

**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІМЕНІ ТАРАСА ШЕВЧЕНКА**

Факультет комп'ютерних наук та кібернетики  
Кафедра математичної інформатики

**Кваліфікаційна робота  
на здобуття ступеня бакалавра**  
за освітньо-професійною програмою “Інформатика”  
спеціальності 122 Комп'ютерні науки на тему:

**ІМПЛЕМЕНТАЦІЯ КРИПТОСИСТЕМ НА ОСНОВІ ЕЛІПТИЧНИХ  
КРИВИХ**

Виконав студент 4-го курсу  
Нікіта СОЛОНКО

---

(підпис)

Науковий керівник:  
Член-кореспондент НАН України, професор  
Анатолій АНІСІМОВ

---

(підпис)

Засвідчую, що в цій роботі немає запозичень з  
праць інших авторів без відповідних посилань.

Студент

---

(підпис)

## РЕФЕРАТ

Обсяг роботи: 41 сторінок, 14 ілюстрацій, 2 таблиць, 33 використаних джерел.

Ключові слова: ЕЛІПТИЧНІ КРИВІ, КРИПТОГРАФІЯ ЕЛІПТИЧНИХ КРИВИХ, CURVE25519, X25519, ED25519, КРИПТОГРАФІЯ З ВІДКРИТИМ КЛЮЧЕМ

Об'єктом дослідження є сучасна криптографія еліптичних кривих.

Метою кваліфікаційної роботи є порівняння 256 бітних еліптичних кривих, що використовуються у сучасній криптографії, шляхом дослідження математичних структур, на основі яких вони розроблені. Як результат дослідження пропозиція найбільш актуальної з них і розробка бібліотеки мовою C++, що імплементує криптосистеми на її основі.

Інструментами створення є безкоштовний редактор коду 'VSCode' та мова розмітки тексту 'Typst', редактор коду 'Clion', мова програмування C++.

Результат роботи: шляхом проведення дослідження визначено, що найбільш перспективною кривою є Curve25519, оскільки вона дає найкращу швидкість роботи, з розглянутих кривих, в протоколі Діффі-Гелламана, що підтверджено бенчмарками. Також ця крива дає можливість використовувати цифровий підпис EdDSA, що перевершує ECDSA. Розроблено бібліотеку мовою C++, що імплементує криптосистеми на основі Curve25519: X25519 та Ed25519.

## ЗМІСТ

Скорочення та умовні позначення .....	5
Вступ .....	6
Розділ 1. ЕЛІПТИЧНІ КРИВІ .....	8
1.1. Переваги над $\mathbb{Z}_p^*$ .....	8
1.2. Загальний огляд еліптичних кривих .....	10
1.2.1. $E / \mathbb{R}$ .....	10
1.2.2. $E / \mathbb{F}_p$ .....	14
1.3. Додаткові обчислення в $E(\mathbb{F}_p)$ .....	16
1.3.1. $ E(\mathbb{F}_p) $ .....	16
1.3.2. Скалярний добуток точки на число .....	16
1.3.3. Дискретний логарифм в $E(\mathbb{F}_p)$ .....	18
1.4. Спарювання точок еліптичної кривої(Pairing) .....	19
1.5. Спеціальні види еліптичних кривих .....	21
1.5.1. Крива Монтгомері .....	21
1.5.2. Крива Едвардса .....	21
1.5.3. Крива Кобліца .....	22
1.6. Припущення на яких будується ECC .....	23
1.6.1. ECDLP .....	23
1.6.2. CDH .....	25
1.6.3. DDH .....	25
1.7. Скручені криві (Twist curves) .....	25
1.8. Застосування еліптичних кривих поза межами ECC .....	26
1.9. Висновки .....	26
Розділ 2. ЗАСТОСУВАННЯ У КРИПТОГРАФІЇ .....	28
2.1. Кодування інформації за допомогою еліптичних кривих .....	28
2.2. Приклади еліптичних кривих у сучасній криптографії .....	28
2.2.1. secp256r1 (P256) .....	29
2.2.2. secp256k1 (Bitcoin curve) .....	29
2.2.3. Curve25519 .....	30
2.2.4. bn256 .....	30
2.3. Протоколи узгодження ключів .....	30
2.3.1. ECDH .....	30
2.4. Протоколи підпису і верифікації .....	32
2.4.1. ECDSA .....	33
2.4.2. EdDSA .....	33
2.4.3. BLS .....	34
2.5. Імплементація бібліотеки .....	36

Висновки .....	38
Перелік джерел посилання .....	39

## СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

**SIMD** Single Instruction Multiple Data;

**ECC** Elliptic curve cryptography;

**EC** Elliptic curve;

**DLP** Discrete Logarithm Problem, задача дискретного логарифму;

**GNFS** General Number Field Sieve;

**CDH** Computational Diffie–Hellman assumption;

**DDH** Decisional Diffie–Hellman assumption;

**NIST** U.S. National Institute of Standards;

**MOV** Menezes, Okamoto and Vanstone

**SEA** Schoof, Elkies, Atkin

## **ВСТУП**

### **Оцінка сучасного стану об'єкта дослідження**

На даний момент, окрім того, що застосування еліптичних кривих у криптографії активно досліджується в академічних роботах, існує безліч прикладних реалізацій ECC, що використовуються повсюдно. Більш того, криптосистеми з відкритим ключем, що використовують еліптичні криві, потроху стають популярнішими за найбільш розповсюджений зараз RSA. Так, наприклад, нові версії ssh рекомендують використовувати алгоритм підпису Ed25519, що побудований на еліптичних кривих Едвардса, замість RSA. Аналогічні рекомендації зараз дають такі сервіси як: GitLab, GitHub, Amazon Web Services, Google Cloud Platform і багато інших.

### **Актуальність роботи та підстави для її виконання**

Як було зазначено раніше, криптосистеми з відкритим ключем побудовані на основі еліптичних кривих вже активно використовуються, але все ще перебувають у стані розвитку. Так, наприклад, стаття [1], в якій була запропонований одна з найбільш популярних зараз еліптичних кривих Curve25519 і алгоритм підпису/верифікації на її основі Ed25519, була написана в 2012 році, що для криптографії недавно. Також зараз існує багато програмних реалізацій таких криптосистем, але, у зв'язку зі складністю їх імплементації, не всі вони є цілком безпечними. Варто зазначити, що з кожним роком архітектура обчислювальних систем розвивається: оптимізуються існуючі інструкції та додаються нові, що відкриває можливості як для оптимізації існуючих реалізацій, так і для написання нових - кращих. Наприклад, в архітектурі x64 апаратна підтримка SIMD обчислень з 512 бітними регістрами AVX-512 [2] з'явилася в 2013 році, вже після виходу статті з описом Ed25519.

### **Мета й завдання роботи**

Метою роботи є дослідження застосування еліптичних кривих у сучасній криптографії та математичних основ криптографії еліптичних кривих, порівняння еліптичних кривих, що використовуються у сучасній криптографії і як результат дослідження імплементація криптографічної бібліотеки, що реалізує криптосистему на основі найбільш актуальної еліптичної кривої.

### **Можливі сфери застосування**

Оскільки програмна реалізація, яка буде розроблена у рамках цієї роботи не буде сертифікована, то її не варто застосовувати в реальних системах, на які може бути здійснена атака, але вона може бути застосована у навчальних і академічних цілях. В перспективі така бібліотека може бути застосована всюди, де потрібна безпечна комунікація в умовах, коли можливе прослуховування, або втручання в канал зв'язку, між сутностями, які ще не узгодили симетричний ключ, наприклад: TLS, HTTPS, SSH і так далі.

## РОЗДІЛ 1. ЕЛІПТИЧНІ КРИВІ

### 1.1. Переваги над $\mathbb{Z}_p^*$

Під  $\mathbb{Z}_p^*$  мається увазі мультиплікативна група лишків за модулем  $p$ , де  $p$  - просте число.

Розглянемо мотивацію використання нового криптографічного примітиву, коли вже побудовано і імplementовано багато криптосистем на основі  $\mathbb{Z}_p^*$ , таких як: Діффі-Геллман, Ель-Гамаль, RSA і багато інших. Проблема використання таких криптосистем полягає у тому, що вони були запропоновані давно: Діффі-Геллман [3] - 1976, RSA [4] - 1978, Ель-Гамаль [5] - 1985. З того часу кратно збільшились обчислювальні можливості. Частота ядер виросла з кГц до ГГц, кількість ядер збільшилась з одиниць до сотень, з'явилися зручні інструменти для об'єднання процесорів у кластери. Також, через бажання зламати дані криптосистеми, багато вчених шукали спосіб знайти більш ефективні алгоритми розв'язання проблем на які вони спираються. Таким чином у 1993 Ленстра [6] придумав ефективний алгоритм розкладання великих чисел на множники - GNFS(General Number Field Sieve), який може розкласти число  $n > 10^{100}$  за час:

$$\exp\left(\left(\left(\frac{64}{9}\right)^{\frac{1}{3}} + o(1)\right)(\log n)^{\frac{1}{3}}(\log \log n)^{\frac{2}{3}}\right)$$

Це стало основною проблемою криптосистем на основі  $\mathbb{Z}_p^*$ , тому що він робить вирішення обчислювальних проблем, на які спираються ці криптосистеми субекспоненційним, замість експоненційних. Варто зазначити, що цей алгоритм не є чисто теоретичною загрозою. За його допомогою у 2019 році [7] було знайдено дискретний логарифм 795 бітного числа. Таким чином, в патенті RSA [4] рекомендований розмір ключів був 200біт, в першій NIST сертифікації -



512 біт, зараз же мінімальний рекомендований розмір - 2048 біт. Важливим також є те, що GNFS неможливо узагальнити на будь-які скінченні циклічні групи, тому він не зачіпає групу точок еліптичної кривої над скінченним полем (позначення буде наведено далі). По цій причині найкращий відомий алгоритм для вирішення проблеми дискретного логарифма для групи точок еліптичних кривих (ECDLP) займає час  $O(\sqrt{q})$ , де  $q$  - розмір групи. Через це розміри ключів, що мають однакову безпеку для  $\mathbb{Z}_p^*$  і для групи точок еліптичної кривої сильно відрізняються. Порівняльна характеристика безпеки ключа в залежності від розміру ключа RSA/Diffie-Hellman і ECC:

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Table 1: NIST Recommended Key Sizes

Рисунок 1: [8]

Як можна побачити з наведеної таблиці, використання еліптичних кривих дає вагому перевагу, оскільки зменшує розмір ключа, що в свою чергу зменшує час, потрібний на проведення операцій з ключем (що дає вигравш у швидкодії), кількість затраченої енергії (що є вагомою перевагою для IoT), навантаження на мережу під час передачі ключа і загалом трафік.

