Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)

Кафедра Безпеки інформаційних технологій
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Електронний цифровий підпис на основі λ-координат
(тема)

Виконав: Назарук Р.Р.
(прізвище, ініціали)
студент 2 курсу, групи БІКСм-18-1
Спеціальність 125 Кібербезпека
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма «Безпека інформаційних і комунікаційних систем»
(повна назва освітньої програми)

Керівник доцент Мельникова О.А.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Халімов Г.З.
(підпис) (прізвище, ініціали)

2019 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)
Кафедра Безпеки інформаційних технологій
(повна назва)
Рівень вищої освіти другий (магістерський)
Спеціальність 125 Кібербезпека
(код і повна назва)
Тип програми освітньо-професійна
(освітньо-професійна, або освітньо-наукова)
Освітня програма «Безпека інформаційних і комунікаційних систем»
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові *Назаруку Роману Руслановичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Електронний цифровий підпис на основі λ-координат*
затверджена наказом по університету від "04" листопада 2019 р. № 1649Ст
2. Термін подання студентом роботи до екзаменаційної комісії
3. Вихідні дані до роботи
1. *ДСТУ 4145-2002. Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка.*
2. Thomaz Oliveira, Julio Lopez, Diego F. Aranha, and Francisco Rodriquez-Henriquez – Lambda coordinates for binary elliptic curves.
3. Криптографічна бібліотека MIRACL.
4. Перелік питань, що потрібно опрацювати в роботі
1. Додавання та подвоєння точок ЕК у проективних координатах Лопеза-Дахаба
2. Додавання та подвоєння точок ЕК у λ-координатах
3. Електронний цифровий підпис згідно з ДСТУ 4145
4. Програмна реалізація ЕЦП згідно з ДСТУ 4145
5. Опис програмної реалізації та проведення експериментів
6. Результати досліджень
5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій *Презентаційний матеріал у вигляді слайдів*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	*Отримання завдання*	*9.09.18*	
2	*Пошук літератури*	*10.09.18-10.02.19*	
3	*Аналіз зібраних даних*	*11.02.19-19.04.19*	
4	*Програмна реалізація цифрового підпису згідно з ДСТУ 4145 у різних типах координат, постановка експерименту*	*20.04.19-25.07.19*	
5	*Аналіз отриманих результатів*	*26.07.19-26.08.19*	
6	*Оформлення пояснювальної записки*	*27.08.19-31.10.19*	

Дата видачі завдання _____ _____20__ р.

Студент _____
(підпис)

Керівник роботи (проекту) _____ ___доцент Мельникова О.А.____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Атестаційна робота містить 96 сторінок, 15 джерел, 4 таблиці, 1 рисунок, 3 додатки.

ЕЛЕКТРОНИЙ ЦИФРОВИЙ ПІДПИС, ЕЛІПТИЧНІ КРИВІ, ОПЕРАЦІЇ НАД ТОЧКАМИ ЕК, λ-КООРДИНАТИ

Об’єкт дослідження – алгоритми ЕЦП в групах точок ЕК над полем $GF(2^m)$.

Предмет дослідження – представлення точок ЕК в λ-координатах.

Основним завданням роботи є дослідження представлення точок в λ-координатах, та порівняння ефективності реалізації формування й перевірки цифрового підпису згідно з ДСТУ 4145 в λ-координатах та проективних координатах Лопеза-Дахаба над розширеним двійковим полем.

РЕФЕРАТ

Аттестационная работа содержит 96 страниц, 15 источников, 4 таблицы, 1 рисунок, 3 приложения.

ЭЛЕКТРОННАЯ ЦИФРОВАЯ ПОДПИСЬ, ЭЛЛИПТИЧКССКИЕ КРИВЫЕ, ОПЕРАЦИИ НАД ТОЧКАМИ ЭК, λ-КООРДИНАТИ

Объект исследования – алгоритмы ЕЦП в группах точек ЕК над полем $GF(2^m)$.

Предмет исследования – представление точек ЕК в λ-координатах.

Основным заданием работы является исследование представления точек в λ-координатах, и сравнение ефективности реализации формирования и проверки цифровой подписи согласно ДСТУ 4145 в λ-координатах и проективных координатах Лопеза-Дахаба над расширенным двоичным полем.

ABSTRACTION

The attestation work contains 96 pages, 15 bibliographic titles, 4 tables, 1 picture, 3 appendixes.

DIGITAL SIGNATURE ALGORITHM, ELIPTIC CURVES, OPERATIONS OVER POINTS EC, λ-COORDINATES

The object of the research is the digital signature algorithms in groups of EC points over field $GF(2^m)$.

The subject of the research is the representation of EC points in λ coordinates.

The main task of the work is to research the representation of points in λ-coordinates, and to compare the efficiency of realization of signing and verification of digital signature according to DSTU 4145 in λ-coordinates and projective coordinates of Lopez-Dahab over an extended binary field.

ЗМІСТ

УМОВНІ ПОЗНАЧЕННЯ, СИМВОЛИ, ОДИНИЦІ СКОРОЧЕННЯ І ТЕРМІНИ

ЕК – еліптична крива;

$GF(2^m)$ – розширене двійкове поле (поле Галуа);

ЕЦП – електронний цифровий підпис;

ДСТУ – державний стандарт України;

FIPS – federal information processing standards (стандарти обробки федеральної інформації).

ВСТУП

Зараз ми живемо у інформаційному світі. Інформація оточує нас всюди і є частиною нашого життя, в окремих випадках від неї може навіть залежати наше життя. Тож, на сьогоднішній день, захист інформації надзвичайно актуальне та життєво необхідне завдання.

Захистом інформації займається криптографія – наука про методи забезпечення конфіденційності (неможливості доступу до інформації сторонніх осіб), цілісності даних (неможливості непомітної зміни інформації) та автентифікації (перевірки справжності авторства чи прав доступу до інформації).

Одним з найефективніших підходів для забезпечення цілісності та неспростовності авторства є використання електронного цифрового підпису.

Сьогодні широке розповсюдження має криптографія на еліптичних кривих. Обчислювальна складність таких алгоритмів залежить від типу координат, у яких представляються точки на еліптичній кривій. В останній час більшість спеціалізованих бібліотек використовують проективні координати Лопеза-Дахаба та їх модифіковані версії.

Відносно недавно з’явилося нове представлення точок – λ-координати, базові операції в яких, теоретично, мають меншу обчислювальну складність ніж в інших відомих на даний момент координатах, у тому числі проективних координат Лопеза-Дахаба.

В даній роботі буде проведено дослідження доцільності використання λ-координат для реалізації цифрового підпису згідно з ДСТУ 4145 у контексті зменшення його обчислювальної складності.

1 ВЛАСТИВОСТІ НЕСИМЕТРИЧНИХ КРИПТОСИСТЕМ

1.1 Несиметрична криптографія над скінченими полями

Криптографія минулих століть мала одну величезну проблему – проблема передачі ключів. В ті часи існували тільки так звані «симетричні» шифри – шифри при якому дані шифруються і розшифровуються одним і тим же ключем.

Наприклад, Аліса зашифрувала деяке повідомлення і хоче відправити його Бобу. Щоб Боб його прочитав, йому потрібен ключ яким було зашифровано дане повідомлення. І тут виникає проблема, як передати ключ щоб його ніхто не зміг перехопити. Одним із виходів може бути передача ключів при особистій зустрічі, а вже потім відправляти повідомлення. Однак, уявімо, що наша інтернет пошта, перед тим як ми авторизуєтесь в ній, потребує особистої поїздки до фізичного місця розташування сервера з поштою. Це не дуже зручно.

Звичайно ключ можна передавати по іншому каналу зв'язку. Але криптографія розглядає всі незахищені канали зв'язку як небезпечні. Тобто передача ключа Бобу, наприклад, по телефону вважається небезпечною так, як ніщо не заважає третій особі прослуховувати, в тому числі, і телефон.

До 70-их років, ця проблема настільки стала звичною, що вважався аксіомою той факт, що під час написання повідомлень потрібно передавати і ключ яким повідомлення зашифровано [1]. Але в 1976 році Діффі та Хеллман запропонували свій «метод експоненціального обміну ключів». З цих років і почався розвиток асиметричних криптосистем.

У основі несиметричної криптографії лежать односторонні функції. Це такі функції $f(x)$, що за відомим x досить легко знайти значення $f(x)$, тоді як

визначити x знаючи $f(x)$ неможливо за розумний термін [2]. Різницю між двосторонніми на односторонніми функціями розглянемо на прикладі. Припустимо є функція подвоєння, тобто $double(4) = 8$, вона двостороння, тому що з результату 8 легко отримати вихідне значення 4. Прикладом односторонньої функції може бути змішування жовтої і синьої фарби. Змішати їх легко, а ось отримати назад вихідні компоненти – неможливо.

У якості односторонньої функції у криптографії було взято обчислення по модулю.

За основу алгоритму Діфі-Хеллмана була запропонована функція $Y^x \ (mod\ P)$. Зворотне перетворення для такої функції дуже складне, і можна сказати що, полягає в повному переборі вихідних значень x. Для використання у криптографії, число P має бути простим числом, а Y – первісним коренем за модулем P.

Алгоритм обміну ключами Діфі-Хелмана вимагає виконання операцій обидвома сторонами та має наступний вигляд:

Етап 1. Обидва учасники домовляються про значення Y та P для загальної односторонньої функції. Ця інформація не є секретною. Припустимо були обрані значення 7 і 11. Загальна функція буде виглядати наступним чином: $7^x \ (mod\ 11)$.

Етап 2. Аліса вибирає випадкове число, наприклад 3, це її секретний ключ, позначимо його як число A. Боб обирає випадкове число, наприклад 6, це його секретний ключ, позначимо його як число B.

Етап 3. Аліса підставляє число A в загальну функцію і обчислює результат $7^3 \ (mod\ 11) = 343 \ (mod\ 11) = 2$, позначає результат цього обчислення як число a, це її публічний ключ. Боб підставляє число B в загальну функцію і обчислює результат $7^6 \ (mod\ 11) = 117649 \ (mod\ 11) = 4$, позначає результат цього обчислення як число b, це його публічний ключ.

Етап 4. Аліса та Боб обмінюються своїми публічними ключами.

Етап 5. Аліса використовуючи отримане від Боба число b обчислює значення $b^A \ (mod\ 11) = 4^3 \ (mod\ 11) = 64 \ (mod\ 11) = 9$. Боб використовуючи отримане від Аліси число a обчислює значення $a^B \ (mod\ 11) = 2^6 \ (mod\ 11) = 64 \ (mod\ 11) = 9$.

У результаті обидва учасники отримали число 9 (їх спільний секретний ключ) не обмінюючись при цьому секретними ключами. А для обчислення спільного ключа, не знаючи секретних параметрів, необхідно вирішити рівняння виду $4^x \equiv 2^y \ (mod\ 11)$, що при великих розмірах параметрів криптосистеми неможливо за розумний час.

