НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра математичних методів захисту інформації

«До захисту допущено»
В.о. завідувача кафедри
_____ Михайло САВЧУК
«__» _____ 2021 р.

Дипломна робота

на здобуття ступеня бакалавра

зі спеціальності: 113 Прикладна математика
на тему: **«Детерминировані методи відображення повідомлення в точку еліптичної кривої, заданої у різних формах»**

Виконав: студент 4 курсу, групи ФІ-73
Талмач Дмитро Павлович

Керівник: Доцент і професор Ковальчук Людмила Василівна

Консультант: _ _____

Рецензент: звання, степінь, посада Прізвище І.П. _____

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

Студент _____

Київ — 2021

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра математичних методів захисту інформації

Рівень вищої освіти — перший (бакалаврський)
Спеціальність (освітня програма) — 113 Прикладна математика,
ОПП «Математичні методи криптографічного захисту інформації»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Михайло САВЧУК
«__» _____ 2021 р.

ЗАВДАННЯ
на дипломну роботу

Студент: Талмач Дмитро Павлович

1. Тема роботи: «*Детерминировані методи відображення повідомлення в точку еліптичної кривої, заданої у різних формах*»,
керівник: Доцент і професор Ковальчук Людмила Василівна,
затверджені наказом по університету №__ від «__» _____ 2021р.

2. Термін подання студентом роботи: 04 червня 2021 р.

3. Вихідні дані до роботи: *(впишіть вихідні дані до роботи)*

4. Зміст роботи: *(впишіть теми та задачі, які ви розкриваєте у роботі; можна робити це попунктно)*

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): *(якщо у вас є окремий ілюстративний матеріал окрім власне роботи (креслення, макети тощо), зазначайте; інакше вказуйте «Презентація доповіді»)*

6. Дата видачі завдання: 10 вересня 2020 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання	Примітка
1	Узгодження теми роботи із науковим керівником	01-15 вересня 2020 р.	Виконано
2	Огляд опублікованих джерел за тематикою дослідження	Вересень-жовтень 2020 р.	Виконано

Студент _____ Талмач Д. П.

Керівник _____ Ковальчук Людмила Василівна

РЕФЕРАТ

Кваліфікаційна робота містить: 63 стор., 1 рисунки, 1 таблиць, 20 джерел.

Данна робота присвячена розробці детермінованих поліноміальних алгоритмів кодування бітових векторів в точки еліптичних кривих представлених у різних формах. У роботі приводяться основні необхідні для розуміння викладок відомості про еліптичні криві, особливо криві в формі Едвардса. Далі детально розглядається проблема кодування елементів поля, над яким визначена крива, у множину точок кривої для використання у криптографічних протоколах, в основі яких лежить хешування або задача інкапсуляції ключа. У останньому розділі презентуються нові алгоритми, наводиться їх порівняльний аналіз.

ЕЛІПТИЧНІ КРИВІ, ФОРМА ЕДВАРДСА, ВКЛАДАННЯ БІТОВОГО ВЕКТОРА, ІНКАПСУЛЯЦІЯ КЛЮЧА, ДЕТЕРМІНОВАНІ АЛГОРИТМИ ПОСТІЙНОГО ЧАСУ ВИКОНННЯ

ABSTRACT

Graduate work contains 63 pages, 1 figure, 1 table and 20 references.

The work is devoted to constructing deterministic polynomial algorithm for encoding sequences of bits into points of Elliptic Curves represented in different forms. The work presents basic information related to the topic of Elliptic Curves, especially in the Edwards form, that is necessary for understanding further mathematical calculations. Next, the problem of encoding underlying field elements, over which the curve is defined, into points of the curve for using this encoding in cryptographic protocols, which are based on hashing or key encapsulation schemes, is considered in more detail. In the last section new algorithms are presented and compared.

ELLIPTIC CURVES, EDWARDS FORM, EMBEDDING A VECTOR OF BITS, KEY ENCAPSULATION, DETERMINISTIC CONSTANT TIME ALGORITHMS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ЕК(англ. "EC")— еліптична крив

СП(англ. "FF") — скінченне поле

ЗДЛ — задача дискретного логарифмування

W — еліптична крива задана у формі Веєрштрасса

M — еліптична крива задана у формі Монтгомері

E — еліптичка крива задана у формі Едвардса

M — операція множення в скінченному полі

S — операція обчислення квадрату в скінченному полі

I — операція знаходження оберненого елемента в скінченному полі

D — операція множення на попередньо визначену константу в скінченному полі

legendre — обчислення символу Лежандр

sqrt — обчислення квадратного кореня в скінченному полі

РАЕ — розширенний алгоритм Евкліда

ВСТУП

Актуальність дослідження. Актуальність роботи пояснюється важливістю кодування, як складової частини багатьох криптосистем та нетривіальністю таких кодувань в групу точок еліптичної кривої. А акцентування уваги на кривих Едвардса в даній роботі пояснюється значною перевагою в швидкості групових операцій кривих Едвардса над іншими еліптичними кривими.

Метою дослідження є розробити методи та алгоритми вкладення бітового вектора у точку еліптичної кривої заданої в різних формах.

У процесі дослідження було поставлено наступні **задачі дослідження**:

1) Огляд літератури за данним питанням, аналіз існуючих алгоритмів;
2) Аналіз особливостей ЕК заданих у різних формах та використання цих особливостей для побудови алгоритмів вкладання бітових векторів в точки ЕК та відновелення з цих точок;
3) Розробка відповідних алгоритмів вкладання і відновлення бітового вектора в точки для ЕК заданих в довільній формі;
4) Аналіз роботоздатності побудованих алгоритмів.

Об'єктом дослідження є процес побудови та функціонування криптосистем на еліптичних кривих в формі Едвардса.

Предметом дослідження є розробка методів та алгоритмів вкладення бітового вектора у точку еліптичної кривої заданої у різних формах.

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: алгебра, алгебраїчна геометрія, теорія складності обчислень.

Наукова новизна отриманих результатів полягає в створенні нових алгоритмів детермінованого кодування в еліптичні криві представлені в формах Монтгомері, Едвардса та неповній Веєрштрасса.

Практичне значення Отримані алгоритми є цілком придатними для практичного застосування. За своєю ефективністю вони не поступаються існуючим, а в деяких випадках можливо навіть перевершують.

1 ОЗНАЙОМЛЕННЯ З ЕЛІПТИЧНИМИ КРИВИМИ ТА ІСНУЮЧИМИ АЛГОРИТМАМИ КОДУВАННЯ В ЕЛІПТИЧНІ КРИВІ

В даному розділі приводяться основні відомості, що стосуються ЕК та їх застосування в криптографії. Більш детально разглядаються питання пов'язані з кривими в формі Едвардса такі, як переваги арифметичних операцій, ізоморфність до інших форм, класифікація тощо. Приводяться деякі відомі на момент написання роботи алгоритми кодування повідомлень в точки на ЕК. Надаються необхідні означення та формулювання.

1.1 Загальні відомості про ЕК

1.1.1 Причина застосування ЕК в криптографії

Використовувати ЕК у криптографії вперше незалежно один від одного запропонували криптографи Ніл Кобліц [13] і Віктор Міллер [14]. Причиною взагалі попиту на криптосистеми на ЕК були поява і розвиток субекспоненційних алгоритмів для розв'язання ЗДЛ у мультиплікативній групі скінченного поля та постійне вдосконалення обчислювальної техніки, що змушувало криптографів збільшувати довжини ключів у криптосистемах, це в свою чергу призводило до уповільнення роботи. Річ в тому, що для скінченного поля всі ці алгоритми базуються на існуванні простих чисел або незвідних поліномів, а у групі точок ЕК таких об'єктів просто не існує. Тому для розв'язання ЗДЛ на ЕК на даний момент існують тільки експоненційні алгоритми. Найкращий відомий алгоритм для розв'язання ЗДЛ на ЕК у загальному випадку має обчислювальну складність [15]:

$$C_{EC}(n) = 2^{n/2} \tag{1.1}$$

В той час, як ЗДЛ у скінченному полі може бути обчислений, використовуючи метод решета числового поля, за час пропорційний до

$$C_{FF}(N) = \exp(c_0 N^{1/3}(\log(N\log 2))^{2/3}), \tag{1.2}$$

відмітимо також, що алгоритми для розв'язання задачі факторизації мають приблизно таку ж саму асимптотичну складність [15], що дозволяє залучити до порівняння таку поширену на практиці криптосистему, як RSA.

Прирівнюючи C_{EC} і C_{FF}, отримаємо, що для встановлення однакового рівня безпеки повинно виконуватись наступне [15]:

$$n = \beta N^{1/3}(\log(N\log 2))^{2/3}, \quad \text{де } \beta = 2c_0/(\log 2)^{2/3} \tag{1.3}$$

Отже, бачимо, що при однаковій криптографічній стікості, довжина ключа для криптосистеми на ЕК росте приблизно як кубічний корінь від довжини ключа для криптосистеми на скінченному полі.

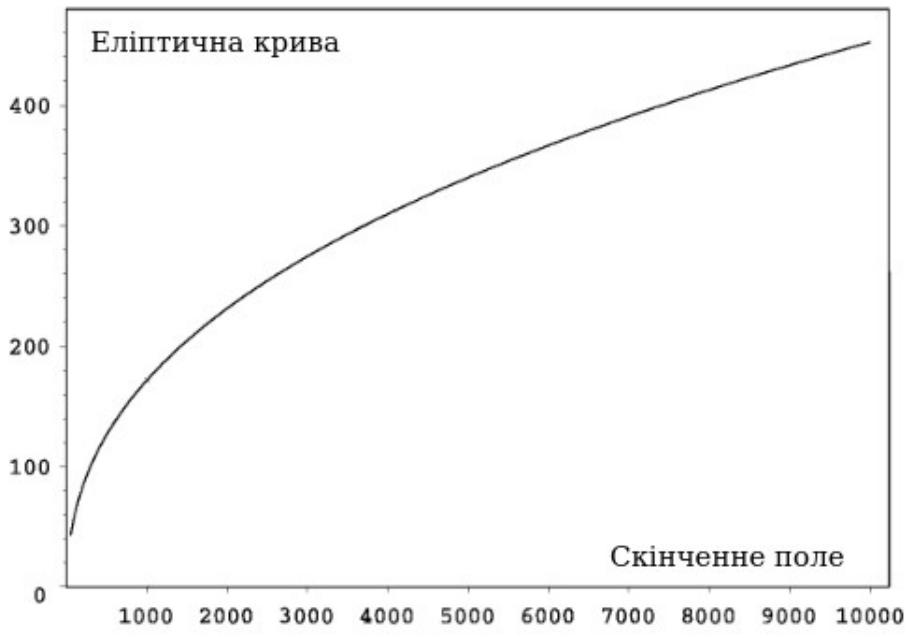

Рисунок 1.1 – Порівняння довжин ключів для криптосистем на ЕК та класичних криптосистем, що забезпечують одинаковий рівень стійкості.

Також з цього виходить, що при однаковій довжині ключа у крипосистемах, які базуються на ЗДЛ на ЕК стійкість буде значно більшою ніж на скінченному полі. Так, наприклад, ключ довжиною 4096 бітів для RSA надає такий же рівень безпеки, що й ключ довжиною 313 бітів в криптосистемі на ЕК. Більш загально різницю в довжині ключів можна побачити на рисунку 1.1 запозиченому з роботи [15].

Це пояснює спалах інтересу до ЕК в криптографії, як до більш дешевшої альтернативи загальноприйнятим криптосистемам, що базуються на мультиплікативній групі скінченного поля. На практиці більш короткі ключі приводять до більш швидких реалізацій з використанням менших об'ємів пам'яті і меншого енергоспоживання.

1.1.2 ЕК в узагальненій формі Веєрштрасса

Означення 1.1. [11] ЕК в узагальненій формі Веєрштрасса над полем F_q

$$C(F_q) : y^2 + a_1xy + a_3 + y = x^3 + a_2x^2 + a_4x + a_6 \tag{1.4}$$

Точки ЕК разом з $\mathcal{O}_E$ утворюють абелеву групу за додаванням, що дозволяє будувати над ЕК криптосистеми. Форма (1.4) використовується для більшої загальності, бо з нею можна працювати над полями будь-якої характеристики. Якщо $char(F) \neq 2$, то можна поділити ліву і праву частини рівняння на 2 і зібрати повний квадрат:

$$(y + \frac{a_1x}{2} + \frac{a_3}{2})^2 = x^3 + (a_2 + \frac{a_1^2}{4}x^2 + (a_4 + \frac{a_1a_3}{2})x + (\frac{a_3^2}{4} + a_6)),$$

що можна переписати, виконавши заміну $y_1 = y + \frac{a_1x}{2} + \frac{a_3}{2}$, як

$$y_1^2 = x^3 + a_2'x^2 + a_4'x + a_6',$$

якщо $char(F) \neq 3$, то можна виконати заміну $x_1 = x + \frac{a_2'}{3}$, тоді отримаємо скорочену форму Веєрштрасса(див. означення (1.6))

$$y_1^2 = x_1^3 + Ax_1 + B,$$

для деяких констант A, B

Отже, будь-яку ЕК визначену над полем F, $char(F) \notin \{2,3\}$ можна привести до скороченої форми Веєрштрасса (1.6), працювати з якою значно простіше ніж з узагальненою формою(1.4).