Ще однією перевагою криптографії еліптичних кривих є те, що можна використати багато напрацювань з криптографії з відкритим, що використовує  $\mathbb{Z}_p^*$ . Це впливає з того ECC як складно обчислювальну проблему використовує

знаходження дискретного логарифма - DLP(у випадку еліптичних кривих вживають термін ECDLP), аналогічно до складно обчислювальної проблеми в Діффі-Геллман і ЕльГамаль. До того ж, еліптичні криві мають додаткову структуру, яку не має  $\mathbb{Z}_p^*$ , що дозволяє побудувати неможливі для  $\mathbb{Z}_p^*$  криптосистеми. Такою додатковою структурою є можливість будувати спарювання точок еліптичних кривих(pairing), алгебраїчні решітки(lattices), ізогенії(isogenies). Алгебраїчні решітки і ізогенії еліптичних кривих зараз активно використовуються для побудови пост-квантових криптосистем з відкритим ключем.

## 1.2. Загальний огляд еліптичних кривих

### 1.2.1. $E / \mathbb{R}$

Існує декілька форм задання еліптичних кривих. Почнемо з найбільш класичної:

Еліптичною кривою  $E$  називається множина точок, що є розв'язком рівняння:  $y^2 = x^3 + ax + b$ .

Наведене рівняння називають рівнянням Веєрштрасса і, відповідно, дану форму задання еліптичної кривої називають формою Веєрштраса. Якщо еліптична крива задана над полем  $\mathbb{F}$ , будемо позначати це як  $E / \mathbb{F}$

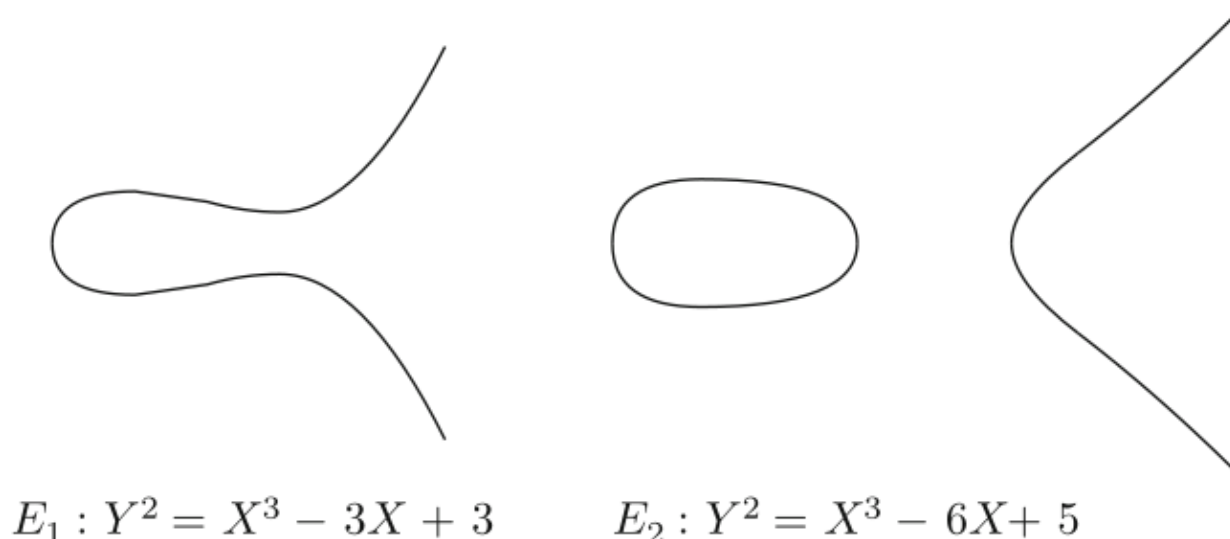


Рисунок 2: Ілюстрація 2 еліптичних кривих з [9]

Еліптичні криві є не просто об'єктом вивчення аналітичної геометрії, а й цікавим об'єктом розгляду алгебри, бо над точками еліптичної кривої можна проводити обчислення. Перед переходом до еліптичних кривих над скінченими полями, для більш інтуїтивного розуміння, розглянемо як «додавати» точки еліптичної кривої над  $\mathbb{R}$ , бо це можна легко інтерпретувати геометрично. Позначемо операцію додавання точок еліптичної кривої як  $\oplus$  і розглянемо її на прикладі з [9].

Для початку визначимо, що якщо  $A = (x, y) \in E$ , то  $-A = (x, -y) \in E$ . Нехай  $E / \mathbb{R}$  - це еліптична крива задана рівнянням  $y^2 = x^3 - 15x + 18$ ; Точки  $P = (7, 16), Q = (1, 2) \in E$ , тоді пряма  $L$ , що їх сполучає задається рівнянням:  $y = \frac{7}{3}x - \frac{1}{3}$ . Для того, щоб знайти в яких точках  $L$  перетинає  $E$  ми можемо підставити замість  $y$  рівність з  $L$  в  $E$  і розв'язати рівняння відносно  $x$ . Для наведеного прикладу маємо:

$$\left(\frac{7}{3}x - \frac{1}{3}\right)^2 = x^3 - 15x + 18 \rightarrow x^3 - \frac{49}{9}x^2 - \frac{121}{9}x + \frac{161}{9} = 0$$

Отримане рівняння є рівнянням третьої степені, отже воно має 3 корені, 2 з яких нам вже відомі, залишається знайти третій корінь. Найлегшим способом

є поділити отриманий поліном на  $(x - 7)(x - 1)$ . Таким чином отримуємо третій корінь  $x = -\frac{23}{9}$ . Підставляючи  $x$  в  $E$  отримуємо  $y = -\frac{170}{27}$ , а отже точку  $R = (-\frac{23}{9}, -\frac{170}{27})$ . Далі відображаємо  $R$  відносно ОХ:

$$P \oplus Q = -R = R' = (-\frac{23}{9}, \frac{170}{27})$$

Цей метод називається методом хорди. Наведене вище в графічно:

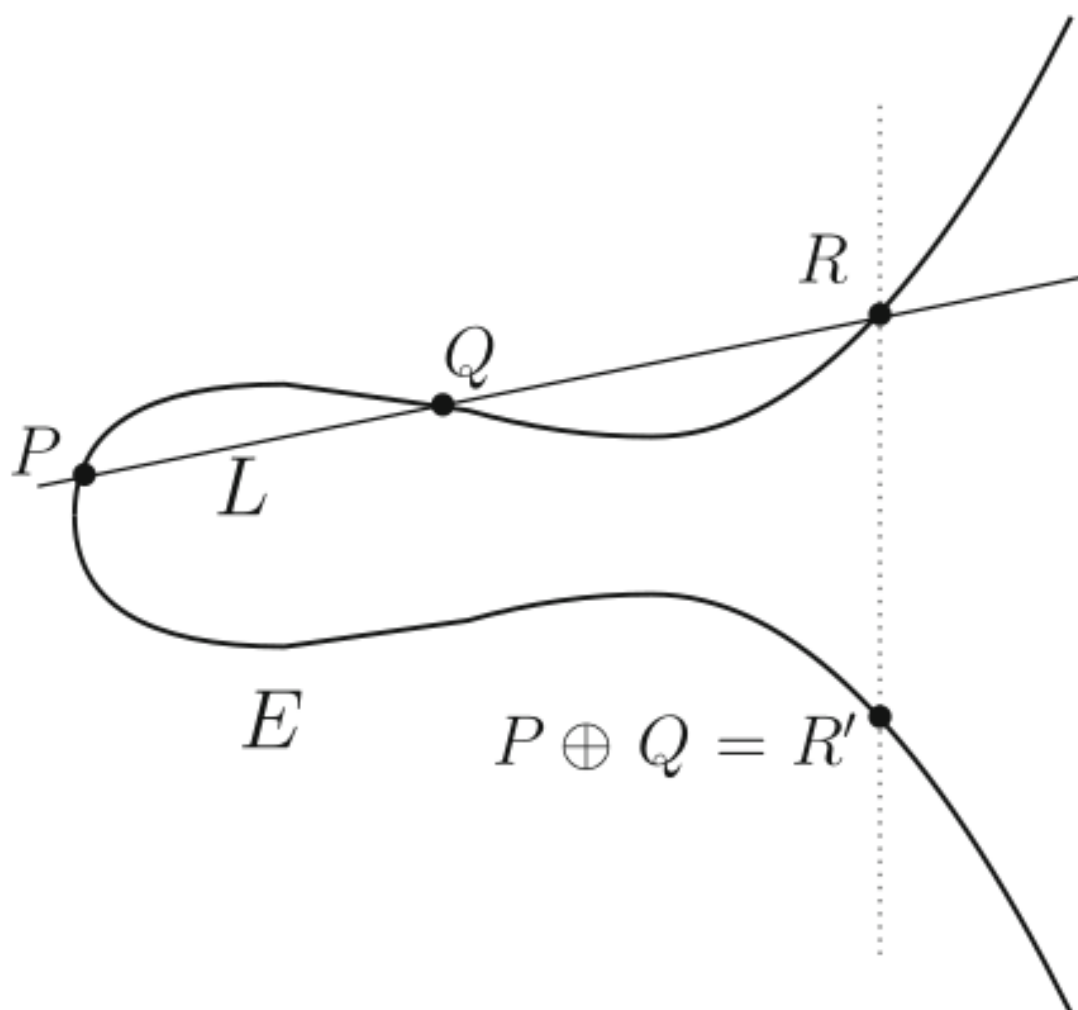


Рисунок 3:  $P \oplus Q$  з [9]

Розглянемо випадок  $P \oplus P = 2P = R'$ . Для цього проведемо дотичну до  $E$  в точці  $P$ . Для цього потрібно диференціювати  $E$  по  $x$ . На прикладі, який розглядався раніше:

$$2y \frac{dy}{dx} = 3x^2 - 15 \rightarrow \frac{dy}{dx} = \frac{3x^2 - 15}{2y}$$

Підставляючи координати  $P = (7, 16)$  отримуємо  $L : y = \frac{33}{8}x - \frac{103}{8}$ .

