Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

Факультет компютерно- інформаційних систем і програмної інженерії

(повна назва факультету)

Кафедра кібербезпека

(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр

(назва освітнього ступеня)

на тему: Аналіз та реалізація криптографічних перетворень для алгоритму ECDH (Elliptic Curve Diffie-Hellman)

Виконав (ла): студент IV курсу, групи СБс-42
Спеціальності 125 Кібербезпека
(шифр і назва спеціальності)

_____ Сава Л.М.
(підпис) (прізвище та ініціали)

Керівник _____ Карпінський М.П.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ _____
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Загородна Н.В.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

2022
Тернопіль

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет компютерно- інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
« » 20_22_ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)

студенту Саві Луці Михайловичу
(прізвище, ім'я, по батькові)

1. Тема роботи Аналіз та реалізація криптографічних перетворень для алгоритму ECDH (Elliptic Curve Diffie-Hellman)

Керівник роботи Карпінський М.П., д.т.н. професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*23*» *березня* 2022_ року № *4/7-178.*

2. Термін подання студентом завершеної роботи 17.06.2022

3. Вихідні дані до роботи

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз предметної області. 1.1 Алгоритм Elliptic Curve Cryptography – ECC. 1.2 Недоліки використання алгоритму ECC. 1.3 Ключі ECC. 2 Дослідження еліптичних кривих. 2.1 Множення точки ECC на ціле число. 2.2 Порядок і кофактор еліптичної кривої. 2.3 Приватний ключ, відкритий ключ і генераторна точка в ECC. 2.4 Група точок елліптичної кривої. 3 Програмна реалізація алгоритму шифрування. 3.1 Реалізація множення в ЕК. 3.2 Стиснення відкритих ключів. 3.3 Вибір параметрів еліптичної кривої. 3.4 Реалізація бміну ключами ECDH, шифрування та дешифрування. 4 Безпека життєдіяльності, основи охорони праці. 4.1 Загальні вимоги охорони праці при роботі з ПК. 4.2 Обов'язки працівника. Висновки. Перелік джерел посилань.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титулка 2. Мета та задачі. 3. Актуальність дослідження. 4. Еліптична крива. 5. Порівняння асиметричних алгоритмів. 6 – Візуалізація кривої та інструменти побудови еліптичних кривих. 7. Множення точки ЕК. 8. Фрагмент коду для виводу точок навчальної ЕК. 9. Результати виконання коду для виводу точок навчальної ЕК. 10. Фрагмент коду, зі зміненою точкою генератора. 11. Результати виконання коду для виводу точок навчальної ЕК. 12. Фрагмент коду для кривої secp192r1. 12. Результат виконання коду для кривої secp192r1. 13. Робота з ключами. 14. Визначення еліптичної кривої. 15. Схема шифрування та дешифрування ECDH. 16. Гібридна схема шифрування. 17. Гібридна схема дешифрування. 18. Реалізація алгоритму обміну ключами ECDH. 19. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці			

7. Дата видачі завдання 23.03.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Ознайомлення з завданням до кваліфікаційної роботи	23.03 – 28.03	*Виконано*
2	Підбір джерел про шифрування асиметричними алгоритмами	29.03 – 05.04	*Виконано*
3	Опрацювання джерел в галузі дослідження	06.04 – 11.04	*Виконано*
4	Дослыдження асиметричного алгоритму шифрування на основі еліптичних кривих	12.04 – 18.04	*Виконано*
5	Розроблення програмного коду	19.04 – 25.04	*Виконано*
6	Оформлення розділу “ Аналіз предметної області ”	26.04 – 29.04	*Виконано*
7	Оформлення розділу “ Дослідження еліптичних кривих ”	02.05 – 05.05	*Виконано*
8	Оформлення розділу “ Програмна реалізація алгоритму шифрування ”	06.05 – 11.05	*Виконано*
9	Виконання завдання до підрозділу “Безпека життєдіяльності, основи хорони праці”	12.05 – 16.05	*Виконано*
10	Оформлення кваліфікаційної роботи	17.05 – 08.06	*Виконано*
11	Нормоконтроль	09.06 – 13.06	*Виконано*
12	Перевірка на плагіат	14.06 – 15.06	*Виконано*
13	Попередній захист кваліфікаційної роботи	16.06 – 17.06	*Виконано*
14	Захист кваліфікаційної роботи	24.06	

Студент _____ Сава Л.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Карпінський М.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Аналіз та реалізація криптографічних перетворень для алгоритму ECDH (Elliptic Curve Diffie-Hellman) // Кваліфікаційна робота ОР «Бакалавр» // Сава Лука Михайлович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБс-42 // Тернопіль, 2021 // С.78, рис. – 37, табл. –2, кресл. – , додат. – 1 .

Ключові слова: RSA, AES, ECC, ECDH, ECDLP, ЕК, SSL

Кваліфікаційна робота присвячена аналізу можливостей асиметричної схеми шифрування алгоритмом Діффі Хеллмана на еліптичних кривих та створенню відповідних програмних модулів.

Метою даного дипломного проекту є аналіз асиметричного алгоритму Діффі Хеллмана на основі використання еліптичних кривих та розробка програмних модулів для виконання: генерації ключів, шифрування інформації та використання гібридної схеми шифрування.

Для досягнення поставленої мети були вирішені такі завдання:

- проведено аналіз алгоритму ECC;
- виявлено недоліки та переваги алгоритму ECC;
- досліджено алгоритм ноження точки ECC на ціле число та порядок, кофактор еліптичної кривої;
- досліджено схеми генерації ключів асиметричної криптографії;
- досліджено особливості генерації ключів алгоритму ECDH та процесу шифрування;
- виконано огляд методів використання списків контролю доступу;
- розроблено і реалізовано програмні модулі для генерації ключів ECDH, шифрування та дешифрування.

ABSTRACT

Analysis and implementation of cryptographic transformations for the ECDH algorithm (Elliptic Curve Diffie-Hellman) // Qualification of the work of the OR "Bachelor" // Sava Luka Mikhailovich // Ternopil National Technical University named after Ivan Pulyui, Faculty of Computer and Information Systems, Department of Software Engineering cybersecurity, SBs-42 group // Ternopil, 2021 // P.78, fig. - 37, tab. -2, armchair. - , add. - one .

Keywords: RSA, AES, ECC, ECDH, ECDLP, EC, SSL

The qualification of the work is devoted to the analysis of the possibilities of an asymmetric encryption scheme by the Diff Hellman algorithm on elliptic curves and the creation of different software modules.

The method of this graduation project is the analysis of the asymmetric Diff Hellman algorithm based on elliptic curves and the development of software modules for viconnancy: key generation, encryption of information, and variety of hybrid encryption schemes.

For the achievement of the delivered mark, the following tasks were fulfilled:

- analysis of the ECC algorithm was carried out;

- shortcomings and advantages of the ECC algorithm were revealed;

- the algorithm for shearing the ECC point on the integer number and order, the cofactor of the elliptic curve has been completed;

- updated key generation scheme for asymmetric cryptography;

- the special features of key generation in the ECDH algorithm and the encryption process were verified;

- review of the methods of selection of access control lists;

- software modules for ECDH key generation, encryption and decryption were developed and implemented.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ЦП	-	центральний процесор
DES	-	Data Encryption Standard симетричний алгоритм шифрування
RSA	-	Rivest, Shamir та Adleman асиметричний криптографічний алгоритм з відкритим ключем
AES	-	симетричний алгоритм блочного шифрування
ECC	-	криптографія на еліптичних кривих
ECDH	-	протокол Діффі-Хеллмана на еліптичних кривих
ECDLP	-	завдання дискретного логарифмування групи точок еліптичної кривої.
ЕК	-	еліптична крива
SSL	-	криптографічний протокол, який забезпечує встановлення безпечного з'єднання
SHA-256	-	безпечний алгоритм хешування, версія 2

ВСТУП

В останні роки швидке розгортання таких програм, як онлайн-банкінг, торгівля акціями та корпоративний віддалений доступ, призвело до стрімкого зростання кількості конфіденційних даних, якими обмінюються через Інтернет. Більше того, ці інтернет-хости все частіше працюють від акумулятора, бездротові, портативні пристрої з жорсткими обмеженнями пам'яті, ЦП, затримки та пропускної здатності. Враховуючи ці тенденції, існує очевидна потреба в ефективних, масштабованих механізмах безпеки та протоколах, які добре працюють як у дротовому, так і в бездротовому середовищі. На сьогоднішній день криптографія з еліптичною кривою набуває широкого визнання як альтернатива звичайним криптосистемам (DES, RSA, AES тощо), які, як правило, потребують енергії. Шифри з еліптичною кривою вимагають меншої обчислювальної потужності, пам'яті та пропускної здатності зв'язку, що дає їм чітке перевагу перед традиційними криптоалгоритмами. У цьому дослідженні описується основний принцип розробки криптографії еліптичної кривої (ECC), проблема дискретного логарифма ЕК, угода з ключем ECDH та протоколи шифрування.

Метою даного дипломного проекту є аналіз асиметричного алгоритму Діффі Хеллмана на основі використання еліптичних кривих та розробка програмних модулів для виконання: генерації ключів, шифрування інформації та використання гібридної схеми шифрування.

Результати дипломного проектування були опубліковані у матеріалах V Міжнародної студентської науково - технічної конференції [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Алгоритм Elliptic Curve Cryptography - ECC

Алгоритм Elliptic Curve Cryptography - ECC швидше і надійніше аналогів, що широко використовуються.

Сьогодні сертифікати DigiCert SSL пропонують на вибір два різні алгоритми шифрування - RSA і ECC - вони допоможуть створити в організації більш надійне і масштабоване середовище.

Сертифікат ECC входить до всіх сертифікатів DigiCert Premium SSL абсолютно безкоштовно. ECC надає більш надійний захист і високу швидкодію: на сьогоднішній день цей алгоритм захищає краще, ніж інші методи шифрування, працюючи при цьому з більш короткими ключами (наприклад, 256-розрядні ключі ECC дають той самий захист, що й 3072-розрядний ключ RSA) . Більш надійний захист, здатний впоратися зі стрімким зростанням кількості підключень з мобільних пристроїв та планшетів. У міру підвищення рівнів захисту розмір ключа ECC збільшується повільніше, ніж у разі використання інших методів шифрування, що дозволяє продовжити термін експлуатації наявного обладнання та збільшити віддачу від інвестицій. Кореневі сертифікати DigiCert ECC не застаріють ще як мінімум п'ять років.

Більш високий рівень безпеки за рахунок того, що сертифікати з алгоритмом ECC з 256-бітним ключом у 10000 разів більш стійкі до злому, ніж сертифікати з алгоритмом RSA з 2048-бітним ключем, і рівень безпеки еквівалентний сертифікатам RSA з 3072-бітним.

Підвищення продуктивності серверів, особливо в періоди пікового навантаження, за рахунок можливості обробляти більшу кількість запитів при меншому завантаженні ЦП, що стає все більш важливим у міру

поширення планшетів та мобільного інтернету і, отже, підвищення вимог до продуктивності веб-інфраструктури;

Підвищення швидкості обробки запитів та зниження часу відгуку. Внутрішні тести DigiCert показали, що сервер з RSA-сертифікатом може обробляти 450 запитів в секунду із середнім часом відгуку в 150 мілісекунд, тоді як у сервера з ECC-сертифікатом при роботі в тих же умовах середній час відгуку - приблизно 75 мілісекунд.

ECC забезпечує найкращу масштабованість та дозволяє здійснювати одночасну взаємодію мільярдів хостів, суттєво підвищуючи ефективність обміну інформацією у віртуальних середовищах, використання хмарних рішень та глобальної електронної торгівлі. З точки зору користувача, покращена обчислювальна продуктивність та ефективне використання інфраструктури створюють більш сприятливі умови для роботи та збільшують продуктивність.

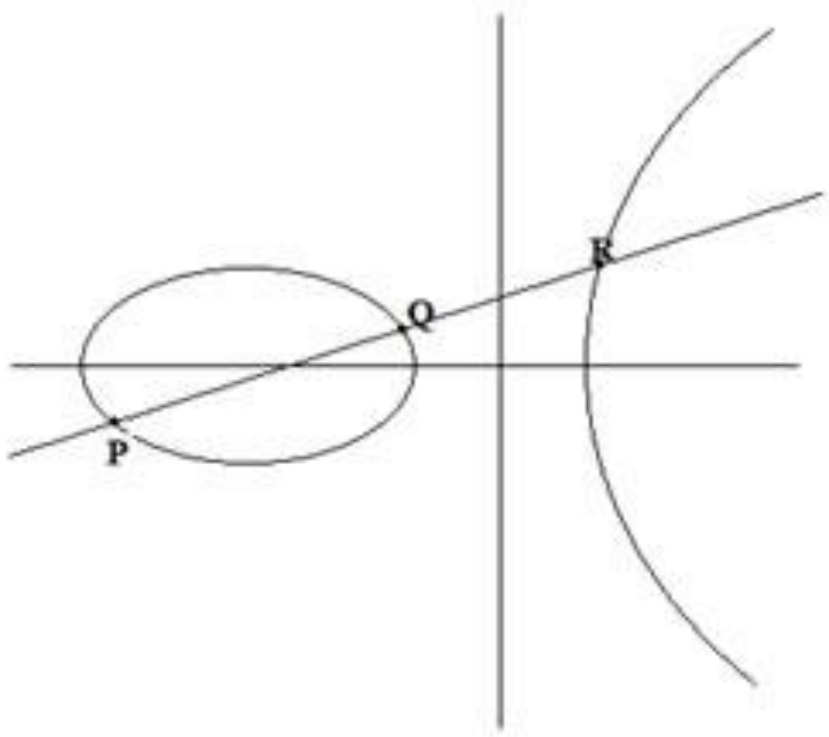

Рисунок 1.1 - Elliptic Curve #1

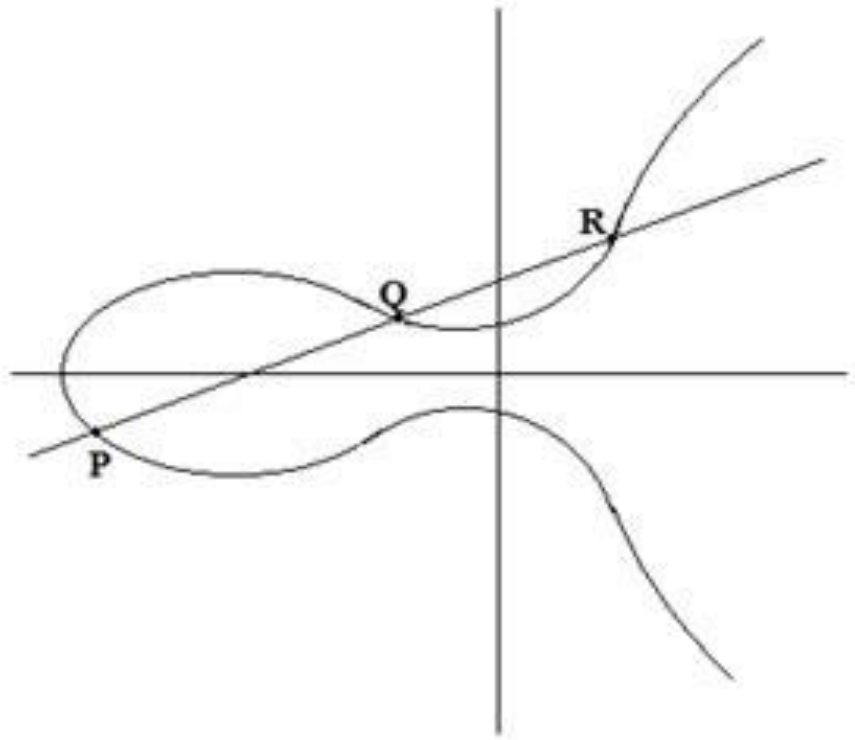

Рисунок 1.2 - Elliptic Curve #2

Еліптична крива криптографії (ECC) існує з середини 1980-х років, але вона, як і раніше, розглядається як новачок у світі SSL і лише почала отримувати визнання за останні кілька років. ECC - принципово інший математичний підхід до шифрування, ніж поважний алгоритм RSA. Еліптична крива є функцією алгебри (y2 = x3 + ax + b), яка виглядає як симетрична крива, паралельна осі x при побудові графіка.

Як і в інших формах криптографії з відкритим ключем, ECC заснований на односторонній властивості, в якій легко виконати обчислення, але не піддається зворотньому обчисленню або інвертувати результати розрахунку, щоб знайти вихідні числа. Для досягнення цієї властивості ECC використовує інші математичні операції, відмінні від RSA. Найпростіший спосіб пояснити цю математику - для еліптичної кривої лінія проходитиме тільки через три точки вздовж кривої (*P*, *Q* і *R*) і що, знаючи дві з точок (*P* і Q), інша (R) можна легко обчислити, але тільки з R, два інші, P і Q, не можуть бути отримані. ECC використовується як у цифрових підписах за

допомогою еліптичної кривої DSA (ECDSA), так і під час обміну ключами через Elliptic Curve Diffie-Hellman (ECDH).

Ці алгоритми застосовуються у різних частинах стандарту SSL. По-перше, сертифікати SSL можуть бути підписані з ECDSA замість RSA. Друге використання для ECC під час рукостискання, коли веб-сервер та клієнт узгоджують ключі сеансу, які використовуються для шифрування всіх даних, що надсилаються між сервером та браузером. У цьому останньому випадку сервер та браузер повинні бути налаштовані для підтримки наборів шифрування ECDH. Уряд США схвалив ECC, включивши його до стандарту Suite B.

