НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК 681.301

«До захисту допущено»
в.о. завідувача кафедри
_____ В.П. Легеза
«___»_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 121 "Інженерія програмного забезпечення"

на тему: МОДИФІКОВАНИЙ МЕТОД БАГАТОКРАТНОГО СКАЛЯРНОГО МНОЖЕННЯ ТОЧОК ЕЛІПТИЧНОЇ КРИВОЇ У СКІНЧЕННИХ ПОЛЯХ

Виконав: студент 6 курсу, групи КП-61м
Дичка Андрій Іванович _____

Науковий керівник ст. викладач, к.т.н., Онай М.В. _____

Рецензент доцент кафедри обчислювальної техніки
ФІОТ, к.т.н., доцент Марковський О.П. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2018

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

в.о. завідувача кафедри

_____ В.П. Легеза

«___»_____2016 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ
Дичці Андрію Івановичу

1.Тема дисертації: МОДИФІКАЦІЯ МЕТОДУ БАГАТОКРАТНОГО СКАЛЯРНОГО МНОЖЕННЯ ТОЧОК ЕЛІПТИЧНОЇ КРИВОЇ У СКІНЧЕННИХ ПОЛЯХ, науковий керівник дисертації: Онай Микола Володимирович, ст. викладач, к.т.н., затверджені наказом по університету від «22» березня 2018 р. №986-С

2. Термін подання студентом дисертації «16» травня 2018 р.

3. Об'єкт дослідження: процеси обміну ключами, шифрування/дешифрування та створення/перевірка електронно-цифрового підпису у еліптичних криптосистемах.

4. Предмет дослідження: методи багатократного скалярного множення точок еліптичної кривої у полі $GF(p)$ та $GF(2^m)$.

5. Перелік завдань, які потрібно вирішити:
- дослідити та розробити алгоритми виконання операцій нижнього рівня у скінченних полях;
- дослідити та розробити алгоритми виконання операцій нижнього рівня на еліптичній кривій: додавання та подвоєння точок ЕК, множення точки ЕК на число;
- розробити програмне забезпечення для виконання алгоритму електронно-цифрового підпису на еліптичних кривих;
- розробити програмне забезпечення для реалізації методів багатократного множення точок еліптичної кривої на число;
- запропонувати метод високошвидкісного множення точок еліптичної кривої на число;
- розробити програмне забезпечення для реалізації методів багатократного множення точок еліптичної кривої на число, а також аналізу їх швидкодії.

6. Орієнтовний перелік ілюстративного матеріалу:
- геометрична ілюстрація додавання та подвоєння точок ЕК;
- схема асиметричного шифрування

- блок-схема алгоритму електронно-цифрового підпису з використанням ЕК;
- блок-схема розробленого методу багатократного множення точок еліптичної кривої на число;
- блок-схема алгоритму тривіального множення точки ЕК на число;
- порівняння швидкості роботи методів багатократного множення точок еліптичної кривої;

7. Орієнтовний перелік публікацій:
   - Тези доповіді "Модифікація методу багатократного скалярного множення точок еліптичної кривої над полем $GF(p)$ та $GF(2^m)$

8. Дата видачі завдання «15» жовтня 2016 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	14.12.2016	
2.	Визначення структури магістерської дисертації; вивчення літератури;	01.03.2017	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	17.05.2017	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації;	18.10.2017	
5.	Проведення наукового дослідження; робота над тезами доповіді;	13.12.2017	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації; підготовка матеріалів доповіді ПМК-2018.	07.03.2018	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу;	16.05.2018	
8.	Оформлення текстової і графічної частини магістерської дисертації	20.05.2018	

Студент _____ Дичка А.І.

Науковий керівник _____ Онай М.В.

РЕФЕРАТ

Актуальність теми: На сьогоднішній день галузь шифрування та захисту інформації відіграє важливу роль в царині інформатики. Це зумовлено масштабним використанням автоматизованих методів обробки та передачі даних та широким розповсюдженням методів та засобів несанкціонованого доступу до інформації, що пересилається незахищеними каналами зв'язку.

Проблему передачі деякої конфіденційної інформації адресату можна вирішити багатьма способами, проте найбільш часто використовуваними в наш час є так звані асиметричні криптосистеми. Низка провідних досліджень показує, що серед асиметричних криптосистем, або криптосистем з відкритим ключем найбільш стабільною є криптосистема з використанням еліптичних кривих. Дані криптосистеми для забезпечення достатньої криптографічної стійкості потребують довжини ключа в кілька разів меншу за ту, що необхідна для криптосистем на основі математчних операцій в скінченних полях.

Одним з яскравих представників сімейства криптографічних алгоритмів з використанням еліптичних кривих є алгоритм електронно-цифрового підпису ECDSA.

При застосуванні криптографічних алгоритмів з використанням еліптичних кривих дуже важливим чинником є час їхньої роботи. Експериментальні дослідження показують, що найбільш ресурсо- та часовитратними є операції, що виконуються безпосередньо з точками еліптичної кривої, зокрема для алгоритму ECDSA найбільш ресурсовитратною є операція багатократного скалярного множення точок еліптичної кривої на число. Таким чином, актуальними є дослідження способів та методів оптимізації даного обчислення.

Об'єкт дослідження: процеси обміну ключами, шифрування, дешифрування та створення/перевірка електронно-цифрового підпису в еліптичних криптосистемах.

Предмет дослідження: методи багатократного скалярного множення точок еліптичної кривої у полі $GF(p)$ та $GF(2^m)$.

Мета дослідження: розробити метод високошвидкісного множення точок еліптичної кривої на число.

Методи дослідження: методи теорії чисел, аналітичної геометрії, методи абстрактної алгебри, дискретної математики та криптографії.

Наукова новизна полягає в наступному**:**

1. Запропоновано модифікацію методу багатократного скалярного множення точки еліптичної кривої на число, яка полягає у виконанні спеціального передобчислення, і на відміну від існуючих методів, дозволяє звести виконання операції множення точки еліптичної кривої на число до додавання, що забезпечує вищу швидкодію (до 25%) порівняно з існуючими для еліптичних кривих з параметрами над $GF(2^m)$.
2. Розроблено архітектуру програмного забезпечення, особливістю якої є інкапсуляція алгоритму багатократного скалярного множення точки еліптичної кривої на число, що дозволяє на основі шаблону «Стратегія» замінювати конкретну реалізацію в екземплярі класу електронно-цифрового підпису без застосування наслідування.

Практична цінність результатів полягає в наступному: запропоновані алгоритми дозволяють швидше виконувати операцію багатократного скалярного множення точок еліптичної кривої на число, що дозволяє скоротити час виконання алгоритму електронно-цифрового підпису.

Розроблене математичне та програмне забезпечення для виконання операцій з точками еліптичної кривої, що дозволяє спростити подальші дослідження у галузі еліптичної криптографії.

Структура пояснювальної записки:

У розділі 1 розглядається загальний контекст досліджень, а саме – математичні основи та принципи еліптичної криптографії, аналіз методів

шифрування, що використовують еліптичну криптографію, огляд алгоритмів генерування та перевірки електронно-цифрового підпису з використанням еліптичної криптографії.

У розділі 2 розглядаються принципи реалізації операцій над точками еліптичної кривої, аналізуються алгоритми додавання та множення точок еліптичної кривої на число, розглядаються алгебраїчні структури, що використовуються для визначення параметрів точки еліптичної кривої та їх особливості, а також узагальнено описуються принципи реалізації дій над точками еліптичної кривої в електронно-обчислювальних пристроях.

У розділі 3 розроблено модифікацію методу багатократного скалярного множення точок еліптичної кривої на число; виконується аналіз особливостей його застосування та порівняння властивостей цього методу з відомими.

У розділі 4 розроблено архітектуру програмного забезпечення для реалізації алгоритму електронно-цифрового підпису. Обгрунтовуються принципи та підходи, покладені в основу програмного комплексу для реалізації алгоритму електронно-цифрового підпису з використанням еліптичних кривих.. Описуються шаблони об’єктно-орієнтованого програмування, що були використані при проектуванні програмного комплексу.

У розділі 5 розглядається бізнес-проект застосування на практиці запропонованої розробки. Виконується аналіз рішень, присутніх на ринку послуг з реалізації електронно-цифрового підпису, окреслюються їхні переваги та недоліки, виконується порівняння розробленого програмного комплексу з існуючими, пропонується бізнес-стратегія просування розробленого програмного забезпечення на ринку.

Ключові слова: еліптична криптографія, електронно-цифровий підпис, ECDSA, багатократне множення точок еліптичної кривої на число, об’єктно-орієнтована архітектура програмного забезпечення.

ABSTRACT

Actuality of the topic: Nowadays the industry of cryptography and information protection plays an important role in the field of informatics. This is due to the large-scale use of automated methods for processing and transmitting data and the widespread use of methods and means of unauthorized access to information transmitted by unprotected communication channels.

The problem of transferring some confidential information to the addressee can be solved in many ways, but the most commonly used at present are the so-called asymmetric cryptosystems. A number of leading studies show that among asymmetric cryptosystems, or public-key cryptosystems, the most stable is the CS using elliptic curves. These cryptosystems to ensure sufficient cryptographic stability require a key length several times smaller than that required for cryptosystems based on mathematical operations in finite fields.

One of the brightest representatives of the family of cryptographic algorithms using elliptic curves is the algorithm of the digital signature ECDSA. When using cryptographic algorithms on elliptic curves, the time of their work is very important. Experimental studies show that the most resource and hourly expenditures are operations performed directly with the points of the elliptic curve, in particular for the ECDSA algorithm, the operation of multiple scalar multiplication of points of an elliptic curve by a number is the most resource-consuming. Thus, it is relevant to study the methods and methods for optimizing this calculation.

Object of study: key exchange processes, encryption, decryption and creation / verification of digital signatures in elliptical cryptosystems.

The subject of the study: methods for multiple scalar multiplication of points of an elliptic curve in the finite field.

Objective: to develop a method for high-speed multiplication of points of an elliptic curve by a number.

Methods of research: methods of theory, methods of algebraic geometry, methods of abstract algebra, methods of discrete mathematics and cryptography.

Scientific novelty consists in the following:

1. A modification of the method of multiple scalar multiplication of an elliptic curve point by a number is proposed, which consists in performing a special re-calculation, and unlike existing methods, it avoids the operation of multiplying the point of an elliptic curve by a number, which shows the best results of a performance comparison with the existing ones for elliptic curves with parameters over GF (2 ^ m).

2. The software architecture is developed, the feature of which is the encapsulation of the algorithm for multiple scalar multiplication of an elliptic curve point by a number, allows replacing a specific implementation in an instance of an electronic digital signature without performing an inheritance.

Practical value is as follows: the proposed algorithms allow you to quickly perform the operation of multiple scalar multiplication of points of the elliptic curve by a number, which allows reducing the execution time of the algorithm of the digital signature.

Structure of the paper note:

Section 1 of this paper describes the general context of research in the framework of this work, namely, the mathematical foundations and principles of elliptical cryptography, a review of encryption methods using elliptical cryptography, a review of the algorithms for generating and verifying an electronic digital signature using elliptical cryptography.

Section 2 describes the principles of operations on points of an elliptic curve, provides algorithms for adding and multiplying points of an elliptic curve by a number, examines the algebraic structures used to determine the parameters of an elliptic curve point and their singularities, and briefly describes the principles of implementing actions on points of an elliptic curve in electronically computing devices.

Chapter 3 describes the methods of multiple scalar multiplication of points of an elliptic curve by a number, a detailed description is given, as well as a

comparison of their characteristics with an improved method proposed in the framework of this paper.

Section 4 describes in detail the principles and approaches of software implementation of the software complex that implements the algorithm for digital signatures using elliptical curves. Describe the patterns of object-oriented programming that were used in the design of this software package.

Section 5 describes how to use this development as a startup. A detailed analysis of the solutions present on the market is carried out, their advantages and disadvantages are determined, the hypothetical developed software complex is compared with the existing ones, the business strategy for promoting this complex on the market is determined.

Keywords: elliptic curve cryptography, digital signature, ECDSA, multiple point multiplication on elliptic curve.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Шифрування – процес перетворення інформації, що захищається (відкритого тексту), в зашифроване повідомлення (шифротекст).

Дешифрування – процес, обернений до шифрування.

Еліптична крива – набір точок проективної площини, що задовольняють рівнянню разом з точкою на нескінченності.

Відкритий (вихідний) текст – дані, що зашифровуються.

Шифротекст – дані, отримані після застосування шифрування.

Асиметричний шифр – шифр, в якому використовуються два ключі, публічний та приватний, що дозволяє убезпечити шифротекст від розшифровування третьою стороною.

Публічний ключ – ключ асиметричної системи, що може бути повідомлений третій стороні.

Приватний ключ –ключ асиметричної системи, який відправник зашифрованого повідомлення повинен зберігати в секреті.

Хеш-функція – функція, яка перетворює повідомлення довільного розміру в число фіксованого розміру.

ЕК – еліптична крива.

NAF – non-adjacent form, неспарена форма.

JSF – joint sparse form, об’єднана розподілена форма.

ECDSA – elliptic curve digital signature algorithm, алгоритм електронно-цифрового підпису на еліптичних кривих.

ECDLP – elliptic curve digital signature problem, проблема дискретного логарифму на еліптичних кривих.

DBNS – double-based number system, двобазисна система числення.

ВСТУП

В сучасному світі важливість захисту інформації має надзвичайно важливу роль. З розвитком інформаційних технологій, кількість даних, що використовується в процесі людської діяльності нестримно зростає. Даний процес веде до збільшення ризиків, пов’язаних з витоками чутливої інформації та використання її третьою стороною в шкідливих цілях. В даному контексті слід згадати процес інформатизації сфер діяльності, що оперують персональними даними такими як: біометричні дані, банківські реквізити, відомості що відносяться до сфери приватного життя тощо. Також ілюстрацією даної проблематики, що говорить сама за себе, є проблема засвідчення оригінальності, незмінності та приватності даних, що передаються через відкриті канали такі як мережа інтернет.

Для засвідчення оригінальності документу або пакету даних, що передається між відправником, що гарантує свою достовірність та отримувачем, що бажає пересвідчитись у факті незмінності отриманої інформації в процесі передачі незахищеними каналами, використовується механізм електронно-цифрового підпису. Даний механізм є за своєю суттю аналогічним до процесу фізичного підписання документу, коли укладачі документу своїм підписом підтверджують свою особистість.

Існує безліч підходів, що дозволяють реалізувати механізм електронно-цифрового підпису. В межах даної роботи особлива увага приділяється методам, заснованим на теоріях алгебраїчної геометрії, зокрема теорії еліптичних кривих.

Використання еліптичних кривих для створення криптосистем було незалежно запропоновано Нілом Коблицем та Віктором Міллером у 1985 році. Ключова перевага даної методології ґрунтується на двох основних особливостях точок еліптичної кривої: розподіл результату додавання точок еліптичної кривої є у великій мірі рівномірним, тобто покриває всю область значень даної кривої; наслідком цієї особливості є проблема дискретного логарифмування для еліптичних кривих (ECDLP). Ці особливості і

обумовлюють високу криптографічну стійкість систем з використанням еліптичних кривих. З іншого боку, слід зазначити, що криптостійкість забезпечена відсутністю субекспоненційних алгоритмів вирішення проблеми дискретного логарифмування. Таким чином, знаходження субекспоненційного підходу унеможливить використання еліптичних кривих в криптографії, хоча це і стосується в однаковій мірі і інших схем шифрування.

В межах даної роботи розглядаються еліптичні криві, параметри яких визначені над скінченними полями з простою характеристикою $GF(p)$, та бінарним розширенням $GF(2^m)$.

Перевагою криптографічних схем з використанням еліптичних кривих над іншими (зокрема RSA) є значно вища швидкодія обчислення та більша криптостійкість схем з аналогічною довжиною ключів. Таким чином, ключі схеми шифрування з використанням еліптичних кривих, що забезпечують аналогічну криптостійкість що і, для прикладу, RSA матимуть значно меншу довжину, що робить їх більш портативними.

Криптосистеми з відкритим ключем не використовуються для шифрування великих об'ємів даних так як дана процедура є обчислювально складною. Тому, вирішенням даної проблеми є комбінування симетричного та асиметричного шифрування, коли криптосистема з відкритим ключем використовується для безпечної передачі секретного ключа шифрування із закритим ключем.

Загалом, задача оптимізації часу роботи алгоритму електронно-цифрового підпису є надзвичайно актуальною, адже це дозволяє пришвидшити загальний процес обміну захищеними даними.

Метою роботи є аналіз та оптимізація методів багатократного скалярного множення точки еліптичної кривої над скінченним полем $GF(p)$ та $GF(2^m)$. Наукова новизна роботи полягає в наступному:

1. Запропоновано модифікацію методу багатократного скалярного множення точки еліптичної кривої на число, яка полягає у виконанні спеціального передобчислення, і на відміну від існуючих методів, дозволяє звести виконання операції множення точки еліптичної кривої на число до додавання, що забезпечує вищу швидкодію (до 25%) порівняно з існуючими для еліптичних кривих з параметрами над $GF(2^m)$.
2. Розроблено архітектуру програмного забезпечення, особливістю якої є інкапсуляція алгоритму багатократного скалярного множення точки еліптичної кривої на число, що дозволяє на основі шаблону «Стратегія» замінювати конкретну реалізацію в екземплярі класу електронно-цифрового підпису без застосування наслідування.

Практична цінність отриманих в роботі результатів полягає в тому, що запропоновані методи та алгоритми дозволяють швидше виконувати операцію багатократного скалярного множення точки еліптичної кривої над скінченним полем $GF(p)$ та $GF(2^m)$, що в свою чергу дозволяє зменшити час перевірки електронно-цифрового підпису з використанням еліптичних кривих.

Розроблене математичне та програмне забезпечення дозволяє як виконувати електронно-цифровий підпис інформаційних повідомлень довільної довжини так і проводити дослідження та порівняльний аналіз методів багатократного скалярного множення точки еліптичної кривої над скінченним полем $GF(p)$ та $GF(2^m)$.

В розділі 1 даної роботи описується загальний контекст досліджень проведених в рамках даної роботи, а саме – математичні основи та принципи еліптичної криптографії, дослідження методів шифрування, що використовують еліптичну криптографію, дослідження алгоритмів генерування та перевірки електронно-цифрового підпису з використанням еліптичної крипторафії.

В розділі 2 описуються принципи операцій над точками еліптичної кривої, наводяться алгоритми додавання та множення точок еліптичної кривої на число, розглядаються алгебраїчні структури, що використовуються для визначення параметрів точки еліптичної кривої та їх особливості, а також описуються принципи реалізації дій над точками еліптичної кривої в електронно-обчислювальних пристроях.

Розділ 3 присвячений методам багатократного скалярного множення точок еліптичної кривої на число, приводиться їхній детальний опис, а також порівняння їх харакатеристик з покращеним методом, запропонованим в рамках даної роботи.

В розділі 4 детально описуються принципи та підходи програмної реалізації програмного комплексу, що реалізує алгоритм електронно-цифрового підпису з використанням еліптичних кривих. Описуються шаблони об’єктно-орієнтованого програмування, що були використані при проектування даного програмного комплексу.

В розділі 5 описується варіант використання даної розробки в якості стартапу. Виконується детальний аналіз рішень, присутніх на ринку, окреслюються їхні переваги та недоліки, виконується порівняння гіпотетичного розробленого програмного комплексу з існуючими, окреслюється бізнес-стратегія просування даного комплексу на ринку.

1. АНАЛІЗ КРИПТОСИСТЕМ, НА ОСНОВІ ЕЛІПТИЧНИХ КРИВИХ

1.1. Принципи еліптичної криптографії

Еліптична криптографія – це розділ криптографії, що використовує еліптичні криві з параметрами, визначеними над скінченними полями для реалізації схем шифрування. Головним напрямком застосування еліптичних кривих в криптографічних схемах є системи з відкритим ключем. Ключовим математичним об'єктом еліптичної криптографії є еліптична крива.

Еліптичною кривою E [1], визначеною над скінченним полем K, називається крива, описана рівнянням Вейєрштраса (1):

$$y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6 \quad (1)$$

де $\{a_1, a_2, a_3, a_4, a_5, a_6\} \in K,\ \Delta \neq 0$.

Δ називається дискримінантом E і визначається як:

$$\Delta = -d_2^2 d_8 - 8d_4^3 - 27d_6^2 + 9d_2 d_4 d_6$$

$$d_2 = a_1^2 + 4a_2$$

$$d_4 = 2a_4 + a_1 a_3$$

$$d_6 = a_3^2 + 4a_6$$

$$d_8 = a_1^2 a_6 + 4a_2 a_6 - a_1 a_3 a_4 + a_2 a_3^2 - a_4^2.$$

Якщо скінченне поле K є простим – $GF(p)$, E трансформується у криву, що описується таким рівнянням:

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

У випадку $K = GF(2^m)$ E трансформується у криву, описану рівнянням:

$$2 + xy = x^3 + ax^2 + b, \quad (3)$$

де $a, b \in K$.

Розглянемо арифметику над точками еліптичної кривої.

Нехай E – еліптична крива, визначена над полем K та $P, Q \in E$ – точки еліптичної кривої. Сума точок P та Q графічно визначається наступним способом [2].

1. Провести пряму через точки P та Q.
2. Відобразити перетин даної прямої з еліптичною кривою відносно осі *OY*.

Подвоєння точки P графічно визначається наступним чином.

1. Провести дотичну до еліптичної кривої в точці P.
2. Перетин даної дотичної, відображений симетрично осі *OY*, називається подвоєною точкою P.

Вимоги до еліптичної кривої

Для того, щоб уникнути відомі атаки, що засновані на проблемі дискретного логарифма у групі точок ЕК, необхідно, щоб кількість точок ЕК ділилась на достатньо велике просте число n. Стандарт ANSI X9.62 вимагає $n > 2^{160}$. Рівняння ЕК будується специфічним способом, використовуючи випадкові/псевдовипадкові коефіцієнти.

Основними параметрами при побудові ЕК над полем $GF(p)$ є:

1. Розмірність поля p, де p є простим числом.
2. Два елементи скінченного поля – a та b, визначені рівнянням еліптичної кривої E, що має наступний вигляд:

$$y^2 = x^3 + ax + b, \tag{4}$$

де $a, b \in GF(p)$ та $4a^3 + 27b^2 \neq 0 (\mathrm{mod}\ p)$.

3. Два елементи поля $GF(p) - x_G$ та y_G, які визначають кінцеву точку $G = (x_G, y_G)$ – генератор групи.
4. Порядок q точки G, де $q > 2^{160}$ та $q > 4\sqrt{p}$.
5. Співмножник $h = \#E / q$, де $\#E$ означає порядок групи точок ЕК.

Генерація основних параметрів

Один із способів генерації криптографічно надійних параметрів полягає в наступному:

1. Обираємо коефіцієнти a та b специфічним чином, використовуючи в обчисленнях випадкові/псевдовипадкові числа. Нехай E – еліптична крива $y^2 = x^3 + ax + b$ [3].
2. Обчислюємо $N = \#E$.
3. Перевіряємо, що N має дільник, який є великим простим числом $q\left(q > 2^{160},\ q > 4\sqrt{p}\right)$. Якщо не виконується, то необхідно повернутись на крок 1.
4. Перевіряємо, що q не ділить $p^k - 1$ для кожного k, $1 \le k \le 100$. Якщо не виконується, то необхідно повернутись на крок 1.
5. Перевірити, що $q \ne p$. Якщо не виконується, то необхідно повернутись на крок 1.
6. Обрати випадкову точку $G' \in E$ і покласти $G = \left(N / q\right)G'$. Повторювати, доки $G \ne O$.

1.2. Методи шифрування в еліптичній криптографії

Найбільш популярними напрямками еліптичної криптографії, тобто сферами, в яких криптостійкість базується на задачі дискретного логарифмування для еліптичних кривих або ECDLP, є шифрування з відкритим ключем та алгоритм електронно-цифрового підпису.

В даному підрозділі буде розглянуто різноманітні схеми обміну ключами. Як зазначалось раніше, існують два види використання шифрування з ключами – симетричний (існує єдиний секретний ключ, що має бути переданий між відправником та отримувачем захищеним каналом) та асиметричний (існує пара ключів – приватний та публічний, що може бути

переданий незахищеним каналом). В межах даної роботи детально розглядається другий варіант, а саме асиметричне шифрування.

1.2.1 Шифрування з відкритим ключем

Еліптичний варіант схеми обміну ключами Діффі-Хелмана

Протокол Діффі-Хелмана – це метод обміну криптографічними ключами. Даний метод дозволяє двом учасникам, що не мають жодних попередніх даних один про одного, отримати спільний секретний ключ, що використовуватиметься для шифрування даних, якими обмінюються сторони, за допомогою незахищеного каналу зв'язку. Цей ключ можна використати для шифрування наступних сеансів зв'язку, що використовують шифр з симетричним ключем.

Нехай існує еліптична крива, що забезпечує достатню криптостійкість (несуперсингулярну, в якій генеруюча точка $G=(x;y)$ має великий порядок, тобто число n, при якому $nG=O$ є дуже великим простим числом), визначена параметрами a та b:

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

Позначимо сторону відправника як A, сторону отримувача як B, тоді обмін ключами між сторонами A та B проводиться таким чином:

1. Сторона A обирає ціле число $Priv_A < n$. Дане число називається приватним ключем учасника, а точка еліптичної кривої $Pub_A = \Pr iv_A \times G$ називається публічним ключем.
2. Сторона B обирає аналогічно секретний ключ $\Pr iv_B$ та обчислює відкритий ключ $Pub_B = \Pr iv_B \times G$.
3. Учасник A генерує секретний ключ $K = k_A \times P_B$, а учасник B генерує секретний ключ $K = k_B \times P_A$.