Іншим класичним прикладом несиметричної криптографії є алгоритм шифрування даних RSA. Як і будь яке несиметричне шифрування, RSA передбачає під собою наявність двох ключів – публічного і приватного (секретного). Дану пару ключів генерує приймаюча сторона до початку передачі даних, і розсилає усім відправникам публічний ключ. Алгоритм генерації ключів RSA має наступний вигляд [3]:

Етап 1. Обираються два простих числа p та q. Нехай це будуть $p = 3$, $q = 7$.

Етап 2. Обчислюється модуль $n = p * q$. Для нашого випадку $n = 3 * 7 = 21$.

Етап 3. Обчислюється функція Ейлера від модуля $\varphi(n) = (p - 1) * (q - 1)$. У нашому випадку $\varphi(n) = (3 - 1) * (7 - 1) = 2 * 6 = 12$.

Етап 4. Обирається таке просте число e, що $e \in (1;\ \varphi(n))$ та e взаємно просте з $\varphi(n)$. Візьмемо $e = 5$. Отримана пара чисел (e, n) і є публічним ключем.

Етап 5. Обчислюється $d = e^{-1} \ (mod\ \varphi(n))$. Для нашого прикладу підходить $d = 17$. Отримана пара чисел (d, n) є секретним ключем.

Зашифрування має наступний вигляд:

Етап 1. Обирається повідомлення $M \in (1; n)$. Нехай $M = 19$.

Етап 2. Обчислюється шифротекст $C = M^e \ (mod\ n)$. У нашому випадку $C = 19^5 \ (mod\ 21) = 2476099 \ (mod\ 21) = 10.$

Розшифрування має наступний вигляд:

Використовуючи шифротекст та приватний ключ обчислюється значення вихідного повідомлення $M' = C^d \ (mod\ n)$. Для нашого випадку $M' = 10^{17} \ (mod\ 21) = 100000000000000000 \ (mod\ 21) = 19 = M.$

Отже будь хто може, використовуючи публічний ключ, зашифрувати повідомлення та відправити його приймаючи стороні, а для його розшифрування необхідний секретний ключ, який приймаюча сторона не розголошує.

1.2 Криптографія на еліптичних кривих

У 1985 році незалежно Нілом Кобилице та Віктором Міллером було запропоновано використовувати в криптографії алгебраїчні властивості еліптичних кривих. З цього моменту почався бурхливий розвиток нового напряму несиметричної криптографії, для якого використовується термін "криптографія на еліптичних кривих". Роль основний криптографічних операції виконує операція скалярного множення точки на еліптичній кривій на дане ціле число, що визначається через операції додавання і подвоєння точок еліптичної кривої. Останні, в свою чергу, виконуються на основі операцій додавання, множення та інвертування в кінцевому полі, над якими розглядається крива. Особливий інтерес до криптографії на еліптичних кривих обумовлений тим, що для забезпечення тієї ж стійкості використовуються менші за розмірами ключі.

Ранні криптосистеми з відкритим ключем, такі як алгоритм RSA, крипостійкість завдяки тому, що складно розкласти складене число на прості множники. При використанні алгоритмів на еліптичних кривих вважається, що не існує субекспоненціальних алгоритмів для вирішення задачі дискретного

логарифмування в групах точок ЕК. При цьому порядок групи точок еліптичної кривої визначає складність завдання. Для досягнення такого ж рівня криптостійкості як і в RSA, потрібні групи менших порядків, що зменшує витрати на зберігання і передачу інформації. Наприклад, на конференції RSA 2005 Агентство національної безпеки оголосило про створення "Suite B", в якому використовуються виключно алгоритми еліптичної криптографії, причому для захисту інформації, що класифікується до "Top Secret", використовуються лише 384-бітові ключі [4].

Еліптичною кривої називається множина точок (x, y), що задовольняють рівняння:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6. \quad (1.1)$$

У криптографії розглядаються два види еліптичних кривих: над скінченим полем Z_p – кільцем відрахувань по модулю простого числа. І над полем GF(2^m) – бінарним скінченим полем.

У еліптичних кривих над полем GF(2^m) є одна важлива перевага, елементи поля GF(2^m) можуть бути легко представлені у вигляді n-бітових кодових слів, що дозволяє збільшити швидкість апаратної реалізації еліптичних алгоритмів [5].

Всі математичні операції на еліптичних кривих над скінченим полем виконуються за законами поля над яким побудована еліптична крива. Тобто, для обчислення, наприклад, суми двох точок кривої E над Z_p всі операції проводяться по модулю числа p.

У класичних несиметричних системах у якості односторонньої функції зазвичай використовується піднесення до степені за модулем. На заміну даній операції у криптографії на еліптичних кривих у якості односторонньої функції використовується множення точки на число.

Розглянемо, наприклад, варіант, вже відомого нам, алгоритму генерації ключа Діфі-Хелмана на еліптичних кривих.

Етап 1. Обидва учасники домовляються про значення параметрів еліптичної кривої, що буде використовуватись та обирають випадкову точку G на даній кривій. Ця інформація не є секретною.

Етап 2. Аліса вибирає випадкове число A, це її секретний ключ. Боб обирає випадкове число B, це його секретний ключ.

Етап 3. Аліса, використовуючи раніше згенероване число A, обчислює свій публічний ключ $G_a = A * G$. Боб, використовуючи раніше згенероване число B, обчислює свій публічний ключ $G_b = B * G$.

Етап 4. Аліса та Боб обмінюються своїми публічними ключами.

Етап 5. Аліса, використовуючи отриману від Боба точку G_b, обчислює спільний секрет $K = A * G_b$. Боб, використовуючи отриману від Аліси точку G_a, обчислює спільний секрет $K = B * G_a$.

У результаті, не обмінюючись секретними ключами, обидва учасники отримали однаковий спільний секрет K, оскільки $K = A * G_b = A * (B * G) = (A * B) * G = (B * A) * G = B * (A * G) = B * G_a = K$. А для знаходження спільного ключа, не знаючи секретних параметрів, необхідно вирішити задачу обчислення дискретного логарифму у групі точок ЕК, тобто знаючи G_x та G знайти таке X, що $G_x = X * G$. На даний момент невідомо про субекспоненційні алгоритми вирішення даної задачі, тож криптографія на еліптичних кривих вважається більш стійкою ніж класична асиметрична криптографія.

1.3 Електронний цифровий підпис згідно ДСТУ 4145-2002

ДСТУ 4145-2002 – стандарт, що установлює механізм цифрового підпису, заснований на властивостях груп точок еліптичних кривих над полями GF(2^m), та правила застосування цього механізму до повідомлень, що пересилаються каналами зв’язку та/або обробляються у комп’ютеризованих системах загального призначення. Застосування цього стандарту гарантує цілісність

підписаного повідомлення, автентичність його автора та неспростовність авторства [6].

Алгоритм обчислення цифрового передпідпису.

Вхідні дані алгоритму: загальні параметри цифрового підпису.

Результат виконання алгоритму – цифровий передпідпис F_e, що відповідає таємному випадковому параметру e, де e – ціле число, $0 < e < n$, $F_e \in GF(2^m)$.

Кроки алгоритму:

1. Обчислюють випадкове ціле число e згідно з розділом 6.3 [6].

2. Обчислюють точку еліптичної кривої $R = eP = (x_R, y_R)$.

3. Якщо координата $x_R = 0$, то переходять до кроку 1, інакше приймають $F_e = x_R$ і переходять до кроку 4.

4. Результат виконання алгоритму – цифровий передпідпис F_e та таємний випадковий параметр e.

Умови обчислення й зберігання таємного параметра e мають унеможливлювати несанкціонований доступ до нього, його частин, а також до проміжних даних, які використовувались у процесі обчислення цифрового передпідпису.

Припускається попереднє обчислення довільного числа цифрових передпідписів. Умови зберігання цифрового передпідпису мають унеможливлювати його модифікацію або підміну. Після використання цифрового передпідпису його негайно знищують разом з відповідним таємним параметром e.

Алгоритм обчислення цифрового підпису.

Вхідні дані алгоритму:

- загальні параметри цифрового підпису;
- особистий ключ цифрового підпису d;
- повідомлення T довжини $L_T > 0$;

- функція гешування H згідно з розділом 6.2 [6];
- довжина цифрового підпису L_D, що вибирається для групи користувачів, виходячи з умов конкретної реалізації алгоритму цифрового підписування, з урахуванням умов кроку 3 даного алгоритму.

Результат виконання алгоритму: повідомлення T і цифровий підпис D, що дають змогу утворити підписане повідомлення (iH, T, D).

Кроки алгоритму:

1) Перевіряють правильність загальних параметрів цифрового підпису згідно з розділів 8.1-8.3 [6]. Якщо загальні параметри цифрового підпису обчислено неправильно, то обчислення цифрового підпису припиняють. Цю перевірку не виконують у випадках, передбачених у розділах 8.1-8.3 [6].

2) Перевіряють правильність особистого ключа цифрового підпису згідно з розділом 10.2 [6]. Якщо особистий ключ неправильний, то обчислення цифрового підпису припиняють. Цю перевірку не виконують у випадках, передбачених у розділі 10.2 [6].

3) Перевіряють виконання умов: L_D – число, кратне 16, $L_D \geq 2L(n)$. Якщо хоча б одна з цих умов не виконана, то обчислення цифрового підпису припиняють.

4) Якщо використовується ідентифікатор геш-функції iH, то перевіряють, чи цей ідентифікатор діє у відповідній групі користувачів. Якщо ні, то обчислення цифрового підпису припиняють.

5) Якщо нормативні документи, що встановлюють обчислення функції гешування, накладають обмеження на довжину повідомлення L_T, то перевіряють виконання цих обмежень. Якщо ці обмеження не виконані, або повідомлення відсутнє, або $L_T \leq 0$, то обчислення цифрового підпису припиняють.

6) За повідомленням T обчислюють функцію гешування $H(T)$.

7) Результат обчислення функції гешування $H(T)$ перетворюють на елемент основного поля h згідно з розділом 5.9 [6]. Якщо $h = 0$, то приймають $h = 1$.

8) Якщо існує набір цифрових передпідписів, обчислених заздалегідь, то беруть будь-який з них разом з відповідним таємним параметром. Інакше обчислюють цифровий передпідпис. Нехай на цьому кроці алгоритму отримано передпідпис F_e та відповідний таємний параметр e.

9) Обчислюють елемент основного поля $y = hF_e$.

10) Елемент основного поля y перетворюють на ціле число r згідно з розділом 5.8 [6].

11) Якщо $r = 0$, то переходять до кроку 8, інакше переходять до кроку 12.

12) Обчислюють ціле число $s = (e + dr)\ mod\ n$.

13) Якщо $s = 0$, то переходять до кроку 8, інакше переходять до кроку 14.

14) Пару цілих чисел (r, s) перетворюють на цифровий підпис D довжини L_D згідно з розділом 5.10 [6].

15) Результат виконання алгоритму – підписане повідомлення (iH, T, D).

Алгоритм перевірки цифрового підпису.

