ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ

КАФЕДРА ІНФОРМАЦІЙНОЇ ТА КІБЕРНЕТИЧНОЇ БЕЗПЕКИ

Пояснювальна записка

до магістерської роботи
на тему:

«ТЕХНОЛОГІЯ ВИБОРУ ПАРАМЕТРІВ АЛГОРИТМУ ECDSA ДЛЯ ЗАБЕЗПЕЧЕННЯ НАЛЕЖНОГО РІВНЯ КРІПТОСТІЙКОСТІ»

Виконав студент 6 курсу, групи БСДМ-51
спеціальності 125 Кібербезпека
освітньо-професійної програми«Інформаційна та кібернетична безпека»
(шифр і назва спеціальності)
Качура А.О.
(прізвище та ініціали)
Керівник Марченко В.В.
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)
Нормоконтролер Чумак Н.С.
(прізвище та ініціали)

КИЇВ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Інститут ННІЗІ
Кафедра Інформаційної та кібернетичної безпеки
Ступінь вищої освіти Магістр
Спеціальність 125 Кібербезпека
Освітньо-професійна програма Інформаційна та кібернетична безпека

ЗАТВЕРДЖУЮ
Завідувач кафедри ІКБ
_____ Гайдур Г.І.
“___”_____ 2022 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ
СТУДЕНТУ

Качурі Андрію Олександровичу
(прізвище, ім’я, по батькові)

1. Тема магістерської роботи: «Технологія вибору параметрів алгоритму ECDSA для забезпечення належного рівня кріптостійкості»

керівник магістерської роботи Марченко Віталій Вікторович, ст. викладач
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «12» жовтня 2022 року №122.

2. Строк подання студентом магістерської роботи 15.12.2022 р.

3. Вихідні дані до магістерської роботи
критпографія на еліптичних кривих;
алгоритм цифрового підпису ECDSA;
наукова та технічна література, експлуатаційна документація, нормативні документи, міжнародні стандарти.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Актуальність проблеми криптографії, побудованої на еліптичних кривих.
2. Параметри алгоритму цифрового підпису ECDSA.
3. Програмний модуль реалізації алгоритму ECDSA.

5. Перелік графічного матеріалу
1. Тема магістерської роботи.
2. Об’єкт, предмет, мета та наукові завдання дослідження.
3. Основні положення криптографії на еліптичних кривих.

4. Основні вимоги до еліптичних кривих.
5. Вимоги до параметрів алгоритму ECDSA.
6. Програмна реалізація алгоритму ECDSA.
7. Висновки за результатами роботи.

6. Дата видачі завдання 26.09.2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ зп	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1.	Визначення актуальності проблеми криптографії, побудованою на еліптичних кривих.	26.09.2022 р.	
2.	Аналіз наукової та технічної літератури з питань теми магістерської роботи.	30.09.2022 р.	
3.	Аналіз положень криптографії на еліптичних кривих.	10.10.2022 р.	
4.	Аналіз параметрів алгоритму цифрового підпису ECDSA.	10.11.2022 р.	
5.	Розроблення програмного модуля реалізації алгоритму ECDSA.	30.112022 р.	
6.	Оформлення результатів дослідження. Проходження плагіату	14.11.2022 р.	
7.	Підготовка доповіді до захисту.	15.12.2022 р.	

Студент _____ Качура А.О.
(підпис) прізвище та ініціали

Керівник магістерської роботи _____ Марченко В.В.
(підпис) прізвище та ініціали

ABSTRACT

The diploma thesis contains 71 pages, 4 tables, 10 figures, a list of used literature from 23 titles.

The relevance of the study is due to the advantage of cryptographic schemes using elliptic curves over others (in particular, RSA), due to a much higher calculation speed and greater cryptographic resistance of schemes with a similar length of keys.

The purpose of the study is to create a technology for selecting parameters of the ECDSA algorithm to ensure an appropriate level of cryptoresistance.

To achieve the goal in the work, the following tasks were solved:

– the principles of cryptography built on elliptic curves are investigated, the requirements for elliptic curves in the field of real numbers and the Galois field are defined, and the advantages of elliptic cryptography are given;

– the main requirements for the parameters of the ECDSA algorithm to ensure the appropriate level of crypto-resistance were investigated;

– a software module for the implementation of the ECDSA digital signature was developed.

The object of the study is to study the properties of digital signature algorithms built on elliptic curves over the Galois field.

The subject of research is the ECDSA digital signature algorithm parameters.

In the first chapter, the principles of cryptography built on elliptic curves are considered, the requirements for elliptic curves in the field of real numbers and the Galois field are determined, and the advantages of elliptic cryptography are given.

The second section discusses the main requirements for ECDSA digital signature parameters to ensure an adequate level of crypto-resistance.

In the third section, the ECDSA digital signature algorithm is practically implemented in the C++ programming language. The functions and algorithm of the created program are described, the results of experiments of this program are demonstrated.

ELLIPTIC CRYPTOGRAPHY, FINITE FIELDS, ELECTRONIC DIGITAL SIGNATURE, ECDSA ALGORITHM, ELLIPTIC CURVE PARAMETERS, PSEUDORANDOM NUMBER GENERATOR, CRYPTOGRAPHIC HASH FUNCTION, ELECTRONIC TRUST SERVICES

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 10
ВСТУП 11
1 ОБГРУНТУВАННЯ АКТУАЛЬНОСТІ КРИПТОГРАФІЇ НА ЕЛІПТИЧНИХ КРИВИХ 13
1.1 Захист інформації на основі криптографії еліптичних кривих 13
1.2 Визначення, властивості та переваги еліптичних кривих 19
1.3 Принципи еліптичної криптографії 25
1.4 Основні поняття про поля Галуа 27
Висновки до розділу 30
2 ПАРАМЕТРИ АЛГОРИТМУ ECDSA ТА ТЕХНОЛОГІЯ ЇХ ВИБОРУ 31
2.1 Алгоритм ECDSA та його параметри 31
2.1.1 Алгоритм створення ключів 32
2.1.2 Алгоритм створення підпису 33
2.1.3 Алгоритм перевірки підпису 34
2.2 Вимоги до параметрів еліптичної кривої 35
2.2.1 Вимоги до еліптичної кривої 35
2.2.2 Генерація основних параметрів 36
2.3 Вимоги до генераторів псевдовипадкових чисел 36
2.4 Вимоги до криптографічних хеш-функцій 38
Висновки до розділу 40
РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ ECDSA 42
3.1 Вибір мови програмування 42
3.2 Архітектура програмного комплексу 46
3.3 Опис логічної структури 47
3.4 Блок – схема алгоритму рішення задачі 48
3.5 Числові експерименти 51
Висновки до розділу 58

ВИСНОВКИ 59
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 61
ДОДАТОК А. Лістинг програми 63

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

DES – Data Encryption Standard – симетричний алгоритм шифрування певних даних.

PGP – Pretty Good Privacy – комп'ютерна програма, також бібліотека функцій, що дозволяє виконувати операції шифрування і цифрового підпису інформації, поданої в електронному вигляді.

TLS – Transport Layer Securitфу – криптографічний протокол, що надає можливості безпечної передачі даних в інтернеті для навігації, отримання пошти, спілкування, обміну файлами, тощо.

RSA – Rivest, Shamir та Adleman – криптографічний алгоритм з відкритим ключем, що базується на обчислювальній складності задачі факторизаціївеликих цілих чисел.

ECC – Elliptic Curve Cryptography – це метод асиметричної криптографії, заснований на використанні еліптичних кривих над скінченними полями.

SHA – Secure Hash Algorithm – алгоритм криптографічного гешування.

ECDLP – Elliptic Curve Discrete Logarithm Problem – задача дискретного логарифму на еліптичній кривій.

ЕЦП – електронно-цифровий підпис – вид електронного підпису, отриманого за результатом криптографічного перетворення набору електронних даних, який додається до цього набору або логічно з ним поєднується і дає змогу підтвердити його цілісність та ідентифікувати підписувача.

ЕК – еліптична крива.

КС – комп’ютерна система.

ECDSA (Elliptic Curve Digital Signature Algorithm) – це криптографічно захищена схема цифрового підпису, заснована на криптографії еліптичної кривої.

ГПВЧ – генератори псевдовипадкових чисел.

ВСТУП

Актуальність *дослідження***.** На даний момент в Україні надзвичайно широко використовуються такі електронні довірчі послуги як електронний цифровий підпис, електронна печатка та мітка часу. Суттєве місце для реалізації електронних довірчих послуг в Україні займає асиметрична криптографія та криптографія на еліптичних кривих як її складова.

Широке використання еліптичних кривих в криптографії засновано на тому, що завдання дискретного логарифмування на еліптичних кривих є більш трудомістким, ніж завдання дискретного логарифмування в скінчених полях.

Криптографія на еліптичних кривих – це розділ криптографії, який вивчає асиметричні криптосистеми, засновані на еліптичних кривих над скінченними полями. Сьогодні криптосистеми на еліптичних кривих використовуються в протоколах TLS, PGP, SSH, в криптовалютах для верифікації транзакцій тощо. Окрім того, в Україні діє національний стандарт ДСТУ 4145-2002, що описує механізм формування та перевірки ЕЦП, що базується на властивостях груп точок еліптичних кривих над полями $GF\left(2^m\right)$.

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, матимуть значно меншу довжину, що робить їх більш портативними.

Метою даної дипломної роботи є створення технології вибору параметрів алгоритму ECDSA для забезпечення належного рівня криптостійкості.

Для досягнення мети в роботі вирішені наступні **завдання**:

– досліджені принципи криптографії, побудованої на еліптичних кривих, визначено вимоги до еліптичних кривих в полі дійсних чисел та полі Галуа та

наведені переваги еліптичної криптографії (РОЗДІЛ 1);

– досліджені основні вимоги до параметрів алгоритму ECDSA для забезпечення належного рівня криптостійкості (РОЗДІЛ 2);

– розроблений програмний модуль реалізації цифрового підпису ECDSA (РОЗДІЛ 3).

При написанні роботи використовувалися теорії скінченних полів та інтегроване середовище розробки Code::Blocks 16.01.

Об’єктом дослідження є дослідження властивостей алгоритмів цифрового підпису, побудованих на еліптичних кривих над полем Галуа.

Предметом дослідження є параметри алгоритму цифрового підпису ECDSA.

Методи дослідження – опрацювання літератури за даною темою, аналіз експлуатаційної документації, міжнародних стандартів та їх порівняння, реалізація алгоритму цифрового підпису ECDSA.

Практичне значення одержаних результатів полягає в створення технології вибору параметрів алгоритму ECDSA для забезпечення належного рівня криптостійкості та розробці програмного модулю реалізації алгоритму ECDSA.

Апробація результатів. магістерської роботи було оприлюднено на Всеукраїнській науковій конференції «Актуальні проблеми кібербезпеки», яка відбулася 27 жовтня 2022 року в Державному університеті телекомунікацій, м. Київ.

1 ОБГРУНТУВАННЯ АКТУАЛЬНОСТІ КРИПТОГРАФІЇ НА ЕЛІПТИЧНИХ КРИВИХ

В даному розділі розглянуті принципи криптографії, побудованої на еліптичних кривих, визначено вимоги до еліптичних кривих в полі дійсних чисел та полі Галуа та наведені переваги еліптичної криптографії. У пункті 1.1 розглядаються причини появи криптографії на ЕК, наводяться переваги й недоліки використання математичного апарату ЕК в задачах шифрування даних, розподілу ключів та використання ЕЦП. У пункті 1.2 розглядається форма ЕК над скінченним полем, основні властивості множини точок ЕК. У пункті 1.3 наведені вимоги до ЕК та генерація основних параметрів. У пункті 1.4 розглянуті основні поняття щодо поля Галуа.

1.1 Захист інформації на основі криптографії еліптичних кривих

Інформаційні ресурси в сучасних умовах являються одним із найважливіших результатів діяльності людського суспільства. Саме тому особлива увага приділяється задачі захисту інформації.

Сучасна криптографія містить у собі чотири великих розділи:

- керування ключовою інформацією;
- системи електронно-цифрового підпису;
- симетричні криптоалгоритми;
- асиметричні криптоалгоритми.

Будь-яка криптографічна система базується на використанні криптографічних ключів. Під ключовою інформацією розуміють сукупність усіх діючих в системі ключів. Якщо не забезпечене досить надійне керування

ключовою інформацією, то, заволодівши нею, зловмисник отримує необмежений доступ до всієї інформації. Керування ключами – інформаційний процес, що включає реалізацію наступних основних функцій:

– генерація ключів;
– збереження ключів;
– розподіл ключів.