Розрізняють сингулярні та несингулярні ЕК. Сингулярними називають ті ЕК, що мають злами або самоперетини. Несингулярними - ті, що їх не мають, їх через те ще називають гладкими.

Твердження 1.1. *Умова несингулярності для ЕК в формі (1.4). Позначимо* $F(x,y) = y^2 + a_1xy + a_3 + y - x^3 - a_2x^2 - a_4x - a_6$. *Тоді ЕК буде несингулярною, якщо система*

$$\begin{cases} F(x,y) = 0 \\ \frac{\partial F(x,y)}{\partial x} = 0 \\ \frac{\partial F(x,y)}{\partial y} = 0 \end{cases} \tag{1.5}$$

не має жодного розв'язку у полі F_q *і у будь-якому його розширенні.*

Для застосування у криптографії цікавими є саме несингулярні криві.

1.2 Різні форми ЕК

Існують різні форми для представлення ЕК. Така різноманітність пояснюється тим, що ці різні форми мають свої особливості та переваги, які виділяють їх з усього пласту ЕК.

1.2.1 Крива в канонічній формі Веєрштрасса

У подальших розділах розглядатиметься зв'язок, а саме ізоморфізм, кривих в формах Едвардса і Монтгомері з кривими в формі Веєрштрасса, тому доцільно привести тут означення останньої.

Для подальших посилань приводимо тут означення кривої в короткій формі Веєрштрасса.

Означення 1.2. [2]

$$W(F_p) : y^2 = x^3+ax+b, \quad a,b \in F_p^*,\ char(F_p) \neq 2,3,\ \Delta=-16(4a^3+27b^2) \neq 0 \tag{1.6}$$

[2] Нейтральним елементом виступає точка на нескінченності $\mathcal{O}_E$. Оберненим елементов до точки $P=(x,y) \neq \mathcal{O}_E$ визначається точка $-P=(x,-y)$, якщо $P=\mathcal{O}_E$, то відповідно $-P=\mathcal{O}_E$. Закон додавання двох різних точок має наступний вигляд:

$$(x_1,y_1)+(x_2,y_2) = (x_3,y_3) = \left(\left(\frac{y_2-y_1}{x_2-x_1}\right)^2 - x_1 - x_2, -y_1 + \frac{y_2-y_1}{x_2-x_1}(x_1-x_3)\right) \tag{1.7}$$

Для двох однокових точок(подвоєння):

$$2(x_1,y_1) = (x_3,y_3) = \left(\left(\frac{3x_1^2+a}{2y_1}\right) - 2x_1, -y_1 + \left(\frac{3x_1^2+a}{2y_1}\right)(x_1-x_3)\right) \tag{1.8}$$

Для обчислення формули додавання (1.7) потрібно виконати **2M** + **1S** + **1I** операцій. А для подво'єння (1.8) — **2M** + **2S** + **1I** [16]. Нажаль операція знаходження оберненого елемена є дуже неприємною, бо потребує більше часу для обчислення(зазвичай за PAE).

Щоб уникнути потреби в обчисленні обернених елементів в формулах (1.7), (1.8), здійснюється перехід до іншої системи координат(проективної або зваженої проективої), куди "переноситься" знаменник з цих формул. Якщо перевести рівняння (1.6) в систему зважених проективних координат(координат Якобі), то складність

групових операцій стане наступною: для додавання — **11M + 5S**, для подвоєння — **1M + 8S + 1D** [16].

ЕК визначену над полем $char(F) \notin \{2,3\}$ в будь-якій формі можна раціональними перетвореннями привести до ізоморфної їй ЕК в скороченій формі Веєрштрасса. Але на навпаки. Тому скорочена форма Веєрштрасса зустрічається дуже часто.

1.2.2 Крива в формі Монтгомері

Рівняння кривої в формі Монгомері має такий вигляд:

Означення 1.3. ЕК в формі Монтгомері за [19].

$$M'(F_p) : By^2 = x^3 + Ax^2 + x \tag{1.9}$$
$$B(A^2 - 4) \neq 0 \text{ - умова несингулярності}$$

[19] Арифметичні операції можуть бути ефективно реалізовані без використання ординат. Нехай $P = (x_1, y_1)$ буде точкою на M. Тоді в проективних координатах $P = (X_1 : Y_1 : Z_1)$, і нехай $nP = (X_n, Y_n, Z_n)$. Тоді сума $(n + m)P = nP + mP$ обчислюється за формулами, де Y_n ніде не фігурує:
додавання $(n \neq m)$:

$$X_{m+n} = Z_{m-n}((X_m - Z_m)(X_n + Z_n) + (X_m + Z_m)(X_n - Z_n))^2$$
$$Z_{m+n} = X_{m-n}((X_m - Z_m)(X_n + Z_n) - (X_m + Z_m)(X_n - Z_n))^2$$

подвоєння $(n = m)$:

$$4X_nZ_n = (X_n + Z_n)^2 - (X_n - Z_n)^2$$
$$X_{2n} = (X_n + Z_n)^2(X_n - Z_n)^2$$
$$Z_{2n} = 4X_nZ_n((X_n - Z_n)^2 + ((A + 2)/4)(4X_nZ_n))$$

Додавання точок за цими формулами можна виконати за **4M + 2S**,

подвоєння — за **3M + 2S** [19], але треба взяти до уваги, що в них є відсутнім Y_n. Для деяких застосувать достатньо тільки X_n, але іноді потрібно мати можливість обчислювати ординату також. Це відновлення ординати $y_n = Y_n/Z_n$ відбувається за формулою:

$$y_n = \frac{(x_1x_n + 1)(x_1 + x_n + 2A) - 2A - (x_1 - x_n)^2 x_{n+1}}{2By_1}$$

де $P = (x_1, y_1)$ і x_n та x_{n+1} є афінними x-координатами точок nP і $(n+1)P$.

Для застосувань в даній роботі зручно буде використовувати криву Монтгомері в вигляді:

Означення 1.4. ЕК в формі Монтгомері за [2].

$$M(F_p) : v^2 = u^3 + Au^2 + Gu, \tag{1.10}$$

Тому при подальших посиланнях матиматься на увазі саме вона.

Завжди можна перетворити криву в формі Монтгомері в криву в формі Веєрштрасса, але не навпаки. Ізоморфність ЕК в формі Монтгомері до ЕК в формі Веєрштрасса буде детально описана в 1.6.3.

Форма Монтгомері, хоч і не так поширена як форма Веєрштрасса, має свою область застосувать у криптографії і теорії чисел. Вона використовується у методі Ленстри факторизації за допомогою ЕК, що має субекспоненційну складність і посуває третє місце за швидкістю [17] серед усіх алгоритмів факторизації(поступаючись тільки методу загального числового решета і методу квадратичного решета). Також зараз є дуже поширеною крива “Curve25519” представлена в [18], яка є кривою в формі Монтгомері.

1.3 Крива Едварса в оригінальній формі

Відносно нещодавно множина еліптичних кривих поповнилася новою формою, презентованою Гарольдом Едвардсом. У своєму дослідженні [5]

він посилається на роботи таких відомих вченних, як Гауса та Абеля, які приблизно 2 століття тому працювали над схожими рівняннями [2].

Означення 1.5. [2] Кривою Едвардса в оригінальній формі називається крива виду:

$$E(F_p) : x^2 + y^2 = e^2(1 + x^2y^2), \quad e \in F_p^*, char(F_p) \neq 2, e^4 \neq 1 \qquad (1.11)$$

[2] Групова операція додавання на тиких кривих вводиться наступним чином:

$$(x_1, y_1) + (x_2, y_2) = \Big(\frac{x_1y_2 + x_2y_1}{e(1 + x_1x_2y_1y_2)}, \frac{y_1y_2 - x_1x_2}{e(1 - x_1x_2y_1y_2)}\Big) \qquad (1.12)$$

[2] Обернений елемент за додаванням визначається симетричним відображенням відносно осі OY: $-(x, y)=(-x, y)$. Нейтральним елементом виступає точка $\mathcal{O}_E(0,e)$. Якщо при додаванні координати точок збігаються, тоді відповідно:

$$(x_1, y_1) + (x_2, y_2) = 2(x_1, y_1) = \Big(\frac{2x_1y_1}{e(1 + x_1^2y_1^2)}, \frac{y_1^2 - x_1^2}{e(1 - x_1^2y_1^2)}\Big) \qquad (1.13)$$

На таких кривих завжди існує точка другого порядку $D_0=(0, -e)$. Крім того існують особливі точки, так, наприклад, $P=(1,1)$ при подвоєнні породжує невизначеність типу $\frac{0}{0}$ для y-координати [2].

З самої форми формули (1.13) випливає існування точки другого порядку $D_0=(0, -e)$ та хоча б двох точок четвертого порядку $\pm F_0=(\pm e, 0)$. Процес знаходження цих точок детально описаний в [2].

Важливою властивістю, що обмежує кількість кривих, які можна представити в формі Едвардса, є обов'язковість існування точок четвертого порядку [4, с. 4] [2].

Едвардсу вдалося знайти зв'язок кривої (1.11) з кривою в формі Веєрштрасса. Виявилося, що за певних визначених раціональних перетворень, криву з форми (1.11) можна привести до вигляду (1.6), що означає що вони ізоморфні [2].

Існування особливих точок(точок на нескінченності) приводить до певних незручностей - доведеться окремо вводити арифметику додавання цих особливих точок [2]. Тому для використання в задачах криптології рівняння (1.11) приводять до більш зручного вигляду, накладаючи дадатокові обмеження [4] до оригінальної кривої, запропонованої Едвардсом. Саме такі криві розглядатимуться в подальших розділах даної роботи.

1.4 Крива форми Едвардса в модифікації Бернстейна та Ланге

Форма еліптичної кривої запропонована Едвардсом незабаром після видання його публікації була допрацьована у статті [4], де була запропонована модифікація, яка усувала проблеми з особливими точками, та робила можливим значне пришвидшення операції додавання. Такі корисні властивості еліптичних кривих не оминули авторів [4] - прискорення майже в 1.5-1.6 рази могло б помітно здешевшити послуги, в основі яких лежать криптосистеми на еліптичних кривих [2].

Означення 1.6. [2]

$$E(F_p) : x^2 + y^2 = e^2(1 + dx^2y^2), \quad char(F_p) \neq 2, d(1 - de^4) \neq 0, \left(\frac{d}{p}\right) = -1 \tag{1.14}$$

Додавання працює таким чином [2]:

$$(x_1, y_1) + (x_2, y_2) = \left(\frac{x_1y_2 + x_2y_1}{e(1 + dx_1x_2y_1y_2)}, \frac{y_1y_2 - x_1x_2}{e(1 - dx_1x_2y_1y_2)}\right) \tag{1.15}$$

Операція додавання однородна - подвоєння рахується за тією ж самою формулою. При $x_1 = x_2, y_1 = y_2$ виконується наступне [2]:

$$(x_1, y_1) + (x_2, y_2) = 2(x_1, y_1) = \left(\frac{2x_1y_1}{e(1 + dx_1^2y_1^2)}, \frac{y_1^2 - x_1^2}{e(1 - dx_1^2y_1^2)}\right) \tag{1.16}$$

Оберна точка та нейтральний елемент за додаванням визначаються таким же самим чином, що й (1.11). Але існує лише одна точка другого порядку $D=(0,-e)$ і рівно дві точки четвертого порядку $\pm F_0=(\pm e,0)$ [2].

Використання однієї формули для додавання та подвоєння точок приводить до простіших програмних імплементацій алгоритмів, що має особливо істотне значення для "embedded" технологій [2].

Основною відмінністю від кривих в оригінальній формі(1.11) є те, що криві виду (1.14) не мають особливих точок. Автори [4] назвали цю властивість повнотою операції додавання. Приведемо формулювання теореми про повноту операції додавання [2, с. 20], детальне доведення описане в [2]

Теорема 1.1. *Для будь-яких пар точок кривої (1.14) знаменники закону додавання (1.15) не обертаються в нуль при* $dx_1x_2y_1y_2 \neq \pm 1$

Наступна теорема [2, с. 21] постулює коректність закону додавання (1.15). Доведення наведене в [2] .

Теорема 1.2. *Нехай* $P=(x,y)$ *та* $Q=(u,v)$ *- точки кривої (1.14) та виконується рівності* $x^2+y^2=e^2(1+dx^2y^2)$, $u^2+v^2=e^2(1+du^2v^2)$. *Визначимо* $X=\frac{xy+uv}{e(1+dxyuv)}$, $Y=\frac{yv-xvu}{e(1-dxyuv)}$. *Тоді виконуватиметься наступне* $X^2+Y^2=e^2(1+dX^2Y^2)$.

При підстановці $\frac{x}{e}\to x, \frac{y}{e}\to y$, отримуємо ізоморфну до (1.14) криву з лише одним параметром d, з чого слідує надлишковість параметру e [2].

$$x^2+y^2=1+d'x^2y^2, \quad d'=de^4 \tag{1.17}$$

Наступна теорема з [2] описує зв'язок квадратичності $(1-d)$ з існуванням певних точок на кривій. Ця теорема знадобиться для посилань при описанні алгоритму кодування в ЕК Едвардса. Теорема приводиться без доведення:

Теорема 1.3. *Необхідною і достатньою умовою існування 4ьох*

точок 8го порядку кривої (1.14) є справедливість рівності

$$\left(\frac{1-d}{p}\right) = 1 \tag{1.18}$$

Усування параметру e скорочує кількість використованих операцій множення в полі F_p на 2 в формулі (1.15) [2].