Головною перевагою ECC є те, що він просто сильніший за RSA для ключових розмірів, що використовуються сьогодні. Типовий розмір ключа ECC 256 біт еквівалентний 3072-бітовому RSA-ключу і в 10 000 разів більше, ніж 2048-бітний ключ RSA. Щоб випередити обчислювальну потужність зловмисника, ключі RSA повинні бути довшими. CA / Browser Forum та провідні розробники браузерів офіційно завершили підтримку 1024-бітних ключів RSA після 2013 року, тому всі нові сертифікати SSL повинні використовувати ключі, які вдвічі більші. Крім того, майбутні розміри RSA швидко збільшуються, а довжина ключа ECC зростає лінійно.

Таблиця 1.1 – Порівняння асиметричних алгоритмів

Key Size bits	RSA та DSA Key Size	ECC Key Size
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Іншою перевагою ECC у плані безпеки є просте надання альтернативи RSA та DSA. Якщо виявлено серйозну слабкість RSA, ECC, ймовірно, стане кращою альтернативою, особливо якщо раптова слабкість RSA вимагає різкого збільшення розміру ключа компенсації.

ECC також швидше з низки причин. По-перше, менші ключі означають менше даних, які мають бути передані із сервера клієнту під час встановлення зв'язку SSL. Крім того, ECC вимагає менше обчислювальної потужності (ЦП) та пам'яті, що призводить до значного збільшення часу відгуку та пропускної спроможності на веб-серверах, коли вони використовуються.

Третьою критичною перевагою використання ECC є Perfect Forward Secrecy (PFS). Хоча PFS не є властивістю ECC, набір шифрів, що підтримується сучасними веб-серверами та браузерами, що їх реалізують PFS, також реалізує ECC. Веб-сервери, які віддають перевагу Ephemeral ECDH (ECDHE) з використанням наборів шифрів, таких як "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA", отримують переваги як від ECC, так і від PFS.

1.2 Недоліки використання алгоритму ECC

Є кілька речей, які слід враховувати при використанні алгоритму елептичних кривих. Найголовніше, що деякі браузери не підтримують сертифікати ECC. Microsoft побудувала підтримку ECC у Windows Vista, але попередні версії, включаючи Windows XP, не підтримують ECC. Mozilla додала підтримку ECC у ранній версії Firefox, а поточна версія OS X від Apple також підтримує ECC, а також поточні версії Chrome та Opera на всіх платформах.

Надійна інформація про підтримку ECC на мобільних платформах наразі недоступна. Одним із рішень цієї проблеми є використання веб-сервера для доставки різноманітних сертифікатів на основі можливостей

клієнта. Наприклад, Apache може бути налаштований для ведення переговорів з ECC із клієнтами, які його підтримують, та для переговорів RSA з рештою клієнтів

Ще одна проблема для веб-сайтів полягає в тому, що хоча тестування показало, що ECC працює швидше, перевірка підпису ECC є складним завданням, і вона може бути повільніше, ніж RSA на пристроях з більш повільними процесорами. Невідомі вразливості є ще одним ризиком для ECC. Теоретично можливі побічні/тимчасові атаки, і оскільки підтримка ECC у багатьох додатках є новішою, виявити вразливість у конкретних реалізаціях не виключено.

Кінцева проблема з ECC полягає в тому, що в цій галузі існує багато патентів, що створює певний ризик та невизначеність. Certicom Corp., дочірня компанія BlackBerry Ltd., має більше 350 патентів, які охоплюють багато аспектів ECC, таких як оптимізація продуктивності та безпеки. Тим не менш, багато хто вважає, що Certicom має тільки патенти на конкретні реалізації ECC, і у всіх випадках існують альтернативи, які не обтяжені патентами.

ECC є фундаментальним покращенням криптографії, що використовується в SSL. Він забезпечує ряд переваг, включаючи підвищену міцність та продуктивність. Понад те, він забезпечує життєздатну альтернативу алгоритмам старіння, куди покладаються багато сучасних систем. Ми рекомендуємо, щоб веб-сервери тепер налаштували перевагу наборів шифрування ECC, і хоча ми очікуємо, що RSA буде домінувати в сертифікатах SSL протягом деякого часу, ECC є альтернативою з яскравим майбутнім.

Таблиця 1.2 – Порівняльна таблиця асиметричних алгоритмів

	RSA	ECC
Алгоритм	Усі SSL сертифікати	Secure Site Pro EV Secure Site Pro Secure Site Pro with EV
Зазвичай використовується	Найбільш широко використовуваний тип ключа шифрування.	ECC набирає обертів завдяки галузі, оскільки вона ефективніша і масштабованіша, ніж інші алгоритми для невеликих пристроїв та систем з високою пропускною здатністю. Мобільні програми та інші дрібніші системи, такі як чіпи смарт-карт, першими використовували його через невеликий розмір ключа.
Питання розгортання	З новішою вимогою для 2048-бітних ключів ви можете побачити зниження продуктивності та потребувати віртуальних машин або кластерів для підтримки однакової кількості підключень.	Деякі браузери та мобільні пристрої ще не можуть бути налаштовані на отримання сертифікатів ECC.
Алгоритм підпису	SHA-256	SHA-256
Реалізація		Ви повинні завантажити та встановити проміжні сертифікати для впровадження сертифікатів ECC.

Криптографія еліптичної кривої (ECC) — це сучасне сімейство криптосистем з відкритим ключем, яке базується на алгебраїчній структурі еліптичних кривих над скінченними полями та на складності проблеми дискретного логарифма еліптичної кривої (ECDLP).

ECC реалізує всі основні можливості асиметричних криптосистем: шифрування, підписи та обмін ключами.

Криптографія ECC вважається природним сучасним наступником криптосистеми RSA, оскільки ECC використовує менші ключі та підписи, ніж RSA, для того ж рівня безпеки і забезпечує дуже швидке генерування ключів, швидке узгодження ключів і швидкі підписи.

1.3 Ключі ECC

Закритими ключами в ECC є цілі числа (в діапазоні розміру поля кривої, зазвичай 256-бітові цілі числа). Приклад 256-бітного приватного ключа ECC (шістнадцяткове кодування, 32 байти, 64 шістнадцяткові цифри): 0x51897b64e85c3f714bba707e867914295a1377a7463a9dae8ea6a6b9142.

Генерація ключа в криптографії ECC має невелику обчислювальну складність, можна порівняти з безпечним генерування випадкового цілого числа в певному діапазоні, тому це надзвичайно швидко. Будь-яке число в діапазоні є дійсним закритим ключем ECC. Відкритими ключами в ECC є точки ЕК - пари цілих координат {x, y}, що лежать на кривій. Завдяки їхнім особливим властивостям точки ЕК можуть бути стиснуті лише до однієї координати + 1 біт (непарний або парний). Таким чином, стиснутий відкритий ключ, що відповідає 256-бітному закритому ключу ECC, є 257-бітовим цілим числом. Приклад відкритого ключа ECC (відповідного вище приватного ключа, закодованого у форматі Ethereum, як шістнадцятковий з префіксом 02 або 03): 0x02f54ba86dc1ccb5bed0224d23f01ed87e4a443c4 7fc690d7797a230d7497a230. У цьому форматі відкритий ключ насправді

займає 33 байти (66 шістнадцяткових цифр), які можна оптимізувати до 257 біт.

Криптоалгоритми ECC можуть використовувати різні базові еліптичні криві. Різні криві забезпечують різний рівень безпеки (криптографічна стыйкість), різну продуктивність (швидкість) і різну довжину ключа, а також можуть включати різні алгоритми.

Криві ECC, прийняті в популярних криптографічних бібліотеках і стандартах безпеки, мають назву (з назвою кривих, наприклад, secp256k1 або Curve25519), розмір поля (який визначає довжину ключа, наприклад, 256 біт), криптостійкістю (зазвичай розмір поля / 2 або менше), продуктивність (операцій/сек) та багато інших параметрів.

Ключі ECC мають довжину, яка безпосередньо залежить від базової кривої. У більшості програм (наприклад, OpenSSL, OpenSSH і Bitcoin) довжина ключа за замовчуванням для закритих ключів ECC становить 256 біт, але залежно від кривої можливі різні розміри ключа ECC: 192-бітний (крива secp192r1), 233-бітний (крива). sect233k1), 224-біт (крива secp224k1), 256-біт (криві secp256k1 і Curve25519), 283-біт (крива sect283k1), 384-біт (криві p384 і secp384r1), 401-біт (криві secp384r1), 409 біт-409 (крива Curve41417), 448-біт (крива Curve448-Goldilocks), 511-біт (крива M-511), 521-біт (крива P-521), 571-біт (крива sect571k1) і багато інших.

Криптографія з еліптичною кривою (ECC) надає кілька груп алгоритмів на основі математики еліптичних кривих над скінченними полями:

- Алгоритми цифрового підпису ECC, такі як ECDSA (для класичних кривих) і EdDSA (для скручених кривих Едвардса).

- Алгоритми шифрування ECC та гібридні схеми шифрування, такі як інтегрована схема шифрування ECIES та EEECC (ElGamal на базі EC).

- Ключові алгоритми угоди ECC, такі як ECDH, X25519 і FHMQV.

Усі ці алгоритми використовують криву позаду (наприклад, secp256k1, curve25519 або p521) для обчислень і покладаються на складність ECDLP (задача дискретного логарифма еліптичної кривої). Усі ці алгоритми

використовують пари відкритий / закритий ключ, де приватний ключ є цілим числом, а відкритим ключем є точка на еліптичній кривій (точка EC). Давайте детальніше розглянемо еліптичні криві над скінченними полями.

У математиці еліптичні криві — це плоскі алгебраїчні криві, що складаються з усіх точок {x, y}, що описуються рівнянням:

$$Ax^3 + Bx^2y + Cxy^2 + Dy^3 + Ex^2 + Fxy + Gy^2 + Hx + ly + J = 0.$$

Криптографія використовує еліптичні криві в спрощеній формі (форма Вейєрстраса), яка визначається як:

$$y^2 = x^3 + ax + b.$$

Наприклад, крива NIST secp256k1 (використовується в Bitcoin) заснована на еліптичній кривій у вигляді: $y^2 = x^3 + 7$ (наведене вище рівняння еліптичної кривої, де a = 0 і b = 7)

На рисунку 1.3 приведена візуалізація наведеної вище еліптичної кривої.

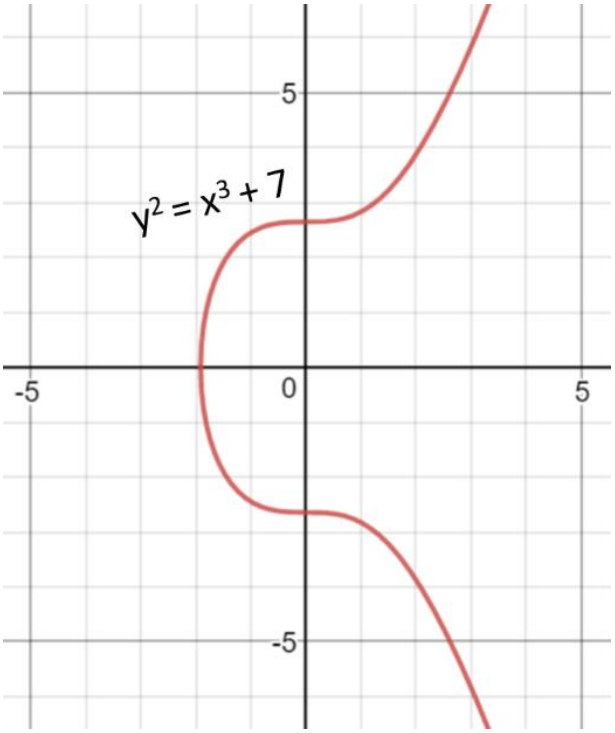

Рисунок 1.3 – Візуалізація кривої $y^2 = x^3 + ax + b$

Для візуалізації кривих зручно користуватись інструментом побудови ЕК який доступний в мережі (рисунок 1.4).

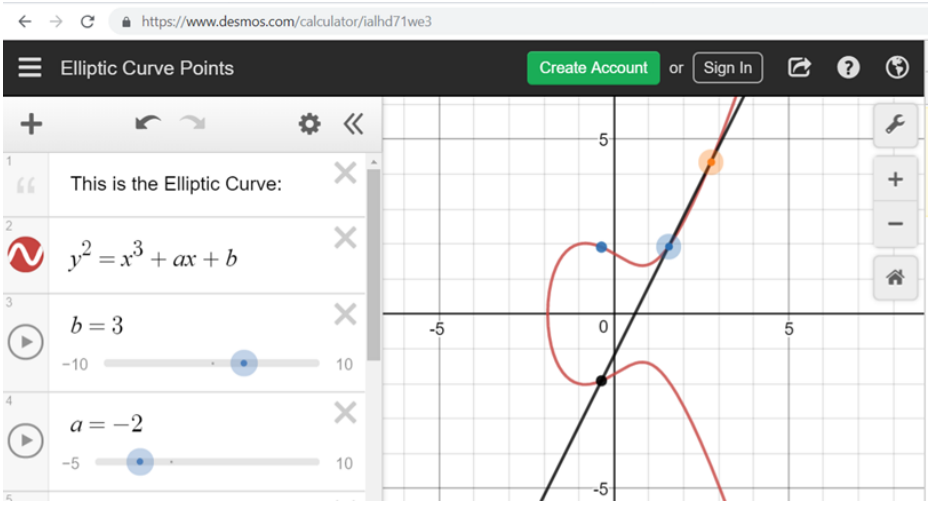

Рисунок 1.4 – Інструменти побудови еліптичних кривих

Криптографія з еліптичною кривою (ECC) використовує еліптичні криві над кінцевим полем 𝔽p (де p є простим, а p > 3) або 𝔽2m (де розмір полів p = 2^m). Це означає, що поле являє собою квадратну матрицю розміру p x p, а точки на кривій обмежені лише цілими координатами всередині поля. Усі алгебраїчні операції в полі (наприклад, додавання та множення) призводять до іншої точки в полі. Рівняння еліптичної кривої над скінченним полем 𝔽p набуває такого модульного вигляду:

$$y^2 \equiv x^3 + ax + b \pmod{p}.$$

Відповідно, «крива біткойна» secp256k1 набуває вигляду:

$$y^2 \equiv x^3 + 7 \pmod{p}.$$

На відміну від RSA, яка використовує для свого ключового простору цілі числа в діапазоні [0...p-1] (поле $\mathbb{Z}$p), ECC використовує точки {x, y} в полі Галуа 𝔽p (де x і y є цілі числа в діапазоні [0...p-1]).

Еліптична крива над скінченним полем $\mathbb{F}p$ складається з набору цілих координат {x, y}, таких що $0 \leq x, y < p$ залишаються на еліптичній кривій: $y^2 \equiv x^3 + ax + b \pmod{p}$.

Приклад еліптичної кривої над скінченним полем $\mathbb{F}17$:

$$y^2 \equiv x^3 + 7 \pmod{17}.$$

Ця еліптична крива на $\mathbb{F}17$ виглядає так як показано на рисунку 1.5.

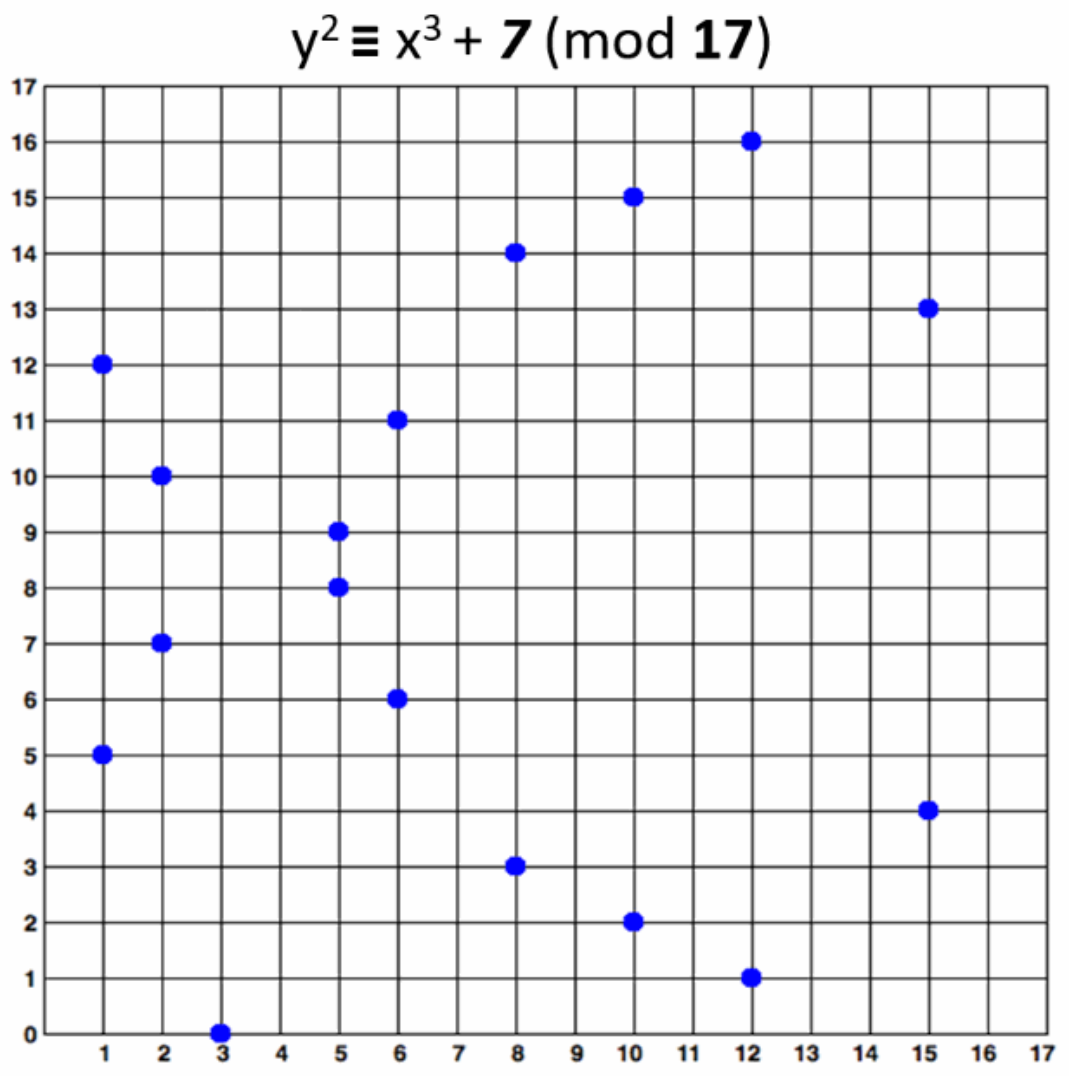

Рисунок 1.5 - Еліптична крива над скінченним полем $\mathbb{F}17$

Зауважимо, що еліптична крива над кінцевим полем y2 ≡ x3 + 7 (mod 17) складається з синіх точок як це показано на рисунку 1.5, тобто на

практиці «еліптичні криві», які використовуються в криптографії, є «наборами точок у квадратній матриці», а не класичними «кривими».