Системи ЕЦП призначені для перевірки авторства і дійсності повідомлення.

Симетричні криптосистеми для зашифрування і розшифрування використовують один і той же ключ. Ці системи виникли першими, одними із найбільш поширених є: DES, IDEA, ГОСТ 28147-89, RC5.

Якими б не були складними і надійними криптографічні системи – їх слабким місцем при практичній реалізації є проблема розподілу ключів. Для того, щоб був можливий обмін конфіденційною інформацією між двома суб’єктами КС, ключ повинен бути згенерований одним з них, а потім якимось чином знову ж у конфіденційному порядку переданий іншому. Тобто, у загальному випадку для передачі ключа знову ж потрібно використання деяких криптосистем.

Для рішення цієї проблеми на основі результатів, отриманих класичною і сучасною алгеброю, були запропоновані системи з відкритим ключем. Суть їх полягає в тому, що кожним адресатом КС генеруються два ключі, зв’язані між собою за визначеним правилом. Один ключ називається відкритим, а інший закритим. Відкритий ключ публікується і доступний кожному, хто бажає послати повідомлення адресату. Секретний ключ зберігається в таємниці.

Повідомлення, тобто деяка інформація, шифрується відкритим ключем адресата і передається йому. Зашифрований текст в принципі не може бути розшифрований тим же відкритим ключем. Дешифрування повідомлення можливе тільки з використанням закритого ключа, що відомий тільки самому адресату (Рис. 1.1).

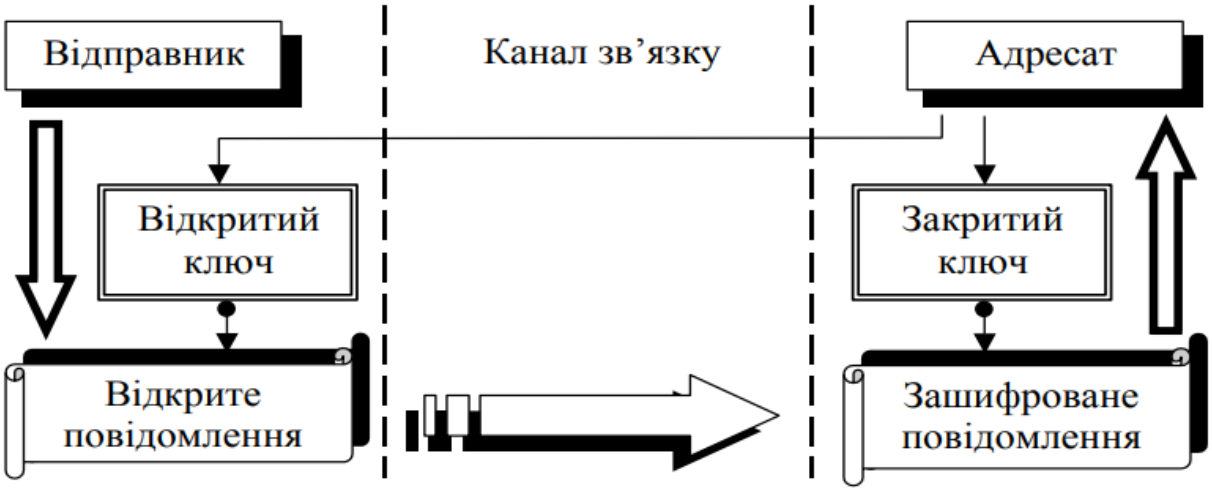


Рисунок 1.1 – Процедура шифрування і дешифрування з відкритим ключем

Безпека будь-якого криптографічного алгоритму визначається використовуваним криптографічним ключем. Криптостійкі криптографічні ключі повинні мати достатню довжину і випадкові значення бітів.

У табл. 1.1 наведені довжини ключів симетричної і асиметричної криптосистем, що забезпечують однакову стійкість до атаки повного перебору, brute-force attack.

Таблиця 1.1

Довжини ключів для криптосистем при однаковій їх криптостійкості

Довжина ключа симетричної криптосистеми, біт	**Довжина ключа асиметричної криптосистеми, біт**
56	384
64	512
80	768
112	1792
128	2304

Криптографічні системи з відкритим ключем використовують так звані необоротні (важкооборотні) або односторонні функції, що мають наступну властивість: при заданому значенні x відносно просто обчислити значення

функції $f(x)$, однак якщо відомо саму функцію $f(x)$, то немає простого шляху для обчислення значення *x* [1].

Велика кількість класів необоротних функцій і породжує всю різноманітність систем з відкритим ключем. Однак не кожна така функція придатна для використання в реальних КС.

В самому визначенні важкооборотність присутня деяка невизначеність. Тобто тут розуміється не теоретична необоротність, а практична неможливість обчислити зворотне значення, використовуючи сучасні обчислювальні засоби за деякий невеликий інтервал часу.

Тому для того, щоб гарантувати надійний захист інформації, до систем з відкритим ключем пред'являються дві важливих і очевидних вимоги:

– перетворення вихідного тексту повинне бути необоротним і виключати його відновлення на основі відкритого ключа;

– визначення закритого ключа на основі відкритого також повинне бути неможливим на сучасному технологічному рівні.

Алгоритми шифрування з відкритим ключем одержали широке поширення в сучасних ІС. Так, алгоритм RSA став світовим стандартом для відкритих систем. Цей алгоритм запропонували в 1978 році троє авторів: Р. Райвест (Rivest), А. Шамір (Shamir) і А. Адлеман (Adleman). Алгоритм одержав свою назву за першими буквами прізвищ його авторів. Алгоритм RSA став першим повноцінним алгоритмом з відкритим ключем, що може працювати як в режимі шифрування даних, так і в режимі ЕЦП. Надійність алгоритму ґрунтується на складності задач факторизації великих чисел.

Взагалі ж усі запропоновані на сьогодні криптосистеми з відкритим ключем спираються на один з наступних типів необоротних перетворень:

– розклад великих чисел на прості множники або задача факторизації;

– обчислення дискретних логарифмів;

– обчислення коренів алгебраїчних рівнянь.

Крім RSA на сьогодні найбільш відомими алгоритмами, які використовують технологію відкритого ключа, є схема шифрування ЕльГамаля, алгоритм Поліга-Хеллмана, криптосистеми на основі еліптичних кривих.

Криптографія з відкритим ключем може застосовуватися як для шифрування повідомлень, так і для автентифікації (тобто цифровий підпис).

Кожна з цих систем покладається на важку математичну проблему для її захисту. Вони є важкооборотними, тому що роки інтенсивного вивчення провідними математиками і комп'ютерними вченими не зуміли створити ефективні алгоритми для їхнього вирішення, так, щоб практично ці алгоритми залишилися важкооброблюваними з поточною обчислювальною технологією. Потрібен час, щоб одержати безпечний ключ із кращим відомим алгоритмом для цієї проблеми.

Еліптичні криві – це математичні конструкції, що вивчалися математиками, починаючи із сімнадцятого століття. У 1985 році Нейл Кобліц і Віктор Міллер незалежно один від одного запропонували криптосистеми з ключем загального користування, що використовують групу точок на еліптичній кривій [2].

Безпека криптосистем на основі еліптичних кривих, як правило, заснована на складності рішення задачі дискретного логарифмування в групі точок еліптичної кривої над скінченним полем. Цим і обумовлена їх висока криптостійкість у порівнянні з іншими алгоритмами. Існують стійкі криптографічні алгоритми на ЕК, засновані на складності розкладання великих цілих чисел, коли ЕК задається над скінченним кільцем по складеному модулю, але вони зустрічаються досить рідко. Однак слід зазначити, що криптостійкість є відносним поняттям, пов'язаним з поняттям найкращого відомого алгоритму злому системи. Стосовно криптосистем на ЕК – це алгоритми Сільвера-Поліга-Хеллмана, Полларда тощо.

За час, який пройшов від моменту перших публікацій про криптосистеми на основі еліптичних кривих, помітного падіння стійкості алгоритмів ЕК не відбулося. У той час, як стійкість системи RSA, заснованої на задачі розкладання на множники, знижувалася приблизно на порядок у рік.

Усе це свідчить про високий рівень криптостійкості алгоритмів на ЕК. Правда, вони ніколи широко не використовувалися на практиці і не залучали до себе такої пильної уваги наукової громадськості як, наприклад, RSA. Зараз ситуація змінюється, отже цілком можна чекати появи багатьох сюрпризів для розробників криптосистем на основі еліптичних кривих.

Іншою перевагою криптосистем на ЕК є висока швидкість обробки інформації. Але і тут не все так просто. Зрозуміло, що, володіючи більш високою криптостійкістю, криптосистеми на ЕК дозволяють використовувати ключ меншої довжини. Однак прийнятна для роботи в мережах швидкість обчислень досягається лише при використанні спеціалізованих обчислювачів (це цілком природно для криптосистем з відкритим ключем) і полів спеціальних характеристик. Якщо використовується ЕК над простим полем з характеристикою у вигляді великого простого числа, то про високу швидкість обчислень говорити важко.

Крім того, для шифрування даних за допомогою еліптичних кривих буде потрібно вводити інформацію, що підлягає шифруванню, в точку ЕК. По-перше, це додаткова процедура протоколу шифрування, а по-друге – це нетривіальна задача. Для її рішення буде потрібно вибрати придатний базис векторного простору над полем і з його допомогою асоціювати повідомлення з елементом поля, а потім і з самою точкою ЕК. При цьому на повідомлення, представлене цілим числом, накладається обмеження, що воно не повинно перевищувати порядок групи кривої. Це приводить до того, що повідомлення прийдеться розбивати на блоки і повторювати цю операцію для кожного блоку.

Тому криптосистеми на основі еліптичних кривих, як і інші криптосистеми з відкритим ключем, недоцільно застосовувати для шифрування великих об’ємів даних. Проте їх можна ефективно використовувати для систем цифрового підпису і обміну ключовою інформацією. Вже існує попередній стандарт ANSI X9.62, що пропонує розробникам принципи створення криптостійких систем цифрового підпису на основі використання еліптичних кривих. Безпека таких систем цифрового підпису спирається не тільки на стійкості алгоритму на ЕК, але і на

стійкості використовуваної хеш-функції. Не варто забувати і про вимоги до генератора випадкових чисел.

І все-таки, на сьогоднішній день криптосистеми на основі ЕК є найбільш перспективними асиметричними криптосистемами й у недалекому майбутньому їх чекає широке застосування в інформаційно-обчислювальних системах і мережах.

В Україні та за кордоном спеціалістами у сфері захисту інформації проводяться дослідження і здійснюється пошук можливих ефективних шляхів застосування еліптичних кривих в криптосистемах різних рівнів захисту.

Однією із найвідоміших є німецька корпорація Certicom. Це центр Європи в сфері вивчення і застосування еліптичних кривих та інших відомих асиметричних та симетричних криптосистем захисту інформації [3].

Департамент стандартизації IEEE в 1999 році прийняв стандарт P1363, в якому описані деякі розділи модульної арифметики, а також алгоритми генерування параметрів, додавання і множення точок та інше, що стосується еліптичних кривих [4].

1.2 Визначення, властивості та переваги еліптичних кривих

Особливий інтерес до криптографії еліптичних кривих обумовлений такими перевагами – швидкодія та невелика довжина ключа. Однак для забезпечення повного ефекту цих переваг необхідні спеціальні алгоритми та засоби для швидкої роботи алгоритму і генерації стійких параметрів [5].

У наведеній нижче табл. 1.2 порівнюються наближені розміри параметрів еліптичних систем і криптосистеми RSA, що забезпечують однакову стійкість шифру. Ці дані отримані на основі сучасних методів розв’язання задачі дискретного логарифмування еліптичної кривої та факторизації для великих цілих чисел.

Таблиця 1.2

Порівняння стійкості основних криптографічних алгоритмів

Система на основі еліптичної кривої		**RSA**	
Довжина ключа (біт)	**Час злому (MIPS-роки)**	**Довжина ключа (біт)**	**Час злому (MIPS-роки)**
150	$3{,}8 \cdot 10^{10}$	512	$3 \cdot 10^{4}$
205	$7{,}1 \cdot 10^{18}$	768	$2 \cdot 10^{8}$
234	$3{,}8 \cdot 10^{28}$	1024	$3 \cdot 10^{11}$
		1280	$1 \cdot 10^{14}$
		1536	$3 \cdot 10^{16}$
		2048	$3 \cdot 10^{20}$
ρ-метод Полларда для проведення дискретного логарифмування		метод факторизації чисел з допомогою решета числового поля загального вигляду	

Проаналізувавши таблицю видно, що використання еліптичних кривих дозволяє будувати стійкі системи з ключами значно менших розмірів у порівнянні до традиційних асиметричних криптографічних алгоритмів. Такі системи потребують меншого обсягу обчислювальних ресурсів, тому зручні для використання у смарт-картках та портативних телефонах.