Роздивимось властивість цих кривих, що становить особливу цікавість для практичних застосувань, - швидке та ефективне додавання і подвоєння, алгоритми яких описуються в роботі [4, с. 9-10].

Швидке додавання:

Щоб оминути трудомістку операцію знаходження оберненого елемента в формулі (1.15) обчислюємо предстaлення точок в проекційних координатах, які задовольняють рівнянню кривої $(X^2 + Y^2)Z^2 = e^2(Z^4 + dX^2Y^2)$.

$$(x_1, y_1) \to (X_1, Y_1, Z_1)$$
$$(x_2, y_2) \to (X_2, Y_2, Z_2)$$
$$Z_i \neq 0, (x_i, y_i) = (X_i/Z_i, Y_i/Z_i)$$

В отриманій структурі (0, e, 1) - нейтральний елемент, обернений: $-(X_1, Y_1, Z_1)=(-X_1, Y_1, Z_1)$. Множення точок у представленні в проекційних координатах обчислюється за наступними формулами:

$$A = Z_1 \cdot Z_2;\ B = A^2;\ C = X_1 \cdot X_2;\ D = Y_1 * Y_2;\ E = d \cdot C \cdot D;$$
$$F = B - E;\ G = B + E;$$
$$X_3 = A \cdot F \cdot ((X_1 + Y_1) \cdot (X_2 + Y_2) - C - D);$$
$$Y_3 = A \cdot G \cdot (D - C);$$
$$Z_3 = e \cdot F \cdot G$$

В результаті маємо (X_3, Y_3, Z_3), за якими знаходиться шукана точка

$$(X_3, Y_3, Z_3) \to (x_3, y_3)$$
$$(x_3, y_3) = (X_3/Z_3, Y_3/Z_3)$$

Як бачимо, алгоритм використовує **10M + 1S**. Для порівняння, найшвидший алгоритм для додавання точок(в зважених проективних координатах) ЕК у формі Веєрштрасса потребує **11M + 5S**.

Швидке подвоєння:

Для формули подвоєння автори [4] використали тотожність $e(1 + dx_1^2y_1^2) = (x_1^2 + y_1^2)/e$, що випливає з вигляду самого рівняння (1.14), щоб переписати знаменники з формули (1.17):

$$e(1 + dx_1^2y_1^2) \to (x_1^2 + y_1^2)/e$$
$$e(1 - dx_1^2y_1^2) \to (2e^2 - (x_1^2 + y_1^2))/e$$

Тоді подвоєння матиме такий вигляд:

$$2(x_1, y_1) = \Big(\frac{2x_1y_1}{e(1 + dx_1^2y_1^2)}, \frac{y_1^2 - x_1^2}{e(1 - dx_1^2y_1^2)}\Big) = \Big(\frac{2x_1y_1e}{x_1^2 + y_1^2}, \frac{(y_1^2 - x_1^2)e}{2e^2 - (x_1^2 + y_1^2)}\Big) \tag{1.19}$$

Переписуючи $2x_1y_1$ як $(x_1+y_1)^2-x_1^2-y_1^2$, ми використовуємо вже обраховані елементи, тому зменшується потрібна для обчислення кількість операцій в полі F_p.

Отже, в результаті виходить такий алгоритм обчислення:

$$B = (X_1 + Y_1)^2;\ C = X_1^2;\ D = Y_1^2;\ E = C + D;\ H = (c \cdot Z_1)^2;$$
$$J = E - 2H;\ X_3 = e \cdot (B - E) \cdot J;\ Y_3 = e \cdot E \cdot (C - D);\ Z_3 = E \cdot J$$

Як бачимо, алгоритм використовує **3M + 4S**. Для порівняння, найшвидший алгоритм для додавання точок(в зважених проективних координатах) ЕК у формі Веєрштрасса потребує **1M + 8S + 1D**.

Операція множення точки на скаляр є однією з критичних обчислень для еліптичної криптографії. Описані алгоритми дозволяють значно прискорити дану операцію.

Клас рівнянь з $e=1$ в роботі [2, с. 21] отримав назву класу повних кривих Едвардса. В наступній секції формулювання цього класу оглядаються більш детально.

1.5 Клас повних кривих в формі Едвардса

В данній роботі означення, що стосуються кривих в формі Едвардса узгоджуються з класифікацією запропонованою в роботі [2].

Означення 1.7. [2] *Повною кривою в формі Едвардса* називають криву, що задається рівнянням:

$$E(F_p) : x^2+y^2 = (1+dx^2y^2), \quad d \in F_p^*, d(1-d) \neq 0, \left(\frac{d}{p}\right) = -1, char(F_p) \neq 2 \tag{1.20}$$

[2] Параметр d був упроваджений авторами [4] для кращого узагальнення кривих оригінальної форми (1.11), запропонованої Едвардсом [5].

Операція додавання та подвоєння на ній вводяться таким чином [2]:

$$(x_1, y_1) + (x_2, y_2) = \left(\frac{x_1x_2 - y_1y_2}{1 - dx_1x_2y_1y_2}, \frac{x_1y_2 + x_2y_1}{1 + dx_1x_2y_1y_2}\right) \tag{1.21}$$

$$2(x_1, y_1) = \left(\frac{x_1^2 - y_1^2}{1 - dx_1^2y_1^2}, \frac{2x_1y_1}{1 + dx_1^2y_1^2}\right) \tag{1.22}$$

Координати x, y переставляються місцями(відносно 1.15 та 1.17) для отримання горизонтальної симетрії обернених точок: обернена точка до $P(x_1,y_1)$ визначається як $Q= - P=(x_1, - y_1)$. Нейтральним елементом виступає точка $O(1,0)$. Існує лише одна точка другого порядку $D=(-1, 0)$ і рівно дві точки четвертого порядку $\pm F_0=(0, \pm 1)$ [2].

[2] Для використання у прикладних криптосистемах обираються

криві порядку $N_E=4n$, де n - велике просте число(велике - щоб мати велику різноманітність точок, просте - щоб базова точка біла генератором циклічної підгрупи). Така базова точка знаходиться дуже просто [2, с. 60]: обирається довільна точка $Q=(x_Q, y_Q)$, що належить кривій, далі двома подвоєннями обчислюється $G=(x_G, y_G)=4Q$, якщо $G\neq O$ - генератор порядка n.

Причиною інтересу до повних кривих Едвардса є безпрецедентно швидка арифметика, яка забезпечує виграш в 1.41 рази над арифметикою кривих Веєрштрасса [20]. До того ж представлення нейтрального елементу групи точок в афінних координатах та універсальність закону додавання дозволять ще ефективіше пришвидшити програмні реалізації криптоалгоритмів [2].

1.6 Ізоморфізм

Якщо криві, які вже використовуються в криптографічних схемах, однозначно приводяться до форми Едвардса раціональними перетвореннями, тоді можна із використанням вищеописаних ефективних операцій швидкого додавання та подвоєння значно пришвидшити роботу цих схем. Саме тому великий інтерес являє в собі ізоморфність кривої в формі Едвардса до кривих в формі Монтгомері і Веєрштрассе при накладанні деяких обмежень.

Скорочена форма Веєрштрасса (1.6) є найбільш поширеною, бо ЕК з будь-якої форми над будь-яким полем, крім тих, що мають $char(F_q) \in \{2, 3\}$, можна привести до ізоморфної їм кривої в скороченій формі Веєрштрасса, але не навпаки. Так, наприклад, ізоморфізм кривих в формі Веєерштрасса і Едвардса можливий не для кожної пари параметрів a, b рівняння (1.6), також крива повинна мати рівно 2 точки четвертого порядку [2].

В [2] перетворення кривої у формі Едвардса у форму Веєерштрасса відбувається за допомогою проміжного перетворення у форму Монтгомері,

що набагато простіше ніж прямі перетворення наведені в [11].

У практичній частині даної роботи будуть використані ізоморфності форм Едвардса та Монтгомері(для алгоритмів 2.1, 2.2, 2.3, 2.4) і Веєрштрасса та Монтгомері (для 2.9, 2.10). Тому доцільним буде навести тут математичні викладки, представлені в роботі [2], які описують ці ізоморфізми.

1.6.1 Перетворення $E \to M_1$

Наведемо тут послідовність раціональних перетворень, потрібну для переходу від кривої в формі Едвардса до кривої в формі Монтгомері. Нехай крива предствалена в вигляді (1.20).

Виконаємо заміну:

$$x = 2\frac{u}{\widetilde{v}}, \quad y = \frac{u-1}{u+1} \tag{1.23}$$

Отримаємо:

$$u\frac{u^2}{\widetilde{v}^2} + \frac{u-1}{u+1}^2 = 1 + 4d\frac{u^2(u-1)^2}{\widetilde{v}^2(u+1)^2}$$

$$\frac{4u^2(u^2+2u+1) + \widetilde{v}^2(\cancel{u^2} - 2u + \cancel{1})}{\widetilde{v}^2(u+1)^2} = \frac{\cancel{\widetilde{v}^2}u^2 + 2\widetilde{v}^2u + \cancel{\widetilde{v}^2} + 4d(u^4 - 2u^3 + u^2)}{\widetilde{v}^2(u+1)^2}$$

$$\frac{4u^4 + 8u^3 + 4u^2 - 2u\widetilde{v}^2}{\widetilde{v}^2(u^2+1)} = \frac{2u\widetilde{v}^2 + 4du^4 - 8du^3 + 4du^2}{\widetilde{v}^2(u^2+1)}$$

$$\frac{4u^4(1-d) + 8u^3(1+d) + 4u^2(1-d)}{\widetilde{v}^2(u^2-1)} = \frac{4u\widetilde{v}^2}{\widetilde{v}^2(u+1)^2}$$

Поділимо знаменник лівої частини на $(1-d)$:

$$\frac{\cancel{4u}(u^3 + 2u^2\frac{1+d}{1-d} + u)}{(1-d)^{-1}\cancel{\widetilde{v}^2}\cancel{(u+1)^2}} = \frac{\cancel{4u}\widetilde{v}^2}{\cancel{\widetilde{v}^2}\cancel{(u+1)^2}}$$

Отримали рівняння:

$$M_1(F_p) : \frac{\widetilde{v}^2}{1-d} = u^3 + 2\frac{1+d}{1-d}u^2 + u \tag{1.24}$$

Отримане рівняння є частковим випадком рівняння в формі Монтгомері (1.3) з $A = \frac{1}{1-d}, B = 2\frac{1+d}{1-d}$, назвемо його M_1 для подальших посилань.

1.6.2 Перетворення $M_1 \to M$

У перетворенні $M_1 \to M$ можливі два випадки: коли $\left(\frac{1-d}{p}\right) = 1$ і $\left(\frac{1-d}{p}\right) = -1$. Нас для практичного застосування цікавитиме саме перший випадок.

$$M_1 : \frac{\widetilde{v}^2}{1-d} = u^3 + 2\frac{1-d}{1+d}u^2 + u$$

$\left(\frac{1-d}{p}\right) = 1$, тому можемо перепозначити $f = \frac{1}{\sqrt{1-d}}$:

$$f^2\widetilde{v}^2 = u^3 + 2\frac{1-d}{1+d}u^2 + u$$

Зробимо заміну:

$$u = u, v = f\widetilde{v} \tag{1.25}$$

Отримали:

$$v^2 = u^3 + 2\frac{1-d}{1+d}u^2 + u$$

Бачимо, що отримали рівняння, яке є частковим випадком (1.10), з $a = 2\frac{1-d}{1+d}, b = 1$.

1.6.3 Перетворення $W \to M$ при $a = 0$

Представлене в даному підрозділі перетворення буде використано в алгоритмі 2.9 для детермінованого кодування в ЕК в неповній формі

Веєрштрасса. Нехай маємо рівняння в неповній формі Веєрштрасса:

$$y^2 = x^3 + c \tag{1.26}$$

Якщо, $(-c)$ — кубічний лишок, то можемо представити його в вигляді $(-c) = s^3$. Тоді можемо переписати:

$$y^2 = x^3 + c = x^3 - s^3 = (x - s)(x^2 + sx + s^2)$$

далі виконуємо заміну описану в [2]:

$$\begin{cases} x - s = u \\ y = v \end{cases}$$

Отримаємо:

$$v^2 = u((u + s)^2 + s(u + s) + s^2)$$
$$v^2 = u(u^2 + 3su + 3s^2)$$
$$v^2 = u^3 + 3su^2 + 3s^2u$$

Отже, в результаті отримали частковий випадок рівняння ЕК в формі Монтгомері (1.10), з $A = 3s$, $G = 3s^2$.

1.7 Існуючі алгоритми кодування

В даному підрозділі в стиснутій формі наводяться найвідоміші, на момент написання роботи, алгоритми для кодування в точки на ЕК, наводятся їх відмінні особливості. Більше уваги приділяється останньому алгоритму, бо на ньому базується практична частина цієї роботи.

Багато криптографічних протоколів (шифрування, підписи, PAKE(password authentication key exchange), IBE(identity based encryption) тощо) потребують можливості представляти числові

значення(або значення деякої множини, яку легко пронумерувати) в якості елементів групи G над якою проходять обчислення. У подібних алгоритмах це потребується для здійснення однієї з двох задач: ін'єктивного кодування(вкладання бітового вектора) або хешування(вкладання хешу).