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

$$k_A \times P_B = k_A \times (k_B \times G) = k_B \times (k_A \times G) = k_B \times P_A.$$

Проблема, що стоїть перед сторонніми спостерігачами, які мають намір дізнатись секретний ключ, полягає в обчисленні $k_A \times k_B \times G$ за відомими $G, k_A \times G, k_B \times G$, але не знаючи k_A та k_B. Це і є проблемою Діффі-Хеллмана для еліптичних кривих.

Протокол Мессі-Омура (Massey-Omura)

Криптосистема Мессі-Омура [4] була запропонована в 1978 році Джеймсом Мессі і Джимом К. Омура в якості поліпшення протоколу Шаміра. Є два варіанти реалізації даного протоколу: класичний і еліптичний. Перший побудований на складності завдання дискретного логарифмування, другий на властивостях еліптичної кривої.

Еліптичний варіант даного протоколу надає можливість передавати повідомлення від відправника A до отримувача B по відкритому каналу. Нехай порядок еліптичної кривої дорівнює N, e – ціле число, взаємно просте з N. За алгоритмом Евкліда можна знайти:

$$d \equiv e^{-1} \bmod N$$

За визначенням,

$$e \times d = j \times N + 1$$

Значить для будь-якої точки P еліптичної кривої порядку N виконується:

$$(e \times d) \times Q = (j \times N + 1)P = (j \times N) \times P + P = P + O = P$$

Тепер, використовуючи e і d і будь-яку точку P еліптичної кривої, можна обчислити:

$$Q = e \times P$$

$$P = d \times Q$$

Де $P = R$.

Обчислення точки P по $e \times P$ еквівалентно вирішенню завдання дискретного логарифма для еліптичної кривої.

Розглянемо загальну схему роботи системи шифрування з відкритим ключем. Для ілюстрації даної системи розглянемо наступну ситуацію: Необхідно забезпечити конфіденційну та захищену передачу деяких секретних даних між відправником та отримувачем використовуючи незахищений канал зв'язку, тобто такий, в якому, з ненульовою ймовірністю, існує сторона, що здатна перехоплювати всі повідомлення, що передаються даною мережею. Позначимо сторону відправника як A, сторону отримувача як B, повідомлення, що передається як m. Для організації безпечного обміну повідомленнями між A та B використовується наступна методика:

1. Сторони A та B генерують пари ключів $(Pub_A, Priv_A)$ для сторони A та $(Pub_B, Priv_B)$ для сторони B, де $(Pub, Priv)$ – публічний та приватний ключі.
2. Сторона B надсилає незахищеним каналом стороні A свій публічний ключ – Pub_B.
3. Сторона A виконує шифрування повідомлення m за допомогою деякої функції Enc, що приймає аргументи Pub та m. Результатом виконання даної функції є деяке значення $e = Enc(Pub_B, m)$.
4. Сторона A надсилає стороні B незахищеним каналом зашифроване повідомлення e.
5. Сторона B отримує зашифроване повідомлення e, та за допомогою деякої функції Dec, що приймає аргументи $Priv$ та e, отримує вихідне повідомлення $m = Dec(Priv_B, e)$.

Для зворотнього зв'язку виконуються аналогічні дії, де відправником є сторона B, отримувачем – сторона A, і замість $(Pub_B, Priv_B)$ виконуються маніпуляції з парою ключів $(Pub_A, Priv_A)$.

1.2.3 Опис алгоритму електронно-цифрового підпису

Ще однією сферою, де активно застосовується еліптична криптографія, є алгоритм електронно-цифрового підпису[5]. Розглянемо детальніше даний алгоритм, проілюструвавши його наступним прикладом. Нехай існує деякий відправник повідомлення, оригінальність та незмінність якого після передачі незахищеним каналом повинна гарантуватись, та отримувач, що повинен мати можливість переконатись у оригінальності та незмінності отриманого повідомлення. Позначимо відправника як A, отримувача як B, повідомлення як m. Для виконання алгоритму електронно-цифрового підпису необхідно виконати наступні дії:

1. Сторона A обирає деяке значення $Priv_A$, що є числом великої розрядності. Обчислюється точка еліптичної кривої $Pub_A = Priv_A \times G$, де G – генеруюча точка еліптичної кривої, тобто точка з великим порядком. Точка Pub_A називається електронно-цифровим підписом відправника A.
2. Обчислюється хеш-сума повідомлення використовуючи деяку функцію $Hash$, $h = Hash(m)$.
3. Використовуючи електронно-цифровий підпис Pub_A, виконується підпис хеш-суми повідомлення, використовуючи деяку функцію $Sign$, що приймає в якості параметрів хеш та електронно-цифровий підпис, $signature = Sign(h, Pub_A)$.
4. Відправнику B надсилається $(m, signatre, Pub_A)$.

Для перевірки електронно-цифрового підпису на стороні отримувача виконуються наступне:

1. Обчислюється хеш повідомлення $h = Hash(m)$.
2. Використовується деяка функція *Verify* , що приймає параметрами хеш-суму h, підпис повідомлення *signature*, електронно-

цифровий підпис Pub_A, та повертає булеве значення $verified = Verify(h, signature, Pub_A)$.

3. Якщо значення $verified = true$, то повідомлення є оригінальним та не було змінене в процесі передачі незахищеним каналом.

1.3. Алгоритми генерування та перевірки цифрового підпису

Еліптична криптографія знайшла своє застосування і в алгоритмах електронно-цифрового підпису, зокрема в алгоритмі ECDSA (elliptic curve digital signature algorithm). Розглянемо схеми генерування та перевірки електронно-цифрового підпису.

1.3.1. Схема цифрового підпису Ель-Гамаля

Схема Ель-Гамаля (ElGamal) [6] – схема шифрування з відкритим ключем, заснована на складності обчислення дискретних логарифмів. Дана схема може використовуватись як для шифрування так і як алгоритм електронно-цифрового підпису. Схема була запропонована Тахером Ель-Гамаль в 1985 році. Він удосконалив систему Діффі-Хеллмана і отримав два алгоритми, які призначені для шифрування і для автентифікації.

При використанні в режимі електронно-цифрового підпису, одержувач підписаного повідомлення може використовувати цифровий підпис для перевірки незмінності та оригінальності повідомлення підписаного відправником. Для перевірки електронно-цифрового підпису необхідно використовувати деяку надійну хеш-функцію. Розглянемо узагальнену схему електронного підпису Ель-Гамаля.

Генерація ключів:

1. Генерується випадкове просте число довжиною p бітів.
2. Вибирається випадковий примітивний елемент $g \in Z_p$.
3. Вибирається випадкове ціле x таке що $1 < x < p-1$.

4. Обчислюється $y = g^x \bmod p$.
5. Відкритим ключем є трійка (p, g, y), а закритим число x.

Алгоритм підпису повідомлення M:

1. Обчислити хеш повідомлення: $m = h(M)$.
2. Вибрати випадкове число $1 < k < p-1$ взаємно просте з $p-1$ і обчислюється $r = g^k \bmod p$.
3. Обчислити число $s \equiv (m - xr)k^{-1} \bmod p - 1$.
4. Підписом повідомлення M є пара (r, s).

Перевірка підпису:

1. Перевіряється справедливість умов: $0 < r < p$ і $0 < s < p-1$. Якщо хоча б одна з них не виконується, то підпис вважається невірним.
2. Обчислюється хеш повідомлення $m = h(M)$.
3. Підпис вважається вірним, якщо виконується порівняння: $y^r r^s \equiv g^m \bmod p$

1.3.2. Алгоритм ECDSA

ECDSA (Elliptic Curve Digital Signature Algorithm) – алгоритм з відкритим ключем для створення цифрового підпису, аналогічний, за своєю будовою, DSA, але визначений, на відміну від нього, не над полем цілих чисел, а у групі точок ЕК. Розглянемо використання алгоритму ECDSA з використанням еліптичної кривої з параметрами, визначеними над полем $GF(p)$.

Генерація ключів ECDSA

Нехай E – еліптична крива (ЕК), визначена над полем $GF(p)$ і P – точка високого порядку q кривої E. Відправник A конструює свій ключ, виконуючи наступні кроки:

1. Обирає випадкове ціле число x з інтервалу $[1,\ q-1]$.
2. Обчислює добуток $Q = x \cdot P$.

Відкритим ключем відправника A є точка Q, а закритим $-x$.

Обчислення цифрового підпису

Для того, щоб підписати деяке повідомлення m відправник A повинен зробити наступне [7]:

1. Обрати випадкове ціле число $k \in [1, q-1]$.
2. Обчислити $k \cdot P = (x_1, y_1)$ та $r = x_1 \bmod q$. Якщо $r \equiv 0 (\bmod q)$, то обирають нове випадкове число $k \in [1, q-1]$.
3. Обчислити $k^{-1} (\bmod q)$ та $s = k^{-1}(h + x \cdot r) \bmod q$, де h – значення хеш-функції повідомлення, що підписується. Якщо $s = 0$, то значення $s^{-1} \bmod q$ не існує, тому необхідно повернутись до п.1.

Підписом для повідомлення є пара цілих чисел (r, s).

Перевірка цифрового підпису

Для того, щоб перевірити підпис відправника A на повідомлення, отримувач B повинен зробити наступне:

1. Отримати копію відкритого ключа Q відправника A.
2. Перевірити, що числа r та s є цілими числами з інтервалу $[1, q-1]$ та обчислити значення хеш-функції h від повідомлення.
3. Обчислити $u_1 = s^{-1} h \bmod q$ та $u_2 = s^{-1} r \bmod q$.
4. Обчислити $u_1 \cdot P + u_2 \cdot Q = (x_0, y_0)$ та $v = x_0 \bmod q$.
5. Прийняти підпис, якщо $v = r$.

Можна легко показати, що даний алгоритм дійсно успішно засвідчує цифровий підпис. Якщо підпис (r, s) повідомлення m було дійсно

згенеровано з використанням секретного ключа, то $s \equiv k^{-1}(h + xr)(\mathrm{mod}\, q)$. Розкриваючи дужки отримаємо:

$$k \equiv s^{-1}(h + xr) \equiv s^{-1}h + s^{-1}rx = u_1 + u_2 x(\mathrm{mod}\, q).$$

Тоді $u_1 \cdot P + u_2 \cdot Q = (u_1 + u_2 x) \cdot P = kP$ і значить $v = r$, що і вимагається для підтвердження підпису.

1.4. Висновки до розділу 1

Загалом, враховуючи інформацію, наведену в даному розділі, можна стверджувати, що криптосистеми, засновані на еліптичних кривих, довели свою цінність та високий рівень захищенності. В даному розділі було розглянуто ряд існуючих протоколів розподілу ключів в асиметричних криптосистемах, а також алгоритмів електронно-цифрового підпису на еліптичних кривих.

У методах шифрування еліптичної криптографії та схемах цифрового підпису, що ґрунтуються на властивостях адитивної абелевої групи, утвореної точками ЕК, так само, найбільш вживаною є операція множення точки ЕК на число.

Очевидно, що у всіх методах, що використовують еліптичні криві, необхідно виконувати арифметичні операції над її точками, а саме: додавання, подвоєння та множення точок еліптичної кривої. Зокрема, в алгоритмі ECDSA, під час виконання алгоритму перевірки електронно-цифрового підпису необхідно виконувати обчислення виду $kP + lQ$, що і називається проблемою багатократного скалярного множення[8] точок еліптичної кривої на число.

Під час досліджень, проведених в рамках даної роботи, було доведено, що операція багатократного множення точок еліптичної кривої на число, є найбільш часо- та ресурсозатратною. Таким чином, підходи, що дозволяють прискорити дане обчислення, представляють велику практичну цінність.

2. АНАЛІЗ ОПЕРАЦІЙ НАД ТОЧКАМИ ЕЛІПТИЧНИХ КРИВИХ

В межах криптографії, активне застосування знаходять еліптичні криві з параметрами, визначеними над скінченними полями, а саме полями з простою характеристикою – $GF(p)$ та бінарним розширенням – $GF(2^m)$. Це в свою чергу означає, що координати точок еліптичної кривої належать деякому скінченному полю, що знімає проблему округлення значень при виконанні операцій над ними.

В даному розділі розглядаються математичні основи теорії скінченних полів, арифметичних дій над її елементами, арифметичні операції над точками еліптичної кривої та особливості програмної реалізації зазначених компонентів.

2.1. Алгебраїчні операції в скінченних полях

В даному розділі розглядаються особливості побудови скінченних полів, або полів Галуа, зокрема полів $GF(p)$ та $GF(2^m)$. Вводяться поняття характеристики поля, мультиплікативної групи, незвідного полінома, додавання, віднімання, множення елементів поля, знаходження оберненого елемента тощо. Для подання однозначного і вичерпного визначення скінченного поля необхідно дати декілька математичних визначень, що використовуються для формулювання означення поля.

Група

Група – це алгебраїчна структура, у якій визначена операція множення елементів даної множини та результат якої належить тій же структурі [9]. Перетворення в групі задовольняють такі властивості як:

1. Асоціативність: для довільних елементів a, b, c групи G виконується правило $(ab)c = a(bc)$.

2. Існування нейтрального елемента: існує елемент e такий, що для кожного елемента a групи G виконується $ae = ea = a$.
3. Існування оберненого елемента: для кожного елемента a групи G існує елемент a^{-1} такий, що $aa^{-1} = e$.

Кільце

Кільце – це алгебраїчна структура, в якій визначені операції додавання та множення з властивостями, подібними до додавання і множення цілих чисел. По суті кільце - це група з додатковою операцією додавання [10]. Перетворення в кільці в додачу до групи задовольняють такі властивості як:

1. Дистрибутивність: для довільних елементів a, b, c групи G виконується правило $(a+b)c = ac + bc$.
2. Існування нейтрального елемента: існує елемент e такий, що для кожного елемента a групи G виконується $ae = ea = a$.
3. Існування адитивної групи кільця і нейтральний елемент в ній позначають як 0.

Скінченне поле

Скінченне поле або поле Галуа поле, яке складається зі скінченної множини елементів. Ненульові елементи поля утворюють групу щодо операції множення, яка називається мультиплікативною групою поля. Ця група є циклічною, тобто вона має твірний елемент, а всі інші елементи отримуються шляхом піднесення до степеня твірного. Найменше поле Галуа містить лише два елементи – 0 та 1, арифметичні операції над якими відбуваються за модулем характеристики, тобто 2 [11].

Для будь-якого простого числа p кільце лишків $\mathrm{mod}(p)$ – це скінчене поле з елементів, яке позначається $GF(p) = \square / p\square$ Елементи цього поля можуть бути представлені цілими числами, для них визначено операції додавання та множення за модулем p. Будь-яке скінчене поле містить p^n

елементів і однозначно задається своєю характеристикою p і степенем n. Загалом, все описане вище, можна сформулювати наступним чином. Полем $F(+,*)$ є множина F, на якій визначені дві операції: додавання та множення, для яких виконуються наступні аксіоми:

1. Комутативність додавання $\forall a,b \in F : a+b=b+a$.
2. Асоціативність додавання $\forall a,b,c \in F : (a+b)+c=(b+c)+a$.
3. Існування нульового елемента $\exists 0 \in F : \forall a \in F \; a+0=a$.
4. Існування оберненого елемента відносно додавання $\forall a \in F \; \exists -a \in F : \; a+-a=0$.
5. Комутативність множення $\forall a,b \in F : a*b=b*a$.
6. Асоціативність множення $\forall a,b,c \in F : (a*b)*c=(b*c)*a$.
7. Існування нейтрального елемента $\exists e \in F : \forall a \in F \; a*e=a$.
8. Існування нейтрального елемента відносно множення $\forall a \in F \; \exists a^{-1} \in F : \; a*a^{-1}=e$.
9. Дистрибутивність додавання відносно множення $\forall a,b,c \in F : (a+b)*c=a*c+b*c$.

Побудову скінченного поля можна проілюструвати конкретним прикладом. Нехай, необхідно побудувати скінченне поле з 16 елементів. Маємо $GF(16)=GF(2^4)$, для побудови елементів поля в такому випадку необхідно обрати незвідний многочлен степеня 4, тобто такий, що не розкладається на множники. Таким многочленом є x^4+x+1.

Тоді, елементи поля задаються у вигляді многочленів – остач від ділення твірного елемента, піднесеного до степеня на незвідний многочлен. Побудову елементів поля $GF(2^4)$ проілюстровано в Табл. 1.

Табл. 1. Елементи поля $GF(2^4)$

Многочлен	**Степінь**	$1, x, x^2, x^3$
α	α	(0,1,0,0)
α^2	α^2	(0,0,1,0)
α^3	α^3	(0,0,0,1)
$1+\alpha$	α^4	(1,1,0,0)
$\alpha+\alpha^2$	α^5	(0,1,1,0)
$\alpha^2+\alpha^3$	α^6	(0,0,1,1)
$\alpha^3+\alpha+1$	α^7	(1,1,0,1)
α^2+1	α^8	(1,0,1,0)
$\alpha^3+\alpha$	α^9	(0,1,0,1)
$\alpha^2+\alpha+1$	α^{10}	(1,1,1,0)
$\alpha^3+\alpha^2+\alpha$	α^{11}	(0,1,1,1)
$\alpha^3+\alpha^2+\alpha+1$	α^{12}	(1,1,1,1)
$\alpha^3+\alpha^2+1$	α^{13}	(1,0,1,1)
α^3+1	α^{14}	(1,0,0,1)
1	α^{15}	(1,0,0,0)

Для знаходження мультиплікативно оберненого елемента (позначимо як *inv*), необхідно застосувати розширений алгоритм Евкліда. Нехай a та b цілі числа, причому $0<b\le a$. Тоді класичний розширений алгоритм Евкліда полягає у виконанні наступних кроків.

1. $u:=a;\quad v:=b;\quad A:=1;\quad B:=0;\quad C:=0;\quad D:=1.$
2. Поки $v\neq 0$
   - 2.1. $q:=\left[\frac{u}{v}\right];$
   - 2.2. $t_1:=u-q\cdot v;\qquad //\ u \bmod v$
     $t_2:=A-q\cdot C;$
     $t_3:=B-q\cdot D;$
   - 2.3. $u:=v;$
     $A:=C;$
     $B:=D.$
   - 2.4. $v:=t_1;$

$C := t_2;$

$D := t_3.$

3. $d := u;\ x := A;\ y := B.$

Результат: d, x, y, inv.

2.2. Додавання та подвоєння точок еліптичної кривої

Введемо наступні правила додавання точок ЕК:

1. Точка O виступає в ролі нульового елементу. Таким чином, $O = -O$ та для будь-якої точки P на еліптичній кривій $P + O = P$.
2. Вертикальні лінія перетинає криву в двох точках з однією й тією ж координатою x. Наприклад, $S = (x, y)$ та $T = (x, -y)$. Ця пряма перетинає криву і в нескінченно віддаленій точці. Тому $P_1 + P_2 + O = O$ та $P_1 = -P_2$.
3. Для того, щоб додати дві точки P та Q з різними координатами x, необхідно провести через ці точки пряму та знайти точку перетину її з ЕК. Якщо пряма не є дотичною до кривої в точках P або Q, то існує тільки одна така точка, позначимо її S. Відповідно до нашого припущення, $P + Q + S = 0$ $P + Q = -S$ або $P + Q = T$. Якщо пряма є дотичною до кривої в якій-небудь з точок P або Q, то в цьому випадку варто покласти $S = P$ або $S = Q$ відповідно.
4. Щоб подвоїти точку Q, варто провести дотичну в точці Q та знайти іншу точку перетину S з ЕК. Тоді $Q + Q = 2Q = -S$.

Введена таким чином операція додавання підпорядковується всім звичайним правилам додавання: комутативному та асоціативному законам.

Множина точок ЕК має такі властивості:

1. $P + O = P$
2. Якщо $P = (x, y)$, то $P + (x, -y) = O$. Точка $(x, -y)$ є від'ємним значенням точки P і позначається $-P$. Зазначимо, що $(x, -y)$ також лежить на ЕК.
3. Якщо $P = (x_1, y_1)$ та $Q = (x_2, y_2)$, де $P \neq Q$, то $P + Q = (x_3, y_3)$ визначається за наступними формулами:

$$x_3 \equiv \lambda^2 - x_1 - x_2 \pmod{p}$$
$$y_3 \equiv \lambda(x_1 - x_3) - y_1 \pmod{p}$$

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

Число λ є кутовим коефіцієнтом січної [13], проведеної через точки P та Q

При $P = Q$ січна перетворюється в дотичну, чим і пояснюється наявність двох формул для обчислення λ. У випадку визначення параметрів еліптичної кривої над деяким скінченним полем, всі арифметичні операції над даними параметрами підпорядковуються законам даного скінченного поля.

Наприклад, у випадку $GF(p)$, всі обчислення проводитимуться за модулем p, а у випадку $GF(2^m)$ – за модулем незвідного многочлена для даного скінченного поля.

2.3. Особливості програмної реалізації операцій над точками еліптичної кривої

Еліптичні криві з параметрами, що визначені над скінченними полями мають ряд істотних переваг у порівнянні з тими, що визначені над раціональними числами, а саме – теоретична ефективність реалізації в обчислювальній техніці. Це обумовлено тим, що у випадку поля $GF(p)$, числа можуть бути легко поданими у вигляді бітових слів, адже обчислювальні пристрої оперують саме з числами у двійковому поданні, а операція обчислення за модулем може бути представлена як звичайна булева функція XOR. Для поля $GF(2^m)$, многочлени, що є елементами поля можуть бути представлені у вигляді кодових двійкових слів, де кожен розряд позначає коефіцієнт при доданку-степені. Наприклад, многочлен $\alpha^3+\alpha^2+1$ може бути представлений на апаратному рівні у вигляді кодового слова «1011» або $1*\alpha^3+1*\alpha^2+0*\alpha^1+1*\alpha^0=\alpha^3+\alpha^2+1$.

Дане представлення дозволяє оперувати елементами поля як звичайними цілочисельними значеннями, коли додавання, віднімання і т.д. виконуються за модулем 2, тобто використовуючи ту саму булеву логічну функцію XOR. Що стосується алгоритмів, що необхідні для знаходження елементів поля та виконання арифметичних операцій над ними, як наприклад, алгоритм Евкліда, то даний алгоритм є за своєю природою ітераційним, що означає можливість його ефективної програмної реалізації для електронно-обчислювальних пристроїв.

Що стосується програмної реалізації операцій над точками еліптичної кривої, то слід зазначити, що операції над точками еліптичної кривої є за своєю суттю комбінацією арифметичних операцій над параметрами еліптичної кривої та координатами її точок. Таким чином еліптична крива в обчислювальній техніці може бути однозначно представлена парою цілочисельних значень координат $(x;y)$. Окремо слід зазначити випадок операцій за участю так званої точки на нескінченності. Складність даного

випадку полягає у тому, що точка на нескінченності є скоріше математичною концепцією, адже у цієї точки невизначені координати (нескінченність), що очевидно не належать скінченному полю, над яким визначена дана крива. Таким чином, результати операцій за участю точки на нескінченності повинні бути оброблені як граничні випадки, тобто аксіоматично визначеними в тілі алгоритму, а саме:

1. $O = -O$
2. $O + P = P$
3. $k * O = O$

З точки зору об'єктного проектування, операції над точками еліптичної кривої є за своєю суттю надбудовою над операціями в скінченних полях. Деталі програмної реалізації даних модулів будуть детальніше описані в розділі 4.

Множення точок еліптичної кривої на 3 та 4

Розглянемо докладніше алгоритми множення точки ЕК на 3 та 4 для ЕК у формі Вейерштраса. Найпростішим способом виконання обчислення $3 \cdot P$ та $4 \cdot P$[14], є подвійне подвоєння та додавання плюс подвоєння. В той же час існує інший підхід, який ґрунтується на обчисленні $3 \cdot P$ та $4 \cdot P$, використовуючи тільки координати точки P.

Існують три найкращі способи обчислення $3 \cdot P$ та $4 \cdot P$.

Спосіб 1:

$t_1 = (2y_1)^2;$

$t_2 = 3x_1^2 + a;$

$t_3 = t_2^2;$

$d = t_1(3 \cdot x_1) - t_3;$

$t_4 = d(2y_1);$

$inv = t_4^{-1};$

$$\lambda_1 = d \cdot inv \cdot t_2;$$
$$\lambda_2 = t_1^2 \cdot inv - \lambda_1;$$
$$y_3 = (x_1 - x_3)\lambda_2 - y_1.$$

Спосіб 2:

$$inv_1 = (2y_1)^{-1};$$
$$\lambda_1 = \left(3x_1^2 + a\right) \cdot inv_1;$$

$$inv_2 = \left(\left(3x_1^2 + a\right)^2 - 12x_1y_1^2\right)^{-1};$$

$$\lambda_2 = -\lambda_1 - 8{y_1}^3 \cdot inv_2;$$

$$x_3 = (\lambda_2 - \lambda_1) \cdot (\lambda_2 + \lambda_1 + x_1);$$
$$y_3 = -\lambda_2(\lambda_2 - \lambda_1) \cdot (\lambda_2 + \lambda_1) - y_1.$$

У даних алгоритмах, при обчисленні мультиплікативно оберненого елемента необхідно перевіряти аргумент на рівність нулю.