На сьогодні еліптичні криві застосовують для реалізації різноманітних класів криптосистем, зокрема їх можна використовувати для побудови симетричних, асиметричних та систем електронного цифрового підпису, для електронних цифрових платежів і генераторів псевдовипадкових послідовностей.

Слід зауважити, що стосовно симетричних криптосистем, в літературі показано лише теоретичну можливість побудови засобів такого класу, стосовно ж практичної реалізації необхідно відмітити, що продуктивність таких систем є нижчою в порівнянні з традиційними.

Незважаючи на вагомі переваги застосування ЕК, поряд з цим існують певні проблеми та труднощі. Зокрема, виділяють такі класи задач:

– **Реальна безпека еліптичних систем досі недостатньо усвідомлена**. Головна проблема полягає в тому, що справжня складність дискретного логарифмування на еліптичній кривій ще не повністю усвідомлена. Недавнє дослідження показало, що деякі еліптичні криві, що використовувалися для відпрацювання алгоритмів шифрування, фактично не підходять для таких операцій.

– **Труднощі генерації відповідних кривих**. При визначенні системи еліптичної кривої потрібні крива та базова точка, створити які важко. Головна проблема – підрахувати кількість точок на кривій. Для цього необхідно вибрати потрібну базову точку, координати якої повинні мати досить велике значення, щоб гарантувати труднощі злому. Але координати повинні ділитися на кількість точок на кривій. І цілком імовірно, що, знайшовши кількість точок на кривій, ми зможемо знайти базову точку.

Підсумовуючи вищевикладене, можна стверджувати, що створення кривих – непросте завдання. Користувачі можуть використовувати «стандартні» криві, використовуючи спеціальне програмне забезпечення або створювати власні криві, що займає багато часу.

– Відносно повільна перевірка цифрового підпису. Як вже згадувалося, системи на основі еліптичної кривої використовують ключі малих розмірів. Це знижує вимоги до обчислювальних потужностей, порівняно з вимогами систем на основі RSA. Як це впливає швидкість обробки? Наступна таблиця показує порівняльні характеристики алгоритмів RSA та ECDSA під час створення та перевірки підписів.

Таблиця 1.3

Порівняння швидкості створення та перевірки підпису основних криптографічних алгоритмів

	Створення підпису	**Перевірка підпису**
RSA (1024 біт)	25 ms	< 2 ms

Продовження таблиці 1.3

Порівняння швидкості створення та перевірки підпису основних криптографічних алгоритмів

	Створення підпису	**Перевірка підпису**
ECDSA (160 біт)	32 ms	33 ms
RSA (2048 біт)	120 ms	5 ms
ECDSA (216 біт)	68 ms	70 ms

Отже, зі збільшенням розмірів ключа створення підписів за допомогою ECDSA відбувається значно швидше, ніж у аналогічних RSA системах. З іншого боку, перевірка підпису за допомогою ECDSA проводиться набагато повільніше, ніж ця ж процедура в системах RSA.

Усі криптосистеми захисту інформації реалізовані апаратно, програмно або так і так. Будь-який криптографічний алгоритм може бути реалізований у вигляді деякої програми. Переваги такої реалізації очевидні:

– програмні засоби шифрування легко копіюються;

– вони прості у використанні;

– їх неважко модифікувати відповідно до конкретних потреб.

В усіх розповсюджених операційних системах засоби шифрування файлів є вбудованими. Звичайно вони призначені для шифрування окремих файлів, і робота з ключами цілком покладається на користувача або на певний алгоритм. Тому застосування цих засобів вимагає особливої уваги: по-перше, ні в якому разі не можна зберігати ключі на диску разом із зашифрованими з їх допомогою файлами, а по-друге, незашифровані копії файлів необхідно стерти відразу ж після шифрування.

Еліптичні криві називаються так, тому що описуються кубічними рівняннями, які подібні до тих, що використовуються для обчислення кривої

еліпса. В загальному випадку кубічні рівняння для еліптичних кривих задаються формулою:

$$y^2 + axy + by = x^3 + cx^2 + dx + e, \qquad (1.1)$$

де *a, b, c, d, e* – є дійсними числами, які задовольняють деякі умови, *x, y* – координати точок.

Визначення еліптичної кривої також включає елемент, який позначається *O* і називається “нульовим елементом”, або “точкою на нескінченності” (“нескінченним елементом”). Такі рівняння називаються кубічними, або рівняннями третього порядку, тому що в них найвищий показник степені рівний 3. З характеристик еліптичної кривої можна вивести наступне: якщо три точки еліптичної кривої лежать на прямій лінії, то їх сума є *O* [6].

У випадку криптографії з використанням еліптичних кривих доводиться мати справу зі зміненою формою еліптичної кривої, яка визначається над скінченним полем. Особливий інтерес для криптографії представляє об’єкт, який називається еліптичною групою по модулю *p*, де *p* є простим числом [7].

В криптографії з використанням еліптичних кривих над скінченним полем всі значення обчислюються по вищезгаданому модулю *p*. Елементами даної кривої будуть пари невід’ємних цілих чисел, які менші *p* і задовольняють вигляду еліптичної кривої представленої у формулі:

$$y^2 \equiv x^3 + ax + b (\mathrm{mod}\ p). \qquad (1.2)$$

Таку криву будемо позначати $E_p(a,b)$. При цьому числа *a* і *b* повинні бути менші від *p* і задовольняти умову представлену у формулі:

$$4a^3 + 27b^2 (\mathrm{mod}\ p) \neq 0. \qquad (1.3)$$

Множина точок на еліптичній кривій обчислюється наступним чином:

– для кожного значення x, де $0 \le x < p$, обчислюється права частина формули (1.2);

– для кожного з отриманих на попередньому кроці значень обчислюється чи має це значення квадратний корінь по модулю p. Якщо ні, то в кривій $E_p(a,b)$ немає точок з цим значенням x. Якщо корінь існує, то маються два значення y, які відповідають операції вилучення квадратного кореня. Однак, виключенням є випадок, коли єдиним значенням буде $y=0$. Ці значення (x,y) і будуть точками на еліптичній кривій $Ep(a,b)$.

Множина точок $E_p(a,b)$ володіє наступними властивостями:

– точка $P=P+O$;

– якщо $P=(x,y)$, то $P+(x,-y)=O$. Точка $(x,-y)$ є від'ємним значенням точки P і позначається $-P$. Необхідно відмітити, що точка $(x,-y)$ лежить на еліптичній кривій і належить $E_p(a,b)$;

– якщо точки $P=(x_1,y_1)$ і $Q=(x_2,y_2)$, де $P \ne Q$, то $P+Q=(x_3,y_3)$ обчислюється за формулами:

$$x_3 \equiv \lambda^2 - x_1 - x_2 (\mathrm{mod}\, p), \qquad (1.4)$$

$$y_3 \equiv \lambda(x_1 - x_3) - y_1 (\mathrm{mod}\, p), \qquad (1.5)$$

де λ – кутовий коефіцієнт січної, проведеної через точки P і Q.

Він обчислюється так: якщо $P \ne Q$, то за формулою (1.6), якщо $P = Q$, то відповідно за формулою:

$$\lambda = (y_2 - y_1)/(x_2 - x_1), \qquad (1.6)$$

$$\lambda = (3x_1^2 + a)/2y_1. \qquad (1.7)$$

Основною арифметичною операцією в еліптичних кривих є операція скалярного множення точок кривої, яка дозволяє обчислити точку $Q = n * P$. Тобто точка *P*, помножена на ціле число n дає точку *Q*. Скалярне множення здійснюється послідовністю декількох комбінацій додавання і подвоєння точок еліптичної кривої.

Наприклад, представлення обчислення точки 11**P* показане у формулі:

$$11 * P = 2 * (2 * (2 * P) + P) + P . \qquad (1.8)$$

Надійність та криптостійкість еліптичної криптографії основана на складності рішення задачі ECDLP, суть якої полягає в задачі пошуку цілого числа *n* за відомими точками *P* і $Q = n * P$. Крім цього, важливим параметром кривої є базова (генеруюча) точка *G*, яка вибирається для кожної кривої окремо. Секретним ключем у відповідності до технології ЕК є велике випадкове число *n*, а відкритим ключем – множення *n* на базову точку *G*.

Три основних способи використання еліптичних кривих в криптографії:

- аналог алгоритму Діффі-Хеллмана обміну ключами;
- алгоритм цифрового підпису на основі еліптичних кривих;
- шифрування та дешифрування.

1.3 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* [8], визначеною над скінченним полем *K*, називається крива, описана рівнянням Вейєрштраса (1.9):

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \qquad (1.9)$$

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:

$$\Delta = -d_2^2 d_8 - 8d_4^3 - 27d_6^2 + 9d_2 d_4 d_6, \qquad (1.10)$$

де $d_2 = a_1^2 + 4a_2$, $d_4 = 2a_4 + a_1a_3$, $d_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2$, $d_6 = a_3^2 + 4a_6$.

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

$$y^2 = x^3 + ax^2 + b. \qquad (1.11)$$

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

$$2 + xy = x^3 + ax^2 + b, \qquad (1.12)$$

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 [9]:

– провести пряму через точки *P* та *Q*.

– відобразити перетин даної прямої з еліптичною кривою відносно осі *OY*.

Подвоєння точки P графічно визначається наступним чином:

– провести дотичну до еліптичної кривої в точці *P*.

– перетин даної дотичної, відображений симетрично осі *OY*, називається подвоєною точкою *P*.

1.4 Основні поняття про поля Галуа

Полем називають множину елементів, на якій визначено дві операції. Одна з них називається додаванням і позначається $a+b$, а інша – множенням і позначається ab, навіть якщо ці операції не є звичайними операціями додавання і множення чисел. Для того, щоб множина елементів, на якій задані операції додавання і множення, була полем, необхідно, щоб для кожної з цих операцій виконувалися всі групові аксіоми, а саме комутативність $a+b=b+a$ і $ab=ba$, асоціативність $a+(b+c)=(a+b)+c$ і $a(bc)=(ab)c$, а також виконувався дистрибутивний закон, тобто для трьох будь-яких елементів поля *a, b, c* була справедлива рівність $a(b+c)=ab+ac$ і $(b+c)a=ba+ca$.

Варто відмітити, що групові властивості для операції множення справедливі для всіх ненульових елементів поля.

Кількість елементів поля q називають порядком поля. Скінченні поля використовуються для побудови більшості відомих кодів і їх декодування.

Найменше число елементів, які утворюють поле, дорівнює 2. Таке поле повинне містити 2 одиничних елементи: 0 щодо операції додавання і 1 щодо операції множення. Це поле $GF(2)$, або двійкове.

Залежно від значення q розрізняють прості або розширені поля. Поле називають простим, якщо q – просте число. Для позначення простих чисел використовуватимемо символ p. Просте поле утворюють числа за модулем p:

$0,1,2,\ldots,(p-1)$, а операції додавання і множення виконуються за модулем *p*. Якщо ж поле утворене з q^m елементів, то таке поле називають розширенням поля ступеня *m* над $GF(p)$, або розширеним полем. Воно містить p^m елементів і позначається $GF\left(p^m\right)$. Надалі розглядатимемо поля $GF\left(2^m\right)$. Будь-яке скінченне поле $GF\left(2^m\right)$ є *m*-вимірним вектором над полем $GF(2)$. Многочлен $f(t)$ ступеня *m* над полем $GF(2)$ – це многочлен вигляду:

$$f(t)=t_m+f^{m-1}t_{m+1}+\ldots+f_0, \qquad (1.14)$$

де коефіцієнти многочлена *f* належать $GF(2)$. Операції над такими многочленами виконуються як операції над звичайними многочленами, тільки операції над коефіцієнтами виконуються в полі $GF(2)$.