Ін'єктивне кодування – обов'язково має бути можливим відновлення закодованого числового значення з відповідного йому елементу групи, тобто має існувати алгоритм для однозначного розкодування.

Хешування – числове значення не повинно бути відновлюємим, але функція хешування повинна відповідати деяким вимогам [6], наприклад, має поводити себе як випадковий оракл.

Якщо, $G = Z_p$, то саме числове значення може служити як ін'єктивне кодування, а в якості хешування підійде взяття числа за модулем p.

Але якщо G = (група точок ЕК), то для цього випадку немає очевидного рішення. Причиною цьому є те, що не можна просто взяти якесь числове значення(з поля над яким визначена крива) і сказати, що воно точно буде абсцисою(або ординатою) деякої точки на ЕК, бо приблизно тільки половина всіх можливих значень x-координати можуть бути x-координатою якоїсь існуючої точки на ЕК.

Для більшості таких протоколів, особливо тих, стійкість яких базуються на складності обчислення дискретного логарифма, тривіальний підхід до кодування $t \to t \cdot G$, де G - базова точка ЕК, не підійде. Пояснюється це тим, що при такій схемі кодування компрометується дискретний логарифм, відносно G.

Перерахуємо найвідоміші алгоритм для відображення елементів поля в точки ЕК.

Try and increment

Універсальним алгоритмом для усіх кривих вважається ймовірнісний алгоритм, варіації якого представлені в [1, с. 202-203] та [7, с. 8]. Тут надамо спрощену схему другого [8]:

Алгоритм 1.1. Try-and-Increment

Вхід: $u \in F_p$, I

Вихід: точка Q кривої $E(F_q)$

1) $i \leftarrow 0$

2) $x = u + i$

3) Якщо $x^3 + ax + b$ - квадратичний лишок в F_p, повернути $Q = (x, (x^3 + ax + b)^{1/2})$, якщо ні, інкрементувати i та повернутись до кроку 2.

4) Якщо $i \geqslant 2^I$ завершитись невдачею.

5) Кінець

Параметр I пов'язаний з можливою ймовірністю алгоритму завершитись невдачею. I обчислюється попередньо за формулою $I = \lceil \log_2 \log_2(1/\delta) \rceil$, де δ - бажана верхня границя ймовірності, що алгоритм завершиться невдачею. Тому що $x^3 + ax + b$ - квадратичний лишок в приблизно половині випадків, ймовірність того, що для усіх 2^I ітерацій буде давати нелишки дорівнює $1/2^{2^I} < \delta$ [8].

Основним недоліком такого алгоритму є непостійність часу виконання - кількість кроків потрібна для кодування залежить від входу u, що може зробити криптосистему вразловою до "timing attack" - різновиду так званих "side-channel attacks" . Для приблизно половини значень u алгоритм завершиться за 1 крок циклу, для чверті - за 2 кроки тощо. Обчислення символу Лежандра для визначення квадратичної характеристики на кроці a також може бути використано зловмисниками, якщо його час виконання залежить від u (при обчисленні через квадратичний закон взаємності). Можливою мірою захисту може буди навмисне виконання всіх операцій за постійних час, що значно уповільнює роботу алгоритму [8].

Зкручені криві

Ще одним простим, але вже детермінованим алгоритмом є алгоритм, що використовує криву та її зкручуння.

Означення 1.8. Зкручаннями кривої представленої у формі (1.6) називають криву:

$$cv^2 = u^3 + au + b, \quad \left(\frac{c}{p}\right) = -1 \tag{1.27}$$

Таким чином для кожного $x \in F_p$, для якого не існувало відповідної йому точки на (1.6), буде існувати точка на зкрученні (1.27). Але така схема має два серйозних недоліки, поперше, вона подвює час обчислення, бо ті ж самі обчислення доведеться робити на кожній кривій, подруге, доведеться приховувати від зловмисників інформацію про криву, яка використовується [8].

Суперсингулярні криві

Еліптична крива в формі Веєерштрасса $E_{a,b}(F_q) : y^2 = x^3 + ax + b$ називається суперсингулярною, якщо її порядок $N{=}q + 1$. Коли $q \neq 1 \pmod 3$ для кожного $x \in F_q$ кубічний корінь однозначно визначається. Тому з рівняння суперсингулярної кривої

$$y^2 = x^3 + b \tag{1.28}$$

можна з легкістю обчислити всі її точки. Отже, можна ввести таке бієктивне відображення елементів поля F_q в множину точок:

$$f : u \to ((u^2 - b)^{1/3}, u) \tag{1.29}$$

Проблемою для таких кривих є те, що обчислення дискретного логарифму на них відбувається набагато легше, бо для них існує парування(білінійне відображення), що дозволяє звести дискретний логаріфм на еліптичній кривій до логарифму в скінченому полі, це і є так звана MOV-атака [8]. Тому для таких кривих використовуються значно більші розміри параметрів [8].

Shallue-Woestijne алгоритм

В 2006 році був запропонований детермінований алгоритм генерації точок еліптичної кривої за поліноміальний час. Алгоритм базується на рівності, що була запропонована вченим Mariusz Skałba. Наведемо схематично ідею цього алгоритму, описану в [8]:

Нехай крива задається в канонічній формі Веєрштрасса (1.6) над полем F_q, $q{=}p^n$. Тоді існують чотири таких відображення $X_1(t), X_2(t), X_3(t), X_4(t)$, що буде виконуватись наступне:

$$f(X_1(t)) \cdot f(X_3(t)) \cdot f(X_3(t)) = X_4(t)^2, \quad X_4(t) \neq 0 \tag{1.30}$$

Тоді в полі F_q для фіксованого параметру t хоча б один з $f(X_i(t))$, $i \in \{1,2,3\}$ буде квадратичним лишком, отже і абсцисою для кривої (1.6). Знаходження $X_i(t)$ потребує обчислення квадратних коренів. Час виконання цього алгоритму $\mathcal{O}(\log^4 q)$.

Спрощений SWU алгоритм і його узагальнення

В роботі [10] було приведено модифікований варіант попереднього алгоритму кодування для використання в алгоритмі вкладання хешу. Нехай $W(F_p) : y^2 = x^3 + ax + b$, $ab \neq 0$, $p \equiv 3 \pmod 4$. Ця умова зазвичай виконується на практиці, бо вона дозволяє швидко обчислювати квадратні корені.

Твердження 1.2. (Спрощене відображення Уласа)
Нехай:

$$X_2(t) = -\frac{b}{a}\left(1 + \frac{1}{t^4 - t^2}\right)$$
$$X_3(t) = -t^2 X_2(t)$$
$$U(t) = t^3 g(X_2(t))$$

Тоді $U(t)^2 = -g(X_2(t)) \cdot g(X_3(t))$

Доведення. Нехай $g(x) = x^3 + ax + b$, та $u \notin Q_p$. Розглянемо рівняння відносно x-ів

$$g(u \cdot x) = u^3 \cdot g(x) \tag{1.31}$$

Помічаємо, що можно розв'язати дане рівняння відносно x, бо члени третьої степені можна скоротити.

$$\begin{aligned} g(u \cdot x) = u^3 \cdot g(x) &\iff u^3x^3 + aux + b = u^3(x^3 + aux + b) \\ &\iff aux + b = u^3(ax + b) \\ &\iff x = -\frac{b}{a} \cdot \frac{u^3 - 1}{u^3 - u} = -\frac{b}{a}\left(\frac{u^3 - u - 1}{u^3 - u} + \frac{u}{u^3 - u}\right) = \\ &= -\frac{b}{a}\left(1 + \frac{u - 1}{u(u^2 - 1)}\right) = -\frac{b}{a}\left(1 + \frac{1}{(u^2 + u)}\right) \end{aligned} \tag{1.32}$$

Далі, бачимо, що так як u - квадратичний нелишок, то або $g(u \cdot x)$ або $g(x)$ повинен бути квадратичним лишком. Тому або x або $u \cdot x$ повинен бути абсцисою деякої точки на кривій. Більш того, з умови $p \equiv 3 \pmod 4$ випливає те, що -1 - буде квадратним нелишком. Тож, можемо взяти $u = -t^2$. Домножуючи рівняння (1.31) з обох боків на $g(x)$ маємо:

$$\begin{gathered} g(u \cdot x) \cdot g(x) = u^3 \cdot g^2(x) = -(t^3 \cdot g(x))^2 \\ g(-t^2 \cdot x) \cdot g(x) = -(t^3 \cdot g(x))^2 \\ (t^3 \cdot g(x))^2 = -g(x) \cdot g(-t^2 \cdot x) \\ U(t)^2 = -g(X_2(t)) \cdot g(X_3(t)) \end{gathered}$$

Отримали вираз з умови твердження. □

Зауважимо, що t може приймати будь-яке значення окрім {-1,0,1}, які приводять до невизначенності ділення на 0 в формулі (1.32). Відмітимо також, що отримане відображення не є взаємно-однозначною відповідністю: наприклад, елементам поля t і $-t$ буде відповідати одна й та сама точка на ЕК. Ця властивість не порушує вимог хеш-функції, але для ін'єктивного кодування(вкладання бітового вектора), коли обов'язково потрібно мати можливість однозначно і правильно відновити

значення t з відповідної для неї точки, алгоритм у такому вигляді буде незастосовний. Рішення для вкладання бітового вектора в точку ЕК, для подальшого використання в еліптичному аналогу алгоритму Ель-Гамаля, представлене в роботі [3], для ознайомлення розглядається в наступній секції.

Отже має місце відображення:

$$t \to \begin{cases} \infty & \text{if } t \in \{-1, 0, 1\} \\ \left(X_2(t), \sqrt{g\Big(X_2(t)\Big)}\right) & \text{if } \chi\Big(g\Big(X_2(t)\Big)\Big) = 1 \\ \left(X_3(t), \sqrt{g\Big(X_3(t)\Big)}\right) & \text{в іншому випадку} \end{cases}$$

Даний алгоритм було добавлено в "draft-irtf-cfrg-hash-to-curve"(irtf — Internet Research Task Force, cfrg — Crypto Forum Research Group) другої версії в 2018 році, де було запропоновано застосовувати його для вкладання хешу в криву P-256.

В роботі [9] наводиться узагальнення даного алгоритму на випадки, коли $p \not\equiv 3 \pmod 4$. Тоді береться $u = \xi t^2$, де ξ - заздалегідь визначений квадратичний нелишок в F_p. В цьому випадку відображення приймає вигляд:

$$t \to \begin{cases} \infty & \text{if}(-1 \notin Q_p) \wedge (t^2 = -\xi^{-1} \pmod p) \\ \left(x_t, \sqrt{g\Big(x_t\Big)}\right) & \text{if } \chi\Big(g\Big(x_t\Big)\Big) = 1 \\ \left(u_t x_t, \sqrt{u_t^3 g\Big(x_t\Big)}\right) & \text{if } \chi\Big(g\Big(x_t\Big)\Big) = -1 \end{cases}$$

В данному випадку u за означенням може бути тільки нелишком, це означає, що x параметризований лише від половини поля(нелишки складають лише половину поля). Також з рівняння (1.32) бачимо, що має бути виключеним випадок, коли $u_t = -1$, що можливо, тільки якщо $-1 \notin Q_p$ та $t^2 = -\xi^{-1} \pmod p$ [3].

1.8 Інкапсуляція ключа в точку на ЕК

Часним випадком застосуванням вкладання бітових векторів в точки на ЕК може бути певний спрощений варіант так званої “інкапсуляції ключа”.

Означення 1.9. [12] Інкапсуляція ключа - “гібридна” схема шифрування, де з використанням ассиметричної криптографії шифрується ключ, що буде потім використано для подальшої передачі інформації з використанням симетричної криптографії.

В загальному випадку, для інкапсуляції ключа можна використовувати довільний алгоритм ассиметричної криптографії. Наріжним каменем для побудови алгоритму інкапсуляції ключа є детермінований бієктивний алгоритм вкладання бітового вектора в елемент групи на якій працює алгоритм асиметричного шифрування, в нашому випадку - точку на ЕК, та відновлення бітового вектора з цього елементу [12]. До 2016 були відомі лише ймовірнісні алгоритми для такого вкладання, але використання таких алгоритмів придає схемі додакової складності та незручності. Тому дуже важливо мати саме детермінований бієктивний алгоритм.

В роботі [3] узагальнений за [9] для довільного p спрощений SWU алгоритм модифікується таким чином, щоб його можна було використати у схемі інкапсуляції ключа. Ця схема представляє собою передачу по відкритому каналу ключа, зашифрованого еліптичним аналогом алгоритму Ель-Гамаля. Там окремо наводяться два алгоритми - для вкладання і для відновлення елемента поля, представленого у вигляді бітового вектора.

Наведемо тут ці два алгоритми, з зазначенням кількості арифметичних операцій, що потребуються для обчислення на кожному кроці(де можлива двозначність, наприклад, коли в одному кроці описуються два взаємовиключних потоки виконання програми, там буде

наводитись складність цього кроку в найгіршому випадку).

Алгоритм 1.2. [3] Алгоритм вкладання бітового вектора k в точку на ЕК.

Вхід: a,b - параметри кривої; k,$\xi \in F_p,\ \xi \notin Q_p$.

Вихід: $P_k(x_k, y_k), t_1, t_2, t_3$.