Аналогічно попередньому прикладу знаходимо  $R = (\frac{193}{64}, -\frac{223}{512})$ , а отже

$$P \oplus P = R' = -R = (\frac{193}{64}, \frac{223}{512})$$

Цей метод називається методом дотичної. Наведене вище в графічно:

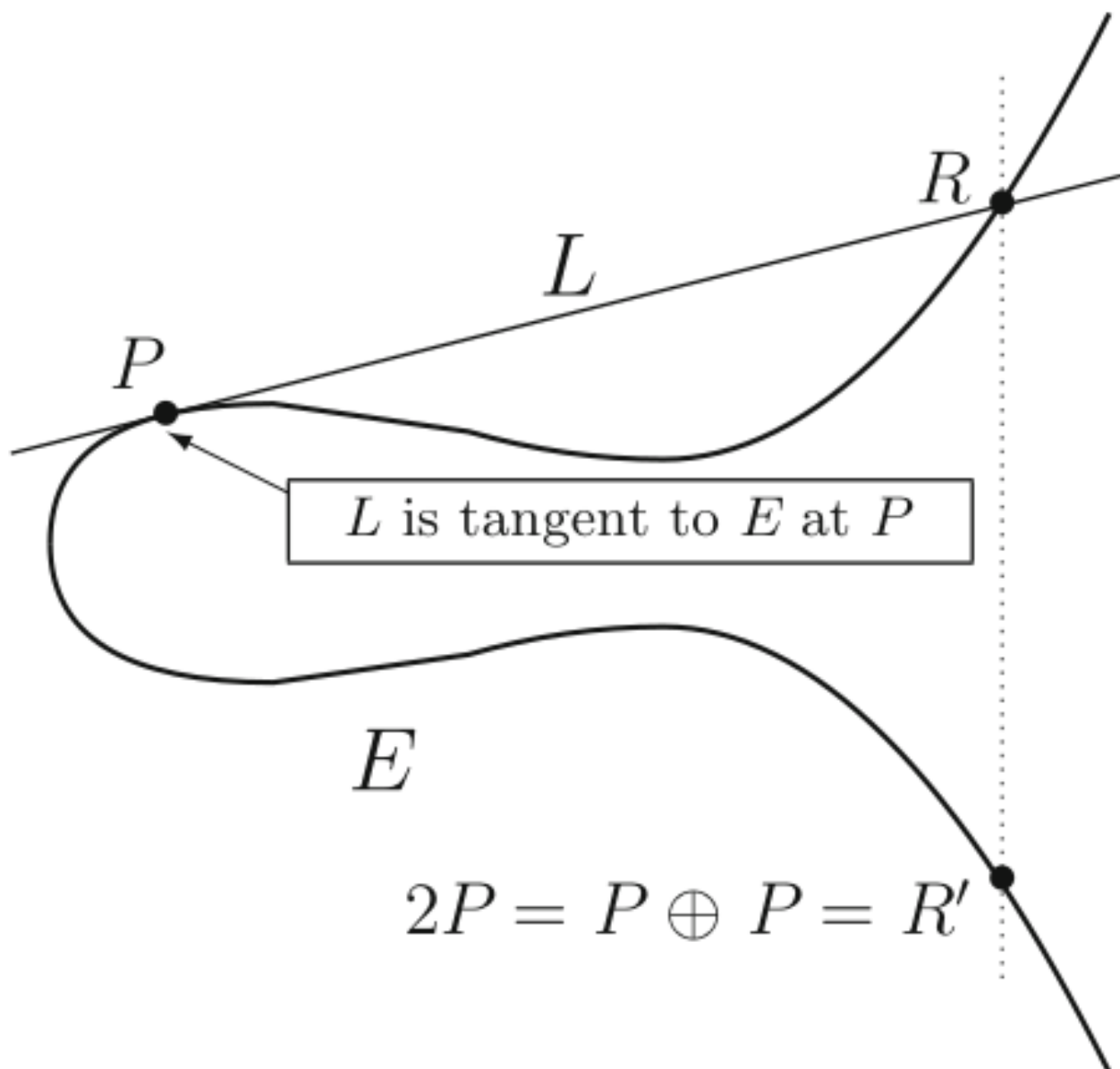


Рисунок 4:  $P \oplus P$  з [9]

Залишилось розглянути останній випадок:  $P \oplus (-P)$ . Для цього введемо спеціальну точку, що називається точкою на нескінченності:  $\mathcal{O}$ , тоді

$P \oplus (-P) = \mathcal{O}$ . Графічно це виглядає наступним чином:

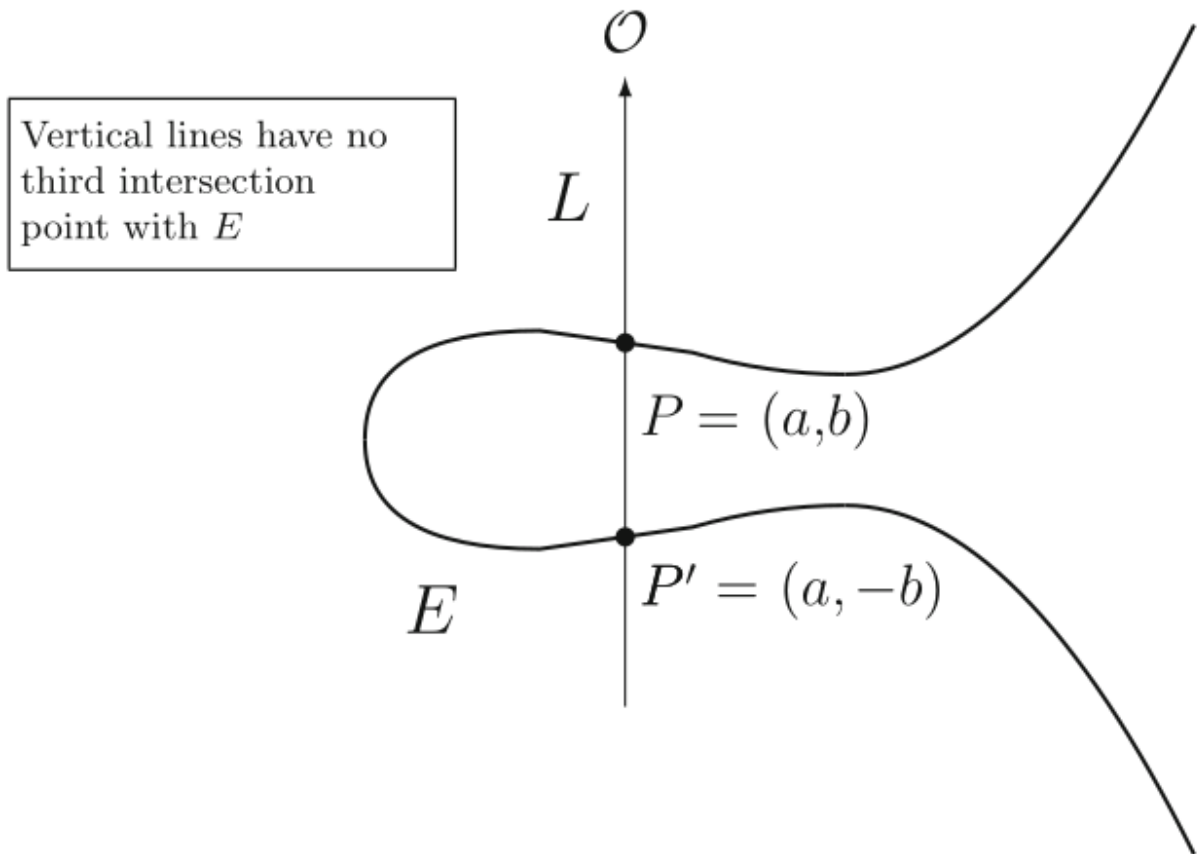


Рисунок 5:  $P \oplus (-P) = \mathcal{O}$  з [9]

Формальний кінцевий алгоритм, що покриває всі вищенаведені випадки, буде наведено у наступному підрозділі.

### 1.2.2. $E / \mathbb{F}_p$

Перейдемо до еліптичних кривих, що застосовуються у криптографії - еліптичних кривих над скінченими полями. Почнемо з визначення:

Нехай  $p > 3$  - просте число. Тоді еліптичною кривою  $E$  визначеною над  $\mathbb{F}_p$  є множина розв'язків рівняння  $y^2 = x^3 + a * x + b$ , де  $a, b \in \mathbb{F}_p$  задовільняють умову  $4a^3 + 27b^2 \neq 0$ . Ця умова потрібна, щоб впевнитись, що рівняння  $x^3 + ax + b = 0$  має тільки один корінь. У випадку, якщо попередня властивість не виконується, рівняння кривої буде мати 2, або 3 однакових кореня. Нехай у

такому випадку коренем рівняння кривої буде  $x_0$ , тоді точка  $(x_0, 0)$  називається сингулярною, і відповідна крива теж називається сингулярною.

Множиною точок еліптичною кривою  $E / \mathbb{F}_p : y^2 = x^3 + ax + b; a, b \in \mathbb{F}_p$  є  $E(\mathbb{F}_p) = \{(x, y) : x, y \in \mathbb{F}_p \wedge y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$ .

Розглянемо приклад. Нехай  $E : y^2 = x^3 + 1$  визначена на  $\mathbb{F}_{11}$ , тоді:  
 $E(\mathbb{F}_{11}) = \{\mathcal{O}, (-1, 0), (0, \pm 1), (2, \pm 3), (5, \pm 4), (7, \pm 5), (9, \pm 2)\}$

Тепер розглянемо  $\oplus$  для  $E(\mathbb{F}_p)$ . Для цього будемо спиратися на результати розгляду  $(E(\mathbb{R}), \oplus)$ , бо для  $E(\mathbb{F}_p)$  алгоритм виглядає аналогічно до алгоритму для  $E(\mathbb{R})$ , але його не вдасться легко інтерпретувати геометрично. Розглядом геометричної інтерпретації обчислень над точками еліптичних кривих над скінченими полями займається алгебраїчна геометрія і це виходить за рамки даної роботи. Нехай  $P = (x_p, y_p) \vee \mathcal{O}, Q = (x_q, y_q) \vee \mathcal{O} \in E(\mathbb{F}_p)$ :

- Якщо  $P = \mathcal{O} \rightarrow P \oplus Q = Q$
- Інакше, якщо  $Q = \mathcal{O} \rightarrow P \oplus Q = P$
- Інакше, якщо  $Q = -P \rightarrow P \oplus Q = \mathcal{O}$  (варто зазначити що це покриває випадок коли  $y_p = y_q = 0$ )
- Інакше, якщо  $P = Q$  використовуємо метод дотичної:  $\lambda = \frac{3x_p + a}{2y_p}$
- Інакше використовуємо метод хорди:  $\lambda = \frac{y_q - y_p}{x_q - x_p}$
- $x = \lambda^2 - x_p - x_q; y = \lambda(x_p - x) - y_p \rightarrow P \oplus Q = (x, y)$

$(E(\mathbb{F}_p), \oplus)$  утворюють циклічну абелеву групу.