Многочлен $f(t)$ ненульового ступеня називається незвідним над полем $GF\left(2^m\right)$, якщо він ділиться без залишку над цим полем на самого себе і на многочлени нульового ступеня. Елемент *x* скінченного поля $GF\left(2^m\right)$ називається коренем многочлена $f(t)$, якщо $f(x)=0$. Якщо *x* – корінь незвідного многочлена $p(x)$ ступеня *m*, то елементи $\left(x^{m+1},\ldots,1\right)$ утворюють базис скінченного поля $GF\left(2^m\right)$ як векторного простору над полем $GF(2)$. Такий базис називається поліноміальним. Многочлен, коренем якого є примітивний елемент, називається примітивним многочленом. Якщо як $P(x)$ вибрати примітивний многочлен ступеня *m*, незвідний над полем $GF(2)$ *m*, отримаємо поле $GF\left(2^m\right)$ зі всіх 2^m двійкових послідовностей довжини *m*. Поліноміальний базис задається примітивним многочленом.[11]

Елементи скінченного поля в поліноміальному базисі зображуються многочленами ступеня, не більшого за $m-1$ або, що еквівалентно, двійковими рядками довжини *m*, що складаються з коефіцієнтів таких многочленів. Операції додавання і множення у скінченному полі в такому разі – це операції над многочленами ступеня не вище за $m-1$ зі зведенням результату за потреби за модулем примітивного многочлена.

Додавання двох елементів виконується як додавання відповідних многочленів або як додавання за модулем 2 відповідних до цих многочленів двійкових рядків. Множення двох елементів виконується як множення відповідних многочленів з подальшим зведенням результатів за модулем примітивного многочлена.

Нормальним базисом поля $GF(2)$ є множина:

$$B=\left\{\theta,\theta^2,\theta^{2^2},\ldots,\theta^{2^{m-1}}\right\}. \qquad (1.15)$$

Елементи множини можна подати у вигляді двійкових рядків $(a_0a_1a_2\ldots a_{m-1})$:

$$a_0\theta+a_1\theta^2+a_2\theta^{2^2}+\ldots+a_{m-1}\theta^{2^{m-1}}. \qquad (1.16)$$

Залежно від параметра *T* розрізняють типи нормального базису і нормальні поліноми:

– якщо $T=1$: $p(x)=t^m+t^{m+1}+\ldots+t^2+t+1$;

– якщо $T=2$: $p_0(t)=1, p_1(t)=t+1, p_{i+1}(t)=tp_i(t)+p_{i-1}(t)$, $i=1,\ldots,m$.

Висновки до розділу

В даному розділі було розглянуто актуальність криптографії з використанням еліптичних кривих. Виявлено переваги й недоліки використання математичного апарату еліптичних кривих в задачах шифрування даних, розподілу ключів та використання електронно-цифрового підпису. Так, до переваг криптографії на ЕК можна віднести:

– значно менший розмір ключа в порівнянні з традиційними криптосистемами;

– безпека систем захисту інформації, побудованих на еліптичних кривих, ґрунтується на задачі дискретного логарифмування, що є значно складнішою, ніж задача розкладення числа на множники (основа системи RSA).

Проте використання ЕК в шифруванні даних є не досить вигідним, оскільки дуже багато часу займає саме розшифрування даних. Саме з цієї причини на даний час ЕК в основному використовуються в системах ЕЦП (де в принципі не використовується розшифрування даних) та в процедурах розповсюдження ключів, де не потрібно розшифровувати великі об’єми інформації. Також суттєвою проблемою криптографії ЕК є генерація параметрів кривої, які б забезпечили необхідну криптографічну стійкість системи.

2 ПАРАМЕТРИ АЛГОРИТМУ ECDSA ТА ТЕХНОЛОГІЯ ЇХ ВИБОРУ

Сьогодні криптосистеми на еліптичних кривих використовуються в TLS, PGP і SSH, найважливіших технологіях, на яких базуються сучасний Інтернет та світ IT, не кажучи про Bitcoin та інші криптовалюти, які базуються на основі цифрового підпису ECDSA.

В даному розділі розглядається технологія вибору параметрів ECDSA. У пункті 2.1 розглянуті основні параметри та алгоритм цифрового підпису ECDSA. У пунктах 2.2–2.4 досліджені основні вимоги до параметрів еліптичної кривої, генераторів псевдовипадкових чисел та криптографічних хеш-функцій.

2.1 Алгоритм ECDSA та його параметри

ECDSA (Elliptic Curve Digital Signature Algorithm) – це криптографічно захищена схема цифрового підпису, заснована на криптографії еліптичної кривої (ECC). ECDSA спирається на математику циклічних груп еліптичних кривих над кінцевими полями та на складність проблеми ECDLP (проблема дискретного логарифмування еліптичної кривої). Алгоритм підписання/перевірки ECDSA базується на множенні точок ЕК і працює, як описано нижче. Ключі та підписи ECDSA коротші, ніж у RSA, для того самого рівня безпеки. 256-бітний підпис ECDSA має таку саму надійність, як і 3072-бітний підпис RSA.

ECDSA використовує криптографічні еліптичні криві (ЕК) над кінцевими полями в класичній формі Вейєрштраса. Ці криві описуються параметрами домену ЕК, визначеними різними криптографічними стандартами, такими як SECG: SEC 2 і Brainpool (RFC 5639).

Пройдемося по необхідним для алгоритму змінним:

I. Список “глобальних” змінних, визначених стандартом:

1. Основні параметри еліптичної кривої (числа *a*, *b*, *p*)
2. Точка *GP*, яка лежить на цій кривій (її *x* та *y* координати)
3. Обчислений порядок *n* точки *GP*. Як ми знаємо, *n* – властивість точки *GP*, таке, що $GP \cdot (n+1) = GP$, $GP \cdot (n+2) = GP \cdot 2$, тощо. Як модуль.

II. Змінні, які належать комусь конкретному:
1. *sk* (секретний ключ) – будь-яке довільне число в діапазоні $sk \in [1; n-1]$. Тримається у секреті його власником.
2. *pk* (публічний ключ) = $GP \cdot sk$. Відомий усім.

III. Змінні, які використовуються під час підпису повідомлення:
1. *m* (повідомлення) – будь-яке ціле число, що не перевищує *n*. Зазвичай використовується хеш від рядка.
2. *k* – ціле ціле число, яке генерується для одного підпису повідомлення $k \in [1; n-1]$. Тримається у секреті тим, хто підписує повідомлення. Якщо хтось його впізнає – з легкістю вирахує приватний ключ із підпису!

Алгоритм ECDSA можна поділити на 3 частини:

- створення ключів;
- створення підпису;
- перевірка підпису.

2.1.1 Алгоритм створення ключів

Секретний ключ *sk* вибирається випадково від 1 до *n* – 1. Як публічний ключ *pk* береться точка $pk = sk \cdot GP$.

Тут набирає чинності ще одна ключова властивість операції множення точки на число. Нехай дані точки *G* і $d \cdot G$, де *d* невідомо. Тоді виявляється, що

знайти *d* – це нерозв'язне з обчислювальної точки зору завдання. Цей факт називається проблемою дискретного логарифмування для ЕК.

Звідси випливає, що, знаючи базову точку *GP* та публічний ключ *pk*, ми ніяк не зможемо знайти відповідний секретний ключ *sk*.

У цьому, оскільки порядок точки *GP* дорівнює *n*, вона має всього *n* – 1 можливих позицій на кривою, отже, для публічного ключа існує всього *n* – 1 можливих значень.

На виході алгоритм видає пару *sk*, *pk*, де *sk* – число, а *pk* – точка, тобто пара чисел.

2.1.2 Алгоритм створення підпису

Алгоритм генерації підпису (*Sig*) приймає на вхід приватний ключ *sk* і число *m* – хеш повідомлення, що підписується.

А на виході як підпис видається не одне число, а два: (*r*, *s*) = *Sig* (*sk*, *m*). Чому два, а не одне?

Це пов'язано з тим, що всередині самого алгоритму є вибір додаткового випадкового числа *k*, що впливає на вигляд підпису. Цей випадковий параметр потрібен для того, щоб забезпечити секретність ключа *sk*. Якщо підпис повністю буде визначатися лише вхідними параметрами *sk* та *m*, то секретний ключ *sk* може бути обчислений на основі лише двох залишених за його допомогою підписів.

Наявність випадкового параметра призводить до того, що на тих самих вхідних значеннях алгоритм *Sig* може видавати різний результат. Тому щоб підпис можна було перевірити, необхідно поділитися деякою інформацією про параметр *k*. Перша компонента підпису *r* містить у собі цю інформацію. Число *r* повністю визначається числом *k* при його підрахунку *sk* і *m* не використовуються. У той же час у обчисленнях числа *s* беруть участь уже всі три параметри *k*, *sk* та *m*.

Алгоритм генерації підпису:

1. Беремо випадкове ціле k, $k \in [1; n-1]$.
2. Обчислюємо точку $P = k \cdot GP$.
3. Обчислюємо число $r = x \bmod n$ (де x – це координата x точки P).
4. Якщо $r = 0$, то вибираємо інше k і пробуємо знову.
5. Обчислюємо $s = k^{-1} (m + r \cdot sk) \bmod n$ (де k^{-1} – мультиплікативна інверсія k по модулю n).
6. Якщо $s = 0$, то вибираємо інше k і пробуємо знову.

Пара (r, s) є підписом.

2.1.3 Алгоритм перевірки підпису

Алгоритм перевірки підпису приймає на вхід публічний ключ *pk*, хеш повідомлення m та підпис (r, s).

На основі підпису та хешу цей алгоритм знаходить два числа u_1 та u_2. Далі розглядаються дві точки на еліптичній кривій: $u_1 \cdot GP$ та $u_2 \cdot pk$.

Якщо підпис s був дійсно порахований за допомогою ключа *sk*, відповідного ключу *pk*, і повідомлення дійсно не змінювалося з моменту створення підпису, то точки $u_1 \cdot GP$ та $u_2 \cdot pk$ будуть відносно один одного в деякому спеціальному положенні. Алгоритм дивиться, так це чи ні. Якщо розташування точок щодо один одного і справді таке, яке має бути, то підпис вважається коректним, і алгоритм повертає 1 (*TRUE*). В іншому випадку повертається 0 (*FALSE*).

Алгоритм перевірки підпису:

1. Обчислюємо ціле $u_1 = s^{-1} \cdot m \bmod n$.
2. Обчислюємо ціле $u_2 = s^{-1}\, r \bmod n$.
3. Обчислюємо точку $P = u_1 \cdot GP + u_2 \cdot pk$.

Підпис дійсний, лише якщо r = $x \bmod n$ (де x – це координата x точки P).

2.2 Вимоги до параметрів еліптичної кривої

2.2.1 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:

– розмірність поля p , де p є простим числом;

– два елементи скінченного поля – a та b , визначені рівнянням еліптичної кривої *E*, що має наступний вигляд:

$$y^2 = x^3 + ax + b, \tag{1.13}$$

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

– два елементи поля $GF(p)$ – x_G та y_G, які визначають кінцеву точку $G = (x_G, y_G)$ генератор групи;

– порядок q точки G , де $q > 2^{160}$ та $q > 4\sqrt{p}$;

– співмножник $h = \#E/q$, де *#E* означає порядок групи точок ЕК.

2.2.2 Генерація основних параметрів

Один із способів генерації криптографічно надійних параметрів полягає в наступному:

– обираємо коефіцієнти *a* та *b* специфічним чином, використовуючи в обчисленнях випадкові/псевдовипадкові числа. Нехай E – еліптична крива $y^2 = x^3 + ax + b$ [10].

– обчислюємо $N = \#E$

– перевіряємо, що N має дільник, який є великим простим числом $q\left(q > 2^{160}, q > 4\sqrt{p}\right)$. Якщо ця умова не виконується, то необхідно повернутись на крок 1.

– 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.

– перевірити, що $q \neq p$. Якщо не виконується, то необхідно повернутись на крок 1.

– обрати випадкову точку $G' \in E$ і покласти $G = (N/q)G'$. Повторювати, доки $G \neq O$.

2.3 Вимоги до генераторів псевдовипадкових чисел

Оскільки збільшується передача даних через загальні і приватні мережі, стає все більш важливим захист інформації яка зберігається і передається між комп'ютерами. Один із стандартних блоків безпеки є генератор випадкових чисел (ГВЧ).

Проблема захисту інформації є багатогранна і вирішується комплексно, з використанням великої кількості способів.

Випадкові числа – фундаментальний елемент для надання обмеженого доступу до інформації. Вони являють собою основний елемент криптографії, цифрового підпису, протоколів безпеки і іншого забезпечення надійності при зв'язку.