1) Обчислити $z_k = k^2\xi$. **(M + S)**

2) Обчислити $t_1 = lsb(z_k)$

3) Обчислити $x_k = -\frac{b}{a}\left(1 + \frac{1}{(z_k^2+z_k)}\right)$. **(2M + S + I)**

4) Обчислити $g_k = x_k^3 + ax_k + b$, $t_2 \longleftarrow 0$. **(2M + S)**

5) Якщо $g_k \notin Q_p$, тоді $x_k \leftarrow z_k x_k$, $t_2 \leftarrow 1$ та $g_k = g_k z_k^3$ **(3M + S + legendre)**

6) Обчислити $y_{1,2} = \sqrt{g_k}$ **(sqrt)**

7) Якщо $lsb(y_k) = 1$ тоді $y_k \leftarrow p - y_k$ (з двох y береться парний)

8) Обчислити $t_3 = lsb(k)$

Проблема відновлення бітового вектора вирішується саме завдяки трьом бітам додаткової інформації t_1, t_2, t_3, що обчислюються на кроках 2, 5, 8.

Алгоритм 1.3. [3] Алгоритм відновлення бітового вектора з точки на ЕК

Вхід: x_k - абсциса точки, t_1, t_2, t_3, $\xi \notin Q_p$

Вихід: k.

1) Якщо $t_2 = 0$, тоді:

$$z_k = \frac{p+1}{2}\left(-1 \pm \sqrt{1 - 4(ab^{-1}x_k + 1)^{-1}}\right) \textbf{ (3M + I + sqrt)}$$

інакше:

$$z_k = \frac{p+1}{2}(ab^{-1}x_k + 1)\left(-1 \pm \sqrt{1 - 4(ab^{-1}x_k + 1)^{-1}}\right) \textbf{ (5M + I + sqrt)}$$

2) З двох отриманих z_k обрати таке, що $lsb(z_k) = t_1$

3) Обчислити $k = \sqrt{\xi^{-1} z_k}$ **(M + sqrt)**

4) Якщо $lsb(k) \neq t_3$, тоді $k = p - k$

Отже, за нашим аналізом цих алгоритмів, що були наведені в роботі [3], складність в гіршому випадку для вкладання буде **(8M + 4S + I + legendre + sqrt)**, для відновлення — **(6M + I + 2sqrt)**.

2 РОЗРОБКА ДЕТЕРМІНОВАНИХ АЛГОРИТМІВ ВКЛАДАННЯ БІТОВОГО ВЕКТОРА В ТОЧКИ ЕЛІПТИЧНИХ КРИВИХ, ПРЕДСТАВЛЕНИХ В РІЗНИХ ФОРМАХ

В даному розділі наводяться нові алгоритми для реалізації вкладання бітового вектора в ЕК представлені в формах Монтгомері, Едвардса і неповній формі Веєрштрасса($a = 0$), побудовані на основі алгоритмів 1.2 і 1.3. Приводиться детальний процес їх побудови з усіма необхідними обгрунтуваннями та математичними викладками необхідними для розуміння. Також виконується аналіз швидкодії цих алгоритмів на основі необхідних для виконання арифметичних операцій.

2.1 Побудова алгоритмів для реалізації вкладання бітового вектора в ЕК в формі Монтгомері

У даному розділі детально описується процес побудови детермінованих алгоритмів кодування(тобто вкладання бітового вектора) в ЕК у формах M та M_1. Кодування в M_1 само по собі не несе ніякої практичної цінності, і потрібно тут лише для того, щоб потім бути використаним в алгоритмі 3).

Підхід базується на адаптації алгоритмів 1.2 і 1.3 до кривих форми M, M_1.

2.1.1 Кодування в форму M

Спочатку рівняння (1.31) застосовуватється до кривої в формі Монтгомері (1.10):

$$\begin{cases} g(u) = u^3 + au^2 + bu \\ g(z_k u) = z_k^3 g(u) \end{cases} \tag{2.1}$$

Тоді отримаємо:

$$\begin{gathered} \cancel{(z_k u)^3} + au^2 z_k^2 + buz_k = \cancel{(z_k^3 u^3)} + au^2 z_k^3 + buz_k^3 \\ \cancel{z_k}(au^2 z_k + bu) = \cancel{z_k}(au^2 z_k^2 + buz_k^2) \\ auz_k + b = auz_k^2 + bz_k^2 \\ auz_k^2 - auz_k = b - bz_k^2 \\ au\cancel{(z_k - 1)}z_k = -b\cancel{(z_k - 1)}(1 + z_k) \\ u_k = u = -\frac{b(1 + z_k)}{az_k} \end{gathered} \tag{2.2}$$

Зауваження. Так як ми ділимо, на $z_k, (z_k - 1)$, то $z_k \notin \{0, 1\}$

Порівнюючи отриману формулу для абсциси з (1.32), бачимо, що отримали навіть простішу формулу ніж в алгоритмі кодуваня для кривої в формі Веєрштрасса. Тож з другого рівняння системи (2.1) бачимо, що або $g(z_k u_k)$ або $g(u_k)$ повинен бути квадратичним лишком, тоді маємо, що або u_k або $z_k u_k$ обов'язково буде абсцисою деякої точки на кривій M. Наступні кроки повторюють оригінальний алгоритм, і для хешування в M отримуємо наступне відображення:

$$k \to \begin{cases} undefined & \text{if } z_k \in \{0, 1\} \\ P_M = (u_M, v_M) = \left(u_k, \sqrt{g\Big(u_k\Big)}\right) & \text{if } \chi\Big(g\Big(u_k\Big)\Big) = 1 \\ P_M = (u_M, v_M) = \left(z_k u_k, \sqrt{z_k^3 g\Big(u_k\Big)}\right) & \text{if } \chi\Big(g\Big(u_k\Big)\Big) = -1 \end{cases} \tag{2.3}$$

Отже, в відображеннях для вкладання хешу в криві W та M відмінність полягає лише у формулах (1.32), (2.2).

Щоб зрозуміти, як буде виглядати алгоритм для вкладання бітового вектора в M, подивимось, яка додаткова інформація нам знадобиться для

однозначного відновлення k з точки P_M. Спочатку, за аналогією з алгоритмом 1.3, треба якимось чином отримати значення z_k з абсцисси u_M. Для цього потрібно знати, за якою з двох формул виконувалося обчислення значень (u_M, v_M), а для цього потрібно знати значення $\chi\Big(g\Big(u_k\Big)\Big)$. Вигляд рівняння $g(u)$ ще може бути відомим розкодувальнику, але саме значення u_k - він ніяк знати не може, тому розкодувальник не зможе, маючи тільки (u_M, v_M), знайти $\chi\Big(g\Big(u_k\Big)\Big)$. Тож, тоді це означає, що при кодуванні окрім самого значення P_M доведеться передавати ще й $t_2 = \Big[\chi\Big(g\Big(u_k\Big)\Big) = 1\Big]$. Отже, маємо тоді два випадки. Якщо $t_2 = 1$, то $\chi\Big(g\Big(u_k\Big)\Big) = 1$, тоді:

$$u_M = u_k$$

Підставляємо u_M в (2.2):

$$\begin{aligned} u_M &= -\frac{b(1+z_k)}{az_k} \\ -b(1+z_k) &= au_M z_k \\ z_k(au_M + b) &= -b \\ z_k &= \frac{-b}{au_M + b} \end{aligned} \tag{2.4}$$

Якщо ж $t_2 = 0$, то $\chi\Big(g\Big(u_k\Big)\Big) = -1$, тоді:

$$u_M = z_k u_k$$

Рівняння (2.2) домножуємо з обох боків на z_k, отримуємо:

$$u_k z_k = -\frac{b(1+z_k)}{a}$$

Підставляємо u_M замість $u_k z_k$:

$$u_M = \frac{-b(1+z_k)}{a}$$
$$z_k = -(\frac{a}{b}u_M + 1) \tag{2.5}$$

Далі після знаходження z_k підставляємо його у формулу $z_k = \xi k^2$, отримуємо:

$$k = \pm\sqrt{z_k \xi^{-1}}$$

Це значить, що для того, щоб при розкодуванні можна було однозначно визначити, що саме було закодовано: k або $-k$, треба при кодуванні передати "біт парності" k, тобто $t_1 = lsb(k)$. Тоді при розкодуванні з двох значень k оберемо те, для якого "біт парності" дорівнює t_1.

Тепер вже бачимо, що для того, щоб однозначно отримати з точки P_M кривої Монтгомері заданої рівнянням (1.10) значення k, потрібно передати ще 2 додаткових біти t_1, t_2. Наведемо тут цілісний алгоритм для кодування в криву M:

Алгоритм 2.1. $k \to (P_M, t_1, t_2)$
Вхід: $k \in F_p$, $M_{a,b}(F_p) : v^2 = u^3 + au^2 + bu$, $\xi \notin Q_p$
Вихід: $P_M \in M_{a,b}(F_p), (t_1, t_2)$

1) обч $z_k = \xi k^2$. **(M + S)**

2) $t_1 = lsb(k)$

3) обч $u_k = -\frac{b(1+z_k)}{az_k}$. **(2M + I)**

4) обч $g(u_k) = u_k^3 + au_k^2 + bu_k$. **(3M + S)**

5) якщо $\chi\Big(g\Big(u_k\Big)\Big) = 1$, тоді : **(legendre)**

5.1) $t_2 = 1$

5.2) обч. $u_M = u_k$, $v_M = +\sqrt{g(u_k)}$. **(sqrt)**

6) інакше:

6.1) $t_2 = 0$

6.2) обч. $u_M = u_k \cdot z_k$, $v_M = +\sqrt{z_k^3 g(u_k)}$. **(3M + S + sqrt)**

7) $P_M = (u_M, v_M)$

8) повернути (P_M, t_1, t_2)

Звернемо увагу на те, що на кроках 5.2 і 6.2, де обчислюється ордината, використвується позначення "$+\sqrt{...}$". Тут це означає обрати "додатній" корень, тобто той з двої коренів, який буде ближчим до 1 ніж до p і відповіндно буде не більший за $\frac{p-1}{2}$. Насправді, тут з тим же успіхом можна було обирати й "від'ємний" корінь, аби тільки кодування було однозначно визначеним. Варто додати, що для однозначного вибору ординати можна використати й іншу стратегію, що базується на тому, що у простому скінченному полі один з квадратних коренів має бути парним, а інший — непарним, і обирати потрібний за бітом парності — найменш значущим бітом(lsb).

В роботі [9] простежується ідея використовувати "двозначніть" ординати для зменшеня (хоч і незначного — всього на 1 біт) передаваємої інформації: в "знак" ординати кодується потрібний біт(в нашому випадку це був би t_2) інформації.

Алгоритм для розкодування відповідно тоді прийме наступний вигляд:

Алгоритм 2.2. $(P_M, t_1, t_2) \to k$
Вхід: $P_M \in M_{a,b}(F_p), (t_1, t_2)$, $M_{a,b}(F_p) : v^2 = u^3 + au^2 + bu$, $\xi \notin Q_p$
Вихід: $k \in F_p$

1) якщо $t_2 = 1$, тоді:

1.1) обч. $z_k = \frac{-b}{au_M + b}$. **(3M + I + sqrt)**

2) інакше:

2.1) обч. $z_k = -(\frac{a}{b}u_M + 1)$. **(5M + I + sqrt)**

3) обч. $k = \sqrt{z_k \xi^{-1}}$. **(M + sqrt)**

4) якщо $t_1 \neq lsb(k)$, тоді $k = p - k$:

5) повернути k

Отже, бачимо, що для владання ключа в M потребується **(9M + 3S + I + legendre + sqrt)**, для відновлення — **(2M + I + sqrt)**. Якщо порівняти отримані оцінки з оцінками для алгоритмів 1.2, 1.3, то побачимо, що хоч при вкладанні різниця не велика, але при відновленні кількість операцій для M є меншою на аж 4 множення. Крім того, кількість бітів необхідних для відновлення менше на 1, що хоч і складно назвати істотою перевагою, все одно заслуговує на те, щоб бути згаданим.