Нехай $P = (x_1, y_1)$, і необхідно обчислити $4P = (x_4, y_4)$.

Припустимо, що:

$$d = \left(3x^2 + a\right)\left(12x_1y_1 - 3\left(x^2 + a\right)^2\right) - 8y_1^4.$$

Легко побачити, що $d = 8y_1y_3$, де y_3 – це координата y результуючої точки. Обчислення значення d потребує одну операцію множення та 5 операцій піднесення до квадрату. Якщо a та b є малими, тоді d може бути обчислено за 4 операції піднесення до квадрату (ми вважаємо, що a^2, a^3 та $27b^2$ обчислюються попередньо та зберігаються):

$$d = y_1^4 + 3ax_1^4 - 6a^2x_1^2 + 18by_1^2 - 24abx_1 - a^3 - 27b^2.$$

Визначивши $D = (2y_1)d$ та $I = D^{-1}$, ми отримаємо:

$$\frac{1}{2y_1} = dI,\ \frac{1}{2y_3} = 8y_1^4 I.$$

Таким чином, спосіб 3 полягає у наступному:

$t_1 = x_1^2;$
$t_2 = 3x_1^2 + a;$
$t_3 = 2y_1^2;$
$t_4 = t_3^2;$
$t_5 = (x_1 + t_3)^2 - t_1 - t_4;$
$d = t_2(3t_5 - t_2^2) - 2t_4;$
$t_6 = (2y_1) \cdot d;$
$inv = t_6^{-1};$
$\lambda_1 = d \cdot inv \cdot t_2;$
$x_3 = \lambda_1^2 - 2x_1;$
$y_3 = \lambda_1(x_1 - x_3) - y_1;$
$t_7 = 3x_3^2 + a;$
$\lambda_2 = 2 \cdot t_4 \cdot inv \cdot t_7;$
$x_4 = \lambda_2^2 - 2x_3;$
$y_4 = \lambda_2(x_3 - x_4) - y_3.$

2.4. Висновки до розділу 2

В даному розділі були розглянуті основи математичної теорії скінченних полів, над якими визначається еліптична крива, яка

використовується в підходах та методах побудови криптосистем, описаних у розділі 1. Скінченні поля представляють особливий інтерес з огляду на ефективність їх застосування в апаратних та програмних реалізаціях криптосистем заснованих на еліптичній кривій, через свою очевидну близькість до апаратного (двійкового) подання значень та виконання операцій на обчислювальних приладах.

Проаналізовані скінченні поля такі як $GF(p)$ та $GF(2^m)$, принципи їх побудови та операції над їх елементами. Продемонстровано спосіб знаходження оберненого елемента в мультиплікативній групі поля за допомогою розширеного алгоритму Евкліда. Продемонстровано взаємозв'язок між теорією скінченних полів та операцій над точками, описаний особливий випадок операцій з точкою на нескінченності тощо.

Таким чином, можна зробити висновок, що оптимізація операцій з точками еліптичної кривої з параметрами, визначеними над скінченними полями $GF(p)$ та $GF(2^m)$, мають особливий практичний інтерес, адже в реаліях сучасного рівня розвитку обчислювальної техніки, криптографічні схеми майже завжди розглядаються з огляду на їх потенційне використання в сфері інформаційних технологій. Отже, скінченне поле $GF(2^m)$ якнайкраще підходить до використання в обчислювальній техніці, оскільки наслідує принципи виконання операцій в ЕОМ.

3. МОДИФІКАЦІЯ МЕТОДУ БАГАТОКРАТНОГО СКАЛЯРНОГО МНОЖЕННЯ ТОЧОК ЕЛІПТИЧНИХ КРИВИХ

Методи багатократного скалярного множення точок еліптичних кривих на число мають особливу важливість в контексті застосування алгоритму електронно-цифрового підпису з використанням еліптичних кривих. Окрім стандартного обчислення, розрізняють ряд підходів, що дозволяють значно пришвидшити даний процес. Умовно, дані методи можна розділити на 4 основні групи: методи, що ґрунтуються на передобчисленні результатів багатократного множення точок; методи що грунтуються на оптимізації представлення множників багатократного множення точок; методи що грунтуються на двобазисному представленні множників багатократного множення точок; метод так званого «плаваючого вікна».

В даному розділі буде детально розглянуто вищезгадані методи, буде проведений їх порівняльний аналіз та описано запропонову модифікацію методу багатократного скалярного множення точок еліптичних кривих на число.

3.1. Аналіз існуючих методів

На сьогоднішній день, існує безліч наукових праць та публікацій, присвячених оптимізації методів багатократного скалярного множення точок еліптичної кривої на число. В даному підрозділі буде наведено ряд методів, що демонструють принципи, спільні до тих, що були використані при розробці запропонованої модифікації.

3.1.1 Метод SMPM

В даному методі числа k та l представляються у вигляді блоків по w біт, кількістю $d = \lceil t / w \rceil$, виконується передобчислення точок $iP + jQ, 0 \le i, j < 2^w$. Тоді, числа k та l представляються у вигляді

$k = K_{d-1}K_{d-2}...K_1K_0$ $l = L_{d-1}L_{d-2}...L_1L_0$. Для обчислення бажаного результату необхідно обчислити $R \leftarrow 2^w R + (K_i P + L_i Q)$ $d-1$ раз [15].

Алгоритм виконання:

1. Обчислити $iP + jQ$ для всіх $i, j \in [0, 2^{w-1}]$.
2. Записати $k = K_{d-1}K_{d-2}...K_1K_0$ та $l = L_{d-1}L_{d-2}...L_1L_0$ де всі K_i та L_i мають довжину w біт та $d = \lceil t / w \rceil$.
3. $R \leftarrow \infty$.
4. Для кожного i від $d-1$ до 0 виконати $R \leftarrow 2^{\omega} R$.
5. Повернути R.

Даний метод має істотний недолік [16] – швидкодія даного методу обернено пропорційна кількості передобчислень. Це означає, що для досягнення великої швидкодії необхідно виконати нераціонально велику кількість передобчислень, для зберігання яких, необхідно виділити постійну пам’ять у нераціонально великих розмірах. Дана закономірність зображена на Рис. 1.

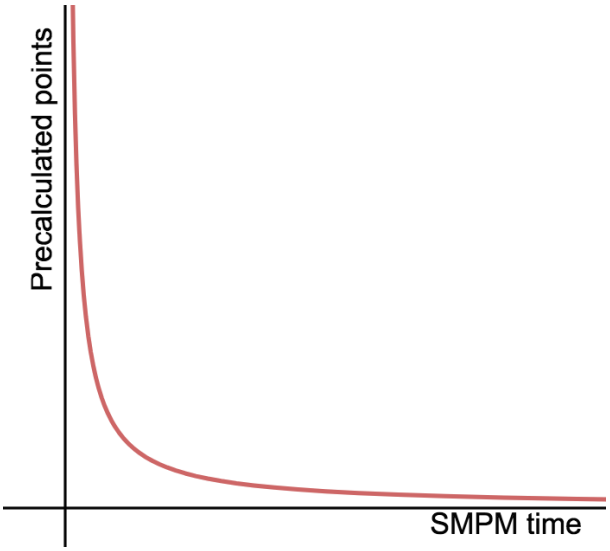


Рис. 1. Залежність швидкодії методу від кількості передобчислених точок

3.1.2 Метод SMPM з використанням NAF та JSF

З використанням фіксованого розміру блоків, в середньому виконується $3t/4$ операцій додавання точок для обчислення $kP + lQ$. При

використанні подання NAF, кількість операцій додавання знижується до $5t/9$. При використанні ж форми JSF, дана кількість знижується до $t/2$. При використанні даних підходів необхідно виконати передобчислення точок $(\pm P \pm Q)$. При використанні JSF кількість операцій подвоєнь дорівнює t. Нижче приведений алгоритм знаходження форми JSF для довільних k_1 та k_2.

1. $l \leftarrow 0,\ d_1 \leftarrow 0,\ d_2 \leftarrow 0$
2. Доки ($k_1 + d_1 > 0$ або $k_2 + d_2 > 0$)
   - 2.1. $l_1 \leftarrow d_1 + k_1,\ l_2 \leftarrow d_2 + k_2$
   - 2.2. Для i від 1 до 2 виконати:
     - 2.2.1. Якщо $l_i \bmod 2 \equiv 0$ $u = 0$ інакше
       - 2.2.1.1 $u \leftarrow l_i \bmod 4$
       - 2.2.1.2 Якщо $l_i \bmod 8 \equiv \pm 3$ та $l_{3-i} \bmod 4 \equiv 2$ тоді $u \leftarrow -u$
     - 2.2.2. $k_l^i \leftarrow u$
   - 2.3. Для i від 1 до 2 виконати:
     - 2.3.1. Якщо $2d_i = 1 + k_l^i$ тоді $d_i \leftarrow 1 - d_1$
     - 2.3.2. $k_i = k_i / 2$
   - 2.4. $l \leftarrow l + 1$
3. Повернути $JSF(k_1,\ k_2)$

При використанні даного підходу кількість передобчислень зменшується до чотирьох: $\{ P + Q;\ P - Q;\ -P - Q;\ -P + Q \}$. Перевагою JSF над NAF є визначеність розподілу ненульових елементів подання для визначених k та l. Наприклад, числа 53 та 102 в поданні NAF мають об'єднану відстань Хемінга $H = 8$, в той час як для подання JSF об'єднана відстань Хемінга становить $H = 6$.

При обчисленні $R = kP + lQ$ з використанням JSF та NAF використовується форма $k = k_{d-1}k_{d-2}..k_1k_0$ де $k_i \in \{-1, 0, 1\}$ та $l = l_{d-1}l_{d-2}..l_1l_0$ де $l_i \in \{-1, 0, 1\}$. Обчислюючи $R \leftarrow 2R + (k_iP + l_iQ)$ отримуємо результат [17].

3.1.3 Метод, що базується на використанні форми DBNS

Двобазисна система подання числа (англ. DBNS- double based number system) - це схема подання чисел, при якій кожне невід'ємне ціле число n представляється у вигляді сум та різниць добутків степенів 2 та 3 [19]. Математично дане представлення можна зобразити як: $n = \sum_{i=1}^{M} s_i 2^{b_i} 3^{t_i}$ при $s \in \{1,-1\}$. Для довільного цілого числа, в загальному випадку, існує велика кількість варіантів подання у формі DBNS. Наприклад, лише для $s = 1$ число 100 має рівно 427 різних варіантів форми DBNS, а число 1000 – рівно 1,295,579 DBNS-представлень.

Серед всіх існуючих варіантів подання DBNS існує таке, що має мінімальну кількість доданків. Таке представлення називається канонічним. В описаному в даному розділі методі необхідно обчислювати кількість елементів DBNS-подання, що менші за деяке число 2^n. Дане значення дозволяє знайти необхідну кількість подвоєнь та потроєнь точок, необхідних для обчислення кінцевого результату.

В даному методі, так само як і в методі, описаному в пункті 3.1.1, числа k та l подаються у вигляді блоків довжиною ω біт. Виконується передобчислення всіх елементів подання DBNS, що менші за 2^{ω} для точок Q, $P+Q$ та $P-Q$. Таким чином, замість операції множення точки на число, з'являється можливість виконати додавання множників виду $kP + lQ = \sum_{i=0}^{M} s^1{}_i 2^{k^1{}_i} 3^{t_i^1} P + s^2{}_i 2^{k^2{}_i} 3^{t_i^2} Q$. Графічно процес обчислення в даному методі представлений на Рис. *2*.

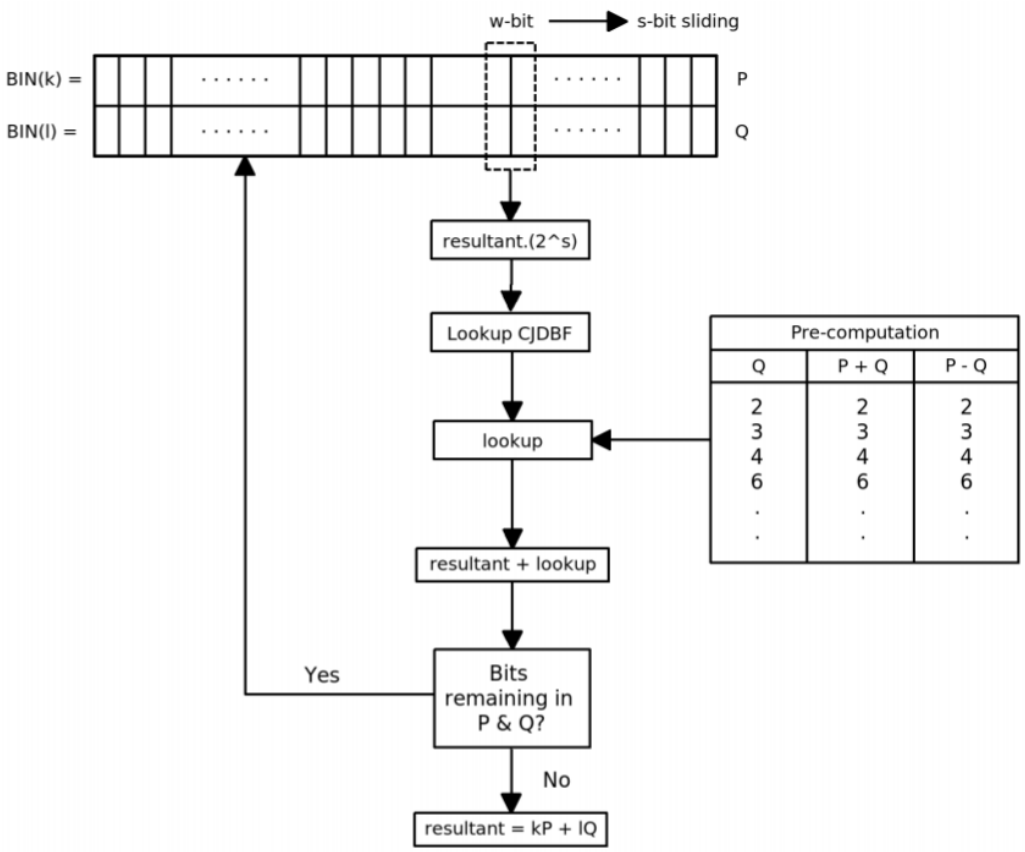


Рис. 2. Схема роботи методу, що базується на використанні форми DBNS

3.2. Модифікований метод на основі подання чисел у вигляді JSF

В методі, описаному в пункті 3.1.2. пропонується передобчислення точок еліптичної кривої: $\{P+Q; P-Q; -P-Q; -P+Q; P; Q; -P; -Q\}$ з виконанням зсуву вліво на один розряд результату після кожної ітерації [20]. Нами пропонується виконання передобчислення зсувів для всіх комбінацій значень $d_i P + d_j Q,\ d_{i,j} \in \{-1, 0, 1\}$. Таким чином, виконується передобчислення $2^n (d_i P + d_j Q),\ d_{i,j} \in \{-1, 0, 1\}$. Наприклад, для $n = 2$, таблиця передобчислень містить $2*8 = 16$ значень, а саме: $\{P+Q;\ P-Q;\ -P-Q;\ -P+Q;\ -P;\ -Q;\ P;\ Q;\ 2P+2Q;\ 2P-2Q;$ $-2P-2Q;\ -2P+2Q;\ -2P;\ -2Q;\ 2P;\ 2Q\}$.

Наприклад, для обчислення значення $R=53P+102Q$ виконуються наступні дії:

1. Отримати подання JSF для чисел $k = 53$ та $l = 102$. $k = \{1, 0, 0, -1, 0, -1, -1\}$, $l = \{1, 0, -1, 0, 0, 1, 1, 0\}$.
2. Для кожного номера розряду i отримати значення з таблиці передобчислених точок, тобто значення $Val = 2^i k_i P + 2^i l_i Q$. Додати до результату. Наприклад, для $i = 2$, $R = R + Val$, де $Val = 2^2 * 0 * P + 2^2 * (-1) * Q = -4Q$.

При використанні даної модифікації, час перевірки електронно-цифрового підпису значно зменшується для еліптичних кривих з параметрами, визначеними над полем $GF(2^m)$. Це пов'язано з тим, що при використанні даної модифікації, всі операції множення відбуваються на етапі формування таблиці передобчислень, а в процесі обчислення результату, виконується лише операція додавання точок еліптичної кривої. Назвемо запропоновану модифікацію методу багатократного множення точок еліптичної кривої JSFImp (JSF Improved).

3.3. Порівняльний аналіз модифікованого та відомих методів

Для виконання порівняння методів багатократного множення точки еліптичної кривої на число проведено замір часу перевірки електронно-цифрового підпису за алгоритмом ECDSA із застосуванням існуючих методів багатократного множення точок еліптичної кривої та запропонованої модифікації. В якості вхідних даних використані всі еліптичні криві, що запропоновані стандартом SECP[23] (Рис. 3, Табл. 2,

Табл. *3*).

Табл. 2. Значення часу перевірки електронно-цифрового підпису для еліптичних кривих з параметрами визначеними над полем $GF(p)$, (мс)

	SMPM4	SMPM5	SMPM6	SMPM7	NONE	JSF	NAF	JSFImp
SECP112R1	11,9	10,6	7,2	7,9	9,2	16,7	12,8	4,2
SECP112R2	4,3	3,9	6,7	5,1	7,3	16,4	16,3	3,2
SECP128R1	11,1	5,5	6,2	8,5	7,9	10,4	25,3	5
SECP128R2	5,3	5,2	4,8	25,4	13,4	9,5	11,6	4,3
SECP192K1	19,3	13,9	18,3	12,5	14,7	26,4	29,4	11,4
SECP192R1	13,8	14,6	17,1	20,5	16	22,5	22,1	11,6
SECP224K1	31,3	16,4	15,5	13,8	19,8	30,9	28,8	16,6
SECP224R1	21,2	13,9	14	14,8	25,3	30,1	32,3	17,9
SECP256K1	19,2	25,8	18,5	19,4	25,8	42,8	46	25
SECP256R1	18,8	18,4	19,5	23,3	29,7	54,6	40,7	23,5
SECP384R1	49,9	50,8	46,4	47	68,3	108,2	114,9	66,8
SECP521R1	110,6	123,1	107,9	106,1	145	246,3	238,7	166

Табл. 3. Значення часу перевірки електронно-цифрового підпису для еліптичних кривих з параметрами визначеними над полем $GF(2^m)$, (мс)

	SMPM4	SMPM5	SMPM6	SMPM7	NONE	JSF	NAF	JSFImp
SECT113R1	119,63	132,76	113,79	107,37	394,04	272,85	212,08	26,72
SECT163K1	382,31	448,27	343,17	308,83	417,16	437,67	398,19	71,92
SECT163R1	348,89	284,97	329,49	310,63	385,1	378,53	349,87	82,76
SECT163R2	349,55	284,17	296,77	336,07	399,61	333,97	349,3	77,83
SECT233K1	882,59	1027,2	683,72	1066,35	922,84	829,23	799,91	164,93
SECT233R1	1066,4	1052,14	969,35	818,54	1241,4	901,1	880,04	172,2
SECT239K1	791,88	807,24	834,56	751,31	1255,24	1061,03	1089,73	191,7
SECT283K1	1141,25	1133,76	1166,88	1132,89	1852,08	1326,8	1339,85	302,2
SECT283R1	1179,77	1123,88	1136,88	1216,29	1558,77	1470,38	1479,51	296,15
SECT409K1	3187,36	3244,52	3235,17	3044,42	4202,8	3540,57	4149,11	797,64
SECT409R1	3022,08	3307,61	2810,1	3047,54	3900,77	3670,78	3452,5	856,4
SECT571K1	7165,59	8101,03	7443,83	6889,09	10513,73	8386,25	8420,83	1811,41

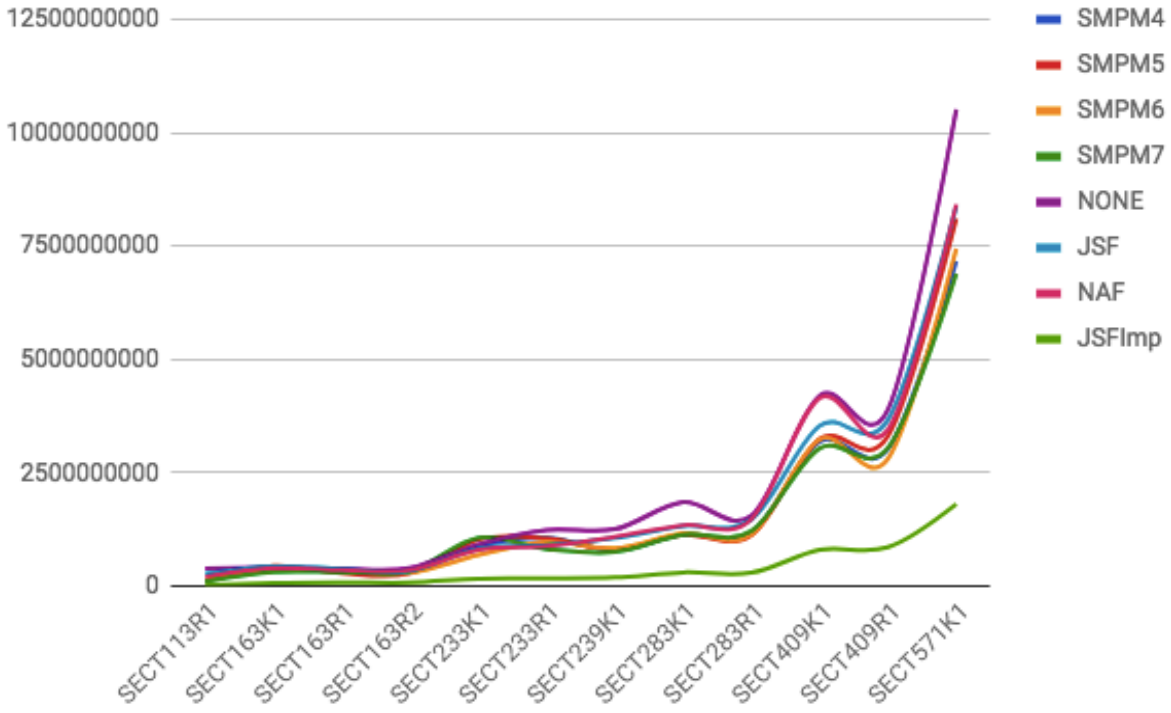


Рис. 3. Залежність часу перевірки електронно-цифрового підпису при застосуванні різних методів багатократного множення точок еліптичної кривої

3.4. Висновки до розділу 3

Запропонована модифікація дає вищу швидкодію при перевірці електронно-цифрового підпису за алгоритмом ECDSA над еліптичною кривою з параметрами, визначеними над скінченним полем $GF(2^m)$ Обчислювальна складність виконання передобчислень для запропонованої модифікації є набагато меншою ніж при використанні методу SMPM, який показує найкращі результати серед існуючих методів: $O(8n)$ проти $O(2^\omega * 2^\omega)$ [24].

4. ПРОГРАМНИЙ КОМПЛЕКС ДЛЯ РЕАЛІЗАЦІЇ ОПЕРАЦІЙ НАД ТОЧКАМИ ЕЛІПТИЧНИХ КРИВИХ

Для програмної реалізації запропонованих методів оптимізації багатократного скалярного множення точок еліптичної кривої на число необхідно було провести грунтовне дослідження програмних засобів, підходів та комплексів, які б забезпечили найбільш оптимальні можливості для успішної їх практичної імплементації.

4.1. Вибір мови програмування

В усіх асиметричних криптосистемах найважливішою складовою є секретний та відкритий ключі, за допомогою яких і відбувається шифрування/дешифрування, створення та перевірка цифрового підпису. Ці ключі мають бути досить великих розмірів (наприклад, по 1024 біта кожен[25]) для забезпечення криптостійкості алгоритму. Тому, при реалізації алгоритмів виконання низькорівневих та високорівневих операцій постає задача виконувати операції над дуже великими числами (більшими за розрядність класичних цілочисельних типів як int).

Іншим важливим моментом, що був врахований на стадії проектування програмного комплексу, є наявність очевидних моделей та абстракцій, що дозволяють ефективно моделювати предметну галузь. Такими абстракціями є скінченні поля, елементи скінченних полів, еліптичні криві тощо. Дані абстракції мають різну реалізацію, як наприклад скінченні поля $GF(p)$ та $GF(2^m)$, де в одному випадку елементи можуть бути подані у вигляді цілочисельних значень, а у іншому – у вигляді многочленів, мають різну логіку виконання операцій над елементами, тощо, але мають ряд спільностей, як скажімо, однакові структури викликів операцій над елементами. Так і еліптичні криві з параметрами, визначеними над різними типами скінченних полів мають за своєю суттю абсолютно однаковий шаблон для викликів операцій над їх точками [26].

Дана особливість спонукала нас до створення ефективної об’єктно-орієнтованої моделі, що дозволяла б моделювати предметну галузь з високим рівнем абстракції. Розроблена нами архітектура буде детально описана в даному розділі.

В ході досліджень було виявлено три основні мови програмування, що найбільш повно забезпечують можливості ефективної реалізації означеного програмного комплексу – Java, C++ та Python.

C++

C++ – мова програмування високого рівня з підтримкою об'єктно-орієнтованої, узагальненої та процедурної парадигм програмування. Розроблена Б'ярном Страуструпом та початково отримала назву «Сі з класами». Базується на мові С. Вперше описана стандартом ISO/IEC 14882:1998, найбільш актуальним же є стандарт ISO/IEC 14882:2014.