В галузі захисту інформації існує окремий напрям, пов’язаний з генерацією випадкових (псевдовипадкових) послідовностей, йде постійна робота по удосконаленню не тільки засобів генерації, але і теорії та термінології в цьому важливому напрямі, проводиться розробка теорії і практики тестування джерел інформаційно-телекомунікаційних мереж випадкових послідовностей, оцінки і вимірювання їх показників.

Найчастіше генератори псевдовипадкових чисел (ГПВЧ) застосовують в криптографії. Випадковість і криптографія дуже сильно взаємопов'язані. Важко знайти добре розроблене криптографічне прикладне забезпечення, яке не використовує випадкові числа. Криптографічні ключі, їх ініціалізація, тонкощі хешування з паролями, унікальні параметри в операціях цифрового підпису системними розробниками повинні прийматися випадковими. ГПВЧ є криптографічно сильним, якщо послідовність, яку він генерує з короткого таємного вихідного ключа, є майже такою самою, як і справжня випадкова послідовність і ніяке практично легко здійснюване обчислення не може дозволити криптоаналітику отримати яку-небудь інформацію про відкритий текст при перехопленні ним шифротексту (за виключенням хіба що дуже малої ймовірності). Для застосування в криптографічних системах ГПВЧ повинні відповідати наступним вимогам:

- послідовність, що генерується повинна мати максимально великий період;
- послідовність, що генерується не повинна мати схованих періодичностей;
- послідовність, що генерується повинна мати рівномірний спектр.

Найважливішою характеристикою ГПВЧ є довжина періоду повторення, після якого випадкові числа, на виході ГПВЧ почнуть повторюватися. Другою за важливістю характеристикою ГПВЧ є його продуктивність, тобто кількість чисел, які генеруються за одиницю часу. Для окремих прикладних програм (статистичне зондування, моделювання в реальному часі і т.д.) може бути потрібною продуктивністю порядку 10^{10} – 10^{12} випадкових чисел за секунду.

2.4 Вимоги до криптографічних хеш-функцій

Криптографічна хеш-функція — це хеш-функція, яка є алгоритмом, що приймає довільний блок даних і повертає рядок встановленого розміру, (криптографічне) хеш-значення, таке що (випадкові або навмисні) зміни даних (з дуже високою ймовірністю) змінять хеш-значення. Дані до кодування часто звуть «повідомлення», а хеш-значення іноді називають дайджест повідомлення (англ. messagedigest) або просто дайджест, дослівно «стислий виклад».

Ідеальна криптографічна хеш-функція має чотири основні або значимі властивості:

– легкість обчислення хеш-значення для будь-якого повідомлення;

– нездійсненно утворити повідомлення для заданого хеш-значення;

– нездійсненно змінити повідомлення без зміни хешу;

– нездійсненно знайти два різних повідомлення з тим самим хешем.

Більшість криптографічних хеш-функцій спроектовано так, щоб на вхід подавався рядок довільної довжини, а на виході ми отримували хеш-значення встановленої довжини.

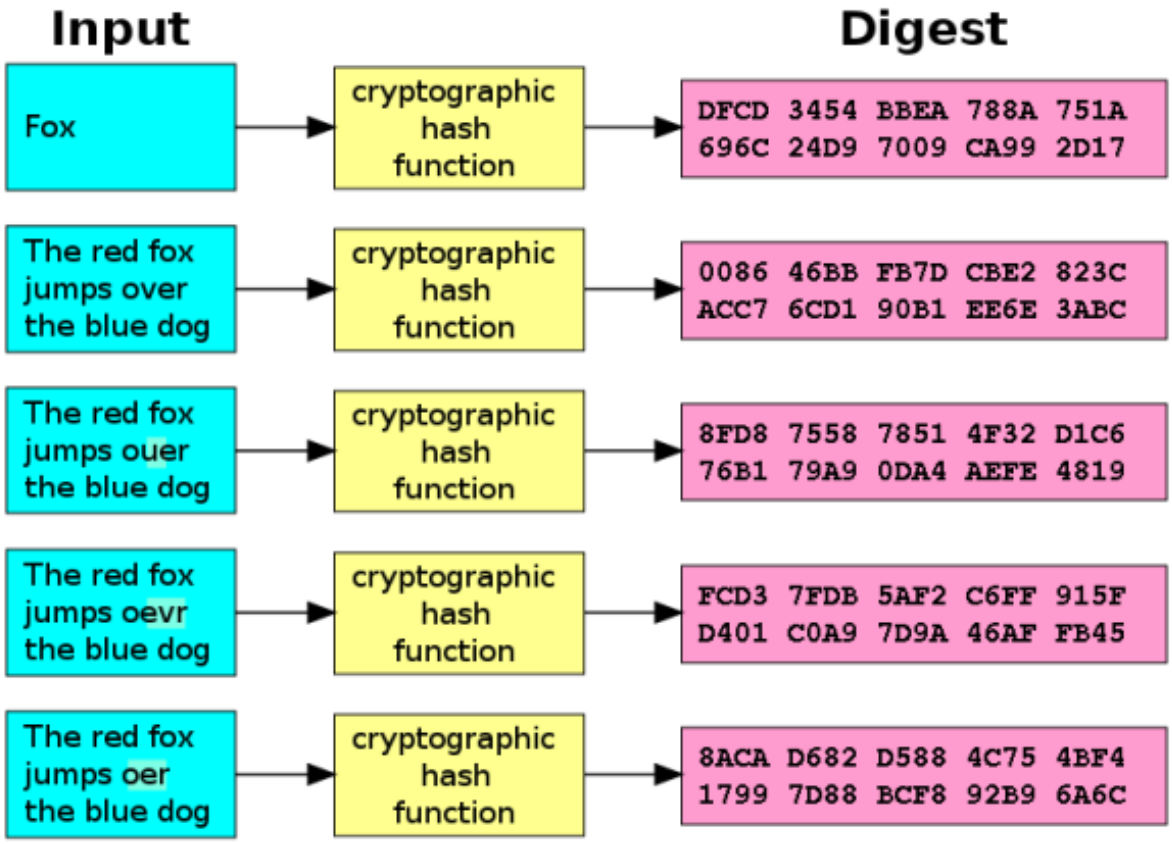


Рис. 2.1.Криптографічна хеш-функція (а саме, SHA-1) в роботі. Зауважте, що навіть мала зміна даних на вході (тут в слові «over») значно змінює вихід, через так званий лавиновий ефект.

Криптографічна хеш-функція повинна витримувати всі відомі типи криптографічних атак. Щонайменше, вона повинна мати наступні властивості:

– *Першовзорова стійкість*. Для певного хешу h має бути складно знайти повідомлення m, таке що $h = hash(m)$. Ця концепція стосується односторонньої функції. Функції, що не відповідають цій властивості вразливі до атак знаходження першовзору.

– *Друга першовзорова стійкість*. Для певного входу m_1 має бути складно знайти інший вхід m_2 – де $m_1 \neq m_2$, такий що $hash(m_1) \neq hash(m_2)$. Цю властивість іноді називають слабка колізійна стійкість (англ. weak collision resistance), і функцій, що не мають цієї властивості вразливі до атак знаходження першовзору другого роду.

– *Колізійна стійкість*. Повинно бути складно знайти два різних повідомлення m_1 і m_2, таких що $hash(m_1) = hash(m_2)$. Така двійка зветься криптографічна хеш-колізія. Цю властивість іноді називають сильна колізійна стійкість (англ. strong collision resistance). Воно вимагає щонайменше вдвічі довшого хеш-значення ніж потрібно для першовзорової стійкості, інакше колізії можна знайти за допомогою атаки «днів народження».

Ці властивості мають на увазі, що зловмисник не може змінити вхідні дані без зміни дайджеста. Отже, якщо два рядки мають однаковий дайджест можна бути впевненим, що й самі вони такі.

Висновки до розділу

В даному розділі були розглянуті функціонування алгоритму цифрового підпису ECDSA (Elliptic Curve Digital Signature Algorithm) та основні вимоги до його параметрів. Визначено, що ECDSA спирається на математику циклічних груп еліптичних кривих над кінцевими полями та на складність проблеми ECDLP (проблема дискретного логарифмування еліптичної кривої).

Проаналізовані основі вимоги до параметрів еліптичної кривої, генераторів псевдовипадкових чисел та криптографічних хеш-функцій, на основі яких працює ECDSA.

Таким чином, можна зробити висновок, що для належного рівня криптостійкості при використанні алгоритму цифрового підпису ECDSA першочергово потрібно:

– підібрати параметри еліптичної кривої таким чином, щоб вони задовольняли рівняння $4a^3 + 27b^2 \neq 0 (\mathrm{mod}\ p)$, де p – просте та велике.

– вибрати n, де $n > 2^{160}$ та $n > 4\sqrt{p}$;

– використовувати генератор псевдовипадкових чисел з максимально великим періодом;

– використовувати тільки ті криптографічні хеш-функції, за допомогою яких нездійсненно утворити повідомлення для заданого хеш-значення.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ ECDSA

Для програмної реалізації алгоритму ECDSA необхідно було провести ґрунтовне дослідження програмних засобів, підходів та комплексів, які б забезпечили найбільш оптимальні можливості для успішної їх практичної реалізації.

В даному розділі проводиться короткий опис трьох різних мов програмування для подальшого вибору кращого з них. Представляється опис функцій та алгоритму створеної програми.

3.1 Вибір мови програмування

В усіх асиметричних криптосистемах найважливішою складовою є секретний та відкритий ключі, за допомогою яких і відбувається шифрування/дешифрування, створення та перевірка ЕЦП. Ці ключі мають бути досить великих розмірів (наприклад, по 1024 біт кожен [17]) 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 $GF\left(2^m\right)$, 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 спільного, як скажімо, однакові структури викликів операцій над елементами. Так і ЕК з параметрами, визначеними над різними типами скінченних полів мають за своєю суттю абсолютно однаковий шаблон для викликів операцій над їх точками [18].

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

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:

- підтримка об'єктно-орієнтованого програмування через класи;
- підтримка узагальненого програмування через шаблони;
- доповнення до стандартної бібліотеки;
- додаткові типи даних;
- обробка винятків;
- простори імен;
- вбудовані функції;
- перевантаження операторів;

– перевантаження імен функцій;

– посилання і оператори управління вільно розподіленою пам'яттю.

До стандарту C++ входить також Стандартна Бібліотека Шаблонів (STL) [19]. 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 безпосереднього виділення та управління пам'яттю. Це дозволяє створювати нові структури даних, що можуть задовольнити вимоги до такого ключового елемента нашого програмного комплексу як елемент скінченного поля.

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 [20].

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 представляти параметри еліптичної кривої.

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:

– чистий та інтуїтивно зрозумілий синтаксис;

– портабельність програм пов’язана з інтерпретативністю;

– наявність вбудованих модулів, шо надають можливість використовувати колекції, особливі структури даних, математичний пакет тощо;

– можливість програмування на Python в скрипковому режимі (з консолі);

– 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 [21]. 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 C++ 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.

3.2 Архітектура програмного комплексу

Розроблений програмний комплекс умовно можна розділити на три основні частини:

- введення та перевірка необхідних для алгоритму ECDSA параметрів;
- створення цифрового підпису;

– перевірка цифрового підпису.

3.3 Опис логічної структури

Програма використовує такі бібліотеки та заголовні файли як:

– iostream – бібліотека і відповідний заголовний файл, які використовується для організації введення-виведення в мові програмування C++;

– math.h – заголовний файл стандартної бібліотеки мови програмування C, який надає прототипи функцій, розроблених для виконання простих математичних операцій;

– Windows.h – це специфічний для Windows заголовний файл для мов програмування C і C ++, який містить декларації для всіх функцій в Windows API, усіх загальних макросів, що використовуються програмістами Windows, і всіх типів даних, що використовуються різними функціями та підсистемами.

– time.h – заголовний файл стандартної бібліотеки мови програмування C, що містить типи і функції для роботи з датою й часом.