2.1.2 Кодування в форму M_1

Нехай крива задана в (1.24) та $\left(\frac{1-d}{p}\right) = -1$. Перепозначимо $\mu = \frac{1}{1-d}$, $\nu = 2\frac{1-d}{1+d}$, тоді отримаємо:

$$\mu\widetilde{v}^2 = u^3 + \nu u^2 + u \tag{2.6}$$

Нехай

$$g(u) = u^3 + \nu u^2 + u \tag{2.7}$$

тоді:

$$\mu\widetilde{v}^2 = g(u)$$

$$\begin{cases} \mu \notin Q_p \\ \widetilde{v}^2 \in Q_p \end{cases} \quad \Rightarrow \quad g(u) \notin Q_p \tag{2.8}$$

Бачимо, що (2.7) є окремим випадком правої частини рівняння (1.10), коли $a = \nu, b = 1$, отже можемо $g(u)$ з (2.7) підставити в систему (2.1). Тоді отримуємо:

$$u_k = u = -\frac{b(1+z_k)}{az_k} = -\frac{1+z_k}{\nu z_k} \tag{2.9}$$

Отримавши u_k, що задовольняє (2.1), далі можемо провести схожі з 2.1 дії, але з однією відмінністю: так як з (2.8) випливає, що $g(u)$ має бути

нелишком, то нас буде цікавити та пара з $(u_k, g_k = g(u_k))$, $(u_k \cdot z_k, g_k = g(u_k \cdot z_k))$, в якій g_k буде нелишком (навідміну від алгоритма 2.1, де нас, навпаки, цікавив саме лишок). Отже, знову маємо два випадки:

$$\begin{cases} g(u_k) \notin Q_p \implies u_{M_1} = u_k - \text{абсциса} \\ g(u_k) \in Q_p \implies g(z_k \cdot u_k) \notin Q_p \implies u_{M_1} = z_k \cdot u_k - \text{абсциса} \end{cases} \tag{2.10}$$

Далі, ордината з рівняння (2.6) знаходиться, як:

$$\widetilde{v} = \sqrt{\mu^{-1} \cdot g(u_{M_1})}$$

Тоді:

$$\begin{cases} g(u_k) \notin Q_p \implies \widetilde{v}_{M_1} = \sqrt{\mu^{-1} \cdot g(u_k)} \\ g(u_k) \in Q_p \implies \widetilde{v}_{M_1} = \sqrt{\mu^{-1} \cdot g(z_k \cdot u_k)} = +\sqrt{\mu^{-1} \cdot z_k^3 \cdot g(u_k)} \end{cases} \tag{2.11}$$

Відображення для вкладання хешу в криву M_1 буде мати наступний вигляд:

$$k \to \begin{cases} \infty & \text{if } z_k \in \{0, 1\} \\ P_{M_1} = (u_{M_1}, \widetilde{v}_{M_1}) = \left(u_k, \sqrt{\mu^{-1} g\Big(u_k\Big)}\right) & \text{if } \chi\Big(g\Big(u_k\Big)\Big) = -1 \\ P_{M_1} = (u_{M_1}, \widetilde{v}_{M_1}) = \left(z_k u_k, \sqrt{\mu^{-1} z_k^3 g\Big(u_k\Big)}\right) & \text{if } \chi\Big(g\Big(u_k\Big)\Big) = 1 \end{cases} \tag{2.12}$$

Алгоритми для кодування та розкодування для вкладання бітового вектора у M_1, будуються аналогічно до 2.1: 2.2:

Алгоритм 2.3. $k \to (P_{M_1}, t_1, t_2)$

Вхід: $k \in F_p$, $M_1(F_p) : \mu\widetilde{v}^2 = u^3 + \nu u^2 + u$, $\xi \notin Q_p$

Вихід: $P_{M_1} \in M_1(F_p), (t_1, t_2)$

1) обч $z_k = \xi k^2$. **(M + S)**

2) $t_1 = lsb(k)$

3) обч $u_k = -\frac{1+z_k}{\nu z_k}$. **(M + I)**

4) обч $g(u_k) = u_k^3 + \nu u_k^2 + u_k$. **(2M + S)**

5) якщо $\chi\Big(g\Big(u_k\Big)\Big) = -1$, тоді : **(legendre)**

5.1) $t_2 = 0$

5.2) обч. $u_{M_1} = u_k$, $\widetilde{v}_{M_1} = +\sqrt{\mu^{-1} \cdot g(u_k)}$. **(M + sqrt)**

6) інакше:

6.1) $t_2 = 1$

6.2) обч. $u_{M_1} = u_k \cdot z_k$, $\widetilde{v}_{M_1} = +\sqrt{\mu^{-1} \cdot z_k^3 \cdot g(u_k)}$. **(4M + S + sqrt)**

7) $P_{M_1} = (u_{M_1}, v_{M_1})$

8) повернути (P_{M_1}, t_1, t_2)

При розкодуванні z_k знаходиться за тими ж формулами, що й в алгоритмі розкодування для M, але з $a = \nu, b = 1$. І, якщо в алгоритмі для M для випадку $t_2 = 0$ використовувалась формула (2.5), а для $t_2 = 1$ - (2.4), то для M_1 буде все навпаки. Тут для $t_2 = 0$:

$$z_k = \frac{-1}{\nu u_M + 1}$$

А для $t_2 = 1$:

$$z_k = -(\nu u_M + 1)$$

Алгоритм 2.4. $(P_{M_1}, t_1, t_2) \to k$
Вхід: $P_{M_1} \in M_1(F_p), (t_1, t_2)$, $M_1(F_p) : \mu\widetilde{v}^2 = u^3 + \nu u^2 + u$, $\xi \notin Q_p$
Вихід: $k \in F_p$

1) якщо $t_2 = 0$, тоді:

1.1) обч. $z_k = \frac{-1}{\nu u_M + 1}$. **(M + I)**

2) інакше:

2.1) обч. $z_k = -(\nu u_M + 1)$. **(M)**

3) обч. $k = \sqrt{z_k \xi^{-1}}$. **(M + sqrt)**

4) якщо $t_1 \neq lsb(k)$, тоді $k = p - k$:

5) повернути k

В результаті бачимо, що для вкладання бітового вектора в M_1 потребується **(8M + 3S + I + legendre + sqrt)**, для відновлення — **(2M + I + sqrt)**.

Отже, отримали відображення (2.12) для вкладання хешу і алгоритми 2.1, 2.2, 2.3, 2.4 для вкладання бітового вектора в криві M, M_1. Приємною несподіванкою при знаходженні формули (2.2) - формули для обчислення потенційної абсциси закодованої точки в криву M і M_1 - виявилося те, що ця формула опинилася простішою аніж її альтернатива для кривої W (1.32). Конкретніше, вона не містить квадрату від z_k, що дозволяє знаходити з неї z_k однозначно (бо не треба обчислювати квадратний корінь), це економить нам 1 біт додаткової інформації необхідної для однозначного розкодування в алгоритмах 2.2, 2.2.

2.2 Побудова алгоритмів для реалізації вкладання бітового вектора в ЕК в формі Едвардса

В нас не вийшло застосувати $g(z_k \cdot x) = z_k^3 g(x)$ для кривої в формі Едвардса, була зроблена спроба, яка не принесла бажаних результатів. Причиною цьому є безпосередньо вигляд рівняння ЕК в формі Едвардса, в якій присутні доданки 2их і 4тих степенів, що приводить до певних складоностей.

Тому було прийнято рішенна змінити підхід до задачі: замість кодування $k \in F_p$ безпосердньо в криву Едвардса, використати концепцію ізоморфізму еліптичних кривих і спочатку закодувати k в точку P_X деякої кривої X, яка б була ізоморфна до вхідної кривої. Потім використовуючи "зворотне"перетворення $P_X \to P_E$ отримати точку P_E.

В якості кандидатів на X були розглянуті криві W(Веєрштрасса) і M(Монтгомері). Так як перетворення $E \to W$ в якомога простішому варіанті [2] являє собою послідовність перетворень $E \to M \to W$ (тобто містить в собі перетворення $E \to M$ як одну з двох складових частин) - воно реалізується складніше ніж перетворення $E \to M$. Тому наш вибір зупинився на M(Монтгомері) в якості X.

Наведемо тут загальну схему алгоритму:

Алгоритм 2.5. Загальна схема кодування $k \to P_E$
Вхід: $k \in F_p$, $E(F_p) : x^2 + y^2 = 1 + dx^2y^2$
Вихід: $P_E \in E(F_p)$

1) $E(F_p) \to M_1(F_p)$ — отримати рівняння ізоморфної ЕК в формі M_1.

2) if $\left(\frac{1-d}{p}\right) = 1$, then:

   2.1) $M_1(F_p) \to M(F_p)$

   2.2) $k \to P_M$ - закодувати k в криву $M(F_p)$

   2.3) $P_M \to P_{M_1}$

   2.4) $P_{M_1} \to P_E$

3) else:

   3.1) $k \to P_{M_1}$

   3.2) $P_{M_1} \to P_E$

4) return P_E

У перетворенні $E \to M$ в залежності від квадратичної характеристики $\frac{1}{1-d}$ можливі два випадки. В обох випадках E можна спочатку привести до M_1, тому на кроці 1 виконується це перетворення описане в підрозділі 1.6.1 - виконуючи заміну:

$$x = 2\frac{u}{v}, \quad y = \frac{u-1}{u+1} \tag{2.13}$$

одержуємо рівняння ЕК в формі M_1, яка ізоморфна вхідній кривій E:

$$M_1 : \frac{\widetilde{v}^2}{1-d} = u^3 + 2\frac{1+d}{1-d}u^2 + u$$

Далі на кроці 2 обчислюється квадратична характеристика $\left(\frac{(1-d)^{-1}}{p}\right) = \left(\frac{(1-d)}{p}\right)$. У випадку, коли $\frac{1}{1-d} \in Q_p$, заміною $v^2 = \frac{1}{1-d}\widetilde{v}^2$ рівняння приводиться до форми M, що детально описано в підрозділі (1.6.2):

$$M : v^2 = u^3 + 2\frac{1+d}{1-d}u^2 + u \tag{2.14}$$

Для кривої цієї форми алгоритм кодування вже описаний в підрозділі 2.1.1, тому на кроці 2.2 за цим алгоритмом проводиться кодування елемента k в (P_M, t_1, t_2) кривої M, в даному випадку $a = 2\frac{1+d}{1-d}$, $b = 1$. На кроках 2.3 - 2.4 відбувається "повернення" з кривої M до кривої E(тобто повернення з точки P_M до відповідної їй точки P_E) за допомогою замін, зворотних до тих, що були проведені на кроках 1, 2.1. Для наочності об'єднаємо їх в одну:

$$\begin{cases} \begin{cases} v^2 = \dfrac{1}{1-d}\widetilde{v}^2 \\ u = u \end{cases} \\ \begin{cases} x = 2\dfrac{u}{\widetilde{v}} \\ y = \dfrac{u-1}{u+1} \end{cases} \end{cases} \implies \begin{cases} \widetilde{v} = \sqrt{1-d}\cdot v \\ u = u \end{cases} \implies \begin{cases} x = \dfrac{2}{\sqrt{1-d}}\cdot\dfrac{u}{v} \\ y = \dfrac{u-1}{u+1} \end{cases} \tag{2.15}$$

Отже для випадку, коли $\frac{1}{1-d} \in Q_p$, в результаті маємо точку $P_E = (x_E, y_E) = (\frac{2}{\sqrt{1-d}}\cdot\frac{u_M}{v_M}, \frac{u_M-1}{u_M+1})$ та 2 біти t_1, t_2.

Якщо ж $\frac{1}{1-d} \notin Q_p$, то заміну $v^2 = \frac{1}{1-d}\widetilde{v}^2 \iff v = \sqrt{\frac{1}{1-d}}\widetilde{v}$ виконати не вдасться(бо $\sqrt{\frac{1}{1-d}}$ не існує), відповідно привести до форми M тим же шляхом, що й для випадку, коли $\frac{1}{1-d} \in Q_p$, не вийде. Тому безпосередньо до кривої в формі M_1(коли $\frac{1}{1-d} \notin Q_p$) було предствалено алгоритм 2.3, що дозволить нам закодувати в криву M_1 елемент k і отримати відповідну трійку значень (P_{M_1}, t_1, t_2). На кроці 3.1 використовується даний алгоритм,

з значеннями параметрів кривої $\mu = \frac{1}{1-d}$, $\nu = 2\frac{1+d}{1-d}$.

Далі, отримавиши значення P_{M_1}, за формулами (2.13), на кроці 3.2 відбувається повернення з точки P_{M1} до відповідної їй точки P_E на кривій E.

Умова представлена на кроці 2 залежить тільки від параметру d самої кривої і характеристики поля p. На практиці зазвичай робота ведеться з визначеною і затвердженною науковим суспільством множиною ЕК, яку визнають стійкими та придатними для використання в криптографії. Тому це значення можна завчасно передобчислити для кожної такої кривої і відповідної характеристики, щоб не додавати зайвих обчислень до алгоритму.

Теорема (1.3) стверджує, що умова $\left(\frac{1-d}{p}\right) = 1$ для (1.14)(а отже і для (1.20)) еквівалентна існуванню чотирьох точок восьмого порядку. Тому, замість обчислення $\left(\frac{1-d}{p}\right) = 1$ достатньо знати порядок кривої, а конкретніше, чи існують ці точки 4 точки.

Також, необхідно відмітити, що значення $(u+1), \widetilde{v}$ з (2.13) та $(u+1), v$ з (2.15) не можуть приймати нульові значення, бо це приводитиме до невизначенності ділання на 0 при переходах $P_M \to P_E$, $P_{M_1} \to P_E$ в алгоритмі відновлення 2.8. Тому в алгоритмі кодування в E треба буде передбачити ці випадки. Цей частний випадок не є критичним для кодування, бо ми втрачаємо(не зможемо закодувати) ще в гіршому випадку 4(по 2 значення k на кожний випадок) елементи поля, що зневажливо мало у порівнянні з величиною p, яку використовють на практиці.