Наведена вище крива є «навчальною». Він забезпечує дуже малу довжину ключа (4-5 біт). У сучасній криптографії зазвичай використовують криві 256 біт або більше.

Еліптичні криві над скінченними полями мають математичні особливості. Досить легко обчислити, чи належить певна точка певній еліптичній кривій над скінченним полем. Наприклад, точка {x, y} належить кривій $y^2 \equiv x^3 + 7 \ (\text{mod}17)$ тоді і тільки тоді, коли:

$$x^3 + 7 - y^2 \equiv 0(\text{mod}17).$$

Вищезгадана еліптична крива і точки {5, 8} і {9, 15} зображені на рисунку 1.6.

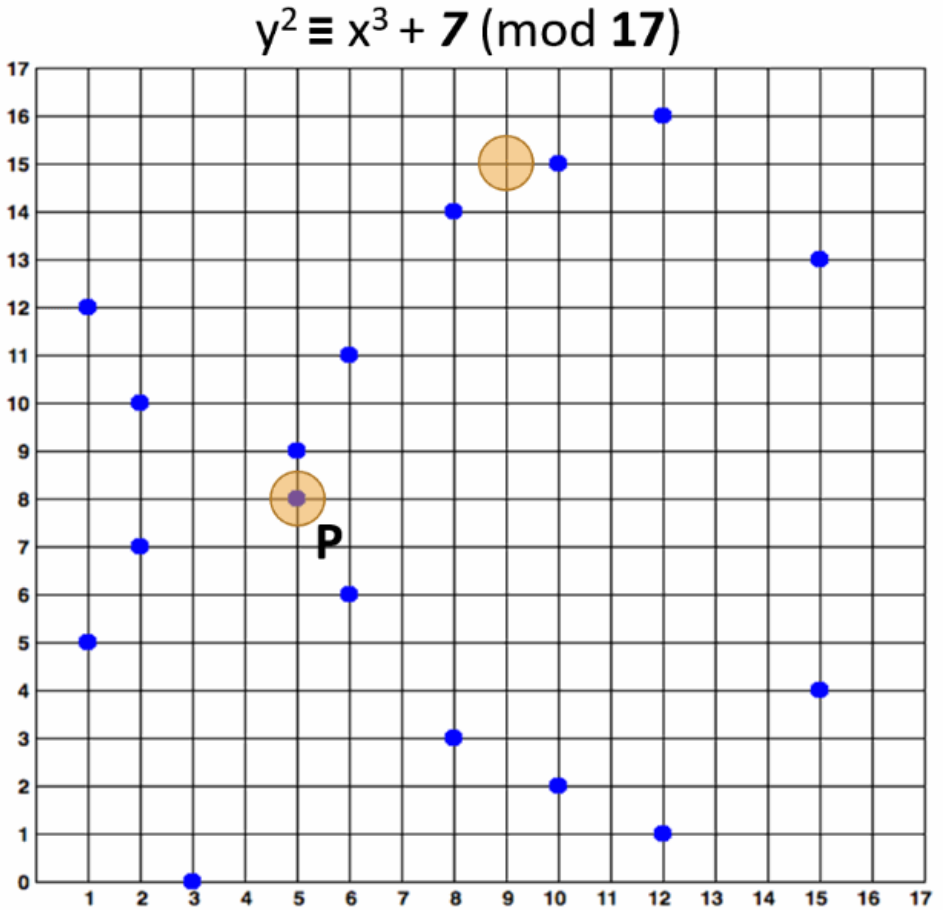

Рисунок 1.6 – Еліптична крива $y^2 \equiv x^3 + 7 \ (\text{mod}17)$ та точка {5, 8}

Точка P {5, 8} належить кривій, оскільки (5**3 + 7 - 8**2) % 17 == 0. Точка {9, 15} не належить кривій, оскільки (9 **3 + 7 - 15**2) % 17 != 0. Ці обчислення виконані у стилі Python.

2 ДОСЛІДЖЕННЯ ЕЛІПТИЧНИХ КРИВИХ

2.1 Множення точки ECC на ціле число

Дві точки над еліптичною кривою (точки EC) можна додати, і в результаті буде інша точка. Ця операція відома як додавання точки EC. Якщо ми додамо до себе точку G, то вийде G + G = 2 * G. Якщо ми знову додамо G до результату, ми отримаємо 3 * G і так далі. Так визначається множення точки EC.

Точку G над еліптичною кривою над скінченним полем (точка EC) можна помножити на ціле число k, і в результаті буде інша точка ЕК P на тій же кривій, і ця операція виконується швидко:

$$P = k * G.$$

Наведена вище операція включає деякі формули та перетворення, але для простоти ми їх пропустимо.

Формули для множення ЕК відрізняються для різних форм представлення кривої. У цьому прикладі ми будемо використовувати еліптичну криву в класичній формі Вейєрштрасса.

Для прикладу візьмемо точку ЕК G = {15, 13} на еліптичній кривій над кінцевим полем y2 ≡ x3 + 7 (mod 17) і помножимо її на k = 6. Отримаємо точку ЕК P = {5, 8 }:

$$P = k * G = 6 * \{15, 13\} = \{5, 8\}.$$

На рисунку 2.1 зображено цей приклад множення точки ЕК:

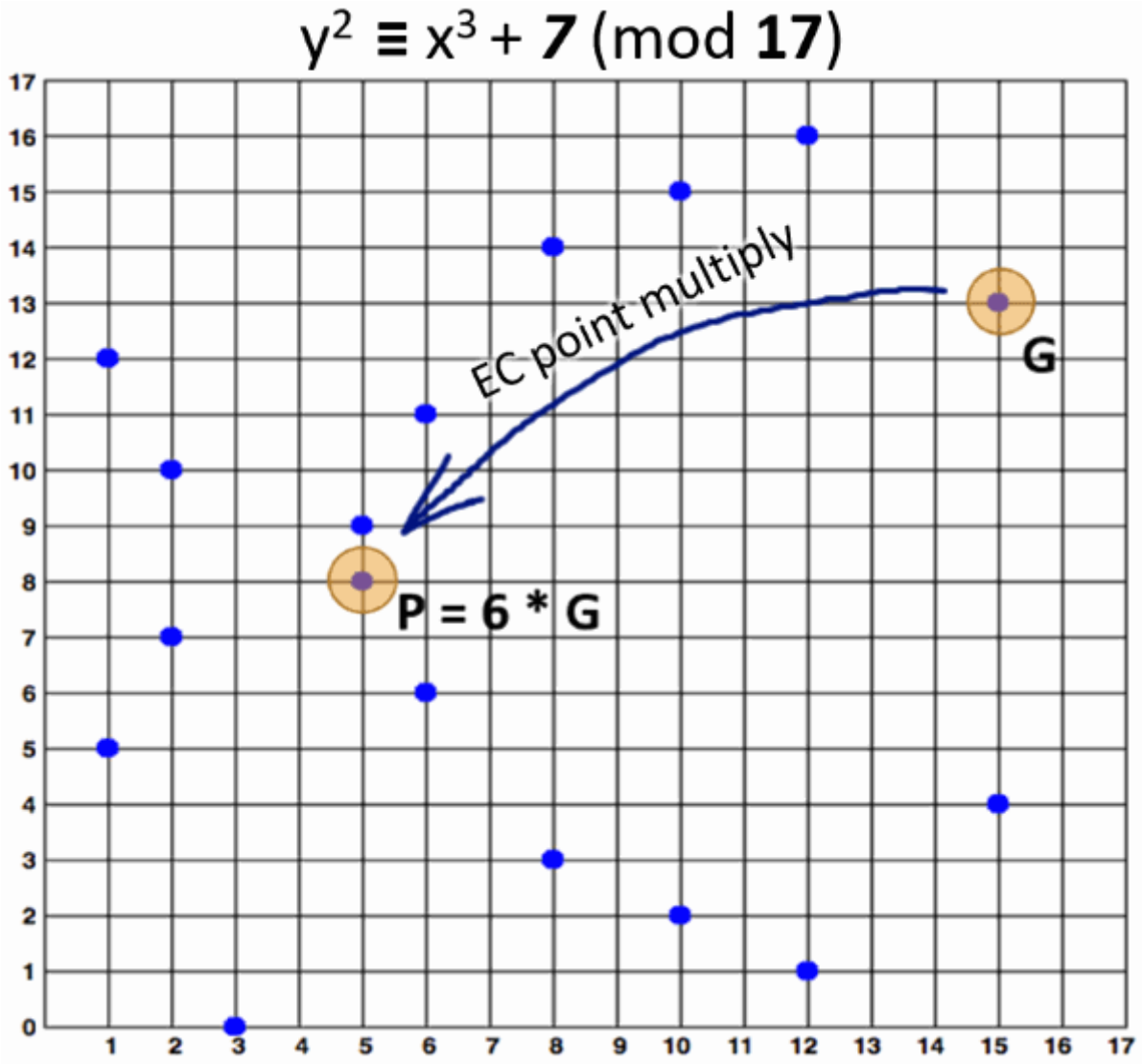

Рисунок 2.1 – Множення точки ЕК

Важливо знати, що множення точки ЕК на ціле число повертає іншу точку ЕК на тій же кривій, і ця операція виконується швидко. Множення точки ЕК на 0 повертає спеціальну точку ЕК, яка називається «нескінченність».

2.2 Порядок і кофактор еліптичної кривої

Еліптична крива над скінченним полем може утворювати скінченну циклічну алгебраїчну групу, яка складається з усіх точок кривої. У циклічній групі, якщо додати дві точки ЕК або точку ЕК помножити на ціле число,

результатом буде інша точка ЕК з тієї ж циклічної групи (і на тій же кривій). Порядок кривої — це загальна кількість усіх точок ЕК на кривій. Ця загальна кількість точок включає також спеціальну точку, яка називається «точка на нескінченності», яка виходить, коли точку помножити на 0.

Деякі криві утворюють одну циклічну групу (містить усі свої точки ЕК), тоді як інші утворюють кілька циклічних підгруп, що не перекриваються (кожна містить підмножину точок ЕК кривої). У другому сценарії точки на кривій розбиваються на h циклічних підгруп (розділів), кожна з яких має порядок r (кожна підгрупа містить однакову кількість точок). Порядок всієї групи n = h * r (кількість підгруп, помножена на кількість очок у кожній підгрупі). Кількість підгруп h, що містять точки ЕК, називається кофактором.

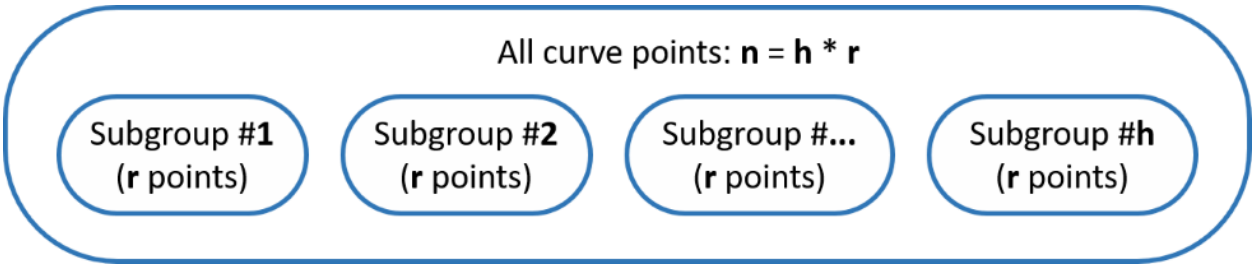

Рисунок 2.2 – Класифікація ЕК за кофактором

Кофактор зазвичай виражається такою формулою:

$$h = n / r,$$

де *n* - порядок кривої (кількість усіх її точок).

h - кофактор кривої (кількість неперекриваються підгруп точок, які разом містять усі точки кривої).

r - порядок підгруп (кількість точок у кожній підгрупі, включаючи точку нескінченності для кожної підгрупи).

Іншими словами, точки над еліптичною кривою залишаються в одній або кількох не перетинаючихся підмножинах, які називаються циклічними підгрупами. Кількість підгруп називається «кофактором». Загальна кількість

точок у всіх підгрупах називається «порядком» кривої і зазвичай позначається n. Якщо крива складається лише з однієї циклічної підгрупи, її кофактор $h = 1$. Якщо крива складається з кількох підгруп, її кофактор > 1.

Прикладом еліптичної кривої з кофактором = 1 є secp256k1. Прикладом еліптичної кривої з кофактором = 8 є Curve 25519. Прикладом еліптичної кривої, що має кофактор = 4, є Curve448.

2.3 Приватний ключ, відкритий ключ і генераторна точка в ECC

Для еліптичних кривих над скінченними полями криптосистеми ECC визначають спеціальну попередньо визначену (постійну) точку EC, яка називається генераторною точкою G (базова точка), яка може генерувати будь-яку іншу точку у своїй підгрупі над еліптичною кривою шляхом множення G на деяке ціле число. в діапазоні [0...r]. Число r називається «порядком» циклічної підгрупи (загальна кількість усіх точок у підгрупі).

Для кривих з кофактором = 1 існує лише одна підгрупа і порядок n кривої (загальна кількість різних точок на кривій, включаючи нескінченність) дорівнює числу r.

Коли G і n ретельно вибрані, а кофактор = 1, усі можливі точки ЕК на кривій (включаючи спеціальну нескінченність точки) можуть бути згенеровані з генератора G шляхом множення його на ціле число в діапазоні [1...n] . Це ціле число n відоме як «порядок кривої».

Важливо знати, що порядок r підгрупи, отриманий з певної точки G генератора ЕК (який може відрізнятися від порядку кривої), визначає загальну кількість усіх можливих закритих ключів для цієї кривої: r = n / h (порядок кривої, поділений на кофактор кривої). Криптографи ретельно вибирають параметри області еліптичної кривої (рівняння кривої, генераторну точку, кофактор тощо), щоб забезпечити, щоб простір ключів був достатньо великим для певної криптографічної міцності.

Підсумовуючи, у криптографії ECC точки ЕК разом з точкою генератора G утворюють циклічні групи (або циклічні підгрупи), що означає, що існує число r ($r > 1$), таке, що $r * G = 0 * G$ = нескінченність і всі точки в підгрупі можна отримати, помноживши G на ціле число в діапазоні [1...r]. Число r називається порядком групи (або підгрупи).

Підгрупи еліптичної кривої зазвичай мають багато точок генератора, але криптографи ретельно вибирають одну з них, яка генерує всю групу (або підгрупу) і підходить для оптимізації продуктивності в обчисленнях. Це генератор, відомий як "G".

Відомо, що для деяких кривих різні точки добутки породжують підгрупи різного порядку. Точніше, якщо порядок групи дорівнює n, для кожного простого d, що ділить n, існує точка Q така, що $d * Q$ = нескінченність. Це означає, що деякі точки, які використовуються як генератори для однієї кривої, створять менші підгрупи, ніж інші. якщо група невелика, безпека слабка. Це відомо як атаки «малих підгруп». Це причина, чому криптографи зазвичай вибирають порядок підгрупи r як просте число.

Для еліптичних кривих з кофактором $h > 1$ різні базові точки можуть генерувати різні підгрупи точок ЕК на кривій. Вибираючи певну точку генератора, ми иожемо оперувати певною підгрупою точок на кривій, і більшість операцій з точкою ЕК та криптоалгоритми ECC будуть працювати добре. Але в деяких випадках слід приділяти особливу увагу, тому рекомендується використовувати лише перевірені реалізації, алгоритми та програмні пакети ECC.

У наведеному прикладі (ЕК над скінченним полем $y^2 \equiv x^3 + 7 \bmod 17$, якщо ми візьмемо точку $G = \{15, 13\}$ як генератор, будь-яку іншу точку з кривої можна отримати, помноживши G на деяке ціле число в діапазон [1...18]. Таким чином, порядок цього ЕК дорівнює $n = 18$, а його кофактор h = 1.

Варто звернути увагу, що крива має 17 звичайних точок ЕК (показаних на наведених вище малюнках) + одну спеціальну «точку на нескінченності», всі залишаються в одній підгрупі, а порядок кривої дорівнює 18 (а не 17).

Зауважимо також, що якщо ми візьмемо точку {5, 9} як генератор, то буде створено лише 3 точки ЕС: {5, 8}, {5, 9} і нескінченність. Оскільки порядок кривої не є простим числом, різні генератори можуть генерувати підгрупи різного порядку. Це приклад того, що власні еліптичні криві можуть мати непередбачувані властивості тому для криптографічних цілей повинні використовувати перевірені криві.