У програмі містяться наступні функції:

а) функція int Prime (int p) – виконує перевірку вхідного числа на простоту;

б) функція int inv (int a, int p) – виконує пошук оберненого числа по модулю за допомогою алгоритму Евкліда;

в) функція int SumX (int x1, int y1, int x2, int y2, int p) – обчислює координату *x* при додаванні двох точок на еліптичній кривій;

г) функція int SumY (int x1, int y1, int x2, int y2, int p, int x) – обчислює координату *y* при додаванні двох точок на еліптичній кривій;

д) функція int doublX (int x1, int y1, int p, int a) – обчислює координату *x* при подвоєнні точки на еліптичній кривій;

е) функція int doublY (int x1, int y1, int p, int a, int x) – обчислює координату *y* при подвоєнні точки на еліптичній кривій;

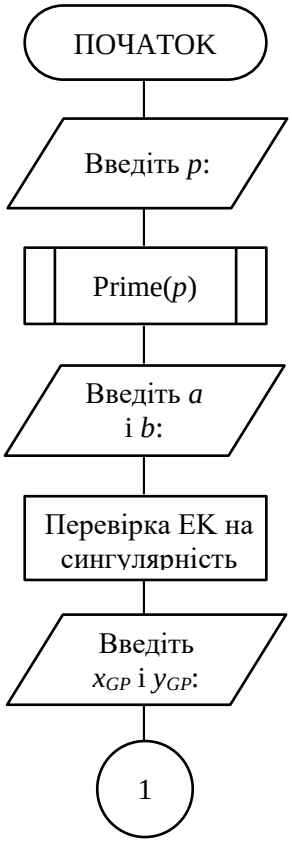
ж) функція int DecToBin (int a, int *bin) – реалізовує перетворення числа з десяткової системи числення у двійкову;

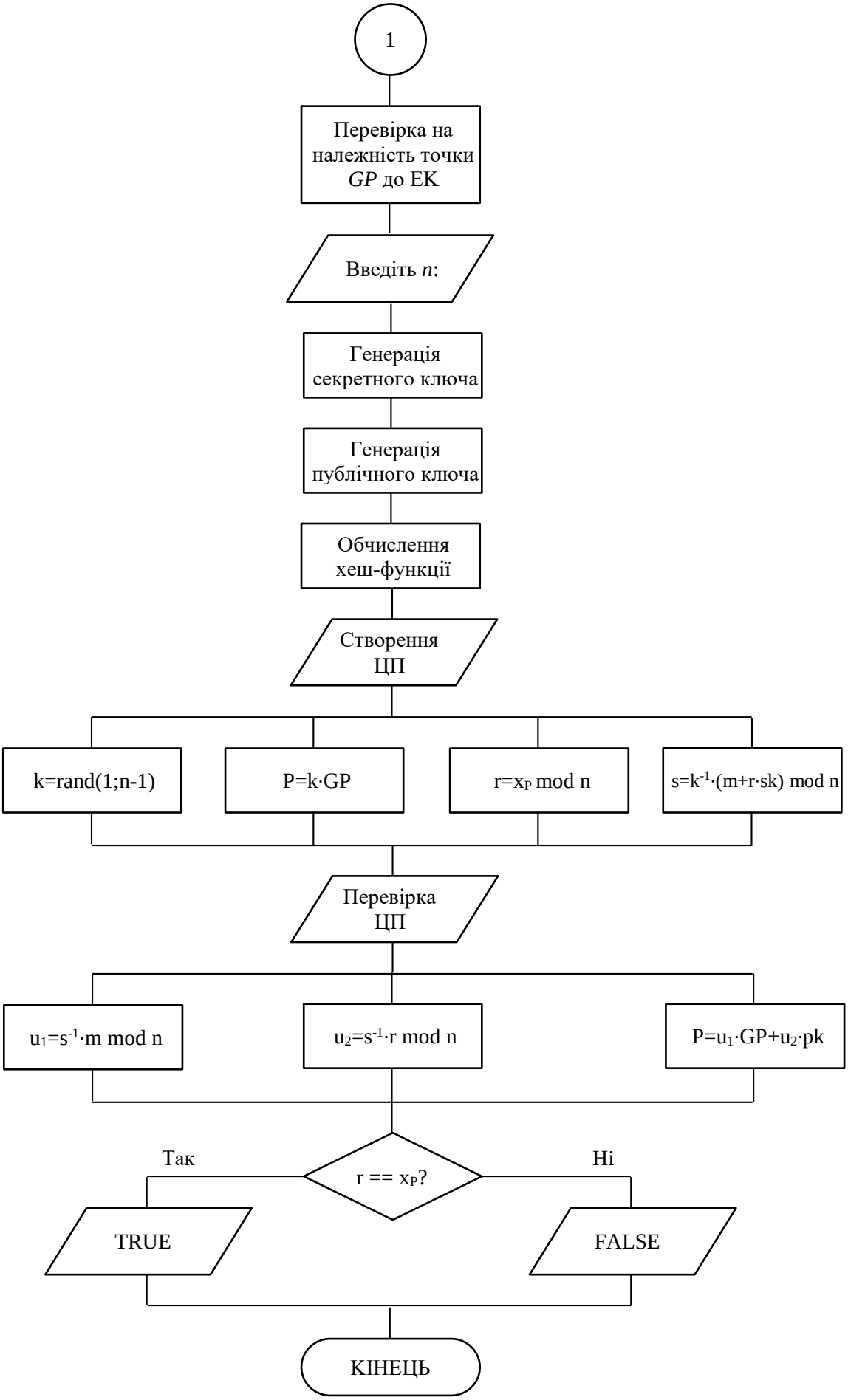
з) функція int Mult(int *g, int k, int x, int y, int a, int p) – реалізовує скалярне множення точки на константу за допомогою алгоритму подвоєння-додавання;

и) функція int main() – здійснює ввід вхідних параметрів та перевірку їх згідно поставлених критеріїв, поетапно демонструє реалізацію алгоритму ECDSA.

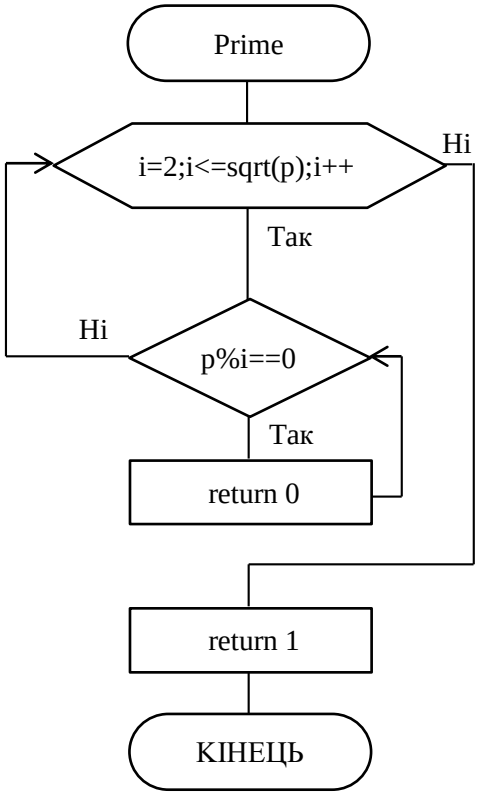
3.4 Блок – схема алгоритму рішення задачі

Загальний алгоритм програми:

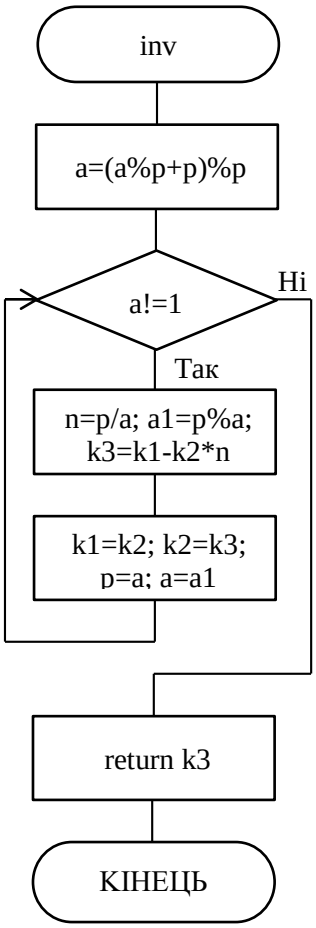



1
Перевірка на належність точки GP до ЕК
Введіть n:
Генерація секретного ключа
Генерація публічного ключа
Обчислення хеш-функції
Створення ЦП
k=rand(1;n-1)
P=k·GP
r=xP mod n
s=k-1·(m+r·sk) mod n
Перевірка ЦП
u1=s-1·m mod n
u2=s-1·r mod n
P=u1·GP+u2·pk
Так
r == xP?
Ні
TRUE
FALSE
КІНЕЦЬ

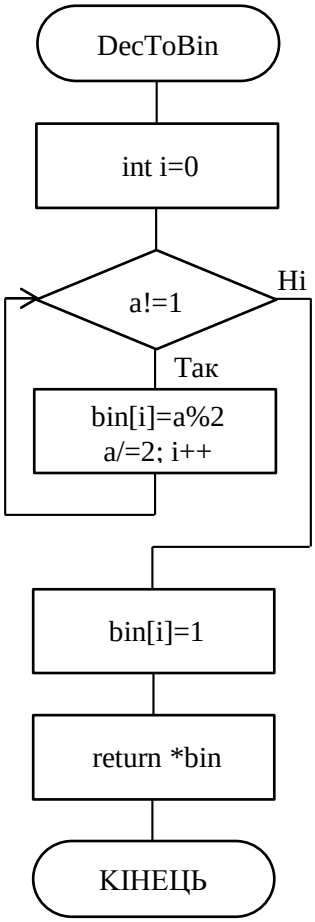
Функція перевірки на простоту:



Функція пошуку оберненого числа по модулю за допомогою алгоритму Евкліда:



Функція, яка перетворює число з десяткової системи числення у двійкову:



3.5 Числові експерименти

Дивлячись на архітектуру програмного комплексу, проведемо перевірку кожної з трьох основних частин програми.

Першою частиною є введення параметрів еліптичної кривої. На даному етапі виконується перевірка модуля *p* на простоту та перевірка еліптичної кривої на сигнулярність (параметри *a* і *b*).

Дослід 1. Нехай для початку *p*=20. Так впевнимося, що функція перевірки числа на простоту виконується правильно. Далі введемо просте *p*, наприклад, 23 (Рис. 3.1).

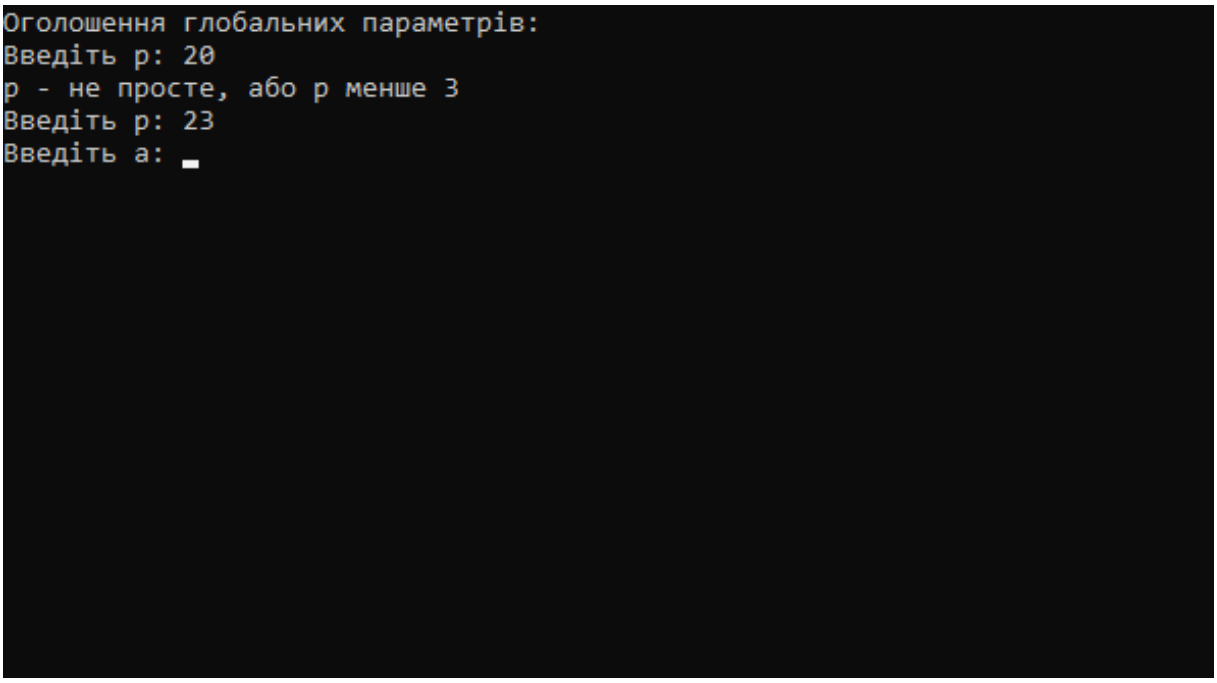


Рисунок 3.1 – Перевірка модуля *p* на простоту

Дослід 2. Наступним кроком потрібно ввести параметри *a* і *b*. Введемо їх таким чином, щоб отримати сингулярну ЕК ($a = -3, b = 2$). Далі введемо *a* і *b* так, щоб вони задовольняли правилу сингулярності ЕК (Рис. 3.2).