Наведений вище алгоритм для еліптичних кривих загального вигляду має 2 основних недоліки:

1. Він не є достатньо швидким. Як буде розглянуто далі, існують спеціальні види еліптичних кривих, для яких його можна прискорити.
2. Імплементация додавання точок еліптичної кривої має працювати за константний час, інакше можлива атака по часу (timing attack) що викриває приватний ключ. Як можна побачити, даний алгоритм важко реалізувати

таким чином, щоб він працював за константиний, час через різну кількість обчислень за різних умов.

Властивість, коли алгоритм обчислення  $\oplus$  має єдину формулу для усіх точок в  $E(\mathbb{F}_p)$  при будь яких умовах, тобто може бути імплементований без умовних операторів, називається повний закон додавання (complete addition law).

У 2015 для кривих Веєрштрасса знайшли формулу для повного закону додавання [10]. Але, оскільки формула дуже об'ємна, вона тут не наводиться.

### 1.3. Додаткові обчислення в $E(\mathbb{F}_p)$

#### 1.3.1. $|E(\mathbb{F}_p)|$

Однією з найважливіших характеристик еліптичної над скінченим полем є  $q = |E(\mathbb{F}_p)|$  - розмір групи. Стосовно цієї характеристики існує теорема Гассе(Hasse):

Нехай  $E / \mathbb{F}_{p^e}$ , тоді:  $|E(\mathbb{F}_{p^e})| = p^e + 1 - t_{p^e}; t_{p^e} \leq 2\sqrt{p^e}$  - називається слідом Фробеніуса.

Важливим є здатність точно обчислити  $|E(\mathbb{F}_{p^e})|$ . Вперше ефективний алгоритм запропонував Скуф(Schoof) в [11], потім Елкіс(Elkies) і Еткін(Atkin) запропонували поліпшення складності роботи цього алгоритму [12]. Результат називається SEA, він залишається найшвидшим алгоритмом до цього часу і має складність  $O(\log^2 p * M(\log^2 p) / \log \log p)$ , де  $M(n)$  - складність множення.

Ще однією характеристикою кривої є кофактор. Нехай  $E / \mathbb{F}_p$  має підгрупу порядку  $q$ , де  $q$  - велике просте число. Зазвичай обчислення над еліптичною кривою проводяться саме в цій підгрупі. Тоді, кофактором  $h = \frac{|E(\mathbb{F}_p)|}{q}$

#### 1.3.2. Скалярний добуток точки на число

Більшість ЕСС криптосистем використовують обчислення  $nP, P \in E(\mathbb{F}_p)$ . Для того, щоб ці системи було можливо використовувати на практиці, це обчислення має бути швидким. Спочатку визначимо формально, що

$$P \in E(\mathbb{F}_p), n \geq 1 \in \mathbb{N} : nP = (n-1)P \oplus P$$



Очевидний приклад:  $3P = P + 2P = P + P + P$ . Також зрозуміло що обчислювати  $nP$  наївно, за  $n - 1$  додавань - неоптимально, тому використовують оптимізації:

**Подвоюй-і-Додавай (Double-and-Add).** За допомогою цієї оптимізації можна обчислити  $nP$  за не більше ніж  $2 \log_2 n$  додавань. Існують 2 версії цього алгоритму: яка використовує інверсію ( $P \rightarrow -P$ ) і яка не використовує. Варто відзначити, що знаходження точки еліптичної кривої оберненою до даної - тривільне і швидке, і перша версія для випадковго  $n$  потребує меншу кількість операцій у середньому. Приклад реалізації другого варіанту з використанням псевдокоду:

Double-and-Add( $P, n$ ):

```

Q := P
R := O
while n > 0:
    if n ≡ 1 (mod 2):
        R := R + Q
    Q := 2 * Q
    n := n / 2
return R

```

На практиці, якщо потрібно обчислити, наприклад,  $947P$  першим методом буде обчислено:

$$947P = P \oplus 2P \oplus (-2^4P) \oplus (-2^6P) \oplus 2^{10}P$$

А другим:

$$947P = P \oplus 2P \oplus 2^4P \oplus 2^5P \oplus 2^7P \oplus 2^8P \oplus 2^9P$$

Недоліком цього алгоритму є те, що час його роботи залежить від значення  $n$ . Це робить його вразливим до timing attack.

**Сходи Монтгомері (Montgomery ladder).** Цей алгоритм має аналогічну часову складність до алгоритму Double-and-Add, але його перевагою є те, що час його роботи залежить від кількості бітів в  $n$ , а не від самого значення, що робить його невразливим до timing-attack при умові якщо  $\oplus$  також не вразлива

до неї. Сам алгоритм [13], нехай  $n = (n_{t-1}, \dots, n_0)_2$ , кількість бітів в  $n = t$  позначимо  $n[i] = n_i$ :

MontgomeryLadder( $P, n$ ):

```

R :=  $\mathcal{O}$ 
Q := P
for i = t - 1 downto 0:
    if n[i] = 0:
        Q := Q + R
        R := R + R
    else:
        R := R + Q
        Q := Q + Q
return R

```

Нехай  $x(P), P \in E(\mathbb{F}_p)$  -  $x$  координата точки  $P$ . Також, модифікація наведеної імплементації цього алгоритму дозволяє обчислити  $x(nP)$  знаючи тільки  $x(P)$ , використовуючи наступні формули:

$$\text{Якщо } (2\alpha)P \neq \mathcal{O} : x_{2\alpha} = \frac{1}{4} \left( (x_\alpha^2 - a)^2 - 8bx_\alpha \right) / (x_\alpha^3 + ax_\alpha + b)$$

$$\text{Якщо } (2\alpha + 1)P \neq \mathcal{O} : x_{2\alpha+1} = ((a - x_\alpha x_{\alpha+1}) - 4b(x_\alpha + x_{\alpha+1})) / (x_1(x_\alpha - x_{\alpha+1}^2))$$

Як висновок, можна резюмувати, що швидкість добутку точки на число є важливою характеристикою еліптичної кривої. Існують криві, для яких можна оптимізувати як одиничну операцію  $P + Q$ , так і обчислення  $nP$  в цілому.

### 1.3.3. Дискретний логарифм в $E(\mathbb{F}_p)$

Нехай  $E$  - еліптична крива,  $\mathbb{F}_p$  - скінчене поле,  $P \in E(\mathbb{F}_p), Q \in \langle P \rangle$ , тоді

дискретним логарифмом  $Q$  за основою  $P$  є таке  $n$ , що  $Q = nP$  і позначається  $n = \log_P Q$ .

Нехай  $q$  - порядок  $P$ , тоді очевидно що, якщо  $Q = n_0 P$ , то

$Q = (n_0 + iq)P, \forall i \in \mathbb{N}$ . Для визначеності будемо вважати, що дискретний логарифм - це найменше таке  $n$ , що  $Q = nP$ . У такому випадку зрозуміло, що  $\log_P(Q) \in \mathbb{Z}_q$ .

Важливо відмітити, що для дискретного логарифму точок еліптичної кривої виконується властивість звичайного логарифму:

$$\log_P(Q_1 + Q_2) = \log_P(Q_1) + \log_P(Q_2), \forall Q_1, Q_2 \in \langle P \rangle$$

З цього випливає, що  $\log_P$  визначає груповий гомоморфізм:

$$\log_P : \langle P \rangle \rightarrow \mathbb{Z}_q$$

Зазвичай на практиці  $P$  вибирається з генераторів  $E(\mathbb{F}_p)$ .

ECDLP - проблема знаходження дискретного логарифму для точок еліптичної кривої.

#### 1.4. Спарювання точок еліптичної кривої(Pairing)

$E / \mathbb{F}_p$  має додаткову структуру - pairing. Вона може бути використана для побудови криптосистем неможливих для циклічних груп у загальному випадку, а також для  $\mathbb{Z}_p^*$  і дозволяє розширити множину криптосистем ECC. Зазвичай у визначенні використовують мультиплікативні групи, замість адитивних, але оскільки ця робота сконцентрована саме на еліптичних кривих, то:

Нехай  $\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_T$  - циклічні групи порядку  $q$ , де  $q$  - просте.

$g_0 \in \mathbb{G}_0, g_1 \in \mathbb{G}_1$  - генератори. Pairing - це ефективно обчислювальна функція  $e : \mathbb{G}_0 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ , яка задовільняє наступні вимоги:

1. Білінеарність.  $\forall u, u' \in \mathbb{G}_0, \forall v, v' \in \mathbb{G}_1$ :

$$e(u + u', v) = e(u, v)e(u', v) \wedge e(u, v + v') = e(u, v)e(u, v')$$

2. Недегенеративність:  $g_T = e(g_0, g_1)$  - генератор в  $\mathbb{G}_T$

Коли  $\mathbb{G}_0 = \mathbb{G}_1$  назвемо це спарювання симетричним, інакше - асиметричним.

$\mathbb{G}_0, \mathbb{G}_1$  називаються pairing groups(спарювані групи), а  $\mathbb{G}_T$  - target group(цільовою групою).

З білінеарності випливає основна властивість спарювання, яка використовується в побудові криптосистем:

$$\forall \alpha, \beta \in \mathbb{Z}_q : e(g_0^\alpha, g_1^\beta) = e(g_0, g_1)^{\alpha\beta} = e(g_0^\beta, g_1^\alpha)$$

Вимога недегенеративності потрібна, щоб  $e$  не повертала завжди  $1 \in \mathbb{G}_T$  для всіх аргументів.

Найбільш зручним і ефективним спарюванням для  $E / \mathbb{F}_p$  - є асиметричне спарювання. Спарювання сконструйоване з  $E / \mathbb{F}_p$  має групи  $\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_T$  з наступними властивостями:

- $\mathbb{G}_0$  - це підгрупа  $E(\mathbb{F}_p)$  порядку  $q$ , де  $q$  - просте.
- $\mathbb{G}_1$  - це підгрупа  $E(\mathbb{F}_{p^d})$  порядку  $q$ , для деякого  $d \in \mathbb{N} > 0$ , де  $\mathbb{G}_0 \cap \mathbb{G}_1 = \{\mathcal{O}\}$
- $\mathbb{G}_T$  - це мультиплікативна група

Число  $d \in \mathbb{N}$  називається embedding degree (ступіню вкладення) кривої.