У 1990-х роках C++ стала однією з найуживаніших мов програмування загального призначення. Мову C++ використовують для системного програмування, розробки програмного забезпечення, написання драйверів, потужних серверних та клієнтських програм, а також для розробки розважальних програм, наприклад, відеоігор. C++ суттєво вплинула на інші популярні сьогодні мови програмування: C# та Java. При створенні C++ прагнули зберегти сумісність з мовою С.

Нововведеннями C++ порівняно з С є:

1. підтримка об'єктно-орієнтованого програмування через класи.
2. підтримка узагальненого програмування через шаблони.
3. доповнення до стандартної бібліотеки.
4. додаткові типи даних.
5. обробка винятків.
6. простори імен.
7. вбудовані функції.
8. перевантаження операторів.

9. перевантаження імен функцій.

10. посилання і оператори управління вільно розподіленою пам'яттю.

До стандарту C++ входить також Стандартна Бібліотека Шаблонів (STL) [27]. Вона надає такі важливі інструменти, як контейнери об'єктів, ітератори що надають доступ до цих контейнерів як до масивів тощо. Крім того, STL дозволяє схожим чином працювати і з іншими типами контейнерів, наприклад, асоціативними списками, стеками, чергами.

Основним способом організації інформації в C++ є класи. На відміну від структур (struct) мови C, що складається тільки з полів, клас C++ складається з полів і функцій-членів або методів.

Основною перевагою мови програмування C++ є можливість безпосередньго виділення та управління пам'яттю. Це дозволяє створювати нові структури данних, що можуть задовольнити вимоги до такого ключового елемента нашого програмного комплексу як елемент скінченного поля.

Java

Java – об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems». Програми, написані на Java компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретного типу платформи, на якій виконується програма. Під «незалежністю від архітектури» мається на увазі те, що програма, написана на мові Java, працюватиме на будь-якій підтримуваній апаратній чи системній платформі без змін у початковому коді та перекомпіляції. На противагу C++, Java з точки зору архітектури даної мови, є більш об'єктно-орієнтованою. Всі дані і дії групуються в класи об'єктів. Виключенням з повної об'єктності є примітивні типи (int, float тощо), що дозволяють збільшити ефективність виконання алгоритмічних задач.

У Java всі об'єкти є успадкованими від головного об'єкта (Object). На відміну від C++, у Java можливе тільки одинарне успадкування, завдяки чому

виключається можливість конфліктів між членами класу (методи і змінні), які успадковуються від базових класів. Таким чином, мова програмування Java є ідеальним кандидатом для побудови об’єктно-орієнтованої архітектури, що була обрана на стадії проектування програмного комплексу. Java використовує автоматичний збирач сміття (garbage collector) для керування пам'яттю під час життєвого циклу об'єкта [28].

Віртуальна машина сканує кучу (heap) на наявність непотрібних більше об'єктів. Коли до певного об'єкта вже не залишається посилань, збирач сміття може автоматично прибирати його із пам'яті. Проте, витік пам'яті все ж може статися, якщо код, написаний програмістом, має посилання на вже непотрібні об'єкти, наприклад на об'єкти, що зберігаються у діючих контейнерах.

Однією з важливих переваг платформи Java є наявність спеціальних класів для роботи з цілими числами довільної довжини – клас BigInteger. Даний клас в нашому програмному комплексі дозволяє реалізувати основні операції над елементами скінченного поля та ефективно передставляти параметри еліптичної кривої.

Python

Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня з строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. В мові програмування Python підтримується такі парадигми програмування, як: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Основними її перевагами є:

1. Чистий та інтуїтивно зрозумілий синтаксис.
2. Портабельність програм пов’язана з інтерпретативнісю.
3. Наявність вбудованих модулів,шо надають можливість використовувати колекції, особливі структури даних, математичний пакет тощо.

4. Можливість програмування на Python в скрипковому режимі (з консолі).
5. Зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини).
6. Політика відкритого коду (open source).

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ. Інтерпретатор мови Python і Стандартна бібліотека можуть вільно розповсюджуватись та використовуватись.

Інтерпретатор мови Python може бути розширений функціями та типами даних, розробленими на C або на іншій мові, яку можна викликати із C [29]. Python також зручна як мова розширення для прикладних програм, що потребують подальшого налагодження.

Описані вище особливості роблять Python привабливим кандидатом, що дозволив би розробити програмний комплекс для роботи з еліптичними кривими та дослідження методів багатократного скалярного множення точок еліптичної кривої на число.

З огляду на особливості обраних для порівняння мов програмування, можливості які вони надають, та потенційну ефективність розробки, було прийнято рішення обрати мову програмування Java як головний інструмент розробки програмного комплексу. Дана мова програмування надає можливість розробки якісної об’єктно-орієнтованої моделі, що ефективно моделює предметну галузь та надає для цього ряд стандартних засобів, як то класи для роботи з цілочисельними значеннями довільної довжини, стандартні контейнери об’єктів, наявність шаблонних типів, можливості написання високоефективних алгоритмів в процедурному стилі тощо.

4.2. Архітектура програмного комплексу

Розроблений програмний комплекс умовно можна розділити на три основні частини: модуль операцій в скінченних полях; модуль операцій над точками еліптичної кривої; модуль електронно-цифрового підпису. Також в рамках даної роботи був розроблений модуль оцінки швидкодії виконання операції електронно-цифрового підпису в залежності від обраного методу оптимізації методу багатократного скалярного множення точки еліптичної кривої на число.

4.3. Модуль виконання операцій в скінченних полях

Модуль виконання в скінченних полях виконаний в об'єктно-орієнтованому стилі з використанням високого рівня абстракції – публічними є лише класи-абстракції, а саме FiniteField, FiniteFieldElement, FiniteFieldFactory, FiniteFieldAtithmetics, FiniteFieldElementIterator. В той самий час, всі конкретні реалізації, а саме реалізації для скінченних полів $GF(p)$ та $GF(2^m)$ мають лише пакетну видимість.

Даний принцип приховування реалізації цілком задовольняє одні з основоположних принципів в об’єктно орієнтованому моделюванні - SOLID принципи[30]. Під час проектування даного модуля були використані підходи описані нижче. Ієрархії класів, що моделюють скінченні поля зображено на Рис.4. Ієрархія класів, що моделюють скінченні поля за допомогою діаграми класів мови моделювання UML.

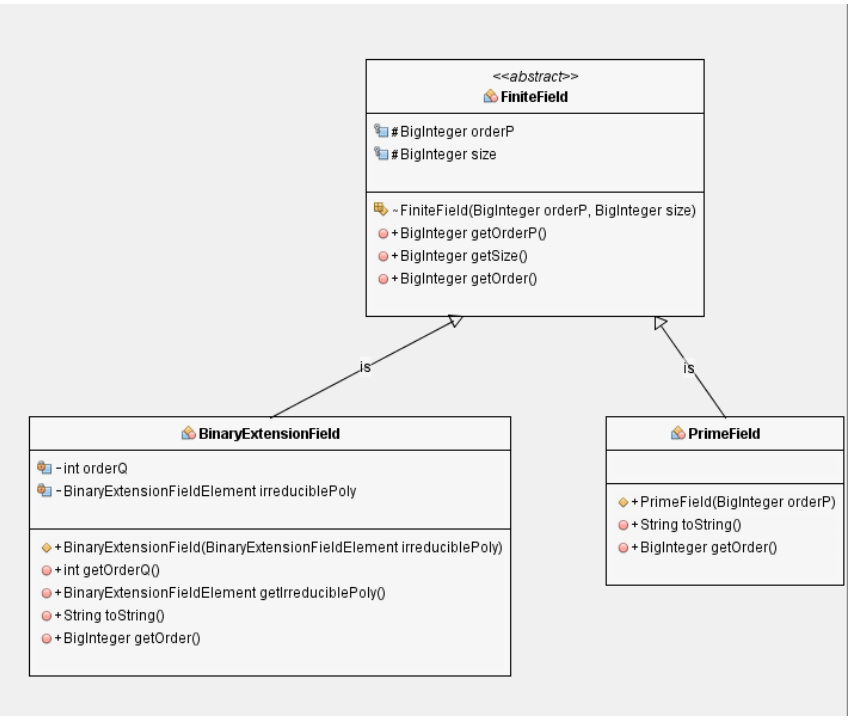


Рис.4. Ієрархія класів, що моделюють скінченні поля

Створення конкретного елемента скінченного поля в залежності від його типу за допомогою шаблона «Фабричний метод»

В залежності від типу скінченного поля $-GF(p)$ або $GF(2^m)$, елементи даного поля відрізняються у ряді своїх властивостей, а саме – відображення, тобто у вигляді числа або у вигляді многочлена, та способа зберігання у пам'яті, тобто одні зберігаються як звичайні цілочисельні значення, а інші мають додаткову характеристику – степінь многочлена. Дану проблему в межах нашої роботи пропонується вирішувати за допомогою шаблона «Фабричний метод» [31], принцип якого полягає у тому, що кожна конкретна реалізації інтерфейсу «фабрика» з єдиними методом «створити елемент» по різному його реалізує, а саме створює деякий клас-потомок із ієрархії наслідування класу «Елемент скінченного поля». Графічно даний підхід представлений на Рис. 5 за допомогою діаграми класів мови моделювання UML.

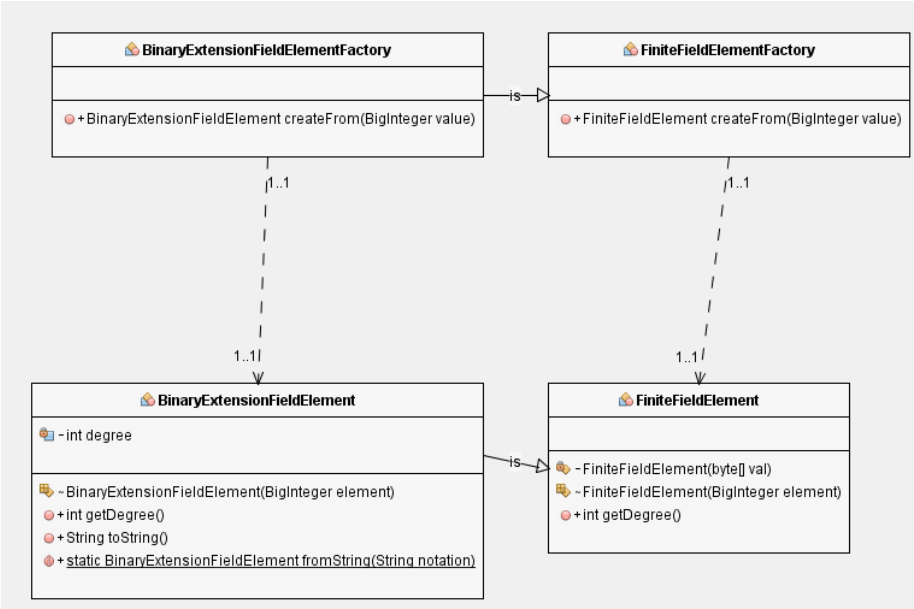


Рис. 5. Модель створення конкретного елемента скінченного поля в залежності від його типу

Ітерування елементів скінченного поля за допомогою шаблона «Ітератор»

Враховуючи той факт, що скінченні поля будуються за певними математичними принципами, описаними докладно в розділі 2 даної роботи, було би недоцільно виділяти сталий об’єм пам’яті для зберігання елементів поля, наприклад використовуючи колекції, для вирішення даної проблеми був застосований шаблон «Ітератор»[32], що дозволяє перебирати елементи деякої колекції, не знаючи її розмірів. В результаті, під час ітерування, елементи поля вираховуються базуючись на поточному елементі ітерування. Це дозволяє обходити елементи скінченного поля, динамічно ініціазуючи їх у пам’яті. Більше того, невикористовувані елементи-об’єкти в пам’яті будуть знищені збирачем сміття під час наступного проходу.

Графічно даний підхід представлений на Рис. 6 за допомогою діаграми класів мови моделювання UML.

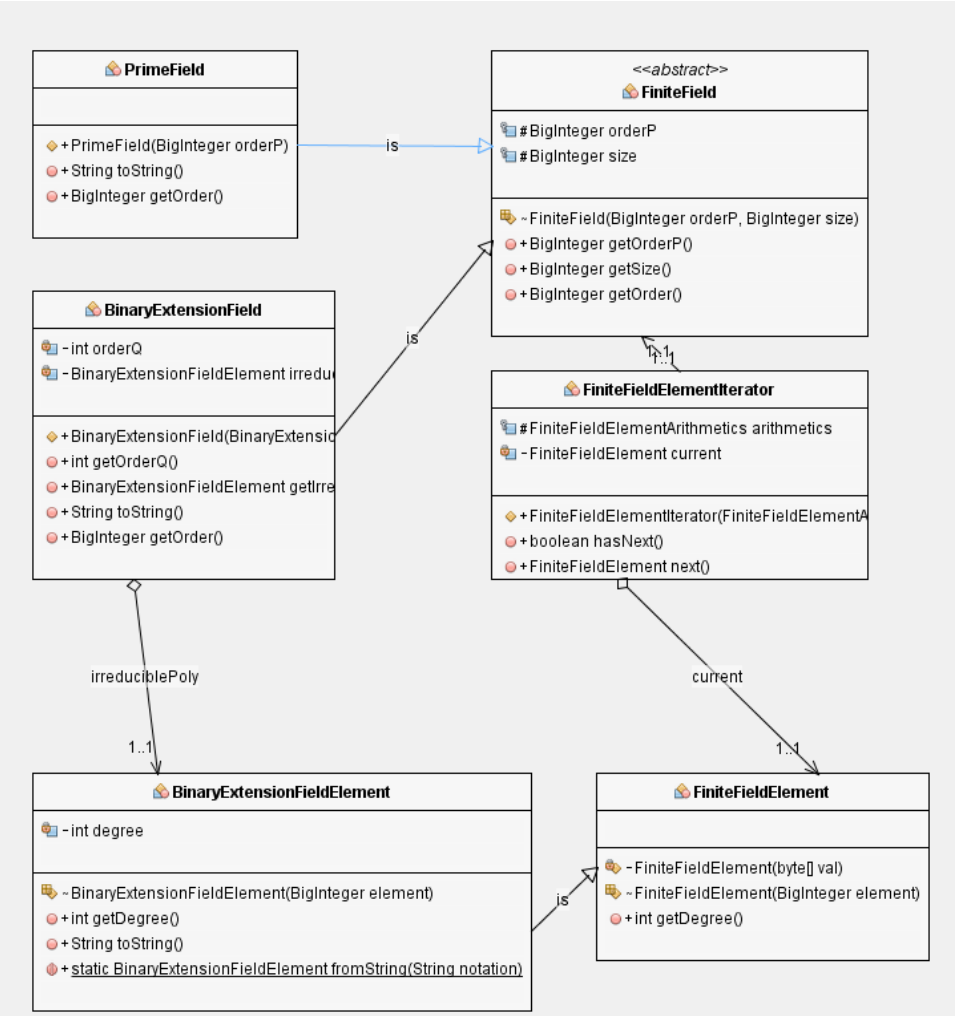


Рис. 6. Модель ітерування елементів скінченного поля

Інкапсуляція виконання арифметичних операцій над елементами скінченного поля

Для виконання арифметичних операцій над елементами скінченного поля таких як додавання, віднімання, множення елементів, а також обчислення оберненого елемента було прийнято рішення виділити абстракцію, що відповідає за виконання операцій. Даний підхід дозволяє інкапсулювати конкретні реалізації алгоритмів, що використовуються в процесі арифметичних обчислень, залишаючи можливість для створення класів-нащадків, що містили б різноманітні оптимізації для покращень характеристик даних обчислень.

Графічно даний підхід представлений на Рис. 7 за допомогою діаграми класів мови моделювання UML.

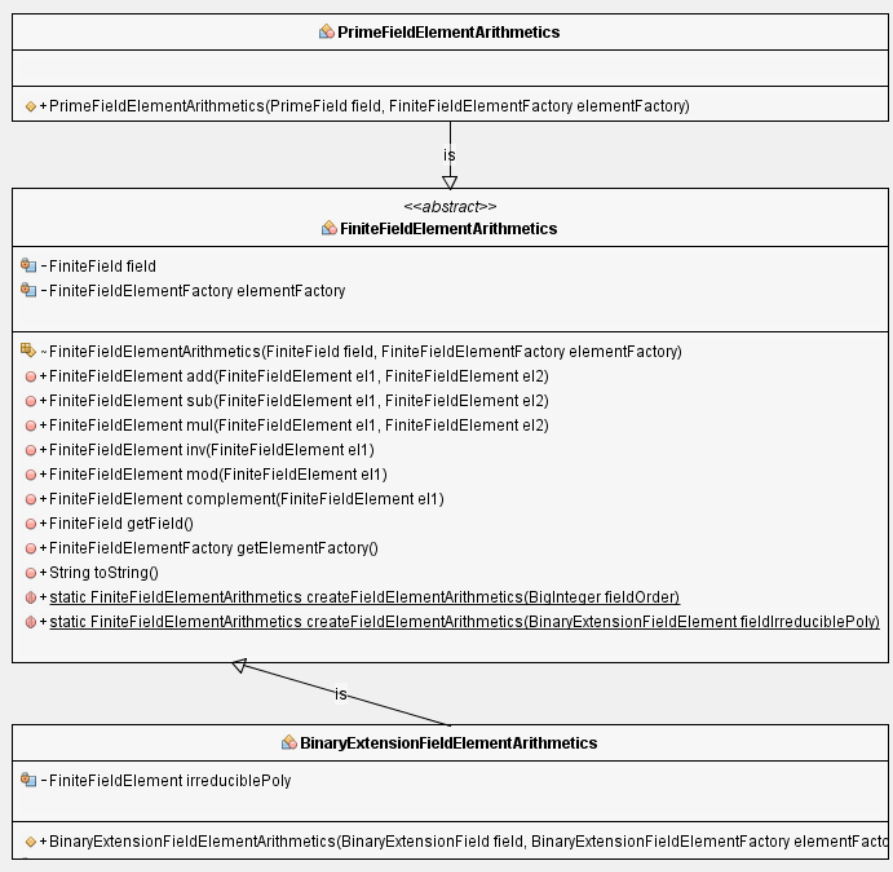


Рис. 7. Модель інкапсуляції виконання арифметичних операцій над елементами скінченного поля

4.4. Модуль виконання операцій над точками еліптичної кривої

Як було зазначено в попередніх розділах, в межах даної роботи особливу увагу приділяється дослідженню еліптичних кривих з параметрами, визначеними саме над скінченними полями $GF(p)$ та $GF(2^m)$. З огляду на це, модуль операцій над точками еліптичної кривої нерозривно пов'язаний з модулем виконання операцій над елементами скінченного поля. З іншого боку, за принципом мінімальної поінформованості в об'єктно-орієнтованій архітектурі програмного забезпечення, необхідно прагнути до зменшення залежності між модулями програмного комплексу.

У розробленому програмному комплексі для виконання операцій над точками еліптичної кривої, за рахунок прийнятих архітектурних рішень, описаних в даному розділі, вдалося досягти залежності лише від однієї абстракції – класу реалізації арифметики над елементами скінченного поля. Всі інші елементи даного модуля використовуються неявно, а саме в тілі конкретних реалізацій класів арифметики. Таким чином розроблений модуль задовольняє вимогу мінімальної поінформованості.

Інкапсуляція виконання арифметичних операцій над точками еліптичної кривої

Аналогічно до принципу інкапсуляції операцій над елементами скінченного поля, з огляду на принцип об'єктно-орієнтованого проектування, що стверджує, що класи повинні бути закритими до модифікації але відкритими для розширення, було прийнято рішення також інкапсулювати арифметичні операції над точками еліптичної кривої. Графічно даний підхід представлений на Рис. 8 за допомогою діаграми класів мови моделювання UML.

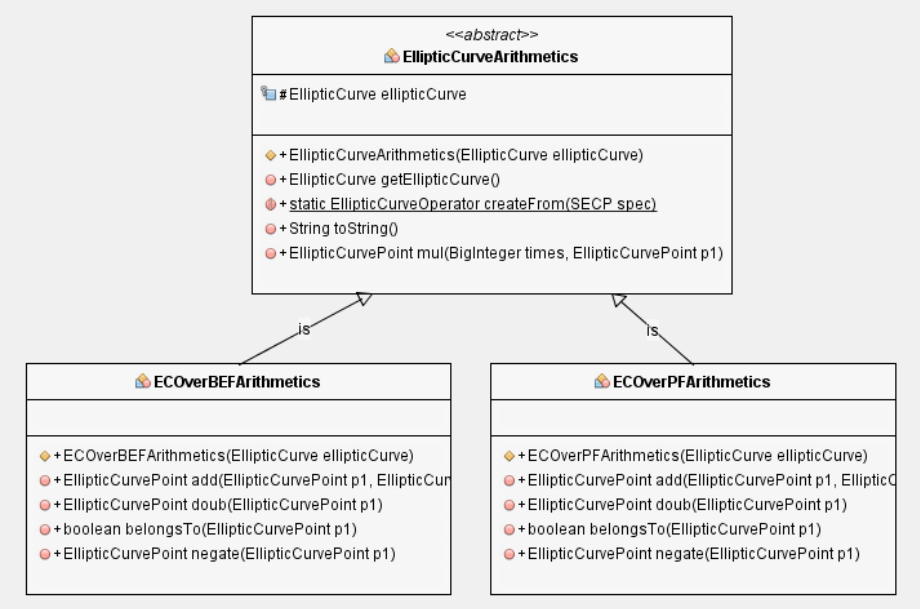


Рис. 8. Модель інкапсуляції виконання арифметичних операцій над елементами скінченного поля

Обробка граничних випадків, пов'язаних з операціями з точкою на нескінченності з використанням шаблону «Декоратор»

Одними із граничних випадків (corner cases) під час виконання операцій над точками еліптичної кривої є операції за участю точки на нескінченності. Детальніше дана проблема була розглянута в розділі 2. В контексті розробки програмного комплексу для виконання операцій з еліптичними кривими, складність реалізації даної функціональності пов'язана з тим, що еліптична точка на нескінченності не може бути виражена у вигляді цілочисельних координат, адже вважається, що вона знаходиться на нескінченності.

Для вирішення даної проблеми було використано ряд підходів, що забезпечили ефективне подолання даної колізії, зокрема декларування статичного члена класу точки еліптичної кривої, що грає роль точки на нескінченності та використання шаблона «Декоратор» [33] для попередньої перевірки вхідних параметрів методів, що приймають об'єкти-точки еліптичної кривої в якості аргументів.

Даний підхід дозволяє логічно розділити граничні випадки від самих алгоритмів, не порушуючи при цьому ієрархію класів, адже розроблений клас-декоратор має конструктор, що приймає клас арифметики, а також і сам наслідує даний клас. Графічно даний підхід представлений на Рис. 9 та Рис. 10 за допомогою діаграми класів мови моделювання UML.

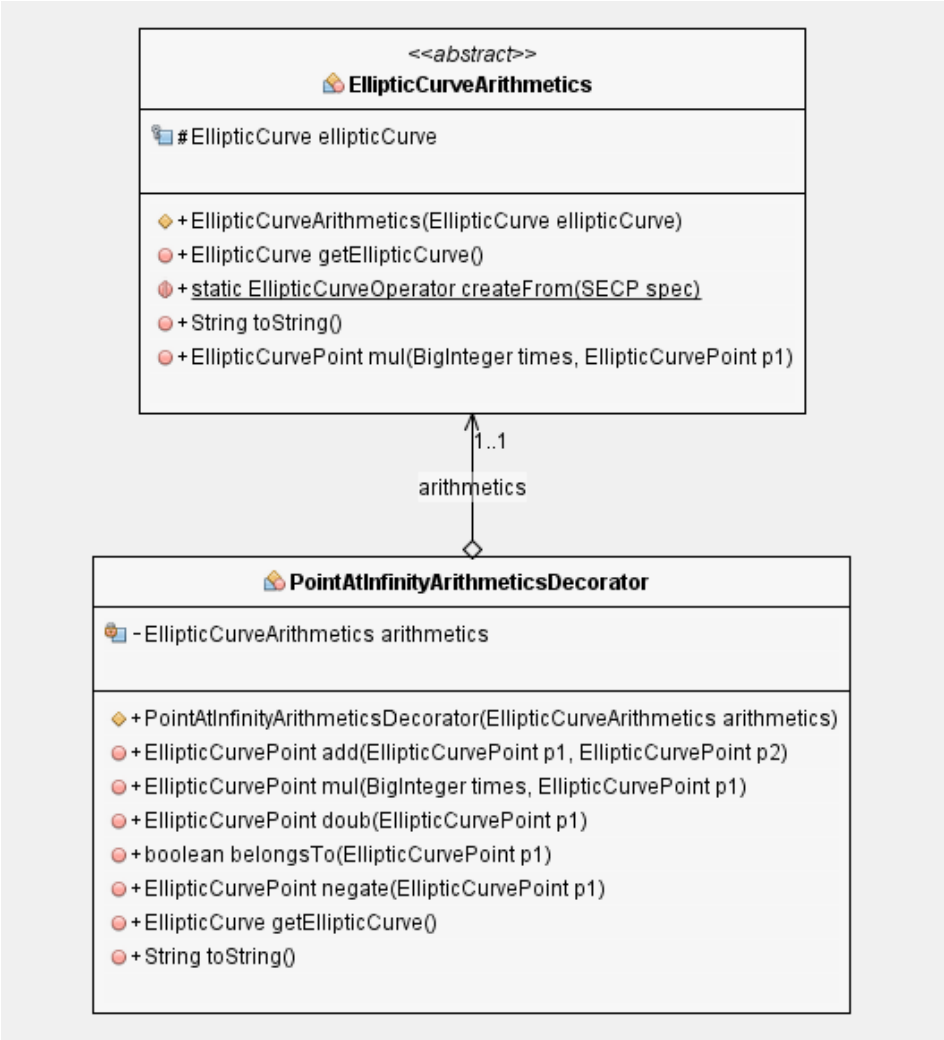


Рис. 9. Обробка крайніх випадків, пов’язаних з операціями з точкою на нескінченності з використанням шаблону «Декоратор»

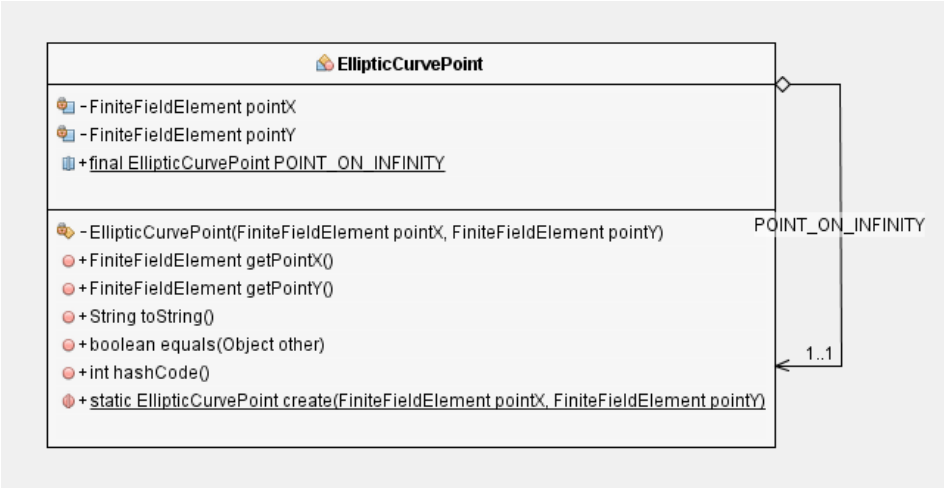


Рис. 10. Визначення точки на нескінченності за допомогою статичного поля

Визначення стандартних параметрів еліптичної кривої за допомогою перечислення його використання в статичному фабричному методі

За наявності стандарту параметрів еліптичної кривої, що забезпечують найкращу криптостійкість, а саме стандарту SECP, в процесі розробки програмного комплексу постала необхідність їх використання в тілі модуля. З огляду на те, що кількість параметрів еліптичної кривої є однаковою незалежно від типу скінченного поля, над яким вона визначена, можливість використання перечислення стало найбільш раціональним способом їх об’явлення в тілі тексту програми.

