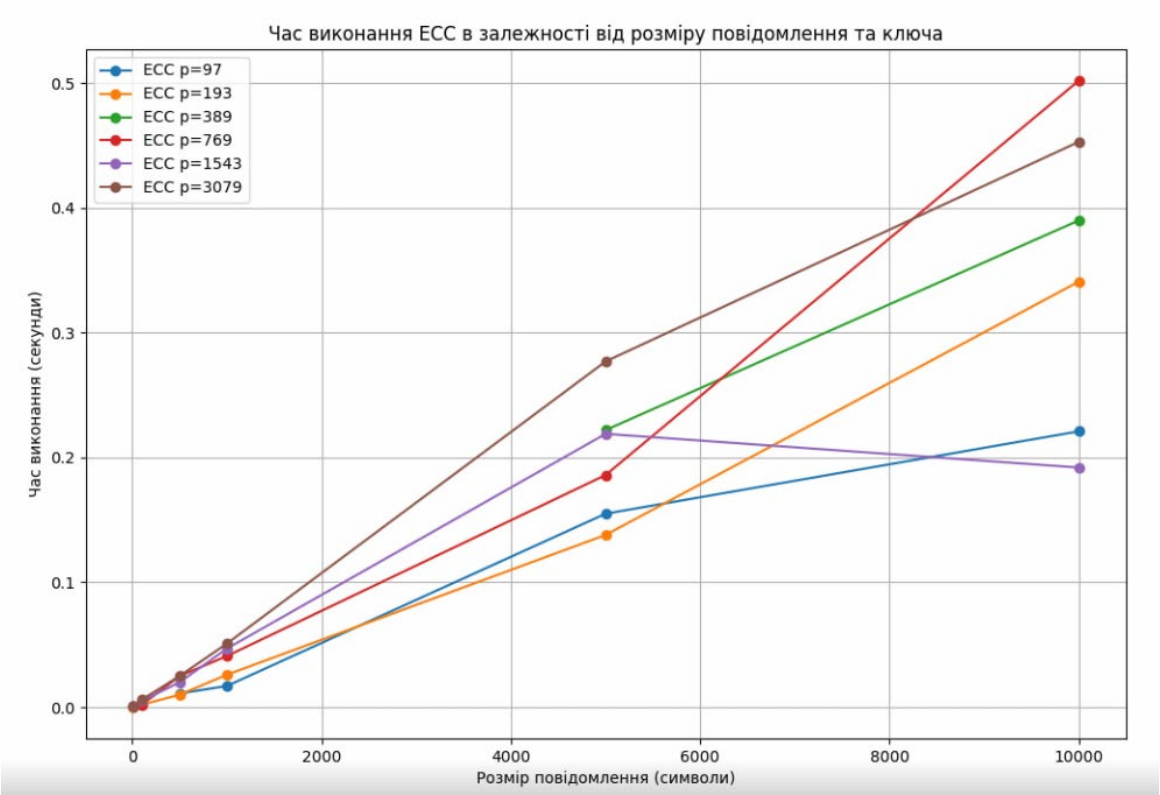

Рисунок 4.5 – Час виконання

Збільшення розміру повідомлення має лінійну залежність часу виконання, оскільки кожен символ шифрується окремо. Залежність часу від розміру ключа є нелінійною. Вона пов’язана з більш тривалим виконанням операцій модульної арифметики, які включають множення точок та обчислення обернених значень на еліптичній кривій.

Висновки за розділом 4

Отже, з отриманих результатів можна стверджувати, що різні криптографічні алгоритми мають найбільшу ефективність кожен на своєму місці, залежно від вимог до безпеки та продуктивності системи.

Алгоритм RSA, завдяки своїй простоті та широкому використанню, є надійним вибором для більшості сучасних протоколів, таких як HTTPS. Однак з огляду на його обмежену ефективність при великих обсягах даних, його використання доцільно обмежити для невеликих за обсягом операцій, таких як підписання та аутентифікація.

Алгоритм AES, зі своєю високою швидкістю і надійністю, є оптимальним для шифрування великих обсягів даних у реальному часі. AES забезпечує відмінний баланс між безпекою та продуктивністю, тому він є стандартом для симетричного шифрування в більшості сучасних протоколів і систем.

ECC (еліптичні криві) продовжують набирати популярність завдяки своїй ефективності в порівнянні з RSA при порівняно малих розмірах ключів, що забезпечує таку ж надійність при меншій витраті ресурсів. Це робить ECC перспективним для використання в мобільних пристроях, IoT, а також у додатках з обмеженими ресурсами.

