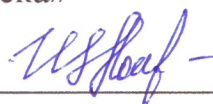


Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

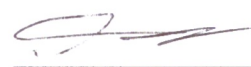
ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи магістра
на тему: «Дослідження та реалізація криптографічних
алгоритмів на еліптичних кривих як складових систем
кібербезпеки»

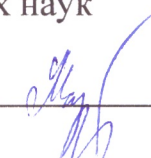
Виконала студентка 2-го курсу групи ЗКБм-22-1
спеціальності 125 «Кібербезпека»

 Н.О. Щур

Керівник доцент кафедри комп'ютерної інженерії та
кібербезпеки, кандидат технічних наук, доцент

 Ю.М. Россінський

Рецензент завідувач кафедри комп'ютерних наук, доктор
філософії (PhD) з комп'ютерних наук

 М.С. Граф

Державний університет «Житомирська політехніка»

Факультет інформаційно-комп'ютерних технологій

Кафедра комп'ютерної інженерії та кібербезпеки

Спеціальність 125 «Кібербезпека»

Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної

інженерії та кібербезпеки



Андрій Єфіменко

13 жовтня 2023 р.

ЗАВДАННЯ

на випускн у кваліфікаційну роботу магістра

Студент: Щур Наталія Олександрівна

Тема роботи: «Дослідження та реалізація криптографічних алгоритмів на еліптичних кривих як складових систем кібербезпеки»

затверджена Наказом університету від 13 жовтня 2023 р. № 573/с

Термін здачі студентом закінченої роботи 22 грудня 2023 р.

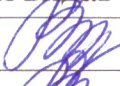
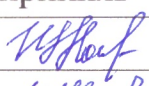
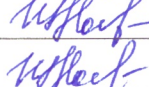
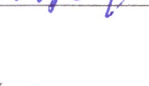
Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

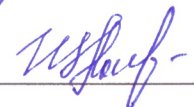
Об'єктом є процес застосування криптографії на еліптичних кривих.

Предметом є дослідження та реалізація криптографічних алгоритмів на еліптичних кривих як складових систем кібербезпеки.

Обладнання та програмне забезпечення: Python 3.11, Python IDLE 3.11.

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	В.В. Воротніков	13.10.23 	13.10.23 
2	В.В. Воротніков	13.10.23 	13.10.23 
3	В.В. Воротніков	13.10.23 	13.10.23 

Студент  Н.О. Щур

Керівник  Ю.М. Россінський

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання та його уточнення	13.10.2023	<i>ушарф</i>
2	Аналіз літературних джерел	18.10.2023	<i>ушарф</i>
3	Вибір середовища розробки	25.10.2023	<i>ушарф</i>
4	Написання вступу	01.11.2023	<i>ушарф</i>
5	Написання розділу 1	15.11.2023	<i>ушарф</i>
6	Написання розділу 2	29.11.2023	<i>ушарф</i>
7	Написання розділу 3	15.12.2023	<i>ушарф</i>
8	Оформлення пояснювальної записки (чистовик)	18.12.2023	<i>ушарф</i>
9	Подання роботи керівнику для допуску до захисту	22.12.2023	<i>ушарф</i>
11	Захист кваліфікаційної роботи	Відповідно до розкладу	<i>ушарф -</i>

Студент *ушарф* Н.О. Щур

Керівник *СЗ* Ю.М. Россінський

АНОТАЦІЯ

Пояснювальна записка до випускної кваліфікаційної роботи на тему «Дослідження та реалізація криптографічних алгоритмів на еліптичних кривих як складових систем кібербезпеки» складається зі вступу, трьох розділів, висновку, списку використаних джерел та додатків. Текстова частина викладена на 58 сторінках друкованого тексту. Загальний обсяг роботи – 73 сторінки. Пояснювальна записка містить 24 рисунки та 4 таблиці.

Кваліфікаційна робота присвячена дослідженню та реалізації криптографічних алгоритмів на еліптичних кривих, які забезпечують високий рівень захисту інформації в сучасних системах кібербезпеки. У першому розділі розглянуто криптографічні аспекти захисту інформаційних ресурсів та математичні основи еліптичних кривих, що використовуються в криптографії. У другому розділі досліджено алгоритми обміну ключами Діффі-Хеллмана (ECDH), цифрового підпису (ECDSA) та інтегрованої схеми шифрування (ECIES) на еліптичних кривих. У третьому розділі описано особливості реалізації криптосистем на еліптичних кривих та наведено деталі розробки програмного коду для алгоритмів ECDH, ECDSA та ECIES.

Результати роботи можуть бути використані для підвищення ефективності та надійності систем кібербезпеки, які вимагають застосування сучасних криптографічних технологій.

КЛЮЧОВІ СЛОВА: КРИПТОГРАФІЯ НА ЕЛІПТИЧНИХ КРИВИХ, АСИМЕТРИЧНІ КРИПТОСИСТЕМИ, АЛГОРИТМ ОБМІНУ КЛЮЧАМИ ДІФФІ-ХЕЛЛМАНА, ЦИФРОВИЙ ПІДПИС НА ЕЛІПТИЧНИХ КРИВИХ, ІНТЕГРОВАНА СХЕМА ШИФРУВАННЯ НА ЕЛІПТИЧНИХ КРИВИХ, ЕСС, ECDH, ECDSA, ECIES

ANNOTATION

The explanatory note for the graduation thesis on the topic «Research and implementation of elliptic curve cryptographic algorithms as components of cybersecurity systems» consists of an introduction, three chapters, conclusions, a list of used sources and appendices. The textual part is presented on 58 pages of printed text. The total volume of the work is 73 pages. The explanatory note includes 24 figures and 4 tables.

The qualification work is dedicated to the exploration and implementation of elliptic curve cryptographic algorithms, which provide a high level of information security in modern cybersecurity systems. The first chapter deals with the cryptographic aspects of information security and the mathematical foundations of elliptic curves used in cryptography. The second chapter investigates Diffie-Hellman key exchange algorithm (ECDH), digital signature (ECDSA), and integrated encryption scheme (ECIES) on elliptic curves. The third chapter describes the peculiarities of elliptic curve cryptographic systems implementing and provides details of the software code development for ECDH, ECDSA, and ECIES algorithms.

The results of the work can be used to improve the efficiency and reliability of cyber security systems that require the use of modern cryptographic technologies.

KEY WORDS: ELLIPTIC CURVE CRYPTOGRAPHY, ASYMMETRIC CRYPTOSYSTEMS, DIFFY-HELLMAN KEY EXCHANGE ALGORITHM, ELLIPTIC CURVE DIGITAL SIGNATURE, ELLIPTIC CURVE INTEGRATED ENCRYPTION SCHEME, ECC, ECDH, ECDSA, ECIES

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	8
ВСТУП	9
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ КРИПТОГРАФІЇ НА ЕЛІПТИЧНИХ КРИВИХ.....	10
1.1. Криптографічні аспекти захисту інформаційних ресурсів	10
1.2. Математичні основи криптосистем на еліптичних кривих.....	14
1.3. Переваги та вразливості ЕСС у контексті кіберзагроз	19
1.4. ЕСС в Україні.....	22
Висновки до Розділу 1	23
РОЗДІЛ 2. КРИПТОГРАФІЧНІ АЛГОРИТМИ НА ЕЛІПТИЧНИХ КРИВИХ ТА ЇХ ЗАСТОСУВАННЯ У СИСТЕМАХ КІБЕРБЕЗПЕКИ	25
2.1. Алгоритм обміну ключами Діффі-Хеллмана на еліптичних кривих	25
2.2. Алгоритм цифрового підпису на еліптичних кривих	29
2.3. Інтегрована схема шифрування на еліптичних кривих	32
2.4. Використання алгоритмів на еліптичних кривих у системах кібербезпеки.....	35
Висновки до Розділу 2	39
РОЗДІЛ 3. РЕАЛІЗАЦІЯ КРИПТОГРАФІЧНИХ АЛГОРИТМІВ НА ЕЛІПТИЧНИХ КРИВИХ	40
3.1. Обґрунтування вибору мови програмування	40
3.2. Особливості реалізації криптосистем на еліптичних кривих	41
3.3. Проектування та імплементація криптографічних алгоритмів на еліптичних кривих	45
3.4. Тестування та аналіз результатів	56
Висновки до Розділу 3	62
ВИСНОВКИ	63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	64

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AES – Advanced Encryption Standard

ECC – Elliptic Curve Cryptography

ECDH – Elliptic Curve Diffie–Hellman

ECDLP – Elliptic Curve Discrete Logarithm Problem

ECDSA – Elliptic Curve Digital Signature Algorithm

ECIES – Elliptic Curve Integrated Encryption Scheme

HMAC – Hash-based Message Authentication Code

KDF – Key Derivation Function,

MAC – Message Authentication Code

NIST – National Institute of Standards and Technology

SHA – Secure Hash Algorithm

ВСТУП

Актуальність теми. Безпека інформаційних систем є одним із пріоритетних напрямків розвитку сучасних технологій. Важливим засобом, що використовується для захисту інформації в комп'ютерних системах, є криптографічні перетворення.

Використання криптографічних систем на еліптичних кривих у кібербезпеці актуальне з ряду причин. По-перше, еліптичні криві надають високий рівень безпеки захищеного обміну даними між різними сторонами, використовуючи менші розміри ключів, ніж традиційні асиметричні криптографічні алгоритми. По-друге, у зв'язку з розвитком квантових обчислювачів, які можуть загрожувати традиційним криптографічним методам, криптографія на еліптичних кривих виступає як перспективна альтернатива для забезпечення стійкості в умовах швидкого технологічного розвитку. По-третє, криптографія на еліптичних кривих має переваги у швидкості обчислень та ефективності використання ресурсів, що особливо важливо для реалізації безпеки в реальному часі та на портативних пристроях.

Метою роботи є дослідження та реалізація криптографічних алгоритмів, що ґрунтуються на еліптичних кривих. Для досягнення мети поставлено такі завдання: аналіз основних принципів та переваг криптографії на еліптичних кривих; дослідження роботи криптосистем обміну ключами Діффі-Хеллмана (ECDH), цифрового підпису (ECDSA) та схеми шифрування (ECIES); вибір оптимальних параметрів та алгоритмів для реалізації криптографічних перетворень; розробка програмного забезпечення для демонстрації роботи розглянутих криптосистем.

Об'єктом є процес застосування криптографії на еліптичних кривих.

Предметом є дослідження та реалізація криптографічних алгоритмів на еліптичних кривих як складових систем кібербезпеки.

Апробація результатів: Щур Н. О., Покотило О. А., Байлюк Є. М. Криптографія на еліптичних кривих та її практичне застосування. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка». 2023. Вип.1 (21), С. 48–64. <https://doi.org/10.28925/2663-4023.2023.21.4864>

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ КРИПТОГРАФІЇ НА ЕЛІПТИЧНИХ КРИВИХ

1.1. Криптографічні аспекти захисту інформаційних ресурсів

В епоху, коли інформаційні технології впроваджуються у всі сфери життя, кібербезпека стає однією з найважливіших викликів сучасності. У цьому контексті, криптографія виступає ключовим компонентом систем безпеки, здійснюючи захист від різноманітних кіберзагроз і забезпечуючи конфіденційність, цілісність та доступність інформаційних ресурсів.

Перш за все, криптографія грає вирішальну роль у забезпеченні *конфіденційності* даних, що полягає у вирішенні проблеми їх захисту від несанкціонованого доступу та ознайомлення. Шифрування є ефективним інструментом для захисту конфіденційних даних, таких як особисті дані, фінансова інформація та корпоративні відомості від кіберзлочинців. Залежно від контексту замість терміну «конфіденційні» дані можуть виступати терміни «секретні», «приватні», «обмеженого доступу».

До того ж, криптографія використовується для забезпечення *цілісності* інформаційних ресурсів, що полягає у гарантуванні неможливості їх несанкціонованої зміни (модифікації). Цифрові підписи дозволяють перевірити автентичність документа, захистити його від підробки та підтвердити особу автора. Якщо дані були змінені, підпис стає недійсним, що дозволяє виявити можливі атаки та втручання в цілісність інформації.

Окрім того, криптографія грає ключову роль у забезпеченні *доступності* даних, що полягає у можливості їх використання за вимогою користувача, який має відповідні повноваження. Системи автентифікації та авторизації часто використовують криптографічні методи для перевірки ідентичності та прав доступу користувачів.

Отже, *криптографія* – це наука про принципи, засоби та методи перетворення даних з метою приховування їх змісту, запобігання несанкціонованого використання або підробки [1, с.56].

Кожне *криптографічне перетворення* однозначно визначається ключем (секретним параметром) та описується криптографічним алгоритмом. *Криптографічний алгоритм* являє собою набір математичних правил та процедур, який описує такі види перетворень, як шифрування, формування та перевірка цифрового підпису, обчислення хеш-значень, спеціальних криптографічних контрольних сум тощо. Сукупність криптографічних алгоритмів, що використовуються для шифрування називають *шифром*.

У свою чергу *шифрування даних* складається із процесів зашифрування та розшифрування. Також, у вузькому значенні, термін «шифрування» використовується як синонім до терміну «зашифрування». *Зашифрування даних* – процес перетворення відкритого тексту до виду, незрозумілого несанкціонованому користувачеві. *Відкритий текст* являє собою вихідне повідомлення, що підлягає зашифруванню. У сучасній комп'ютерній криптографії це повідомлення найчастіше являє собою двійкові дані. Зауважимо, що за визначеннями, поданими у деяких джерелах, під відкритим текстом розуміються дані з доступним семантичним змістом. Однак, це не завжди так, наприклад, при багаторазовому шифруванні, попередньо зашифрований текст може бути відкритим по відношенню до наступного кроку алгоритму шифрування. Результатом зашифрування відкритого тексту є *шифротекст*, що також називають *криптотекстом* або *криптограмою*. При чому, шифротекст за розмірами не завжди може співпадати з відкритим.

Криптографічний ключ (ключ) – це таємний стан деяких параметрів алгоритму криптографічного перетворення, що забезпечує вибір одного перетворення із сукупності можливих для даного алгоритму [2, с.270]. Криптографічні ключі мають певну довжину (в бітах), яка визначає рівень безпеки шифру. Чим більша довжина ключа, тим складніше його зламати за допомогою перебору всіх можливих значень. Однак більша довжина ключа також потребує більше обчислювальних ресурсів для зашифрування і дешифрування. Тому варто шукати оптимальний баланс між безпекою та продуктивністю.

Важливим аспектом використання криптографічних ключів є їх безпечне зберігання та обмін між легітимними сторонами. Втрата або компрометація ключів може призвести до серйозних порушень безпеки. Ефективне керування ключами включає в себе їх генерацію, збереження, обмін та видалення у безпечний спосіб.

Вважається, що зашифроване повідомлення передається від відправника до отримувача по незахищеним каналам зв'язку, в той час, коли ключ має передаватися цілком надійним способом. Знаючи шифр та ключ, отримувач виконує зворотний процес перетворення шифротексту у відкритий текст, що називається *розшифруванням* або *дешифруванням*.

Центральним поняттям у криптографії є поняття криптографічної системи або криптосистеми. Ряд дослідників розглядають криптосистему аналогічно до поняття шифру, як сукупність алгоритмів, що використовуються для зашифрування та розшифрування. Проте, сам по собі опис криптографічних алгоритмів не є криптосистемою. Лише доповнена усіма можливими відкритими текстами, шифротекстами і ключами, система може назватися криптографічною [3, с.18]. Отже, *криптосистема* – це система криптографічного перетворення даних, що містить у собі п'ять компонентів: множину відкритих текстів, множину шифротекстів, множину ключів, сімейство зашифровуючих та розшифровуючих перетворень [2, с.312].

Криптостійкість – це властивість криптосистеми протидіяти атакам супротивника, спрямованим на отримання секретного ключа або відкритого повідомлення. Стійкість криптосистеми визначається її здатністю протидіяти усім можливим атакам. Під *атакою на криптосистему* розуміється спроба порушення безпеки конкретної реалізації криптосистеми. Вдалу криптоатаку називають *зломом*.

За особливостями ключа розрізняють симетричні та асиметричні криптосистеми. У *симетричних* криптосистемах (із закритим ключем) для зашифрування та розшифрування використовується один і той самий ключ. Основними перевагами цих криптосистем є висока швидкість роботи, достатня вивченість, простота реалізації. Значним недоліком є необхідність реалізації

абсолютно захищеного каналу зв'язку для передачі даних, що породжує проблему обміну та зберігання ключів. Оскільки компрометація ключа несе собою загрозу безпеці всієї криптосистеми. Крім того, при використанні криптосистем із закритим ключем виникає проблема підтвердження істинності даних. Одним із найефективніших способів подолання цих недоліків стало винайдення у 1976 р. американськими математиками Вітфілдом Діффі, Мартіном Хеллманом, а також Ральфом Мерклом, *асиметричних криптосистем* (із відкритим ключем), у яких використовуються два ключі – відкритий (публічний) і закритий (секретний), які математично пов'язані один з одним. Повідомлення зашифровується за допомогою відкритого ключа, що доступний усім бажаючим, а розшифровується за допомогою закритого ключа, відомого тільки одержувачу. Криптостійкість таких систем ґрунтується на односторонніх функціях, що легко обчислюються в прямому напрямку й утворюють математичну проблему надзвичайної обчислювальної складності при способі розв'язку оберненої задачі.

Як зазначалося вище, для захисту від підробки, перевірки цілісності даних та достовірності джерела повідомлення використовують *цифровий підпис* – вид електронного підпису, отриманого за результатом криптографічного перетворення набору електронних даних, який додається до цього набору або логічно з ним поєднується і дає змогу підтвердити його цілісність та ідентифікувати підписувача.

Існує декілька алгоритмів побудови цифрового підпису (ЦП). Найбільш ефективним та найпоширенішим у застосуванні на даний момент є алгоритм ЦП на основі асиметричних криптосистем з використанням хеш-функцій. Хеш-функція H , яка використовується у алгоритмі ЦП, призначена для того, щоб стиснути повідомлення M довільної довжини до двійкового хеш-значення $h(M)$ фіксованої довжини. Хешування не входить до складу алгоритму ЦП, тому в схемі може бути використана будь-яка надійна хеш-функція.

Таким чином, криптографія не лише визначає безпеку систем, але й грає ключову роль у створенні довіри до цифрового середовища. Використання сучасних криптографічних методів і практик є необхідністю для ефективного боротьби з кіберзагрозами та забезпечення сталої кібербезпеки у сучасному світі.

1.2. Математичні основи криптосистем на еліптичних кривих

Криптографія на еліптичних кривих (Elliptic Curve Cryptography, ECC) – це сучасне сімейство криптосистем із відкритим ключем, що базується на алгебраїчних структурах еліптичних кривих над скінченними полями та на складності розв’язання задачі дискретного логарифмування в групі точок еліптичної кривої (Elliptic Curve Discrete Logarithm Problem, ECDLP). ECC використовує властивості еліптичних кривих для створення криптографічних протоколів та алгоритмів, які можуть гарантувати високий рівень безпеки за відносно невеликої довжини ключа. ECC реалізує всі основні функції асиметричних криптосистем такі як шифрування, підписи та обмін ключами.