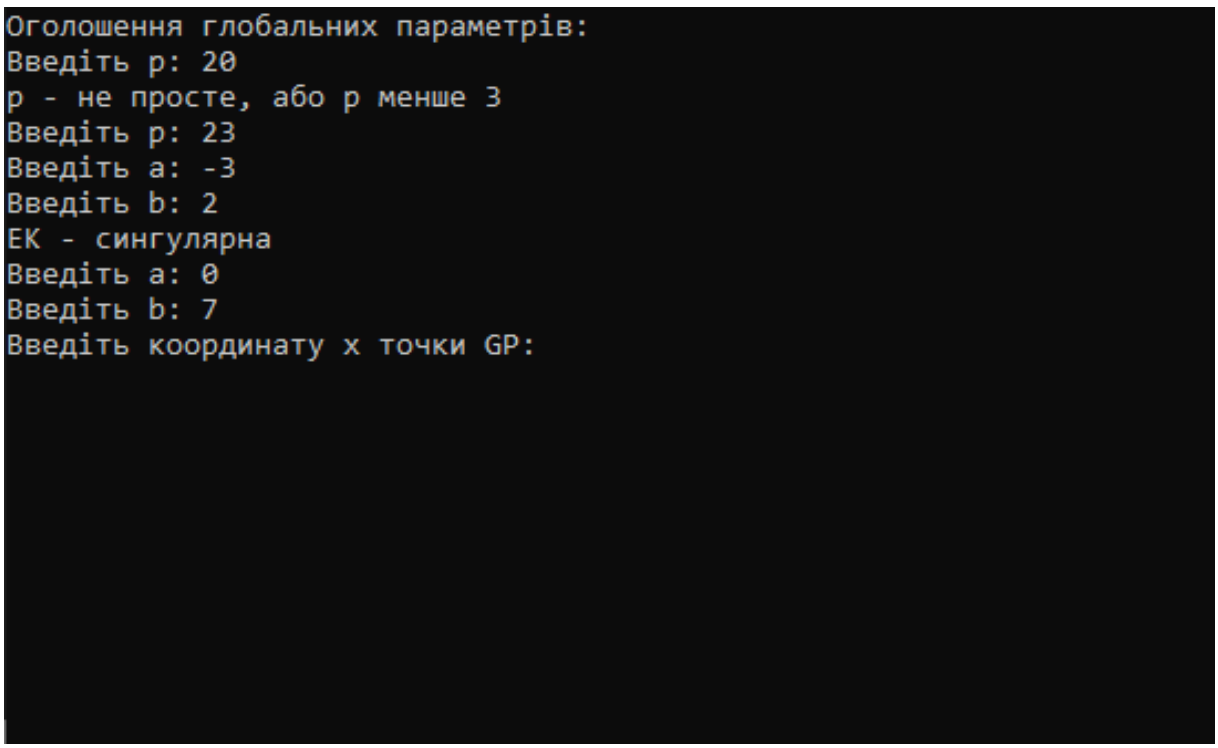


Рисунок 3.2 – Перевірка ЕК на сингулярність

Дослід 3. Далі потрібно ввести базову точку *GP*. На даному етапі відбувається перевірка приналежності точки *GP* до заданої ЕК та виведення порядку базової точки. Спочатку введемо точку, яка не належить ЕК (Рис. 3.3).

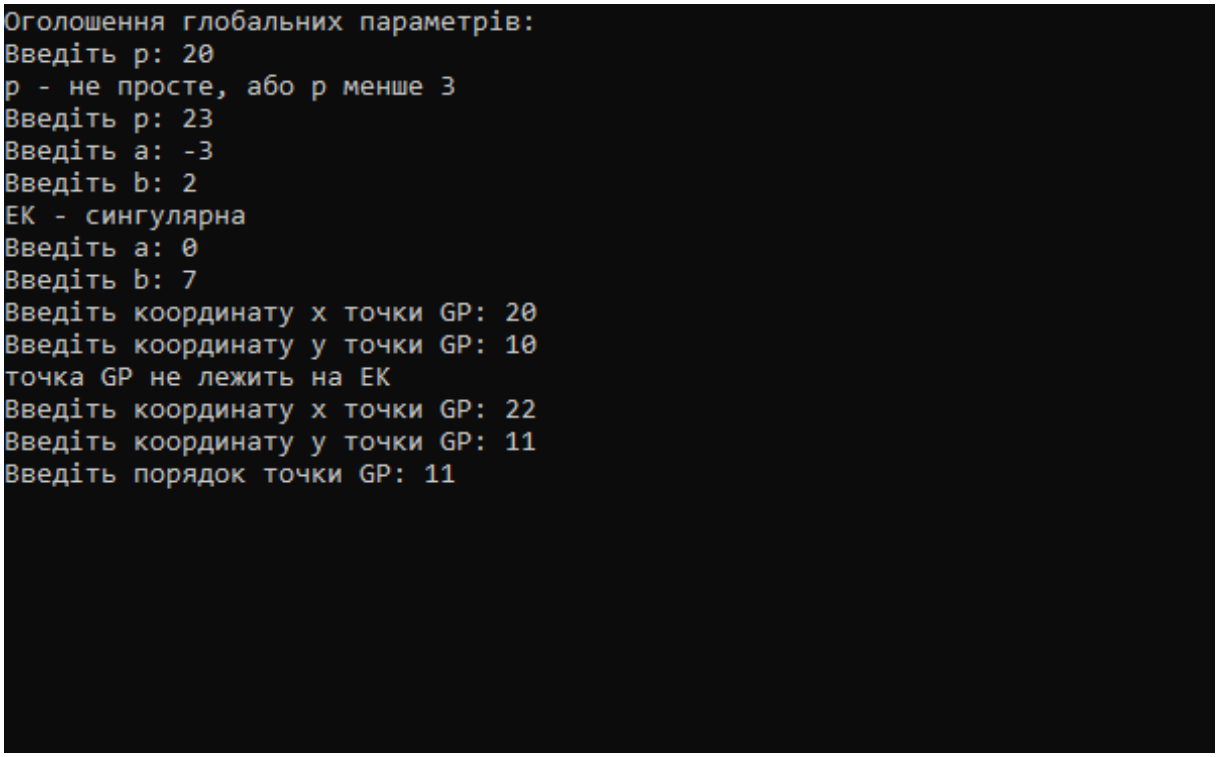


Рисунок 3.3 – Ввід та перевірка належності базової точки до заданої ЕК

Дослід 4. Після оголошення «глобальних» параметрів, які потрібні для алгоритму ECDSA, Аліса починає генерувати секретний та публічні ключі та обраховувати хеш-функцію свого повідомлення (Рис. 3.4).

```
Оголошення глобальних параметрів:
Введіть p: 20
p - не просте, або p менше 3
Введіть p: 23
Введіть a: -3
Введіть b: 2
EK - сингулярна
Введіть a: 0
Введіть b: 7
Введіть координату x точки GP: 22
Введіть координату y точки GP: 11
Введіть порядок точки GP: 11

Аліса генерує:
Секретний ключ: 10
Публічний ключ: (11;21)
Хеш-функція повідомлення: 9
```

Рисунок 3.4 – Генерація секретного та публічного ключів

Дослід 5. Після генерації ключів Аліса може приступати до створення цифрового підпису (Рис 3.6)

```
Оголошення глобальних параметрів:
Введіть p: 20
p - не просте, або p менше 3
Введіть p: 23
Введіть a: -3
Введіть b: 2
EK - сингулярна
Введіть a: 0
Введіть b: 7
Введіть координату x точки GP: 22
Введіть координату y точки GP: 11
Введіть порядок точки GP: 11

Аліса генерує:
Секретний ключ: 10
Публічний ключ: (11;21)
Хеш-функція повідомлення: 9

Аліса створює цифровий підпис: (4;3)
```

Рисунок 3.5 – Створення цифрового підпису згідно відповідного алгоритму

Дослід 6. Останнім етап відбувається обмін незахищеними каналами зв’язку відкритою інформацією та перевірки Бобом цифрового підпису Аліси (Рис. 3.5).

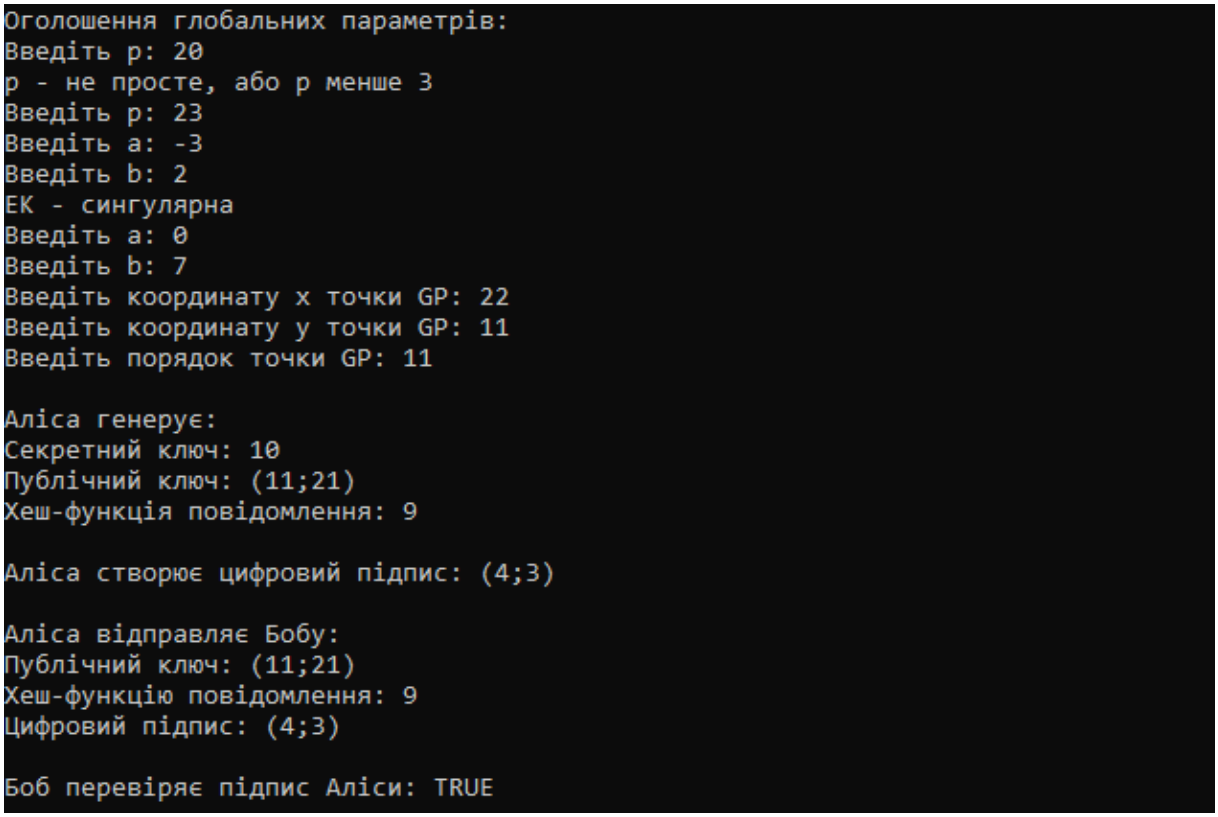


Рисунок 3.6 – Перевірка цифрового підпису згідно відповідного алгоритму

Дослід 7. Перевіримо даний алгоритм ще для двох наборів параметрів. У першому випадку візьмемо p=19, a=–7, b=10, x_{GP}=2, y_{GP}=17, n=7 (Рис 3.7). Для другого досліду візьмемо p=29, a=1, b=6, x_{GP}=10, y_{GP}=1, n=19 (Рис 3.8).

```
Оголошення глобальних параметрів:
Введіть p: 19
Введіть a: -7
Введіть b: 10
Введіть координату x точки GP: 2
Введіть координату y точки GP: 17
Введіть порядок точки GP: 7

Аліса генерує:
Секретний ключ: 1
Публічний ключ: (2;17)
Хеш-функція повідомлення: 4

Аліса створює цифровий підпис: (1;4)

Аліса відправляє Бобу:
Публічний ключ: (2;17)
Хеш-функцію повідомлення: 4
Цифровий підпис: (1;4)

Боб перевіряє підпис Аліси: TRUE
```

Рисунок 3.7 – Додаткова перевірка №1

```
Оголошення глобальних параметрів:
Введіть p: 29
Введіть a: 1
Введіть b: 6
Введіть координату x точки GP: 10
Введіть координату y точки GP: 1
Введіть порядок точки GP: 19

Аліса генерує:
Секретний ключ: 18
Публічний ключ: (10;28)
Хеш-функція повідомлення: 18

Аліса створює цифровий підпис: (3;18)

Аліса відправляє Бобу:
Публічний ключ: (10;28)
Хеш-функцію повідомлення: 18
Цифровий підпис: (3;18)

Боб перевіряє підпис Аліси: TRUE
```

Рисунок 3.8 – Додаткова перевірка №2

В наведеній нижче табл. 3.1 продемонстроване узагальнення отриманих результатів.