У ECC, коли ми множимо фіксовану точку ЕК *G* (точку генератора) на певне ціле число *k* (*k* можна розглядати як закритий ключ), ми отримуємо точку ЕК P (його відповідний відкритий ключ).

Отже, в ECC маємо:

Еліптична крива (EC) над скінченним полем $\mathbb{F}p$.

G == генераторна точка (фіксована константа, базова точка на EC).

k == закритий ключ (ціле число).

P == відкритий ключ (точка).

Дуже швидко обчислити $P = k * G$, використовуючи добре відомі алгоритми множення ECC у часі $\log_2(k)$.

Для 256-бітових кривих знадобиться всього кілька сотень простих операцій EC.

Дуже повільною операцією (вважається нездійсненним для великих *k*) є обчислення $k = P / G$.

Ця асиметрія (швидке множення і нездійсненна повільна протилежна операція) є основою надійності безпеки криптографії ECC, також відомої як проблема ECDLP.

Проблема дискретного логарифма еліптичної кривої (ECDLP) в інформатиці визначається наступним чином.

За даною еліптичною кривою над скінченним полем $\mathbb{F}p$ і генеруючою точкою G на кривій і точкою P на кривій знайдіть ціле число k (якщо воно існує), таке, що $P = k * G$.

Для ретельно вибраних (криптографами) скінченних полів і еліптичних кривих проблема ECDLP не має ефективного рішення.

Множення точок еліптичної кривої в групі $\mathbb{F}p$ подібне до піднесення цілих чисел у групу $\mathbb{Z}p$ (це мультиплікативний запис), і проблема ECDLP схожа на проблему DLP (задача дискретного логарифма).

У криптографії ECC багато алгоритмів покладаються на обчислювальну складність задачі ECDLP щодо ретельно вибраного поля $\mathbb{F}p$ та еліптичної кривої, для яких не існує ефективного алгоритму.

Оскільки найшвидший відомий алгоритм для розв'язання ECDLP для ключа розміру k потребує n кроків, це означає, що для досягнення k-бітової міцності безпеки необхідна принаймні $2*k$-бітна крива. Таким чином, 256-бітові еліптичні криві (де розмір поля p є 256-бітним числом) зазвичай забезпечують майже 128-бітну криптостійкість.

Насправді криптостійкість дещо менша, оскільки порядок кривої (n) зазвичай менший за розмір полів (p) і оскільки крива може мати кофактор $h > 1$ (і порядок підгрупи $r = n / h$, менший за n) і тому, що кількість кроків не зовсім $\sqrt{k}$, а є $0.886*\sqrt{k}$. Точна оцінка криптостійкості для найпопулярніших стандартних еліптичних кривих наперед досліджена та наведена в джерелах[34].

Наприклад, крива secp256k1 ($p = 256$) забезпечує захист ~ 128 біт (точніше 127,8 біт), а Curve448 ($p = 448$) забезпечує ~ 224-бітну безпеку (точніше 222,8 біт).

2.4 Група точок елліптичної кривої

Елліптичні криві (ESS – криптографія з еліптичною кривою) – це не елліпси. Вони так називаються просто тому, що описуються кубічними

рівняннями, подібними кривими, які використовуються для обчислення кривого еліпса. У загальному випадку кубічні рівняння для еліптичних кривих мають вид:

$$y^2 + ax + by = x^3 + cx^2 + dx + e,$$

де *a, b, s, d,* і *e* дійсними числами, що задовільняють простим умовам. Визначення еліптичної кривої включає також деякий елемент, що визначає *O* і називається неособовим елементом (а також безкінечним елементом, або нульовим елементом). Такі рівняння називаються кубічними, або рівняннями третього порядку, оскільки в них найвищий показник ступеня рівня.

Розглянемо елліптичну криву *E* (рис. 2.3), відповідну рівнянню: $y^2 + y = x^3 - x^2$.

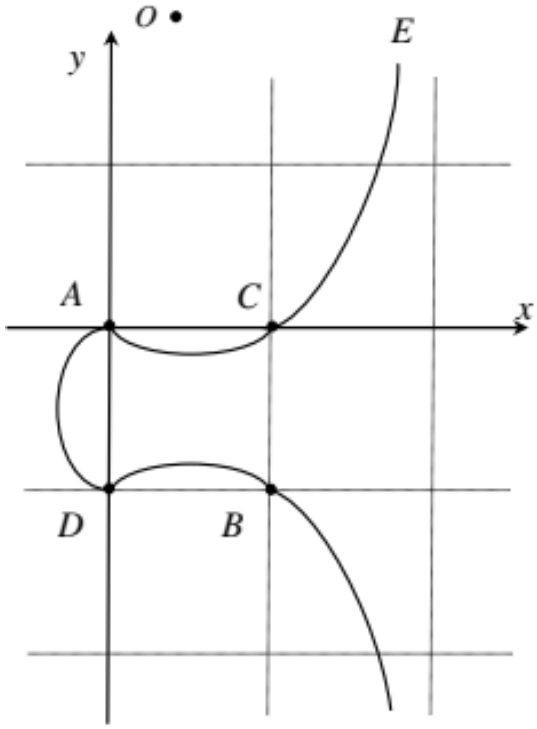

Рисунок 2.3 – Група із п'яти точок еліптичної кривої *E*;
O – безкінечно віддалена точка

На цій кривій лежать тільки чотири точки, координати, які є цілими числами. Це точки *A*(0, 0), *B*(1, –1), *C*(1, 0), *D*(0, –1).

Для визначення операцій складання в групу точок еліптичної кривої будем рахувати, що: на площині існує безкінечно віддалена точка $O \in E$, в якій сходяться всі вертикальні прямі;

Дотична до кривої пересікає точку дотику *P* два раза.

Тепер можна сформулювати правила додавання точок $P, Q \in E$ (рисунок 2.4).

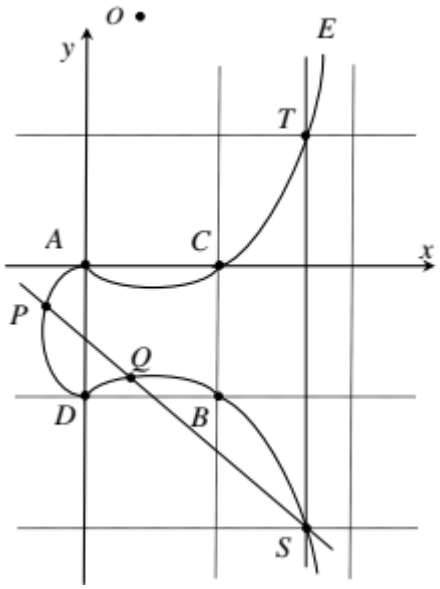

Рисунок 2.4 – Додавання точок на еліптичній кривій

Для додавання точок на еліптичній кривій $P + Q = T$ проведемо пряму лінію через точки *P* та *Q*:

- знайдемо третю точку S перетину цієї прямої з кривою *E*;

- проведемо через точку S вертикальну пряму до перетину з кривою *E* у точці *T*;

- потрібна сума $P + Q = T$.

Застосувавши ці правила до групи точок $G = \{A, B, C, D, O\}$ отримаємо (рисунок 2.5):

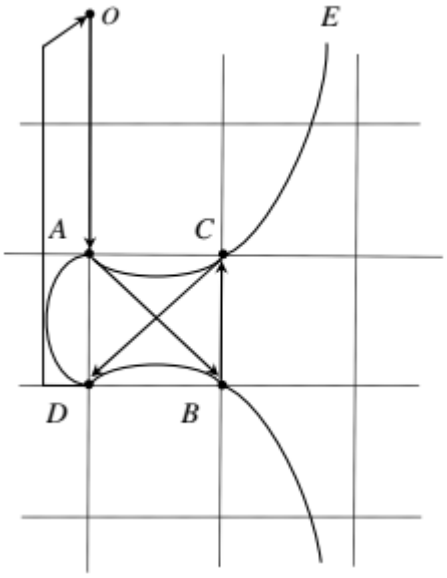

Рисунок 2.5 – Адаптивна абелева група {*A*, *B*, *C*, *D*, *O*} на еліптичній кривій *E*

$$A + A = B;\ A + B = C;\ A + C = D;\ A + D = O,\ \text{або}\ 2A = B;\ 3A = C;\ 4A = D;$$
$$5A = O;\ 6A = A.$$

Для будь-яких точок $P, Q \in G$ справедливе відношення $P + Q = Q + P$. Для будь-якої точки $P \in G$ справедливо $P + O = P$, інакше кажучи, точка O – адитивний одиничний елемент групи G.

В реальних криптосистемах використовується рівняння $y^2 \equiv x^3 + ax + b (\mathrm{mod}\, p)$, де $a, b \in GF(p)$, $4a^3 + 27b^2 (\mathrm{mod}\, p) \neq 0,\ p > 3$- просте. Група $E(GF(p))$ складається з точок $(x, y); x, y \in GF(p)$ що задовольняють рівнянню, і нескінченно віддаленої точки О.

Множина $E_p(a,b)$ складається з всіх точок $(x, y), x \geq 0, p > y$, що задовільняють рівнянню $y^2 \equiv x^3 + ax + b (\mathrm{mod}\, p)$ і точок безкінечності O. Кількість точок у $E_p(a,b)$ будемо позначати $\#E_p(a,b)$.

Ця величина має велике значення для криптографічних програм еліптичних кривих. Визначена над точками з $E(GF(p))$ операція додавання алгебраїчно може бути описана в такий спосіб:

1 $P+O=O+P=P$.

2 Якщо P = (x, y), то P + (x, -y) = O. Точка (x, -y) є негативним значенням точки P та позначається $-P$. Зауважимо, що (x, -y) лежить на еліптичній кривій і належить $E_p(a,b)$. Наприклад, в випадку $E_{23}(1,1)$ для $P=(13,7)$ отримаємо $-P=(13,-7)$. Але $-7 \bmod 23 \equiv 16$, таким чином, $-P=(13,16)$.

Якщо $P=(x_1,y_1)$ і $Q(x_2,y_2)$, то $P+Q=(x_3,y_3)$ визначається згідно правил:

$$x_3 \equiv \lambda^2 - x_1 - x_2 (\bmod p);$$

$$x_3 \equiv \lambda(x_1 - x_3) - y_1 (\bmod p),$$

де:

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, \textit{якщо } P \neq Q; \\ \dfrac{3x_1^2 + a}{2y_1}, \textit{якщо } P = Q. \end{cases}$$

Число λ - кутовий коефіцієнт січної, проведений через точки $P=(x_1,y_1)$ і $Q(x_2,y_2)$. При $P=Q$ січна перетворюється в дотичну чим і пояснюється наявність двох формул для обчислення λ.

Для прикладу розглянкмо криву:

$$E_7(2,6): y^2 \equiv x^3 + 2x + 6 (\bmod 7).$$

Провіряємо умову:

$$(4 \cdot 2^3 + 27 \cdot 6^2) \bmod 7 \equiv 3 \neq 0.$$

Отже, ця крива несингулярна. Знайдемо якусь (випадкову) точку в $E_7(2,6)$. Нехай $x=5$, тоді:

$$y^2 \equiv 5^3 + 2 \cdot 5 + 6 (\bmod 7) \equiv (125+10+6) \bmod 7 \equiv 1 \bmod 7,$$

$y \equiv 1 (\mathrm{mod}\, 7)$ або $y \equiv -1 \equiv 6 (\mathrm{mod}\, 7)$. Знайдені відразу дві точки: (5,1) і (5,6). Знайдемо ще декілька точок шляхом обчислення композиції. Спочатку обчислимо 2(5,1).

$$\lambda = \frac{3 \cdot 5^2 + 2}{2 \cdot 1} = \frac{0}{2} \equiv 0 (\mathrm{mod}\, 7);$$

$$x_3 = 0 - 2 \cdot 5 \equiv 4 (\mathrm{mod}\, 7);$$

$$y_3 = 0(5 - 4) - 1 \equiv 6 (\mathrm{mod}\, 7).$$

Отримали 2(5, 1) = (4, 6) (можна переконатися, що отримана точка лежить на кривою, підставивши її координати рівняння). Знайдемо ще одну точку 3(5, 1) = (5, 1) + (4, 6).

$$\lambda = \frac{6 - 1}{4 - 5} = -\frac{5}{1} \equiv 2 (\mathrm{mod}\, 7);$$

$$x_3 = 2^2 - 5 - 4 \equiv 2 (\mathrm{mod}\, 7);$$

$$y_3 = 2(5 - 2) - 1 \equiv 5 (\mathrm{mod}\, 7).$$

Отримали 3(5,1)=(2,5).

Отже, знайдено чотири точки. Для криптографічного використання кривої важливо знати, скільки всього точок у множині E7(2, 6).

Нехай p = 23. Розглянемо еліптичну криву E: $y^2 = x^3 + x + 1$. E_{23} (1, 1) складається з точки, а також з наступних точок: (0, 1); (0, 22); (1, 7); (1, 16); (3, 10); (3, 13); (4, 0); (5, 4); (5, 19); (6, 4); (6, 19); (7, 11); (7, 12); (9, 7); (9, 16); (11, 3); (11, 20); (12, 4); (12, 19); (13, 7); (13, 16); (17, 3); (17, 20); (18, 3); (18, 20); (19, 5); (19, 18).

Нехай P = (3, 10) та Q = (9, 7). Знайдемо P+Q та 2P. Нехай $P + Q = (x_3, y_3)$, тоді:

$$\lambda = \frac{7 - 10}{9 - 3} = -\frac{1}{2} \equiv 11 (\mathrm{mod}\, 23);$$

$$x_3 = 121 - 3 - 9 = 109 = -6 \equiv 17 (\mathrm{mod}\, 23);$$

$$y_3 = 11(3 + 6) - 10 = 89 \equiv 20 (\mathrm{mod}\, 23).$$

Отже, P + Q = (17, 20). Знайдемо $2P = P + P = (x_3, y_3)$, тоді:

$$\lambda = \frac{3 \cdot 9 + 1}{20} = \frac{1}{4} \equiv 6 (\mathrm{mod}\, 23);$$

$$x_3 = 36 - 6 = 30 \equiv 7 (\mathrm{mod}\, 23);$$

$$y_3 = 6(3 - 7) - 10 = -34 \equiv 12 (\mathrm{mod}\, 23).$$

Таким чином, $2P = (7,12)$.

Нехай p = 5, a = b = 1, $4a^3 + 27b^2 = 4 + 4 \equiv 8 (\mathrm{mod}\, 5) \neq 0$.

Розглянемо еліптичну криву $y^2 = x^3 + ax + b$. $E_5(1, 1)$ складається з точок що на рисунку 2.6.

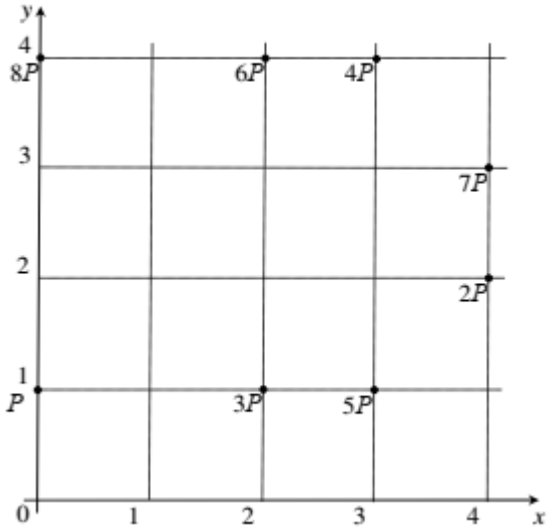

Рисунок 2.6 – Аддитивна группа $E(Z_5)$

P = (0, 1), $2P$ = (4, 2), $3P$ = (2, 1), $4P$ = (3, 4), $5P$ = (3, 1), $6P$ = (2, 4), $7P$ = (4, 3), $8P$ = (0, 4), $9P = O$, при цьому $10P = P$ (рис. 2.6).

Розглянемо основні рекомендації щодо вибору параметрів еліптичної кривої, призначеної для вирішення криптографічних завдань, а саме задач на вибір коефіцієнтів a, b і модуля p. Фактично критерієм вибору є неможливість здійснення певного роду атак, запропонованих для деяких класів кривих. Нижче наведені рекомендації слідують стратегії вибору

випадкової кривої Ця стратегія вважається найбільш надійною з метою забезпечення стійкості результуючої криптосистеми.

Альтернативний підхід, що не розглядається тут, полягає в систематичному конструювання кривої із заданими властивостями, що зазвичай виявляється більш ефективним з обчислювальної точки зору. Для реалізації цього підходу запропоновані спеціальні методи, але одержувані криві фактично вибираються з відносно невеликого класу та викликають підозри на наявність деяких специфічних властивостей, які можуть дозволити з часом винайти алгоритми їхнього злому.

Опишемо кроки процес формування випадкової кривої:

1. Вибираємо випадково просте число *p*. Бітова довжина числа p $t = \lfloor \log p \rfloor + 1$ повинна бути такою, щоб унеможливити застосування загальних методів знаходження логарифмів на кривій, що мають трудомісткість $T(2^{t/2})$.

Величина *t* = 128 біт (чотири машинні слова на 32-бітових комп'ютерах) сьогодні недостатні, так як є повідомлення про взлом відповідних кривих. Інше міркування полягає в тому, що шифр на еліптичній кривій повинен бути не менш стійким, ніж блоковий шифр AES (Advanced Encryption Standard). Вважається, що стійкість AES забезпечується повною довжиною його ключа, що становить 128, 196 або 256 біт. Так як стійкість шифру на еліптичній кривій визначається величиною *t*/2, довжина модулів еліптичних кривих повинна становити відповідно 256, 392 та 512 біт.