Окрім цього, з використанням даного підходу, ефективним способом створення об’єктів еліптичної кривої є використання статичного шаблонного методу. Даний підхід, на відміну від використання конструктора, дозволяє повертати класи-нащадки, тобто класи, що залежать від об’єктів, обраних в залежності від типу скінченного поля, над яким визначається дана крива. Графічно даний підхід представлений на Рис. 11 та Рис. 12 за допомогою діаграми класів мови моделювання UML.

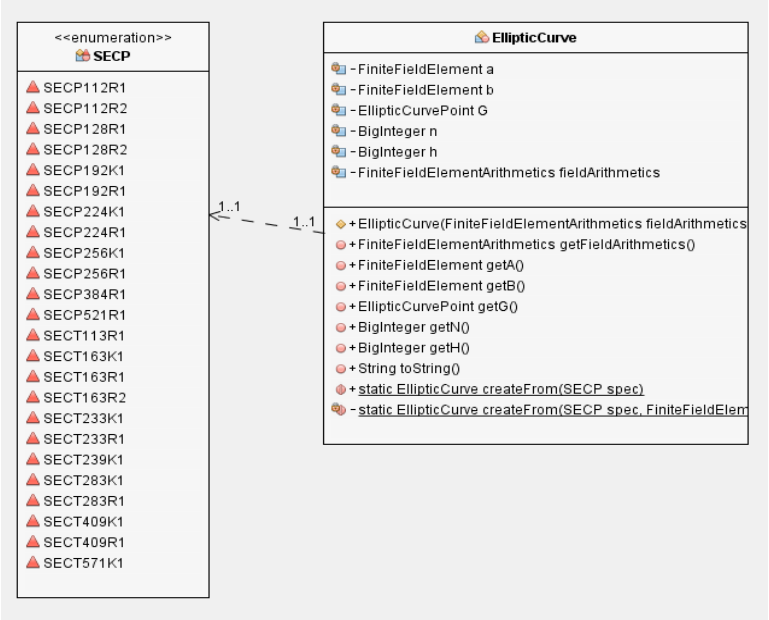


Рис. 11. Використання стандартних кривих SECP

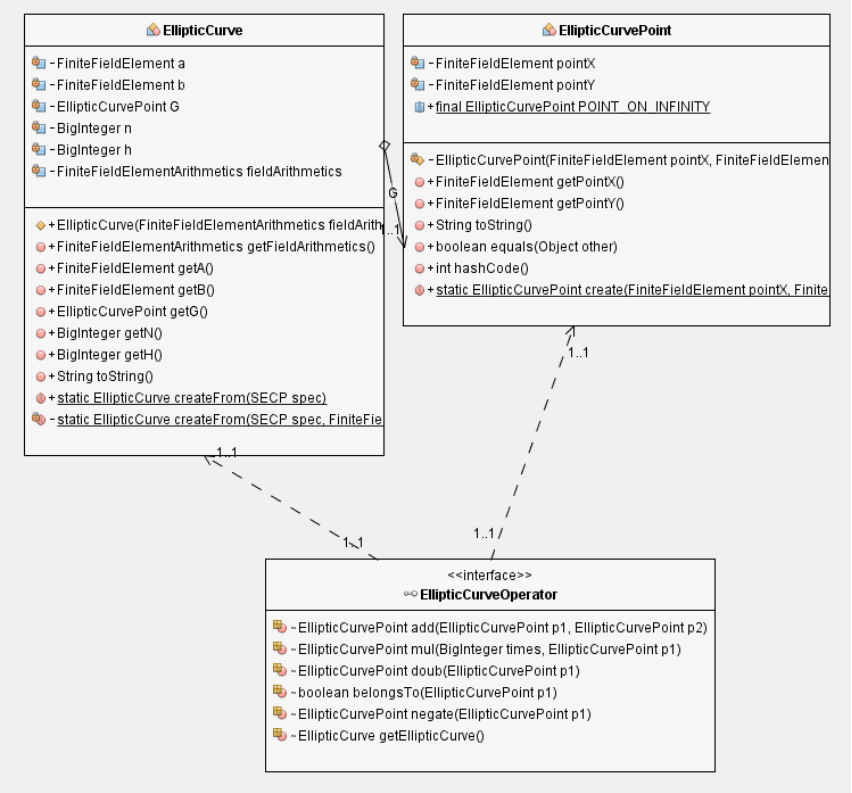


Рис. 12. Використання статичного шаблонного методу, що приймає стандарт SECP

4.5. Модуль електронно-цифрового підпису

Враховуючи наявність модуля виконання операцій над елементами скінченного поля та модуля виконання операцій над точками еліптичної кривої, модуль електронно-цифрового підпису найбільш високорівневим інтерфейсом доступу до розробленого програмного комплексу та його головною частиною. Публічний інтерфейс даного модуля представлений абстракцією, що презентує виконання підпису та перевірки ЕЦП та абстракцією пари ключів криптографічної схеми ECDSA.

Основний інтерес в даному модулі представляє спосіб організації класу, що реалізує абстракцію електронно-цифрового підпису, що дозволяв би інкапсулювати алгоритм багатократного скалярного множення точки еліптичної кривої на число. Теорія об’єктно-орієнтованого проектування

містить класичний підхід до вирішення такого типу проблем, а саме – шаблон проектування під назвою «Стратегія»[35].

Даний шаблон, з огляду на принцип об’єктно-орієнтованого проектування, що проголошує пріоритет композиції над наслідуванням, дозволяє динамічно замінювати конкретні реалізації алгоритму, не змінюючи при цьому тіло класу, у якому викликається даний алгоритм. Конкретно в даному програмному комплексі, були реалізовані різноманітні стратегії методів багатократного скалярного множення точки еліптичної кривої на число, розглянутих в межах нашого дослідження, а саме – методи SMPM, JSF, NAF, DBNS, що були детально описані в розділі 3.

Графічно даний підхід представлений на Рис. 13 за допомогою діаграми класів мови моделювання UML.

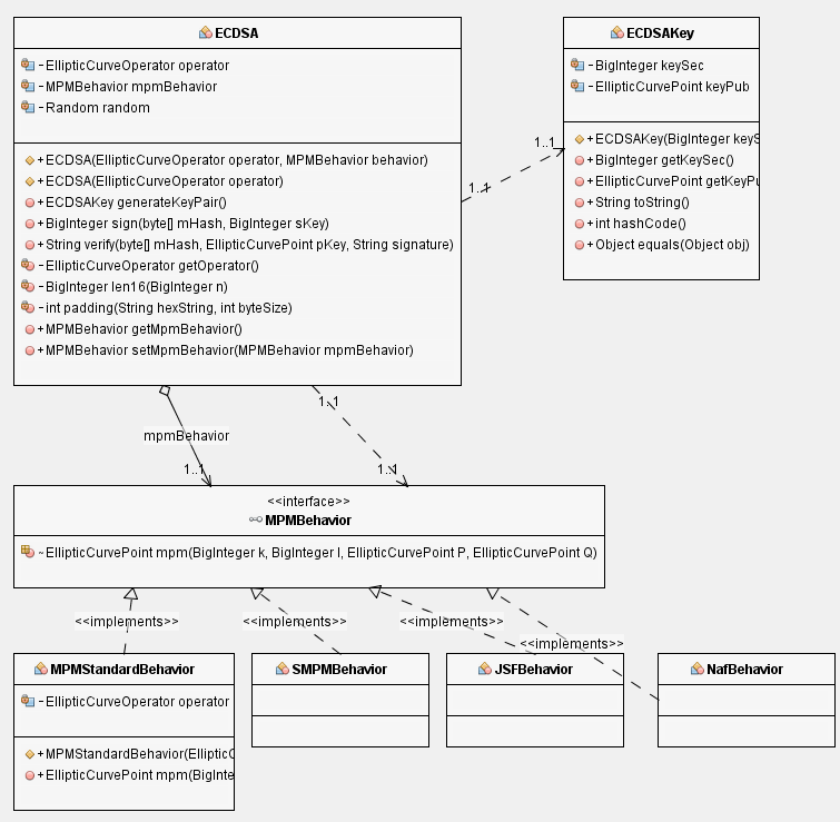


Рис. 13. Використання шаблону «Стратегія» в модулі електронно-цифрового підпису

4.6. Висновки до розділу 4

В даному розділі були розглянуті підходи, що використовувались на стадії проектування програмного комплексу для дослідження та оптимізації методу багатократного скалярного множення точки еліптичної кривої на число.

Дані підходи, а саме використання шаблону «Декоратор» для інкапсуляції обробки виключних ситуацій, пов’язаних з колізіями точки на нескінченності, шаблону «Фабричний метод» для реалізації неявнового виклику на створення непублічних субкласів, шаблону «Стратегія» для інкапсуляції різноманітних методів оптимізації багатократного скалярного множення точки еліптичної кривої на число тощо, дозволили виконати архітектуру програмного комплексу з дотриманням кращих стандартів об’єктно-орієнтованого проектування, а саме принципів SOLID та інших.

5. БІЗНЕС-ПРОЕКТ ВПРОВАДЖЕННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО КОМПЛЕКСУ

В даному розділі описується варіант використання розробленого покращеного методу багатократного скалярного множення точок еліптичної кривої на число в якості стартапу. Виконується детальний аналіз рішень, присутніх на ринку, окреслюються їхні переваги та недоліки, виконується порівняння гіпотетичного розробленого програмного комплексу електронного документообігу з існуючими, окреслюється бізнес-стратегія просування даного комплексу на ринку.

5.1. Аналіз потреб ринку електронно-цифрового підпису

З бурхливим розвитком галузі електронного документообігу, питання захисту та забезпечення достовірності даних та документів, що передаються незахищеними каналами зв'язку, такими як мережа інтернет, займає чільне місце серед досліджень, що проводяться в даній галузі. Розроблений в даній роботі модифікований метод багатократного скалярного множення точок еліптичної кривої на число має особливу цінність в контексті зменшення часу на перевірку електронно-цифрового підпису, що в свою чергу впливає на загальну швидкодію алгоритму.

Однією з галузей застосування методів електронно-цифрового підпису, що має практичний інтерес та потенціал до адаптації запропонованих в межах даної роботи розробок є сфера державного електронного документообігу.

Основна проблема існуючого стану речей в даній сфері полягає в низькому ступені проникнення алгоритмів, заснованих на еліптичній криптографії в сфери державного управління. На даний час, державні стандарти використання еліптичної криптографії хоч і існують, але поки що не набули широкого використання на рівні державних структур, адже основні операції електронно-цифрового підпису, як наприклад, використання

ЕЦП в регуляції діяльності фізичних осіб підприємців Державною Фіскальною Службою України засноване на використанні алгоритму RSA, що має значно нижчі показники швидкодії у порівнянні з ECDSA, що були детально описані в попередніх розділах даної роботи. Виходячи з вищеописаної інформації, можна виділити ряд конкретних недоліків притаманних нинішній реалізації ЕЦП ДФС України.

По-перше це низька криптостійкість електронно-цифрового підпису з визначеною довжиною в порівнянні з підписом аналогічного розміру з використанням алгоритму ECDSA. Це означає що підпис виконаний з використанням принципів еліптичної криптографії мав би набагато вищу криптостійкість.

По-друге, нижча швидкодія використовуваного алгоритму RSA в порівнянні із запропонованим модифікованим ECDSA означає, що за певний виділений проміжок часу, укладник документів може виконати меншу кількість операцій, що призводить до уповільнення процесу загалом.

По-третє, навіть припускаючи використання ECDSA з існуючими методами багатократного скалярного множення точок еліптичної кривої на число, необхідний великий об'єм пам'яті для зберігання масиву передобчислених значень [36]. Дана проблема була детально описана в розділі 3. Виходячи з цього, існує нагальна потреба скорочення об'ємів апаратних ресурсів, що використовуються в процесі електронно-цифрового підпису.

5.2. Зацікавлені сторони

Очевидно, що в процесі електронного документообігу в сфері державної регуляції підприємницької діяльності беруть участь декілька сторін, а саме: керівники та представники компанії, що виконує підприємницьку діяльність, представники державного податкового

управління, представники юридичних компаній, що виконують супровід підприємницької діяльності приватних осіб тощо.

Представники бізнесу, зі своєї сторони, зацікавлені у збільшенні захисту інформації, що підписується електронно-цифровим підписом, підвищенні швидкодії роботи даних алгоритмів, а саме зменшенні часу на підпис та перевірку ЕЦП.

Представники юридичних фірм зацікавлені в автоматизації та оптимізації документообігу з представниками бізнесу не менше ніж самі підприємці, адже покращення процесу елетронного документообігу неодмінно призводить до оптимізації загального бізнес-процесу та в кінцевому підсумку до зменшення побічних витрат та збільшення кінцевого прибутку.

Представники державного сектору зацікавлені в кінцевому результаті, а саме – збільшенні надходжень від підприємницької діяльності до держбютжету України. Дана мета може бути досягнута за рахунок оптимізації процесу обміну документами між сторонами процесу.

Описані зацікавлені сторони узагальнено в матрицю зацікавлених осіб, що наведена в Табл. *4*.

Табл. 4. Матриця зацікавлених осіб

Група зацікавлених осіб	Інтереси зацікавленої особи	Вплив	Стратегії приваблення
Власники або керуючі органи підприємств	Підвищення ефективності документообігу	Великий	Участь у державних тендерах. Рекламні матеріали. Презентації продукту, демонстрації. Участь в тематичних заходах. Забезпечення технічної підтримки.
Робітники підприємств	Підвищення швидкості роботи	Середній	
Юристи	Зменшення часу отримання результатів при попередній оцінці документів на відповідність критеріям	Середній	
Державна	Збільшення кількості	Малий	

фіскальна служба	податків від підприємств. Забезпечення виконання державних норм.		

5.3. Основні характеристики рішення

Поставлену проблему може вирішити спеціальний програмний продукт, що реалізує описаний метод багатократного скалярного множення точки еліптичної кривої на число. Завдяки описаному в даній роботі алгоритму, процес електронно-цифрового підпису вдалось вирішити вищеописані проблеми існуючих підходів, а саме збільшити швидкодію алгоритму та зменшити апаратні витрати, використовувані в процесі підпису.

Основними клієнтами є державні регулюючі служби, представники всіх видів підприємницької діяльності, юридичні компанії, заклади освіти, міністерства тощо. Саме тому програмне рішення повинно бути представлене у вигляді компактного кросплатформенного додатку, що може бути встановленим на ПК. Завдяки кросплатформенності забезпечується гнучкість рішення, наприклад, великі організації зможуть легко інтегрувати сервіс у свої бізнес процеси. Запропонований алгоритм гарно кладеться в основу рішення, що легко масштабується, що дозволяє будувати велику і розподілену систему.

5.4. Конкурентні переваги рішення

Конкурентів даного програмного рішення на ринку досить багато. Умовно їх можливо розділити на дві великі групи:

– спеціалізовані програмні комплекси електронного документообігу;
– сайти електронно-цифрового підпису документів.

Веб-сервіси у вигляді сайтів, частіше всього, розраховані на індивідуальне використання та мають обмеження у вигляді загальної черги на обробку та загальний допустимий ліміт кількості файлів. Очевидно, що

такі сервіси можна використовувати лише за наявності підключення до мережі Інтернет. Дане обмеження ставить під серйозне питання доцільність використання подібних сервісів.

Спеціалізовані програмні комплекси є дуже залежними від обчислювальних потужностей машини, на якій вони використовуються. Зазвичай, подібні комплекси не дають змогу управляти методами криптографії, що використовуються в процесі електронно-цифрового підпису. Дане обмеження і є найбільшою проблемою даних комплексів, адже метод багатократного скалярного множення точок еліптичної кривої на число є оптимізацією алгоритму ECDSA, що наряду з іншими оптимізаціями має свої переваги та недоліки. Саме за рахунок налаштування оптимальних параметрів виконання електронно-цифрового підпису і досягається максимально вигідна для користувача конфігурація. Дана особливість і є головною конкурентною перевагою пропонованого комплексу.

Отже, конкурентними перевагами програмного продукту, що пропонується є:

- висока швидкість виконання операції електронно-цифрового підпису;
- відсутність залежності вартості від об’єму документів, що підписуються;
- можливість легкого інтегрування сервісу у існуючі програмні комплекси;
- можливість конфігурації методів електронно-цифрового підпису, що використовуються;
- політика відкритого вихідного коду, що заохочує до користувацьких розширень;
- декілька варіантів роботи з сервісом для різних категорій клієнтів – корпоративних і індивідуальних.

5.5. Сегменти ринку

Як було зазначено вище, потенційних клієнтів можна умовно розділити на дві основні групи– індивідуальні та корпоративні клієнти[37].

Очевидно, що дані групи мають свої індивідуальні особливості та вимоги. Корпоративні клієнти, а саме компанії (державні та приватні) а також державні структури чи юридичні фірми характеризується великими об’ємами документообігу протягом тривалих часових проміжків, наявністю власних програмних комплексів та баз документів. Отже, для таких клієнтів доцільно пропонувати підписки, що характеризуються часовим інтервалом (наприклад, місячну, квартальну, річну т.д.) Також, таким клієнтам можна пропонувати послуги інтеграції сервісу до існуючих комплексів.

Для індивідуальних клієнтів є характерною низька періодичність використання комплексу, що означає те, що використання буде як правило розірване у часі, обмежене у кількості та матиме непостійну періодичність. Виходячи з цього, було прийняте рішення застосувати безкоштовну модель підписки для індивідуальних користувачів, а саме так звану модель freemium. Тобто, у випадку приватного використання, дозволена буде лімітована кількість електронно-цифрових підписів на місяць, а при перевищенні даного показника – можливість платного підвищення ліміту, або переходу на корпоративний план.

5.6. Ціннісна пропозиція

Ціннісні пропозиції – це причина, чому клієнти віддають перевагу одній компанії перед іншою[38]. Вони вирішують проблеми клієнтів або задовольняють їх потреби. Кожна ціннісна пропозиція представляє певну сукупність товарів та / або послуг, які відповідають запитам певного споживчого сегмента. Іншими словами, ціннісна пропозиція - це сукупність переваг, які компанія готова запропонувати споживачу.

Одні ціннісні пропозиції можуть бути інноваційними, тобто новими або революційними. Інші - подібні тим, що вже існують на ринку, але з деякими відмінностями, з якими-небудь новими характеристиками. Ціннісна пропозиція створює переваги для конкретного споживача сегмента за рахунок певної комбінації елементів, що відповідають вимогам цього сегмента. Переваги можуть бути кількісними (такими як ціна, швидкість обслуговування) або якісними (наприклад, дизайн, позитивні емоції клієнта). Для кожного сегменту споживачів зазвичай пропонується виділяти окрему ціннісну пропозицію.

Ціннісною пропозицією для сегменту корпоративних клієнтів є швидка, точна система, що легко, за необхідності, інтегрується в існуючі рішення та дозволяє обробляти одночасно певну кількість документів.

Для сегменту індивідуальних клієнтів ціннісною пропозицією є наявність безкоштовного та платного плану використання системи, що дозволяє більш гнучко підлаштовувати використання сервісу під конкретні потреби клієнта.

5.7. Доходи та витрати

Для визначення сумарного або валового доходу компанії необхідно визначити суму доходів від продажів товарів та послуг, якими торгує дана компанія та доходів від реалізації супутніх послуг[39]. На основі інформації, поданої у попередніх розділах, був сформований список товарів і послуг, для двох груп клієнтів – корпоративних та приватних. Для сегменту корпоративних клієнтів пропонується продавати наступні товари та надавати наступні послуги:

– короткотивалі, довготривалі та постійні підписки;
– розширена технічна підтримка.

Під короткотивалими, довготривалими та постійними підписками мається на увазі те, що клієнт може обрати один строк підписки від одного місяця до двох років, а також придбати даний продукт на необмежену

кількість часу, тобто назавжди. Очевидно, що вартість підписки напряму залежить від обраного терміну підписки. Розширення технічна підтримка пропонується як послуга на термін дії підписки і включає в себе гарантоване оброблення запитів в швидкі терміни в робочі та вихідні дні.

Для сегменту індивідуальних клієнтів окрім базової безкоштовної версії пропонується продавати наступні товари:

– підписку з обмеженням на кількість документів для аналізу;
– підписку з короткотривалим часовим обмеженням;
– підписку на одноденне використання.

Дохід по місяцям протягом першого року наведено в Табл. 5. Всі суми наведено в гривнях.

Табл. 5. Розрахунок доходів по місяцям першого року

	Довготривалі підписки	Розширена технічна підтримка	Підписка з обмеженням на кількість документів	Короткотривалі підписки	Всього
1	38928	3378	26458	22267	91031
2	23433	7533	24081	41972	97019
3	22404	9050	41114	28832	101400
4	39854	10525	39244	30071	119694
5	45913	9099	39587	31099	125698
6	22310	11786	43754	34711	112560
7	48782	9407	34885	35290	128365
8	45468	8292	21284	20351	95396
9	40418	14543	36038	40306	131304
10	41070	10757	31901	42484	126212
11	24557	19819	33940	46670	124986
12	34426	6132	42021	43757	126336
Сумарно за рік:					1379999

Загальні витрати складаються з таких статей як:

- витрати організацію діяльності компанії (покупка і встановлення необхідного програмного забезпечення, організація робочого простору, покупка та налаштування апаратних засобів (сервери, персональні комп'ютери, інші пристрої);
- витрати на оплату праці (заробітна плата та інші виплати працівникам);
- комунальні витрати (абонплата);
- адміністративні витрати;
- витрати на рекламу та дослідження ринку.

Витрати по місяцям протягом першого року наведено в Табл. 6. Всі суми наведено в гривнях.

Табл. 6. Розрахунок доходів по місяцям першого року

	Технічна підтримка	Оплата праці	Комунальні та адміністративні	Реклама	Всього
1	5380	50381	10000	10308	76069
2	5651	50325	10000	10217	76193
3	5142	51152	10000	10219	76513
4	5641	51791	10000	10488	77920
5	5318	51545	10000	11349	78212
6	7887	50590	10000	12400	80877
7	6918	50130	10000	11416	78464
8	8559	50506	10000	12857	81922
9	5231	51977	10000	13093	80301
10	5711	53896	10000	10180	79787
11	7194	50088	10000	13717	80999
12	10070	51030	10000	15104	86204
				Сумарно за рік:	953461

5.8. Бізнес-модель

Узагальнимо вище наведену інформацію в виглядів блоків, що складають так звану канву бізнес-моделі котра зображена в Табл. 7.

Табл. 7. Канва бізнес моделі

<table>
<tr><td rowspan="3">Проблема

Низька швидкість обчислення електронно-цифрового підпису.

Велика обчислювальна складність алгоритмів.</td><td>Рішення

Розроблений програмний комплекс

Можливість налаштування параметрів алгоритмів</td><td rowspan="3">Унікальна ціннісна пропозиція

Гнучкі підписки для корпоративних і індивідуальних клієнтів. Наявність безкоштовної приватної підписки.