Вхідні дані алгоритму:

- загальні параметри цифрового підпису;
- відкритий ключ цифрового підпису Q;
- підписане повідомлення (iH, T, D) довжини $L = L(iH) + L_T + L_D$;
- функція гешування H згідно з розділом 6.2 [6].

Результат виконання алгоритму: повідомлення “підпис дійсний” або повідомлення “підпис недійсний”.

Кроки алгоритму:

1) Якщо використовується ідентифікатор геш-функції iH, то перевіряють, чи діє цей ідентифікатор у відповідній групі користувачів. Якщо ні, то видають повідомлення “підпис недійсний” і припиняють перевірку цифрового підпису.

2) Виходячи з iH (або за промовчанням) визначають L_H.

3) Перевіряють виконання умов: L_D – число, кратне 16, $L_D \geq 2L(n)$. Якщо хоча б одна з цих умов не виконана, то видають повідомлення “підпис недійсний” і припиняють перевірку цифрового підпису.

4) Перевіряють правильність обчислення загальних параметрів цифрового підпису згідно з розділів 8.1-8.3 [6]. Якщо загальні параметри цифрового підпису обчислено неправильно, то видають повідомлення “підпис недійсний” і припиняють перевірку цифрового підпису. Цю перевірку не виконують у випадках, передбачених у розділах 8.1-8.3 [6].

5) Перевіряють правильність відкритого ключа цифрового підпису згідно з розділом 10.1 [6]. Якщо відкритий ключ обчислено неправильно, то видають повідомлення “підпис недійсний” і припиняють перевірку цифрового підпису. Цю перевірку не виконують у випадках, передбачених у розділі 10.1 [6].

6) Обчислюють $L_T = L - L_D - L(iH)$. У випадку відсутності тексту, або при $L_T \leq 0$ видають повідомлення “підпис недійсний” і припиняють перевірку цифрового підпису. Якщо нормативні документи, які встановлюють обчислення функції гешування, накладають обмеження на довжину повідомлення L_T, то перевіряють виконання цих умов. Якщо ці умови не виконані, то видають повідомлення “підпис недійсний” і припиняють перевірку цифрового підпису.

7) За повідомленням T обчислюють функцію гешування $H(T)$.

8) Геш-код $H(T)$ перетворюють на елемент основного поля h згідно з розділом 5.9 [6]. Якщо $h = 0$, то приймають $h = 1$.

9) Цифровий підпис D перетворюють на пару цілих чисел (r, s) згідно з розділом 5.11 [6].

10) Якщо умова $0 < r < n$ не виконана, то видають повідомлення “підпис недійсний” і припиняють перевірку цифрового підпису.

11) Якщо умова $0 < s < n$ не виконана, то видають повідомлення “підпис недійсний” і припиняють перевірку цифрового підпису.

12) Обчислюють точку еліптичної кривої $R = sP + rQ$, $R = (x_R,\ y_R)$.

13) Обчислюють елемент основного поля $y = hx_R$.

14) Елемент основного поля y перетворюють на ціле число $\tilde{r}$ згідно з розділом 5.8 [6].

15) Якщо $r = \tilde{r}$, то видають повідомлення “підпис дійсний”, інакше видають повідомлення “підпис недійсний”.

1.4 EC-DSA згідно ISO/IEC 15946-2

У міжнародному стандарті ISO/IEC 15946-2 представлено декілька алгоритмів електронного цифрового підпису, серед яких – EC-DSA. Даний алгоритм ЕЦП також представлений у стандарті США FIPS 186-4 та був гармонізований у державному стандарті ДСТУ ISO/IEC 15946-2 тож розглянемо саме його.

Схема цифрового підпису EC-DSA є аналогом схеми цифрового підпису DSA на еліптичних кривих. Схема є прикладом механізму вироблення цифрового підпису з доповненням [7].

Параметри домену та параметри користувачів.

Бітова довжина модуля n повинна бути більше ніж бітова довжина вихідного значення геш-функції $h()$.

Особистий й відкритий ключі об’єкта A, d_A і P_A відповідно, повинно виробляти у відповідності із процедурою 7.1, що визначена у стандарті ISO/IEC 15946-1.

Процес вироблення цифрового підпису.

Вхідними даними для процесу вироблення цифрового підпису є такі:

– параметри домену;

– особистий ключ d_A підписувача;

– повідомлення M.

Виходом процесу вироблення цифрового підпису є пара $(r, s) \in F(n) * F(n)$, яка становить цифровий підпис повідомлення M об'єкта A.

Для підписування повідомлення M об'єктом A виконуються такі кроки:

1) Обчислення геш-значення $e = h(M)$.

2) Вибір випадкового цілого числа k в інтервалі $\{1, \dots, n-1\}$.

3) Обчислення точки на еліптичній кривій $(x_1, y_1) = kG$.

4) Обчислення $r = \pi(kG)\ mod\ n$.

5) Обчислення k^{-1} в полі $F(n)$.

6) Обчислення $s = (d_A r + e)k^{-1}\ mod\ n$.

Якщо у процесі вироблення цифрового підпису формується або $s = 0$, або $r = 0$, тоді процес вироблення цифрового підпису необхідно повторити з новим випадковим значенням k. (Але необхідно зазначити, що ймовірність того, що $s = 0$ або $r = 0$ є надзвичайно малою, якщо k обрано відповідно до розділів 6.2.2. та 4.2.1 [7]).

Пара цілих чисел (r, s) становить цифровий підпис повідомлення M об'єкта A.

У зв'язку з тим, що обчислене значення r не залежить від повідомлення, що підписується, число r може обчислюватися попередньо, і надалі зберігатися та використовуватися під час вироблення цифрового підпису.

Процес перевірки цифрового підпису.

Процес перевірки цифрового підпису складається з 4 кроків: перевірки розміру ЕЦП, обчислення геш-значення повідомлення, обчислення на еліптичних кривих та перевірка цифрового підпису.

Вхідними даними для перевірки цифрового підпису є такі:

– параметри домену;

– відкритий ключ P_A об'єкта A;

- одержане повідомлення M';
- одержаний цифровий підпис повідомлення M', представлений двома цілими числами r' та s'.

Для перевірки цифрового підпису повідомлення M' об'єкта A, об'єктом B виконуються такі кроки:

1) Перевіряння того, що $0 < r' < n$ та $0 < s' < n$. Якщо хоч одна умова не виконуються, то цифровий підпис відхиляється.

2) Обчислення геш-значення $e' = h(M')$, використовуючи геш-функцію $h()$.

3) Обчислення $w = (s')^{-1}\ mod\ n$.

4) Обчислення $u_1 = e'w\ \ mod\ n$ та $u_2 = r'w\ \ mod\ n$.

5) Обчислення точки на ЕК $(x_1,\ y_1) = u_1G + u_2P_A$.

6) Обчислення $v = \pi\big((x_1,\ y_1)\big)\ mod\ n$.

Якщо $r' = v$, тоді цифровий підпис повинен прийматися перевірником. Якщо $r' \neq v$, тоді цифровий підпис повинен відхилятися перевірником.

2 ПРЕДСТАВЛЕННЯ ТОЧОК ЕК У РІЗНИХ ТИПАХ КООРДИНАТ

Для обчислення подвоєння або суми пари точок на еліптичній кривій у афінних координатах потрібно не тільки кілька операцій додавання і множення в скінченних полях, але й операція звернення, тобто для заданого $x \in GF(2^m)$ знаходження такого $y \in GF(2^m)$, що $xy = 1$, яка на один-два порядки повільніше, ніж множення [8]. Саме тому були створенні різні системи координат, відмінні від афінних в яких можуть бути представлені точки на еліптичній кривій, та які не вимагають використання звернення (пошуку інверсного елементу) при додаванні точок.

Усі види систем координат мають різні формули для додавання/подвоєння точок еліптичної кривої, а отже і різну складність виконання даних операцій. Розглянемо найрозповсюдженіші системи координат.

2.1 Афінні координати

В афінних координатах еліптичні криві задаються наступним виразом $y^2 + xy = x^3 + ax^2 + b$ (рівняння Веєрштраса), де a та b – коефіцієнти рівняння еліптичної кривої [9].

Нехай дана крива E що задовольняє даному рівнянню, та дані точки $P_1(x_1, y_1) \in E$ й $P_2(x_2, y_2) \in E$, тоді результуюча точка $P_3(x, y) = P_1 + P_2, \in E$ розраховується за формулами:

$$\lambda = \frac{y_1 + y_2}{x_1 + x_2};$$
$$x = \lambda^2 + \lambda + x_1 + x_2 + a; \quad (2.1)$$
$$y = (x_1 + x)\lambda + x + y_1.$$

Нехай дана крива E що задовольняє даному рівнянню, та точка $P_1(x_1, y_1) \in E$, тоді результуюча точка $P_2(x, y) = 2P_1, \in E$ розраховується за формулами:

$$\lambda = \frac{y_1}{x_1} + x_1;$$
$$x = \lambda^2 + \lambda + a; \quad (2.2)$$
$$y = (x_1 + x)\lambda + x + y_1.$$

Таким чином, для виконання додавання двох різних точок, необхідно виконати: одне зведення в квадрат, одне множення, одне ділення, 9 додавань. Для виконання операції подвоєння необхідно виповнити: одне зведення в квадрат, одне множення, одне ділення, 6 додавань. Найбільш затратною операцією, серед перерахованих вище, є ділення. Тож афінні координати являються найменш ефективними при реалізації криптосистем.

2.2 Стандартні проективні координати

Якщо ділення у полі GF(2^m) є затратною операцією, то від неї можна відмовитись відстежуючи чисельник та знаменник окремо. Таким чином, можна замінити ділення елемента на α множенням знаменника на α. Це досягається проективними координатами X, Y та Z, заданими [10]:

$$x = \frac{X}{Z};$$
$$y = \frac{Y}{Z}. \quad (2.3)$$

Проективні координати точки не унікальні, тому що

$$(X, Y, Z) = (\lambda X, \lambda Y, \lambda Z) \quad (2.4)$$

для кожного ненульового $\lambda \in GF\ (2^m)$.

Проективні координати точки на нескінченності $(\lambda, \lambda, 0)$, де $\lambda \neq 0$.

У проективних координатах рівняння кривої має вигляд

$$Y^2Z + XYZ = X^3 + aX^2Z + bZ^3 \quad (2.5)$$

Додавання точок ЕК у проективних координатах відбувається наступним чином. Нехай у нас є точки $P_1(X_1, Y_1, Z_1)$ та $P_2(X_2, Y_2, Z_2)$, тоді координати точки $P(X, Y, Z) = P_1 + P_2$ обчислюються за формулами [11]:

$$
\begin{gathered}
O1 = Y_1 * Z_2; \\
O2 = X_1 * Z_2; \\
H = Z_1 * Y_2; \\
I = Z_1 * X_2; \\
A = O1 + H; \\
B = O2 + I; \\
V = A + B; \\
C = B^2; \\
D = Z_1 * Z_2; \\
E = B * C; \\
J = A * V; \\
K = a * C; \\
M = J + K; \\
N = M * D; \\
F = N + E; \\
R1 = A * O2; \\
R2 = B * O1; \\
R = R1 + R2; \\
Q = C * R; \\
S = V * F; \\
X = B * F; \\
Y = Q + S; \\
Z = E * D.
\end{gathered}
\tag{2.6}
$$

Обчислювальна складність даної операції становить

$$I_{add} = 14m + 1mp + 1s + 7a, \tag{2.7}$$

де m – множення двох елементів поля GF(2^m);

mp – множення на параметр кривої у полі GF(2^m);

s – піднесення до квадрату елементу поля GF(2^m);

a – додавання двох елементів поля GF(2^m).