Значний внесок у дослідження криптосистем на еліптичних кривих зробили такі зарубіжні вчені як N. Koblitz [4], V. Miller [5], A. Menezes [6], S. Vanstone [7], D. Bernstein [8], T. Lange [9], L. Washington [10], H. Edwards [11] та інші.

Різні методи криптографічного захисту інформаційних ресурсів, засновані на особливостях перетворень в групі точок еліптичних кривих висвітлювали такі вітчизняні науковці як Юдін О.К., Вадясов К.А. [12], Чевардін В.С., Пономарьов О.А. [13].

Дослідженню властивостей еліптичних кривих у формі Едвардса присвячено праці Бессалова А.В. [14], Циганкової О.В. [15], Беспалова О.Ю., Кучинської Н.В. [16], Ковальчук Л.В. [17], Скуратовського Р.В. [18].

Перспективи розвитку цифрових підписів на основі властивостей груп точок еліптичної кривої розглянуто в працях Ільєнко А.В., Ільєнко С.С., Мазур Я.С., Прокопенко О.В. [19], Нікуліщева Г.І. [20]. Сучасні напрями застосування еліптичних кривих в криптографії розглянуто у роботах Мелешко О.О., Ковальського О.О. [21], R. Harkanson, Y. Kim [22].

Використання еліптичних кривих у криптографії було незалежно запропоновано Нілом Кобліцом та Віктором Міллером у 1985 році. Ніл Кобліц припустив, що ECC забезпечуватиме кращий захист порівняно із звичайними асиметричними алгоритмами, оскільки задача дискретного логарифмування в групі точок еліптичної кривої є складнішою для розв’язання ніж стандартна задача

дискретного логарифмування [4]. Віктор Міллер дослідив математичні властивості еліптичних кривих та дійшов висновку, що еліптичні криві можна застосувати до протоколу обміну ключами Діффі-Хеллмана [5].

У спрощеному вигляді еліптична крива описується рівнянням (форма Вейєрштрасса):

$$y^2 = x^3 + ax + b \quad (1.1)$$

Залежно від значень параметрів a і b еліптичні криві можуть приймати на площині різні форми. Так як $y = \pm\sqrt{x^3 + ax + b}$, то графік кривої симетричний відносно Ox .

Дискримінант рівняння: $D = \left(\frac{a}{3}\right)^3 + \left(\frac{b}{2}\right)^2$:

- $D < 0$ – три різних дійсних корені (Рис.1.1, графік 1);
- $D = 0$ – три дійсних корені, два з яких однакові (Рис.1.1, графік 2 – сингулярна крива, такі криві виключають з розгляду);
- $D > 0$ – один дійсний корінь та два комплексних (Рис.1.1, графік 3).

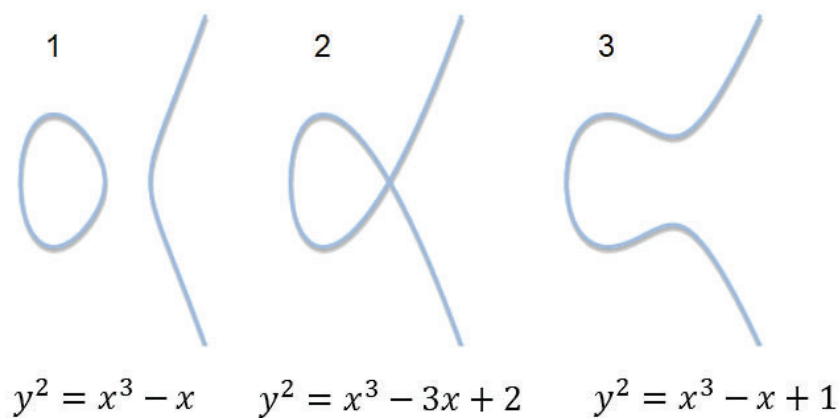


Рисунок 1.1 – Варіанти еліптичних кривих при $D < 0$, $D = 0$ та $D > 0$

У багатьох реальних криптосистемах використовуються еліптичні криві над простим скінченним полем $F(p)$, що описуються рівнянням:

$$y^2 \equiv x^3 + ax + b \pmod{p}, \quad (1.2)$$

де (x, y) – точки еліптичної кривої, a, b – параметри еліптичної кривої, p – просте число ($p \neq 2$, $p \neq 3$). При цьому параметри кривої a та b мають задовольняти умову: $4a^3 + 27b^2 \neq 0 \pmod{p}$.

Точка належить еліптичній кривій, якщо пара чисел (x, y) задовольняє рівнянню (1.2). Множину точок еліптичної кривої позначають через $E_p(a, b)$. Також у множину точок еліптичної кривої включається нескінченно віддалена точка O , яка грає роль нульового елементу. Кількість усіх точок кривої називається *порядком кривої*.

Множина $E_p(a, b)$ разом із введеною точкою на нескінченності O утворює адитивну абелеву групу щодо операції додавання точок (виконуються такі властивості як замкнутість, комутативність, асоціативність, існування оберненого та нейтрального елементів), що являється важливим чинником для застосування в криптографії.

Деякі криві утворюють одну циклічну групу (зберігаючи всі свої точки ЕС), тоді як інші утворюють декілька циклічних підгруп, що не перетинаються (кожна містить підмножину точок еліптичної кривої). У другому сценарії точки на кривій розбиваються на h циклічних підгруп, кожна з яких має порядок r (кожна підгрупа містить однакову кількість точок). Порядок усієї групи $n = h * r$ (кількість підгруп, помножена на кількість точок у кожній підгрупі). Кількість підгруп h , що містять точки еліптичної кривої, називається *кофактором*.

Нехай $P(x, y)$ – точка еліптичної кривої. *Оберненою точкою* до $P(x, y)$ називають точку еліптичної кривої, що симетрична відносно осі Ox та позначають її $-P(x, -y)$ (Рис.1.2). Варто зауважити, що $-P$ має належати $E_p(a, b)$.

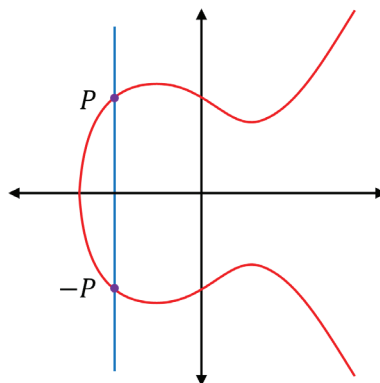


Рисунок 1.2 – Обернена точка еліптичної кривої

Додавання точок. Якщо через дві різні точки $P(x_1, y_1)$ та $Q(x_2, y_2)$, які належать $E_p(a, b)$ провести пряму, то вона обов'язково перетне криву в третій

точці R . Сумою двох точок P та Q буде точка $-R = P + Q$, обернена до третьої точки перетину еліптичної кривої і прямої, що проходить через задані точки (Рис.1.3).

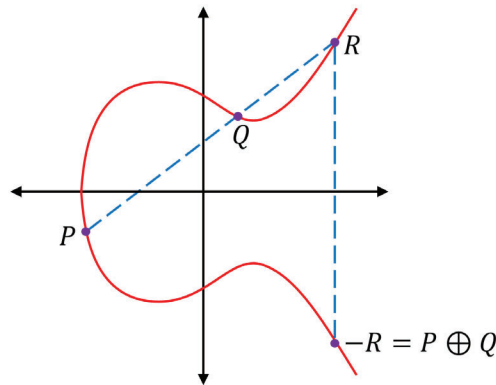


Рисунок 1.3 – Додавання точок еліптичної кривої

Координати $-R(x_3, y_3)$ визначаються за формулами:

$$x_3 = \lambda^2 - x_1 - x_2 \pmod{p}, y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p}, \quad (1.3)$$

де $\lambda = \frac{y_2 - y_1}{x_2 - x_1} \pmod{p}$ – кутовий коефіцієнт січної, що проведена через точки $P(x_1, y_1)$ та $Q(x_2, y_2)$.

Подвоєння точки. Якщо дві точки $P(x_1, y_1)$ та $Q(x_2, y_2)$ співпадають, то $P + Q = P + P$, що рівнозначно подвоєнню точки $2P = -R$. При $P = Q$ січна перетворюється на дотичну, тому точка $2P$ є оберненою до точки R (Рис.1.4).

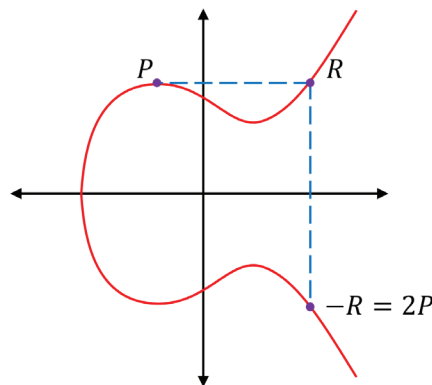


Рисунок 1.4 – Подвоєння точки еліптичної кривої

Координати $-R(x_3, y_3)$ визначаються за формулами:

$$x_3 = \lambda^2 - 2x_1 \pmod{p}, y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p}, \quad (1.4)$$

де $\lambda = \frac{3x_1^2 + a}{2y_1} \pmod{p}$ – кутовий коефіцієнт дотичної, що проведена через точку $P(x_1, y_1)$.

Скалярне множення точки на число. Із попередніх операцій додавання точок та подвоєння точки впливає операція скалярного множення точки на число:

$$2P = P + P$$

$$3P = P + P + P$$

...

$$mP = P + P + P + \dots + P.$$

Скалярне множення є аналогом піднесення до степеню в звичайних асиметричних алгоритмах. Прямою задачею є обчислення $mP = Q$. Зворотна задача полягає у тому, що знаючи точки P та Q , знайти m важко. Дану задачу називають задачею дискретного логарифмування у групі точок еліптичної кривої або ж ECDLP. Для ретельно підібраних параметрів еліптичних кривих та скінченних полів ECDLP не має ефективного рішення.

Точка $G \in E_p(a, b)$ називається базовою точкою підгрупи точок еліптичної кривої $E_p(a, b)$, якщо будь-яка точка P цієї підгрупи може бути подана у вигляді $P = mG$, де $m = 1, 2, \dots, n$, де n – порядок підгрупи. Для базової точки G має місце рівність $nG = O$.

Пара ключів в ЕСС складається із закритого (приватного) і відкритого (публічного) ключів. Приватний ключ використовується для розшифрування або підпису повідомлень, тоді як публічний ключ використовується для шифрування або перевірки підпису. Закритий ключ є випадковим цілим числом, що генерується в діапазоні розміру поля кривої (зазвичай 256-бітне ціле число). Тільки власник закритого ключа повинен знати це число і тримати його в таємниці. Відкритий ключ обчислюється шляхом операції скалярного множення базової точки на значення закритого ключа. Тобто відкритим ключем є точка еліптичної кривої, яка є парою цілих координат (x, y) , що належать кривій.

Окрім форми Вейерштрасса, еліптичні криві можуть бути представлені і в інших формах. Наприклад, у формі Монтгомері рівняння еліптичної кривої має наступний вигляд:

$$by^2 = x^3 + ax^2 + x, \tag{1.5}$$

де $a \neq \pm 2$ і $b \neq 0$.

З міркувань продуктивності ECC також використовує криві у формі Едвардса, які є еліптичними кривими наступного вигляду:

$$x^2 + ay^2 = 1 + dx^2y^2, \quad (1.6)$$

де $d \neq 1$ і $a \neq d$.

Кожна крива Едвардса біраціонально еквівалентна еліптичним кривим у формі Вейерштрасса та у формі Монтгомері, що означає, що можна ефективно користуватися усіма цими формами без втрати інформації про криву. У цій роботі ми будемо використовувати еліптичні криві в класичній формі Вейерштрасса.

1.3. Переваги та вразливості ECC у контексті кіберзагроз

Хоча ECC стала проривом серед криптосистем, вона не знайшла широкого практичного застосування до початку 2000-х років. На сьогоднішній день використання еліптичних кривих для вирішення криптографічних завдань закріплено в численних міжнародних та американських стандартах, зокрема таких як:

- ANSI X9.62 (1999 рік) та ANSI X9.63 (2001 рік) – це стандарти, розроблені Американським Національним Інститутом Стандартів (ANSI) та комітетом X9. Обидва стандарти визначають схеми підпису та обміну ключами на еліптичних кривих і широко застосовуються у фінансовій галузі для захисту фінансових операцій, зокрема для автентифікації платіжних транзакцій [23], [24];
- FIPS 186-3 (2009 рік), FIPS 186-4 (2013 рік) та нова редакція FIPS 186-5 (2023 рік) – стандарти, прийняті Національним інститутом стандартів і технологій США (NIST), що визначають алгоритми цифрового підпису, зокрема ECDSA, а також надають список рекомендованих для використання кривих та їх параметрів [25];
- ISO/IEC 15946-4 (2004 рік) та ISO/IEC 15946-5 (2022 рік) – це міжнародні стандарти, що є складовими частинами серії стандартів ISO/IEC 15946, схвалених Міжнародною організацією по стандартизації (ISO) та Міжнародною електротехнічною комісією (IEC). ISO/IEC 15946-4 встановлює вимоги та

рекомендації для генерації та перевірки електронних підписів на основі еліптичних кривих [24], а ISO/IEC 15946-5 визначає методи щодо генерації еліптичної кривої над скінченним полем [26].

Державні стандарти України, що визначають застосування еліптичних кривих для криптографічного захисту даних, розглянуто у пункті 1.4 цієї роботи.

Важливо зазначити, що нормативні документи та стандарти з криптографії на еліптичних кривих постійно оновлюються. Тому, окрім згаданих стандартів, існує безліч інших документів та рекомендацій, що слугують основою для розробки та впровадження безпечних систем криптографічного захисту інформації на основі еліптичних кривих.

До основних переваг ECC відносять:

1. *Менші розміри ключів.* ECC забезпечує той самий рівень захисту, що й інші асиметричні алгоритми, використовуючи ключі значно меншого розміру (Табл.1.1). Наприклад, 256-бітні ключі ECC забезпечують захист, еквівалентний 3072-бітним ключам RSA [28]. Водночас невеликі розміри ключів значно полегшують процедуру керування ключами, їх зберігання та передачу.

Таблиця 1.1

Порівняння довжини ключів звичайних асиметричних алгоритмів та криптосистем на еліптичних кривих

Ступінь захисту (на кожен біт ключа)	Мінімальна довжина ключа (в бітах)	
	RSA/DSA/DH	ECC
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

2. *Швидкість операцій.* Оскільки використовуються ключі невеликої довжини, то обчислення на еліптичних кривих відбуваються досить швидко. Це забезпечує високу швидкодію обробки даних та підвищує продуктивність систем,

особливо в ситуаціях, де потрібно здійснювати багато криптографічних перетворень. У контексті серверних технологій це допомагає пришвидшити TLS-рукописання, що призводить до надзвичайно швидкого завантаження вебсторінки та підвищення безпеки.

3. *Висока ефективність*. Менша довжина ключа також призводить до того, що пристрої вимагають менше процесорної потужності для виконання криптографічних операцій, що робить ECC ідеальним рішенням для мобільних пристроїв, систем Інтернету речей та інших пристроїв з обмеженою обчислювальною потужністю.

Однак, як і будь-яка криптографічна система, ECC не повністю захищена від атак, а вдосконалення обчислювальних технологій вимагають постійної пильності для гарантії безперервної безпеки ECC. Розглянемо деякі види можливих атак на ECC:

1. Атака грубою силою (brute force attack) є найпростішою атакою на будь-яку криптографічну систему, включаючи ECC. У цій атаці зловмисник намагається знайти приватний ключ, шляхом перебору всіх можливих комбінацій. Хоча ECC вимагає довжини ключа принаймні 128 біт для запобігання атакам грубої сили, прогрес у обчислювальній потужності може зробити цей захист недостатнім у майбутньому. Тому важливо використовувати ключі достатньої довжини та правильно обирати параметри кривих.

2. Атака по бічному каналу (side-channel attack), коли зловмисник намагається здобути інформацію про ключ шляхом моніторингу фізичних параметрів криптосистеми, таких як споживання енергії чи електромагнітне випромінювання, під час виконання криптографічних операцій. Для захисту від таких атак можуть використовуватися методи, які зменшують витрати енергії або випромінювання під час обчислень, такі як шумові імпульси.

3. Квантова атака (quantum-based attack) з використанням квантових обчислювальних систем. Квантові комп'ютери потенційно можуть зламати певні алгоритми ECC, такі як ECDSA та ECDH, ефективно розв'язуючи задачу дискретного логарифмування в групі точок еліптичної кривої за допомогою

алгоритму Шора. Це створює значну загрозу безпеці ЕСС, тому докладаються зусилля для розробки постквантових криптографічних алгоритмів, стійких до квантових атак.

Варто зауважити, що різні криві можуть забезпечувати різний рівень криптографічного захисту, різну продуктивність (швидкодію), а також можуть використовувати різні алгоритми і різну довжину ключа. Вибір конкретної еліптичної кривої залежить від потреб і вимог конкретної системи. Рекомендується використання стандартизованих кривих з метою реалізації сумісності з іншими системами і протоколами. Прикладами таких кривих є NIST P-256, Curve25519, BrainpoolP256r1, secp256k1, Ed25519 [29].

1.4. ЕСС в Україні

В Україні основним документом, що регулює процедури формування та перевірки цифрового підпису є Національний стандарт ДСТУ 4145-2002 «Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка» [30]. Цей стандарт встановлює механізм цифрового підпису, оснований на властивостях груп точок еліптичних кривих над скінченними полями, та правила застосування цього механізму до повідомлень, що пересилаються каналами зв'язку і обробляються у комп'ютеризованих системах загального призначення. Застосування цього стандарту гарантує цілісність підписаного повідомлення, автентичність його автора та неспростовність авторства.

Національний стандарт ДСТУ 4145-2002 регламентує використання еліптичних кривих над скінченним полем $GF(2^m)$. Згідно зі стандартом визначено допустимі основні поля і рекомендовані незвідні многочлени, що їм відповідають. Дозволено використовувати еліптичні криві з параметрами, що наведені в окремому додатку стандарту. Національний стандарт також встановлює алгоритми перевірки правильності вибору рівняння еліптичної кривої та базової точки. ДСТУ 4145-2002 досить гнучкий і щодо вибору параметрів безпеки (наприклад, для нього можна обирати функцію хешування, генератор псевдовипадкових чисел тощо).

На сьогоднішній день низка державних установ, зокрема і банки України ефективно використовують цифровий підпис на основі стандарту ДСТУ 4145-2002 для здійснення банківських операцій у корпоративних та інших телекомунікаційних мережах. Закони України прирівнюють за юридичною силою електронні документи, підписані цифровим підписом, до документів із власноручним підписом, що дозволяє виконувати юридично значущі дії шляхом електронного документообігу.

Також у 2020 році в Україні запроваджено новий Національний стандарт ДСТУ 9041-2020. «Інформаційні технології. Криптографічний захист інформації. Алгоритм шифрування коротких повідомлень, що ґрунтується на скручених еліптичних кривих Едвардса» [31]. Цей алгоритм призначений для шифрування коротких повідомлень довжиною до 616 біт.

Як і стандарт цифрового підпису ДСТУ 4145-2002, новий алгоритм використовує криптографічні перетворення у групі точок еліптичних кривих, використовуючи замість кривих у формі Вейерштрасса криві у формі Едвардса. Це дає суттєві переваги у швидкодії більш ніж у 3 рази. Новий стандарт розроблений з урахуванням усіх найсучасніших вимог до стійкості криптографічних алгоритмів.