Детально розпишемо цей алгоритм:

Алгоритм 2.6. $k \to (P_E, t_1, t_2)$
Вхід: $k \in F_p$, $E(F_p) : x^2 + y^2 = 1 + dx^2y^2$, передобчислене $\gamma = \left[\left(\frac{1-d}{p}\right) = 1\right]$
Вихід: $P_E \in E(F_p)$

1) if $\gamma = 1$, then:

1.1) обч $z_k = \xi k^2$

1.2) $t_1 = lsb(k)$

1.3) обч $u_k = -\frac{(1+z_k)}{2\frac{1+d}{1-d}z_k}$

1.4) обч $g(u_k) = u_k^3 + 2\frac{1+d}{1-d}u_k^2 + u_k$

1.5) якщо $\chi\Big(g\Big(u_k\Big)\Big) = 1$, тоді :

1.5.1) $t_2 = 1$

1.5.2) обч. $u_M = u_k$, $v_M = +\sqrt{g(u_k)}$

1.6) інакше:

1.6.1) $t_2 = 0$

1.6.2) обч. $u_M = u_k \cdot z_k$, $v_M = +\sqrt{z_k^3 g(u_k)}$

1.7) $P_M = (u_M, v_M)$

1.8) якщо $u_M = -1$ або $v_M = 0$, тоді повернути -1(“невдачу”)

1.9) $P_E = (x_E, x_E) = (\frac{2}{\sqrt{1-d}} \cdot \frac{u_M}{v_M}, \frac{u_M - 1}{u_M + 1})$. **(3M + 2I)**

2) інакше:

2.1) обч $z_k = \xi k^2$

2.2) $t_1 = lsb(k)$

2.3) обч $u_k = -\frac{1+z_k}{2\frac{1+d}{1-d}z_k}$

2.4) обч $g(u_k) = u_k^3 + 2\frac{1+d}{1-d}u_k^2 + u_k$

2.5) якщо $\chi\Big(g\Big(u_k\Big)\Big) = -1$, тоді :

2.5.1) $t_2 = 0$

2.5.2) обч. $u_{M_1} = u_k$, $\widetilde{v}_{M_1} = +\sqrt{(1-d) \cdot g(u_k)}$

2.6) інакше:

2.6.1) $t_2 = 1$

2.6.2) обч. $u_{M_1} = u_k \cdot z_k$, $\widetilde{v}_{M_1} = +\sqrt{(1-d) \cdot z_k^3 \cdot g(u_k)}$

2.7) $P_{M_1} = (u_{M_1}, \widetilde{v}_{M_1})$

2.8) якщо $u_{M_1} = -1$ або $\widetilde{v}_{M_1} = 0$, тоді повернути -1(“невдачу”)

2.9) $P_E = (x_E, y_E) = (2\frac{u_{M_1}}{\widetilde{v}_{M_1}}, \frac{u_{M_1} - 1}{u_{M_1} + 1})$. **(3M + 2I)**

3) повернути (P_E, t_1, t_2)

Кроками 1.1 - 1.7 даний алгоритм повторює алгоритм вкладання для M, а на кроці 1.9 додаються операції **(3M + 2I)**. Кроками 2.1 - 2.7 - повторює алгоритм вкладання в M_1, а на кроці 2.9 додається також **(3M + 2I)**. З цього випливає, що коли $\gamma = 1$, складність вкладання в E буде **(12M + 3S + 3I + legendre + sqrt)**, а коли $\gamma = 0$ — **(11M + 3S + 3I + legendre + sqrt)**.

Отже отримали алгоритм, що дозволяє детерміновано кодувати в E. Групи кроків 1.1-1.6, 2.1-2.6 можна об'єднати, представивши їх, як виклики відповідних підпроцедур $k \to (P_M, t_1, t_2)$ і $k \to (P_{M_1}, t_1, t_2)$, але тут вони наводяться в розширеному вигляді для наочності.

В загальному схематичному вигляді алгоритм для декодування буде мати наступний вигляд:

Алгоритм 2.7. Загальна схема відновлення бітового вектора P_E
Вхід: $P_E \in E(F_p)$, (t_1, t_2), $E(F_p) : x^2 + y^2 = 1 + dx^2y^2$, передобчислене $\gamma = \left[\left(\frac{1-d}{p}\right) = 1\right]$
Вихід: $k \in F_p$

1) якщо $\gamma = 1$:

1.1) $P_E \to P_M$

1.2) $P_M \to k$

2) інакше:

2.1) $P_E \to P_{M_1}$

2.2) $P_{M_1} \to k$

3) повернути k

Перетворення $P_E \to P_M$ є оберненим перетворенням до тих, що були наведені в системі (2.15), воно буде виглядати таким чином:

$$\begin{cases} u = \dfrac{1+y}{1-y} \\ v = \dfrac{1}{\sqrt{1-d}} \cdot \dfrac{2}{x} \cdot \dfrac{1+y}{1-y} \end{cases} \tag{2.16}$$

А перетворення $P_E \to P_{M1}$, обернене до (1.23):

$$\begin{cases} u = \dfrac{1+y}{1-y} \\ \widetilde{v} = \dfrac{2}{x} \cdot \dfrac{1+y}{1-y} \end{cases} \tag{2.17}$$

При декодуванні $P_M \to k$ в якості параметрів кривої M, як і при кодуванні, будуть передаватись $a = 2\frac{1+d}{1-d}$, $b = 1$. При декодуванні $P_{M_1} \to k$ $\mu = \frac{1}{1-d}$, $\nu = 2\frac{1+d}{1-d}$, як і при кодуванні в P_{M_1}.

Тепер вже , визначившись з загальною схемою, можемо цілістно оформити алгоритм для декодування для кривої Едвардса.

Алгоритм 2.8. $(P_E, t_1, t_2) \to k$

Вхід: $P_E \in E(F_p)$, (t_1, t_2), $E(F_p) : x^2 + y^2 = 1 + dx^2y^2$, передобчислене $\gamma = \left[\left(\frac{1-d}{p}\right) = 1\right]$

Вихід: $k \in F_p$

1) якщо $\gamma = 1$:

1.1) $P_M = (u_M, v_M)$, обч. $u_M = \frac{1+y}{1-y}$, $v_M = \frac{1}{\sqrt{1-d}} \cdot \frac{2}{x} \cdot \frac{1+y}{1-y}$. **(3M + 2I)**

1.2) якщо $t_2 = 1$, то:

1.2.1) $z_k = \frac{-1}{2\frac{1+d}{1-d}u_M + 1}$

1.3) інакше:

1.3.1) $z_k = -(2\frac{1+d}{1-d}u_M + 1)$

2) інакше:

2.1) $P_{M_1} = (u_{M_1}, \widetilde{v}_{M_1})$, обч. $u_{M_1} = \frac{1+y}{1-y}$, $\widetilde{v}_{M_1} = \frac{2}{x} \cdot \frac{1+y}{1-y}$. **(3M + 2I)**

2.2) якщо $t_2 = 0$, то:

2.1.1) $z_k = \frac{-1}{2\frac{1+d}{1-d}u_{M_1} + 1}$

2.3) інакше:

2.2.1) $z_k = -(2\frac{1+d}{1-d}u_{M_1} + 1)$

3) обч. $k = \sqrt{z_k \xi^{-1}}$

4) якщо $t_1 \neq lsb(k)$, тоді $k = p - k$:

5) повернути k

Як бачимо, різниця між випадками з різними γ є лише на кроках 1.1,2.1 і в умовах 1.2, 2.2. Кроки 1.2 - 1.3, 3 - 5 є такими ж, що й для відновлення з M, але тут завдяки тому, що $b = 1$, множень буде на 1 менше. Тож, для випадки $\gamma = 1$ складність буде **(4M + 3I + sqrt)**. Кроки 2.2 - 2.3, 3 - 5 є такими ж, що й для відновлення з M_1, тому складність для $\gamma = 0$ буде **(5M + 3I + sqrt)**.

2.3 Побудова алгоритмів для реалізації вкладання бітового вектора в ЕК в неповній формі Веєрштрасса

Якщо уважно придивитися до формули (1.32), яка використовується в алгоритмах 1.2, 1.3, можна побачити, що ці алгоритми є незастосовними до випадків, коли $a = 0$ або $b = 0$ — тобто, коли крива Веєрштрасса є неповною. Бо, коли $a = 0$, це приводить до невизначеності ділення на 0, а коли $b = 0$ — до того, що всі значення $x_k = 0$ — неприпустимої для кодування ситуації. Тому для неповних кривих Веєрштрасса потрібен інший підхід.

Криві в неповній формі Веєрштрасса зараз можна вважати дуже актуальними, бо, наприклад, крива secp256k1, яка використовується в Bitcoin, має форму $y^2 = x^3 + 7$, крива BLS12-381, яка використовується для zk-SNARK, має вигляд $y^2 = x^3 + 4$. Зараз ці технології є дуже поширеними і привертають увагу наукової спільноти.

В роботі [9] проблема $a = 0$ для хешування в криву secp256k1, сімейство кривих BLS(Barreto-Lynn-Scott) і Barreto-Naehrig, вирішується за допомогою "непрямого" відображення до кривої C', ізогенної до вхідної кривої C, в якої параметри a, b не нульові. Спочатку хешування за узагальненим Simplified SWU алгоритмом проводиться у точку кривої C', з якої потім оберненим перетворенням повертається бажана точка на C.

В даному підрозділі пропонується алгоритми для вкладання бітового

вектора і вкладання хешу побудувати іншим чином: замість використання ізогенії C до деякої іншої кривої C'(також форми Веєрштрасса, але вже без нульових параметрів), використати ізоморфність форм Веєрштрасса і Монтгомері.

Нехай вхідна крива представлена в формі (1.26):

$$y^2 = x^3 + c \tag{2.18}$$

Спочатку треба знайти ізоморфну до вхідної кривої криву в формі Монтгомері, за процесом описаним в підрозділі(1.6.3), що можливо, лише коли $(-c)$ — кубічний лишок. І якщо $(-c)$ — кубічний лишок(тобто $-c = s^3$), то за допомогою перетворень (1.6.3), отримаємо рівняння в формі Монтгомері (1.10), з параметрами $a = 3s, b = 3s^2$, яке назвемо M':

$$M' : v^2 = u^3 + 3su^2 + 3s^2u \tag{2.19}$$

Звернемо увагу на те, що кубічний корінь s не може бути рівним 0, бо якщо $s = 0$, то $c = 0$, а рівняння $y^2 = x^3$ — вже не вважається еліптичною кривою [9].

Тож, якщо кожен з параметрів кривої M' не нульовий(в результаті чого в формулі (2.2) не з'являється ніяких невизначеностей ділення на 0 тощо), то в теорії ніщо не забороняє нам застосувати алгоритми описані в підрозділі 2.1.1 до кривої M'.

Отже, далі до форми M' можна без ніяких перешкод застосувати для вкладання хешу — відображення 2.3, для вкладання бітового вектора — алгоритми 2.1, 2.2.

При вкладанні ключа формула для потенційної абсцисси (2.2) кривої M' прийме наступний вигляд(підставляємо в неї $a = 3s, b = 3s^2$):

$$u_k = -\frac{b(1+z_k)}{az_k} = u_k = -\frac{\cancel{3}s^{\cancel{2}}(1+z_k)}{\cancel{3s}z_k} = -\frac{s(1+z_k)}{z_k}$$

Алгоритм 2.9. $k \to (P_W, t_1, t_2)$

Вхід: $k \in F_p$, $W(F_p) : y^3 = x^3 + c$, $(-c) = s^3$

Вихід: $P_W \in W(F_p)$

1) обч $z_k = \xi k^2$

2) $t_1 = lsb(k)$

3) обч $u_k = -\frac{s(1+z_k)}{z_k}$

4) обч $g(u_k) = u_k^3 + 3su_k^2 + 3s^2u_k$

5) якщо $\chi\Big(g\Big(u_k\Big)\Big) = 1$, тоді :

5.1) $t_2 = 1$

5.2) обч. $u_{M'} = u_k$, $v_{M'} = +\sqrt{g(u_k)}$

6) інакше:

6.1) $t_2 = 0$

6.2) обч. $u_{M'} = u_k \cdot z_k$, $v_{M'} = +\sqrt{z_k^3 g(u_k)}$

7) $P_M = (u_{M'}, v_{M'})$

8) $P_W = (x_W, x_W) = (u_{M'} + s, v_{M'})$

При відновленні ключа тоді, будуть мати місце такі формули(аналогічно знаходимо їх підставляючи $a = 3s, b = 3s^2$ в (2.4), (2.5)):

При $t_2 = 1$:

$$z_k = \frac{-b}{au_M + b} = \frac{-3s^2}{3su_M + 3s^2} = \frac{\cancel{3s}(-s)}{\cancel{3s}(u_M + s)} = \frac{-s}{u_M + s}$$

При $t_2 = 0$:

$$z_k = -(\frac{a}{b}u_M + 1) = -(\frac{3s}{3s^2}u_M + 1) = -(\frac{1}{s}u_M + 1)$$

Алгоритм 2.10. $(P_W, t_1, t_2) \to k$

Вхід: $P_W \in W(F_p), t_1, t_2$, $W(F_p) : y^3 = x^3 + c$, $(-c) = s^3$

Вихід: $k \in F_p$

1) $P_M = (u_{M'}, v_{M'})$, обч. $u_{M'} = x - s$, $v_{M'} = y$

2) якщо $t_2 = 1$, то:

2.1) $z_k = \frac{-s}{u_{M'}+s}$

3) інакше:

3.1) $z_k = -(\frac{1}{s}u_{M'} + 1)$

4) обч. $k = \sqrt{z_k \xi^{-1}}$

5) якщо $t_1 \neq lsb(k)$, тоді $k = p - k$:

6) повернути k

В результаті маємо, що для вкладання бітового вектора в W при $(a = 0)$ потребується **(9M + 3S + I + legendre + sqrt)**, для відновлення — **(2M + I + sqrt)**. Бачимо, що відновлення за даним алгоритмом буде швидше ніж відновлення з звичайної W $(a, b \neq 0)$ за алгоритмом 1.3 на 4 операції множення.