Можливість легкої інтеграції в існуючі системи і індивідуального налаштування</td><td rowspan="2">Прихована перевага

Розроблений метод багатократного скалярного множення точок еліптичної кривої на число для ECDSA</td><td rowspan="3">Споживачі

Ранні клієнти: приватні підприємці, окремі індивідуальні клієнти.

Великі підприємства, даржавний сектор, індивідуальні клієнти.</td></tr>
<tr><td>Ключові метрики

Кількість проданих підписок різних типів.

Кількість запитів в службу технічної підтримки..</td></tr>
<tr><td>Канали

Тематичні ресурси та спільноти.

Профільні видання.

Прямі контакти для продажів</td></tr>
</table>

Структура витрат	**Потоки доходів**
Витрати на розроблення, підтримку та вдосконалення продуктів.	Продаж різнотривалих підписок для корпоративних клієнтів
Оплата праці.	Продаж розширеної технічної підтримки.
Комунальні послуги.	Продаж підписки з обмеженням на кількість документів для індивідуальних клієнтів.
Адміністративні витрати.	Продаж підписки з короткотривалим часовим обмеженням та обмеження на один документдля індивідуальних клієнтів
Витрати на рекламу	

5.9. Висновки до розділу 5

В даному розділі розглянуто питання практичного застосування розробленого методу багатократного скалярного множення точок еліптичної кривої на число.

Отримана бізнес модель показує, що дана розробка є потенційно привабливою в контексті приватного використання компаніями, що мають великі обсяги електронного документообігу та те, що алгоритм ECDSA, для якого власне і стосується дана розробка може бути використаний в якості альтернативи прийнятим до використання на сьогоднішний день стандартам в державних органах України.

Розраховані доходи та витрати протягом першого року роботи. Знайдений маржинальний прибуток за цей період.

ВИСНОВКИ

В даній роботі було виконано дослідження підходів та методів шифрування/дешифрування з використанням еліптичних кривих, зокрема алгоритми електронно-цифрового підпису.

В основному, в межах даної роботи були розглянуті еліптичні криві, параметри яких визначені над скінченними полями з простою характеристикою($GF(p)$), та бінарним розширенням $GF(2^m)$.

Був виконаний огляд математичних основ побудови еліптичних кривих, опис основних алгебраїчних структур, над якими визначаються параметри точок еліптичної кривої таких як кільце, група, поле, скінченне поле та інші.

Умовно, операції, що виконуються в рамках застосування алгоритмів шифрування з використанням еліптичної криптографії можна розділити на 2 основних класи: операції нижнього та верхнього рівня, де операції нижнього рівня – це арифметичні операції над елементами скінченних полів та арифметичні операції над точками еліптичної кривої, а операції верхнього рівня, тобто ті, що використовують операції нижнього рівня – знаходження оберненого елемента та скалярне множення точки еліптичної кривої на число.

Найбільш обчислювально-витратною операцією у еліптичній криптографії є багатократне скалярне множення точки еліптичної кривої на число. Було проведено дослідження різноманітних методів багатократного скалярного множення точок еліптичної кривої на число таких як: метод SMPM (simultaneous multiple point multiplication) та його модифікації, що використовують подання множника у вигляді NAF та JSF та методів з поданням множника у вигляді DBNS. Було запропоновано новий метод багатократного скалярного множення точкок еліптичної кривої на число, що

показує кращу швидкодію в порівнянні з існуючими методами для еліптичних кривих з параметрами, визначеними над полем $GF(2^m)$.

Був розроблений програмний комплекс електронно-цифрового підпису з відкритим вихідним кодом з розширюваною програмною архітектурою, що дозволяє динамічно додавати та використовувати різні програмні реалізації методу багатократного скалярного множення точок еліптичної кривої на число. Даний комплекс має унікальну архітектуру, що задовольняє основним законам об'єктно-орієнтованого проектування SOLID.

Практична цінність отриманих в роботі результатів полягає в тому, що запропоновані методи та алгоритми дозволяють швидше виконувати операцію багатократного скалярного множення точки еліптичної кривої над скінченним полем $GF(p)$ та $GF(2^m)$, що в свою чергу дозволяє зменшити час перевірки електронно-цифрового підпису з використанням еліптичних кривих.

Розроблене математичне та програмне забезпечення дозволяє як виконувати електронно-цифровий підпис інформаційних повідомлень довільної довжини так і проводити дослідження та порівняльний аналіз методів багатократного скалярного множення точки еліптичної кривої над скінченним полем $GF(p)$ та $GF(2^m)$.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Darrel Hankerson, Alfred Menezes, Scott Vanstone Guide to Elliptic Curve Cryptography / Darrel Hankerson // 2004,Springer – Verlag New York Inc. – 205 p
2. Мао, Венбао. Современная криптография: теория и практика. : Пер. с англ. — М. : Издательский дом «Вильямс», 2005. — 768 с.
3. N.Koblitz, Elliptic Curve Cryptosystems [Text]/ Neal Koblitz// Mathematics of Computation – Volume 48 – Number 177 – January 1987 – pp. 203-209
4. Cetin Kaya Coc, Cryptographic Engineering/ Cetin Kaya Coc// Springer Science+Business Media, LLC 2009 – 171 p
5. Василенко О. Н., Теоретико-числовые алгоритмы в криптографии/ Василенко О. Н.// М.: МЦНМО, 2003. – 178 с.
6. J. Solinas. Low-weight binary representations for pairs of integers/ J. Solinas// National Security Agency, USA 2001.
7. V. S. Dimitrov, L. Imbert, and P. Mishra. Fast elliptic curve point multiplication using double-base chains. 2005.
8. Сидельников, В.М. Криптография и теория кодирования [Текст] / В.М. Сидельников. — М. : ФИЗМАТЛИТ, 2008. — 324 с.
9. Черемушкин, А.В. Лекции по арифметическим алгоритмам в криптгографии. — М. : МЦНМО, 2002. — 104 с.
10. Панасенко, С.П. Алгоритмы Шифрования. Специальный Справочник [Текст] / С.П. Панасенко. — М. : Академический Проект, 2006. — 529 с.
11. Василенко, О.Н. Теоретико-числовые алгоритмы в криптографии [Текст] / О.Н. Василенко. — М. : МЦНМО, 2003. — 328с.
12. Joye, M. Cryptographic Hardware and Embedded Systems — CHES 2004 [Text] / Marc Joye. — Cambridge, USA : Springer, 2004. — 471 p.
13. Фергюсон. Практическая криптография: Пер. с англ. [Текст] / Нильс, Шнайер, Брюс. — М. : «Вильямс», 2005. — 424 с.

14. Князькіна, О.С. Спосіб генерування випадкового простого числа заданої довжини [Текст] / М.В. Онай, О.С. Князькіна // Праці міжнародн. конф «СНКПМІ-2012». — 2012. — С. 52-54.
15. Ярочкин, В.И. Информационная безопасность: Учебник для вузов [Текст] / В.И. Ярочкин. — М. : Академический Проект, 2008. — 544 с.
16. W. Diffie and M. Hellman New directions in cryptography. Information Theory, IEEE Transactions on elliptic Curves, 22 : 644-654, 1976.
17. N. Koblitz Elliptic curve cryptosystems. Mathematics of Computation, 48 : 203-209, 1987.
18. V.S. Miller Use of Elliptic Curves in Cryptography, page 417, 1986.
19. D.Hankerson, A.Menezes, S.Vanstone Guide to Elliptic Curve Cryptography, page 311, 2004.
20. Jithra Adikari, Vasil S.Dimitrov and Pradeep Mishra Fast Multiple Point Multiplication on Elliptic Curves over Prime and Binary Fields using the Double-Base Number System.
21. Mathiew Ciet and Marc Joye Trading Inversions for Multiplications in Elliptic Curve Cryptography
22. Vasyl Dimitrov, Laurent Imbert and Pradeep Kumar Mishra Efficient and Secure Elliptic Curve Point Multiplication Using Double-Base Chains
23. Duc-Phong Le, Bin P.Nguyen Fast Point Quadrupling on Elliptic Curves
24. Gordon H. Bredley. Algorithm and bound for the greatest common divisor of n integers. Communications of the ACM, 13(7):433-436, 1970.
25. Tudor Jebelean. A generalization of the binary GCD algorithm. In M. Bronstein, editor, 1993 ACM International Symposium on Symbolic and Algebraic Computation, pages 111—116, Kiev, Ukraine, 1993. ACM Press.
26. D. E. Knuth. The Art of Computer Programming: Seminumerical Algorithms, volume 2. Addison-Wesley, Reading, Mass., 2nd edition, 1981.
27. D. H. Lehmer. Euclid's algorithm for large numbers. Amer. Math. Monthly, 45:227-233, 1938.

28. G. Norton. Extending the binary GCD algorithm. In J. Calmet, editor, Proceedings of the 3rd International Conference on Applied Algebra, pages 363-372. Springer-Verlag, 1985. LNCS 229.
29. J. M. Pollard. Theorems on factorization and primality testing. Proc. Camb. Phil. Soc., 76:521-528, 1974.
30. Jonathan P. Sorenson. Two fast GCD algorithms. Journal of Algorithms, 16:110-144, 1994.
31. Jonathan P. Sorenson. An analysis of Lehmer's Euclidean GCD algorithm. In A. H. M. Levelt, editor, 1995 ACM International Symposium on Symbolic and Algebraic Computation, pages 254-258, Montreal, Canada, July 1995.
32. Князькіна, О.С. Алгоритми виконання низькорівневих операцій на еліптичній кривій [Текст] / М.В. Онай, О.С. Князькіна // Міжнародна науково-практична конференція «Проблеми інформатики та комп'ютерної техніки». Праці конференції. — Чернівці: Видавничий дім "Родовід", 2014. — С. 87 - 89.
33. Дичка А.І. Модифікація методу багатократного скалярного множення точок еліптичної кривої на число [Текст] / М.В. Онай, Дичка А.І. // Прикладна математика та комп'ютинг. ПМК, 2014 : десята наук. конф. магістрантів та аспірантів, Київ, 16-18 квітня 2014 р. : зб. тез доп. / [ред кол.: Дичка І.А. та ін.] . – К. : Просвіта, 2018. – С. 235-240.
34. Дичка А.І. Дослідження швидкодії методів вирішення задачі дискретного логарифма на еліптичних кривих [Текст] / М.В. Онай, О.С. Дичка А.І. // Праці міжнародної науково-практичної конференції «Інформаційні технології та комп'ютерна інженерія». — 2017. — С. 172-176.
35. Knyazkina, O.S. Finding of reverse element in the multiplicative group of the ring residues Z/mZ [Text] / M.V. Onay, O.S. Knyazkina // Праці міжнародн. конф. «Імені академіка М. Кравчука». Том 2. — К. : НТУУ «КПІ». — 2012. — С. 23-24.

36. Value Proposition Canvas Template [Електронний ресурс]. – дата візиту 04.12.2017. – Режим доступу до ресурсу: https://goo.gl/MbhgG4
37. Шаблон — GN1403: Моделирование бизнес-процессов — Бизнес-информатика [Електронний ресурс]. – дата візиту 04.12.2017. – Режим доступу до ресурсу: https://goo.gl/43VDek
38. Маржинальная прибыль. Пример анализа. Формула расчета [Електронний ресурс]. – дата візиту 04.12.2017. – Режим доступу до ресурсу: https://goo.gl/kyFxfH
39. Как правильно заполнять Lean Canvas | Wiki.Rademade.com [Електронний ресурс]. – дата візиту 04.12.2017. – Режим доступу до ресурсу: https://goo.gl/Gkb4Uq

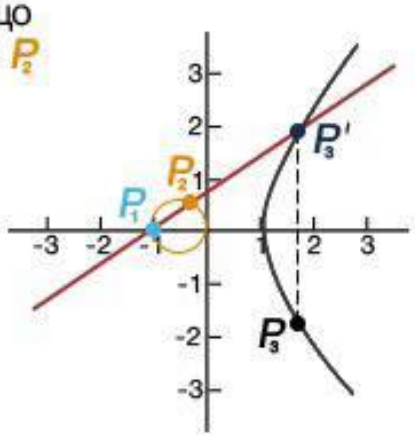
ДОДАТКИ

ДОДАТОК 1

Копії графічних матеріалів

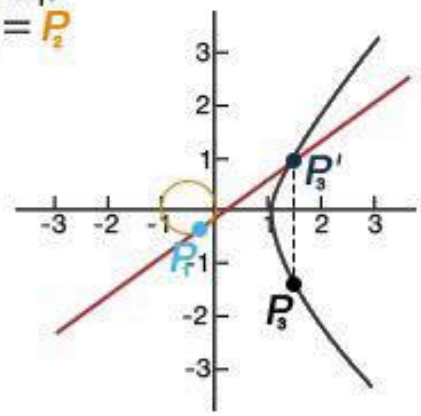
Додавання точок еліптичної кривої: $P_1 + P_2 = P_3$

Якщо $P_1 \neq P_2$



1. Провести пряму між P_1 та P_2
2. Знайти точку перетину прямої з еліптичною кривою
3. Відобразити відносно осі OX

Якщо $P_1 = P_2$



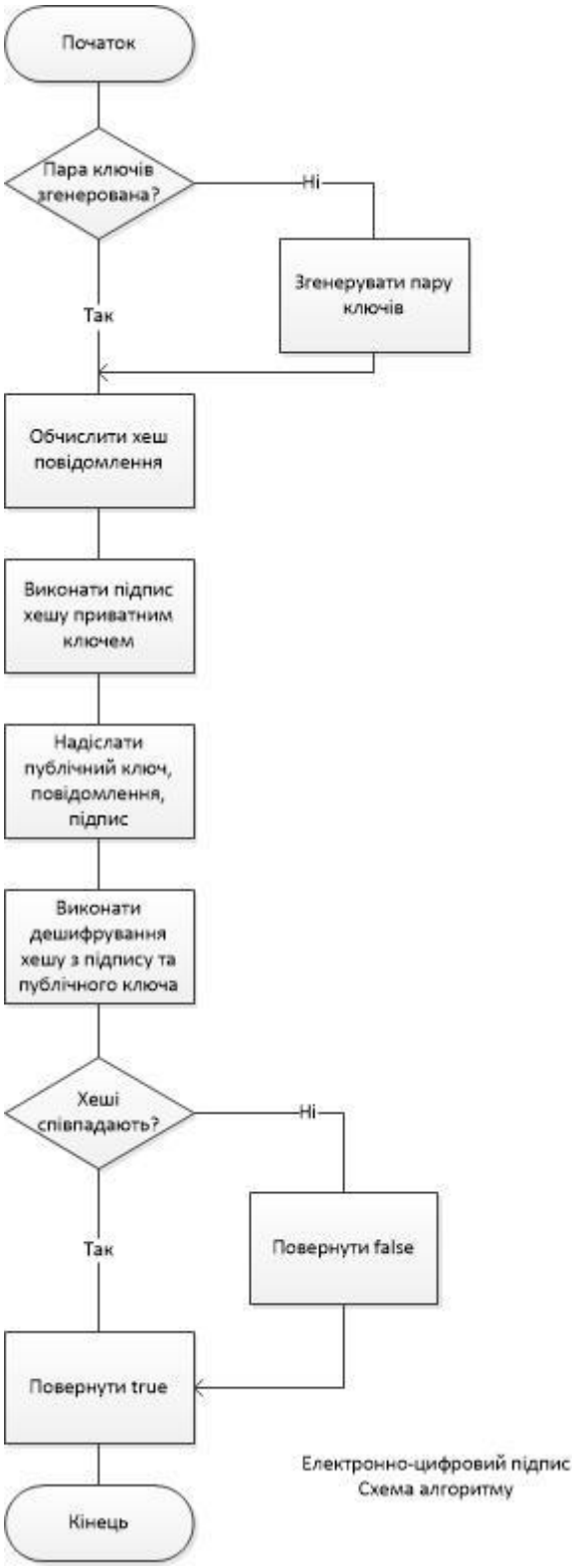
1. Знайти дотичну до кривої у точці P_1
2. Знайти точку перетину прямої з еліптичною кривою
3. Відобразити відносно осі OX

Модифікація методу багатократного скалярного множення точок еліптичної кривої в скінченних полях
Дичка Андрій Іванович, КП-61м

СХЕМА ОБМІНУ СИМЕТРИЧНИМ КЛЮЧЕМ З ВИКОРИСТАННЯМ АСИМЕТРИЧНОГО ШИФРУВАННЯ

Модифікація методу багатократного
скалярного множення точок еліптичної
кривої в скінченних полях
Дичка Андрій Іванович, КП-61м

БЛОК-СХЕМА АЛГОРИТМУ ЕЛЕКТРОННО-ЦИФРОВОГО ПІДПИСУ З ВИКОРИСТАННЯМ ЕК



Електронно-цифровий підпис
Схема алгоритму

Модифікація методу багатократного
скалярного множення точок еліптичної
кривої в скінченних полях
Дичка Андрій Іванович, КП-61м

БЛОК-СХЕМА РОЗРОБЛЕНОГО МЕТОДУ БАГАТОКРАТНОГО МНОЖЕННЯ ТОЧОК ЕЛІПТИЧНОЇ КРИВОЇ НА ЧИСЛО

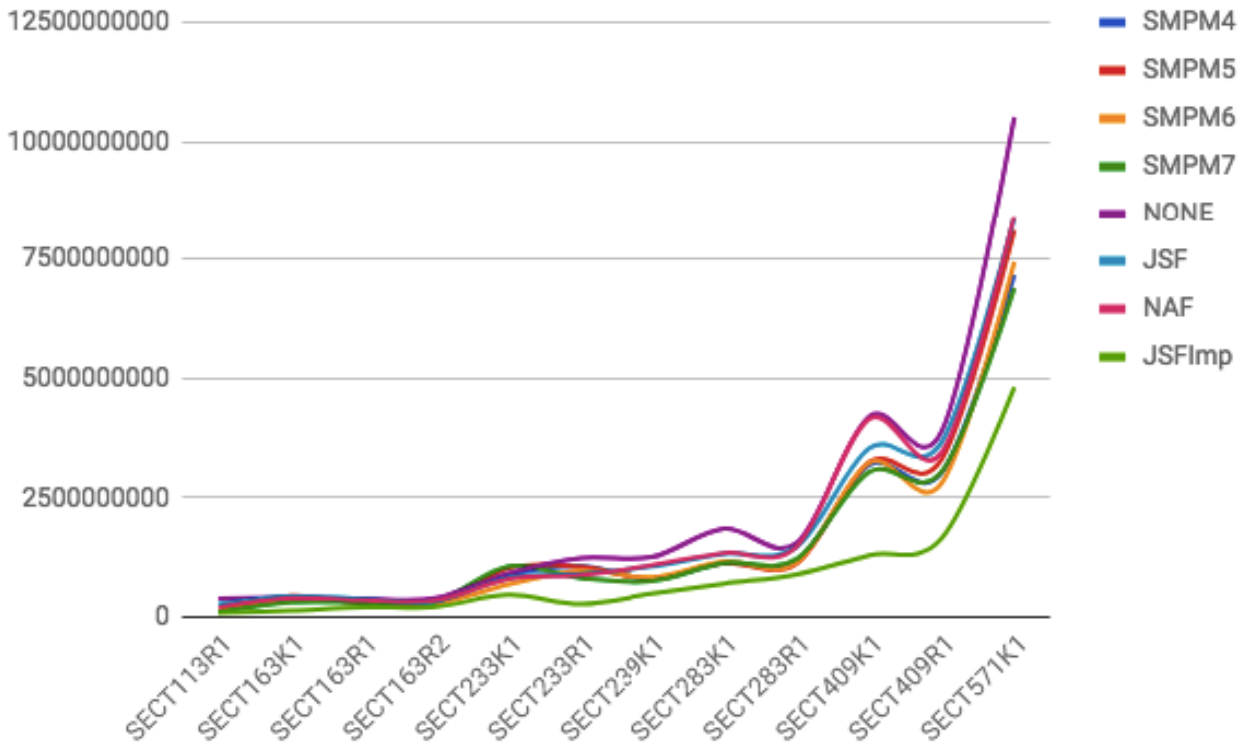
Загальна схема модифікованого методу багатократного скалярного множення точок еліптичної кривої на число

Модифікація методу багатократного
скалярного множення точок еліптичної
кривої в скінченних полях
Дичка Андрій Іванович, КП-61м

БЛОК-СХЕМА АЛГОРИТМУ ТРИВІАЛЬНОГО МНОЖЕННЯ ТОЧКИ ЕК НА ЧИСЛО

Тривіальне множення точки еліптичної кривої на число
Схема алгоритму

ПОРІВНЯННЯ ШВИДКОСТІ РОБОТИ МЕТОДІВ БАГАТОКРАТНОГО МНОЖЕННЯ ТОЧОК ЕЛІПТИЧНОЇ КРИВОЇ



Модифікація методу багатократного скалярного множення точок еліптичної кривої в скінченних полях
Дичка Андрій Іванович, КП-61м

ДОДАТОК 2

Лістинг коду

EllipticCurveArithmetics.java

```
/*
 * Copyright 2018 trident.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.trident.crypto.elliptic.arithmetics;

import com.trident.crypto.elliptic.EllipticCurve;
import com.trident.crypto.elliptic.EllipticCurveOperator;
import com.trident.crypto.elliptic.EllipticCurvePoint;
import com.trident.crypto.elliptic.nist.SECP;
import java.math.BigInteger;

/**
 *
 * defines class representing the arithmetics over provided elliptic curve, i.e.
 * point addition, multiplication, doubling, belonging test
 * @author trident
 */
public abstract class EllipticCurveArithmetics implements EllipticCurveOperator{

    //olliptic curve over which this arithmetics is defined
    protected final EllipticCurve ellipticCurve;

    public EllipticCurveArithmetics(EllipticCurve ellipticCurve) {
        this.ellipticCurve = ellipticCurve;
    }

    @Override
    public EllipticCurve getEllipticCurve() {
        return ellipticCurve;
    }

    /**
     * static factory method to create the arithmetics over the elliptic curve
     * defined by the standard specification
     *
     * should prefer this instead of constructor call
     * @param spec
     * @return
     */
    public static EllipticCurveOperator createFrom(SECP spec){
        return new PointAtInfinityArithmeticsDecorator(spec.getType()?
                new ECOverPFArithmetics(EllipticCurve.createFrom(spec)):
                new ECOverBEFArithmetics(EllipticCurve.createFrom(spec)));
    }

    /**
     * provides the human readable information about this arithmetics and its elliptic curve
     * @return
     */
    @Override
    public String toString(){
```

```
        return new StringBuilder().append("Elliptic curve arithmetics defined
over:\n").append(getEllipticCurve()).toString();
    }

    @Override
    public EllipticCurvePoint mul(BigInteger times, EllipticCurvePoint p1){
        if(times.signum()==-1) throw new RuntimeException("negative times");
        if(times.compareTo(BigInteger.ZERO) == 0)
            throw new RuntimeException("multiply to zero");
        if(times.compareTo(BigInteger.ONE) == 0)
            return p1;

        EllipticCurvePoint temp = EllipticCurvePoint.create(p1.getPointX(), p1.getPointY());
        times = times.subtract(BigInteger.ONE);
        while (times.compareTo(BigInteger.ZERO)>0){
            if (times.testBit(0)){
                if (temp.equals(p1))
                    temp = doub(temp);
                else{
                    if(temp.equals(negate(p1))) // if adding p + (-p)
                        temp = EllipticCurvePoint.POINT_ON_INFINITY;
                    else
                        temp = add(temp,p1);
                }
                times = times.subtract(BigInteger.ONE);
            }
            times = times.shiftRight(1);
            p1 = doub(p1);
        }
        return temp;
    }
}
```

PointAtInfinityArithmeticsDecorator.java

```
/*
 * Copyright 2018 trident.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.trident.crypto.elliptic.arithmetics;

import com.trident.crypto.elliptic.EllipticCurve;
import com.trident.crypto.elliptic.EllipticCurveOperator;
import com.trident.crypto.elliptic.EllipticCurvePoint;
import java.math.BigInteger;

/**
 * decorator pattern usage for edge cases check while facing with
 * points on infinity
 * @author trident
 */
public class PointAtInfinityArithmeticsDecorator implements EllipticCurveOperator{

    private final EllipticCurveArithmetics arithmetics;
```

```
    public PointAtInfinityArithmeticsDecorator(EllipticCurveArithmetics arithmetics){
        this.arithmetics = arithmetics;
    }

    @Override
    public EllipticCurvePoint add(EllipticCurvePoint p1, EllipticCurvePoint p2) {
        if(p1.equals(negate(p2))) return EllipticCurvePoint.POINT_ON_INFINITY;
        if(p1.equals(EllipticCurvePoint.POINT_ON_INFINITY)) return p2;
        if(p2.equals(EllipticCurvePoint.POINT_ON_INFINITY)) return p1;
        return arithmetics.add(p1, p2);
    }

    @Override
    public EllipticCurvePoint mul(BigInteger times, EllipticCurvePoint p1) {
        if(p1.equals(EllipticCurvePoint.POINT_ON_INFINITY)) return EllipticCurvePoint.POINT_ON_INFINITY;
        return arithmetics.mul(times, p1);
    }

    @Override
    public EllipticCurvePoint doub(EllipticCurvePoint p1) {
        if(p1.equals(EllipticCurvePoint.POINT_ON_INFINITY)) return EllipticCurvePoint.POINT_ON_INFINITY;
        return arithmetics.doub(p1);
    }

    @Override
    public boolean belongsTo(EllipticCurvePoint p1) {
        if(p1.equals(EllipticCurvePoint.POINT_ON_INFINITY)) return true;
        return arithmetics.belongsTo(p1);
    }

    @Override
    public EllipticCurvePoint negate(EllipticCurvePoint p1) {
        if(p1.equals(EllipticCurvePoint.POINT_ON_INFINITY)) return EllipticCurvePoint.POINT_ON_INFINITY;
        return arithmetics.negate(p1);
    }

    @Override
    public EllipticCurve getEllipticCurve() {
        return arithmetics.getEllipticCurve();
    }

    @Override
    public String toString(){
        return arithmetics.toString();
    }
}
```