2. Вибираємо випадкові числа *a*, і *b* такі, що $a, b \pmod p \neq 0$ і $4a^3 + 27b^2 \pmod p \neq 0$. Звернемо увагу на те, що при обчисленні композиції точок параметр *b* ніде не фігурує. Тому для підвищення ефективності рахунки іноді рекомендують випадково вибирати тільки *b*, а приймати рівним невеликого цілого числа. Так, стандарт США FIPS 186-2 передбачає використання кривих з параметром *a* = -3, що дещо спрощує обчислення. Величина *t* = 128 біт (чотири машинні слова на 32-бітових комп'ютерах сьогодні недостатня, так як є повідомлення про злом відповідних кривих. Інше міркування полягає в тому, що шифр на еліптичній кривій повинен бути

не менш стійким, ніж блоковий шифр AES (Advanced Encryption Standard). Вважається, що стійкість AES забезпечується повною довжиною його ключа, що становить 128, 196 або 256 біт. Так як стійкість шифру на еліптичній кривій визначається величиною $t/2$, довжина модулів еліптичних кривих повинна становити відповідно 256, 392 та 512 біт.

3. Визначаємо число точок на кривій $n = \#E_p(a,b)$ (це найбільш трудомісткий етап описуваного процесу). Важливо, щоб n мав великий простий дільник q, а найкраще саме було простим числом, $n = q$. Якщо п розкладається на малі множники, то в E_p(a, b) існує багато малих підмножин зі своїми генераторами і алгоритм Поліга-Хеллмана [25] швидко обчислює логарифм на кривій через логарифми у цих малих підмножинах. Якщо пошук кривою з $n = q$ займає дуже багато часу, можна допустити $n = h\ q$, де h – невелика величина. Ще раз слід підкреслити, що стійкість криптосистеми на еліптичній кривій визначається не модулем p, а числом елементів q у безлічі точок кривої. Але якщо множник h - невелика кількість, то q є величиною того ж порядку, що й p. Якщо n не відповідає вимогам що висуваються, необхідно повернутися до кроку 2.

4. Перевіряємо, чи виконуються нерівності $(p^k - 1) \bmod q \neq 0$ для всіх k, $0 < k < 32$. Якщо ні, то повертаємось до кроку 2. Ця перевірка запобігає можливість MOV-атаки (названої на прізвища її авторів Menezes, Okamoto, Vanstone), і навіть виключає з розгляду звані суперсингулярні криві і криві з $\#E_p(a,b) = p - 1$ [26, 27]. Метод MOV та зазначені особливі типи кривих дозволяють звести завдання обчислення логарифму над кривою до більш простих завдань.

5. Перевіряємо, чи нерівність $q \neq p$. Якщо ні, то повертаємось до кроку 2. Справа в тому, що для кривих з $q = p$, які називаються аномальними, є ефективні методи обчислення логарифмів [26, 27].

6. На цьому кроці крива отримана для криптографічних додатків. Ми маємо параметри p, a, b, кількість точок n та розмір підмножини точок q. Зазвичай ще потрібно знайти точку G – генератор цієї підмножини. Якщо $q =$

n то будь-яка точка (крім *O*) є генератором. Якщо *q* < *n*, то вибираємо випадкові точки G ', доки отримаємо $G = [n/q]G' \neq O$.

Щоб отримати випадкову точку на кривій, беремо випадкове число $x < p$, обчислюємо $\mathrm{e} \equiv (\mathrm{x}^3 + \mathrm{ax} + \mathrm{b}) \bmod \mathrm{p}$ і намагаємось витягти квадратний корінь $y = \sqrt{e} \bmod p$. Якщо корінь існує, то отримуємо точку (x, y), інакше у разі пробуємо інше число x. Алгоритми обчислення квадратних коренів за модулем простого числа можна знайти у [25].

Завдання, яке вирішує криптоаналітик при використанні криптосистеми на базі еліптичних рівнянь, називається завданням дискретного логарифмування на еліптичній кривій і формулюється таким чином. Дано точки *P* і *Q* на еліптичній кривій порядку п, де п – число точок на кривій. Необхідно знайти єдину точку x таку, що *P* = *xQ*.

Розглянемо використання еліптичних кривих у криптографії.

Обмін ключами з використанням еліптичних кривих може бути виконаний в такий спосіб. Спочатку вибирається просте число p параметри a і b для еліптичної кривої. Це задає еліптичну групу точок $E_p(a,b)$. Потім $E_p(a,b)$ вибирається генеруюча точка $G(x,y)$. При виборі *G* важливо, щоб найменше значення *n*, за якого *nG* = *O*, виявилося дуже великим простим числом. Параметри $E_p(a,b)$ та *G* криптосистеми є параметрами, відомими всім учасникам. Обмін ключами між учасниками інформаційного процесу *A* та *B* можна провести за наступною схемою.

1 Сторона *A* вибирає ціле число k_a, менше за *n*. Це число буде особистим ключем учасника *A*. Потім сторона *A* генерує відкритий ключ $Y_a = k_b G$. Відкритий ключ є деякою точкою з $E_p(a,b)$.

2 Точно так само, як і в п. 1, сторона вибирає особистий ключ k_b і обчислює відкритий ключ $Y_a = k_b G$.

3 Учасник А генерує секретний ключ $K = k_a Y_b$, а учасник *B* генерує секретний ключ $K = k_b Y_a$.

Дві формули в п. 3 дають той самий результат, оскільки:

$$k_a Y_b = k_a(k_b G) = k_b(k_a G) = k_b Y_a.$$

Нехай p = 211; G = (2, 2); $E_p(0,-4)$, що відповідає кривій $y^2 = x^3 - 4$. Можна підрахувати, що $241G = O$. Особистим ключем учасника

А є $k_a = 121$, тож відкритим ключем сторони А буде:

$$Y_a = 121(2,2) = (115,48).$$

Особистим ключем учасника B є $k_b = 203$, тому відкритим ключем сторони В буде:

$$Y_b = 203(2,2) = (130,203).$$

Загальним секретним ключем є $K = 121(130,203) = 203(115,48) = (161,69)$.

Слід звернути увагу, що загальний секретний ключ є пару чисел. Якщо цей ключ передбачається використовувати як сеансового ключа для традиційного шифрування, то з цієї пари чисел необхідно генерувати одне потрібне значення. Можна, наприклад, просто використовувати координату x або деяку просту функцію від x.

У літературі можна знайти аналіз кількох підходів до зашифровування/розшифровування, що передбачають використання еліптичних кривих. Розглянемо найпростіший із цих підходів. Першим завданням у системі є зашифровування відкритого тексту повідомлення M, яке пересилатиметься у вигляді значення x – y для точки P_M. Тут точка P_M представлятиме зашифрований текст і згодом розшифровуватиметься.

Сторона А вибирає особистий ключ k_a та генерує відкритий ключ Y_a.

Щоб зашифрувати та надіслати повідомлення P_M користувачу B, користувач А вибирає випадкове позитивне ціле число r та обчислює зашифрований текст C_M, що складається з пари точок $C_M = (rG P_M + rY_b)$.

Слід зазначити, що сторона А використовує відкритий ключ Y_b сторони *B*. Щоб розшифрувати цей шифртекст, сторона *B* множить першу точку в парі на секретний ключ і віднімає результат з другої точки:

$$P_M + rY_b - k_b(rG) = P_M + r(k_bG) - k_b(rG) = P_M.$$

Користувач *A* замаскував повідомлення P_M за допомогою додавання до ньому rY_b. Ніхто, крім цього користувача, не знає значення *r*, тому, хоча Y_b є відкритим ключем, ніхто не зможе прибрати маску rY_b. Проте користувач *A* включає повідомлення і "підказку", якої буде достатньо, щоб прибрати маску тому, хто має особистий ключ k_b. Криптоаналітика для відновлення повідомлення доведеться обчислити *r* за даними *G* та *rG*, що є важким завданням.

Розглянемо випадок: $p = 751$; $G = (0,376)$ G = (0, 376); $E_p(-1,88)$, що відповідає кривій $y^2 = x^3 - x + 188$.

Припустимо, що сторона *A* збирається надіслати стороні повідомлення, яке кодується еліптичною точкою $P_M = (562,201)$. Для цього учасник *A* вибирає випадкове число *r* = 386 та знаходить відкритий ключ учасника $B - Y_b = (201,5)$. Обчислює 386 (0, 376) = (676, 558) та (562, 201) + 386 (201, 5) = (385, 328). Таким чином, учасник А повинен надіслати повідомлення {(676, 558),(385, 328)}.

Безпека, що забезпечується криптографічним підходом на основі еліптичних кривих, залежить від того, наскільки важким для вирішення виявляється завдання визначення *r* за даними *rP* і *P*. Це завдання зазвичай називають проблемою логарифмування на еліптичній кривій. Найбільш швидким з відомих на сьогодні методів логарифмування на еліптичній кривій є ρ-метод Полларда (Pollard) [28].

3 ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ ШИФРУВАННЯ

3.1 Реалізація множення в ЕК

Тепер, після всіх понять, давайте напишемо деякий код. Ми будемо використовувати бібліотеку Python tinyec, яка надає примітиви ECC, такі як циклічні групи (клас SubGroup), еліптичні криві над скінченними полями (клас Curve) і точки ЕК (клас Point). Спочатку встановимо пакет tinyec

Проведемо дослідження з навчальною кривою з попередніх прикладів $y^2 \equiv x^3 + 7 \pmod{17}$, з точкою генерації $G = \{15, 13\}$, яка має порядок $n = 18$. Назвемо її $p1707$ (рисунок 3.1).

```
1 from tinyec.ec import SubGroup, Curve
2
3 field = SubGroup(p=17, g=(15, 13), n=18, h=1)
4 curve = Curve(a=0, b=7, field=field, name='p1707')
5 print('curve:', curve)
6
7 for k in range(0, 25):
8     p = k * curve.g
9     print(f"{k} * G = ({p.x}, {p.y})")
```

Рисунок 3.1 – Фрагмент коду для виводу точок навчальної ЕК

Наведений вище код демонструє множення ЕС. Він множить точку генератора G на 0, 1, 2, ..., 24. Результати виконання коду представлені на рисунку 3.2. Проведено підключення модуля tinyec та встановлено параметри еліптичної кривої, а саме вибрано параметри підгрупи та початкові параметри кривої.

```
curve: "p1707" => y^2 = x^3 + 0x + 7 (mod 17)
0 * G = (None, None)
1 * G = (15, 13)
2 * G = (2, 10)
3 * G = (8, 3)
4 * G = (12, 1)
5 * G = (6, 6)
6 * G = (5, 8)
7 * G = (10, 15)
8 * G = (1, 12)
9 * G = (3, 0)
10 * G = (1, 5)
11 * G = (10, 2)
12 * G = (5, 9)
13 * G = (6, 11)
14 * G = (12, 16)
15 * G = (8, 14)
16 * G = (2, 7)
17 * G = (15, 4)
18 * G = (None, None)
19 * G = (15, 13)
20 * G = (2, 10)
23 * G = (8, 3)
24 * G = (12, 1)
25 * G = (6, 6)
26 * G = (5, 8)
```

Рисунок 3.2 – Результати виконання коду для виводу точок навчальної ЕК

Видно, що 0 * G = нескінченність. Також добре видно, що група ЕК є циклічною і порядок групи ЕК дорівнює n = 18, тому що починаючи з k = 18 наступні точки повторюють перші:

18 * G = 0 * G = нескінченність

19 * G = 1 * G = {15, 13}

20 * G = 2 * G = {2, 10}

21 * G = 3 * G = {8, 3}

Точки ЕС, утворені множенням точки генератора G на 2, 3, 4, ..., 17, показані на рисунку 3.3.

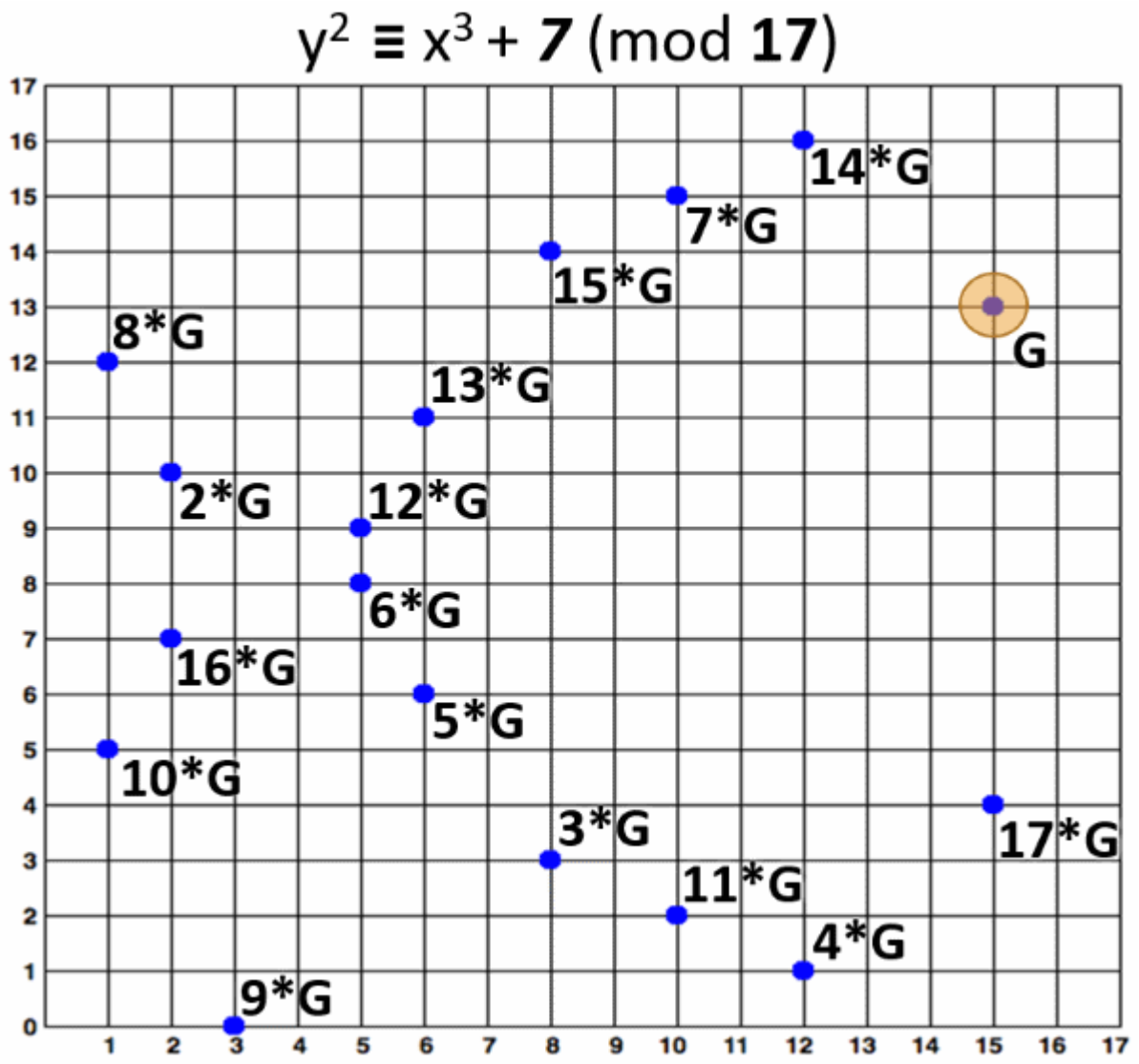

Рисунок 3.3 – Результати виконання коду для виводу точок навчальної ЕК

Модифікуємо наведений вище приклад і змінимо точку генератора на G' = {5, 9}. Це істотно змінить вихід, як це показано на рисунку 3.4.

```
1 from tinyec.ec import SubGroup, Curve
2
3 field = SubGroup(p=17, g=(5, 9), n=18, h=1)
4 curve = Curve(a=0, b=7, field=field, name='p1707')
5 print('curve:', curve)
6
7 for k in range(0, 25):
8     p = k * curve.g
9     print(f"{k} * G' = ({p.x}, {p.y})")
```

Рисунок 3.4 – Фрагмент коду, зі зміненою точкою генератора

Вихідні дані показують, що порядок підгрупи нової генераторної точки не 18, а 3. Це можливо, оскільки 18 не є простим. З результату видно, що 3 * G' = нескінченність і отриманий порядок підгрупи дорівнює 3 (рисунок 3.5).

```
curve: "p1707" => y^2 = x^3 + 0x + 7 (mod 17)
0 * G' = (None, None)
1 * G' = (5, 9)
2 * G' = (5, 8)
3 * G' = (None, None)
4 * G' = (5, 9)
5 * G' = (5, 8)
6 * G' = (None, None)
...
```

Рисунок 3.5 – Результат виконання коду зі зміненою точкою генератором

Наведений вище приклад ще раз підтверджує, що розробкою еліптичної кривої для криптографії повинні займатися криптографи, а не розробники. Розробники повинні покладатися на добре встановлені крипто-стандарти та перевірені криптобібліотеки.