Стандарт ДСТУ 9041-2020 узгоджений з усіма діючими в Україні національними стандартами. Його характерними особливостями є відносно невелика довжина ключа, найсучасніший математичний апарат, а також новий алгоритм генерації псевдовипадкових послідовностей, який, на відміну від аналогічного алгоритму генерації з ДСТУ 4145-2002, використовує виключно національні криптографічні алгоритми та не містить посилань на відповідні пострадянські стандарти, термін дії яких вже практично вичерпався.

Висновки до Розділу 1

У Розділі 1 розглянуто криптографічні аспекти захисту інформаційних ресурсів та описано математичний апарат еліптичних кривих, що використовуються для криптографічних цілей. Наводяться основні операції в групі

точок еліптичних кривих, такі як додавання точок, подвоєння точки та скалярне множення точки на число.

Проведено огляд стандартів, що встановлюють рекомендації та вимоги щодо використання еліптичних кривих у криптографічних системах. Проаналізовано переваги криптографії на еліптичних кривих порівняно із традиційними асиметричними алгоритмами, такі як менші розміри ключів, швидкість обчислень та ефективне використання ресурсів. Розглянуто потенційні загрози та вразливості криптографічних алгоритмів на основі еліптичних кривих. Наведено приклади популярних стандартизованих кривих (Curve25519, Curve448, secp256k1), що були перевірені та рекомендовані спеціалізованими організаціями, зокрема такими як NIST. Також досліджується використання криптографії на еліптичних кривих в Україні.

РОЗДІЛ 2. КРИПТОГРАФІЧНІ АЛГОРИТМИ НА ЕЛІПТИЧНИХ КРИВИХ ТА ЇХ ЗАСТОСУВАННЯ У СИСТЕМАХ КІБЕРБЕЗПЕКИ

2.1. Алгоритм обміну ключами Діффі-Хеллмана на еліптичних кривих

Алгоритм обміну ключами Діффі-Хеллмана на еліптичних кривих (Elliptic Curve Diffie-Hellman, ECDH) є протоколом обміну ключами, що дозволяє двом сторонам отримати спільний секретний ключ, використовуючи незахищений від прослуховування, але захищений від модифікації канал зв'язку [32, с. 8]. Припустимо, користувачі A і B мають намір обмінятися ключами за алгоритмом ECDH, суть якого полягає в наступному (Рис. 2.1):

1. Абоненти A і B спільно обирають просте число p та параметри еліптичної кривої a та b .
2. У групі точок еліптичної кривої $E_p(a, b)$ також обирається спільна базова точка $G = (x, y)$, що має дуже великий порядок n .
3. Абонент A обирає $x < n$, обчислює $X_A = xG$ та відправляє його B .
4. Абонент B обирає $y < n$, обчислює $Y_B = yG$ та відправляє його A .
5. Користувач A обчислює закритий ключ за формулою $K_A = xY_B$.
6. Користувач B обчислює закритий ключ за формулою $K_B = yX_A$.

Отриманий спільний секретний ключ $K_A = K_B$ можна використовувати для подальшого шифрування повідомлень із використанням симетричного алгоритму.

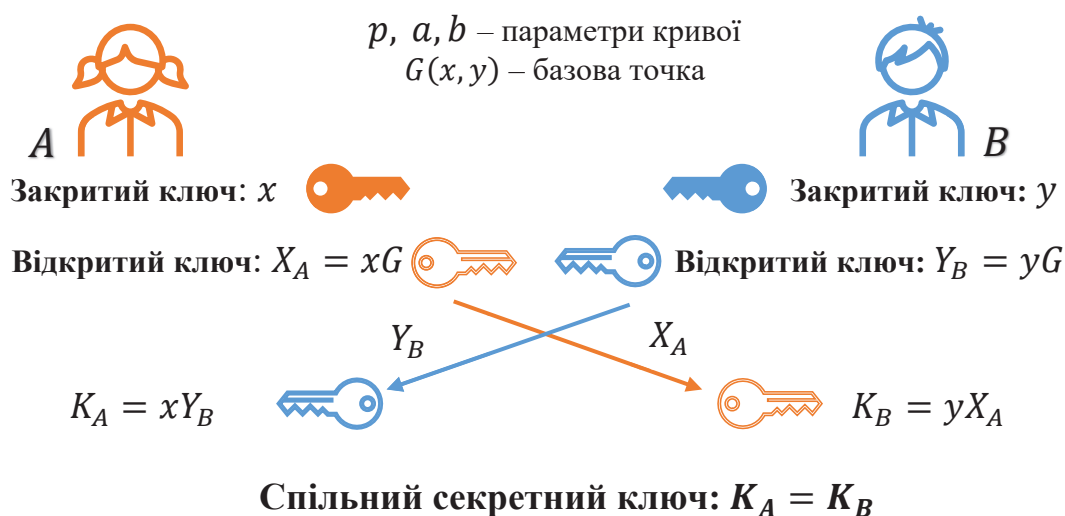


Рисунок 2.1 – Протокол обміну ключами Діффі-Хеллмана

Приклад 2.1. Розглянемо роботу алгоритму ECDH на простому прикладі:

1. Нехай абоненти обрали параметри еліптичної кривої $p = 23$, $a = -2$, $b = 15$, тобто $y^2 \equiv x^3 - 2x + 15 \pmod{23}$.

2. Нехай $G = (4, 5)$ – базова точка.

3. Абонент А обирає $x = 3$ та обчислює $X_A = 3G = 2G + G = (13, 22)$.

Знайдемо спочатку $2G$, виконавши подвоєння базової точки $(4, 5)$.

$$\lambda = \frac{3x_1^2 + a}{2y_1} \pmod{p} = \frac{3 \cdot 16 - 2}{2 \cdot 5} \pmod{23} = \frac{46}{10} \pmod{23} = 0.$$

$$x_3 = \lambda^2 - 2x_1 \pmod{p} = 0 - 8 \pmod{23} = -8 \pmod{23} = 15.$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p} = 0 - 5 \pmod{23} = -5 \pmod{23} = 18.$$

Отже $2G = (15, 18)$.

Далі знайдемо суму $G + 2G = (4, 5) + (15, 18)$ за формулами для додавання.

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \pmod{p} = \frac{18 - 5}{15 - 4} \pmod{23} = \frac{13}{11} \pmod{23}.$$

Знайдемо обернений елемент, розв'язавши рівняння:

$$11 \cdot Z \equiv 1 \pmod{23} \rightarrow Z = 21.$$

$$\lambda = 13 \cdot 21 \pmod{23} = 20.$$

Знаходимо:

$$x_3 = \lambda^2 - x_1 - x_2 \pmod{p} = 400 - 4 - 15 \pmod{23} = 381 \pmod{23} = 13$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p} = 20(4 - 13) - 5 \pmod{23} = -185 \pmod{23} = 22.$$

Отже $3G = X_A = (13, 22)$.

4. Абонент В обирає $y = 7$ обчислимо $Y_B = 7G = 2G + 4G + G = (17, 8)$.

Щоб знайти $4G$, потрібно подвоїти $2G = (15, 18)$ за наступними формулами:

$$\lambda = \frac{3x_1^2 + a}{2y_1} \pmod{p} = \frac{3 \cdot 225 - 2}{2 \cdot 18} \pmod{23} = \frac{673}{36} \pmod{23} = \frac{6}{13} \pmod{23}.$$

Знайдемо обернений елемент, розв'язавши рівняння:

$$13 \cdot Z \equiv 1 \pmod{23} \rightarrow Z = 16.$$

$$\lambda = 6 \cdot 16 \pmod{23} = 96 \pmod{23} = 4.$$

$$x_3 = \lambda^2 - 2x_1 \pmod{p} = 16 - 30 \pmod{23} = -14 \pmod{23} = 9.$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p} = 4(15 - 9) - 18 \pmod{23} = 6 \pmod{23} = 6.$$

Отже $4G = (9, 6)$.

Знайдемо суму точок $3G + 4G = (13, 22) + (9, 6)$.

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} (\text{mod } p) = \frac{6 - 22}{9 - 13} (\text{mod } 23) = \frac{16}{4} (\text{mod } 23) = 4.$$

$$x_3 = \lambda^2 - x_1 - x_2 (\text{mod } p) = 16 - 13 - 9 (\text{mod } 23) = -6 (\text{mod } 23) = 17$$

$$y_3 = \lambda(x_1 - x_3) - y_1 (\text{mod } p) = 4(13 - 17) - 22 (\text{mod } 23) = -38 (\text{mod } 23) = 8.$$

Отже $7G = Y_B = (17, 8)$.

5. Абонент А обчислює закритий ключ $K_A = 3Y_B = 2Y_B + Y_B = (15, 5)$.

Виконаємо подвоєння $Y_B = (17, 8)$, щоб знайти $2Y_B$.

$$\lambda = \frac{3x_1^2 + a}{2y_1} (\text{mod } p) = \frac{3 \cdot 289 - 2}{2 \cdot 8} (\text{mod } 23) = \frac{865}{16} (\text{mod } 23) = \frac{14}{16} (\text{mod } 23) = \frac{7}{8} (\text{mod } 23).$$

Знайдемо обернений елемент, розв'язавши рівняння:

$$8 \cdot Z \equiv 1 (\text{mod } 23) \rightarrow Z = 3.$$

$$\lambda = 7 \cdot 3 (\text{mod } 23) = 21.$$

$$x_3 = \lambda^2 - 2x_1 (\text{mod } p) = 441 - 34 (\text{mod } 23) = 407 (\text{mod } 23) = 16.$$

$$y_3 = \lambda(x_1 - x_3) - y_1 (\text{mod } p) = 21(17 - 16) - 8 (\text{mod } 23) = 13 (\text{mod } 23) = 13.$$

Отже $2Y_B = (16, 13)$.

Далі знайдемо суму точок $Y_B + 2Y_B = (17, 8) + (16, 13)$.

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} (\text{mod } p) = \frac{13 - 8}{16 - 17} (\text{mod } 23) = -\frac{5}{1} (\text{mod } 23) = 18.$$

Знаходимо:

$$x_3 = \lambda^2 - x_1 - x_2 (\text{mod } p) = 324 - 17 - 16 (\text{mod } 23) = 291 (\text{mod } 23) = 15$$

$$y_3 = \lambda(x_1 - x_3) - y_1 (\text{mod } p) = 18(17 - 15) - 8 (\text{mod } 23) = 28 (\text{mod } 23) = 5.$$

Отже $K_A = 3Y_B = (15, 5)$.

6. Абонент В обчислює закритий ключ $K_B = 7X_A = 2X_A + 4X_A + X_A = (15, 5)$.

Виконаємо подвоєння $X_A = (13, 22)$, щоб знайти $2X_A$.

$$\lambda = \frac{3x_1^2 + a}{2y_1} (\text{mod } p) = \frac{3 \cdot 169 - 2}{2 \cdot 22} (\text{mod } 23) = \frac{505}{44} (\text{mod } 23) = \frac{22}{21} (\text{mod } 23).$$

Знайдемо обернений елемент, розв'язавши рівняння:

$$21 \cdot Z \equiv 1 (\text{mod } 23) \rightarrow Z = 11$$

$$\lambda = 22 \cdot 11 \pmod{23} = 12.$$

$$x_3 = \lambda^2 - 2x_1 \pmod{p} = 144 - 26 \pmod{23} = 118 \pmod{23} = 3.$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p} = 12(13 - 3) - 22 \pmod{23} = 98 \pmod{23} = 6.$$

Отже $2X_A = (3, 6)$.

Ще раз виконаємо подвоєння $2X_A = (3, 6)$, щоб знайти $4X_A$.

$$\lambda = \frac{3x_1^2 + a}{2y_1} \pmod{p} = \frac{3 \cdot 9 - 2}{2 \cdot 6} \pmod{23} = \frac{25}{12} \pmod{23} = \frac{2}{12} \pmod{23}.$$

Знайдемо обернений елемент, розв'язавши рівняння:

$$12 \cdot Z \equiv 1 \pmod{23} \rightarrow Z = 2$$

$$\lambda = 2 \cdot 2 \pmod{23} = 4.$$

$$x_3 = \lambda^2 - 2x_1 \pmod{p} = 16 - 6 \pmod{23} = 10 \pmod{23} = 10.$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p} = 4(3 - 10) - 6 \pmod{23} = -34 \pmod{23} = 12.$$

Отже $4X_A = (10, 12)$.

Далі знайдемо суму точок $2X_A + 4X_A = (3, 6) + (10, 12)$.

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \pmod{p} = \frac{12 - 6}{10 - 3} \pmod{23} = \frac{6}{7} \pmod{23}.$$

Знайдемо обернений елемент, розв'язавши рівняння:

$$7 \cdot Z \equiv 1 \pmod{23} \rightarrow Z = 10$$

$$\lambda = 6 \cdot 10 \pmod{23} = 14.$$

Знаходимо:

$$x_3 = \lambda^2 - x_1 - x_2 \pmod{p} = 196 - 3 - 10 \pmod{23} = 183 \pmod{23} = 22$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p} = 14(3 - 22) - 6 \pmod{23} = -272 \pmod{23} = 4.$$

Отже $6X_A = (22, 4)$.

Виконаємо додавання точок $6X_A + X_A = (22, 4) + (13, 22)$.

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \pmod{p} = \frac{22 - 4}{13 - 22} \pmod{23} = \frac{18}{-9} \pmod{23} = -2 \pmod{23} = 21.$$

Знаходимо:

$$x_3 = \lambda^2 - x_1 - x_2 \pmod{p} = 441 - 22 - 13 \pmod{23} = 406 \pmod{23} = 15$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p} = 21(22 - 15) - 4 \pmod{23} = 143 \pmod{23} = 5.$$

Отже $7X_A = (15, 5)$.

Таким чином, секретний ключ, обчислений обома сторонами – (15, 5).

2.2. Алгоритм цифрового підпису на еліптичних кривих

Алгоритм цифрового підпису на еліптичних кривих (Elliptic Curve Digital Signature Algorithm, ECDSA) – це варіант алгоритму цифрового підпису (DSA), який використовує еліптичні криві для генерації підписів, які можуть бути перевірені за допомогою публічного ключа [32, с. 4]. ECDSA широко використовується в різних протоколах для забезпечення автентифікації та цілісності даних.

Як відомо, цифровим підписом повідомлення є блок даних невеликого розміру, одержаний в результаті криптографічного перетворення повідомлення з використанням особистого (закритого) ключа відправника. Обов'язковою складовою цифрового підпису є хеш-функція H , яка призначена для того, щоб стиснути повідомлення M довільної довжини до двійкового хеш-значення $h(M)$ фіксованої довжини.

Подібно до ECDH, обидві сторони спочатку мають узгодити параметри еліптичної кривої a та b , просте число p , базову точку $G = (x, y)$ та n (просте число), таке що $nG = O$. Якщо розмірність n в бітах менше розмірності в бітах хеш-значення повідомлення $h(M)$, то використовуються тільки ліві біти хеш-значення – z .

Обирається закритий ключ d – випадкове ціле число, таке що $0 < d \leq n - 1$. Обчислюється відкритий ключ $Q = dG$. Для формування підпису використовується закритий ключ, а для перевірки – відкритий ключ.

Підписування та перевірка підпису повідомлення складається з наступних кроків (рис. 2.2):

1. Вибирається випадкове ціле число k – разовий секретний ключ, де $0 < k \leq n - 1$.
2. Обчислюється $(x_1, y_1) = kG$
3. Обчислюється $r = x_1 \bmod n$. Якщо $r = 0$, то повертаємося до кроку 1.
4. Обчислюється $s = k^{-1}(z + dr) \bmod n$. Якщо $s = 0$, то повертаємося до п. 1.

5. Підписом для повідомлення M є пара (r, s) .
6. Для перевірки підпису одержувач, отримавши пару (r, s) та підтверджене значення відкритого ключа Q , виконує наступні дії:
7. Обчислюється $w = s^{-1} \bmod n$.
8. Обчислюється $u_1 = z \cdot w \bmod n$ та $u_2 = r \cdot w \bmod n$.
9. Обчислюється $(x_1, y_1) = u_1 G + u_2 Q$.
10. Якщо $(x_1, y_1) = O$ – підпис недійсний.
11. Якщо $r \equiv x_1 \bmod n$ – підпис дійсний.

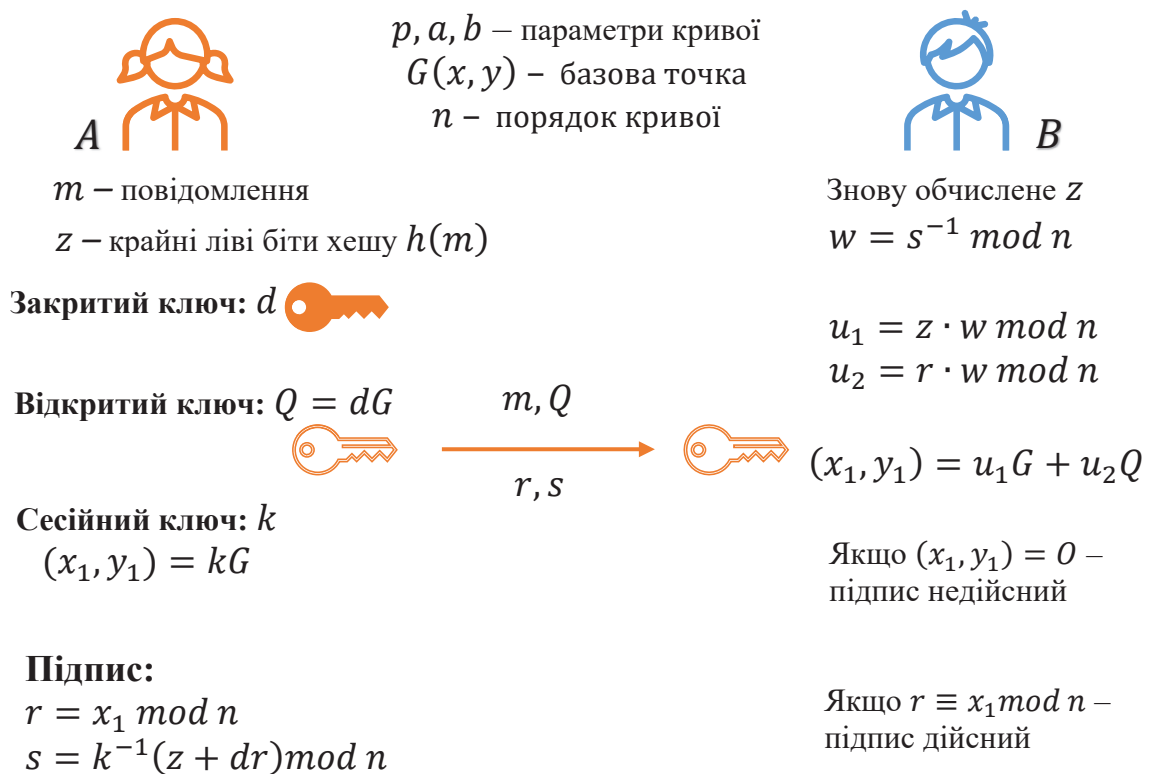


Рисунок 2.2 – Алгоритм цифрового підпису

Приклад 2.2. Наприклад, необхідно підписати та перевірити підпис повідомлення M хеш-значення, якого $z = 10$.