Зрозуміло, для того що спарювання було ефективним  $d$  має бути малим, тому що, якщо  $d$  буде великим неможливо буде ефективно записати елементи в  $\mathbb{G}_1$  та  $\mathbb{G}_T$ . Еліптичні криві, де спарювання мале, наприклад  $d \leq 16$ , називають pairing friendly elliptic curves (еліптичними кривими дружніми для спарювання).

Оригінально почали використовувати Спарювання Вейля (Weil pairing). Нехай  $m \in \mathbb{N}, P \in E : mP = \mathcal{O}$ . Точка  $P$  є точкою скінченного порядку на еліптичній кривій, або торсіонною точкою (torsion point). Більш точно кажуть, що  $P$  є точкою порядку  $m$ . Позначимо множину точок порядку  $m$  як:

$$E(\mathbb{F}_p)[m] = \{P \in E(\mathbb{F}_p) : mP = \mathcal{O}\}$$

Тоді, нехай  $f_P, f_Q$  - раціональні функції над  $E$  (не наводжу тут визначення, ознайомитися з цією концепцією можна в [9], [14]), такі що:

$\text{div}(f_P) = m[P] - m[\mathcal{O}], \text{div}(f_Q) = m[Q] - m[\mathcal{O}]$ , тоді Weil pairing це:

$$e_m(P, Q) = \frac{f_P(Q+S)}{f_P(S)} / \frac{f_P(P-S)}{f_P(-S)}, \text{ де } S \in E, S \notin \{\mathcal{O}, P, -Q, P-Q\}$$

Для побудови спарювання Вейля існує ефективний алгоритм Міллера [15] (Miller's algorithm). Але на практиці зараз використовують спарювання Тейта (Tate pairing) та спарювання Ейта (Ate pairing), оскільки алгоритм Міллера для них ефективніший.

## 1.5. Спеціальні види еліптичних кривих

### 1.5.1. Крива Монтгомері

Еліптична крива  $E / \mathbb{F}_p$  у формі Монтгомері у змінних  $u, v$  має вигляд:

$$Bv^2 = u^3 + Au^2 + u; A, B \in \mathbb{F}_p, B(A^2 - 4) \neq 0$$

Це рівняння кривої може бути легко приведено до рівняння Веєрштрасса заміною змінних:  $u = Bx - \frac{A}{3}; v = By$ . Цікавою особливістю кривої Монтгомері є те, що  $|E(\mathbb{F}_p)|$  завжди ділиться на 4. З цього також випливає, що не кожену криву у формі Веєрштрасса можна привести до кривої у формі Монтгомері заміною змінних.

Криві Монтгомері дозволяють пришвидшити операцію додавання, шляхом оптимізації алгоритма Монтгомері. Нехай  $X_1 = x(P), Z_1 = 1$ , і:

$$X_{2\alpha} = (X_\alpha^2 - Z_\alpha^2)^2$$

$$Z_{2\alpha} = 4X_\alpha Z_\alpha (X_\alpha^2 + AX_\alpha Z_\alpha + Z_\alpha^2)$$

$$X_{2\alpha+1} = 4Z_1 (X_\alpha X_{\alpha+1} - Z_\alpha Z_{\alpha+1})^2$$

$$Z_{2\alpha+1} = 4X_1 (X_\alpha Z_{\alpha+1} - Z_\alpha X_{\alpha+1})^2$$

$$\text{Тоді } x(\alpha P) = X_\alpha / Z_\alpha, \text{ якщо } \alpha P \neq \mathcal{O}$$

Варто зазначити, що чим менше  $A$  - тим швидше буде обчислення.

### 1.5.2. Крива Едвардса

Еліптична крива  $E / \mathbb{F}_p$  формі Едвардса має вигляд:

$$x^2 + y^2 = 1 + dx^2y^2, d \neq 0, 1 \in \mathbb{F}_p$$

Аналогічно кривій Монтгомері, крива Едвардса може бути приведена до форми Веєрштрасса заміною змінних, але не навпаки.  $|E(\mathbb{F}_p)|$  завжди ділиться на 4. Перевагою кривих Едвардса є те, що  $\oplus$  дуже просто імплементувати:

$$\forall P, Q \in E(\mathbb{F}_p) : P \oplus Q = \left( \frac{x_1y_2 + x_2y_1}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - x_1x_2}{1 - dx_1x_2y_1y_2} \right)$$

Для цього визначемо  $\mathcal{O} = (0, 1)$ . Як можна побачити криві Едвардса мають повний закон додавання.

### 1.5.3. Крива Кобліца

Оригінально крива була запропонована Кобліцом над  $\mathbb{F}_2$  і мала вигляд:

$$E : y^2 + xy = x^3 + ax^2 + 1, a \in \{0, 1\}$$

Після цього почалося дослідження цієї кривої над  $\mathbb{F}_{2^m}$ , для цього поля вона має вигляд:

$$y^2 + xy = x^3 + ax^2 + b, b \neq 0$$

Але, оскільки на практиці частіше застосовують криві цього виду над  $\mathbb{F}_p$ , і дана робота зосереджена на  $E / \mathbb{F}_p$ , сконцентруємося саме на кривих Кобліца над  $\mathbb{F}_p$ . Крива Кобліца  $E$  над  $\mathbb{F}_p$  задається рівнянням [16] :

$$E : y^2 = x^3 + b, b \neq 0, p \equiv 1 \pmod{3}$$

В більш широкому розумінні кривими Кобліца називають еліптичні криві, для яких існує ефективно обчислювальний нетривіальний ендоморфізм  $\phi : \forall P, Q : \phi(P + Q) = \phi(P) + \phi(Q)$ , який дозволяє оптимізувати операцію множення точки на число. Для  $E(\mathbb{F}_{2^m})$  використовують ендоморфізм Фробеніуса:  $\phi : \mathbb{F}_{p^k} \rightarrow \mathbb{F}_{p^k}, \alpha \rightarrow \alpha^p$ . Застосовуючи його для  $E / \mathbb{F}_{2^m}$  отримуємо  $P = (x, y) \in E(\mathbb{F}_{2^m}), \phi(x, y) = (x^2, y^2)$ . Для  $E / \mathbb{F}_p$  використовують ендоморфізм  $\phi(x, y) = (\omega x, y), 1 \neq \omega \in \mathbb{F}_p, \omega^3 = 1$ . Зрозуміло, що таке не одиничне  $\omega$  знайдеться, оскільки  $p \equiv 1 \pmod{3}$ , що означає, що 1 має 3 кубічних кореня:  $1, u, u^2 (u \in \mathbb{F}_p)$ .

Легко перевірити, що якщо  $(x, y) \in E(\mathbb{F}_p)$ , то і  $\phi(x, y) = (\omega x, y) \in E(\mathbb{F}_p)$ , оскільки  $y^2 = x^3 + b = (\omega x)^3 + b = \omega^3 x^3 + b = x^3 + b$ . Також додатково визначемо, що  $\phi(\mathcal{O}) = \mathcal{O}$ . Тобто доведено, що  $\phi : E(\mathbb{F}_p) \rightarrow E(\mathbb{F}_p)$ .

Нехай  $q = |E(\mathbb{F}_p)|$ . Оскільки,  $\phi$  є ендоморфізмом і  $E(\mathbb{F}_p)$  - циклічна група, має існувати таке  $\lambda \in \mathbb{Z}_q$ , що:  $\forall P \in E(\mathbb{F}_p) : \phi(P) = \lambda P$ . Можливо визначити  $\lambda$  наступним чином. Очевидно що  $P, \phi(P), \phi^2(P)$ , мають однакову  $y$

координату, а отже знаходяться на одній прямій, до того ж горизонтальній. З геометричної інтерпретації  $\oplus$  відомо, що це означає, що їх сума  $\mathcal{O}$ , отже:

$$\forall P \in E(\mathbb{F}_p) : \mathcal{O} = P \oplus \phi(P) \oplus \phi^2(P) = P \oplus \lambda P \oplus \lambda^2 P = (1 + \lambda + \lambda^2)P$$

Звідки можемо зробити висновок, що

$$1 + \lambda + \lambda^2 = 0 \rightarrow (\lambda - 1)(1^2 + 1 * \lambda + \lambda^2) = 0 * (1 - \lambda) \rightarrow \lambda^3 - 1 = 0 \rightarrow \lambda^3 = 1$$

Тобто  $\lambda$ , аналогічно до  $\omega$ , це один з двох нетривіальних кубічних коренів 1.

Тепер розглянемо як  $\phi$  може допомогти в прискоренні обчислення  $nP$ .

Нехай  $n \in \mathbb{Z}_q$  і мета обчислити  $nP$  для деякого  $P \in E(\mathbb{F}_p)$ . Для більшості таких  $n$  можливо знайти такі  $\tau_0, \tau_1, \tau_2 \in \mathbb{Z}$ , що:

$$n = \tau_0 + \tau_1 \lambda + \tau_2 \lambda^2 \wedge |\tau_i| \leq 2q^{\frac{1}{3}} \text{ для } i = 0, 1, 2$$

$$\text{Тобто, } nP = \tau_0 P + \tau_1 \phi(P) + \tau_2 \phi^2(P)$$

Така оптимізація дозволяє прискорити операцію множення у 2-3 рази [7] в порівнянні з Double-and-Add.

## 1.6. Припущення на яких будується ЕСС

### 1.6.1. ECDLP

Основне припущення ЕСС полягає у тому, що ECDLP для точок еліптичної кривої над скінченим полем є складнообчислювальною проблемою, тобто не існує ефективного алгоритму обчислення дискретного логарифму.

На ECDLP існує багато атак - алгоритмів знаходження дискретного логарифма. Ним присвячена ціла глава в [14]. Нехай  $P, Q \in E(\mathbb{F}_p)$ ,  $|E(\mathbb{F}_p)| = q$ . Перша група атак працює для будь якої циклічної групи:

- Baby Step, Giant Step. Має часову складність  $O(\sqrt{q})$  і потребує  $O(\sqrt{q})$  місця.

Цей алгоритм є детерміністичним

- Алгоритми  $\rho$  і  $\lambda$  Полларда.  $\rho$  є узагальненням  $\lambda$  алгоритму. Вони також мають часову складність  $O(\sqrt{q})$ , але потребують тільки  $O(1)$  додаткового місця.

Також треба зазначити, що цей алгоритм є ймовірнісним, тобто він знаходить

дискретний логарифм з якоюсь високою, не 100%, ймовірністю, якщо логарифм існує.