Подвоєння точки ЕК у проективних координатах відбувається наступним чином. Нехай дано точку $P_1(X_1, Y_1, Z_1)$, тоді координати точки $P(X, Y, Z) = 2P_1$ обчислюються за формулами [11]:

$$\begin{gathered} A = X_1^2; \\ F = Y_1 * Z_1; \\ B = A + F; \\ C = X_1 * Z_1; \\ V = B + C; \\ D = C^2; \\ G = B * V; \\ H = a * D; \\ E = G + H; \\ X = C * E; \\ I = V * E; \\ J = A^2; \\ K = J * C; \\ Y = I + K; \\ Z = C * D. \end{gathered} \tag{2.8}$$

Обчислювальна складність даної операції становить

$$I_{double} = 7m + 1mp + 3s + 4a, \tag{2.9}$$

де m – множення двох елементів поля GF(2m);

mp – множення на параметр кривої у полі GF(2m);

s – піднесення до квадрату елементу поля GF(2m);

a – додавання двох елементів поля GF(2m).

Додавання точок ЕК у змішаних координатах (коли одна точка в афінних координатах, а інша в проективних) відбувається наступним чином. Нехай у нас є точки $P_1(X_1, Y_1, Z_1)$ та $P_2(X_2, Y_2, 1)$, тоді координати точки $P(X, Y, Z) = P_1 + P_2$ обчислюються за формулами [11]:

$$
\begin{gathered}
R1 = Z_1 * Y_2; \\
R2 = Z_1 * X_2; \\
A = Y_1 + R1; \\
B = X_1 + R; \\
V = A + B; \\
C = B^2; \\
E = B * C; \\
D1 = A * V; \\
D2 = a * C; \\
D = D1 + D2; \\
G = D * Z_1; \\
F = G + E; \\
O1 = A * X_1; \\
O2 = B * Y_1; \\
O = O1 + O2; \\
H = C * O; \\
J = V * F; \\
X = B * F; \\
Y = H + J; \\
Z = E * Z_1.
\end{gathered}
\tag{2.10}
$$

Обчислювальна складність даної операції становить

$$I_{mix} = 11m + 1mp + 1s + 7a, \tag{2.11}$$

де m – множення двох елементів поля GF(2^m);

mp – множення на параметр кривої у полі GF(2^m);

s – піднесення до квадрату елементу поля GF(2^m);

a – додавання двох елементів поля GF(2^m).

2.3 Проективні координати Якобі

У проективних координатах Якобі точка задається трьома параметрами X, Y, Z, що співвідносяться з афінними наступним чином [10]:

$$x = \frac{X}{Z^2};$$

$$y = \frac{X}{Z^3}. \quad (2.12)$$

У проективних координатах Якобі рівняння кривої має вигляд

$$Y^2 + XYZ = X^3 + aX^2Z^2 + bZ^6 \quad (2.13)$$

Додавання точок ЕК у проективних координатах Якобі відбувається наступним чином. Нехай у нас є точки $P_1(X_1, Y_1, Z_1)$ та $P_2(X_2, Y_2, Z_2)$, тоді координати точки $P(X, Y, Z) = P_1 + P_2$ обчислюються за формулами [12]:

$$O1 = Z_1^2;$$

$$O2 = Z_2^2;$$

$$A = X_1 * O2;$$

$$B = X_2 * O1;$$

$$C1 = O2 * Z_2;$$

$$C = Y_1 * C1;$$

$$D1 = O1 * Z_1;$$

$$D = Y_2 * D1;$$

$$E = A + B;$$

$$F = C + D;$$

$$G = E * Z_1;$$

$$H1 = F * X_2;$$

$$H2 = G * Y_2;$$

$$H = H1 + H2; \tag{2.14}$$

$$Z = G * Z_2;$$

$$I = F + Z;$$

$$K1 = Z^2;$$

$$K = a * K1;$$

$$L = F * I;$$

$$M1 = E^2;$$

$$M = E * M1;$$

$$N = L + M;$$

$$R = I * X;$$

$$T1 = G^2;$$

$$T = T1 * H;$$

$$X = K + N;$$

$$Y = R + T.$$

Обчислювальна складність даної операції становить

$$I_{add} = 14m + 1mp + 5s + 7a, \tag{2.15}$$

де m – множення двох елементів поля GF(2m);

mp – множення на параметр кривої у полі GF(2m);

s – піднесення до квадрату елементу поля GF(2m);

a – додавання двох елементів поля GF(2^m).

Подвоєння точки ЕК у проективних координатах Якобі відбувається наступним чином. Нехай дано точку $P_1(X_1, Y_1, Z_1)$, тоді координати точки $P(X, Y, Z) = 2P_1$ обчислюються за формулами [13]:

$$A = X_1^2;$$

$$B = A^2;$$

$$C = Z_1^2;$$

$$D = C^2;$$

$$E1 = D^2;$$

$$E = b * E1;$$
$$X = B + E;$$
$$Z = X_1 * C;$$
$$F = B * Z;$$
$$G = Y_1 * Z_1;$$
$$H = A + G;$$
$$I = H + Z;$$
$$J = I * X;$$
$$Y = F + J.$$

(2.16)

Обчислювальна складність даної операції становить

$$I_{double} = 4m + 1mp + 5s + 4a,$$

(2.17)

де m – множення двох елементів поля GF(2m);

mp – множення на параметр кривої у полі GF(2m);

s – піднесення до квадрату елементу поля GF(2m);

a – додавання двох елементів поля GF(2^m).

Додавання точок ЕК у змішаних координатах (коли одна точка в афінних координатах, а інша в проективних Якобі) відбувається наступним чином. Нехай у нас є точки $P_1(X_1, Y_1, Z_1)$ та $P_2(X_2, Y_2, 1)$, тоді координати точки $P(X, Y, Z) = P_1 + P_2$ обчислюються за формулами [12]:

$$O1 = Z_1^2;$$
$$B = X_2 * O1;$$
$$O2 = O1 * Z_1;$$
$$D = Y_2 * O2;$$
$$E = X_1 + B;$$
$$F = Y_1 + D;$$
$$Z = E * Z_1;$$
$$H1 = F * X_2;$$
$$H2 = Z * Y_2;$$

$$H = H1 + H2;$$
$$I = F + Z;$$
$$G = Z^2;$$
$$A = a * G;$$
$$J = F * I;$$
$$C1 = E^2;$$
$$C = E * C1;$$
$$K = A + B;$$
$$X = K + C;$$
$$L = I * X;$$
$$M = G * H;$$
$$Y = L + M. \quad (2.18)$$

Обчислювальна складність даної операції становить

$$I_{mix} = 10m + 1mp + 3s + 7a, \quad (2.19)$$

де m – виконання множення двох елементів поля GF(2m);

mp – множення на параметр кривої у полі GF(2m);

s – виконання піднесення до квадрату елементу поля GF(2m);

a – додавання двох елементів поля GF(2^m).

2.4 Проективні координати Лопеза-Дахаба

У проективних координатах Лопеза-Дахаба точка задається трьома параметрами X, Y, Z, що співвідносяться з афінними наступним чином [10]:

$$x = \frac{X}{Z};$$
$$y = \frac{X}{Z^2}. \quad (2.20)$$

У проективних координатах Лопеза-Дахаба рівняння кривої має вигляд

$$Y^2 + XYZ = X^3Z + aX^2Z^2 + bZ^4 \quad (2.21)$$

Додавання точок ЕК у проективних координатах Лопеза-Дахаба відбувається наступним чином. Нехай у нас є точки $P_1(X_1, Y_1, Z_1)$ та $P_2(X_2, Y_2, Z_2)$, тоді координати точки $P(X, Y, Z) = P_1 + P_2$ обчислюються за формулами [14]:

$$
\begin{gathered}
A = X_1 * Z_2; \\
B = X_2 * Z_1; \\
C = A^2; \\
D = B^2; \\
E = A + B; \\
F = C + D; \\
G1 = Z_2^2; \\
G = Y_1 * G1; \\
H1 = Z_1^2; \\
H = Y_2 * H1; \\
I = G + H; \\
J = I * E; \\
K = Z_1 * Z_2; \\
L1 = H + D; \\
L = A * L1; \\
M1 = C + G; \\
M = B * M1; \\
N1 = A * J; \\
N2 = F * G; \\
N = N1 + N2; \\
R = N * F; \\
Z = F * K; \\
X = L + M; \\
S1 = J + Z;
\end{gathered}
\qquad (2.22)
$$

$$S = S1 * X;$$

$$Y = R + S.$$

Обчислювальна складність даної операції становить

$$I_{add} = 13m + 4s + 9a, \quad (2.23)$$

де m – множення двох елементів поля GF(2m);

s – піднесення до квадрату елементу поля GF(2m);

a – додавання двох елементів поля GF(2^m).

Подвоєння точки ЕК у проективних координатах Лопеза-Дахаба відбувається наступним чином. Нехай дано точку $P_1(X_1, Y_1, Z_1)$, тоді координати точки $P(X, Y, Z) = 2P_1$ обчислюються за формулами [14]:

$$A1 = X_1^2;$$

$$A2 = Z_1^2;$$

$$B1 = A1^2;$$

$$B2 = A2^2;$$

$$C = b * B2;$$

$$D = Y_1^2;$$

$$Z = A1 * A2;$$

$$X = B1 + C; \quad (2.24)$$

$$E1 = a * Z;$$

$$E2 = D + C;$$

$$E = E1 + E2;$$

$$F = C * Z;$$

$$G = X * E;$$

$$Y = F + G.$$

Обчислювальна складність даної операції становить

$$I_{double} = 3m + 2mp + 5s + 4a, \quad (2.25)$$

де m – множення двох елементів поля GF(2m);

mp – множення на параметр кривої у полі GF(2m);

s – піднесення до квадрату елементу поля GF(2m);

a – додавання двох елементів поля GF(2m).