ECDSA.java

```
package com.trident.crypto.algo;

import com.trident.crypto.algo.mpmbehavior.MPMBehavior;
import com.trident.crypto.algo.mpmbehavior.MPMStandardBehavior;
import com.trident.crypto.elliptic.EllipticCurveOperator;
import com.trident.crypto.elliptic.EllipticCurvePoint;
import java.math.BigInteger;
import java.util.Random;


/**
 * implements EDSA (elliptic curve digital signature algorithm)
 * @author trident
 */
public class ECDSA {

    /**
     * arithmetics over the elliptic curve
     * @see EllipticCurveArithmetics
     */
    private final EllipticCurveOperator operator;

    /**
     * multiple point multiplication calculation strategy
     */
    private MPMBehavior mpmBehavior;

    private final Random random;

    public ECDSA(EllipticCurveOperator operator, MPMBehavior behavior){
        this.operator = operator;
        this.random = new Random();
        this.mpmBehavior = behavior;
    }

    public ECDSA(EllipticCurveOperator operator){
        this(operator, new MPMStandardBehavior(operator));
    }

    /**
     * generate key pair i.e. secret and public
     * such that
     * secret = b (random BigInteger)
     * public  = Q = b*G (product of generator point of the elliptic curve on b)
     * @return
     */
    public ECDSAKey generateKeyPair(){
        BigInteger sKey = new BigInteger(getOperator().getEllipticCurve().getN().bitLength(), random);
        EllipticCurvePoint pKey = getOperator().mul(sKey, getOperator().getEllipticCurve().getG());
        return new ECDSAKey(sKey,pKey);
    }

    /**
     * returns the digital signature
     * @param mHash - hash of the plain message M, produced using hash function such as SHA-256
     * @param sKey - secret key of the key pair
     * @return hexadecimal string, digital signature of mHash
     */
    public String sign(byte[] mHash, BigInteger sKey){
        BigInteger n = getOperator().getEllipticCurve().getN();
        EllipticCurvePoint G = getOperator().getEllipticCurve().getG();

        BigInteger alpha = new BigInteger(mHash);
        BigInteger e = alpha.mod(n);
        if(e.equals(BigInteger.ZERO)) e = BigInteger.ONE;
```

```
    BigInteger k;
    EllipticCurvePoint C;
    BigInteger r;
    BigInteger s;
    do{
        do {
            k = new BigInteger(n.bitLength(), random);
        } while (k.compareTo(BigInteger.ZERO) == -1 || k.compareTo(n) >= 1); // k<0 || k>n
        C = getOperator().mul(k, G);
        r = C.getPointX().mod(n);
        s = r.multiply(sKey).add(k.multiply(e)).mod(n);
    } while (r.equals(BigInteger.ZERO)||s.equals(BigInteger.ZERO));

    return padding(r.toString(16),len16(n))+padding(s.toString(16),len16(n));
}

/**
 * verify that provided signature corresponds to the provided hash
 * @param mHash - hash of the plain message M, produced using hash function such as SHA-256
 * @param pKey - public key of the key pair
 * @param signature - hexadecimal string, digital signature of mHash
 * @return if the signature is verified
 */
public boolean verify(byte[] mHash, EllipticCurvePoint pKey, String signature){
    BigInteger n = getOperator().getEllipticCurve().getN();
    EllipticCurvePoint G = getOperator().getEllipticCurve().getG();

    BigInteger r = new BigInteger(signature.substring(0, len16(n)), 16);
    BigInteger s = new BigInteger(signature.substring(len16(n), 2*len16(n)), 16);

    if(r.compareTo(BigInteger.ZERO)<=0||r.compareTo(n)>=0) return false;
    if(s.compareTo(BigInteger.ZERO)<=0||s.compareTo(n)>=0) return false;

    BigInteger alpha = new BigInteger(mHash);
    BigInteger e = alpha.mod(n);
    if(e.equals(BigInteger.ZERO)) e = BigInteger.ONE;

    BigInteger v = e.modInverse(n);
    BigInteger z1 = (s.multiply(v)).mod(n);
    BigInteger z2 = n.add(r.multiply(v).negate()).mod(n);

    EllipticCurvePoint C = mpmBehavior.mpm(z1, z2, G, pKey);
    BigInteger R = C.getPointX().mod(n);

    return R.equals(r);
}

private EllipticCurveOperator getOperator() {
    return operator;
}

private int len16(BigInteger n){
    if(n.bitLength()%4==0)
        return n.bitLength()/4;
    else return n.bitLength()/4+1;
}

private String padding(String hexString, int byteSize){
    if(byteSize<hexString.length()) throw new RuntimeException("input string length is bigger than required");
    if(byteSize==hexString.length()) return hexString;

    StringBuilder sb =  new StringBuilder(hexString).reverse();
    while (sb.length()<byteSize) {
        sb.append("0");
```

```
        }
        return sb.reverse().toString();
    }

    public MPMBehavior getMpmBehavior() {
        return mpmBehavior;
    }

    public void setMpmBehavior(MPMBehavior mpmBehavior) {
        if(mpmBehavior == null) throw new RuntimeException("should not be null");
        this.mpmBehavior = mpmBehavior;
    }
}
```

EllipticCurve.java

```
/*
 * Copyright 2018 trident.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.trident.crypto.elliptic;

import com.trident.crypto.elliptic.nist.SECP;
import com.trident.crypto.field.element.BinaryExtensionFieldElement;
import com.trident.crypto.field.element.BinaryExtensionFieldElementFactory;
import com.trident.crypto.field.element.FiniteFieldElement;
import com.trident.crypto.field.element.FiniteFieldElementFactory;
import com.trident.crypto.field.operator.FiniteFieldElementArithmetics;
import java.math.BigInteger;

/**
 * @see http://www.secg.org/SEC2-Ver-1.0.pdf
 * @author trident
 */
public class EllipticCurve{

    /**
     * parameter a of the curve equation
     */
    private final FiniteFieldElement a;

    /**
     * parameter b of the curve equation
     */
    private final FiniteFieldElement b;

    /**
     * point on the curve with high order
     */
    private final EllipticCurvePoint G;

    /**
     * order of the point G
```

```
     */
    private final BigInteger n;

    /**
     * cofactor - relation between number of points of curve and order of point G
     */
    private final BigInteger h;

    /**
     * arithmetics of the finite field over which this curve is defined
     * i.e. GF(p) or GF(2^m)
     */
    private final FiniteFieldElementArithmetics fieldArithmetics;

    public EllipticCurve(FiniteFieldElementArithmetics fieldArithmetics, FiniteFieldElement a, FiniteFieldElement b,
EllipticCurvePoint G, BigInteger n, BigInteger h) {
        this.a = a;
        this.b = b;
        this.G = G;
        this.n = n;
        this.h = h;
        this.fieldArithmetics = fieldArithmetics;
    }

    public FiniteFieldElementArithmetics getFieldArithmetics() {
        return fieldArithmetics;
    }

    public FiniteFieldElement getA() {
        return a;
    }

    public FiniteFieldElement getB() {
        return b;
    }

    public EllipticCurvePoint getG() {
        return G;
    }

    public BigInteger getN() {
        return n;
    }

    public BigInteger getH() {
        return h;
    }

    @Override
    public String toString(){
        StringBuilder sb = new StringBuilder();
        sb.append("Elliptic curve with:")
                .append("\n")
                .append(getFieldArithmetics())
                .append("\n")
                .append("With params:\n")
                .append("A = ")
                .append(getA())
                .append("\n")
                .append("B = ")
                .append(getB())
                .append("\n")
                .append("G = ")
                .append(getG())
                .append("\n")
```

```
                .append("n = ")
                .append(getN())
                .append("\n")
                .append("H = ")
                .append(getH())
                .append("\n");
        return sb.toString();
    }

    /**
     * static factory method producing elliptic curve from
     * standard specification
     *
     * should prefer this over custom constructor call
     * @param spec
     * @return
     */
    public static EllipticCurve createFrom(SECP spec){
        return createFrom(spec, spec.getType()?new FiniteFieldElementFactory():new
BinaryExtensionFieldElementFactory());
    }

    private static EllipticCurve createFrom(SECP spec, FiniteFieldElementFactory factory){
        FiniteFieldElementArithmetics arithmetics =
                FiniteFieldElementArithmetics.createFieldElementArithmetics(spec.getType()?new
BigInteger(spec.getP(),16):BinaryExtensionFieldElement.fromString(spec.getP()));

        return new EllipticCurve(arithmetics,
                factory.createFrom(spec.getA()),
                factory.createFrom(spec.getB()),
                EllipticCurvePoint.create(factory.createFrom(spec.getGx()), factory.createFrom(spec.getGy())),
                spec.getN(),
                spec.getH());
    }
}
```

SECP.java

```
/*
 * Copyright 2018 trident.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.trident.crypto.elliptic.nist;
import java.math.BigInteger;

/**
 * Recommended Elliptic Curve parameters by SECP
 * @see http://www.secg.org/sec2-v2.pdf
 * @author trident
 */
public enum SECP {
    SECP112R1(
            "DB7C2ABF 62E35E66 8076BEAD 208B",
            "DB7C2ABF 62E35E66 8076BEAD 2088",
            "659EF8BA 043916EE DE891170 2B22",
```

```
        "09487239 995A5EE7 6B55F9C2 F098",
        "A89CE5AF 8724C0A2 3E0E0FF7 7500",
        "DB7C2ABF 62E35E76 28DFAC65 61C5",
        "01",true
        ),
    SECP112R2(
        "DB7C2ABF 62E35E66 8076BEAD 208B",
        "6127C24C 05F38A0A AAF65C0E F02C",
        "51DEF181 5DB5ED74 FCC34C85 D709",
        "4BA30AB5 E892B4E1 649DD092 8643",
        "ADCD46F5 882E3747 DEF36E95 6E97",
        "36DF0AAF D8B8D759 7CA10520 D04B",
        "04",true
        ),
    SECP128R1(
        "FFFFFFFD FFFFFFFF FFFFFFFF FFFFFFFF",
        "FFFFFFFD FFFFFFFF FFFFFFFF FFFFFFFC",
        "E87579C1 1079F43D D824993C 2CEE5ED3",
        "161FF752 8B899B2D 0C28607C A52C5B86",
        "CF5AC839 5BAFEB13 C02DA292 DDED7A83",
        "FFFFFFFE 00000000 75A30D1B 9038A115",
        "01",true
        ),
    SECP128R2(
        "FFFFFFFD FFFFFFFF FFFFFFFF FFFFFFFF",
        "D6031998 D1B3BBFE BF59CC9B BFF9AEE1",
        "5EEEFCA3 80D02919 DC2C6558 BB6D8A5D",
        "7B6AA5D8 5E572983 E6FB32A7 CDEBC140",
        "27B6916A 894D3AEE 7106FE80 5FC34B44",
        "3FFFFFFF 7FFFFFFF BE002472 0613B5A3",
        "04",true
        ),
    SECP192K1(
        "FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFEE37",
        "00000000 00000000 00000000 00000000 00000000 00000000",
        "00000000 00000000 00000000 00000000 00000000 00000003",
        "DB4FF10E C057E9AE 26B07D02 80B7F434 1DA5D1B1 EAE06C7D",
        "9B2F2F6D 9C5628A7 844163D0 15BE8634 4082AA88 D95E2F9D",
        "FFFFFFFF FFFFFFFF FFFFFFFE 26F2FC17 0F69466A 74DEFD8D",
        "01",true
        ),
    SECP192R1(
        "FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF FFFFFFFF",
        "FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF FFFFFFFC",
        "64210519 E59C80E7 0FA7E9AB 72243049 FEB8DEEC C146B9B1",
        "188DA80E B03090F6 7CBF20EB 43A18800 F4FF0AFD 82FF1012",
        "07192B95 FFC8DA78 631011ED 6B24CDD5 73F977A1 1E794811",
        "FFFFFFFF FFFFFFFF FFFFFFFF 99DEF836 146BC9B1 B4D22831",
        "01",true
        ),
    SECP224K1(
        "FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFE56D",
        "00000000 00000000 00000000 00000000 00000000 00000000 00000000",
        "00000000 00000000 00000000 00000000 00000000 00000000 00000005",
        "A1455B33 4DF099DF 30FC28A1 69A467E9 E47075A9 0F7E650E B6B7A45C",
        "7E089FED 7FBA3442 82CAFBD6 F7E319F7 C0B0BD59 E2CA4BDB 556D61A5",
        "01 00000000 00000000 00000000 0001DCE8 D2EC6184 CAF0A971 769FB1F7",
        "01",true
        ),
    SECP224R1(
        "FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 00000000 00000001",
        "FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFE",
        "B4050A85 0C04B3AB F5413256 5044B0B7 D7BFD8BA 270B3943 2355FFB4",
        "B70E0CBD 6BB4BF7F 321390B9 4A03C1D3 56C21122 343280D6 115C1D21",
        "BD376388 B5F723FB 4C22DFE6 CD4375A0 5A074764 44D58199 85007E34",
```

```
        "FFFFFFFF FFFFFFFF FFFFFFFF FFFF16A2 E0B8F03E 13DD2945 5C5C2A3D",
        "01",true
        ),
    SECP256K1(
        "FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFC2F",
        "00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000",
        "00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000007",
        "79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798",
        "483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8",
        "FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141",
        "01",true
        ),
    SECP256R1(
        "FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFF",
        "FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFC",
        "5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E 27D2604B",
        "6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C296",
        "4FE342E2 FE1A7F9B 8EE7EB4A 7C0F9E16 2BCE3357 6B315ECE CBB64068 37BF51F5",
        "FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84 F3B9CAC2 FC632551",
        "01",true
        ),
    SECP384R1(
        "FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF
00000000 00000000 FFFFFFFF",
        "FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF
00000000 00000000 FFFFFFFC",
        "B3312FA7 E23EE7E4 988E056B E3F82D19 181D9C6E FE814112 0314088F 5013875A C656398D
8A2ED19D 2A85C8ED D3EC2AEF",
        "AA87CA22 BE8B0537 8EB1C71E F320AD74 6E1D3B62 8BA79B98 59F741E0 82542A38 5502F25D
BF55296C 3A545E38 72760AB7",
        "3617DE4A 96262C6F 5D9E98BF 9292DC29 F8F41DBD 289A147C E9DA3113 B5F0B8C0 0A60B1CE
1D7E819D 7A431D7C 90EA0E5F",
        "FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF C7634D81 F4372DDF 581A0DB2
48B0A77A ECEC196A CCC52973",
        "01",true
        ),
    SECP521R1(
        "01FF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF",
        "01FF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFC",
        "0051 953EB961 8E1C9A1F 929A21A0 B68540EE A2DA725B 99B315F3 B8B48991 8EF109E1 56193951
EC7E937B 1652C0BD 3BB1BF07 3573DF88 3D2C34F1 EF451FD4 6B503F00",
        "00C6 858E06B7 0404E9CD 9E3ECB66 2395B442 9C648139 053FB521 F828AF60 6B4D3DBA A14B5E77
EFE75928 FE1DC127 A2FFA8DE 3348B3C1 856A429B F97E7E31 C2E5BD66",
        "0118 39296A78 9A3BC004 5C8A5FB4 2C7D1BD9 98F54449 579B4468 17AFBD17 273E662C 97EE7299
5EF42640 C550B901 3FAD0761 353C7086 A272C240 88BE9476 9FD16650",
        "01FF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFA 51868783
BF2F966B 7FCC0148 F709A5D0 3BB5C9B8 899C47AE BB6FB71E 91386409",
        "01",true
        ),
    SECT113R1(
        "x^113 x^9 1",
        "00308825 0CA6E7C7 FE649CE8 5820F7",
        "00E8BEE4 D3E22607 44188BE0 E9C723",
        "009D7361 6F35F4AB 1407D735 62C10F",
        "00A52830 277958EE 84D1315E D31886",
        "01000000 00000000 D9CCEC8A 39E56F",
        "02",false
        ),
    SECT163K1(
        "x^163 x^7 x^6 x^3 1",
        "00 00000000 00000000 00000000 00000000 00000001",
        "00 00000000 00000000 00000000 00000000 00000001",
        "02 FE13C053 7BBC11AC AA07D793 DE4E6D5E 5C94EEE8",
```

```
        "02 89070FB0 5D38FF58 321F2E80 0536D538 CCDAA3D9",
        "04 00000000 00000000 00020108 A2E0CC0D 99F8A5EF",
        "02",false
        ),
    SECT163R1(
        "x^163 x^7 x^6 x^3 1",
        "07 B6882CAA EFA84F95 54FF8428 BD88E246 D2782AE2",
        "07 13612DCD DCB40AAB 946BDA29 CA91F73A F958AFD9",
        "03 69979697 AB438977 89566789 567F787A 7876A654",
        "00 435EDB42 EFAFB298 9D51FEFC E3C80988 F41FF883",
        "03 FFFFFFFF FFFFFFFF FFFF48AA B689C29C A710279B",
        "02",false
        ),
    SECT163R2(
        "x^163 x^7 x^6 x^3 1",
        "00 00000000 00000000 00000000 00000000 00000001",
        "02 0A601907 B8C953CA 1481EB10 512F7874 4A3205FD",
        "03 F0EBA162 86A2D57E A0991168 D4994637 E8343E36",
        "00 D51FBC6C 71A0094F A2CDD545 B11C5C0C 797324F1",
        "04 00000000 00000000 000292FE 77E70C12 A4234C33",
        "02",false
        ),
    SECT233K1(
        "x^233 x^74 1",
        "0000 00000000 00000000 00000000 00000000 00000000 00000000 00000000",
        "0000 00000000 00000000 00000000 00000000 00000000 00000000 00000001",
        "0172 32BA853A 7E731AF1 29F22FF4 149563A4 19C26BF5 0A4C9D6E EFAD6126",
        "01DB 537DECE8 19B7F70F 555A67C4 27A8CD9B F18AEB9B 56E0C110 56FAE6A3",
        "80 00000000 00000000 00000000 00069D5B B915BCD4 6EFB1AD5 F173ABDF",
        "04",false
         ),
    SECT233R1(
        "x^233 x^74 1",
        "0000 00000000 00000000 00000000 00000000 00000000 00000000 00000001",
        "0066 647EDE6C 332C7F8C 0923BB58 213B333B 20E9CE42 81FE115F 7D8F90AD",
        "00FA C9DFCBAC 8313BB21 39F1BB75 5FEF65BC 391F8B36 F8F8EB73 71FD558B",
        "0100 6A08A419 03350678 E58528BE BF8A0BEF F867A7CA 36716F7E 01F81052",
        "0100 00000000 00000000 00000000 0013E974 E72F8A69 22031D26 03CFE0D7",
        "02",false
        ),
    SECT239K1(
        "x^239 x^158 1",
        "0000 00000000 00000000 00000000 00000000 00000000 00000000 00000000",
        "0000 00000000 00000000 00000000 00000000 00000000 00000000 00000001",
        "29A0 B6A887A9 83E97309 88A68727 A8B2D126 C44CC2CC 7B2A6555 193035DC",
        "7631 0804F12E 549BDB01 1C103089 E73510AC B275FC31 2A5DC6B7 6553F0CA",
        "2000 00000000 00000000 00000000 005A79FE C67CB6E9 1F1C1DA8 00E478A5",
        "04",false
        ),
    SECT283K1(
        "x^283 x^12 x^7 x^5 1",
        "00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000",
        "00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000001",
        "0503213F 78CA4488 3F1A3B81 62F188E5 53CD265F 23C1567A 16876913 B0C2AC24 58492836",
        "01CCDA38 0F1C9E31 8D90F95D 07E5426F E87E45C0 E8184698 E4596236 4E341161 77DD2259",
        "01FFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFE9AE 2ED07577 265DFF7F 94451E06 1E163C61",
        "04",false
        ),
    SECT283R1(
        "x^283 x^12 x^7 x^5 1",
        "00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000001",
        "027B680A C8B8596D A5A4AF8A 19A0303F CA97FD76 45309FA2 A581485A F6263E31 3B79A2F5",
        "05F93925 8DB7DD90 E1934F8C 70B0DFEC 2EED25B8 557EAC9C 80E2E198 F8CDBECD 86B12053",
        "03676854 FE24141C B98FE6D4 B20D02B4 516FF702 350EDDB0 826779C8 13F0DF45 BE8112F4",
        "03FFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFEF90 399660FC 938A9016 5B042A7C EFADB307",
```

```
        "02",false
        ),
    SECT409K1(
        "x^409 x^87 1",
        "00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000",
        "00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000001",
        "0060F05F 658F49C1 AD3AB189 0F718421 0EFD0987 E307C84C 27ACCFB8 F9F67CC2 C460189E
B5AAAA62 EE222EB1 B35540CF E9023746",
        "01E36905 0B7C4E42 ACBA1DAC BF04299C 3460782F 918EA427 E6325165 E9EA10E3 DA5F6C42
E9C55215 AA9CA27A 5863EC48 D8E0286B",
        "7FFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFE5F 83B2D4EA 20400EC4
557D5ED3 E3E7CA5B 4B5C83B8 E01E5FCF",
        "04",false
        ),
    SECT409R1(
        "x^409 x^87 1",
        "00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000001",
        "0021A5C2 C8EE9FEB 5C4B9A75 3B7B476B 7FD6422E F1F3DD67 4761FA99 D6AC27C8 A9A197B2
72822F6C D57A55AA 4F50AE31 7B13545F",
        "015D4860 D088DDB3 496B0C60 64756260 441CDE4A F1771D4D B01FFE5B 34E59703 DC255A86
8A118051 5603AEAB 60794E54 BB7996A7",
        "0061B1CF AB6BE5F3 2BBFA783 24ED106A 7636B9C5 A7BD198D 0158AA4F 5488D08F 38514F1F
DF4B4F40 D2181B36 81C364BA 0273C706",
        "01000000 00000000 00000000 00000000 00000000 00000000 000001E2 AAD6A612 F33307BE 5FA47C3C
9E052F83 8164CD37 D9A21173",
        "02",false
         ),
    SECT571K1(
        "x^571 x^10 x^5 x^2 1",
        "00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000",
        "00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000001",
        "026EB7A8 59923FBC 82189631 F8103FE4 AC9CA297 0012D5D4 60248048 01841CA4 43709584
93B205E6 47DA304D B4CEB08C BBD1BA39 494776FB 988B4717 4DCA88C7 E2945283 A01C8972",
        "0349DC80 7F4FBF37 4F4AEADE 3BCA9531 4DD58CEC 9F307A54 FFC61EFC 006D8A2C 9D4979C0
AC44AEA7 4FBEBBB9 F772AEDC B620B01A 7BA7AF1B 320430C8 591984F6 01CD4C14 3EF1C7A3",
        "02000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 131850E1
F19A63E4 B391A8DB 917F4138 B630D84B E5D63938 1E91DEB4 5CFE778F 637C1001",
        "04",false
        );


    private final String p;
    private final BigInteger a;
    private final BigInteger b;
    private final BigInteger Gx;
    private final BigInteger Gy;
    private final BigInteger n;
    private final BigInteger h;
    private final boolean type;

    private SECP(String p, String a, String b, String Gx, String Gy, String n, String h, boolean type) {
        this.p = type?p.replace(" ", ""):p;
        this.a = new BigInteger(a.replace(" ", ""), 16);
        this.b = new BigInteger(b.replace(" ", ""), 16);
        this.Gx = new BigInteger(Gx.replace(" ", ""), 16);
        this.Gy = new BigInteger(Gy.replace(" ", ""), 16);
        this.n = new BigInteger(n.replace(" ", ""), 16);
        this.h = new BigInteger(h.replace(" ", ""), 16);
        this.type = type;
    }
```

```
/**
 * @return type of these elliptic curve parameters
 * true if defined over prime field
 * false if defined over binary extension field
 */
public boolean getType() {
    return type;
}

/**
 * @return
 * if prime field - > order of the field
 * if binary extension field - > irreducible polynomial in x^n notation
 *
 */
public String getP() {
    return p;
}

/**
 *
 * @return parameter a of elliptic curve equation
 */
public BigInteger getA() {
    return a;
}

/**
 *
 * @return parameter b of elliptic curve equation
 */
public BigInteger getB() {
    return b;
}

/**
 *
 * @return x coordinate of generator point
 */
public BigInteger getGx() {
    return Gx;
}

/**
 *
 * @return y coordinate of generator point
 */
public BigInteger getGy() {
    return Gy;
}

/**
 *
 * @return order of generator point
 */
public BigInteger getN() {
    return n;
}

/**
 *
 * @return cofactor i.e. h = order of field/n
 */
public BigInteger getH() {
```

```
        return h;
    }
}
```

FiniteFieldElementArithmetics.java

```
/*
 * Copyright 2018 trident.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.trident.crypto.field.operator;

import com.trident.crypto.field.BinaryExtensionField;
import com.trident.crypto.field.FiniteField;
import com.trident.crypto.field.PrimeField;
import com.trident.crypto.field.element.BinaryExtensionFieldElement;
import com.trident.crypto.field.element.BinaryExtensionFieldElementFactory;
import com.trident.crypto.field.element.FiniteFieldElement;
import com.trident.crypto.field.element.FiniteFieldElementFactory;
import java.math.BigInteger;

/**
 * arithmetics of elements in prime field
 * @author trident
 */
public abstract class FiniteFieldElementArithmetics{

    private final FiniteField field;
    private final FiniteFieldElementFactory elementFactory;

    FiniteFieldElementArithmetics(FiniteField field, FiniteFieldElementFactory elementFactory){
        this.field = field;
        this.elementFactory = elementFactory;
    }

    /**
     * add two finite field elements
     * @param el1
     * @param el2
     * @return sum of elements
     */
    public abstract FiniteFieldElement add(FiniteFieldElement el1, FiniteFieldElement el2);

    /**
     * subtract two finite field elements
     * @param el1
     * @param el2
     * @return
     */
    public abstract FiniteFieldElement sub(FiniteFieldElement el1, FiniteFieldElement el2);

    /**
     * multiply finite field elements
     * @param el1
```

```
     * @param el2
     * @return multiple of elements
     */
    public abstract FiniteFieldElement mul(FiniteFieldElement el1, FiniteFieldElement el2);

    /**
     * find inverse of element
     * @param el1
     * @return inverse
     */
    public abstract FiniteFieldElement inv(FiniteFieldElement el1);

    /**
     * find the rest of element which belongs to the field
     * @param el1
     * @return
     */
    public abstract FiniteFieldElement mod(FiniteFieldElement el1);

    /**
     * return the element x such that el1 + x = 0 mod order
     * @param el1
     * @return
     */
    public abstract FiniteFieldElement complement(FiniteFieldElement el1);

    /**
     *
     * @return the field over which this arithmetics is performed
     */
    public FiniteField getField(){
        return field;
    }

    /**
     *
     * @return the factory producing the elements of this field
     */
    public FiniteFieldElementFactory getElementFactory(){
        return elementFactory;
    }

    @Override
    public String toString(){
        return "Arithmetics defined over field:"+getField();
    }

    /**
     * static factory method to create the arithmetics based on order
     * if fieldOrder instanceof BigInteger -> creates PrimeFieldElementArithmetics
     * if fieldOrder instanceof BinaryExtensionFieldElement -> creates BinaryExtensionFieldElementArithmetics
     * @param fieldOrder
     * @return
     */
    public static FiniteFieldElementArithmetics createFieldElementArithmetics(BigInteger fieldOrder){
        if(fieldOrder instanceof BinaryExtensionFieldElement) return
createFieldElementArithmetics((BinaryExtensionFieldElement)fieldOrder);
        return new PrimeFieldElementArithmetics(new PrimeField(fieldOrder), new FiniteFieldElementFactory());
    }

    public static FiniteFieldElementArithmetics createFieldElementArithmetics(BinaryExtensionFieldElement
fieldIrreduciblePoly){
        return new BinaryExtensionFieldElementArithmetics(new BinaryExtensionField(fieldIrreduciblePoly), new
BinaryExtensionFieldElementFactory());
    }
```

ДОДАТОК 3

Копія презентації

МОДИФІКАЦІЯ МЕТОДУ БАГАТОКРАТНОГО СКАЛЯРНОГО МНОЖЕННЯ ТОЧОК ЕЛІПТИЧНОЇ КРИВОЇ У СКІНЧЕННИХ ПОЛЯХ

Магістрант Дичка А.І.