- Метод Поліга-Геллмана [17] (The Pohlig-Hellman method). Ефективний якщо  $q$  не є простим і в його факторизації нема великих простих чисел. Нехай  $q = \prod_i p_i^{e_i}$ , тоді складність часова складність цього алгоритму  $O\left(\sum_i e_i (\log q + \sqrt{p_i})\right)$

Також існує група атак, що використовують pairing. Такими атаками є:

- MOV(Menezes, Okamoto and Vanstone) [18]. Ця атака використовує спарювання Вейля щоб звести ECDLP в  $E(\mathbb{F}_p)$  до DLP в  $\mathbb{F}_{p^m}$
- Атака Фрея-Рюка [19] (The Frey-Rück Attack). Аналогічно MOV, тільки використовується більш ефективно спарювання Тейта.

Оскільки ЕСС ґрунтується на припущенні про обчислювальну складність ECDLP і ЕСС активно використовується в реальних криптосистемах, можна припустити, що для будь якої кривої  $E / \mathbb{F}_p$ , де  $p$  - достатньо велике просте число - ECDLP складнообчислювальна проблема. Це є хибним твердженням, у тому числі через наведені вище атаки, наприклад:

- Якщо  $|E(\mathbb{F}_p)|$  не є простим і всі прості множники у його розкладі менші деякого  $q_{\max}$ , то тоді можливо застувати атаку з використанням метода Поліга-Геллмана. Зокрема, якщо  $q_{\max} < 2^{80}$ , то ECDLP не є складнообчислювальною проблемою на практиці. Тому в реальних системах використовують криві  $E / \mathbb{F}_p$  для яких  $|E(\mathbb{F}_p)|$  рівне  $q$ ,  $4q$ , або  $8q$  для деякого простого  $q$ .
- Якщо  $|E(\mathbb{F}_p)| = p$ , то ECDLP для  $E(\mathbb{F}_p)$  розв'язний за поліноміальний час. Такі криві називаються аномальними і не використовуються на практиці
- Нехай існує таке мале  $\tau > 0 \in \mathbb{N}$ , що  $|E(\mathbb{F}_p)|$  ділить  $p^\tau - 1$ . Тоді ECDLP в  $E(\mathbb{F}_p)$  можна звести до DLP в  $\mathbb{F}_{p^\tau}$  використовуючи MOV, або атаку Фрея-Рюка, що в свою чергу дозволить використати GNFS. На практиці, якщо  $p$  - 256 бітне число і  $\tau = 2$ , то розв'язок ECDLP займе приблизно пару годин. Для захисту



від цієї атаки, треба переконатися що  $p^\tau$  достатньо велике для того щоб GNFS в  $\mathbb{F}_{p^\tau}$  був нездійсненний.

Але, якщо параметри кривої підбрано правильно, то найоптимальніший алгоритм вирішення ECDLP має часову складність  $O(\sqrt{q})$ , де  $q = |E(\mathbb{F}_p)|$ .

### 1.6.2. CDH

Нехай  $\mathbb{G} = \langle g \rangle$  - циклічна група порядку  $q$ ,  $a, b \in \mathbb{Z}_q$ . Припущення полягає у тому, що якщо зломисник перехопив  $g, g^a, g^b$  він не зможе обчислити  $g^{ab}$ .

Зараз єдиним відомим способом розв'язати CDH є розв'язати DLP, але не доведено, що DLP і CDH - це рівнозначні проблеми. CDH є сильнішим припущенням, ніж DLP. Окрім цього цікавою особливістю CDH є те, що на відміну від RSA, не існує ефективного способу перевірити по заданому  $g^c$ , чи дійсно  $g^c = g^{ab}$  маючи  $g, g^a, g^b$ .

### 1.6.3. DDH

Нехай  $\mathbb{G} = \langle g \rangle$  - циклічна група порядку  $q$ ,  $a, b, c \in \mathbb{Z}_q$  обрані випадково.

Припущення полягає у тому, що зломисник не може знаючи  $g, q$  відрізнити  $(g^a, g^b, g^c)$  від  $(g^a, g^b, g^{ab})$ . Це припущення є сильнішим ніж CDH.

## 1.7. Скручені криві (Twist curves)

Кожна еліптична крива  $E / \mathbb{F}_p$  має пов'язану з нею криву  $\tilde{E} / \mathbb{F}_p$ , яка називається twist of E (скрученою кривою E). Нехай  $c \in \mathbb{F}_p$  не є квадратичним лишком в  $\mathbb{F}_p$ . Якщо E задається рівнянням  $y^2 = x^3 + ax + b$ , тоді  $\tilde{E}$  задається рівнянням  $cy^2 = x^3 + ax + b$ . Нескладно показати, що  $|E(\mathbb{F}_p)| + |\tilde{E}(\mathbb{F}_p)| = 2p + 2$ , з цього, з використанням теореми Гассе, випливає що  $|\tilde{E}(\mathbb{F}_p)| = p + 1 - t$ .

Визначемо, що крива  $E / \mathbb{F}_p$  twist secure (безпечно скручена), якщо ECDLP складно розв'язний як в  $E / \mathbb{F}_p$ , так і в  $\tilde{E} / \mathbb{F}_p$ .

Розглянемо для чого потрібна twist security. Нехай Боб має секретний ключ  $\alpha \in \mathbb{Z}_q$ , а надіслане  $P \in E(\mathbb{F}_p)$  Боб відправляє  $\alpha P$ . Для того, щоб система була безпечною Боб має перевіряти чи  $P \in E(\mathbb{F}_p)$ , але нехай замість того щоб

відсилати  $P = (x, y)$ , відсилається тільки  $x(P)$  координата, тоді також можна як і отримати  $y$ , так і перевірити чи  $\in E(\mathbb{F}_p)$ . Тоді, перевірка  $P \in E(\mathbb{F}_p)$  зводиться до перевірки чи  $y^2 = x^3 + ax + b$  квадратичним лишком в  $\mathbb{F}_p$ , що є відносно дорогою по часу операцією. Нехай, у такому випадку при імплементації крок перевірки був пропущений. Тоді, зловмисник може надіслати Бобу  $x_1 = x(\tilde{P}) \in \mathbb{F}_q, \tilde{P} \in \tilde{E} / \mathbb{F}_q$ . Якщо ECDLP легко розв'язний в  $\tilde{E} / \mathbb{F}_q$ , тобто  $E / \mathbb{F}_p$  не twist secure, то відповідь Боба розкриє його секретний ключ. У випадку, якщо  $E / \mathbb{F}_p$  twist secure, то Бобу не загрожує така атака і він може використовувати оптимізацію при якій відправляється тільки  $x$  координата і не витрачати час на її перевірку.

### 1.8. Застосування еліптичних кривих поза межами ЕСС

Еліптичні криві можна застосувати у криптографії поза межами ЕСС:

1. Для факторизації, наприклад алгоритм факторизації Ленстра [20]. Він має часову складність  $O\left(\exp\left(\left(\sqrt{2} + o(1)\right)\sqrt{\log p \log \log p}\right)(\log N^2)\right)$  [21]
2. Для тестування на простоту. Існують, наприклад, тести на простоту Еткіна-Морея (Atkin–Morain elliptic curve primality test) зі складністю  $O(\log^{6+\varepsilon} N)$ , для деякого  $\varepsilon > 0$  і алгоритм Голдвошера-Кілана (Goldwasser–Kilian algorithm) зі складністю  $O(\log^{10+c_2} N)$ , де  $c_2$  деяке число більше 0.

Найбільше число, простоту якого було доведено використовуючи еліптичні криві, має 86453 знаків [22].

### 1.9. Висновки

У цьому розділі було розглянуто і досліджено теоретичний фундамент ЕСС, а саме: еліптичні криві над  $\mathbb{R}$ , еліптичні криві над полями скінченної характеристики  $p$ , обчислення над точками еліптичних кривих і їх геометрична інтерпретація для еліптичних кривих над  $\mathbb{R}$ , обчислення кількості точок еліптичної кривої, способи обчислення і оптимізації скалярного добутку точки на ціле число, спарювання точок еліптичної кривої, спеціальні випадки еліптичних кривих (Монгомері, Едвардса, Кобліца), проблему знаходження

дискретного логарифма для точок еліптичної кривої і можливі атаки на неї, припущення на яких будується криптографія еліптичних кривих, скручування еліптичних кривих і застосування еліптичних кривих поза межами ЕСС. Як висновок, з цього розділу має бути очевидно, що розробка своєї еліптичної кривої - це складна справа з великою кількістю нюансів, до того ж, неправильно побудована еліптична крива може призвести до того, що криптосистеми з її використанням не будуть безпечними. Тому на практиці використовують набір затверджених, або просто перевірених кривих. Деякі з них будуть розглянуті у наступному розділі.

## РОЗДІЛ 2. ЗАСТОСУВАННЯ У КРИПТОГРАФІЇ

В цьому розділі розглянуто еліптичні криві які використовуються у сучасній криптографії і побудовані на них криптосистеми. Також описується побудова криптосистеми, що використовує еліптичні криві, з нуля, наводяться деякі бенчмарки. Для того, щоб бенчмарки мали більше сенсу, опис оточення в якому вони проводились: процесор Ryzen7 7700X(8 ядер, 16 потоків, частота 4.5Гц - 5.4ГГц), 32Гб оперативної пам'яті DDR5 частотою 6000МГц, операційна система Fedora 37 з версією ядра Linux 6.2.15, версія OpenSSL 3.0.8, версія libsodium 2.3.

### 2.1. Кодування інформації за допомогою еліптичних кривих

В ЕСС публічний ключ, шифротекст і цифровий підпис - це елементи  $E(\mathbb{F}_p)$ , приватний ключ - це елемент  $\mathbb{Z}_q$ .

Тепер розглянемо яким чином кодуються/декоднуються точки еліптичної кривої, найрозповсюдженішим стандартом є SEC1 [23]. Нехай, у нас є закодована точка  $P \in E(\mathbb{F}_p)$  в послідовність байт:

- Якщо перший байт 0x00, це означає, що  $P = \mathcal{O}$ .
- Інакше, якщо перший байт 0x02 або 0x03, це означає, що точка закодована у стисненому форматі (compressed form)  $(0x02|0x03)||x$ , де  $x$  - це  $x(P)$ . Знаючи  $x$  можна обчислити  $y^2$  і знайти з нього 2 кореня. 0x02 означає, що треба взяти парний корінь, 0x03 - непарний.
- Інакше, якщо перший байт 0x04, це означає, що точка закодована у форматі  $0x04||x||y$  (uncompressed form).
- Інакше цей байтовий рядок відкидається.

### 2.2. Приклади еліптичних кривих у сучасній криптографії

Розглянемо одні з найрозповсюдженіших кривих, що використовуються у сучасній криптографії та надають 128 бітний рівень безпеки.