Додавання точок ЕК у змішаних координатах (коли одна точка в афінних координатах, а інша в проективних Лопеза-Дахаба) відбувається наступним чином. Нехай у нас є точки $P_1(X_1, Y_1, Z_1)$ та $P_2(X_2, Y_2, 1)$, тоді координати точки $P(X, Y, Z) = P_1 + P_2$ обчислюються за формулами [14]:

$$
\begin{gathered}
E1 = Z_1^2; \\
E = Y_2 * E1; \\
F = X_2 * Z_1; \\
A = Y_1 + E; \\
B = X_1 + F; \\
C = B * Z_1; \\
Z = C^2; \\
D = X_2 * Z; \\
G1 = A^2; \\
G2 = B^2; \\
H1 = A + G2; \\
H2 = a * C; \\
H = H1 + H2; \\
I = C * H; \\
X = G1 + I; \\
J = D + X; \\
K1 = Z^2; \\
K2 = Y_2 + X_2; \\
K = K1 * K2; \\
L1 = A * C; \\
L = L1 + Z; \\
M = J * L;
\end{gathered}
\qquad (2.26)
$$

$$Y = M + K.$$

Обчислювальна складність даної операції становить

$$I_{mix} = 8m + 1mp + 5s + 9a, \quad (2.27)$$

де m – виконання множення двох елементів поля GF(2m);

mp – множення на параметр кривої у полі GF(2m);

s – виконання піднесення до квадрату елементу поля GF(2m);

a – додавання двох елементів поля GF(2^m).

2.5 λ-координати

Окрім вище описаних систем координат відносно нещодавно був представлений новий спосіб відображення точок на еліптичній кривій – λ-координати. Розрізняють λ-афінні та λ- проективні координати.

У λ-афінних координатах точка задається парою $(x; \lambda)$, $\lambda = x + \frac{y}{x}$, де x та y – афінні координати точки (у класичному представленні) [15]. Як видно з даної формули, в λ-координатах неможливо задати точку, x координата якої становить 0. Однак, виходячи із формули подвоєння точки (у класичних афінних координатах), після подвоєння даної точки ми отримаємо точку на нескінченності. Отже порядок такої точки становить 2, що недопустимо для використання в криптографічних цілях. Відповідно, для точок, порядком яких являється просте число (а саме такі точки і використовуються у криптографії в якості базових точок), відображення у λ-афінних координатах має місце.

Для оберненого переходу (від λ-афінних координат то класичних афінних), y координата розраховується за формулою

$$y = x(\lambda - x). \quad (2.28)$$

У λ-проективних координатах точка задається трьома параметрами X, L, Z, що співвідносяться з λ-афінними наступним чином [15]:

$$x = \frac{X}{Z};$$

$$\lambda = \frac{L}{Z}. \quad (2.29)$$

Виходячи із вище описаних формул для перехід від класичних афінних координат до λ-проективних має наступний вигляд

$$(x; x + \frac{y}{x}; 1). \quad (2.30)$$

Зворотній перехід від λ-проективних координат до класичних афінних має вигляд

$$(\frac{X}{Z}; \frac{X(L-X)}{Z^2}). \quad (2.31)$$

Рівняння еліптичної кривої Веєрштраса у λ-проективних координатах має наступний вигляд [15]:

$$(L^2 + L * Z + a * Z^2) * X^2 = X^4 + b * Z^4. \quad (2.32)$$

Додавання точок ЕК у λ-проективних координатах відбувається наступним чином. Нехай у нас є точки $P_1(X_1, L_1, Z_1)$ та $P_2(X_2, L_2, Z_2)$, тоді координати точки $P(X, L, Z) = P_1 + P_2$ обчислюються за формулами [15]:

$$C = L_1 * Z_2;$$

$$D = L_2 * Z_1;$$

$$A = C + D;$$

$$E = X_1 * Z_2;$$

$$F = X_2 * Z_1;$$

$$B1 = E + F;$$

$$B = B1^2;$$

$$G = A * F;$$

$$V = A * B;$$

$$W = V * Z_2; \quad (2.33)$$

$$K = A * E;$$

$$X = K * G;$$

$$H1 = G + B;$$

$$H = H1^2;$$

$$I = L_1 + Z_1;$$

$$J = W * I;$$

$$L = H + J;$$

$$Z = W * Z_1.$$

Обчислювальна складність даної операції становить

$$I_{add} = 11m + 2s + 5a, \quad (2.34)$$

де m – множення двох елементів поля GF(2m);

s – піднесення до квадрату елементу поля GF(2m);

a – додавання двох елементів поля GF(2^m).

Подвоєння точки ЕК у λ-проективних координатах відбувається наступним чином. Нехай дано точку $P_1(X_1, L_1, Z_1)$, тоді координати точки $P(X, L, Z) = 2P_1$ обчислюються за формулами [15]:

$$A = L_1^2;$$

$$B = Z_1^2;$$

$$C = L_1 * Z_1;$$

$$T1 = A + C;$$

$$T2 = a * B;$$

$$T = T1 + T2;$$

$$X = T^2;$$

$$Z = T * B; \quad (2.35)$$

$$D = X_1^2;$$

$$E1 = D * B;$$

$$E2 = T * C;$$

$$E = E1 + E2;$$

$$K = X + Z;$$

$$L = E + K.$$

Обчислювальна складність даної операції становить

$$I_{double} = 4m + 1mp + 4s + 5a. \quad (2.36)$$

Однак, у ситуації коли множення на параметри кривої швидше ніж множення повнорозмірних елементів поля, ми можемо заздалегідь розрахувати константні значення $a^2 + b$ і $a + 1$, та з їх урахуванням отримаємо наступні формули для обчислення подвоєння точки [15]:

$$\begin{gathered} m = a^2 + b; \\ n = a + 1; \\ A = L_1^2; \\ B = Z_1^2; \\ C = L_1 * Z_1; \\ D = L_1 + X_1; \\ E = D^2; \\ T1 = A + C; \\ T2 = a * B; \\ T = T1 + T2; \\ X = T^2; \\ Z = T * B; \\ F1 = E + T; \\ F2 = F1 + B; \\ F = E * F2; \\ G1 = B^2; \\ G = m * G1; \\ H1 = F + G; \\ H = H1 + X; \\ J = n * Z; \\ L = H + J. \end{gathered} \quad (2.37)$$

Обчислювальна складність даної операції становить

$$I_{double} = 3m + 3mp + 5s + 8a, \quad (2.38)$$

де m – множення двох елементів поля GF(2m);

mp – множення на параметр кривої у полі GF(2m);

s – піднесення до квадрату елементу поля GF(2m);

a – додавання двох елементів поля GF(2^m).

У другому алгоритмі подвоєння точки виконується на одне множення повнорозмірних елементів менше, однак виконуються два додаткових множення на $a^2 + b$ та $a + 1$. Отже коли виконання даних двох множень швидше ніж одне повне (при розмірі параметрів a та b значно менших за розмір поля) множення слід використовувати другий варіант алгоритму подвоєння точки, у інших ж випадках – перший.

Додавання точок ЕК у змішаних координатах (коли одна точка в λ-афінних координатах, а інша в λ-проективних) відбувається наступним чином. Нехай у нас є точки $P_1(X_1, L_1, Z_1)$ та $P_2(X_2, L_2, 1)$, тоді координати точки $P(X, L, Z) = P_1 + P_2$ обчислюються за формулами [15]:

$$\begin{gathered} D = L_2 * Z_1; \\ A = L_1 + D; \\ F = X_2 * Z_1; \\ B1 = X_1 + F; \\ B = B1^2; \\ G = A * F; \\ V = A * B; \\ C = A * X_1; \\ X = C * G; \\ E1 = G + B; \\ E = E1^2; \\ H1 = L_1 + Z_1; \\ H = V * H1; \end{gathered} \quad (2.39)$$

$$L = E + H;$$

$$Z = V * Z_1.$$

Обчислювальна складність даної операції становить

$$I_{mix} = 8m + 2s + 5a, \quad (2.40)$$

де m – множення двох елементів поля GF(2m);

s – піднесення до квадрату елементу поля GF(2m);

a – додавання двох елементів поля GF(2^m).

2.6 Порівняльний аналіз

Усі з вище перерахованих систем координат мають різні властивості. Лише в афінних координатах для виконання базових операцій над точками еліптичної кривої необхідно знаходити мультиплікативно зворотній елемент (таке x^{-1}, що $x^{-1} * x \equiv 1 \, mod \, f(x)$ у полі GF(2^m)). Дана операція є надзвичайно розрахунково складною, навіть у порівнянні з множенням двох елементів поля, через що дана система координат не використовується при криптографічних розрахунках. Для найбільш затратних обчислень точки переводять у один із типів проективних координат, а після їх завершення назад до афінних.

Тож порівнювати ми будемо ефективність використання різних видів проективних координат, які мають різну складність операцій додавання, подвоєння та змішаного додавання точок.

Теоретична оцінка обчислювальної складності виконання базових операцій над точками ЕК у різних типах проективних координат наведена у таблиці 2.1. Для відображення обчислювальної складності у таблиці використовуються наступні позначення: m – операція множення, mp – операція множення на параметр кривої, s – операція піднесення до квадрату, a – операція додавання, всі операції виконуються у полі GF(2^m).

Таблиця 2.1 – Теоретична оцінка складності виконання базових операцій над точками ЕК

Тип координат	Додавання	Подвоєння	Змішане додавання
Стандартні проективні	$14m + 1mp + 1s + 7a$	$7m + 1mp + 3s + 4a$	$11m + 1mp + 1s + 7a$
Проективні Якобі	$14m + 1mp + 5s + 7a$	$4m + 1mp + 5s + 4a$	$10m + 1mp + 3s + 7a$
Проективні Лопеза-Дахаба	$13m + 4s + 9a$	$3m + 2mp + 5s + 4a$	$8m + 1mp + 5s + 9a$
λ-проективні	$11m + 2s + 5a$	$^{*}4m + 1mp + 4s + 5a$ $^{*}3m + 3mp + 5s + 8a$	$8m + 2s + 5a$

*представлено формули (2.36), (2.38) для 2 алгоритмів подвоєння точки в λ-координатах, наведених у розділі 2.5.

Формули складності із наведеної вище таблиці громіздкі та незручні для аналізу, тож ми їх дещо спростимо. Оскільки Обчислювальна складність операцій додавання та піднесення до квадрату значно менша ніж операції множення ними можна знехтувати. Отже у наведених в таблиці 2.1 формулах ми відкинули дані операції. Отримана теоретична оцінка обчислювальної складності виконання базових операцій над точками ЕК за спрощеними формулами наведена у таблиці 2.2, умовні позначення ті самі що й для попередньої таблиці.

Таблиця 2.2 – Спрощена теоретична оцінка складності виконання базових операцій над точками ЕК

Тип координат	Додавання	Подвоєння	Змішане додавання
Стандартні проективні	$14m + 1mp$	$7m + 1mp$	$11m + 1mp$
Проективні Якобі	$14m + 1mp$	$4m + 1mp$	$10m + 1mp$
Проективні Лопеза-Дахаба	$13m$	$3m + 2mp$	$8m + 1mp$
λ-координати	$11m$	$^{*}4m + 1mp$ $^{*}3m + 3mp$	$8m$

*представлено формули (2.36), (2.38) для 2 алгоритмів подвоєння точки в λ-координатах, наведених у розділі 2.5.

Виходячи із даних представлених у таблиці 2.2 проективні координати Якобі швидше справляються із подвоєнням та змішаним додавання точок ніж звичайні проективні, та мають ідентичну розрахункову складність при додаванні точок.