Нехай $p = 23, a = -2, b = 15$, тобто $y^2 \equiv x^3 - 2x + 15 \pmod{23}$. Виберемо базову точку $G = (4, 5); n = 23$.

Оберемо $d = 3$ – закритий ключ.

Обчислимо $Q = dG = 3G = 2G + G = (13, 22)$ – відкритий ключ. Процес обчислення $3G$ детально наведено у прикладі 2.1.

Сесійний ключ: $k = 19$. Обчислимо $kG = 19G = (9, 17)$.

Знаючи з прикладу 2.1, що $4G = (9, 6)$, виконаємо подвоєння цієї точки, щоб знайти $8G$.

$$\lambda = \frac{3x_1^2 + a}{2y_1} \pmod{p} = \frac{3 \cdot 81 - 2}{2 \cdot 6} \pmod{23} = \frac{241}{12} \pmod{23} = \frac{11}{12} \pmod{23}.$$

Знайдемо обернений елемент, розв'язавши рівняння:

$$12 \cdot Z \equiv 1 \pmod{23} \rightarrow Z = 2$$

$$\lambda = 11 \cdot 2 \pmod{23} = 22.$$

$$x_3 = \lambda^2 - 2x_1 \pmod{p} = 484 - 18 \pmod{23} = 466 \pmod{23} = 6.$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p} = 22(9 - 6) - 6 \pmod{23} = 60 \pmod{23} = 14.$$

Отже $8G = (6, 14)$.

Далі знайдемо суму точок $8G + G = (6, 14) + (4, 5)$.

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \pmod{p} = \frac{5 - 14}{4 - 6} \pmod{23} = \frac{9}{2} \pmod{23}.$$

Знайдемо обернений елемент, розв'язавши рівняння:

$$2 \cdot Z \equiv 1 \pmod{23} \rightarrow Z = 12$$

$$\lambda = 9 \cdot 12 \pmod{23} = 16.$$

Знаходимо:

$$x_3 = \lambda^2 - x_1 - x_2 \pmod{p} = 256 - 6 - 4 \pmod{23} = 246 \pmod{23} = 16$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p} = 16(6 - 16) - 14 \pmod{23} = -174 \pmod{23} = 10.$$

Отже $9G = (16, 10)$.

Знову виконаємо подвоєння $9G = (16, 10)$, щоб знайти $18G$.

$$\lambda = \frac{3x_1^2 + a}{2y_1} \pmod{p} = \frac{3 \cdot 256 - 2}{2 \cdot 10} \pmod{23} = \frac{766}{20} \pmod{23} = \frac{7}{20} \pmod{23}.$$

Знайдемо обернений елемент, розв'язавши рівняння:

$$20 \cdot Z \equiv 1 \pmod{23} \rightarrow Z = 15$$

$$\lambda = 7 \cdot 15 \pmod{23} = 13.$$

$$x_3 = \lambda^2 - 2x_1 \pmod{p} = 169 - 32 \pmod{23} = 137 \pmod{23} = 22.$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p} = 13(16 - 22) - 10 \pmod{23} = -88 \pmod{23} = 4.$$

Отже $18G = (22, 4)$.

Далі знайдемо суму точок $18G + G = (22, 4) + (4, 5)$.

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \pmod{p} = \frac{5 - 4}{4 - 22} \pmod{23} = -\frac{1}{18} \pmod{23}.$$

Знайдемо обернений елемент, розв'язавши рівняння:

$$18 \cdot Z \equiv 1 \pmod{23} \rightarrow Z = 9$$

$$\lambda = -1 \cdot 9 \pmod{23} = 14.$$

Знаходимо:

$$x_3 = \lambda^2 - x_1 - x_2 \pmod{p} = 196 - 22 - 4 \pmod{23} = 170 \pmod{23} = 9$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p} = 14(22 - 9) - 4 \pmod{23} = 178 \pmod{23} = 17.$$

Таким чином $19G = (9, 17)$.

Підписом хеш-значення повідомлення буде:

$$r = 9 \pmod{23} = 9;$$

$$s = 19^{-1}(10 + 3 \cdot 9) \pmod{23} = 629 \pmod{23} = 8;$$

$$19^{-1} \pmod{23} = 17.$$

Відомо підпис $(9, 8)$ та $Q = (13, 22)$. Для перевірки підпису проводяться наступні обчислення:

$$w = 8^{-1} \pmod{23} = 3;$$

$$u_1 = 10 \cdot 3 \pmod{23} = 7;$$

$$u_2 = 9 \cdot 3 \pmod{23} = 4;$$

$$7G + 4Q = (17, 8) + (10, 2) = (9, 17).$$

Оскільки $9 \equiv 9 \pmod{23}$ – підпис дійсний.

2.3. Інтегрована схема шифрування на еліптичних кривих

Інтегрована схема шифрування на еліптичних кривих (Elliptic Curve Integrated Encryption Scheme, ECIES) – це гібридна система шифрування, яка поєднує криптографію з відкритим ключем для розподілу ключів і криптографію із закритим ключем для шифрування. ECIES базується на протоколі обміну ключами Діффі-Хеллмана (ECDH) на еліптичних кривих для генерації спільного ключа між двома абонентами. ECIES також використовує функцію формування ключів (Key Derivation Function, KDF) для отримання додаткових ключів для шифрування та автентифікації. Є кілька відомих KDF, таких як HKDF (HMAC-

based Key Derivation Function), PKDF2 (Password-Based Key Derivation Function 2) або `scrypt`. Саме шифрування відбувається за допомогою симетричного алгоритму AES зі 128-бітним ключем в режимі GCM (лічильник Галуа).

Розглянемо детальніше алгоритм роботи ECIES (Рис.2.3).

Обмін ключами:

1. Абоненти A і B спільно обирають просте число p та параметри еліптичної кривої a та b .
2. У групі точок еліптичної кривої $E_p(a, b)$ також обирається спільна базова точка $G = (x, y)$, що має дуже великий порядок n .
3. Абонент A випадковим чином генерує свій закритий (приватний) ключ d і обчислює відкритий (публічний) ключ $Q = dG$ та відправляє його абонентові B .
4. Абонент B генерує випадкове число r , таке що $0 < r \leq n - 1$. Потім обчислює $R = rG$ та надсилає його абонентові A .
5. Абонент B обчислює $S = rQ$ – спільний секретний ключ.
6. Абонент A отримує R і також обчислює спільний ключ $S = dR$, що збігається із ключем абонента B , оскільки $S = drG \rightarrow S = rdG \rightarrow S = rQ$.

Шифрування повідомлення абонентом A :

1. Використовуючи спільний секретний ключ S і метод отримання ключів KDF, абонент A отримує ключ шифрування k_{ENC} , а також ключ k_{MAC} для обчислення коду автентифікації повідомлення (Message Authentication Code, MAC), який також називають тегом автентифікації.
2. За допомогою симетричного алгоритму шифрування абонент A шифрує відкрите повідомлення m ключем k_{ENC} і отримує шифротекст c .
3. Використовуючи k_{MAC} , зашифроване повідомлення c та інші, заздалегідь обговорені сторонами параметри, абонент A обчислює тег повідомлення tag_A , наприклад за допомогою функції HMAC (Hash-based Message Authentication Code).
4. Абонент A відправляє (c, tag_A) абонентові B .

Розшифрування повідомлення абонентом B :

1. Абонент B також генерує ключі k_{ENC} та k_{MAC} за допомогою спільного секретного ключа S та методу отримання ключів KDF.
2. Обчислює тег tag_B зашифрованого повідомлення c із використанням ключа k_{MAC} і порівнює його з отриманим тегом tag_A . Якщо вони не збігаються, абонент B відхиляє повідомлення через невдачу у процедурі перевірки MAC.
3. Якщо теги виявляються однаковими, то абонент B може продовжити процес, розшифровуючи зашифроване повідомлення, використовуючи симетричний алгоритм шифрування та ключ k_{ENC} .

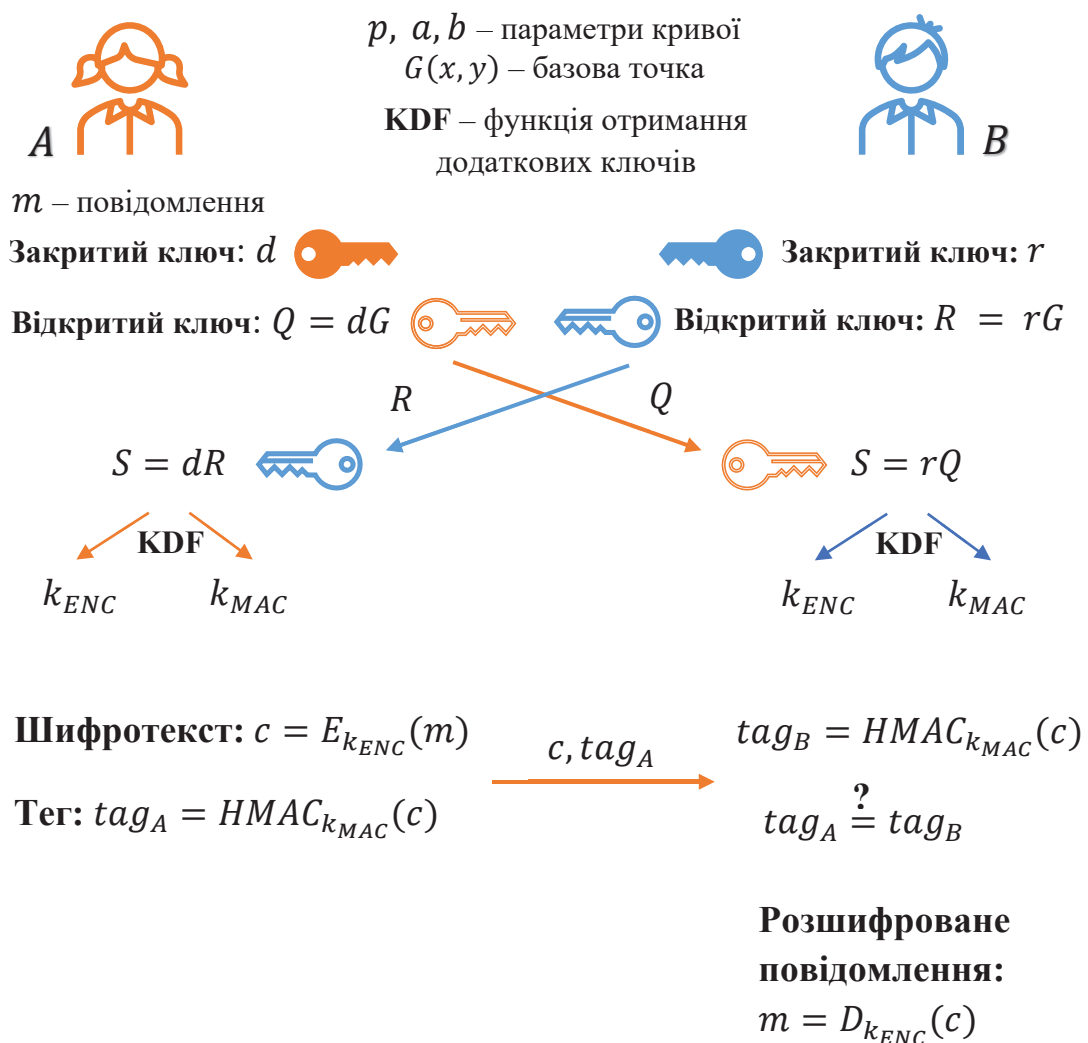


Рисунок 2.3 – Інтегрована схема шифрування на еліптичних кривих

2.4. Використання алгоритмів на еліптичних кривих у системах кібербезпеки

Мережеве з'єднання. Криптосистеми на еліптичних є важливим елементом сучасних реалізацій багатьох протоколів безпеки, зокрема протоколу TLS, що використовується для передачі даних між вузлами комп'ютерної мережі. TLS дозволяє здійснювати автентифікацію сторін, забезпечувати конфіденційність даних, контролювати їх цілісність за допомогою кодів автентифікації повідомлень [33].

На початку встановлення безпечного з'єднання клієнт та сервер узгоджують версію протоколу, наприклад TLS 1.2, TLS 1.3. Після чого сервер обирає набір параметрів шифрування (cipher suites), які підтримуються клієнтом та відповідають усім необхідним вимогам.

Розглянемо приклад такого набору параметрів шифрування як TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, який визначає наступне:

- для обміну ключами використовується ECDHE – алгоритм Діффі-Хеллмана на еліптичних кривих (остання літера «Е» означає «ефемерний»). Ефемерний криптографічний ключ є тимчасовим ключем, створеним спеціально для використання в межах однієї операції обміну ключами. Сам по собі алгоритм Діффі-Хеллмана нестійкий до атаки типу «людина посередині» та зазвичай реалізується разом із засобами автентифікації, що включають у себе використання цифрових сертифікатів;
- для забезпечення автентифікації даних на етапі встановлення з'єднання застосовується алгоритм цифрового підпису ECDSA. Варто зауважити, що ECDSA була введена до специфікації TLS 1.2 у 2008 році.
- повідомлення шифруються симетричним алгоритмом AES зі 128-бітним ключем в режимі GCM (лічильник Галуа);
- в якості алгоритму хешування використовується SHA-256.

Зазначимо, що одним з найбільш поширених застосувань TLS є протокол HTTPS – більш просунута і безпечна версія HTTP. Аналогічним чином, електронна

пошта з протоколом SMTPS фактично є SMTP у поєднанні із TLS, а FTPS – протокол для безпечної передачі файлів – це також FTP разом із TLS.

Криві NIST, рекомендовані для використання в TLS, такі: P-256, P-384, P-521. Окрім цих кривих NIST, існує також кілька інших кривих, які вважаються безпечними для використання в TLS, зокрема Curve25519 і Curve448. Ці криві все частіше використовуються в реалізаціях TLS, особливо в його нових версіях. Зокрема крива Curve25519 визначена над простим полем $p = 2^{255} - 19$ і є кривою Монтгомері, що означає, що її можна представити рівнянням:

$$y^2 = x^3 + 486662x^2 + x \pmod{p}, \quad (2.1)$$

Еліптична крива Curve448 визначена над простим полем $p = 2^{448} - 2^{224} - 1$ і задається рівнянням:

$$x^2 + y^2 = 1 - 39081x^2y^2 \pmod{p}, \quad (2.2)$$

Однією з ключових відмінностей між Curve25519 та Curve448 є розміри ключів. Тоді як Curve25519 має розмір ключа 256 біт, Curve448 має розмір ключа 448 біт. Рекомендується надавати перевагу Curve448 перед Curve25519, якщо потрібен вищий рівень безпеки, але варто мати на увазі, що Curve448 приблизно в три рази повільніше за Curve25519.

Криптовалюта. Найвідомішою і найпоширенішою криптовалютою у світі є біткоїн – криптографічно безпечна децентралізована система однорангових (Peer To Peer, P2P) електронних платежів, яка дозволяє здійснювати транзакції з використанням віртуальної валюти. Біткоїн заснований на технології блокчейн, що являє собою розподілену базу даних для зберігання інформації про усі транзакції учасників системи у вигляді ланцюжка блоків. При цьому кожен блок містить свій власний унікальний ідентифікатор, що пов'язує його з попереднім блоком ланцюжка.

Верифікацію транзакцій між учасниками мережі дозволяє здійснювати алгоритм цифрового підпису ECDSA. Для роботи цього алгоритму використовується варіант еліптичної кривої відомий як secp256k1, який належить до сімейства стандартів, пропонованих для використання в криптографії (Standards for Efficient Cryptography, SEC).

У випадку біткоїнів еліптична крива secp256k1 задається рівнянням:

$$y^2 = x^3 + 7 \pmod{p}, \quad (2.3)$$

де p – велике просте число $2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$.

Стандартом secp256k1 також визначена базова точка $G = (x, y)$, що належить заданій кривій (9) та має дуже великий порядок n (близький до 2^{256}). Основне використання secp256k1 полягає у генерації ключів, підписуванні та перевірці підпису. Кроки формування та перевірки цифрового підпису за алгоритмом ECDSA описано вище, а даними для підписування будуть транзакції, які передають право власності на біткоїни.

Обмін повідомленнями. ECC часто використовується в системах наскрізного шифрування (End-to-End Encryption, E2EE) голосових викликів, відеодзвінків і миттєвих повідомлень. Наскрізне шифрування означає, що повідомлення шифруються на пристрої відправника та можуть бути розшифровані лише на пристрої одержувача. Технологія E2EE гарантує, що ніхто, навіть сервер, який з'єднує користувачів, не зможе отримати доступ до повідомлень.

Одним із популярних прикладів E2EE, що використовує апарат еліптичних кривих, є криптографічний протокол Signal, розроблений Open Whisper Systems у 2013 році і вперше представлений в додатку TextSecure з відкритим вихідним кодом, який пізніше було об'єднано в додаток Signal. Кілька додатків з закритим вихідним кодом, наприклад, такі як WhatsApp, стверджують, що реалізували цей протокол, який, за їхніми словами, шифрує розмови більше мільярда людей у всьому світі. Розробники Facebook Messenger також стверджують, що вони пропонують цей протокол для додаткових секретних сеансів зв'язку.

Протокол Signal об'єднує розширений протокол узгодження ключів Діффі-Хелмана на еліптичних кривих (X3DH) та алгоритм Double Ratchet для шифрування. X3DH встановлює початковий спільний секретний ключ між двома сторонами, які взаємно автентифікують одна одну на основі відкритих ключів. До або після угоди про ключ X3DH сторони можуть порівнювати свої ідентифікаційні відкриті ключі через певний автентифікований канал. Наприклад, вони можуть порівнювати відбитки відкритих ключів вручну або скануючи QR-код.

Надсилання та отримання зашифрованих повідомлень відбувається за алгоритмом Double Ratchet. Для кожного надісланого або отриманого повідомлення Double Ratchet генерує новий набір ключів. Відкритий текст шифрується за допомогою симетричного алгоритму AES-256 у режимі CBC. Автентифікація та хешування у протоколі Signal відбувається за допомогою HMAC-SHA256. Для асиметричних криптографічних перетворень рекомендовано використовувати еліптичні криві Curve25519 або Curve448.

Інтернет речей. Інтернет речей (Internet of Things, IoT) являє собою мережу підключених до Інтернету пристроїв, що аналізують, обробляють та передають дані. За допомогою цієї технології існує можливість керувати кількома пристроями віддалено. Ця галузь стрімко розвивається, проте на даний момент головною проблемою IoT є вразливість до кібератак. Зі збільшенням кількості підключених «розумних» пристроїв, ростуть ризики несанкціонованого доступу до IoT-системи.

Для того, щоб пристрої, що працюють відповідно до технології IoT могли повноцінно функціонувати і безпечно взаємодіяти між собою, між ними необхідно встановити захищене з'єднання. З цією метою, як правило, використовують «легкі» криптографічні алгоритми, оскільки вони вимагають менше обчислювальних ресурсів і пам'яті, ніж звичайні алгоритми. Так звана «легковагова» криптографія на еліптичних кривих являє собою спеціалізовану реалізацію ECC, розроблену для пристроїв IoT з обмеженими ресурсами. За рахунок використання ключа меншого розміру або вибору спеціальних параметрів кривої, алгоритми, які використовуються для обміну ключами (ECDH) та цифрових підписів (ECDSA) можна оптимізувати для пристроїв IoT.