2.4 Порівняння складностей алгоритмів

Порівнємо складності алгоритмів з роботи [3] та наших алгоритмів:

Назва алгоритму	Складність в операціях
вкладання бітового вектора в ЕК в формі Веєрштрасса [3]	**(8M + 4S + I + legendre + sqrt)**
відновлення бітового вектора з ЕК в формі Веєрштрасса [3]	**(6M + I + 2sqrt)**
вкладання бітового вектора в ЕК в формі Монтгомері	**(9M + 3S + I + legendre + sqrt)**

відновлення бітового вектора з ЕК в формі Монтгомері	**(2M + I + sqrt)**
вкладання бітового вектора в ЕК в формі Едвардса	**(12M + 3S + 3I + legendre + sqrt)**
відновлення бітового вектора з ЕК в формі Едвардса	**(5M + 3I + sqrt)**
вкладання бітового вектора в ЕК в неповній($a = 0$) формі Веєрштрасса	**(9M + 3S + I + legendre + sqrt)**
відновлення бітового вектора з ЕК в неповній($a = 0$) формі Веєрштрасса	**(2M + I + sqrt)**

Таблиця 2.1 – Порівняння складності отриманих алгоритмів в арифметичних операціях

Бачимо, що найшвидшими за таблицею є алгоритми для кривих в формі Монтгомері та неповній формі Веєрштрасса($a = 0$). Усі нові презентовані в даній роботі алгоритми при відновленні використовують на 1 операцію обчислення квадратного кореня менше(що є доволі затратною операцією в скінченному полі) ніж алгоритм представлений в роботі [3], а конкретно алгоритми для Монгомері і неповної форми Веєрштрасса використовують ще на 4 множення менше. Нажаль, для

Едвардса алгоритми потребують обчисленяя аж 3ьох зворотних елементів(що, як і корінь, є доволі затратною операцією), але це все одно є непоганим результатом.

ВИСНОВКИ

Отже, ми розробили алгоритми вкладення бітового вектора у точку ЕК, заданої у формах Монтгомері, Едвардса та неповної форми Веєрштрасса. І тепер для того, щоб вкласти бітовий вектор в ЕК в цих формах, не обов'язково переходити з них до форми Веєрштрасса, щоб застосувати алгоритми 1.2, 1.3 — можна відразу безпосердньо використати побудовані в даній роботі алгоритми.

В результаті аналізу, можемо сказати, що час роботи даних алгоритмів поліноміальний, усі вони є детермінованими. Отримані алгоритми відновлення 2.2 для форми Монтгомері та 2.10 і неповної Веєрштрасса опинилися на 4 множення більш швидкими ніж 1.3, що є помітним покращенням. Усі побудовані алгоритми відновлення використовують на 1 операцію знаходження квадратного кореня менше ніж 1.3. Тому можемо підсумувати, що отримані алгоритми є ефективними, роботоздатними та можуть бути застосованими на практиці.

Тепер на основі побудованих алгоритмів вкладання вектора можна будувати криптосистеми шифрування типу Ель-Гамаля, схеми інкапсуляції ключів над еліптичними кривими, пердставленими в формах Монтгомері, Едвардса, неповній Веєрштрасса, більше ефективно ніж це було можливо раніше.

ПЕРЕЛІК ПОСИЛАНЬ

1. Коблиц Н. Курс теории чисел и криптографии. / Перев. с англ. -Москва: Научное изд-во ТВП, 2001. - ISBN 5-85484-014-6.

2. Эллиптические кривые в форме Эдвардса и криптография: монография. –Киев:ІВЦ «Видавництво «Політехника»», 2017. –272с.

3. Tsygankova O.V., ANALYZING OF POSSIBILITY OF USING ELGAMAL ALGORITHM WITH DETERMINISTIC EMBEDDING FOR KEY ENCAPSULATION. Department of Mathematical Methods of Information Protection, National Technical University of Ukraine "Igor Sikorsky KPI", Institute of Physics and Technology.- UDC 621.391.15:519.7

4. Bernstein D.J., Lange T.Faster Addition and Doubling on Elliptic Curves. Advances in Cryptology — ASIACRYPT'2007. Lect. NotesComp. Sci. V. 4833. Berlin: Springer, 2007. PP. 29–50.

5. Edwards H.M. A normal form for elliptic curves. Bulletin of the American Mathematical Society, Volume 44, Number 3, July 2007, PP.393-422.

6. Pierre-Alain Fouque, Mehdi Tibouchi. Deterministic Encoding and Hashing to Odd Hyperelliptic Curves.

7. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing.J. Cryptology, 17(4):297–319,2004.

8. Thomas Icart. How to Hash into Elliptic Curves

9. Wahby, Riad S. and Dan Boneh. "Fast and simple constant-time hashing to the BLS12-381 elliptic curve." IACR CryptologyePrintArchive 2019 (2019): 403.

10. Eric Brier, Jean-Sébastien Coron, Thomas Icart, David Madore, Hugues Ran-driam, and Mehdi Tibouchi. Efficient indifferentiable hashing into ordinary elliptic curves. In Tal Rabin, editor, CRYPTO 2010, volume 6223 of LNCS, pages 237–254. Springer, Heidelberg, August 2010.

11. L.C.Washington. Elliptic Curvres. Number Theory and Cryptography. Second Edition. CRC Press, 2008.

12. V.Shoup. A Proposal for an ISO Standard for Public Key Encryption. Preprint, December 2001.

13. Koblitz, N. (1987). "Elliptic curve cryptosystems". Mathematics of Computation. 48 (177): 203–209. doi:10.2307/2007884. JSTOR 2007884.

14. V. Miller, Use of elliptic curves in cryptography, Advances in cryptology—CRYPTO 85, Springer Lecture Notes in Computer Science vol 218, 1985.

15. I. F. Blake, G. Seroussi, and N. P. Smart.Elliptic curves in cryptography, volume 265 of London Mathematical Society Lecture Note Series.

16. Henri Cohen, Atsuko Miyaji, and Takatoshi Ono. Efficient Elliptic Curve Exponentiation Using Mixed Coordinates.

17. Parker D. Elliptic curves and Lenstra's factorization algorithm. University of Chicago: REU 2014.

18. Bernstein, Daniel J. Curve25519: New Diffie-Hellman Speed Records. Public Key Cryptography. Lecture Notes in Computer Science. 3958. New York: Springer.

19. Roberto M. Avanzi, Henri Cohen, Christophe Doche, Gerhard Frey, Tanja Lange, Kim Nguyen, Frederik Vercauteren. Handbook of elliptic and hyperelliptic curve cryptography. 2005.

20. Бессалов А.В., Дихтенко А.А.,Третьяков Д.Б. Сравнительная оценка быстродействия канонических эллиптических кривых и кривых в форме Эдвардса над конечным полем. Сучасний захист інформації, No4, 2011.

ДОДАТОК А ТЕКСТИ ПРОГРАМ

Програми написані на мові програмування Python і використовують допоміжну бібліотеку з відкритим кодом sagemath для виконання базових арифметичних операцій у скінченному полі.

А.1 Допоміжні функції

```
# looking for non-residue in F_p
def find_ksi(p):
    i = 2
    while(true):
        if kronecker(i, p) == -1:
            return i
        i += 1
```

А.2 Вкладання та відновлення бітового вектора в ЕК в формі Монтгомері(M)

```
def encode_to_M(a, b, p, ksi, k):
    # initialize finite field object
    field = Integers(p, category=Fields())
    a, b, ksi, k = field(a), field(b), field(ksi), field(k)
    z_k = ksi * k**2
    if(z_k == 0 or z_k == 1):
        return None
    t1 = Mod(k, 2)
    u_k = - b * (1 + z_k) / (a * z_k)
    g_k = u_k**3 + a * u_k**2 + b * u_k
    t2 = 1 if kronecker(g_k, p) == 1 else 0
    if(t2 == 1):
        u_M= u_k
        v_M= g_k.nth_root(2)
    else:
        u_M= z_k * u_k
        v_M= ((z_k**3) * g_k).nth_root(2)
    if(v_M>field((p-1)/2)):
        v_M= p - v_M
    return ((u_M, v_M), t1, t2)
def decode_from_M(a, b, p, ksi, P, t1, t2):
    field = Integers(p, category=Fields())
    a, b, ksi, P = field(a), field(b), field(ksi), (field(P[0]), field(P[1]))
    if(t2 == 1):
        z_k = -b / (a*P[0] + b)
    else:
        z_k = -((a/b)*P[0] + 1)
    k = (z_k*((ksi)**(-1))).nth_root(2)
    if(Mod(k, 2) != t1):
        k = p - k
    return k
```

А.3 Вкладання і відновлення бітового вектора для ЕК в формі Монтгомері(M_1)

```
def encode_to_M1(mu, nu, p, ksi, k):
    field = Integers(p, category=Fields())
    mu, nu, ksi, k = field(mu), field(nu), field(ksi), field(k)
    z_k = ksi * k**2
    if(z_k == 0 or z_k == 1):
        return None
    t1 = Mod(k, 2)
    u_k = -(field(1) + z_k) / (nu * z_k)
    g_k = u_k**3 + nu * u_k**2 + u_k
    t2 = 1 if kronecker(g_k, p) == 1 else 0
    if(t2 == 0):
        u_M= u_k
        v_M= (mu**(-1)*g_k).nth_root(2)
    else:
        u_M= z_k * u_k
        v_M= (mu**(-1)*(z_k**3) * g_k).nth_root(2)
    if(v_M>field((p-1)/2)):
        v_M= p - v_M
    return ((u_M, v_M), t1, t2)
def decode_from_M1(mu, nu, p, ksi, P, t1, t2):
    field = Integers(p, category=Fields())
    mu, nu, ksi, P = field(mu), field(nu), field(ksi), (field(P[0]), field(P
        [1]))
    if(t2 == 0):
        z_k = -field(1)/(nu*P[0] + field(1))
    else:
        z_k = -(nu*P[0] + field(1))
    k = (z_k*((ksi)**(-1))).nth_root(2)
    if(Mod(k, 2) != t1):
        k = p - k
    return k
```

А.4 Вкладання і відновлення бітового вектора для ЕК в формі Едвардса

```
def encode_to_E(d, p, ksi, k):
    field = Integers(p, category=Fields())
    d, ksi, k = field(d), field(ksi), field(k)
    if(kronecker(field(1)-d, p) == 1):
        a = field(2)*(field(1)-d)/(field(1)+d)
        b = field(1)
        P_tmp, t1, t2 = encode_to_M(a, b, p, ksi, k)
        if(P_tmp[0] == field(-1) or P_tmp[1] == field(0)):
            return None
        x_E = (field(2)/(field(1)-d).nth_root(2)) * (P_tmp[0]/P_tmp[1])
        y_E = (P_tmp[0] - field(1))/(P_tmp[0] + field(1))
    else:
        mu = field(1)/(field(1)-d)
        nu = field(2)*(field(1)-d)/(field(1)+d)
        P_tmp, t1, t2 = encode_to_M1(mu, nu, p, ksi, k)
        if(P_tmp[0] == field(-1) or P_tmp[1] == field(0)):
```

```
            return None
        x_E = field(2) * (P_tmp[0]/P_tmp[1])
        y_E = (P_tmp[0] - field(1))/(P_tmp[0] + field(1))
    return ((x_E, y_E), t1, t2)
def decode_from_E(d, p, ksi, P, t1, t2):
    field = Integers(p, category=Fields())
    d, ksi, P = field(d), field(ksi), (field(P[0]), field(P[1]))
    if(kronecker(field(1)-d, p) == 1):
        u = (field(1)+P[1])/(field(1)-P[1])
        v = field(2)/(1-d).nth_root(2)/P[0]*(field(1)+P[1])/(field(1)-P[1])
        a = field(2)*(field(1)-d)/(field(1)+d)
        b = field(1)
        k = decode_from_M(a, b, p, ksi, (u, v), t1, t2)
    else:
        u = (field(1)+P[1])/(field(1)-P[1])
        v = field(2)/P[0]*(field(1)+P[1])/(field(1)-P[1])
        mu = field(1)/(field(1)-d)
        nu = field(2)*(field(1)-d)/(field(1)+d)
        k = decode_from_M1(mu, nu, p, ksi, (u,v), t1, t2)
    return k
```

A.5 Вкладання і відновлення бітового вектора для ЕК в неповній формі Веєрштрасса(a=0)

```
def encode_to_W_incomplete(c, s, p, ksi, k):
    field = Integers(p, category=Fields())
    c, s, ksi, k = field(c), field(s), field(ksi), field(k)
    a = field(3)*s
    b = field(3)*s**2
    P_tmp, t1, t2 = encode_to_M(a, b, p, ksi, k)
    if(P_tmp[0] == field(-1) or P_tmp[1] == field(0)):
        return None
    x_W = P_tmp[0]+s
    y_W = P_tmp[1]
    return ((x_W, y_W), t1, t2)
def decode_from_W_incomplete(c, s, p, ksi, P, t1, t2):
    field = Integers(p, category=Fields())
    c, s, ksi, P = field(c), field(s), field(ksi), (field(P[0]), field(P[1]))
    a = field(3)*s
    b = field(3)*s**2
    u = P[0]-s
    v = P[1]
    k = decode_from_M(a, b, p, ksi, (u, v), t1, t2)
    return k
```