Проективні координата Лопеза-Дахаба та λ координати показують себе краще ніж стандартні проективні та координати Якобі при виконанні всіх базових операцій над точками ЕК.

λ-координати мають меншу розрахункову складність ніж проективні Лопеза-Дахаба при додаванні. Відносна ефективність подвоєння та змішаного додавання точок у даних системах координат залежить від розмірів параметрів кривої a та b. При змішаному додаванні у проективних координатах Лопеза-Дахаба виконується на одну операцію множення на параметр a більше. Отже при значному розмірі даного параметра λ-координати будуть помітно

ефективніші, при значенні a 1 або 0, дані системи координат матимуть незначну різницю у ефективності.

Обчислювальна складність подвоєння точки у координатах Лопеза-Дахаба залежить від розмірів обох параметрів a та b, у λ-координатах в залежності від обраного алгоритму або тільки від a, або від a та b. Отже при значному розмірі параметру b слід використовувати алгоритм подвоєння, що залежить тільки від параметру a, при $b = 1$, алгоритм, що залежить від обох параметрів. При правильному використанні алгоритму подвоєння точки у λ-координатах в залежності від значення параметру b координати Лопеза-Дахаба та λ-координати мають однакову спрощену теоретичну складність.

3 МЕТОДИ СКАЛЯРНОГО ДОБУТКУ ТОЧКИ ЕК НА ЧИСЛО

3.1 Бінарний метод від старших розрядів

Бінарний метод від старших розрядів (або бінарний лівосторонній) є одним із найпростіших методів множення точки ЕК на число, який полягає в наступному.

Нехай $R = k * P$, де R та P – точки на еліптичний кривій, а k – число.

Представимо k у бінарному вигляді:

$$k = (k_{t-1}, k_{t-2}, \dots, k_1, k_0)_2, \ k_{t-1} \neq 0. \tag{3.1}$$

Виходячи з властивостей бінарної системи числення отримуємо, що

$$k = (\dots(k_{t-1}2 + k_{t-2})2 + \dots + k_3)2 + k_2)2 + k_1)2 + k_0. \tag{3.2}$$

Отже скалярний добуток матиме вигляд:

$$R = k * P = \sum_{i=0}^{t-1} 2^i k_i P \tag{3.3}$$

Запишемо даний вираз у вигляді алгоритму.

1. $R = I$;
2. $for(i = t - 1; i \geq 0; \ i--)$
    2.1 $R = 2R$;
    2.2 $if \ (k_i == 1)$
        2.2.1 $R = R + P$;
3. $return \ R$.

Фактично даний алгоритм і є бінарним лівостороннім методом скалярного добутку точки на число. Однак прийнявши до уваги, що $k_{t-1} \neq 0$ з (2.1) стає зрозуміло що на першій ітерації циклу в будь-якому разі виконається крок 2.2.1. Подвоєння I дасть I, а $I + P = P$. Виходячи з цього даний алгоритм можна спростити прибравши першу ітерацію циклу.

Отримаємо наступний алгоритм:

1. $R = P$;

2. $for(i = t - 2; i \geq 0;\ i - -)$

    2.1 $R = 2R$;

    2.2 $if\ (k_i == 1)$

        2.2.1 $R = R + P$;

3. $return\ R$.

Складність даного алгоритму становить:

$$I_{bin}^{L} = (t - 1)(\frac{1}{2}I_A + I_D), \tag{3.4}$$

де t – бітова довжина числа k;

I_A – операція додавання точок ЕК (оскільки додавання виконується з однією і тією ж початковою точко P, можна використовувати додавання у змішаних координатах);

I_D – операція подвоєння точки ЕК.

3.2 Метод Монтгомері

Montgomery ladder – підхід до обчислення скалярного добутку точки ЕК на число, кожна ітерація якого працює за фіксований проміжок часу, та не залежить від кількості та послідовності одиничних чи нульових бітів у представленні числа. Це може бути корисним, коли ймовірний витік через побічні канали та зловмисник може виміряти час роботи алгоритму чи споживання електроенергії [5].

Нехай $R = kP$, тоді алгоритм Монтгомері для обчислення R має наступний вигляд:

1. $R_0 = I$;

2. $R_1 = P$;

3. $for(i = t - 1; i \geq 0;\ i - -)$

    3.1 $if\ (k_i == 0)$

3.1.1 $R_1 = R_0 + R_1$;

3.1.2 $R_0 = 2R_0$;

3.2 $else$

3.2.1 $R_0 = R_0 + R_1$;

3.2.2 $R_1 = 2R_1$;

4. $return\ R_0$.

Враховуючи, що на першій ітерації циклу $k_{t\text{-}1} = 1$ то в будь-якому випадку виконається крок 3.2. Виходячи з цього, даний алгоритм можна спростити, прибравши першу ітерацію циклу.

Отримаємо наступний алгоритм:

1. $R_0 = P$;

2. $R_1 = 2P$;

3. $for(i = t - 2; i \geq 0;\ i - -)$

3.1 $if\ (k_i == 0)$

3.1.1 $R_1 = R_0 + R_1$;

3.1.2 $R_0 = 2R_0$;

3.2 $else$

3.2.1 $R_0 = R_0 + R_1$;

3.2.2 $R_1 = 2R_1$;

4. $return\ R_0$.

Складність даного алгоритму становить:

$$I_{mont} = (t - 1)(I_A + I_D) + I_D, \tag{2.5}$$

де t – бітова довжина числа k;

I_A – операція додавання точок ЕК;

I_D – операція подвоєння точки ЕК.

При використанні класичних формул для додавання та подвоєння точок складність даного алгоритму є досить значною тож даний алгоритм модифікують для використання в тих чи інших типах координат.

Наприклад, для стандартних проективних координат алгоритм Монтгомері має вигляд:

1. $X_1 = x$;
2. $Z_1 = 1$;
3. $X_2 = x^4 + b$;
4. $Z_2 = x^2$;
5. $for(i = t - 2; i \geq 0;\ i--)$
    5.1 $if\ (k_i == 0)$
        5.1.1 $T = Z_2$;
        5.1.2 $Z_2 = (X_1 Z_2 + X_2 Z_1)^2$;
        5.1.3 $X_2 = xZ_2 + X_1 X_2 T Z_1$;
        5.1.4 $T = X_1$;
        5.1.5 $X_1 = X_1^4 + bZ_1^4$;
        5.1.6 $Z_1 = T^2 Z_1^2$;
    5.2 $else$
        5.2.1 $T = Z_1$;
        5.2.2 $Z_1 = (X_1 Z_2 + X_2 Z_1)^2$;
        5.2.3 $X_1 = xZ_1 + X_1 X_2 T Z_2$;
        5.2.4 $T = X_2$;
        5.2.5 $X_2 = X_2^4 + bZ_2^4$;
        5.2.6 $Z_2 = T^2 Z_2^2$
6. $x_3 = \frac{X_1}{Z_1}$;
7. $y_3 = (x + \frac{X_1}{Z_1})[(X_1 + xZ_1)(X_2 + xZ_2) + (x^2 + y)(Z_1 Z_2)](xZ_1 Z_2)^{-1} + y$;
8. $return\ (x_3,\ y_3)$.

3.3 Блоковий метод

Традиційний блоковий метод схожий на бінарний, але множник розглядається не у двійковій системі числення, а у системі числення з основою 2^w

Нехай $R = k * P$, де R та P – точки на еліптичний кривій, а k – число.

У бінарному представлені k має вигляд:

$$k = (k_{t\text{-}1}, k_{t\text{-}2}, \dots, k_1, k_0)_2,\ k_{t\text{-}1} \neq 0. \tag{3.5}$$

У системі числення з основою 2^w k має вигляд:

$$k = (k_{d-1}, k_{d-2}, \dots, k_1, k_0)_{2^w},\ d = \frac{t}{w},\ k_{d-1} \neq 0. \tag{3.6}$$

Отже скалярний добуток матиме вигляд:

$$R = k * P = \sum_{i=0}^{d-1} 2^{iw} k_i P \tag{3.7}$$

Даний алгоритм дуже схожий на бінарний, однак замість ітерації по бітам числа, виконується ітерація по блокам довжини w.

Через блоковий метод вимагає попереднього обчислення таблиці значень $i * P$, для $i \in [0;\ 2^w)$.

Алгоритм попередніх обчислень:

1 $T[0] = I;\ T[1] = P;$

2 $for\ (i = 2; i <\ 2^w; ++i)$

2.1 $if\ (i\%2 == 1)$

2.1.1 $T[i] = T[i-1] + P;$

$else$

2.1.2 $T[i] = 2 * T[i/2];$

3 $return\ T;$

Складність даного алгоритму становить:

$$I^0_{block} = (2^{w-1} - 1)(I_A + I_D), \tag{3.8}$$

де w – розмір блоку;

I_A – операція додавання точок ЕК (оскільки додавання виконується з однією і тією ж початковою точко P, можна використовувати додавання у змішаних координатах);

I_D – операція подвоєння точки ЕК.

Безпосередньо алгоритм множення:

1 $R = T[k_{d-1}]$;

2 $for\ (i = d - 2; i \geq 0;\ --i)$

2.1 $R = 2^w R$;

2.2 $R = R + T[k_i]$;

3 $return\ R$;

Його складність становить:

$$I_{block}^1 = \left(\frac{t}{w} - 1\right)(I_A + wI_D), \tag{3.9}$$

де t – бітова довжина числа;

w – розмір блоку;

I_A – операція додавання точок ЕК (оскільки додавання виконується з однією і тією ж початковою точко P, можна використовувати додавання у змішаних координатах);

I_D – операція подвоєння точки ЕК.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Постановка завдання

У даній роботі було поставлене завдання провести експериментальну оцінку обчислювальної складності алгоритмів формування та перевірки цифрового підпису, реалізованих згідно з ДСТУ 4145.

Оскільки проективні координати Лопеза-Дахаба та λ-координати мають найкращу розрахункову складність зазначені вище алгоритми були реалізовані у даних типах координат для порівняння їх обчислювальної складності.

При реалізації формування цифрового підпису не виконується перевірка загальносистемних параметрів, а також формування геш-значення повідомлення та перетворення його на елемент поля. Виконуються кроки 8-13 алгоритму формування цифрового підпису, описаного у розділі 1.3.

Для заміру часу виконання перевірки ЕЦП також не виконуються перевірки коректності вхідних параметрів та формування геш-значення й перетворення його на елемент поля. Виконуються кроки 12-15 алгоритму перевірки цифрового підпису, описаного у розділі 1.3.

Перевірка здійснювалась на 9 еліптичних кривих рекомендованих додатком Г ДСТУ 4145: U-163, U-167, U-173, U-179, U-191, U-233, U-307, U-367, U-431, а також для більшої деталізації експериментів, на 3 еліптичних кривих, рекомендованих додатком D FIPS-186-4.