Науковий керівник к.т.н. Онай М.В.

Об’єкт дослідження: процеси обміну ключами, шифрування/дешифрування та створення/перевірка електронно-цифрового підпису у еліптичних криптосистемах.

Предмет дослідження: методи багатократного скалярного множення точок еліптичної кривої у полі GF(p) та GF(2^m).

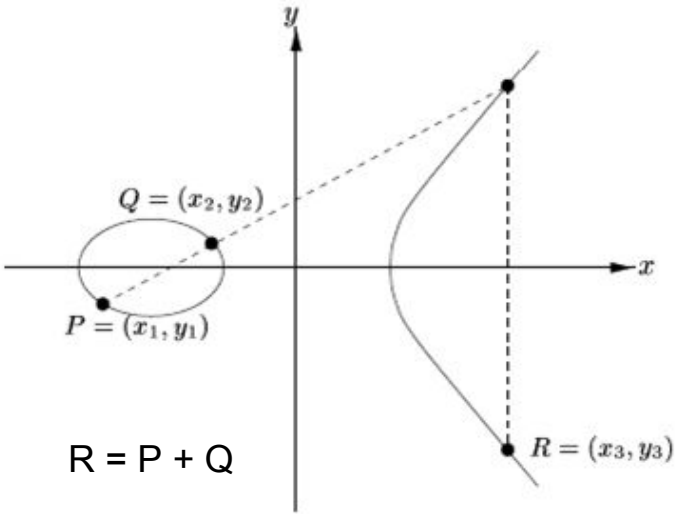
Мета дослідження: розробити метод високошвидкісного множення точок еліптичної кривої на число.

Теоретичні відомості

Еліптичною кривою називається крива, визначена наступним рівнянням:

$y^2 = x^3 + ax + b$ для полів з простою характеристикою GF(p)

$y^2 + xy = x^3 + ax^2 + b, \quad b \neq 0$ для полів виду GF(2^m)



R = P + Q

R = k•Q = Q+Q+Q…(k разів)

Q+(-Q) = O (точка на нескінченності)

Теоретичні відомості. Еліптична криптографія

Обчислюється точка C=k•G де k - велике випадкове число, G - точка еліптичної кривої великого порядку.
Пара C,k називається публічним та приватним ключами

Знаходження k при відомому C (ECDLP)
має велику обчислювальну складність: $O(\exp(c(\log p \log\log p)^d))$

Використання: алгоритм електронно-цифрового підпису (ECDSA)

Теоретичні відомості. ECDSA

Алгоритм електронно-цифрового підпису дозволяє засвідчити, що інформація, надіслана відправником та отримана отримувачем не була змінена в процесі передачі.

Під час виконання алгоритму необхідно виконати обчислення $R = k \cdot P + l \cdot Q$, що є найбільш ресурсоємкою операцією

Обчислення $R = k \cdot P + l \cdot Q$ називається багатократним скалярним множенням точки еліптичної кривої на число, англ. multiple point multiplication

Multiple point multiplication. Існуючі підходи

В межах даної роботи були розглянуті та проаналізовані наступні методи:

- SMPM (simultaneous multiple point multiplication)
- NAF (non-adjacent form)
- JSF (joint sparse form)
- DBNS (double based number system)

Огляд існуючих підходів. NAF, JSF, DBNS

DBNS - форма подання числа у вигляді:

$$n = \sum_{i=1}^{M} s_i 2^{b_i} 3^{t_i}$$

NAF (non-adjacent form) це знаково-розрядна форма подання числа, тобто подання вигляду:

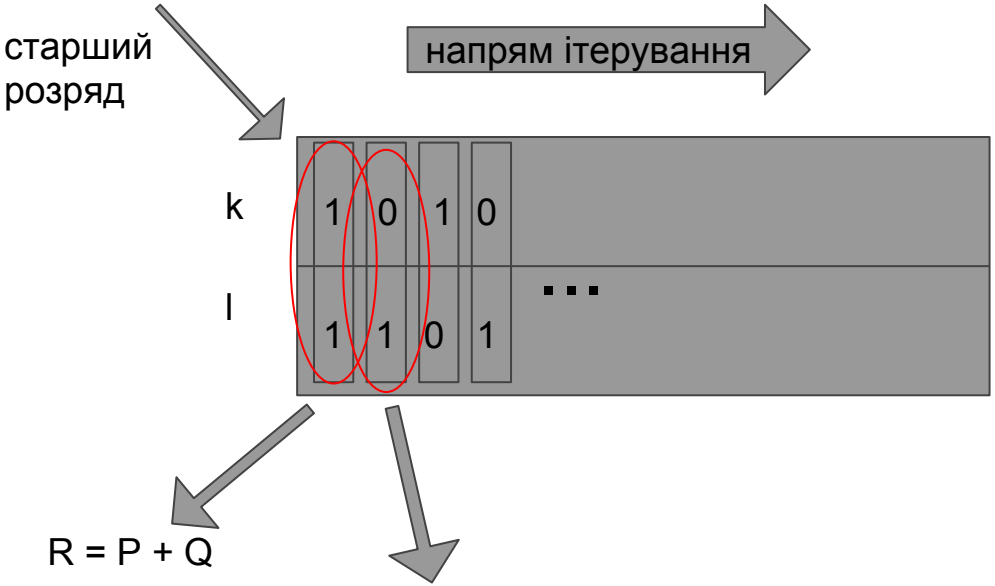
$$\sum_{i=0}^{l} x_i d_i^{\,i},\ x_i \in \{1, -1, 0\}$$

JSF (joint sparse form) це покращення NAF, що дозволяє ще більше зменшити кількість ненульових розрядів.

Таблиця 1. Приклад подання NAF та JSF для чисел 53 та 102

Число	Двійкове подання	NAF подання	JSF подання
53	110101	10-10101	100-10-1-1
102	1110101	10-1010-10	10-100110

Огляд існуючих підходів. SMPM, NAF, JSF



В методі SMPM виконується аналогічна операція, але не над розрядом, а над блоком (w-біт), тому необхідне передобчислення.

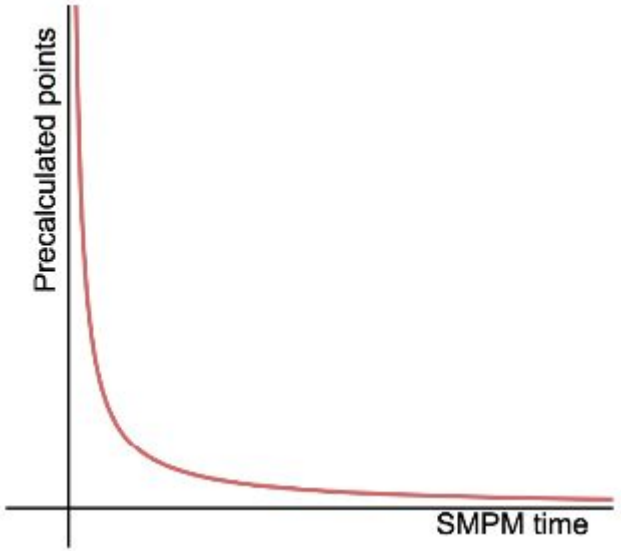
Огляд існуючих підходів. Підсумки аналізу

Загалом, для методів з використанням NAF та JSF необхідне передобчислення всього 4-х значень:

$$\{P+Q;\ P-Q;\ -P-Q;\ -P+Q\}$$

З іншого боку, кількість передобчислень при використанні SMPM пропорційна розміру блоку.

Хоч швидкодія SMPM і вища у порівнянні з JSF та NAF, кількість передобчислень може займати нераціонально великий час.



Запропонована модифікація

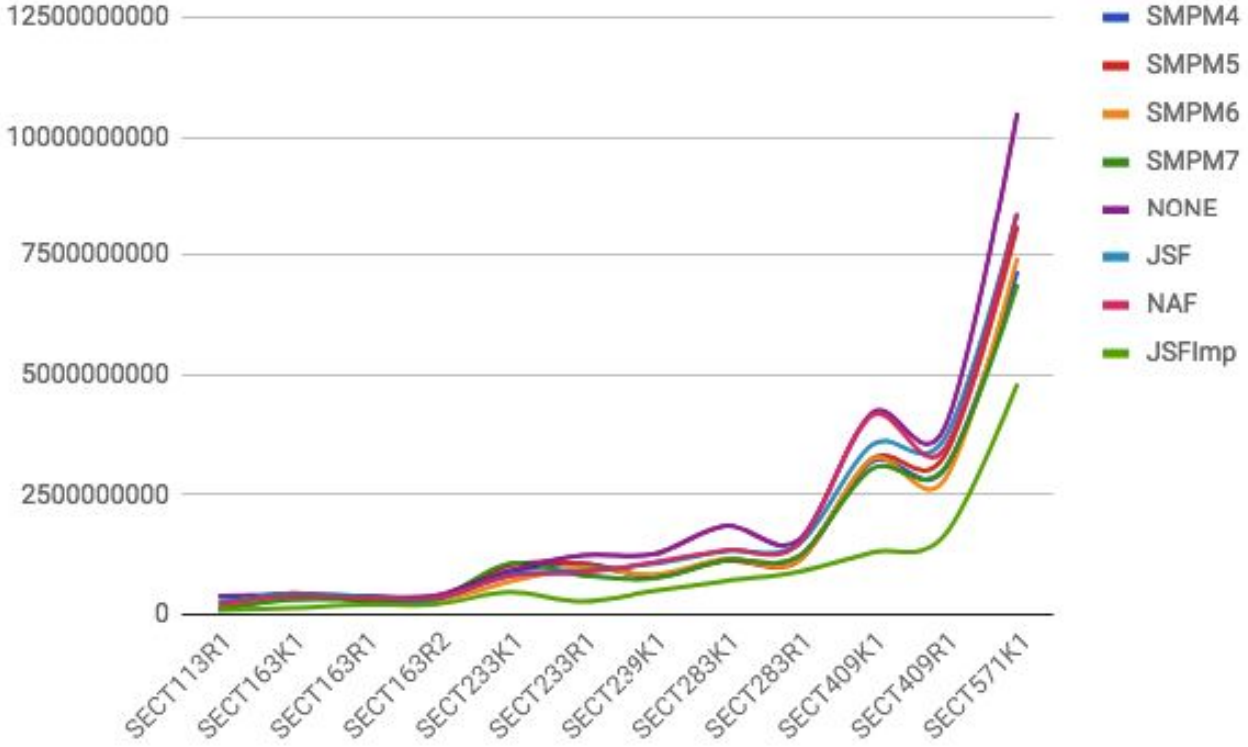
Для прискорення виконання методу багатократного скалярного множення пропонується передобчислення, що дозволяє уникнути операції зсуву результату вліво, тобто множення на 2. Для цього пропонується виконати передобчислення:

$$2^n(d_iP + d_jQ),\ d_{ij} \in \{-1, 0, 1\}. \quad n \in [0, l)$$

Таким чином, кількість передобчислень зростає до 8n, але загальна швидкодія є значно кращою.

Запропонований метод названо JSFImp (JSF Improved)

Запропонована модифікація. Результати



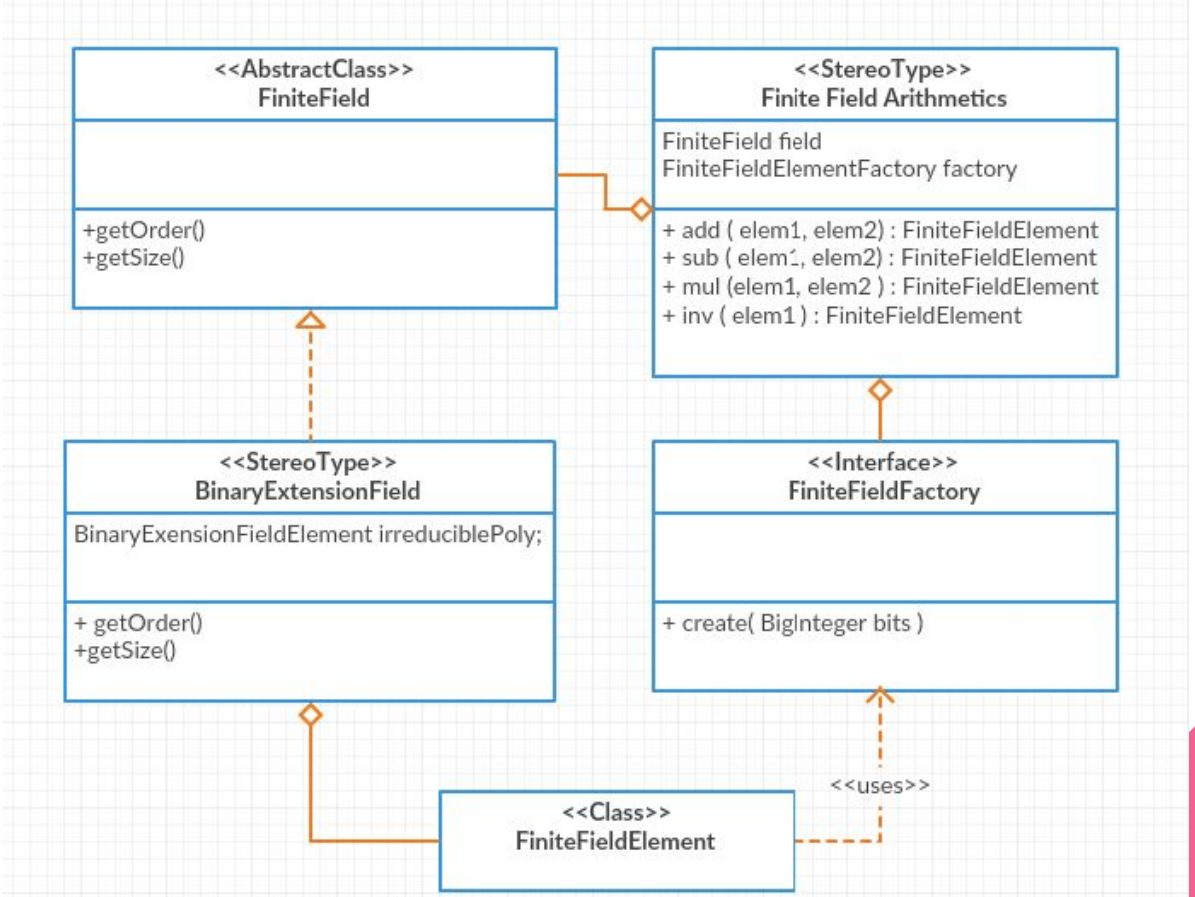
Інженерія програмного забезпечення

Розроблено програмний комплекс електронно-цифрового підпису (алгоритм ECDSA) та операцій в скінченних полях з використанням парадигми ООП на мові програмування JAVA.

Ключові особливості архітектури ПЗ:

1. Інкапсуляція виконання арифметичних операцій над елементами СП та точок ЕК
2. Обробка крайніх випадків, пов’язаних з операціями з “точкою на нескінченності”
3. Інкапсуляція методу МРМ в алгоритмі ЕЦП

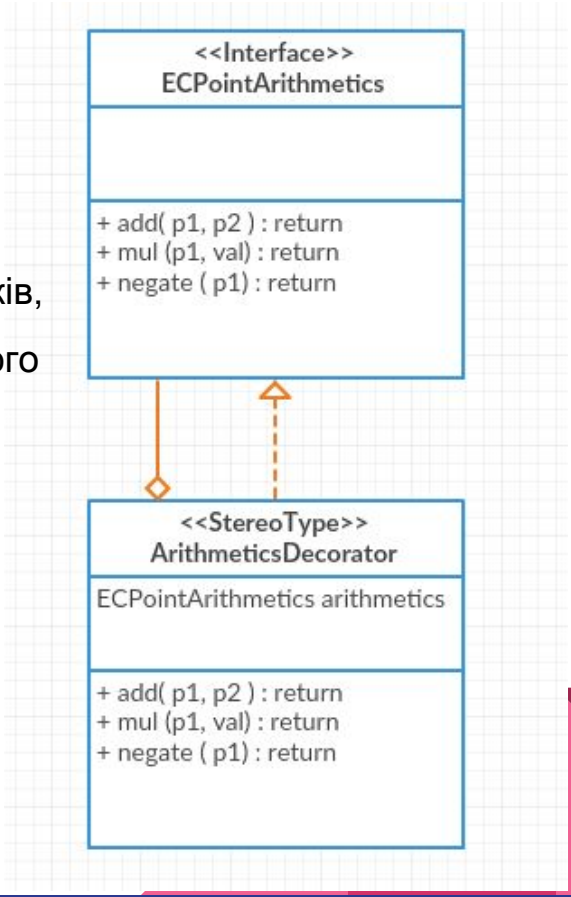
Особливості архітектури програмного комплексу



Особливості архітектури програмного комплексу

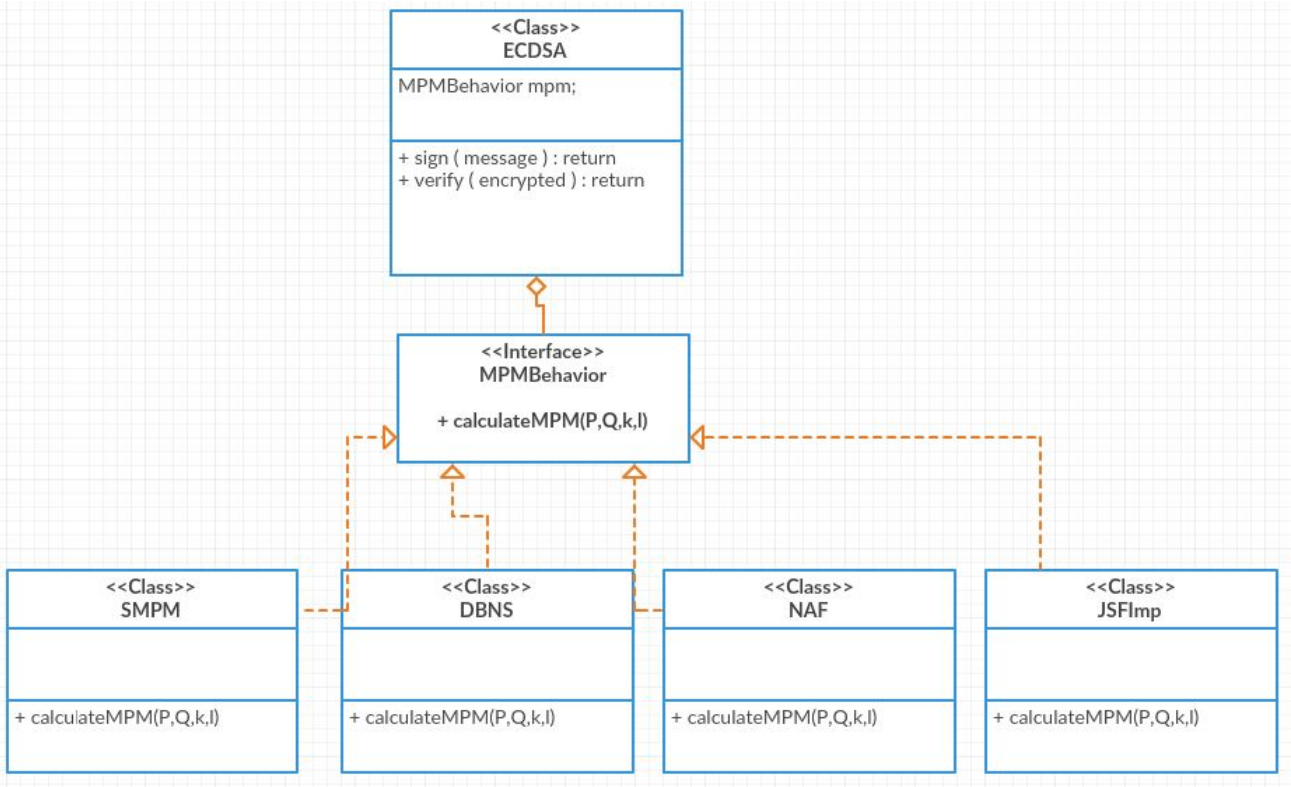
Обробка крайніх випадків, пов’язаних з операціями з “точкою на нескінченності”

Шаблон “Декоратор” дозволив відділити обробку крайніх випадків, пов’язаних з так званою “точкою на нескінченності” від звичайного виконання операцій. Тобто, код залишається залежним від абстракцій, обробка крайніх випадків інкапсулюється.



Особливості архітектури програмного комплексу

Інкапсуляція методу MPM в алгоритмі ЕЦП



Шаблон “Стратегія” дозволив інкапсулювати
виконання методу MPM

Інкапсуляція алгоритму - ключове архітектурне рішення.

Наукова новизна

1. Запропоновано модифікацію методу багатократного скалярного множення точки еліптичної кривої на число, яка полягає у виконанні спеціального передобчислення, і на відміну від існуючих методів, дозволяє звести виконання операції множення точки еліптичної кривої на число до додавання, що забезпечує вищу швидкодію (до 25%) порівняно з існуючими для еліптичних кривих з параметрами над.
2. Розроблено архітектуру програмного забезпечення, особливістю якої є інкапсуляція алгоритму багатократного скалярного множення точки еліптичної кривої на число, що дозволяє на основі шаблону «Стратегія» замінювати конкретну реалізацію в екземплярі класу електронно-цифрового підпису без застосування наслідування.

Результати перевірки тексту дисертації на предмет плагіату

Структурна одиниця	Відсоток оригінальності
Вступ	88%
Розділ 1	84%
Розділ2	80%
Розділ 3	95%
Розділ 4	80%
Розділ 5	98%
Висновок	88%
Загалом	88%

Висновки

Запропонована модифікація дає вищу швидкодію при перевірці електронно-цифрового підпису за алгоритмом ECDSA над еліптичною кривою з параметрами, визначеними над скінченним полем GF(2^m). Обчислювальна складність виконання передобчислень для запропонованої модифікації є набагато меншою ніж при використанні методу SMPM, який показує найкращі результати серед існуючих методів: O(8n) на відміну від O(2^2(n/d)) .

Розроблений програмний комплекс дозволяє ефективно проводити подальші дослідження в даній галузі.

Дякую за увагу