Впровадження полегшених криптографічних перетворень часто включає компроміси між безпекою та ефективністю. Наприклад, зменшення розміру ключа або використання спрощених алгоритмів може підвищити ефективність, але може знизити безпеку. Незважаючи на це, на сьогоднішній день питання застосування ECC для IoT залишається актуальним. Низка досліджень присвячена проектуванню і впровадженню криптографії на еліптичних кривих для підтримки безпечної

роботи таких складових IoT як бездротові сенсорні мережі (WSN), радіочастотні ідентифікатори (RFID), «розумні» будинки та міста тощо.

ECC є важливою технологією для захисту пристроїв і комунікацій IoT, оскільки вона може надавати надійну безпеку з мінімальними витратами ресурсами. Однак правильна реалізація та оптимальний вибір параметрів роботи алгоритмів на еліптичних кривих мають вирішальне значення для реалізації довгострокової безпеки системи.

Висновки до Розділу 2

У Розділі 2 детально розглянуті основні кроки алгоритмів обміну ключами Діффі-Хеллмана (ECDH), цифрового підпису (ECDSA) та інтегрованої схеми шифрування (ECIES) на еліптичних кривих. Для наочності представлено ілюстративні схеми описаних алгоритмів та наведено прості приклади, що демонструють процеси безпечного обміну ключами між абонентами, створення та перевірки цифрових підписів на еліптичних кривих. Здійснено огляд основних сфер практичного застосування криптографічних алгоритмів на еліптичних кривих, зокрема таких як захист мережевого з'єднання, криптовалютні операції, обмін повідомленнями, Інтернет речей.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ КРИПТОГРАФІЧНИХ АЛГОРИТМІВ НА ЕЛІПТИЧНИХ КРИВИХ

3.1. Обґрунтування вибору мови програмування

Вибір мови програмування для реалізації криптографічних алгоритмів є важливим завданням, яке залежить від конкретних вимог проекту та особливостей самого алгоритму. Адже криптографічні операції часто вимагають високої швидкодії та ефективності обчислень, особливо для великих обсягів даних. Мови програмування низького рівня, такі як C та C++, зазвичай мають вищу продуктивність, ніж мови високого рівня, такі як Python та Java, оскільки вони дозволяють прямий доступ до пам'яті, регістрів та апаратних засобів шифрування. Вони широко використовуються для розробки криптографічних бібліотек, таких як OpenSSL, Crypto++, NaCl і інші. Однак мови низького рівня мають свої недоліки, такі як складність написання та підтримки коду.

Мови програмування високого рівня, такі як Python та Java, зазвичай мають простий та логічний синтаксис, гнучкість та широкий спектр готових бібліотек для роботи з криптографією. Java має вбудований пакет `java.security`, який надає інтерфейси та класи для реалізації криптографічних функцій, таких як шифрування, дешифрування, хешування, цифровий підпис, генерація ключів та інші. Мова програмування Python завдяки своєму зручному синтаксису та багатому набору криптографічних бібліотек є хорошим інструментом для розробки та імплементації різноманітних криптографічних функцій та алгоритмів. Вона має багато готових бібліотек для роботи з криптографією, таких як `cryptography`, `PyCrypto`, `secrets` та інші. Вона також дозволяє використовувати C код з Python, що збільшує швидкодію та ефективність криптографічних програм. Python підтримує об'єктно-орієнтоване програмування, що дозволяє створювати абстракції та інкапсулювати дані та функціонал. Це полегшує розробку та підтримку програм, а також сприяє повторному використанню коду.

Таким чином, згідно вищевказаного, прийнято рішення використовувати мову програмування Python. Це обумовлено тим, що вона пропонує найзручніші

інструменти для швидкої та якісної розробки криптосистем на еліптичних кривих. Також Python підтримує стандартні алгоритми хешування, такі як MD5, SHA-1, SHA-2 і SHA-3. Python дозволяє генерувати послідовності випадкових значень в якості ключів за допомогою модуля random. Загалом, спрощений синтаксис разом із потужними функціональними можливостями для обробки математичних операцій і великих чисел роблять Python оптимальним вибором для успішної реалізації ECC.

3.2. Особливості реалізації криптосистем на еліптичних кривих

Особливості реалізації криптосистем на еліптичних кривих включають наступні важливі аспекти.

Вибір правильних параметрів еліптичної кривої. Для використання ECC сторони, що спілкуються, повинні узгодити параметри кривої. Рекомендується використовувати іменовані криві із принаймні 256-бітним модулем. Стандартні криві добре вивчені криптографами, щоб гарантувати їх надійність безпеки.

Наприклад, одна із еліптичних кривих, яку будемо використовувати в роботі є *secp256k1*, що належить до сімейства стандартів, пропонує для використання в криптографії (Standards for Efficient Cryptography, SEC).

Еліптична крива *secp256k1* задається рівнянням:

$$y^2 = x^3 + 7 \pmod{p}, \quad (3.1)$$

де p – дуже велике просте число $2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$.

Стандартом *secp256k1* також визначена базова точка $G = (x, y)$, що належить заданій кривій (9) та має дуже великий порядок n (близький до 2^{256}).

У програмній реалізації мовою Python параметри кривої задаються у 16-вій системі числення наступним чином:

- **p** (модуль) =
= 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFC2F
- **n** (порядок; кількість усіх можливих точок еліптичної кривої) =
= 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141

4) використовуємо рекурентне співвідношення $v_{i+1} = v_{i-1} - q_{i+1}v_i$;

5) якщо $r_i = 1$, тоді зупиняємо процес, $v_i = x$ (шукане значення).

Наприклад, при додаванні двох точок $P(12, 19)$ та $Q(5, 4)$ еліптичної кривої заданої рівнянням: $y^2 = x^3 + x + 1 \pmod{23}$ отримуємо:

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \pmod{p} = \frac{4 - 19}{5 - 12} \pmod{23} = \frac{-15}{-7} \pmod{23} = \frac{15}{7} \pmod{23}.$$

Якщо записати $15 \cdot \frac{1}{7} \pmod{23} \rightarrow 15 \cdot 7^{-1} \pmod{23}$, то потрібно знайти обернений елемент, розв'язавши рівняння: $7 \cdot x \equiv 1 \pmod{23} \rightarrow x = 10$ (за розширеним алгоритмом Евкліда):

i	r_i	q_i	u_i	v_i
0	23		1	0
1	7		0	1
2	$23 \bmod 7 = 2$	$23 \div 7 = 3$	$1 - 3 \cdot 0 = 1$	$0 - 3 \cdot 1 = -3$
3	$7 \bmod 2 = 1$	$7 \div 2 = 3$	$0 - 3 \cdot 1 = -3$	$1 - 3 \cdot (-3) = 10$

Далі підставимо 10 замість $\frac{1}{7}$ у вираз $15 \cdot \frac{1}{7} \pmod{23}$ і обчислимо:

$$\lambda = 15 \cdot 10 \pmod{23} = 12.$$

$$x_3 = \lambda^2 - x_1 - x_2 \pmod{p} = 144 - 12 - 5 \pmod{23} = 127 \pmod{23} = 12.$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p} = 12(12 - 12) - 19 \pmod{23} = 4 \pmod{23} = 4.$$

$$\text{Отже } P + Q = (12, 19) + (5, 4) = (12, 4).$$

Операція скалярного множення точки еліптичної кривої на число є доволі ресурсозатратною через те, що потрібно виконати багато операцій додавання та подвоєння над дуже великими числами (256 біт та більше), що вимагає багато часу та ресурсів. Однак існує ряд методів, які дозволяють значно пришвидшити даний процес. Наприклад, якщо ми хочемо обчислити $9P$ для якоїсь точки P кривої, то замість виконання 9 додавань, ми можемо виконати всього лише 4 операції – 3 подвоєння та 1 додавання:

$$P + P = 2P$$

$$2P + 2P = 4P$$

$$4P + 4P = 8P$$

$$8P + P = 9P.$$

Алгоритм, побудований за допомогою такого підходу називається *Double-and-Add* (подвоєння та додавання). Цей метод використовує розклад скаляра у двійковий вигляд (бінарний розклад) для прискорення операції множення.

Основні кроки методу Double-and-Add:

- 1) перевести скаляр m у двійкову форму, щоб отримати його бінарний розрядний запис числа;
- 2) початкове значення результату ініціалізувати як нескінченно віддалену точку;
- 3) для кожного біта скаляра, починаючи від старших бітів до молодших, виконати наступні дії:
 - якщо біт дорівнює 1:
 - подвоїти поточний результат, використовуючи операцію подвоєння;
 - додати початкову точку P до результату, використовуючи операцію додавання;
 - Якщо біт дорівнює 0:
 - просто подвоїти поточний результат;
- 4) після проходження усіх бітів кінцевий результат дорівнюватиме mP .

Наприклад, нехай $m = 131$, у двійковій формі це 1000 0011. Тоді операції будуть виконуватися наступним чином:

Біт	Подвоєння	Додавання
1	-	-
0	$P + P = 2P$	-
0	$2P + 2P = 4P$	-
0	$4P + 4P = 8P$	-
0	$8P + 8P = 16P$	-
0	$16P + 16P = 32P$	-
1	$32P + 32P = 64P$	$64P + P = 65P$
1	$65P + 65P = 130P$	$130P + P = 131P$

Таким чином основна ідея застосування даного методу полягає у тому, що подвоєння точки є обчислювально ефективною операцією, а додавання точки виконується лише у тих випадках, коли відповідний біт скаляра рівний 1.

3.3. Проектування та імплементація криптографічних алгоритмів на еліптичних кривих

Базуючись на результатах попередніх розділів, для реалізації було обрано алгоритми обміну ключами ECDH, підпису ECDSA та шифрування ECIES описані у розділі 2. Такий вибір було зроблено тому, що дані схеми лежать в основі багатьох сучасних технологій захисту даних, зокрема таких як захист мережевого з'єднання, криптовалютні операції, обмін повідомленнями, Інтернет речей тощо.

Стандартні бібліотеки та модулі, які використовуються у проєкті:

- ***collections*** використовується для створення іменованих кортежів, а саме для представлення параметрів еліптичної кривої в зручний спосіб;
- ***hashlib*** дозволяє використовувати різні хеш-функції, наприклад, SHA-256;
- ***random*** дозволяє генерувати випадкові числа для генерації закритих ключів;
- ***binascii*** забезпечує функціонал для конвертації даних між бінарним та текстовим представленням;
- ***libnum*** містить корисні функції для роботи з числами, зокрема для операцій над простими числами.

Розроблено модуль ***ecc***, який містить реалізацію основних операцій над точками еліптичних кривих та складається із нижче описаних функцій.

Функція ***inverse_mod(k, p)*** реалізує пошук мультиплікативного оберненого елемента за модулем p , де k – число, для якого шукаємо обернений елемент (Рис.3.1). У випадку, якщо k дорівнює 0, функція викидає *ZeroDivisionError*. Якщо k від'ємне число, то використовується властивість: $k^{-1} = p - (-k)^{-1} \pmod{p}$.

Далі використовується розширений алгоритм Евкліда для знаходження найбільшого спільного дільника (*gcd*) та коефіцієнтів x та y , таких, що $k * x + p * y = gcd$. Перевіряємо чи *gcd* (найбільший спільний дільник) дорівнює 1, що гарантує існування мультиплікативно оберненого елемента. Переконуємося, що $(k * x) \% p == 1$, тобто знайдений елемент є дійсно оберненим за модулем p . Функція повертає $x \% p$, оскільки результат має бути обмежений модулем p .

```

def inverse_mod(k, p):
    """Повертає мультиплікативно обернений елемент до k за модулем p
    (x * k) % p == 1
    """
    if k == 0:
        raise ZeroDivisionError('Ділення на 0')

    if k < 0:
        # k ** -1 = p - (-k) ** -1 (mod p)
        return p - inverse_mod(-k, p)

    # Розширений алгоритм Евкліда
    v, old_v = 0, 1
    u, old_u = 1, 0
    r, old_r = p, k

    while r != 0:
        quotient = old_r // r
        old_r, r = r, old_r - quotient * r
        old_v, v = v, old_v - quotient * v
        old_u, u = u, old_u - quotient * u

    gcd, x, y = old_r, old_v, old_u

    assert gcd == 1
    assert (k * x) % p == 1

    return x % p

```

Рисунок 3.1 – Функція пошуку мультиплікативного оберненого елемента за модулем

Функція *is_on_curve(point)* перевіряє, чи належить задана точка еліптичній кривій (Рис.3.2). Якщо вхідна точка рівна *None*, точка вважається нескінченно відділеною. В іншому випадку, функція використовує координати точки, щоб перевірити, чи вони задовольняють рівняння еліптичної кривої. Якщо справджується рівність $y^2 - x^3 - ax - b \pmod{p} = 0$, то точка належить еліптичній кривій, і функція повертає *True*. В протилежному випадку, функція повертає *False*, що означає, що точка не належить еліптичній кривій.

```

def is_on_curve(point):
    """Повертає True, якщо точка належить еліптичній кривій"""
    if point is None:
        # None - нескінченно відділена точка
        return True

    x, y = point

    return (y * y - x * x * x - curve.a * x - curve.b) % curve.p == 0

```

Рисунок 3.2 – Функція, яка перевіряє чи належить точка кривій

Функція *point_add(point1, point2)* виконує операцію додавання (подвоєння) точок еліптичної кривої, відповідно групового закону, основна ідея якого полягає у тому, що сума точок, які належать еліптичній кривій, також належить цій кривій (Рис.3.3).

Спочатку перевіряється, чи кожна з точок належить еліптичній кривій за допомогою функції *is_on_curve*. Якщо одна з точок дорівнює *None*, функція повертає іншу точку, оскільки додавання точки до нескінченно відділеної точки буде ця сама точка. Потім обчислюється сума точок (x_1, y_1) та (x_2, y_2) за формулами додавання (1.3) чи подвоєння (1.4). Для обчислення кутового коефіцієнту січної (дотичної) λ використовується функція *inverse_mod(k, p)*, описана вище. В результаті отримується нова точка (x_3, y_3) та перевіряється, чи вона також належить еліптичній кривій.

```
def point_add(point1, point2):
    """Повертає результат додавання point1 + point2"""
    assert is_on_curve(point1)
    assert is_on_curve(point2)

    if point1 is None:
        # 0 + point2 = point2
        return point2
    if point2 is None:
        # point1 + 0 = point1
        return point1

    x1, y1 = point1
    x2, y2 = point2

    if x1 == x2 and y1 != y2:
        # point1 + (-point1) = 0
        return None

    if x1 == x2:
        # Додавання точок
        l = (3 * x1 * x1 + curve.a) * inverse_mod(2 * y1, curve.p)
    else:
        # Подвоєння точок
        l = (y2 - y1) * inverse_mod(x2 - x1, curve.p)

    x3 = l * l - x1 - x2
    y3 = l * (x1 - x3) - y1
    result = (x3 % curve.p, y3 % curve.p)

    assert is_on_curve(result)

    return result
```

Рисунок 3.3 – Функція додавання (подвоєння) точок еліптичної кривої

Функція *scalar_mult(m, point)* реалізує операцію множення точки еліптичної кривої на ціле число m (Рис.3.4). Використовуються алгоритми подвоєння та додавання для оптимального обчислення множення точки на скаляр.

Починаючи із ініціалізації змінних результату та доданка, програма використовує бітові операції та порозрядний зсув, щоб ефективно виконати операції множення. У циклі, для кожного біту в числі m , який визначається за допомогою операції $m \& 1$, виконується або подвоєння, або додавання точок, залежно від значення біта. Таким чином, результат роботи цього алгоритму – точка, яка є результатом множення вхідної точки на скаляр m .

Функція також має перевірки на випадки, коли m кратне порядку групи точок або коли вхідна точка є нескінченно відділеною, а також коли m є від’ємним значенням.

```
def scalar_mult(m, point):
    """Повертає результат m * point"""
    assert is_on_curve(point)

    if m % curve.n == 0 or point is None:
        return None

    if m < 0:
        # m * point = -m * (-point)
        return scalar_mult(-m, point_neg(point))

    result = None
    addend = point

    while m:
        if m & 1:
            # Додавання
            result = point_add(result, addend)

            # Подвоєння
            addend = point_add(addend, addend)

        m >>= 1 #порозрядний зсув вправо

    assert is_on_curve(result)

    return result
```

Рисунок 3.4 – Функція скалярного множення точки еліптичної кривої на число

Функція *make_keypair()* генерує пару ключів для криптосистеми, що використовує еліптичну криву (Рис.3.5). Функція випадковим чином обирає закритий ключ *private_key* з діапазону від 1 до порядку групи точок еліптичної

кривої. Далі, використовуючи функцію множення точки на скаляр *scalar_mult*, обчислюється відкритий ключ *public_key* як результат множення генератора групи точок на вибраний закритий ключ. В результаті функція повертає пару ключів: закритий та відкритий.

```
def make_keypair():
    """Закритий ключ генерується випадково, відкритий обчислюється"""
    private_key = random.randrange(1, curve.n)
    public_key = scalar_mult(private_key, curve.g)

    return private_key, public_key
```

Рисунок 3.5 – Функція генерування пари ключів

Для реалізації роботи алгоритмів можна використовувати будь-яку еліптичну криву. Як приклад, виберемо *secp256k1*, яка є добре дослідженою іменованою кривою, що використовується в різних криптографічних протоколах та системах. Крива *secp256k1* має дуже великий порядок, що робить її стійкою до атак. Для зручного представлення параметрів еліптичної кривої, таких як назва (*name*), модуль (*p*), коефіцієнти *a* та *b*, базова точка (*g*), порядок групи точок (*n*), та кофактор (*h*) використовується іменований кортеж *EllipticCurve* (Рис.3.6). Таким чином, за потреби, можна легко задати будь-яку іншу криву.

```
EllipticCurve = collections.namedtuple('EllipticCurve', 'name p a b g n h')
curve = EllipticCurve(
    'secp256k1',
    # Модуль поля
    p=0xfffffffffffffffffffffffffffffffffffffffffffffffffeffffc2f,
    # Коефіцієнти
    a=0,
    b=7,
    # Базова точка
    g=(0x79be667ef9dcbbac55a06295ce870b07029bfcd2dce28d959f2815b16f81798,
        0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08fffb10d4b8),
    # Порядок підгрупи
    n=0xfffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141,
    # Кофактор підгрупи
    h=1,
)
```

Рисунок 3.6 – Іменований кортеж *EllipticCurve* для представлення параметрів еліптичної кривої

Алгоритм обміну ключами Діффі-Хеллмана на еліптичних кривих. Для наочності виводяться на екран параметри обраної абонентами еліптичної кривої. Для кожного абонента випадковим чином генерується закритий ключ, обчислюється відповідний йому відкритий ключ. Після чого ці ключі також виводяться на екран. Далі відбувається обчислення та виведення спільних секретних ключів сторін, які можуть використовуватися для безпечного обміну даними між цими сторонами (Рис.3.7).