### 2.2.1. secp256r1 (P256)

Ця крива була затверджена NIST в стандарті SEC2 [24] у 1999 році. Це крива є єдиною обов'язковою кривою TLS 1.3.  $r$  у назві означає random(випадкова).

secp256r1 визначена наступним чином:

- Крива визначена над  $p_r = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$ . Така структура  $p_r$  призначена пришвидшити арифметику по модулю  $p_r$
- Крива має рівняння у формі Веєрштрасса  $y^2 = x^3 - 3x + b$ , де  $b \in \mathbb{F}_{p_r}$  можна записати у шістнадцятиричному вигляді як  

$$b = 5ac635d8 \text{ aa3a93e7 b3ebbd55 769886bc 651d06b0 cc53b0f6 3bce3c3e 27d2604b}$$
- $|E(\mathbb{F}_p)|$  - просте число
- Стандарт також специфікує генератор  $g$ , який має використовуватись.

Стандарт стверджує, що  $b$  було згенеровано публічним детерміністичним генератором випадкових чисел, в якому як seed використовували деяке  $S$ , значення якого не вказується. Існує нічим не підтверджена теорія, що  $b$  було обрано таким чином, щоб існувала деяка невідома загально атака на ECDLP для цієї кривої, про яку знає тільки уряд США.

### 2.2.2. secp256k1 (Bitcoin curve)

Ця крива також була затверджена в SEC2. Вона широко використовується в блокчейнах, наприклад, ця крива використовується в блокчейнах Bitcoin і Ethereum. secp256r1 визначена наступним чином:

- Крива визначена над  $p_r = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$ . Аналогічно secp256r1 така структура  $p_r$  призначена пришвидшити арифметику по модулю  $p_r$
- Крива має рівняння у формі Веєрштрасса  $y^2 = x^3 + 7$ . Як нескладно помітити - це крива є кривою Кобліца, що означає, що для неї можна прискорити операцію множення точки на число.  $k$  у назві означає Koblitz.
- $|E(\mathbb{F}_p)|$  - просте число

- Стандарт також специфікує генератор  $g$ , який має використовуватись.

Як можна зазначити, у цієї кривої немає випадкового параметра.

### 2.2.3. Curve25519

Ця крива, на відміну від попередніх не сертифікована NIST. Curve25519 [25]

визначена наступним чином:

- Крива визначена над  $p = 2^{255} - 19$ , звідки і походить її назва.  $p$  - це найбільше просте число менше  $2^{255}$ , що дозволяє прискорити арифметику в  $\mathbb{F}_p$
- Крива має рівняння у формі Монгомері  $y^2 = x^3 + 486662 * x^2 + x$ . Також цю криву можна задати у формі Едвардса  $x^2 + y^2 = 1 + \left(\frac{121665}{121666}\right)x^2y^2$ . Тобто для цієї кривої працюють як і оптимізації для кривих Едвардса, так і оптимізації для кривих Монгомері. А обрано найменшим з можливих щоб прискорити алгоритм сходів Монгомері, вибір гарно обгрунтовано в [25].
- Генератором є точка  $P = (x, y)$  де  $x = 9$
- $|E(\mathbb{F}_p)| = 8q$ , де  $q$  просте число. Тобто кофактор даної кривої 8.

Також це крива є скручено безпечною (twist secure). Варто зазначити, що всі константи в цій кривій чітко обгрунтовані.

### 2.2.4. bn256

Якщо минулі криві були розроблені для криптосистем на основі ECDLP, то ця крива була розроблена для криптосистем на основі спарювання. Ступінь вкладення(embedding degree)  $d = 12$ . Додаткову інформацію про цю криву і побудову криптосистем на її основі можна знайти в [26].

## 2.3. Протоколи узгодження ключів

### 2.3.1. ECDH

ECDH(Elliptic curve Diffie-Hellman) - протокол Діффі-Геллмана для еліптичних кривих. Цей протокол використовується для узгодження спільного секрету, на практиці цим спільним секретом є симетричний ключ. Розглянемо сам протокол.

1. Аліса і Боб узгоджують криву  $E / \mathbb{F}_p$  в якій ECDLP складнообчислювальна проблема і точку  $P \in E(\mathbb{F}_p)$ .  $P$  має бути або генератором  $E(\mathbb{F}_p)$ , або підгрупи степені  $q$ , де  $q$  - велике просте число.
2. Аліса і Боб обирають секретні числа  $n_A, n_B \in \mathbb{Z}_q$  відповідно, не розголошуючи їх.
3. Аліса обчислює  $Q_A = n_A P$ , Боб обчислює  $Q_B = n_B P$
4. Аліса відсилає  $Q_A$  Бобу, а Боб відсилає  $Q_B$  Алісі.
5. Аліса обчислює  $n_A Q_B = n_A n_B P$ . Боб обчислює  $n_B Q_A = n_A n_B P$
6.  $n_A n_B P$

ECDH виходить з припущення CDH. Варто зазначити, що на практиці точки  $Q_A$  і  $Q_B$  відсилають у стисненому форматі, оскільки еліптична крива над  $n$  бітним простим  $p$  має рівень захисту  $n / 2$  бітного симетричного ключа. Тобто маючи 256 бітну криву можливо домовитись максимум про 128 бітний спільний симетричний ключ. Також на практиці 1 крок протоколу пропускається, оскільки він зафіксований у конкретному стандарті. ECDH, що використовує P256, часто позначають ECDH/P256, а ECDH, що використовує Curve25519 - X25519. Порівняємо швидкодію ECDH/P256, X25519 і класичного DH над  $\mathbb{Z}_p^*$ . Візьмемо DH-2048 щоб він погоджував аналогічно до ECDH/P256 і X25519 128 бітний симетричний ключ. Заміри проводилися за допомогою OpenSSL [27] - найрозповсюдженішої бібліотеки, що імплементує TLS протокол.

```
~ ) openssl speed ffdh2048
Doing 2048 bits ffdh's for 10s: 53737 2048-bits FFDH ops in 9.99s
```

Рисунок 6: Швидкість роботи DH-2048

```
~ ) openssl speed ecdhp256
Doing 256 bits ecdh's for 10s: 332204 256-bits ECDH ops in 9.99s
```

Рисунок 7: Швидкість роботи ECDH/P-256

```
~ > openssl speed ecdhx25519
Doing 253 bits ecdh's for 10s: 472399 253-bits ECDH ops in 9.99s
```

Рисунок 8: Швидкість роботи X25519

	DP-2048	ECDH/P-256	X25519
Абсолютна кількість погоджень/с	53737	332204	472399
Відносна кількість погоджень/с	1	6.2	8.8

Таблиця 1: Порівняльна таблиця швидкодій стандартизацій протоколу Діффі-Геллмана

Як можна побачити з (Таблиця 1) стандарти, що використовують еліптичні криві в протоколі Діффі-Геллмана працюють набагато швидше стандартів, що використовують  $\mathbb{Z}_q$ . Цей бенчмарк показав приріст швидкості роботи у 6 разів. Варто відзначити швидкість роботи X25519, що побудований на Curve25519.

#### 2.4. Протоколи підпису і верифікації

Схемою цифрового підпису є трійка  $(G, S, V)$ , де:  $G() \rightarrow sk, pk$  - алгоритм генерації приватного і публічного ключа,  $S(sk, m) \rightarrow \sigma(\text{sign})$  - алгоритм підписання, що по повідомленню і приватному ключу генерує підпис і  $V(pk, m, \sigma) \rightarrow \frac{\text{accept}}{\text{reject}}$  - алгоритм верифікації, що по публічному ключу, повідомленню і підпису або підтверджує підпис, або відкидає його.

Приватним ключем у нижченаведених алгоритмах є випадковий елемент  $sk = \mathbb{Z}_q *$ , публічним є  $pk = (sk) * P$ . Нехай  $R()$  - функція що повертає випадковий елемент з  $\mathbb{Z}_q$ ,  $H()$  - безпечна хеш функція.



### 2.4.1. ECDSA

Алгоритм підпису, який, з розглянутих кривих, може бути побудований на secp256r1 та secp256k1, але не на Curve25519:

```
S(sk, m):
  do:
    nonce := R()
    u := nonce * P = (x, y)
    r := reduce x modulo q
    s := (H(m) + r * sk) / nonce
  until r != 0 and s != 0

  return sig := (r, s)
```

Алгоритм верифікації:

```
V(pk, m, sign = (r, s)):
  if r = 0 or s = 0
    return reject
  a := H(m) / s
  b := r / s
  u := a * P + b * pk
  if u is point at infinity:
    return reject
  let u = (x, y)
  r' = reduce x modulo q
  if r' = r:
    return accept
  else:
    return reject
```

Для безпеки критично, щоб нонс (nonce) був отриманий випадково і мав рівномірний розподіл. У ситуації, якщо нонс для кожного підпису однаковий, зломисник може дізнатися приватний ключ. За допомогою такої атаки, наприклад, взламали PlayStation 3.

### 2.4.2. EdDSA

Ed від Edwards curve. Такою кривою з розглянутих є тільки Curve25519. EdDSA простіший за ECDSA. Алгоритм підпису:

```

S(sk, m):
  pk := sk * P
  nonce := H(H(sk) + m)
  r := reduce nonce modulo q
  R := r * P
  h := reduce H(R + pk + m) modulo q
  s := reduce (r + h * sk) modulo q
  return sign = (R, s)

```

Алгоритм верифікації:

```

V(pk, m, sign = (R, s)):
  h := reduce H(r + pk + m) modulo q
  Q1 := s * P
  Q2 := r + h * pk
  if Q1 = Q2
    return accept
  else:
    reject

```

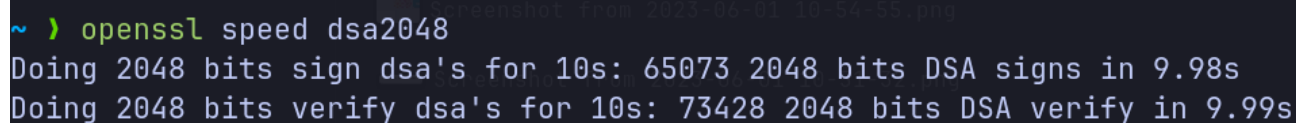
На відміну від ECDSA EdDSA має детерміністичний нонс, з чого слідує, що 2 однакові повідомлення підписані однаковим ключем будуть мати однаковий підпис. У зв'язку з детерміністичністю нонса зникають проблеми недостатньої його випадковості.

### 2.4.3. BLS

Це підпис який реалізований за допомогою спарювання точок еліптичної кривої. Тут не наводиться докладний опис, з ним можна ознайомитися в [7].