Тепер напишемо реальний приклад. Замість використання навчальної кривої p1707 (4-5-бітова крива, p = 17), будемо використовувати 192-бітну криптографічну криву secp192r1 (192-розрядна, p = 6277101735386680763835789423206706910206393). Наведений нижче приклад схожий на попередній (рисунок 3.6).

```python
from tinyec import registry

curve = registry.get_curve('secp192r1')
print('curve:', curve)

for k in range(0, 10):
    p = k * curve.g
    print(f"{k} * G = ({p.x}, {p.y})")

print("Cofactor =", curve.field.h)

print('Cyclic group order =', curve.field.n)

nG = curve.field.n * curve.g
print(f"n * G = ({nG.x}, {nG.y})")
```

Рисунок 3.6 – Фрагмент коду для кривої secp192r1

Результати приведені на рисунку 3.7 схожі до результатів попереднього прикладу.

```
curve: "secp192r1" => y^2 = x^3 + 6277101735386680763835789423207666416083908700390324961
0 * G = (None, None)
1 * G = (602046282375688656758213480587526111916698976636884684818, 174050332293622031480
2 * G = (5369744403678710563432458361254544170966096384586764429448, 5429234379789071039
3 * G = (2915109630280678890720206779706963455590627465886103135194, 2946626711558792003
4 * G = (1305994880430903997305943738697779408316929565234787837114, 3981863977451150342
5 * G = (4102832511167848740189935621365668701106767069367626602 40, 1206654674899825246
6 * G = (4008504146453526025173196900303594155799995627910231899946, 3263759301305176906
7 * G = (3473339081378406123852871299395262476289672479707038350589, 2152713176906603604
8 * G = (1167950611014894512313033362696697441497340081390841490910, 4002177906111215127
9 * G = (3176317450453705650283775811228493626776489433309636475023, 4460189377466938476
Cofactor = 1
Cyclic group order = 6277101735386680763835789423176059013767194773182842284081
n * G = (None, None)
```

Рисунок 3.7 – Результат виконання коду для кривої secp192r1

Крива secp192r1 використовує циклічну групу дуже великого порядку n = 6277101735386680763835789423176059013767194773182842284081 (просте число) з кофактором *h*, як у прикладі, ми можемо очікувати, що кофактор *h* = 1.

Тепер згенеруємо випадковий закритий ключ privKey (ціле число в діапазоні [0...*n*-1]) і відповідний відкритий ключ pubKey = privKey * *G* (рисунок 3.8).

```python
from tinyec import registry
import secrets

curve = registry.get_curve('secp192r1')

privKey = secrets.randbelow(curve.field.n)
pubKey = privKey * curve.g
print("private key:", privKey)
print("public key:", pubKey)
```

Рисунок 3.8 – Генерація випадкового закритого ключа

Результат виконання приведеного коду приведений на рисунку 3.9.

```
1 private key: 42256553189779620312642301302421807488186031474676158689O2
2 public key: (539603083445677019039677653093837488227383617948783415229l, 342216058816691
```

Рисунок 3.9 – Згенеровані приватний та публічний ключі

Будемо використовувати такі пари ключів ECC {приватний ключ, відкритий ключ} для шифрування даних, підписання повідомлень і перевірки підписів.

В реальних проектах 192-бітові криві вважаються слабкими, тому рекомендутся викрсанн 256-бітові криві (або більше біт), де ключі також мають 256-бітові (або відповідно більше). У наведеному вище прикладі ми використовуємо 192-бітну криву, щоб зробити вибірку меншою.

3.2 Стиснення відкритих ключів

Еліптичні криві над скінченними полями $\mathbb{F}p$ (у формі Вейєрштрасса) мають щонайбільше 2 точки на координату y (непарні *x* і парні *x*). Ця властивість походить від природи рівняння еліптичної кривої та проілюстрована на наступному графіку (рисунок 3.10).

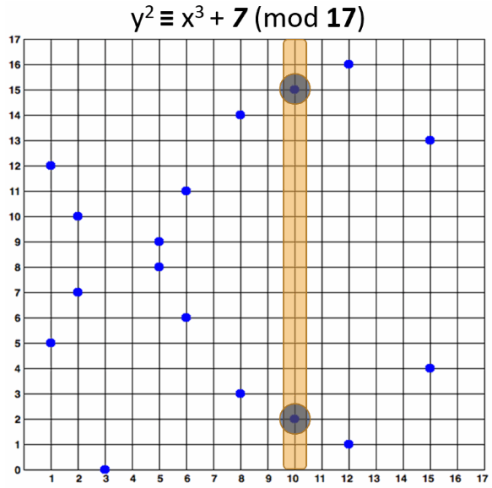

Рисунок 3.10 – Еліптичні криві у формі Вейєрштрасса

Завдяки цій властивості точку еліптичної кривої (і відповідно відкритий ключ ECC) P {x, y} можна стиснути як C {x, непарний/парний). Це означає стерти координату y з точки та представити її як 1 біт (непарне y або парне y).

Стиснута точка ЕК – це точка ЕК {x, y}, представлена в її коротшій формі {x, непарне/парне}. Відкриті ключі ECC є точками ЕК, тому їх також можна стиснути таким же чином.

Щоб розпакувати точку, ми можемо обчислити її дві можливі координати y за формулами:

$$y_1 = \text{mod\_sqrt}(x^3 + ax + b, p);$$
$$y_2 = p - \text{mod\_sqrt}(x^3 + ax + b, p).$$

Потім ми беремо непарну або парну з наведених вище координат (відповідно до додаткового біта парності в стисненому представленні).

Модульний квадратний корінь (mod_sqrt) можна обчислити за допомогою алгоритму Тонеллі-Шенкса.

Візьмемо приклад: на еліптичній кривій $y^2 \equiv x^3 + 7 \pmod{17}$ точку P {10, 15} можна стиснути як C {10, nepar}. Для декомпресії спочатку обчислюємо дві можливі координати y для x = 10 за наведеними вище формулами: $y_1 = 2$ і $y_2 = 15$. Потім вибираємо непарну: $y = 15$. Розпакована точка — {10, 15}.

Наведений нижче код реалізує стиснення та декомпресію відкритого ключа в Python. Він використовує бібліотеку під назвою nummaster для функції «модульний квадратний корінь», яка недоступна в Python. Спочатку потрібно встановити пакет nummaster

Тепер реалізуємо функції стиснення та декомпресії точки ЕК в Python, як це показано на рисунку 3.11.

```python
from nummaster.basic import sqrtmod

def compress_point(point):
    return (point[0], point[1] % 2)

def uncompress_point(compressed_point, p, a, b):
    x, is_odd = compressed_point
    y = sqrtmod(pow(x, 3, p) + a * x + b, p)
    if bool(is_odd) == bool(y & 1):
        return (x, y)
    return (x, p - y)
```

Рисунок 3.11 – Функції стиснення та декомпресії точки ЕК

Нарешті, стиснемо і розпакуємо точку {10, 15} на кривій $y^2 \equiv x^3 + 7 \pmod{17}$, як показано на рисунку 3.12.

```python
p, a, b = 17, 0, 7
point = (10, 15)
print(f"original point = {point}")
compressed_p = compress_point(point)
print(f"compressed = {compressed_p}")
restored_p = uncompress_point(compressed_p, p, a, b)
print(f"uncompressed = {restored_p}")
```

Рисунок 3.12 – Стискання та розпаковка точки ЕК

Результат виконання наведеного коду приведений на рисунку 3.13.

```
original point = (10, 15)
compressed = (10, 1)
uncompressed = (10, 15)
```

Рисунок 3.13 – Стиснута та розпакована точки ЕК

Виконано стиснення точки на еліптичній кривій, для перевірки написаних функцій та подальшого використання коду для шифрування з використаннням ECC.

3.3 Вибір параметрів еліптичної кривої

Еліптичні криві ECC описуються набором параметрів області еліптичної кривої, таких як параметри рівняння кривої, параметри поля та

координати точки генератора. Ці параметри вказуються в стандартах криптографії, наприклад:

- SEC 2: Рекомендовані параметри області еліптичної кривої.
- NIST FIPS PUB 186-4 Стандарт цифрового підпису (DSS).
- Стандарт Brainpool ECC (RFC-5639).

Ці стандарти визначають параметри для набору іменованих кривих, таких як secp256k1, P-521 і brainpoolP512t1. Еліптичні криві над скінченними полями, описані в цих криптостандартах, добре досліджені та проаналізовані криптографами і вважаються такими, що мають певну надійність захисту, також описану в цих стандартах.

Деякі криптографи (наприклад, Деніел Бернштейн) вважають, що більшість кривих, описаних в офіційних крипто-стандартах, є «небезпечними» і визначають власні криптостандарти, які розглядають безпеку ECC на набагато ширшому рівні.

У стандарті SafeCurves Бернштейна перераховані криві, які є безпечними відповідно до набору вимог безпеки ECC.

Для використання ECC всі сторони, що спілкуються, повинні узгодити параметри області ЕК (всі елементи, що визначають еліптичну криву). Рекомендується використовувати іменовану криву з наведених вище стандартів з принаймні 256-бітовим модулем. Стандартні криві добре вивчені криптографами, щоб гарантувати надійність їх безпеки.

Не варто використовувати власну еліптичну криву (з нестандартними параметрами домену), якщо не були проведені дослідження її криптовластивостей. Багато кривих мають слабкі сторони, які роблять проблему ECDLP не такою складною та ставлять під загрозу безпеку. Якщо з питань безпеки не можна використовувати закриті криві, варто використовувати стандартну безпечну криву зі списку SafeCurves.

У криптографії ECC використовуються еліптичні криві над скінченними полями, де модуль p і порядок n є дуже великими цілими числами (n зазвичай є простим числом), напр. 256-бітове число. Кінцеве поле

кривої має квадратну форму розміру *p* * *p*, яка неймовірно велика, і всі можливі точки ЕК на кривій (порядок кривої *n*) також є дуже великим цілим числом, напр. 256-біт. Наприклад, параметри домену для кривої secp256k1 (крива Bitcoin) визначаються таким чином:

p (модуль) = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC2F

n (порядок; розмір; кількість усіх можливих балів EC) = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141

a (константа "a" в y^2 ≡ x^3 + a*x + b (mod p)) = 0x0000000000000000000000000000000000000000000000000000000000000000

b (постійна "b" в y^2 ≡ x^3 + a*x + b (mod p)) = 0x0000000000000000000000000000000000000000000000000000000000000000

g (точка генератора кривої G {x, y}) = (0x79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798,
0x483ada7726a3c4655da40bd140b40b40b40b40b40b40b40bdc40b7dc40b7dc40bd140b7dc46bd16bdc40bdc40bdcdc4655dcdc40bdcdc465dd

h (кофактор, зазвичай 1) = 01

Ми вже знаємо, що 256-бітна крива (це означає, що p і n є 256-бітовими числами) забезпечує 128-бітну надійність безпеки, що означає, що для пошуку приватного ключа з відкритого ключа або підпису, найвідоміший неквантовий алгоритм займе приблизно 2128 операцій. Визначена вище крива ECC secp256k1 має 128-бітну криптостійкість.

Тепер введемо в дію наведені вище параметри домену для кривої secp256k1. Давайте визначимо ЕК та обчислимо відкритий ключ для певного приватного ключа. Для цього побудуємо код приведени на рисунку 3.14.

```python
from tinyec.ec import SubGroup, Curve

# Domain parameters for the `secp256k1` curve
# (as defined in http://www.secg.org/sec2-v2.pdf)
name = 'secp256k1'
p = 0xfffffffffffffffffffffffffffffffffffffffffffffffffffffffefffffc2f
n = 0xfffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141
a = 0x0000000000000000000000000000000000000000000000000000000000000000
b = 0x0000000000000000000000000000000000000000000000000000000000000007
g = (0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798,
     0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8)
h = 1
curve = Curve(a, b, SubGroup(p, g, n, h), name)
print('curve:', curve)

privKey = int('0x51897b64e85c3f714bba707e867914295a1377a7463a9dae8ea6a8b914246319', 16)
print('privKey:', hex(privKey)[2:])

pubKey = curve.g * privKey
pubKeyCompressed = '0' + str(2 + pubKey.y % 2) + str(hex(pubKey.x)[2:])
print('pubKey:', pubKeyCompressed)
```

Рисунок 3.14 – Визначення еліптичної кривої. Фрагмент коду

Наведений код(рисунок 3.14) визначає криву secp256k1 через параметри її домену та обчислює відкритий ключ за даним приватним ключем. Це робиться шляхом множення генератора кривих G на закритий ключ. Результат правильний, як це видно з виводу програми(рисунок 3.15).

```
curve: "secp256k1" => y^2 = x^3 + 0x + 7 (mod 115792089237316195423570985008687907853269
privKey: 51897b64e85c3f714bba707e867914295a1377a7463a9dae8ea6a8b914246319
pubKey: 02f54ba86dc1ccb5bed0224d23f01ed87e4a443c47fc690d7797a13d41d2340e1a
```

Рисунок 3.15 – Крива та ключі

Відкритий ключ стискається і кодується в стандартному форматі (кодує координату y як префікс 02 або 03).

Еліптичні криві в криптографії еліптичної кривої (ECC) можуть бути представлені в кількох формах (уявленнях), які, як доведено, є біраціонально еквівалентними (ізоморфними):

Форма еліптичної кривої Вейєрштрасса:

$$y^2 = x^3 + ax + b$$

Прикладом кривої Вейєрштрасса, використаної в ECC, є secp256k1, яка має вигляд $y^2 = x^3 + 7$

Форма еліптичної кривої Монтгомері:

$$By^2 = x^3 + Ax^2 + x.$$

Прикладом кривої Монтгомері, що використовується в ECC, є Curve25519, яка має вигляд $y^2 = x^3 + 486662x^2 + x$.

Форма еліптичної кривої Едвардса:

$$x^2 + y^2 = 1 + dx^2y^2.$$

Прикладом кривої Едвардса, що використовується в ECC, є Curve448, яка має вигляд $x^2 + y^2 = 1 - 39081x^2y^2$.

З міркувань продуктивності криптографія з еліптичною кривою (ECC) іноді використовує криві Едвардса, які є еліптичними кривими в такій формі:

$$x^2 + y^2 = 1 + dx^2y^2.$$

Наприклад, якщо d = 300, крива Едвардса $x^2 + y^2 = 1 + 300x^2y^2$ виглядає так як показано на рисунку 3.16.

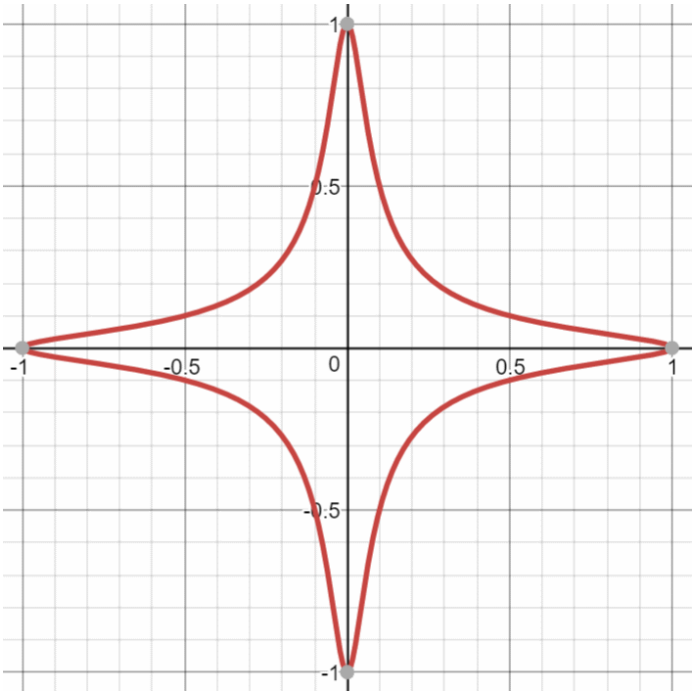

Рисунок 3.16 – Крива Едвардса

Кожна крива Едвардса біраціонально еквівалентна еліптичній кривій у формі Вейєрштрасса ($y^2 = x^3 + ax + b$) і, таким чином, має ті самі властивості, що й класичні еліптичні криві.

Криві Едвардса над скінченним простим полем $\mathbb{F}p$ (де p — велике просте число) забезпечують швидке множення від цілого до точки ЕС, яке має подібні криптографічні властивості, як і класичні еліптичні криві, а проблема ECDLP має таку саму обчислювальну складність, що підходить для криптографічних цілей.

Прикладами добре відомих криптографічних еліптичних кривих Едвардса над кінцевими простими полями є: Curve1174 (251-біт), Curve25519 (255-біт), Curve383187 (383-біт), Curve41417 (414-біт), Curve448 (448-біт), E-521 (521-розрядний) та інші.

Curve25519 має порядок (у своїй базовій циклічній групі) n = 2252 + 0x14def9dea2f79cd65812631a5cf5d3ed і кофактор h = 8 і забезпечує 125,8-бітну надійність безпеки (інколи її називають ~ 128-бітною). Закриті ключі для Curve25519 мають 251 біт і зазвичай кодуються як 256-бітові цілі числа (32 байти, 64 шістнадцяткові цифри). Відкриті ключі зазвичай кодуються також як 256-бітові цілі числа (255-бітова координата y + 1-бітова координата x), і це дуже зручно для розробки програмних засобів.

На основі Curve25519 виведена функція ECDH, яка називається X25519 (використовується для схем узгодження ключів Діффі-Хеллмана з еліптичним ключем), і схема швидкого цифрового підпису під назвою Ed25519 на основі алгоритму EdDSA. Ці схеми дуже швидкі, оскільки передбачають множення та інші прості операції з невеликими цілими числами (переважно 32-розрядна арифметика), які можуть бути ефективно реалізовані в сучасних мікропроцесорах (ЦП). Зауважимо, що X25519 і Ed25519 використовують різні кодування для точок ЕС, тому вони несумісні безпосередньо і вимагають перетворення, якщо ви хочете використовувати одні й ті ж пари публічного та приватного ключів.

Крива 448 (Curve448) — це розкручена крива Едвардса, що визначається рівнянням:

$$x^2 + y^2 = 1 - 39081x^2y^2,$$

над скінченним простим полем $\mathbb{F}p$, де p = 2448 - 2224 - 1. Воно має порядок n = 2446 - 0x8335dc163bb124b65129c96fde933d8d723a70aadc873d6d54s у будь-якій криві: $ax + b$), але наведена вище форма Едвардса забезпечує значну оптимізацію обчислень точки ЕК та покращує продуктивність.