Рисунок 3.7 – Алгоритм обміну ключами Діффі-Хеллмана на еліптичних кривих

Перевірка підпису відбувається за допомогою відкритого ключа підписувача. На екран виводиться результат *res* – значення отримане в результаті перевірки. Якщо *res* дорівнює *r*, то підтверджується дійсність підпису (Рис.3.8).

```
import sys
import random
import hashlib
import libnum

from ecc import curve, scalar_mult, point_add, make_keypair

msg="hello world"

if (len(sys.argv)>1):
    msg=(sys.argv[1])

# Пара ключів абонента А (підписувача)
dA,QA = make_keypair()

# Обчислення хешу
h=int(hashlib.sha256(msg.encode()).hexdigest(),16)

# Сесійний ключ
k = random.randint(0, curve.n-1)
rpoint = scalar_mult(k,curve.g)

# Підписування
r = rpoint[0] % curve.n
inv_k = libnum.invmod(k,curve.n)
s = (inv_k*(h+r*dA)) % curve.n

print ("Повідомлення:", msg)
print ("Закритий ключ підписувача =", hex(dA))
print(f"Відкритий ключ підписувача = (0x{QA[0]:x}, \n\t\t\t\t\t 0x{QA[1]:x}) ")
print ("Сесійний ключ k =",hex(k))
print(f"\nПідпис:\nr={hex(r)}\ns={hex(s)}")

# Перевірка підпису
inv_s = libnum.invmod(s,curve.n)
w = inv_s
u1=(h*w) % curve.n
u2=(r*w) % curve.n
P = point_add(scalar_mult(u1,curve.g), scalar_mult(u2,QA))

print ("\nПеревірка підпису:")
print ("w =", hex(w))
print ("u1 =", hex(u1))
print ("u2 =",hex(u2))
print(f"P =(0x{P[0]:x}, \n\t\t\t\t\t 0x{P[1]:x}) ")

res = P[0] % curve.n
print ("Результат: =", hex(res))

if (res==r):
    print("!!!Підпис дійсний!!!")
else:
    print("!!!Підпис НЕдійсний!!!")
```

Рисунок 3.8 – Алгоритм цифрового підпису на еліптичних кривих

Інтегрована схема шифрування на еліптичних кривих. Для реалізації роботи алгоритму ECIES створено окремий модуль ***aes-gcm***, що містить функції для генерації ключів, шифрування та дешифрування даних за алгоритмом AES-GCM. Для виконання даних операцій використано бібліотеку ***cryptography*** для Python та інші бібліотеки та модулі, що до неї входять і призначені для надання криптографічної функціональності:

- ***PBKDF2HMAC*** використовується для створення ключів (key derivation) на основі паролю або іншого секрету за допомогою алгоритму PBKDF2 та HMAC в якості псевдовипадкової функції;
- ***hashes*** надає реалізації різних хеш-функцій, таких як SHA-256, SHA-3, і багато інших;
- ***hmac*** використовується для створення та перевірки кодів аутентифікації повідомлень;
- ***Cipher, algorithms, modes*** надають можливість шифрування та розшифрування даних за допомогою різних алгоритмів і режимів роботи;
- ***default_backend*** визначає бекенд (реалізацію) для конкретних операцій криптографії та зазвичай використовується для вибору найкращої доступної реалізації залежно від операційної системи та конфігурації.

Основні функції модуля ***aes-gcm*** описано нижче.

Функція ***derive_keys(secret_key, salt)*** забезпечує генерацію ключа шифрування та ключа автентифікації на основі деякого секретного ключа (Рис.3.9). Використовуючи PBKDF2HMAC із заданою кількістю ітерацій, сіллю та хеш-функцією SHA-256, генерується ключовий матеріал. Отриманий ключовий матеріал поділяється на дві частини: перші 32 байти використовуються для шифрування, а наступні 32 байти – для автентифікації.

```
def derive_keys(secret_key, salt):
    # Перетворення великого цілого числа в байтовий рядок
    secret_key_bytes = secret_key.to_bytes((secret_key.bit_length() + 7) // 8, byteorder='big')
    # Генерація ключів за допомогою KDF
    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        iterations=1000,
        salt=salt,
        length=64,
        backend=default_backend()
    )

    key_material = kdf.derive(secret_key_bytes)
    k_ENC = key_material[:32] # 256 біт для шифрування
    k_MAC = key_material[32:] # 256 біт для MAC

    return k_ENC, k_MAC
```

Рисунок 3.9 – Функція генерування ключів шифрування та автентифікації

Функція *encrypt_and_tag(k_ENC, k_MAC, plaintext)* призначена для зашифрування повідомлення та генерації тегу автентифікації з використанням алгоритму AES-GCM (Рис.3.10). Спочатку функція генерує випадковий вектор ініціалізації (*IV*), використання якого дозволяє безпечно шифрувати дані та генерувати тег автентифікації. Потім ініціалізується об'єкт AES-GCM з використанням заданого ключа шифрування, отриманого вектору ініціалізації та бекенду за замовчуванням. Далі, об'єкт *encryptor* використовується для зашифрування відкритого тексту. Після завершення шифрування отримується тег автентифікації, який використовується для перевірки цілісності та автентифікації даних при їхньому розшифруванні. Функція повертає три значення: *IV*, шифротекст та тег автентифікації, які потім можуть бути використані для подальшого розшифрування та перевірки автентичності даних.

```
def encrypt_and_tag(k_ENC, k_MAC, plaintext):
    # Генерація випадкового IV (Initialization Vector)
    iv = os.urandom(16)
    # Ініціалізація об'єкта AES-GCM
    cipher = Cipher(algorithms.AES(k_ENC), modes.GCM(iv), backend=default_backend())
    encryptor = cipher.encryptor()
    # Шифрування даних
    ciphertext = encryptor.update(plaintext) + encryptor.finalize()
    # Отримання тегу автентифікації
    tag = encryptor.tag

    return iv, ciphertext, tag
```

Рисунок 3.10 – Функція шифрування та автентифікації за алгоритмом AES-GCM

Функція *decrypt_and_verify(k_ENC, k_MAC, iv, ciphertext, tag)* призначена для розшифрування та перевірки тегу автентифікації повідомлення (Рис.3.11). Ключі (*k_ENC* та *k_MAC*), *IV* та тег, які отримані разом із зашифрованим текстом,

використовуються для створення об'єкта AES-GCM. Використовуючи цей об'єкт, функція дешифрує зашифрований текст. Під час цього процесу також автоматично проводиться верифікація тегу автентифікації. Якщо тег правильний, повертається розшифрований текст. У протилежному випадку, генерується виняток і функція виводить повідомлення про помилку.

```
def decrypt_and_verify(k_ENC, k_MAC, iv, ciphertext, tag):
    try:
        # Ініціалізація об'єкта AES-GCM
        cipher = Cipher(algorithms.AES(k_ENC), modes.GCM(iv, tag), backend=default_backend())
        decryptor = cipher.decryptor()
        # Розшифрування даних
        plaintext = decryptor.update(ciphertext) + decryptor.finalize()

        return plaintext
    except InvalidTag:
        # Обробка винятку, коли тег автентифікації не вірний
        print("Помилка: Тег автентифікації не вірний або дані були змінені.")
        return None
```

Рисунок 3.11 – Функція дешифрування та верифікації за алгоритмом AES-GCM

При роботі інтегрованої схеми шифрування на еліптичних кривих (Рис.3.12) спочатку обчислюється спільний секретний ключ за допомогою раніше реалізованого алгоритму ECDH. Кожен абонент використовує свій закритий ключ та відкритий ключ іншого абонента для обчислення спільного секретного ключа за допомогою операції множення точки еліптичної кривої на скаляр.

На основі значення x-координати спільного секретного ключа обчислюються ключі k_ENC та k_MAC для кожного абонента. Для посилення безпеки процесу обчислення ключів використовується випадкова сіль.

Абонент А, за допомогою функції шифрування, шифрує повідомлення та отримує тег автентифікації. Використовуючи власні ключі, абонент В, за допомогою функції дешифрування та верифікації виконує розшифрування отриманого шифротексту і перевірку тегу автентифікації. Повідомлення, шифротекст, тег та розшифроване повідомлення виводяться на екран.

```

import random
import os
from ecc import curve, scalar_mult, point_add, make_keypair
from aes_gcm import derive_keys, encrypt_and_tag, decrypt_and_verify

print(f"Крива:\t {curve.name}")
print(f"a:\t {curve.a}")
print(f"b:\t {curve.b}")
print(f"G:\t (0x{curve.g[0]:x}, \n\t 0x{curve.g[1]:x}) ")
print(f"p:\t {curve.p}")

print("=====")

ASecretKey, APublicKey = make_keypair()
BSecretKey, BPublicKey = make_keypair()

print(f"Закритий ключ абонента A: {hex(ASecretKey)}")
print(f"Відкритий ключ абонента A: (0x{APublicKey[0]:x}, \n\t\t\t\t\t 0x{APublicKey[1]:x}) ")
print(f"Закритий ключ абонента B: {hex(BSecretKey)}")
print(f"Відкритий ключ абонента B: (0x{BPublicKey[0]:x}, \n\t\t\t\t\t 0x{BPublicKey[1]:x}) ")

sharedSecret1 = scalar_mult(BSecretKey, APublicKey)
sharedSecret2 = scalar_mult(ASecretKey, BPublicKey)

print("=====")
print(f"Спільний секретний ключ KA: (0x{sharedSecret1[0]:x}, \n\t\t\t\t\t 0x{sharedSecret1[1]:x})")
print(f"Спільний секретний ключ KB: (0x{sharedSecret2[0]:x}, \n\t\t\t\t\t 0x{sharedSecret2[1]:x})")

shared_secret_key=sharedSecret1[0]
print("=====")
print(f"Секретний ключ x-координата: \t", hex(shared_secret_key))

plaintext = b'Elliptic Curve Cryptography'
salt = os.urandom(32)

# Шифрування (Абонент A)
k_ENC_A, k_MAC_A = derive_keys(shared_secret_key, salt)
print("=====")
print(f"k_ENC: 0x{k_ENC_A.hex()}")
print(f"k_MAC: 0x{k_MAC_A.hex()}\n")
iv, ciphertext, tag = encrypt_and_tag(k_ENC_A, k_MAC_A, plaintext)

# Дешифрування (Абонент B)
k_ENC_B, k_MAC_B = derive_keys(shared_secret_key, salt)
decrypted_text = decrypt_and_verify(k_ENC_B, k_MAC_B, iv, ciphertext, tag)

print(f"Повідомлення: {plaintext.decode('utf-8')}")
print(f"IV: 0x{iv.hex()}")
print(f"Шифротекст: 0x{ciphertext.hex()}")
print(f"Тег: 0x{tag.hex()}")
print(f"Розшифроване повідомлення: {decrypted_text.decode()}")

```

Рисунок 3.12 – Інтегрована схема шифрування на еліптичних кривих

Таким чином, основний програмний модуль *ecc* надає ряд функціональних можливостей для виконання основних криптографічних операцій на еліптичних кривих. За його допомогою можна генерувати закриті та обчислювати відкриті ключі із використанням властивостей еліптичних кривих. Додатковий модуль *aes-gcm* містить функції для генерації ключів, шифрування та дешифрування даних за алгоритмом AES-GCM. Повний лістинг програмного коду наведено у додатках.

Крім того, програма надає гнучкість у виборі параметрів кривої, дозволяючи використовувати стандартні криві або визначати власні. При виборі параметрів еліптичної кривої для криптографічних застосувань важливо керуватися визнаними стандартами, рекомендаціями організацій, зокрема такими як NIST.

3.4. Тестування та аналіз результатів

Алгоритм обміну ключами Діффі-Хеллмана на еліптичних кривих. Результат роботи алгоритму обміну ключами Діффі-Хеллмана на еліптичних кривих для кривої з прикладу 2.1 та кривої *secp256k1* наведено нижче (Рис.3.13, Рис.3.14). Як бачимо, обчислений спільний секретний ключ для обох абонентів завжди буде однаковим.

```
Крива: test
a: -2
b: 15
G: (4, 5)
p: 23
=====
Закритий ключ абонента А: 3
Відкритий ключ абонента А: (13, 22)
Закритий ключ абонента В: 7
Відкритий ключ абонента В: (17, 8)
=====
Спільний секретний ключ КА: (15, 5)
Спільний секретний ключ KB: (15, 5)
```

Рисунок 3.13 – Результат роботи алгоритму ECDH для навчальної кривої з прикладу 2.1

```
Крива: secp256k1
a: 0
b: 7
G: (0x79be667ef9dcbbac55a06295ce870b07029bfcd2dce28d959f2815b16f81798,
0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8)
p: 115792089237316195423570985008687907853269984665640564039457584007908834671663
=====
Закритий ключ абонента А: 0xfe40c94c2c4467ea15aaea9db8a31f27c43269f89c4d1e80cdb4aa227ff49709
Відкритий ключ абонента А: (0x22755bb91b8565759895ee286b840f18f324eeac4a39dd875bbd4aeb844ab25c,
0x3de0508d86c7cbe382920c43fed7337d5856be30ab8e3d6f452ea707b72130b9)
Закритий ключ абонента В: 0x951fbbcb681cab46d54a74c255fe44603c46586c97af4a6550a1f616b15b0fcf7
Відкритий ключ абонента В: (0x71a42564d05deea5c7f5e2e77dc5c09ec75ca6a0cf3a113fb6b54ca12aa123dd,
0x6ce7e6dc2eac912370d66ed2f0cfe174888ce5cfef720e8f699079e6d2e7ef59)
=====
Спільний секретний ключ КА: (0x8bcf37dfbdlcbf715b0c1aec3fa220ece51f65c640682c3464a0f771b535db0c,
0xe54fabe6c0fef0297b134a64c07fd4f1785d55889796597224f8b5a95d009cb2)
Спільний секретний ключ KB: (0x8bcf37dfbdlcbf715b0c1aec3fa220ece51f65c640682c3464a0f771b535db0c,
0xe54fabe6c0fef0297b134a64c07fd4f1785d55889796597224f8b5a95d009cb2)
```

Рисунок 3.14 – Результат роботи алгоритму ECDH для кривої *secp256k1*

Результати тестування роботи алгоритму ECDH для різних вхідних даних наведено у таблиці 3.1. Отримані секретні ключі відповідають очікуваним значенням, що свідчить про правильну реалізацію алгоритму ECDH.

Результати роботи алгоритму ECDH для різних вхідних даних

Тест 1
<i>Вхідні дані:</i> Крива: <i>secp256k1</i> $x = 0\text{xef}720\text{ee}1\text{de}1\text{a}875\text{f}78\text{fd}7\text{d}05\text{f}58\text{bbec}6969\text{ad}1831792\text{d}0530\text{e}8\text{c}0200543798\text{e}$ $y = 0\text{x}951\text{fbb}681\text{cab}46\text{d}54\text{a}74\text{c}255\text{fe}44603\text{c}46586\text{c}97\text{af}4\text{a}6550\text{a}1\text{f}616\text{b}15\text{b}0\text{fc}7$
<i>Вихідні дані:</i> $X_A = (0\text{x}22755\text{bb}91\text{b}8565759895\text{ee}286\text{b}840\text{f}18\text{f}324\text{eeac}4\text{a}39\text{dd}875\text{bbd}4\text{aeb}844\text{ab}25\text{c},$ $0\text{x}3\text{de}0508\text{d}86\text{c}7\text{cbe}382920\text{c}43\text{fed}7337\text{d}5856\text{be}30\text{ab}8\text{e}3\text{d}6\text{f}452\text{ea}707\text{b}72130\text{b}9)$ $Y_B = (0\text{x}71\text{a}42564\text{d}05\text{deea}5\text{c}7\text{f}5\text{e}2\text{e}77\text{dc}5\text{c}09\text{ec}75\text{ca}6\text{a}0\text{cf}3\text{a}113\text{fb}6\text{b}54\text{ca}12\text{aa}123\text{dd},$ $0\text{x}6\text{ce}7\text{e}6\text{dc}2\text{eac}912370\text{d}66\text{ed}2\text{f}0\text{cf}174888\text{ce}5\text{cfef}720\text{e}8\text{f}699079\text{e}6\text{d}2\text{e}7\text{ef}59)$ Спільний секретний ключ: $K = (0\text{x}8\text{bcf}37\text{dfb}1\text{cbf}715\text{b}0\text{c}1\text{aec}3\text{fa}220\text{ece}51\text{f}65\text{c}640682\text{c}3464\text{a}0\text{f}771\text{b}535\text{db}0\text{c},$ $0\text{x}\text{e}54\text{fab}\text{e}6\text{c}0\text{fef}0297\text{b}134\text{a}64\text{c}07\text{fd}4\text{f}1785\text{d}55889796597224\text{f}8\text{b}5\text{a}95\text{d}009\text{cb}2)$
Тест 2
<i>Вхідні дані:</i> Крива: P-256 $x = 0\text{x}4\text{c}58\text{f}50\text{b}8\text{f}74\text{d}3\text{e}13680\text{f}63\text{e}79108\text{e}321\text{ae}24305\text{d}477\text{d}487\text{d}91647\text{efd}8\text{f}033\text{d}$ $y = 0\text{xac}673188\text{ded}871\text{a}069\text{ad}8\text{d}1\text{a}792\text{e}5\text{be}7\text{f}3\text{e}158\text{e}6901\text{e}5\text{f}6\text{e}2239\text{b}141\text{a}08\text{cba}7\text{f}$
<i>Вихідні дані:</i> $X_A = (0\text{x}6\text{cd}7257\text{dd}3\text{c}88\text{ae}84\text{eb}86\text{cf}7\text{b}282\text{d}0\text{ca}4\text{ceff}7\text{ff}0901\text{b}9\text{f}4\text{ef}1794706\text{d}5\text{fbe}4\text{c},$ $0\text{x}894\text{cd}5\text{af}6680\text{e}50\text{c}61\text{b}3\text{db}983\text{b}321614492\text{c}50491\text{e}35\text{e}0\text{da}0797\text{b}7\text{c}084\text{f}5\text{b}3\text{e}3)$ $Y_B = (0\text{x}53\text{b}6\text{d}4\text{e}7\text{a}716\text{df}9\text{f}6364\text{c}6\text{cf}974\text{bf}8484\text{add}16090\text{c}0341\text{fc}5838\text{b}4\text{b}7\text{f}643\text{a}844,$ $0\text{x}1816\text{a}9\text{ac}21\text{fd}77\text{a}8\text{cdb}2201\text{dce}1177944\text{e}7\text{eececc}0\text{e}526\text{b}3\text{ac}85\text{d}68\text{c}17\text{c}464\text{b})$ Спільний секретний ключ: $K = (0\text{xd}65\text{e}55\text{d}9\text{c}82\text{e}4\text{a}51\text{a}7\text{b}03\text{c}47722785\text{d}51\text{ff}141\text{ed}93\text{e}344\text{d}93\text{dea}91\text{d}495613360,$ $0\text{xd}6266\text{b}5\text{a}89\text{d}02\text{c}2\text{c}0\text{bb}420\text{a}80\text{a}9\text{f}15\text{f}72\text{f}3\text{ce}72231\text{b}087001\text{b}40\text{f}23\text{e}9\text{d}2\text{d}5\text{b}59)$
Тест 3
<i>Вхідні дані:</i> Крива: <i>Curve25519</i> $x = 0\text{x}4\text{e}21\text{d}86\text{ee}85254\text{c}09655\text{c}4\text{ea}4\text{fca}77\text{e}0\text{fb}8159\text{eb}37\text{c}2893\text{dd}0279160\text{e}07\text{e}046$ $y = 0\text{x}848216\text{b}34\text{d}8\text{b}177\text{a}743\text{e}93\text{e}05\text{efc}1571791\text{dd}4\text{a}28\text{e}8\text{f}38\text{b}1\text{b}0341\text{bdab}91\text{f}9\text{c}4$
<i>Вихідні дані:</i> $X_A = (0\text{x}163\text{e}3\text{b}2\text{e}91420\text{ec}9603\text{f}798144\text{be}1\text{a}11\text{bcf}50860\text{aa}3\text{a}01\text{bfc}59\text{a}40\text{be}6\text{c}97\text{d}02\text{e},$ $0\text{x}111\text{fe}9\text{fe}481220\text{a}4\text{e}83\text{dce}56\text{c}01\text{d}704130\text{c}145730\text{b}7\text{fd}8\text{c}58\text{addfe}05\text{d}98\text{c}520)$ $Y_B = (0\text{x}7\text{e}294485\text{f}730\text{c}68\text{a}6382\text{d}54660\text{fad}38\text{bbe}04\text{fae}70\text{f}7212\text{b}093\text{b}80\text{c}536\text{d}68320,$ $0\text{x}610\text{f}1901\text{d}874\text{e}279834\text{bae}5362\text{cf}974\text{d}74\text{e}6\text{c}05\text{c}57\text{b}6\text{fe}00\text{d}2\text{c}935\text{c}2\text{e}5\text{dc}59\text{ef})$ Спільний секретний ключ: $K = (0\text{x}599327466718\text{a}1\text{da}48\text{f}56\text{f}31\text{e}0415\text{f}708\text{a}37\text{ec}68\text{af}376\text{b}3\text{b}469136\text{ce}4\text{a}6\text{ecac}7,$ $0\text{xd}790\text{d}112952\text{a}238\text{ab}331220564\text{b}856\text{bd}8876\text{ea}8\text{bcc}086443\text{fb}828795\text{f}5\text{ace}1\text{f})$