4.2 Загальні відомості

Програма під назвою «Lamda» реалізує базові операції над точками ЕК, операцію еліптичного скалярного множення точки на число, а також

формування й перевірку цифрового підпису в λ-координатах та проективних координатах Лопеза-Дахаба. Виконує перевірку правильності цифрового підпису, а також час роботи алгоритмів (в тактах).

Для виконання операцій над елементами поля GF(2^m) використовується бібліотека MIRACL.

Програма написана на мові програмування С у середовищі розробки програмного забезпечення Microsoft Visual Studio 2017, бібліотека MIRACL була додана до проекту у вигляді вихідного коду. Для запуску програми (exe-файлу) необхідно мати операційну систему Windows XP або новіше з підтримкою архітектури x64. Для компіляції програми достатньо будь-якого компілятору, що підтримує стандарт мови програмування С.

4.3 Функціональне призначення

Програма розроблена для отримання експериментальної оцінки обчислювальної складності формування та перевірки цифрового підпису згідно ДСТУ 4145 при використанні λ-координат та проективних координат Лопеза-Дахаба. Оцінка проводиться на еліптичних кривих, що рекомендовані до використання додатками ДСТУ 4145 та FIPS 186-4.

4.4 Опис логічної структури

Програма складається з файлу base.c (містить реалізацію базових операцій над точками та реалізацію скалярного множення, див. додаток А), файлу main.c (головний код програми, містить реалізацію формування та перевірки цифрового підпису, а також замір часу виконання даних алгоритмів, див. додаток А) та файлів вихідного коду бібліотеки MIRACL. Програма побудована наступним чином. У головній функції (точці входу) викликається

функція test_on_curve, якій передаються параметри кривої на якій необхідно провести тести. Вона виконує ініціалізацію бібліотеки MIRACL і еліптичної кривої та викликає функції, що проводять тестування формування й перевірки цифрового підпису у λ-координатах та проективних координатах Лопеза-Дахаба та заміряє час їх роботи у тактах.

У програмі містяться наступні структури та функціїї:

1) Структура point_l_projective – описує точку задану у λ-проективних координатах.

2) Структура point_p – описує точку задану у проективних координатах Лопеза-Дахаба.

3) Функція inline timetype getTSC() – використовується для заміру часу виконання операцій у тактах процесора.

4) Функція point_l_projective* mypoint_init() – виконує ініціалізацію точки, представленої у λ-проективних координатах та повертає покажчик на неї.

5) Функція point_p* ld_point_init() – виконує ініціалізацію точки, представленої у проективних координатах Лопеза-Дахаба, та повертає покажчик на неї.

6) Функція void mypoint_kill(point_l_projective* p) – виконує видалення (звільнення зайнятої пам’яті) точки p.

7) Функція void ld_point_kill(point_p* p) – виконує видалення (звільнення зайнятої пам’яті) точки p.

8) Функція void mypoint_set(point_l_projective* p, big x, big y) – задає точці p афінні координати x, y.

9) Функція void ld_point_set(point_p* p, big x, big y) – задає точці p афінні координати x, y.

10) Функція void mypoint_get(point_l_projective* p, big x, big y) – обчислює і повертає афінні координати (x, y) точки p.

11) Функція void ld_point_get(point_p* p, big x, big y) – обчислює і повертає афінні координати (x, y) точки p.

12) Функція void mypoint_normalize(point_l_projective* p) – виконує нормалізацію точки p (приводить Z-координату до 1).

13) Функція void ld_point_normalize(point_p* p) – виконує нормалізацію точки p (приводить Z-координату до 1).

14) Функція void mypoint_copy(point_l_projective* source, point_l_projective* dest) – виконує копіювання точки source в точку dest.

15) Функція void ld_point_copy(point_p* source, point_p* dest) – виконує копіювання точки source в точку dest.

16) Функція void ld_point_double(point_p* p, point_p* p2) – виконує подвоєння точки p, записаної в проективних координатах Лопеза-Дахаба і заносить результат у точку p2.

17) Функція void point_double_1mp(point_l_projective* p, point_l_projective* p2) – виконує подвоєння точки p, записаної в λ-проективних координатах (алгоритмом зі спрощеною складністю $6m + 1mp$) та заносить результат у точку p2.

18) Функція void ld_point_add(point_p* p1, point_p* p2, point_p* p3) – виконує додавання точок p1 і p2, записаних в проективних координатах Лопеза-Дахаба і заносить результат у точку p3.

19) Функція void point_add(point_p* p1, point_p* p2, point_p* p3) – виконує додавання точок p1 і p2, записаних в λ-проективних координатах і заносить результат у точку p3.

20) Функція void ld_point_mixed_add(point_p* p1, point_p* p2, point_p* p3) – виконує додавання точок p1 і p2, де p1 записана у проективних координатах Лопеза-Дахаба, а p2 в афінних координатах, результат заносить у точку p3.

21) Функція void point_mixed_add(point_l_projective* p1, point_l_projective* p2, point_l_projective* p3) – виконує додавання точок p1 і p2, де p1 записана у λ-проективних координатах, а p2 в λ-афінних координатах, результат заносить у точку p3.

22) Функція void block_scalar_ld(big k, point_p* p, point_p* result) – виконує еліптичне скалярне множення точки p на число k з використанням проективних координат Лопеза-Дахаба, результат заносить до точки result.

23) Функція void block_scalar_lambda(big k, point_l_projective* p, point_l_projective* result) – виконує еліптичне скалярне множення точки p на число k з використанням λ-проективних координат, результат заносить до точки result.

24) Функція void init_data_for_scalar(int m, int k3, int k2, int k1, const char* a_s, const char* b_s) – виконує ініціалізацію бібліотеки MIRACL, ініціалізацію еліптичної кривої, що задається параметрами поля m, k3, k2, k1, та параметрами кривої a_s, b_s, і допоміжних змінних, необхідних для виконання скалярного множення та базових операцій над точками ЕК.

25) Функція void free_data_for_scalar() – звільняє ресурси, що були виділені функцією init_data_for_scalar.

26) Функція int get_bit_length(big x) – повертає бітову довжину числа x.

27) Функція void generate_random_digit(big n, big a) – виконує генерацію псевдовипадкового числа n, бітова довжина якого ≤ a.

28) Функція void truncate(big x, int new_bit_len) – виконує усічення числа x до бітової довжини new_bit_len.

29) Функція void ld_point_negate(point_p* p) – виконує відображення точки p, представленої у проективних координатах Лопеза-Дахаба відносно осі Ox (p = - p).

30) Функція void mypoint_negate(point_l_projective* p) – виконує відображення точки p, представленої у λ-проективних координатах відносно осі Ox (p = - p).

31) Функція void init_internal_variables() – виконує ініціалізацію допоміжних змінних, необхідних для виконання формування та перевірки цифрового підпису.

32) Функція void free_internal_variables() – звільняє ресурси, що були виділені функцією init_internal_variables.

33) Функція void generate_private_public_keys(big n, point_p* P_ld, point_l_projective* P_lambda, big d, point_p* Q_ld, point_l_projective* Q_lambda) – виконує генерацію таємного ключа d та відкритого ключа Q у проективних координатах Лопеза-Дахаба та λ-проективних координатах.

34) Функція void generate_presignature_ld(big n, point_p* P, big res_e, big res_Fe) – виконує формування цифрового передпідпису з використанням проективних координат Лопеза-Дахаба.

35) Функція void generate_presignature_lambda(big n, point_l_projective* P, big res_e, big res_Fe) – виконує формування цифрового передпідпису з використанням λ-проективних координат.

36) Функція void generate_signature_ld(int m, big n, point_p* P, big d, big h, big* res_r, big* res_s) – виконує формування цифрового підпису з використанням проективних координат Лопеза-Дахаба.

37) Функція void generate_signature_lambda(int m, big n, point_l_projective* P, big d, big h, big* res_r, big* res_s) – виконує формування цифрового підпису з використанням λ-проективних координат.

38) Функція bool check_signature_ld(big r, big s, int m, big n, point_p* P, point_p* Q, big h) – виконує перевірку цифрового підпису з використанням проективних координат Лопеза-Дахаба.

39) Функція bool check_signature_lambda(big r, big s, int m, big n, point_l_projective* P, point_l_projective* Q, big h) – виконує перевірку цифрового підпису з використанням λ-проективних координат.

40) Функція void test_and_print_signature(int m, big n, point_p* P_ld, point_l_projective* P_lambda) – виконує формування та перевірку цифрових підписів з використанням проективних координат Лопеза-Дахаба та λ-проективних координат, заміряє та виводить на екран час виконання даних алгоритмів.

41) Функція void test_on_curve(const char* name, int m, int k3, int k2, int k1, const char* a_s, const char* b_s, const char* gx_s, const char* gy_s, const char* n_s) – виконує ініціалізацію даних за допомогою функцій init_data_for_scalar та init_internal_variables. Задає базову точку з координатами gx_s, gy_s і виконує тестування за допомогою функції test_and_print_signature.

42) Функція int main() – виводить на екран кількість повторів що буде використовуватись для взяття мінімального та усереднення часу виконання базових операцій над точками. Проводить тестування формування та перевірки цифрового підпису на еліптичних кривих U-163, U-167, U-173, U-179, U-191, U-233, B-283, U-307, U-367, B-409, U-431, B-571 шляхом викликання функції test_on_curve.

4.5 Вимоги до технічних засобів

Дана програма не накладає обмеження на використовувані технічні засоби. Для реалізації поставленого завдання і тестування був використаний комп'ютер наступної конфігурації: CPU Intel Core i7-8750H 2.2 GHZ, Memory: 16GB, OS Windows 10.

4.6 Виклик і завантаження

Запуск програми здійснюється відкриттям виконуваного файлу «Lamda.exe».

4.7 Вхідні дані

Програма не потребує вхідних даних. Дані про еліптичні криві, кількість повторів для тестування та інші параметри вшиті у код.

4.8 Вихідні дані

Вихідними даними є текстовий вивід роботи програми, що включає дані що використовувалися для ініціалізації еліптичної кривої та базової точки, результати тестування по кожній ЕК, що включають в себе оцінку правильності цифрового підпису, а також час виконання (в тактах) алгоритмів формування та перевірки цифрового підпису у наведених вище типах координат.

Повний результат роботи програми наведено у додатку Б.

4.9 Експериментальні оцінки обчислювальних характеристик

Згідно з результатами роботи програми була отримана експериментальна оцінка обчислювальної складності формування й перевірки цифрового підпису при використанні проективних координат Лопеза-Дахаба та λ-координат. Отримані дані у вигляді: кількість тактів необхідних для виконання операції, занесено до таблиці 4.1. Еліптичні криві що починаються з «U» – рекомендовані до використання додатком Г стандарту ДСТУ 4145. Також для

більшої деталізації експериментів було взято ЕК, що рекомендовані додатком D стандарту FIPS-186-4, вони починаються з «B».