Curve448 забезпечує ~ 224-бітний рівень безпеки (точніше 222,8-біт). Закриті ключі для Curve448 мають 446 біт і зазвичай кодуються як 448-бітові цілі числа (56 байт, 112 шістнадцяткових цифр). Відкриті ключі також кодуються як 448-бітові цілі числа.

Curve448 підходить для узгодження ключів ECDH (функція ECDH, відома як X448) і для швидких цифрових підписів (алгоритм EdDSA, відомий як Ed448 або edwards448). X448 і Ed448 використовують різні кодування для точок ЕС, тому вони несумісні безпосередньо і вимагають перетворення, якщо потрібно використовувати ті самі пари публічно-приватних ключів.

Крива Curve448 має перевагу в криптостійкості над Curve25519 і використовується коли програмному засобу потрібен більш високий рівень безпеки, але необхідно врахувати, що Curve448 приблизно в 3 рази повільніша Curve25519 і використовує більшу довжину ключа та довжину підпису.

Curve25519 надають перевагу коли потрібна більша швидкодія ніж в Curve448, а самі ключі та підписи менші.

У загальному випадку майте на увазі, що Curve25519 швидше, ніж secp256k1 та інші 256-бітові стандартні криві NIST, і вважається більш безпечним, тому це рекомендований вибір для ~ 128-бітної безпеки. Аналогічно, Curve448 має кращу продуктивність, ніж класичні криві з такою ж довжиною ключа, тому це рекомендована крива для ~ 224-бітної безпеки.

Щоб продемонструвати еліптичну криву Curve25519 на практиці, спочатку встановимо криптобібліотеку pynacl для Python:

Прив’язка Python до бібліотеки Networking and Cryptography (NaCl) (PyNaCl) реалізує багато сучасних криптографічних алгоритмів, включаючи арифметику точки ЕК над сигнатурами Curve25519 і Ed25519.

Далі згенеруємо випадковий 252-бітовий закритий ключ і відповідний йому відкритий ключ (точка ЕС) на Curve25519 (обидва ключі будуть закодовані всередині як 256-бітові цілі числа):

```
1 from nacl.public import PrivateKey
2 import binascii
3
4 privKey = PrivateKey.generate()
5 pubKey = privKey.public_key
6
7 print("privKey:", binascii.hexlify(bytes(privKey)))
8 print("pubKey: ", binascii.hexlify(bytes(pubKey)))
```

Рисунок 3.17 – Фрагмент коду для генерації ключів кривої Curve25519

Зразок результату з наведеного вище коду (рисунок 3.17) показує, що і відкритий, і закритий (секретні) ключі на Curve25519 закодовані як 256-бітові цілі числа (64 шістнадцяткові цифри, 32 байти), і це спрощує розроку.

```
1 privKey: b'8175f7cd524a59b6efbd447985ce5d97c546b319521ff236203970e50052c641'
2 pubKey:  b'cf97a96568fee4ddb232f617fd5b9df2d2e5b90e68ba7f6d5129ea92d7d8f95e'
```

Рисунок 3.18 – Згенеровані ключі для кривої Curve25519

Насправді, різні криптобібліотеки можуть використовувати різні кодування ключів, і зазвичай ключі X25519 ECDH кодуються інакше, ніж ключі Ed25519 (координати кривої Монтгомері та скручені координат кривої Едвардса).

3.4 Реалізація бміну ключами ECDH, шифрування та дешифрування

Реалізуэмо шифрування/дешифрування з відкритим ключем на основі еліптичної кривої (асиметрична схема шифрування на основі ECC). Це нетривіально і зазвичай передбачає розробку гібридної схеми шифрування, що включає криптографію ECC, обмін ключами ECDH та алгоритм симетричного шифрування.

Припустимо, що у нас є пара приватних і відкритих ключів ECC. Потрыбно шифрувати та дешифрувати дані за допомогою цих ключів. За визначенням асиметричне шифрування працює наступним чином: якщо ми зашифруємо дані за допомогою приватного ключа, ми зможемо пізніше розшифрувати зашифрований текст за допомогою відповідного відкритого ключа(рисунок 3.19).

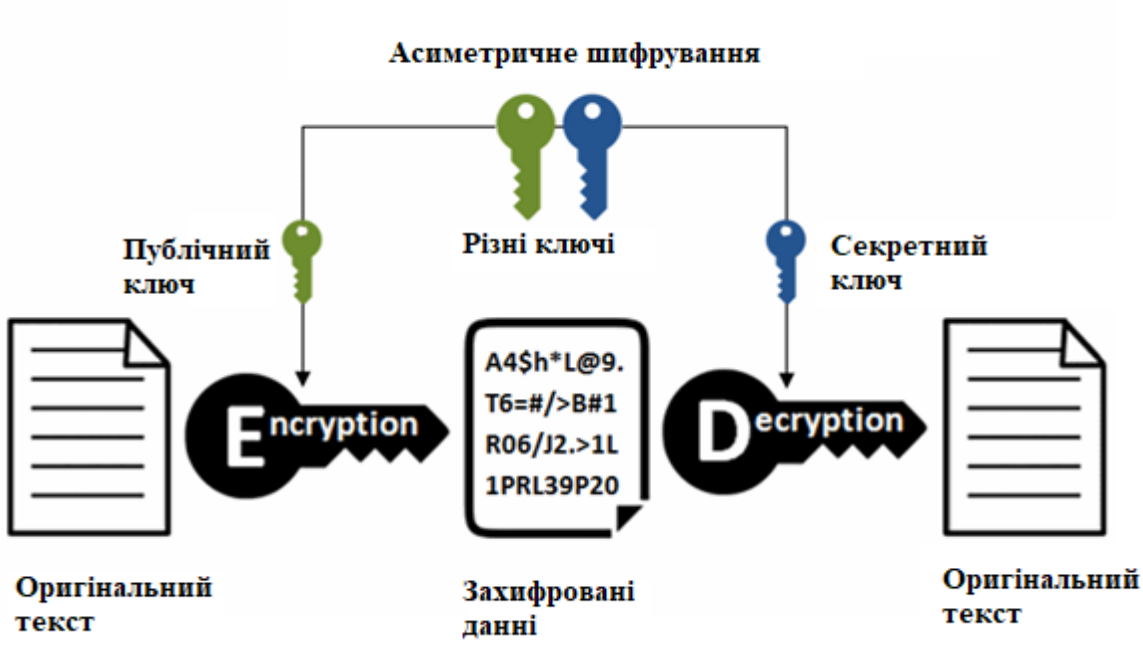

Рисунок 3.19 – Схема шифрування та дешифрування ECDH

Вищезазначений процес можна застосувати безпосередньо для криптосистеми RSA, але не для ECC. Криптографія з еліптичною кривою (ECC) безпосередньо не забезпечує метод шифрування. Замість цього ми можемо розробити гібридну схему шифрування, використовуючи схему

обміну ключами ECDH (еліптична крива Діффі-Хеллмана), щоб отримати спільний секретний ключ для симетричного шифрування та дешифрування даних.

Більшість гібридних схем шифрування (процес шифрування) працюють як показано на рисунку 3.20.

Рисунок 3.20 – Гібридна схема шифрування

Більшість гібридних схем шифрування мають схему роботи зображену на рисунку 3.21 (процес дешифрування).

Рисунок 3.21 – Гібридна схема дешифрування

Детальніше розберем та реалізуємо гібридну схему шифрування на основі ECC.

Припустимо, що ми маємо криптографічну еліптичну криву над кінцевим полем разом з точкою її генератора G. Ми можемо використовувати наступні дві функції, щоб обчислити загальний секретний ключ для шифрування та дешифрування (отриманий зі схеми ECDH) обчислити:

- EncryptionKey(pubKey) --> (sharedECCKey, ciphertextPubKey).

- Згенерувати ciphertextPrivKey = новий випадковий закритий ключ.

- Розрахувати ciphertextPubKey = ciphertextPrivKey * G.

Обчислимо загальний секрет ECDH: sharedECCKey = pubKey * ciphertextPrivKey.

Повернути обидва ключі sharedECCKey + ciphertextPubKey. Необхідно використати sharedECCKey для симетричного шифрування. Використовуйте випадково згенерований ciphertextPubKey, щоб пізніше обчислити ключ дешифрування.

Обчислити ключ дешифрування (приватний ключ, шифр тексту ключ) --> sharedECCKey

Обчисліть загальний секрет ECDH: sharedECCKey = ciphertextPubKey * privKey.

Поверніть sharedECCKey і використовуйте його для розшифрування.

Наведені вище обчислення використовують ту саму математику, як і алгоритм ECDH. Нагадаємо, що точки EC мають таку властивість:

$$(a * G) * b = (b * G) * a.$$

Тепер припустимо, що a = privKey, a * G = pubKey, b = ciphertextPrivKey, b * G = ciphertextPubKey.

Вищенаведене рівняння набуває такого вигляду:

pubKey * ciphertextPrivKey = ciphertextPubKey * privKey = sharedECCKey

Саме це обчислюють дві вищенаведені функції, безпосередньо слідуючи схемі узгодження ключів ECDH. У гібридних схемах шифрування інкапсульований ключ ciphertextPubKey також відомий як «ефемерний ключ», оскільки він використовується тимчасово для отримання симетричного ключа шифрування за допомогою схеми угоди з ключем ECDH.

ECDH (Elliptic Curve Diffie–Hellman Key Exchange) — це анонімна схема угоди про ключі, яка дозволяє двом сторонам, кожна з яких має пару відкритих і приватних ключів з еліптичною кривою, встановлювати загальний секрет через незахищений канал. ECDH дуже схожий на класичний алгоритм DHKE (Обмін ключами Діффі-Хеллмана), але він використовує множення точок ECC замість модульних підведень до степенів. ECDH базується на такій властивості точок EC:

$$(a * G) * b = (b * G) * a.$$

Якщо у нас є два секретних числа a і b (два приватні ключі, що належать Алісі та Бобу) і еліптична крива ECC з точкою генератора G, ми можемо обмінюватися через незахищений канал значеннями (a * G) і (b * G) (відкриті ключі Аліси та Боба), а потім ми можемо отримати загальний секрет: секрет = (a * G) * b = (b * G) * a. Досить просто. Вищенаведене рівняння набуває такого вигляду: alicePubKey * bobPrivKey = bobPubKey * alicePrivKey = секрет

Алгоритм ECDH (Еліптична крива Діффі-Хеллмана обмін ключами) є тривіальним та містить декілька кроків:

- Аліса генерує випадкову пару ключів ECC: {alicePrivKey, alicePubKey = alicePrivKey * G}.

- Боб генерує випадкову пару ключів ECC: {bobPrivKey, bobPubKey = bobPrivKey * G}.

- Аліса і Боб обмінюються своїми відкритими ключами через незахищений канал (наприклад, через Інтернет).

- Аліса обчислює sharedKey = bobPubKey * alicePrivKey.

- Боб обчислює sharedKey = alicePubKey * bobPrivKey.

Тепер і Аліса, і Боб мають однаковий sharedKey == bobPubKey * alicePrivKey == alicePubKey * bobPrivKey.

Тепер давайте реалізуємо алгоритм ECDH (Elliptic Curve Diffie–Hellman Key Exchange) в Python.

Ми будемо використовувати бібліотеку tinyec для ECC в Python (рисунок 3.22).

```
from tinyec import registry
import secrets

def compress(pubKey):
    return hex(pubKey.x) + hex(pubKey.y % 2)[2:]

curve = registry.get_curve('brainpoolP256r1')

alicePrivKey = secrets.randbelow(curve.field.n)
alicePubKey = alicePrivKey * curve.g
print("Alice public key:", compress(alicePubKey))

bobPrivKey = secrets.randbelow(curve.field.n)
bobPubKey = bobPrivKey * curve.g
print("Bob public key:", compress(bobPubKey))

print("Now exchange the public keys (e.g. through Internet)")

aliceSharedKey = alicePrivKey * bobPubKey
print("Alice shared key:", compress(aliceSharedKey))

bobSharedKey = bobPrivKey * alicePubKey
print("Bob shared key:", compress(bobSharedKey))

print("Equal shared keys:", aliceSharedKey == bobSharedKey)
```

Рисунок 3.22 – Реалізація алгоритму обміну ключами ECDH

Еліптична крива, яка використовується для обчислень ECDH, є 256-бітовою кривою brainpoolP256r1(рисунок 3.22). Закриті ключі мають 256 біт

(64 шістнадцяткові цифри) і генеруються випадковим чином. Відкриті ключі будуть мати 257 біт (65 шістнадцяткових цифр) через стиснення ключа.

Результат наведеного вище коду приведений на рисунку 3.23.

```
Alice public key: 0x66c808e6b5be6d6620934bc6ffa2b8b47f9786c002bfb06d53a0c2753564
Bob public key: 0x7d15195432d1ac7f38aeb054d07d9b2e1faa913b78ad04d5efdd4a1ee8d9a3
Now exchange the public keys (e.g. through Internet)
Alice shared key: 0x90f5a1cf2ed1dbb0322178df6bb0dd72c541884618b2989a3e5e66319866
Bob shared key: 0x90f5a1cf2ed1dbb0322178df6bb0dd72c541884618b2989a3e5e663198667a
Equal shared keys: True
```

Рисунок 3.23 – Результат роботи алгоритму обміну ключами ECDH

Наведений код Python використовує бібліотеку tinyec для створення пари приватних і відкритих ключів ECC для одержувача повідомлення (на основі кривої brainpoolP256r1), а потім отримання секретного спільного ключа (для шифрування) і відкритого ключа ефемерного зашифрованого тексту (для ECDH) з відкритий ключ одержувача, а пізніше отримати той самий секретний спільний ключ (для розшифрування) із приватного ключа одержувача та згенерованого раніше відкритого ключа ефемерного зашифрованого тексту(рисунок 3.24).

```python
import secrets

curve = registry.get_curve('brainpoolP256r1')

def compress_point(point):
    return hex(point.x) + hex(point.y % 2)[2:]

def ecc_calc_encryption_keys(pubKey):
    ciphertextPrivKey = secrets.randbelow(curve.field.n)
    ciphertextPubKey = ciphertextPrivKey * curve.g
    sharedECCKey = pubKey * ciphertextPrivKey
    return (sharedECCKey, ciphertextPubKey)

def ecc_calc_decryption_key(privKey, ciphertextPubKey):
    sharedECCKey = ciphertextPubKey * privKey
    return sharedECCKey

privKey = secrets.randbelow(curve.field.n)
pubKey = privKey * curve.g
print("private key:", hex(privKey))
print("public key:", compress_point(pubKey))

(encryptKey, ciphertextPubKey) = ecc_calc_encryption_keys(pubKey)
print("ciphertext pubKey:", compress_point(ciphertextPubKey))
print("encryption key:", compress_point(encryptKey))

decryptKey = ecc_calc_decryption_key(privKey, ciphertextPubKey)
print("decryption key:", compress_point(decryptKey))
```

Рисунок 3.24 – Генерація набору ключів brainpoolP256r1 для алгоритму ECDH

Завдяки рандомізації, якщо запустити наведений код, ключі будуть іншими, але обчислений загальний секрет для Аліси та Боба в кінці завжди буде однаковим. Згенерований загальний секрет — це 257-бітове ціле число (стислена точка ЕК для 256-бітної кривої, закодована як 65 шістнадцяткових цифр).

Код досить простий і демонструє, що ми можемо згенерувати пару { секретний ключ + відкритий ключ зашифрованого тексту } з даного відкритого ключа ЕС, а пізніше ми можемо відновити той самий секретний ключ із пари { відкритий ключ зашифрованого тексту + закритий ключ}. Наведений вище код видає такий результат(рисунок 3.25).

```
private key: 0x2e2921b4cde59cdf01e7a014a322abd530b3015085c31cb6e59502da761d29e9
public key: 0x850d3873cf4ac50ddb54ddbd27f8225fc43bd3f4c2cc0a4f9d1f9ce15fc4eb711
ciphertext pubKey: 0x71586f9999d3ee050005054bc681c1d96c5eb054ca15b080ba245e49562
encryption key: 0x9d13d3f8f9747669432f575731926b5ed99a6883f00146cbd3203ffa7ff8b1
decryption key: 0x9d13d3f8f9747669432f575731926b5ed99a6883f00146cbd3203ffa7ff8b1
```

Рисунок 3.25 – Результати генерації ключів для кривої brainpoolP256r1

З наведеного вище результату ясно, що ключ шифрування (отриманий з відкритого ключа) і ключ дешифрування (отриманий з відповідного приватного ключа) однакові. Це пов’язано з обговореною вище властивістю ECC: pubKey * ciphertextPrivKey = ciphertextPubKey * privKey. Ці ключі будуть використовуватися для шифрування та дешифрування даних в інтегрованій схемі шифрування. Наведений вище вихід буде іншим, якщо ви запустите код (через випадковість, що використовується для генерації ciphertextPrivKey, але ключі шифрування та дешифрування завжди будуть однаковими (загальний секрет ECDH).

Продемонстрований механізм для створення спільного ефемерного секретного ключа на основі пари ключів ECC є прикладом KEM (механізм інкапсуляції ключів), заснований на ECC і ECDH.

Криптосистеми на еліптичних кривих [22] відносяться до класу криптосистем з відкритим ключом. Їх безпека, як правило, заснована на трудности вирішення завдань дискретного логарифмування в групі точок

еліптичної кривої над кінцевим полем. Этим и обусловлена їх висока криптостойкость порівняння з іншими алгоритмами. Существуют стойкість

криптоалгоритми на еліптичних кривих, засновані на трудностях розкладання великих цілих чисел, коли еліптична кривая задається над кінцевим кольцом за складним модулем, але вони зустрічаються досить рідко. Однак слід відзначити, що криптостойкость є відповідним поняттям, пов'язаним з поняттям найлучшого відомого алгоритму взлома системи.