Алгоритм цифрового підпису на еліптичних кривих. Скріншот результатів роботи алгоритму цифрового підпису на еліптичних кривих для навчальної кривої з прикладу 2.2 та кривої *secp256k1* наведено нижче (Рис.3.15, Рис.3.16). З прикладу видно, що результат, обчислений при перевірці підпису, співпадає з r . Це вказує на дійсність підпису та відображає коректну роботу алгоритму перевірки цифрового підпису ECDSA.

```
Повідомлення: 10
Закритий ключ підписувача = 3
Відкритий ключ підписувача = (13, 22)

Сесійний ключ k = 19

Підпис:
r=9
s=8

Перевірка підпису:
w = 3
u1 = 7
u2 = 4
P = (9, 17)

Результат:
r=9
!!!Підпис дійсний!!!
```

Рисунок 3.15 – Результат роботи алгоритму ECDSA для навчальної кривої з прикладу 2.2

```
Повідомлення: hello world
Закритий ключ підписувача = 0xef720ee1de1a875f78fdf7d05f58bbec6969ad1831792d0530e8c0200543798e
Відкритий ключ підписувача = (0xc987f2353559343cf7b9deeb92acdb6c4d386ea00dfc9129fa62ee62a33a8e4c,
0x2baad990f69e2e9f51c888bed67c06b8a03a7346fd923487c006eeec0c0a3096)

Сесійний ключ k = 0xfc8d2cfc123080429d36beb59e1213426582174b7ce69f5a5efba05c300c75ae

Підпис:
r=0x28608271241e0c555a0e536dc8e8675786deeb25e672814c38ae1192b68912e4
s=0x4932e7e9024e34a71593ea6f47ef4496c06f7059f33c3668b8ebf4c4c6e94b36

Перевірка підпису:
w = 0xa6785d6fa4d8725e786c11baef5dcb47844d1b6f0e924e0dbd2484141372030a
u1 = 0x75ee28c62c85df6063ea2c7c46daaf0e05e7bb11a466732a38abc5103cc9a314
u2 = 0xf3e04ce3961098bab789734b0725c318302714d8f579d257374a4a609e095c96
P = (0x28608271241e0c555a0e536dc8e8675786deeb25e672814c38ae1192b68912e4,
0xc501ba567586285c15ff3f0c25b5d4ee6d04051c53d1f4bdae50e85f2ee38c20)
Результат: = 0x28608271241e0c555a0e536dc8e8675786deeb25e672814c38ae1192b68912e4
!!!Підпис дійсний!!!
```

Рисунок 3.16 – Результат роботи алгоритму ECDSA для кривої *secp256k1*

Результати тестування роботи алгоритму ECDSA для різних вхідних даних та різних кривих наведено у таблиці 3.2. Кожен тест показує успішне створення підпису для вказаного повідомлення та секретного ключа. Отримані значення r і s є коректними для алгоритму ECDSA.

Результати роботи алгоритму ECDSA для різних вхідних даних

Тест 1
<i>Вхідні дані:</i> Крива: P-256 Повідомлення: sample (SHA-256) $d = 0xc9afa9d845ba75166b5c215767b1d6934e50c3db36e89b127b8a622b120f6721$ $k = 0xa6e3c57dd01abe90086538398355dd4c3b17aa873382b0f24d6129493d8aad60$
<i>Вихідні дані:</i> $Q = (0x60fed4ba255a9d31c961eb74c6356d68c049b8923b61fa6ce669622e60f29fb6,$ $0x7903fe1008b8bc99a41ae9e95628bc64f2f1b20c2d7e9f5177a3c294d4462299)$ Підпис: $r = 0xefd48b2aacb6a8fd1140dd9cd45e81d69d2c877b56aaf991c34d0ea84eaf3716$ $s = 0xf7cb1c942d657c41d436c7a1b6e29f65f3e900dbb9aff4064dc4ab2f843acda8$
Тест 2
<i>Вхідні дані:</i> Крива: P-512 Повідомлення: test (SHA-1) $d = 0xfad06daa62ba3b25d2fb40133da757205de67f5bb0018fee8c86e1b68c7e75caa896eb32$ $f1f47c70855836a6d16fcc1466f6d8fbec67db89ec0c08b0e996b83538$ $k = 0xbb9f2bf4fe1038ccf4dabd7139a56f6fd8bb1386561bd3c6a4fc818b20df5ddba80795a9$ $47107a1ab9d12daa615b1ade4f7a9dc05e8e6311150f47f5c57ce8b222$
<i>Вихідні дані:</i> $Q = (0x1894550d0785932e00eaa23b694f213f8c3121f86dc97a04e5a7167db4e5bcd371123$ $d46e45db6b5d5370a7f20fb633155d38ffa16d2bd761dcac474b9a2f5023a4,$ $0x493101c962cd4d2fddf782285e64584139c2f91b47f87ff82354d6630f746a28a0db$ $25741b5b34a828008b22acc23f924faafbd4d33f81ea66956dfeaa2bfdfcf5)$ Підпис: $r = 0x13bad9f29abe20de37eb823c252ca0f63361284015a3bf430a46aaa80b87b0693$ $f0694bd88afe4e661fc33b094cd3b7963bed5a727ed8bd6a3a202abe009d0367$ $s = 0x1e9bb81ff7944ca409ad138dbbee228e1afcc0c890fc78ec8604639cb0dbdc90f71$ $7a99ead9d272855d00162ee9527567dd6a92cbd629805c0445282bbc916797ff$
Тест 3
<i>Вхідні дані:</i> Крива: <i>secp256k1</i> Повідомлення: hello world (SHA-256) $d = 0xef720ee1de1a875f78fdf7d05f58bbec6969ad1831792d0530e8c0200543798e$ $k = 0xfc8d2cfc123080429d36beb59e1213426582174b7ce69f5a5efba05c300c75ae$
<i>Вихідні дані:</i> $Q = (0xc987f2353559343cf7b9deeb92acdb6c4d386ea00dfc9129fa62ee62a33a8e4c,$ $0x2baad990f69e2e9f51c888bed67c06b8a03a7346fd923487c006eeec0c0a3096)$ Підпис: $r = 0x28608271241e0c555a0e536dc8e8675786deeb25e672814c38ae1192b68912e4$ $s = 0x4932e7e9024e34a71593ea6f47ef4496c06f7059f33c3668b8ebf4c4c6e94b36$

Інтегрована схема шифрування на еліптичних кривих. Результат роботи інтегрованої схеми шифрування на еліптичних кривих для кривої P-256 наведено нижче (Рис.3.17). Для інших кривих результати наведено у таблиці 3.3.

```
Крива: P-256
a: -3
b: 41058363725152142129326129780047268409114441015993725554835256314039467401291
G: (0x6b17d1f2e12c4247f8bce6e563a440f277037d812deb33a0f4a13945d898c296,
0x4fe342e2fe1a7f9b8ee7eb4a7c0f9e162bce33576b315ececbb6406837bf51f5)
p: 115792089210356248762697446949407573530086143415290314195533631308867097853951
=====
Закритий ключ абонента A: 0xc5f3a619b222da1d4187a293bee82411e6ed2f7e9f09e5f9705544d40937e4ec
Відкритий ключ абонента A: (0x5614e7444a1995a14d31c6ca5bcf2d5a55b4175aae4fa1952e9d62440947b267,
0xd9ebbbcd9f5f1a7e9ded92f4c55ee479e1b2f8ce693bdac3e3237afdd1fd6fb9)
Закритий ключ абонента B: 0xe9cde84af23896afc79f548caac8e75de2387e7e6da6da2686d4d1dd01e66907
Відкритий ключ абонента B: (0xe7090c51a57f17e956fff30de49abd2a5b6366204c1cd643c9f2eb7079708c66,
0x860f246c24f6627d11372f7088b9f7ee94d974dc045fffc84672c9cd112f927)
=====
Спільний секретний ключ KA: (0xf09a8f5a7c7fc98d253f286fe31a0214057082c7a087f176241f50603298f8f0,
0xd68e2eb8aa7477aacdb2f338d859d813d1818d5b8ef73eccec02620a259221c9)
Спільний секретний ключ KB: (0xf09a8f5a7c7fc98d253f286fe31a0214057082c7a087f176241f50603298f8f0,
0xd68e2eb8aa7477aacdb2f338d859d813d1818d5b8ef73eccec02620a259221c9)
=====
Секретний ключ x-координата: 0xf09a8f5a7c7fc98d253f286fe31a0214057082c7a087f176241f50603298f8f0
=====
k_ENC: 0x7f40a5f6fa24cc553ba4cbbfef0644c951cfa2ec6eb02897220c3d2284a1e865
k_MAC: 0x3532f18efd3c685ee78f8f0c55962173327e6b4a93a0ad295dd1d7fe588715b2

Повідомлення: hello, Bob!
IV: 0x4f95d86eaf88fab5c960b728b8adc93e
Шифротекст: 0x7dfdf1245ebeec37e210b1
Тег: 0x9c5c9f228885322f0b3657b7eb9c1193
Розшифроване повідомлення: hello, Bob!
```

Рисунок 3.17 – Результат роботи алгоритму ECIES для кривої P-256

Таблиця 3.3

Результати роботи алгоритму ECIES для різних вхідних даних

Тест 1
Вхідні дані: Крива: P-256 Повідомлення: hello, Bob! $x = 0xc5f3a619b222da1d4187a293bee82411e6ed2f7e9f09e5f9705544d40937e4ec$ $y = 0xe9cde84af23896afc79f548caac8e75de2387e7e6da6da2686d4d1dd01e66907$ $IV = 0x4f95d86eaf88fab5c960b728b8adc93e$
Вихідні дані: $X_A = (0x5614e7444a1995a14d31c6ca5bcf2d5a55b4175aae4fa1952e9d62440947b267,$ $0xd9ebbbcd9f5f1a7e9ded92f4c55ee479e1b2f8ce693bdac3e3237afdd1fd6fb9)$ $Y_B = (0xe7090c51a57f17e956fff30de49abd2a5b6366204c1cd643c9f2eb7079708c66,$ $0x860f246c24f6627d11372f7088b9f7ee94d974dc045fffc84672c9cd112f927)$ Спільний секретний ключ: $K = (0xf09a8f5a7c7fc98d253f286fe31a0214057082c7a087f176241f50603298f8f0,$ $0xd68e2eb8aa7477aacdb2f338d859d813d1818d5b8ef73eccec02620a259221c9)$ $K_{ENC} = 0x7f40a5f6fa24cc553ba4cbbfef0644c951cfa2ec6eb02897220c3d2284a1e865$ $K_{MAC} = 0x3532f18efd3c685ee78f8f0c55962173327e6b4a93a0ad295dd1d7fe588715b2$ Шифротекст: 0x7dfdf1245ebeec37e210b1 Тег: 0x9c5c9f228885322f0b3657b7eb9c1193 Розшифроване повідомлення: hello, Bob!

Таблиця 3.3 (продовження)

Результати роботи алгоритму ECIES для різних вхідних даних

Тест 2
<i>Вхідні дані:</i> Крива: P-256 Повідомлення: Elliptic Curve Cryptography! $x = 0\text{xef}320\text{e}178243\text{db}28\text{a}9\text{d}9531073\text{b}63037\text{b}6\text{cfa}5829\text{f}05\text{c}76\text{a}678733\text{a}614\text{bf}42\text{a}1$ $y = 0\text{x}2\text{be}3\text{adee}2\text{f}5880455\text{cd}4\text{de}3\text{dd}7\text{d}86709\text{e}0\text{b}59817\text{deb}2\text{d}4\text{a}1\text{b}9\text{bad}685\text{ccaf}98\text{e}5$ $IV = 0\text{x}132\text{dbc}3\text{f}2\text{cf}3\text{ba}4785347252071\text{c}3\text{a}18$
<i>Вихідні дані:</i> $X_A = (0\text{x}853\text{bb}98\text{c}0500\text{a}94910\text{f}5\text{eb}843\text{b}4438\text{f}7\text{b}4\text{d}faa025\text{b}94\text{b}4789031\text{b}998\text{cb}5\text{d}3\text{dea},$ $0\text{x}fa1\text{ad}687\text{bcb}4\text{be}0\text{dac}f8\text{c}2\text{d}83\text{fafd}8\text{a}8\text{b}5\text{a}3\text{c}4\text{ffc}067\text{ede}65420\text{c}3\text{fecdc}2\text{f}65\text{d})$ $Y_B = (0\text{x}b0658\text{b}3394\text{a}33\text{b}9\text{bf}2\text{d}93\text{bfc}8\text{e}4\text{c}7\text{d}197839\text{b}31\text{6d}7134\text{aeb}4\text{ec}0903\text{c}7\text{d}468776,$ $0\text{x}9\text{bb}8436\text{ec}396\text{b}62112\text{dba}099\text{e}26\text{bde}467\text{c}262969\text{eb}4064\text{d}186\text{c}79\text{d}7\text{cb}08\text{a}6\text{be}8)$ Спільний секретний ключ: $K = (0\text{x}3\text{d}477\text{def}4\text{a}69\text{c}8543\text{e}272\text{d}4\text{b}79781\text{fa}59441\text{ac}74\text{c}0\text{dd}2\text{f}93\text{ff}5\text{f}6985\text{eb}1\text{e}4\text{f}90,$ $0\text{x}2972\text{c}382\text{dfab}04\text{b}94\text{ba}2\text{ab}8\text{eca}14895\text{a}3058\text{a}5735\text{fb}1324\text{e}0\text{e}525\text{b}078\text{e}88\text{ea}71)$ $K_{ENC} = 0\text{x}b1\text{c}341\text{bab}294\text{b}47865\text{dcc}1\text{f}4\text{ccf}3\text{d}5\text{aa}4\text{e}9488957\text{dc}297\text{eaecf}7\text{b}171069\text{bf}80\text{b}$ $K_{MAC} = 0\text{x}1\text{ea}8\text{a}6\text{ccb}32\text{fae}ec\text{faba}076\text{ec}05355\text{cc}2\text{b}8\text{d}69\text{c}6\text{c}02\text{c}76208497286\text{ffaa}54\text{d}7\text{b}$ Шифротекст: $0\text{x}75\text{a}76\text{e}2\text{ff}30\text{b}469\text{bb}7\text{ceaec}312\text{c}356\text{d}8849\text{c}512\text{f}4387\text{cac}11\text{ff}1913\text{f}$ Тег: $0\text{x}87\text{c}8\text{bea}ec6\text{d}229882364\text{cf}6733\text{ba}3\text{f}4\text{a}$ Розшифроване повідомлення: Elliptic Curve Cryptography!
Тест 3
<i>Вхідні дані:</i> Крива: <i>secp256k1</i> Повідомлення: some plaintext for encryption $x = 0\text{x}5\text{d}38\text{f}4\text{f}2\text{d}6\text{cf}4\text{edc}9\text{af}08\text{d}77\text{d}67\text{d}2\text{a}50\text{fb}8\text{f}9424\text{aede}806\text{c}72823382\text{d}150\text{af}0\text{b}$ $y = 0\text{x}471\text{fe}02\text{f}8\text{d}8\text{b}2\text{da}3\text{fbdf}510\text{fe}73\text{eabd}89\text{e}4\text{b}5\text{eb}271\text{fa}92\text{fe}053\text{b}5539526\text{d}2\text{d}27$ $IV = 0\text{x}8976\text{c}0304\text{d}713\text{c}14\text{eda}599\text{f}8763\text{e}7\text{bd}3$
<i>Вихідні дані:</i> $X_A = (0\text{x}ebf5\text{c}388924192\text{fb}2\text{dbc}203\text{fb}4\text{cab}733091\text{b}1\text{b}49\text{ffa}21\text{f}68\text{c}60\text{fb}1\text{f}2\text{a}18\text{fd}351,$ $0\text{x}40\text{f}02\text{a}88\text{bddd}2\text{e}38\text{add}6\text{e}0600\text{d}38394\text{c}8\text{e}39\text{bd}4\text{d}6\text{c}4\text{b}6\text{dfb}66\text{a}8\text{b}75\text{b}8\text{d}0616\text{cb})$ $Y_B = (0\text{x}101\text{dc}5\text{fd}576\text{a}95\text{c}0178\text{abc}2034\text{e}755\text{fbb}81\text{f}9\text{e}4\text{be}1\text{dccf}3\text{a}056607\text{e}970406656,$ $0\text{x}e381\text{bbe}631782\text{a}7\text{ec}5\text{b}0\text{ab}1\text{ddf}601\text{b}061\text{adf}40\text{aa}22\text{faf}9\text{fa}431\text{beeca}87\text{fc}71\text{ca})$ Спільний секретний ключ: $K = (0\text{x}f235\text{cc}76\text{d}55\text{a}4\text{c}72458867\text{c}62\text{db}08\text{f}3\text{ecb}506\text{a}7\text{bd}3\text{b}38535851\text{add}4\text{d}016\text{b}921\text{f},$ $0\text{x}51037\text{f}6\text{a}8383\text{f}33111\text{f}2010\text{d}82\text{edeac}1693392758\text{d}174\text{d}3\text{eacc}54\text{f}45\text{eef}0\text{d}469)$ $K_{ENC} = 0\text{x}da5\text{e}2\text{c}18789953867\text{c}987\text{bdbc}2\text{a}0\text{af}814\text{f}791960\text{bc}334\text{ed}914\text{ff}6\text{b}0\text{b}4\text{c}008962$ $K_{MAC} = 0\text{x}0\text{f}9\text{fd}4037\text{d}832104\text{dc}9431\text{a}8\text{f}6\text{a}19\text{f}0\text{a}2984912\text{dff}1\text{abfc}314\text{a}06959\text{cada}3\text{b}8\text{b}$ Шифротекст: $\text{xcb}8\text{eb}989\text{a}39\text{cb}22\text{b}36\text{a}7\text{bb}16\text{d}1\text{e}4\text{c}260672\text{b}8\text{c}8\text{ad}3\text{f}56\text{f}7\text{f}08\text{cbaa}5052$ Тег: $0\text{x}e29\text{cb}22733131837\text{ff}6\text{ad}1\text{d}60\text{e}1\text{c}155\text{f}$ Розшифроване повідомлення: hello, Bob!