Зазвичай використовує bn256.

Порівняємо ECDSA і EdDSA. Заміряємо швидкість ECDSA/P-256, Ed25519(EdDSA що використовує Curve25519) та алгоритму що використовує  $\mathbb{Z}_p^*$  DSA-2048:



```

~ > openssl speed dsa2048
Doing 2048 bits sign dsa's for 10s: 65073 2048 bits DSA signs in 9.98s
Doing 2048 bits verify dsa's for 10s: 73428 2048 bits DSA verify in 9.99s

```

Рисунок 9: Швидкість роботи DSA-2048

```

~ ) openssl speed ecdsap256
Doing 256 bits sign ecdsa's for 10s: 794237 256 bits ECDSA signs in 9.95s
Doing 256 bits verify ecdsa's for 10s: 254639 256 bits ECDSA verify in 9.99s

```

Рисунок 10: Швидкість роботи ECDSA/P-256

```

~ ) openssl speed ed25519
Doing 253 bits sign Ed25519's for 10s: 408284 253 bits Ed25519 signs in 9.96s
Doing 253 bits verify Ed25519's for 10s: 149077 253 bits Ed25519 verify in 9.98s

```

Рисунок 11: Швидкість роботи Ed25519

	DSA-2048	Ed25519	ECDSA/P-256
Абсолютна кількість підписів/с	6520.3	40992.4	79822.8
Відносна кількість підписів/с	1	6.3	12.2
Абсолютна кількість верифікацій/с	7350.2	14937.6	25489.4
Відносна кількість верифікацій/с	1	2	3.5

Таблиця 2: Порівняльна таблиця алгоритмів підпису/верифікації

В цих бенчмарках є нюанс в тому, що OpenSSL зорієнтована на оптимізацію P256, через її розповсюдження в TLS 1.3, на відміну від Curve25519. Щоб переконатися у цьому порівняємо швидкість роботи підпису імплементації Ed25519 в OpenSSL і в libsodium [28]. Додатково проведемо такий самий бенчмарк для libsecp256k1 [29] ECDSA/secp256k1.

```

-----
Benchmark                                Time          CPU    Iterations
-----
benchmark_openssl_ed25519_sign          52873 ns       52818 ns       13241

```

Рисунок 12: Швидкість роботи ECDSA/P-256

Benchmark	Time	CPU	Iterations
benchmark_libsodium_ed25519_sign	22090 ns	22070 ns	31759

Рисунок 13: Швидкість роботи Ed25519

Benchmark	Time	CPU	Iterations
secp256k1_sign_bench	73264 ns	73152 ns	9544

Рисунок 14: Швидкість роботи ECDSA/secp251k1

Отже, OpenSSL може підписати 2546 повідомлень за мікросекнду, а libsodium 14435, що у 7 разів більше. Тобто можна висунути припущення, що libsodium Ed25519 в 3.5 рази швидший за ECDSA/P-256 OpenSSL. Як можна побачити вже вдруге, криптосистеми реалізовані за допомогою Curve25519 показують найкращу швидкодiю. Також можемо зробити висновок що ECC алгоритми підпису/верифікації перевищують по швидкодiї алгоритми підпису/верифікації, що використовують  $\mathbb{Z}_p^*$  щонайменше у 3.5 рази.

## 2.5. Імплементація бібліотеки

Цільовим рівнем безпеки для імплементованої бібліотеки є 256 бітний ключ ECC, який відповідає рівню безпеки симетричного шифрування з 128 бітним ключем. Як було визначено раніше, Curve25519 має кращі показники швидкодiї порівнюючи з P256 та secp256k1. Також, деякі нові блокчейни(наприклад [30]) створюються з використанням підписів на основі Curve25519, замість secp256k1, тож можна припустити, що бенчмарки покзали правильний результат і Curve25519 дійсно має найкращу швидкодiю серед описаних у цій роботі кривих. Окрім цього, EdDSA перевершує ECDSA, оскільки EdDSA не потребує недетермінованого генератора випадкових чисел. Також, ECDSA, на відміну від

EdDSA, не має властивості ексклюзивного володіння. Це означає, що для заданої пари  $(m, s)$ , де  $m$  - повідомлення,  $s$  - підпис нескладно згенерувати нову валідну пару  $(m, s')$  [31]. Тому бібліотека імплементує криптосистеми саме на Curve25519.

Для розробки бібліотеки, яка імплементує криптосистеми з використанням еліптичних кривих використаний 23 стандарт мови програмування C++. Імплементация бібліотеки, тести і наведені вище бенчмарки знаходиться за посиланням [32].

Коректна, швидка і безпечна імплементация самої кривої, а також елементів  $\mathbb{F}_p$ , дуже складна і часозатратна задача, оскільки неправильна імплементация може призвести до того, що криптосистема на її основі буде вразлива до атак. Тому було вирішено взяти імплементацию елементів  $\mathbb{F}_p$  з libsodium. На основі Curve25519 в бібліотеці імплементовано X25519 і Ed25519.

## ВИСНОВКИ

У цій роботі ґрунтовно розглянуто математичні основи криптографії еліптичних кривих: група точок еліптичної кривої над полем скінченної характеристики і обчислення над нею, припущення на яких ґрунтується криптографія еліптичних кривих та можливі атаки на них, можливі способи оптимізації швидкості роботи криптографічних систем на основі еліптичних кривих.

Як результат дослідження визначено, що крива Curve25519 є найбільш актуальною з розглянутих 256 бітних кривих. Однією з причин є те, що це крива Монтгомері, що дозволяє оптимізувати скалярний добуток точки еліптичної кривої на число. Як доказ, було проведено вимірювання швидкості роботи, які показали що найшвидшим стандартом протоколу Діффі-Геллмана є X25519 побудований на Curve25519. Також було порівнянню схеми цифрового підпису Ed25519 і ECDSA. Було виміряно, що Ed25519 має кращу продуктивність. Додатково серед переваг EdDSA є детермінованість, що дозволяє не залежати від наявності недетермінованого генератора випадкових чисел у системі, і ексклюзивне володіння підписом.

Було розроблено бібліотеку мовою C++, що імплементує протоколи Ed25519 і X25519.

Дослідження, проведені у цій роботі, можуть бути використані розробниками, які обирають алгоритми підпису і погодження ключів для розробки нової системи, або для вдосконалення існуючих

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

- [1] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang, “High-speed high-security signatures,” 2011. [Online]. Available: <https://ed25519.cr.yp.to/ed25519-20110926.pdf>
- [2] “Intel® intrinsics guide Avx.” [Online]. Available: [https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html#techs=AVX\\_ALL](https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html#techs=AVX_ALL)
- [3] W. Diffie, and M. E Hellman, “New directions in cryptography,” 1976. [Online]. Available: <https://ee.stanford.edu/~hellman/publications/24.pdf>
- [4] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” 1985. [Online]. Available: <https://people.csail.mit.edu/rivest/Rsapaper.pdf>
- [5] Taher ElGamal, “Public key cryptosystem and a signature scheme based on discrete logarithms.” [Online]. Available: <https://caislab.kaist.ac.kr/lecture/2010/spring/cs548/basic/B02.pdf>
- [6] Arjen K. Lenstra, and H. W. Lenstra, “The development of the number field sieve.”
- [7] Dan Boneh, and Victor Shoup, *A Graduate Course in Applied Cryptography*, 2023. [Online]. Available: <http://toc.cryptobook.us/book.pdf>
- [8] “What is ECC and why would I want to use it?” [Online]. Available: <https://www.globalsign.com/en/blog/elliptic-curve-cryptography>
- [9] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman, *An Introduction to Mathematical Cryptography*, 2014.
- [10] oost Renes, Craig Costello, and Lejla Batina, “Complete addition formulas for prime order elliptic curves,” 2015. [Online]. Available: <https://eprint.iacr.org/2015/1060>
- [11] R. Schoof., “Elliptic curves over finite fields and the computation of square roots mod  $p$ ,” 1985.

- [12] R. Schoof, “Counting points on elliptic curves over finite fields,” 1995.
- [13] Daniel J. Bernstein, and Tanja Lange, “Montgomery curves and the montgomery ladder.” [Online]. Available: <https://eprint.iacr.org/2017/293.pdf>
- [14] Lawrence C. Washington, *Elliptic Curves: Number Theory and Cryptography*, 2008.
- [15] V.S. Miller, “The weil pairing, and its efficient calculation,” 2004.
- [16] Han Wu, and Guangwu Xu, “A note on koblitz curves over prime fields,” 2020. [Online]. Available: <https://eprint.iacr.org/2020/1136.pdf>
- [17] S. Pohlig, and M. Hellman, “An improved algorithm for computing logarithms over  $Gf(p)$  and its cryptographic significance,” 1978. [Online]. Available: <https://ee.stanford.edu/~hellman/publications/28.pdf>
- [18] A. J. Menezes, T. Okamoto, and S. A. Vanstone, “Reducing elliptic curve logarithms to logarithms in a finite field,” 1993.
- [19] G. Frey, M. Müller, and H.-G. Rück, “The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems,” 1999.
- [20] H. W. Lenstra, “Factoring integers with elliptic curves,” 1987.
- [21] Leo Lai, “Lenstra’s elliptic curve factorization method,” 2016. [Online]. Available: <https://web.ma.utexas.edu/users/sl55444/CompsciTalk.pdf>
- [22] “Elliptic curve primality proof rank.” [Online]. Available: <https://t5k.org/top20/page.php?id=27>
- [23] Certicom Research, “SEC 1: Elliptic curve cryptography,” 2009. [Online]. Available: <https://www.secg.org/sec1-v2.pdf>
- [24] Certicom Research, “SEC 2: Recommended elliptic curve domain parameters,” 2010. [Online]. Available: <https://www.secg.org/sec2-v2.pdf>
- [25] D. J. Bernstein, “Curve25519: New diffie-hellman speed records,” 2006.
- [26] Michael Naehrig, Ruben Niederhagen, and Peter Schwabe, “New software speed records for cryptographic pairings.”
- [27] “Openssl.” [Online]. Available: <https://www.openssl.org/>



- [28] “Libsodium.” [Online]. Available: <https://doc.libsodium.org/>
- [29] “Libsecp256k1.” [Online]. Available: <https://github.com/bitcoin-core/secp256k1>
- [30] “Monero.” [Online]. Available: <https://www.getmonero.org/>
- [31] “A collection of common (interesting) cryptographic mistakes and learning resources. [Online]. Available: <https://gotchas.salusa.dev/>
- [32] “Implementation of library.” [Online]. Available: <https://github.com/Niksol15/eccpp>