Еліптичні криві – математичний об'єкт, який може бути визначений над будь-яким полем. В криптографії зазвичай використовуються кінцеві поля. Для точок на еліптичній кривій вводиться операція додавання, яка має ту саму роль, що і операція множення в криптосистемах RSA та ЕльГамаля.

Ще однією перевагою криптосистеми на еліптичних кривих є висока швидкість обробки інформації.

ECC, володіє більш високою криптостійкістю, криптосистеми на еліптичних кривих дозволяє використовувати ключ меншої довжини. Однак прийнятна для роботи в мережах швидкість вирахування досягається лише при використанні спеціалізованих обчислювальних систем (це цілком можливо для криптосистеми з відкритим ключом) і поля спеціальних характеристик.

Криптосистеми на еліптичних кривих, як і інші криптосистеми з відкритим ключом, не варто застосовувати для шифрування великих обсягів данних. Але тому їх можна ефективно використовувати для системи цифрових підписів і ключового обміну. З 1998 року використання еліптичних кривих для рішень криптографічних задач, таких як цифровий підпис, було закріплено в стандартах США ANSI X9.62 і FIPS 186–2.

В Україні прийнято стандарт цифрової підписи, заснований на еліптичних кривых ДСТУ 4145–2002). Зазначимо, що безпека таких систем цифрового підпису опирається не тільки на стійкість алгоритму на еліптичних кривих, але і на стійкість використовуваної хеш-функції.

Багаточисленні дослідження показали, що криптосистеми на основі еліптичні кривих перевершують інші системи з відкритим ключом по двома важливими параметрами: ступеня захисту в розрахунку на кожен біт ключа і швидкодії при програмній і апаратній реалізації. Це пояснюється тем, що для вичислення зворотних функцій на еліптичних кривих відомі тільки алгоритми з експоненційним ростом трудоемкости, тоді як для звичайних систем пропозиції субекспоненциальних методів. В результаті рівня стійкості, який досягається, скажем, в RSA при використанні 1024-бітових модулів, в системах на еліптичних кривих реалізується при розмірі модуля 160 біт, що забезпечує більш просту програмну, таку і апаратну реалізацію.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Загальні вимоги охорони праці при роботі з ПК

Комп'ютери значно підвищують продуктивність праці кожного конкретного працівника бібліотеки.

Однак необхідно враховувати, що поряд з безперечними перевагами використання ПЕОМ у роботі може негативно позначитися на здоров'ї працівника, особливо якщо комп'ютер використовується у роботі постійно або значну частину робочого часу.

Працюючи з персональним комп'ютером (ПК) працівник може піддаватися впливу небезпечних і шкідливих виробничих чинників. Під впливом таких факторів може виникнути тимчасове погіршення здоров'я та, відповідно, знизитися працездатність. Так, у працівників, значну частину робочого часу які проводять за комп'ютером, можуть виникнути біль голови, різь в очах і погіршення зору, загальна втома, дратівливість. Як побічні ефекти роботи з комп'ютером можуть виникнути безсоння, біль у суглобах, попереку, кистях рук.

Типовою інструкцією встановлено вимоги з охорони праці під час роботи з ПЕОМ, які є обов'язковими для наймачів незалежно від виду діяльності та форми власності.

У правилах зазначено, що до виконання робіт з ПЕОМ допускаються працюючі, які пройшли у встановлених законодавством випадках та порядку медогляд, інструктаж з охорони праці.

Жінкам з дня встановлення вагітності та під час годування дитини грудьми слід обмежити час роботи з ПЕОМ до 3 годин за робочу зміну з урахуванням забезпечення оптимальних умов праці та регламентованих перерв відповідно до законодавства.

У разі неможливості організації робіт відповідно до вимог частини першої цього пункту з причин, пов'язаних з особливостями технологічного процесу, жінки з часу встановлення вагітності та в період годування дитини грудьми повинні бути переведені на роботи, не пов'язані з використанням ПЕОМ.

Законодавцем перераховані можливі шкідливі та (або) небезпечні виробничі фактори, які можуть вплинути на працюючих:

– підвищений рівень електромагнітних випромінювань; підвищений рівень іонізуючих випромінювань;

– підвищений рівень статичної електрики;

– підвищена напруженість електростатичного поля;

– підвищена чи знижена іонізація повітря;

– підвищена яскравість світла; пряма та відбита блискітність;

– підвищене значення напруги в електричному ланцюзі, замикання якого може статися через тіло людини;

– статичні перевантаження кістково-м'язового апарату та динамічні локальні навантаження м'язів кистей рук; перенапруга зорового аналізатора; розумова перенапруга; емоційні навантаження; монотонність праці.

До самостійної роботи на персональному комп'ютері допускаються особи, які пройшли медичний огляд та не мають протипоказань для роботи на ПК, інструктаж з охорони праці, навчання безпечним методам виконання робіт, перевірку знань вимог охорони праці, мають 1 групу з електробезпеки.

Працівник повинен знати, що під час виконання робіт на ПК на нього можуть впливати такі шкідливі та (або) небезпечні виробничі фактори:

- підвищена температура поверхонь ПК;

- підвищена або знижена температура повітря робочої зони;

- підвищене значення напруги в електричному ланцюзі, замикання;

- підвищений рівень статичної електрики;

- підвищений рівень електромагнітних випромінювань;

- підвищена напруженість електричного поля;

- відсутність чи нестача природного світла;

- недостатня штучна освітленість робочої зони;

- підвищена яскравість світла;

- підвищена контрастність;

- пряма та відбита блискітність;

- зорова напруга;

- тривалі статичні навантаження;

- монотонність трудового процесу.

4.2 Обов’язки працівника

Серед обовязків працюючих за ПК є наступні правила:

- знати та дотримуватись вимог експлуатаційних документів організацій-виготовлювачів використовуваної ПК;

- дотримуватися режиму праці та відпочинку, встановленого законодавством, правилами внутрішнього трудового розпорядку організації, трудової дисципліни, виконувати вимоги з охорони праці, правил особистої гігієни; виконувати вимоги пожежної безпеки; курити лише у спеціально призначених для куріння місцях;

- дбати про особисту безпеку та особисте здоров'я, а також про безпеку оточуючих у процесі виконання робіт або під час перебування на території організації;

- утримувати робоче місце у порядку та чистоті;

- знати місцезнаходження аптечки першої медичної допомоги;

- повідомляти безпосередньому керівнику або іншій уповноваженій посадовій особі наймача про несправність ПК та периферійних пристроїв (принтера, сканера, клавіатури ПК, електричних комп'ютерних мережевих пристроїв, блоку безперебійного живлення та інших пристроїв) та не розпочинати роботу до їх усунення;

- негайно повідомляти безпосереднього керівника або іншої уповноваженої посадової особи наймача про будь-яку ситуацію, яка загрожує життю або здоров'ю працюючих та оточуючих; виконувати інші обов'язки, передбачені законодавством.

Не допускається знаходження працюючих у стані алкогольного, наркотичного або токсичного сп'яніння, а також розпивання спиртних напоїв, споживання наркотичних засобів, психотропних речовин, їх аналогів, токсичних засобів на робочому місці та у робочий час.

При виконанні робіт на персональному комп'ютері працівник зобов'язаний:

- дотримуватися виробничої та технологічної дисципліни;

- дотримуватися режиму праці та відпочинку залежно від тривалості, виду та категорії трудової діяльності;

- виконувати лише ту роботу, яка визначена його посадовою інструкцією;

- Підтримувати порядок на робочому місці протягом усього робочого часу;

- про всі несправності ПК та електроживлення негайно повідомляти безпосередньому керівнику;

- дотримуватись вимог пожежної безпеки та електробезпеки.

ВИСНОВКИ

В роботі розглянута актуальна тема аналіз можливостей асиметричної схеми шифрування алгоритмом Діффі Хеллмана на еліптичних кривих. Проаналізовано етапи алгоритм ECC, що дало можливість виявити переваги та недоліки використання алгоритму ECC.

Проведено аналіз схем генерування ключів ECC та ECDF, що дозволяє в подальшому реалізувати відповідне програмне забезпечення на мові Python.

Проведено аналіз алгоритму множення точки ECC на ціле число, який використовується в криптографії на еліптичних кривих та є частиною схеми шифрування. Наведено визначення та проаналізовано основні поняття, що використовуються в еліптичній криптографії.

Проведено дослідження схем генерування та обміну приватного та відкритого ключів та роботу з генераторною точкою в ECC. Описано основні методи роботи з групами точок на еліптичних кривих.

Виконано практичну реалізацію модулів для виконання операцій в еліптичній криптографії, таких як, множення в ЕК, стиснення відкритих ключів , вибір параметрів еліптичної кривої, реалізація бміну ключами ECDH, шифрування та дешифрування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Сава Л.М. Аналіз та реалізація криптографічних перетворень для алгоритму ECDH. Матеріали V Міжнародної студентської науково - технічної конференції / Тернопіль: Тернопільський національний технічний університет ім. І.Пулюя (м. Тернопіль, 28-29 квітня 2022 р.), 2022.-с. 144.

2. Якименко І.З. Підвищення ефективності обчислення точок на еліптичних кривих над обмеженими полями.// І.З.Якименко, А.О.Ботюк, М.П.Карпінський/ Вісник Тернопільського державного технічного університету імені Ів.Пулюя, - Том 8, - №4 – 2003. – С: 67 – 73.

2. Карпінський М.П. Еліптична крива для асиметричної криптографічної системи. // М.П.Карпінський, І.З.Якименко, І. Дуда /Вісник Тернопільського державного технічного університету імені Ів.Пулюя, - Том 6, - №3 – 2001. – С: 91 – 95.

4. X9.62-1998 [2]. Public Key Cryptography For The Financial Services Industry. Public Key Cryptography For The Financial Services Industry.

5. Карпінський М.П. Використання символів Якобі в криптографії ЕК // М.П.Карпінський, І.З.Якименко, А.А.Хомінчук / ДУІКТ/06/ Матеріали ІІІ Міжнародної науково-технічної конференції "Світ інформації та телекомунікацій-2006" Україна.– Київ. – 2006. – С.208.

6. М.Карпінський Метод генерування параметрів еліптичних кривих. // М.Карпінський, І.Васильцов, І.Якименко, Я.Кінах./ Правове, нормативне, та метрологічне забезпечення системи захисту інформації в Україні. К.:–2003– № 6.– С.74.

9. Бессалов А.В. Криптосистемы на эллиптических кривых: Учеб. пособие.// А.В.Бессалов, А.Б.Телиженко / – К.: ІВЦ «Видавництво «Політехніка»».– 2004. – 224 С.

10. IEEE P1363 / D8(Draft Version 8). Standard Specifications for Public Key Cryptography.

24. А.А. Болотов Алгоритмические основы эллиптической криптографии./ А.А. Болотов, С.Б. Гашков, А.Б. Фролов, А.А. Часовских./ – Москва:МЭИ.– 2000. – 100 с.

26. Варновский Н. П. Криптография и теория сложности // Математическое просвещение. / Н. П. Варновский / – М:1998. - №2. - С. 71 - 86

38. Schoof R. Elliptic curves over finite fields and the computation of square roots modulo p.// R. Schoof / –Bordeaux: Math. Comput. –1985.–№ 44.– 483–494p.

39. Schoof R. Counting points on elliptic curves over finite fields. J. The´or. Nombres. // R. Schoof / – Bordeaux 7.– 1995.– 219 –254p.

40. ELKIES N. D. Elliptic and modular curves over finite fields and related computational issues. In Computational Perspectives on Number Theory: Proceedings of a Conference in Honor of A. O. L. Atkins, D. A. Buell and J. T. Teitelbaum, eds. AMS/IP Studies in Advanced Mathematics, vol. 7. American Mathematics Society, Providence, R. I.– 1998.– pp. 21–76.

60. Кінах Я.І Удосконалення мережевих криптоаналітичних алгоритмів на еліптичних кривих. // Я.І Кінах, І.З. Якименко/ Вісник Східноукраїнського національного університету імені Володимира Даля.-№6(136), Т.1.- 2009.- С.48-51.

61. Карпінський М.П. Використання технології паралельних обчислень для рахування кількості точок еліптичної кривої. // М.П. Карпінський, І.З. Якименко, А.А. Хомінчук / Вісник Східноукраїнського національного університету імені Володимира Даля.– 2008. – № 8 (126), Частина 1. – С. 207-210.

ДОДАТОК А

Код програмних модулів

```
from tinyec.ec import SubGroup, Curve

field = SubGroup(p=17, g=(15, 13), n=18, h=1)
curve = Curve(a=0, b=7, field=field, name='p1707')
print('curve:', curve)

for k in range(0, 25):
    p = k * curve.g
    print(f"{k} * G = ({p.x}, {p.y})")

from tinyec.ec import SubGroup, Curve

field = SubGroup(p=17, g=(5, 9), n=18, h=1)
curve = Curve(a=0, b=7, field=field, name='p1707')
print('curve:', curve)

for k in range(0, 25):
    p = k * curve.g
    print(f"{k} * G' = ({p.x}, {p.y})")

from tinyec import registry

curve = registry.get_curve('secp192r1')
print('curve:', curve)

for k in range(0, 10):
    p = k * curve.g
```

```
    print(f"{k} * G = ({p.x}, {p.y})")

print("Cofactor =", curve.field.h)

print('Cyclic group order =', curve.field.n)

nG = curve.field.n * curve.g
print(f"n * G = ({nG.x}, {nG.y})")

from tinyec import registry
import secrets

curve = registry.get_curve('secp192r1')

privKey = secrets.randbelow(curve.field.n)
pubKey = privKey * curve.g
print("private key:", privKey)
print("public key:", pubKey)

from nummaster.basic import sqrtmod

def compress_point(point):
    return (point[0], point[1] % 2)

def uncompress_point(compressed_point, p, a, b):
    x, is_odd = compressed_point
    y = sqrtmod(pow(x, 3, p) + a * x + b, p)
    if bool(is_odd) == bool(y & 1):
        return (x, y)
    return (x, p - y)
```

```
p, a, b = 17, 0, 7
point = (10, 15)
print(f"original point = {point}")
compressed_p = compress_point(point)
print(f"compressed = {compressed_p}")
restored_p = uncompress_point(compressed_p, p, a, b)
print(f"uncompressed = {restored_p}")

from tinyec.ec import SubGroup, Curve

# Domain parameters for the `secp256k1` curve
# (as defined in http://www.secg.org/sec2-v2.pdf)
name = 'secp256k1'
p = 0xfffffffffffffffffffffffffffffffffffffffffffffffffffffffefffffc2f
n = 0xfffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141
a =
0x0000000000000000000000000000000000000000000000000000000000000000
b =
0x0000000000000000000000000000000000000000000000000000000000000007
g =
(0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798,

0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8)
h = 1
curve = Curve(a, b, SubGroup(p, g, n, h), name)
print('curve:', curve)

privKey =
int('0x51897b64e85c3f714bba707e867914295a1377a7463a9dae8ea6a8b914246319', 16)
```

```
print('privKey:', hex(privKey)[2:])

pubKey = curve.g * privKey
pubKeyCompressed = '0' + str(2 + pubKey.y % 2) + str(hex(pubKey.x)[2:])
print('pubKey:', pubKeyCompressed)

from nacl.public import PrivateKey
import binascii

privKey = PrivateKey.generate()
pubKey = privKey.public_key

print("privKey:", binascii.hexlify(bytes(privKey)))
print("pubKey: ", binascii.hexlify(bytes(pubKey)))

from tinyec import registry
import secrets

def compress(pubKey):
    return hex(pubKey.x) + hex(pubKey.y % 2)[2:]

curve = registry.get_curve('brainpoolP256r1')

alicePrivKey = secrets.randbelow(curve.field.n)
alicePubKey = alicePrivKey * curve.g
print("Alice public key:", compress(alicePubKey))

bobPrivKey = secrets.randbelow(curve.field.n)
bobPubKey = bobPrivKey * curve.g
print("Bob public key:", compress(bobPubKey))
```

```
print("Now exchange the public keys (e.g. through Internet)")

aliceSharedKey = alicePrivKey * bobPubKey
print("Alice shared key:", compress(aliceSharedKey))

bobSharedKey = bobPrivKey * alicePubKey
print("Bob shared key:", compress(bobSharedKey))

print("Equal shared keys:", aliceSharedKey == bobSharedKey)

from tinyec import registry
import secrets

curve = registry.get_curve('brainpoolP256r1')

def compress_point(point):
    return hex(point.x) + hex(point.y % 2)[2:]

def ecc_calc_encryption_keys(pubKey):
    ciphertextPrivKey = secrets.randbelow(curve.field.n)
    ciphertextPubKey = ciphertextPrivKey * curve.g
    sharedECCKey = pubKey * ciphertextPrivKey
    return (sharedECCKey, ciphertextPubKey)

def ecc_calc_decryption_key(privKey, ciphertextPubKey):
    sharedECCKey = ciphertextPubKey * privKey
    return sharedECCKey

privKey = secrets.randbelow(curve.field.n)
```

```
pubKey = privKey * curve.g
print("private key:", hex(privKey))
print("public key:", compress_point(pubKey))

(encryptKey, ciphertextPubKey) = ecc_calc_encryption_keys(pubKey)
print("ciphertext pubKey:", compress_point(ciphertextPubKey))
print("encryption key:", compress_point(encryptKey))

decryptKey = ecc_calc_decryption_key(privKey, ciphertextPubKey)
print("decryption key:", compress_point(decryptKey))
```