Таблиця 4.1 – Експериментальні оцінки обчислювальної складності алгоритмів формування та перевірки ЕЦП з використанням різних типів координат

№	Варіант еліптичної кривої	Формування підпису		Перевірка підпису	
		λ	Лопез-Дахаб	λ	Лопез-Дахаб
1	U-163	1132292	1239732	2206252	2431737
2	U-167	1052332	1159338	2039972	2245786
3	U-173	1143616	1257650	2227076	2453828
4	U-179	1229823	1350554	2393636	2639552
5	U-191	1181309	1289266	2313069	2526933
6	U-233	2012492	2180532	3970463	4318532
7	B-283	3033849	3264194	6019470	6496606
8	U-307	3339337	3609151	6579866	7118107
9	U-367	4597206	4917164	9163301	9790633
10	B-409	6390594	6765020	12667567	13413237
11	U-431	7233704	7707669	14347022	15293576
12	B-571	12763705	13552958	25345179	26942844

Як видно із таблиці 4.1, λ-координати виявилися дещо ефективнішими за проективні координати Лопеза-Дахаба. Для більшої наглядності дані було переведено з абсолютних величин у відносні. Для кожного алгоритму (та для кожної ЕК) за 100% взято час виконання з використанням проективних координат Лопеза-Дахаба. Дані занесено до таблиці 4.2.

Таблиця 4.2 –Відносна оцінка обчислювальної складності алгоритмів формування та перевірки ЕЦП з використанням різних типів координат

№	Варіант еліптичної кривої(3)	Формування підпису		Перевірка підпису	
		λ	Лопез-Дахаб	λ	Лопез-Дахаб
1	U-163	91%	100%	91%	100%
2	U-167	91%	100%	91%	100%
3	U-173	91%	100%	91%	100%
4	U-179	91%	100%	91%	100%
5	U-191	92%	100%	92%	100%
6	U-233	92%	100%	92%	100%
7	B-283	93%	100%	93%	100%
8	U-307	93%	100%	92%	100%
9	U-367	93%	100%	94%	100%
10	B-409	94%	100%	94%	100%
11	U-431	94%	100%	94%	100%
12	B-571	94%	100%	94%	100%
В середньому		92,5%	100%	92,5%	100%

Згідно з даними, представленими у таблиці 4.2, що є результатами проведеної експериментальної оцінки, використання λ-координат дозволить пришвидшити формування та перевірку цифрового підпису, реалізованого згідно ДСТУ4145 на 7,5% у порівнянні з проективними координатами Лопеза Дахаба.

Для наочного представлення, нрафічна інтерпретація експериментальних оцінок обчислювальної складності з таблиць 4.1 та 4.2 наведена на рисунку 4.1.

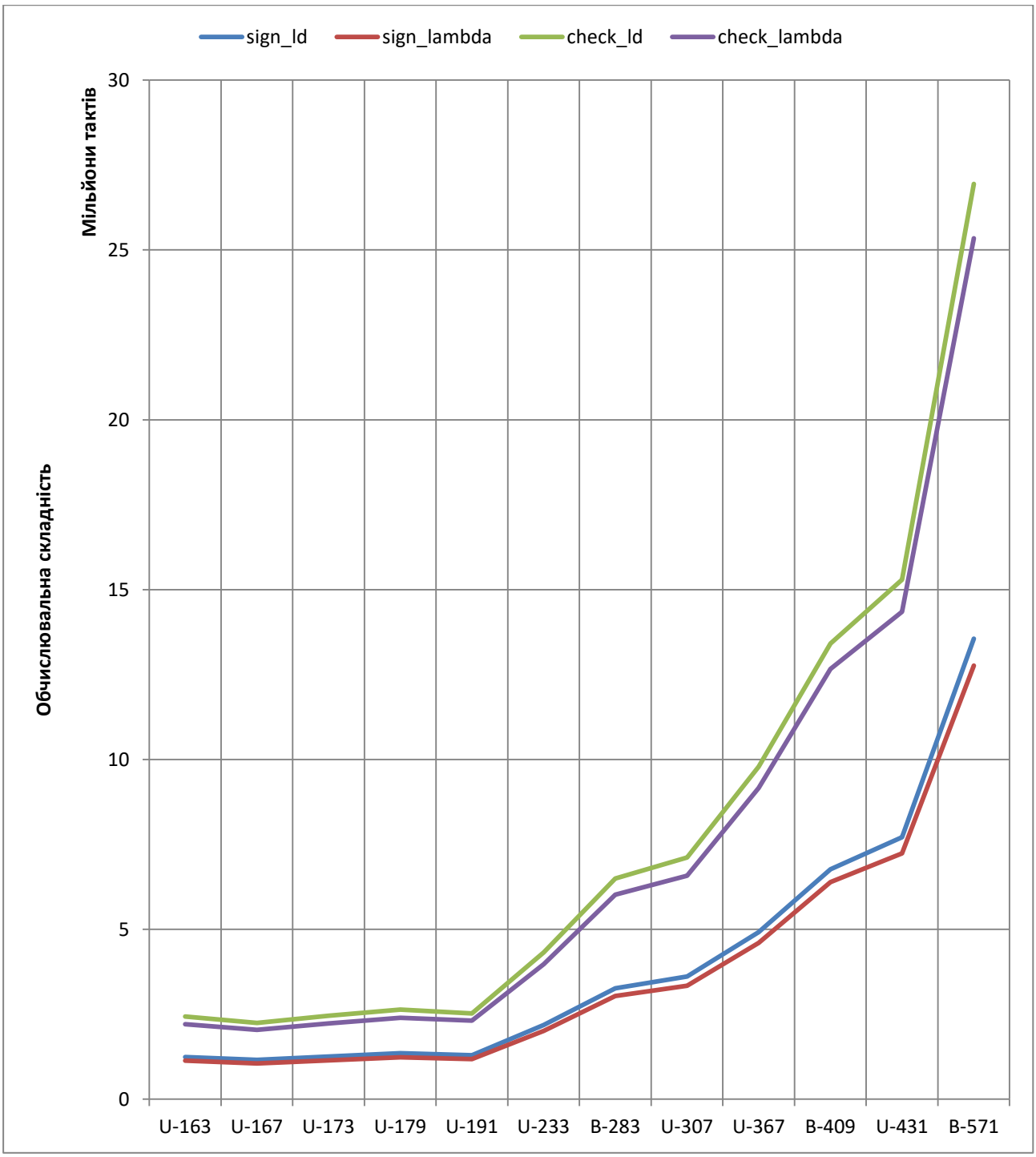

Рисунок 4.1 – Графічна інтерпретація експериментальних оцінок

Для більш ретельного дослідження доцільності використання λ координат у криптографії на еліптичних кривих необхідні подальші експерименти з реалізацією методів еліптичного скалярного множення різного типу. Наприклад, розробка модифікації методу Монтгомері для використання з

проективними λ-координатами або використання методів зі значним обсягом попередніх розрахунків для виконання операцій при фіксованій базовій точці (для формування передпідписів); розробка варіантів метода Шаміра для реалізації одночасного двократного скалярного множення при перевірці ЕЦП.

ВИСНОВКИ

У роботі було проведено оцінку обчислювальної складності електронного цифрового підпису на еліптичних кривих (згідно з ДСТУ 4145) при використанні проективних координат Лопеза-Дахаба та λ-координат. Оцінка проводилась на еліптичних кривих, що рекомендовані стандартами ДСТУ 4145 та FIPS-186-4.

Під час досліджень було виявлено, що зі збільшенням степені основного поля ефективність λ-координат відносно проективних координат Лопеза-Дахаба зменшується. Це відбувається через те, що використання λ-координат дає значне зменшення кількості виконання таких операцій як додавання та піднесення до квадрату елементів базового поля $GF(2^m)$ при виконанні операцій додавання та подвоєння точок ЕК, але незначне зменшення кількості операцій множення елементів поля $GF(2^m)$. Оскільки операції додавання та піднесення до квадрату мають лінійну складність, а операція множення – квадратичну, то зі збільшенням розмірів операндів зменшується відносна частка ресурсів, необхідних для додавання та піднесення до квадрату.

У середньому використання λ-координат дозволяє зменшити обчислювальну складність формування та перевірки ЕЦП на 7,5%.

ПЕРЕЛІК ПОСИЛАНЬ

1. Как работает ассиметричное шифрование | Безопасность [Електронний ресурс]: Режим доступу: https://intsystem.org/security/asymmetric-encryption-how-it-work/ (07.04.2018).

2. Криптосистема с открытым ключем [Електронний ресурс]: Режим доступу: https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%BC_%D0%BA%D0%BB%D1%8E%D1%87%D0%BE%D0%BC (07.04.2018).

3. RSA шифрование [Електронний ресурс]: Режим доступу: http://www.michurin.net/computer-science/rsa.html (07.04.2018).

4. Болотов А.А. Элементарное введение в эллиптическую криптографию. / А.А. Болтов, С.Б. Гашков, А.Б. Фролов. – Москва: КомКнига, 2006. – 328 с.

5. Эллиптическая криптография: теория [Електронний ресурс]: Режим доступу: https://habr.com/post/188958/ (07.04.2018).

6. Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка: ДСТУ 4145-2002. – [Чинний від 2003-07-01]. – Київ: Держстандарт України, 2003. – 92 с. – (Національний стандарт України).

7. Information technology -- Security techniques -- Cryptographic techniques based on elliptic curves -- Part 2: Digital signatures: ISO/IEC 15946-2. – [Чинний від 2002-12-01]. – International Organization for Standardization, 2002. – 36 с.

8. Hankrson D., Lopez Hernandez J., Menezes A. Software implementation of ellip-tic curve cryptography over binary fields // Advances in Cryptology Crypto'99. – P. 37 – 46.

9. Эллиптическая кривая [Електронний ресурс]: Режим доступу: https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%BB%D0%B8%D0%BF%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BA%D1%80%D0%B8%D0%B2%D0%B0%D1%8F (07.04.2018).

10. EFD / Ordinary genus-1 binary / short Weierstrass curves [Електронний ресурс]: Режим доступу: https://www.hyperelliptic.org/EFD/g12o/auto-shortw.html (07.04.2018).

11. EFD / Ordinary genus-1 binary / Projective coordinates for short Weierstrass curves [Електронний ресурс]: Режим доступу: https://www.hyperelliptic.org/EFD/g12o/auto-shortw-projective.html (07.04.2018).

12. EFD / Ordinary genus-1 binary / Jacobian coordinates for short Weierstrass curves [Електронний ресурс]: Режим доступу: https://www.hyperelliptic.org/EFD/g12o/auto-shortw-jacobian.html (07.04.2018).

13. IEEE 1363. Standard Specifications for Public-Key Cryptography – 2000 – 236c.

14. EFD / Ordinary genus-1 binary / Lopez-Dahab coordinates for short Weierstrass curves [Електронний ресурс]: Режим доступу: https://www.hyperelliptic.org/EFD/g12o/auto-shortw-lopezdahab.html (07.04.2018).

15. Thomaz Oliveira, Julio Lopez, Diego F. Aranha, and Francisco Rodriguez-Henriquez. Lambda coordinates for binary elliptic curves. CHES «Cryptographic Hardware and Embedded Systems». – 2013. – №15 – p.311-330.