У підсумку, вибір криптографічного алгоритму повинен залежати від специфіки завдання: для великих обсягів даних рекомендується використовувати AES, для забезпечення цілісності та аутентифікації – RSA, а для ресурсозберігаючих систем – ECC.

ВИСНОВКИ

У даній роботі були побудовані математичні моделі криптографічних алгоритмів RSA, AES та ECC но основі яких була виконана програмна реалізація. Також були побудовані графіки результатів порівняння алгоритмів за такими параметрами як час виконання програми в залежності від розміру повідомлення, час виконання програми в залежності від розміру ключа та споживання пам’яті в залежності від розміру повідомлення.

Практична значимість цієї роботи полягає в тому, що вона допомагає краще зрозуміти ефективність і доцільність використання різних криптографічних алгоритмів у реальних умовах. Зокрема, ця робота може бути корисною для вибору оптимальних криптографічних методів, оптимізації ресурсів у системах з обмеженими ресурсами, підвищення рівня безпеки в різних секторах та аналіз й оцінка криптографічної ефективності в реальних умовах. Ця робота надає можливість порівняти алгоритми за різними параметрами, що дозволяє фахівцям приймати обґрунтовані рішення при впровадженні криптографічних методів у різних інформаційних системах.

Подальший розвиток цієї роботи може включати кілька напрямків, що дозволяють поглибити дослідження та розширити застосування криптографічних алгоритмів у різних сферах, а саме дослідження нових криптографічних алгоритмів, оптимізація алгоритмів для специфічних умов, аналіз впливу нових типів атак на криптографічні алгоритми, моделювання та тестування криптографічних систем в умовах реальних загроз та розробка інструментів для автоматизованого аналізу криптографії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Луханін В. С., Удовенко Д. Ю. Аналіз криптографічних алгоритмів та їх ефективності. *13-а Міжнародна науково-технічна конференція «Інформаційні системи та технології ІСТ-2024»* : зб. матеріалів конференції (м. Харків, 26-28 листопада 2024 р.). Частина 2. Молодіжна секція. Харків : ХНУРЕ, 2024. С. 30–31.

2. National Institute of Standards and Technology (NIST). URL: https://www.nist.gov/programs-projects/cryptography (дата звернення: 27.10.2024).

3. Stallings, William. Cryptography and Network Security: Lecture Notes. URL: https://williamstallings.com/cryptography-and-network-security/ (дата звернення: 27.10.2024).

4. Khan Academy. Cryptography. URL: https://www.khanacademy.org/computing/computer-science/cryptography (дата звернення: 27.10.2024).

5. Wikipedia. Cryptography. URL: https://en.wikipedia.org/wiki/Cryptography (дата звернення: 27.10.2024).

6. Coursera. Cryptography Courses. URL: https://www.coursera.org/courses?query=cryptography (дата звернення: 27.10.2024).

7. IEEE Xplore. Cryptography Research Papers. URL: https://ieeexplore.ieee.org/Xplore/home.jsp (дата звернення: 27.10.2024).

8. SpringerLink. Cryptography and Security. URL: https://link.springer.com/subject/cc (дата звернення: 27.10.2024).

9. Crypto101. Crypto101 Book. URL: https://crypto101.io/ (дата звернення: 27.10.2024).

10. OpenSSL. OpenSSL Documentation. URL: https://www.openssl.org/docs/ (дата звернення: 27.10.2024).

11. The Handbook of Applied Cryptography. URL: http://cacr.uwaterloo.ca/hac/ (дата звернення: 27.10.2024).

12. Stallings W. Cryptography and Network Security: Principles and Practice. Boston : Pearson, 2020. С. 100–120.

13. Menezes A. J., van Oorschot P. C., Vanstone S. A. Handbook of Applied Cryptography. Boca Raton : CRC Press, 1996. С. 130–150.

14. Koblitz N., A Course in Number Theory and Cryptography. New York : Springer-Verlag, 1994. С. 75–85.

15. Stinson D. R., Paterson M. B. Cryptography: Theory and Practice. Boca Raton : CRC Press, 2019. С. 115–135.

16. Goldwasser S., Bellare M. Lecture Notes on Cryptography. Cambridge : MIT Press, 2008. С. 200–220.

17. Katz J., Lindell Y. Introduction to Modern Cryptography. Boca Raton : CRC Press, 2020. С. 145–157.

18. Paar C., Pelzl J. Understanding Cryptography: A Textbook for Students and Practitioners. Berlin : Springer, 2010. С. 160–180.