В цілому, результати тестування свідчать про те, що реалізація криптографічних алгоритмів на еліптичних кривих є коректною. Всі алгоритми правильно обробляють вхідні дані, а отримані вихідні значення демонструють коректність у формуванні спільних секретних ключів, підписів та шифруванні повідомлень.

Висновки до Розділу 3

У розділі 3 обґрунтовано вибір мови програмування, розглянуто особливості реалізації криптосистем на еліптичних кривих. Наведено деталі розробки програмного коду, який містить реалізацію алгоритмів ECDH, ECDSA та ECIES.

Детально описано роботу модуля `ess`, що містить функції для основних операцій над точками еліптичних кривих та модуля `aes-gcm`, що містить функції для генерації ключів, шифрування та дешифрування даних за алгоритмом AES-GCM. Представлені результати тестування для різних вхідних даних, у тому числі на прикладах тестових векторів із офіційної документації. Результати тестування програми підтверджують коректність роботи криптографічних систем.

ВИСНОВКИ

Еліптичні криві є одним із найперспективніших інструментів для побудови сучасних криптографічних алгоритмів. Криптографія на еліптичних кривих дає змогу реалізувати захищений обмін даними між різними сторонами, використовуючи алгоритми шифрування та підписування на основі еліптичних кривих. Еліптичні криві дозволяють досягти еквівалентного рівня безпеки з меншими розмірами ключів порівняно з іншими асиметричними криптографічними алгоритмами.

В результаті проведеного дослідження були отримані наступні результати:

- описано математичний апарат еліптичних кривих, що використовуються для криптографічних цілей, наведено основні операції в групі точок еліптичних кривих, такі як додавання точок, подвоєння точки та скалярне множення точки на число;
- проведено огляд стандартів, що встановлюють рекомендації та вимоги щодо використання еліптичних кривих у криптографічних системах;
- проаналізовано переваги криптографії на еліптичних кривих порівняно із традиційними асиметричними алгоритмами, такі як менші розміри ключів, швидкість обчислень та ефективне використання ресурсів;
- розглянуто потенційні загрози та вразливості криптографічних алгоритмів на основі еліптичних кривих;
- розглянуто кроки і принципи роботи криптосистем обміну ключами Діффі-Хеллмана (ECDH), цифрового підпису (ECDSA) та схеми шифрування (ECIES) на еліптичних кривих;
- розроблено програмне забезпечення для демонстрації роботи розглянутих криптосистем;
- представлено результати тестування програми, які підтверджують коректність роботи криптографічних алгоритмів.

Результати роботи можуть бути використані для підвищення ефективності та надійності систем кібербезпеки, які вимагають застосування сучасних криптографічних технологій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kissel R. Glossary of key information security terms. National Institute of Standards and Technology, 2013. URL: <https://doi.org/10.6028/nist.ir.7298r2>
2. Богущ В. М. Інформаційна безпека: Термінологічний навчальний довідник/ В. Г. Кривуца, А. М. Кудін // За ред. Кривуци В.Г. – К.: ООО «Д.В.К.» 2004. – 508 с.
3. Schneier B. Foundations. Applied Cryptography, Second Edition. Indianapolis, Indiana, 2015. P. 1–18. URL: <https://doi.org/10.1002/9781119183471.ch1>
4. Koblitz N. Elliptic curve cryptosystems. Mathematics of computation. 1987. Vol. 48, no. 177. P. 203. URL: <https://doi.org/10.1090/s0025-5718-1987-0866109-5>
5. Miller V.S. Use of elliptic curves in cryptography. Lecture notes in computer science. Berlin, Heidelberg. P. 417–426. URL: https://doi.org/10.1007/3-540-39799-x_31
6. Menezes A. Introduction to public key cryptography. Elliptic curve public key cryptosystems. Boston, MA, 1993. P. 1–14. URL: https://doi.org/10.1007/978-1-4615-3198-2_1
7. Vanstone S. Elliptic curve cryptosystem – The answer to strong, fast public-key cryptography for securing constrained environments. Information security technical report. 1997. Vol. 2, no. 2. P. 78–87. URL: [https://doi.org/10.1016/s1363-4127\(97\)81331-3](https://doi.org/10.1016/s1363-4127(97)81331-3)
8. Bernstein D. J., Lange T. Hyper-and-elliptic-curve cryptography. LMS journal of computation and mathematics. 2014. Vol. 17, A. P. 181–202. URL: <https://doi.org/10.1112/s1461157014000394>
9. Lange T. Edwards curves. Encyclopedia of cryptography and security. Boston, MA, 2011. P. 380–382. URL: https://doi.org/10.1007/978-1-4419-5906-5_243
10. Washington L. C. Elliptic curves: number theory and cryptography. 2nd ed. Boca Raton, FL: Chapman & Hall/CRC, 2008.
11. Edwards H. M. A normal form for elliptic curves. Bulletin of the american mathematical society. 2007. Vol. 44, no. 03. P. 393–423. URL: <https://doi.org/10.1090/s0273-0979-07-01153-6>
12. Юдін О., Вадясов К. Method of elliptic curves imposition in cryptographic protection tasks of graphic information. Science-based technologies. 2010. Т. 6, № 2. URL: <https://doi.org/10.18372/2310-5461.6.5202>

13. Чевардін В.Є., Пономарьов О.А. Перспективи розвитку криптосистем на основі перетворень в групі точок еліптичних кривих. Збірник наукових праць ВІТІ № 1. 2020. URL: https://viti.edu.ua/files/zbk/2020/12_1_2020.pdf
14. Бессалов А.В. Эллиптические кривые в форме Эдвардса и криптография: монография. Киев: изд-во «Политехника», 2017. 272 с.
15. Циганкова О.В. Методи підвищення швидкодії асиметричних криптосистем з використанням еліптичних кривих у формі Едвардса: дис... канд. техн. наук: 05.13.21. Київ, 2021. 145 с.
16. Беспалов О. Ю., Кучинська Н. В. Крива Едвардса над кільцем лишків як декартів добуток кривих Едвардса над скінченими полями. Прикладная радиоэлектроника. 2017. Т. 16, № 3-4. С. 170-175. URL: http://nbuv.gov.ua/UJRN/Prre_2017_16_3-4_13
17. Ковальчук Л.В., Бессалов А.В., Беспалов О.Ю. Порівняльний аналіз алгоритмів генерації базової точки на кривій Едвардса. Безпека інформації у інформаційно-телекомунікаційних системах: матеріали XVII міжнар. наук.-прак. конференції, 26-28 травня 2015р., м. Київ. С. 32-33.
18. Skuratovskii R. Supersingular edwards curves and edwards curve points counting method over finite field. Journal of numerical and applied mathematics. 2020. No. 1 (133). P. 68–88. URL: <https://doi.org/10.17721/2706-9699.2020.1.06>
19. Ільєнко А.В., Ільєнко С.С., Мазур Я.С., Прокопенко, О.В. Перспективи використання еліптичної криптографії для забезпечення цілісності та конфіденційності інформації. Вісник Університету «Україна» Серія Інформатика, обчислювальна техніка та кібернетика, 2021, 2(23). URL: <https://visn-it.uu.edu.ua/index.php/visn-icct/article/view/61>
20. Нікуліщев Г.І. Протокол сліпого електронного цифрового підпису на еліптичних кривих над скінченим векторним полем. Радіоелектроніка, інформатика, управління, 2013 (2). URL: <https://doi.org/10.15588/1607-3274-2013-2-12>
21. Meleshko O. O., Kovalskiy O. O. Elliptic curve cryptography. Science-based technologies. 2014. Vol. 22, no. 2. URL: <https://doi.org/10.18372/2310-5461.22.6815>
22. Harkanson R., Kim Y. Applications of elliptic curve cryptography: a light introduction to elliptic curves and a survey of their applications. CISRC'17: twelfth annual cyber and information security research conference, Oak Ridge Tennessee USA. New York, NY, USA, 2017. URL: <https://doi.org/10.1145/3064814.3064818>

23. ANSI X9.62. Public key cryptography for the financial services industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). American National Standards Institute, Washington, 1999.
24. ANSI X9.63. Public key cryptography for the financial services industry: Key agreement and key transport using elliptic curve cryptography. American National Standards Institute, Washington, 2001.
25. FIPS 186-5. Digital Signature Standard (DSS). National Institute of Standards and Technology, U.S. Department of Commerce, Washington, 2013. URL: <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf>
26. ISO/IEC 15946-4. Information technology. Security techniques. Cryptographic techniques based on elliptic curves. Part 4: Digital signatures giving message recovery, Geneva, 2004.
27. ISO/IEC 15946-5. Information technology. Security techniques. Cryptographic techniques based on elliptic curves. Part 5: Elliptic curve generation, Geneva, 2022.
28. Saho N.J.G., Ezin E.C. Comparative study on the performance of elliptic curve cryptography algorithms with cryptography through RSA algorithm, CARI, 2020. URL: <https://hal.science/hal-02926106/document?ref=panther-protocol-blog>
29. SafeCurves: choosing safe curves for elliptic-curve cryptography. URL: <https://safecurves.cr.yp.to/>
30. ДСТУ 4145-2002. Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевіряння. Київ, Держстандарт України, 2002.
31. ДСТУ 9041-2020. Інформаційні технології. Криптографічний захист інформації. Алгоритм шифрування коротких повідомлень, що ґрунтується на скручених еліптичних кривих Едвардса, 2020.
32. Blake I. F., Smart N. P., Seroussi G. Advances in elliptic curve cryptography. Cambridge University Press, 2009.
33. Nir Y., Josefsson S., Pegourie-Gonnard M. Elliptic curve cryptography (ECC) cipher suites for transport layer security (TLS) versions 1.2 and earlier. RFC Editor, 2018. URL: <https://doi.org/10.17487/rfc8422>

ДОДАТКИ

Лістинг програмного коду есс.ру

```
import collections
import random

EllipticCurve = collections.namedtuple('EllipticCurve', 'name p a b g n h')

curve = EllipticCurve(
    'secp256k1',
    # Модуль поля
    p=0xfffffffffffffffffffffffffffffffffffffffffffffffffeffffc2f,
    # Коефіцієнти
    a=0,
    b=7,
    # Базова точка
    g=(0x79be667ef9dcbbac55a06295ce870b07029bfcd2dce28d959f2815b16f81798,
        0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8),
    # Порядок підгрупи
    n=0xffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141,
    # Кофактор підгрупи
    h=1,
)

def inverse_mod(k, p):
    """Повертає мультиплікативно обернений елемент до k за модулем p
    (x * k) % p == 1
    """
    if k == 0:
        raise ZeroDivisionError('Ділення на 0')

    if k < 0:
        # k ** -1 = p - (-k) ** -1 (mod p)
        return p - inverse_mod(-k, p)

    # Розширений алгоритм Евкліда
    v, old_v = 0, 1
    u, old_u = 1, 0
    r, old_r = p, k

    while r != 0:
        quotient = old_r // r
        old_r, r = r, old_r - quotient * r
        old_v, v = v, old_v - quotient * v
        old_u, u = u, old_u - quotient * u

    gcd, x, y = old_r, old_v, old_u

    assert gcd == 1
    assert (k * x) % p == 1

    return x % p

def is_on_curve(point):
    """Повертає True, якщо точка належить еліптичній кривій"""
    if point is None:
        # None - нескінченно відділена точка
        return True

    x, y = point

    return (y * y - x * x * x - curve.a * x - curve.b) % curve.p == 0
```

```

def point_add(point1, point2):
    """Повертає результат додавання point1 + point2"""
    assert is_on_curve(point1)
    assert is_on_curve(point2)

    if point1 is None:
        # 0 + point2 = point2
        return point2
    if point2 is None:
        # point1 + 0 = point1
        return point1

    x1, y1 = point1
    x2, y2 = point2

    if x1 == x2 and y1 != y2:
        # point1 + (-point1) = 0
        return None

    if x1 == x2:
        # Додавання точок
        l = (3 * x1 * x1 + curve.a) * inverse_mod(2 * y1, curve.p)
    else:
        # Подвоєння точок
        l = (y2 - y1) * inverse_mod(x2 - x1, curve.p)

    x3 = l * l - x1 - x2
    y3 = l * (x1 - x3) - y1
    result = (x3 % curve.p, y3 % curve.p)

    assert is_on_curve(result)

    return result

def scalar_mult(m, point):
    """Повертає результат m * point"""
    assert is_on_curve(point)

    if m % curve.n == 0 or point is None:
        return None

    if m < 0:
        # m * point = -m * (-point)
        return scalar_mult(-m, point_neg(point))

    result = None
    addend = point

    while m:
        if m & 1:
            # Додавання
            result = point_add(result, addend)

            # Подвоєння
            addend = point_add(addend, addend)

        m >>= 1 #порозрядний зсув вправо

    assert is_on_curve(result)

    return result

def make_keypair():
    """Закритий ключ генерується випадково, відкритий обчислюється"""
    private_key = random.randrange(1, curve.n)
    public_key = scalar_mult(private_key, curve.g)

    return private_key, public_key

```

Лістинг програмного коду esdh.py

```

import sys
import collections
import hashlib
import random
import binascii

from ecc import curve, scalar_mult, make_keypair

# Виведення параметрів кривої
print(f"Крива:\t {curve.name}")
print(f"a:\t {curve.a}")
print(f"b:\t {curve.b}")
print(f"G:\t (0x{curve.g[0]:x}, \n\t 0x{curve.g[1]:x}) ")
print(f"p:\t {curve.p}")

print("=====")

# Створення ключів для абонентів А та В
ASecretKey, APublicKey = make_keypair()
BSecretKey, BPublicKey = make_keypair()

# Виведення ключів
print(f"Закритий ключ абонента А: {hex(ASecretKey)}")
print(f"Відкритий ключ абонента А: (0x{APublicKey[0]:x}, \n\t\t\t\t\t 0x{APublicKey[1]:x}) ")
print(f"Закритий ключ абонента В: {hex(BSecretKey)}")
print(f"Відкритий ключ абонента В: (0x{BPublicKey[0]:x}, \n\t\t\t\t\t 0x{BPublicKey[1]:x}) ")

# Обчислення та виведення спільного секретного ключа
sharedSecret1 = scalar_mult(BSecretKey, APublicKey)
sharedSecret2 = scalar_mult(ASecretKey, BPublicKey)

print("=====")
print(f"Спільний секретний ключ КА: (0x{sharedSecret1[0]:x}, \n\t\t\t\t\t 0x{sharedSecret1[1]:x})")
print(f"Спільний секретний ключ KB: (0x{sharedSecret2[0]:x}, \n\t\t\t\t\t 0x{sharedSecret2[1]:x})")

```

Лістинг програмного коду ecdsa.py

```

import sys
import random
import hashlib
import libnum

from ecc import curve, scalar_mult, point_add, make_keypair

msg="hello world"

if (len(sys.argv)>1):
    msg=(sys.argv[1])

# Пара ключів абонента А (підписувача)
dA,QA = make_keypair()

# Обчислення хешу
h=int(hashlib.sha256(msg.encode()).hexdigest(),16)

# Сесійний ключ
k = random.randint(0, curve.n-1)
rpoint = scalar_mult(k,curve.g)

# Підписування
r = rpoint[0] % curve.n
inv_k = libnum.invmod(k,curve.n)
s = (inv_k*(h+r*dA)) % curve.n

print ("Повідомлення:", msg)
print ("Закритий ключ підписувача =", hex(dA))
print(f"Відкритий ключ підписувача = (0x{QA[0]:x}, \n\t\t\t\t\t 0x{QA[1]:x}) ")
print ("\nCесійний ключ k =",hex(k))
print(f"\nПідпис:\nr={hex(r)}\ns={hex(s)}")

# Перевірка підпису
inv_s = libnum.invmod(s,curve.n)
w = inv_s
u1=(h*w) % curve.n
u2=(r*w) % curve.n
P = point_add(scalar_mult(u1,curve.g), scalar_mult(u2,QA))

print ("\nПеревірка підпису:")
print ("w =", hex(w))
print ("u1 =", hex(u1))
print ("u2 =",hex(u2))
print(f"P =(0x{P[0]:x}, \n\t\t\t\t\t 0x{P[1]:x}) ")

res = P[0] % curve.n
print ("Результат: =", hex(res))

if (res==r):
    print("!!!Підпис дійсний!!!")
else:
    print("!!!Підпис НЕдійсний!!!")

```

Лістинг програмного коду aes_gcm.py

```

import random
import os
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.primitives import hmac
from cryptography.exceptions import InvalidTag

def derive_keys(secret_key, salt):
    # Перетворення великого цілого числа в байтовий рядок
    secret_key_bytes = secret_key.to_bytes((secret_key.bit_length() + 7) // 8,
byteorder='big')
    # Генерація ключів за допомогою KDF
    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        iterations=1000,
        salt=salt,
        length=64,
        backend=default_backend()
    )

    key_material = kdf.derive(secret_key_bytes)
    k_ENC = key_material[:32] # 256 біт для шифрування
    k_MAC = key_material[32:] # 256 біт для MAC

    return k_ENC, k_MAC

def encrypt_and_tag(k_ENC, k_MAC, plaintext):
    # Генерація випадкового IV (Initialization Vector)
    iv = os.urandom(16)
    # Ініціалізація об'єкта AES-GCM
    cipher = Cipher(algorithms.AES(k_ENC), modes.GCM(iv),
backend=default_backend())
    encryptor = cipher.encryptor()
    # Шифрування даних
    ciphertext = encryptor.update(plaintext) + encryptor.finalize()
    # Отримання тегу автентифікації
    tag = encryptor.tag

    return iv, ciphertext, tag

def decrypt_and_verify(k_ENC, k_MAC, iv, ciphertext, tag):
    try:
        # Ініціалізація об'єкта AES-GCM
        cipher = Cipher(algorithms.AES(k_ENC), modes.GCM(iv, tag),
backend=default_backend())
        decryptor = cipher.decryptor()
        # Розшифрування даних
        plaintext = decryptor.update(ciphertext) + decryptor.finalize()

        return plaintext
    except InvalidTag:
        # Обробка винятку, коли тег автентифікації не вірний
        print("Помилка: Тег автентифікації не вірний або дані були змінені.")
        return None

```