Таблиця 3.1

Тести програми

Дослід №	**Вхідні дані**	**Результати програми**	**Висновок**
1.	$p = 20$	p - не просте, або p менше 3	Правильно
2.	$p = 23$ $a = -3$ $b = 2$	ЕК - сингулярна	Правильно
3.	$p = 23$ $a = 0$ $b = 7$ $x_{GP} = 20$ $y_{GP} = 10$	Точка *GP* не лежить на ЕК	Правильно
4.	$p = 23$ $a = 0$ $b = 7$ $x_{GP} = 22$ $y_{GP} = 11$	$sk = 10$ $pk = (11; 21)$	Правильно
5.	$p = 23$ $a = 0$ $b = 7$ $x_{GP} = 22$ $y_{GP} = 11$	$(r; s) = (4; 3)$	Правильно
6.	$p = 23$ $a = 0$ $b = 7$ $x_{GP} = 22$ $y_{GP} = 11$	TRUE	Правильно
7.	$p = 19$ $a = -7$ $b = 10$ $x_{GP} = 4$ $y_{GP} = 36$	$(r; s) = (1; 4)$ => TRUE	Правильно
	$p = 29$ $a = 1$ $b = 6$ $x_{GP} = 10$ $y_{GP} = 1$	$(r; s) = (3; 18)$ => TRUE	Правильно

Висновки до розділу

В даному розділі було проаналізовано різні мови програмування та зроблено висновок, що з огляду на особливості мов програмування, можливості, які вони надають, та потенційну ефективність розробки, варто обрати мову програмування C++.

Данною мовою була розроблена програма, що реалізовує алгоритм ECDSA. Були розглянуті алгоритм та функції, які використовуються в даній програмі.

Зроблено ряд дослідів, які підтверджують, що програма працює без помилок та правильно реагує на різні випадки.

ВИСНОВКИ

Виходячи зі змісту та завдань нашої роботи сформулюємо наступні висновки:

По-перше, розглянуто основні задачі криптографії з використанням еліптичних кривих. Виявлено переваги й недоліки використання математичного апарату еліптичних кривих в задачах шифрування даних, розподілу ключів та використання електронно-цифрового підпису. Так, до переваг криптографії ЕК можна віднести:

– значно менший розмір ключа в порівнянні з традиційними криптосистемами;

– безпека систем захисту інформації, побудованих на еліптичних кривих, ґрунтується на задачі дискретного логарифмування, що є значно складнішою, ніж задача розкладення числа на множники (основа системи RSA)

Проте використання ЕК в шифруванні даних є не досить вигідним, оскільки дуже багато часу займає саме розшифрування даних. Саме з цієї причини на даний час ЕК в основному використовуються в системах ЕЦП та в процедурах розповсюдження ключів, де не потрібно розшифровувати великі обсяги інформації. Також суттєвою проблемою криптографії ЕК є генерація параметрів кривої, які б забезпечили необхідну криптографічну стійкість системи.

По-друге, розглянуте функціонування алгоритму цифрового підпису ECDSA (Elliptic Curve Digital Signature Algorithm) та основні вимоги до його параметрів. Визначено, що ECDSA спирається на математику циклічних груп еліптичних кривих над кінцевими полями та на складність проблеми ECDLP (проблема дискретного логарифмування еліптичної кривої).

Проаналізовані основі вимоги до параметрів еліптичної кривої, генераторів псевдовипадкових чисел та криптографічних хеш-функцій, на основі яких працює ECDSA.

Таким чином, можна зробити висновок, що для належного рівня криптостійкості при використанні алгоритму цифрового підпису ECDSA першочергово потрібно:

– підібрати параметри еліптичної кривої таким чином, щоб вони задовольняли рівняння $4a^3 + 27b^2 \neq 0 (\mathrm{mod}\ p)$, де p – просте та велике.

– вибрати n, де $n > 2^{160}$ та $n > 4\sqrt{p}$;

– використовувати генератор псевдовипадкових чисел з максимально великим періодом;

використовувати тільки ті криптографічні хеш-функції, за допомогою яких нездійсненно утворити повідомлення для заданого хеш-значення.

По-третє, мовою програмування C++ була розроблена програма, що реалізовує алгоритм ECDSA. Були розглянуті алгоритм та функції, які використовуються в даній програмі.

Зроблено ряд дослідів, які підтверджують, що програма працює без помилок та правильно реагує на різні випадки.

Дана програма може бути використана для навчального процесу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бияшев Р. Г., Диев С. И., Размахнин М. К. Основные направления развития и совершенствования криптографического закрытия информации // Зарубежная радиоэлектроника. - 1989. - № 12. - С. 35 - 39.
2. Rivest R., Shamir A., Adleman L. A method for obtaining digital signatures and public key cryptosystems // Commun. ACM. - 1978. - Vol.21. № 2. - P. 120 - 126.
3. ДСТУ 3396.2-97. Захист інформації. Технічний захист інформації. Терміни та визначення. – Введ. 01.01.98. – К.:Держстандарт України,1997. –11 с.
4. Lenstra A., Lenstra H. The Development of the Number Field Sieve // Lect. Notes in Math. - 1993. - Vol. 1554, №2. - P. 155 - 157.
5. Диффи У. Первые десять лет криптографии с открытым ключом // ТИИЭР. - 1988. - Т. 76, №5. - С. 54 - 63.
6. A. Menezes, Y.-H. Wu, and R. Zuccherato. An Elementary Introduction to Hyperelliptic Curves. In Algebraic Aspects of Cryptography
7. Elliptic Curve Cryptography. Certicom Research, 1999. Working Draft.
8. IEEE P1363 / D8(Draft Version 8). Standard Specifications for Public Key Cryptography.
9. N.Koblitz, Elliptic Curve Cryptosystems [Text]/ Neal Koblitz// Mathematics of Computation – Volume 48 – Number 177 – January 1987 – pp. 203-209
10. Добуш А.Р. Переваги реалізації у НВІС пристроїв для обробки цифрового підпису, що ґрунтується на властивостях груп точок еліптичної кривої – 2012 – УДК 004.382 – С. 81 – 82.
11. Черемушкин, А.В. Лекции по арифметическим алгоритмам в криптгографии. — М. : МЦНМО, 2002. — 104 с.
12. Панасенко, С.П. Алгоритмы Шифрования. Специальный Справочник [Текст] / С.П. Панасенко. — М. : Академический Проект, 2006. — 529 с.
13. Василенко, О.Н. Теоретико-числовые алгоритмы в криптографии [Текст] / О.Н. Василенко. — М. : МЦНМО, 2003. — 328с.

14. Elliptic Curve Cryptography: a gentle introduction [Електронний ресурс]: https://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/

15. Фергюсон. Практическая криптография: Пер. с англ. [Текст] / Нильс, Шнайер, Брюс. — М. : «Вильямс», 2005. — 424 с.

16. 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.

17. D. E. Knuth. The Art of Computer Programming: Seminumerical Algorithms, volume 2. Addison-Wesley, Reading, Mass., 2nd edition, 1981.

18. D. H. Lehmer. Euclid's algorithm for large numbers. Amer. Math. Monthly, 45:227-233, 1938.

19. 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.

20. J. M. Pollard. Theorems on factorization and primality testing. Proc. Camb. Phil. Soc., 76:521-528, 1974.

21. 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.

22. Don Johnson, Alfred Menezes, "The Elliptic Curve Digital Signature Algorithm (ECDSA)", pages 36-49.

23. ДСТУ 4145-2002 «Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка», Введ. 28.12.2002. – К.:Держспоживстандарт України,2002. – 9 с.

Додаток А

Лістинг програми

```
#include <iostream>
#include <math.h>
#include <Windows.h>
#include <time.h>

using namespace std;

int Prime (int p){///Функція, яка проводить перевірку числа на простоту.
   for(int i=2;i<=sqrt(p);i++)
      if(p%i==0)
         return 0;
   return 1;
}

int inv(int a, int p){///Функція пошуку оберненого числа по модулю за
                       допомогою алгоритму Евкліда.
   int k1=0,k2=1,k3,a1,n;
   a=(a%p+p)%p;
   if (a==1)
      return 1;
   while(a!=1){
      n=p/a;
      k3=k1-k2*n;
      a1=p%a;
      k1=k2;
      k2=k3;
      p=a;
      a=a1;
   }
   return k3;
}

int SumX (int x1, int y1, int x2, int y2, int p){///Функція, яка обчислює
                                                  координату x при додаванні двох точок на ЕК.
   int z=(y2-y1)*inv(x2-x1,p);
```

```
    int x=z*z-x1-x2;
    return (x%p+p)%p;
}
int SumY (int x1, int y1, int x2, int y2, int p, int x){///Функція, яка обчислює
                                                координату y при додаванні двох точок на ЕК.
    int y=(y2-y1)*inv(x2-x1,p)*(x1-x)-y1;
    return (y%p+p)%p;
}

int doublX(int x1, int y1, int p, int a){///Функція, яка обчислює координату x
                                                при подвоєнні точки на ЕК.
    int z=(3*pow(x1,2)+a)*inv(2*y1,p);
    int x=z*z-2*x1;
    return (x%p+p)%p;
}
int doublY(int x1, int y1, int p, int a, int x){///Функція, яка обчислює координату
                                                y при подвоєнні точки на ЕК.
    int y=(3*pow(x1,2)+a)*inv(2*y1,p)*(x1-x)-y1;
    return (y%p+p)%p;
}

int DecToBin(int a, int *bin){///Функція, яка переводить число з десяткової
                                        системи числення у двійкову.
    int i=0;
    while(a!=1){
        bin[i]=a%2;
        a/=2;
        i++;
    }
    bin[i]=1;
    return *bin;
}

int Mult(int *g, int k, int x, int y, int a, int p){///Скалярне множення точки на
                                    константу за допомогою алгоритму подвоєння-додавання.
    int bin[8]={0},doubl[8][2], d=7;
    DecToBin(k,bin);
    for (d; d>0; d--)
```

```
        if(bin[d]==1)
            break;
    doubl[0][1]=x;
    doubl[0][2]=y;
    for(int i=1; i<=d; i++){///Реалізація подвоєння точки.
        if (doubl[i-1][2]==0||doubl[i-1][2]==-1)
            doubl[i][1]==-1;
        else {
            doubl[i][1]=doublX(doubl[i-1][1],doubl[i-1][2],p,a);
            doubl[i][2]=doublY(doubl[i-1][1],doubl[i-1][2],p,a,doubl[i][1]);
        }
    }
    int f=0;
    for(f; f<8; f++)
        if(bin[f]==1)
            break;
    for(int i=f+1; i<=d; i++)///Реалізація додавання точок.
        if(bin[i]==1){
            if (doubl[f][1]==doubl[i][1])
                doubl[f][1]==-1;
            else if (doubl[f][1]==-1){
                doubl[f][1]=doubl[i][1];
                doubl[f][2]=doubl[i][2];
            }
            else {
                x=SumX(doubl[f][1],doubl[f][2],doubl[i][1],doubl[i][2],p);
                y=SumY(doubl[f][1],doubl[f][2],doubl[i][1],doubl[i][2],p,x);
                doubl[f][1]=x;
                doubl[f][2]=y;
            }
        }
    g[1]=doubl[f][1];
    g[2]=doubl[f][2];
    return *g;
}

int main()
{
```

```
SetConsoleCP(1251);
SetConsoleOutputCP(1251);

int p, a, b, n, sk, hashM, k, xG, yG, r, s, u1, u2;
int PP1[2];
int PP2[2];
int pk[2];

cout << "Оголошення глобальних параметрів:" << endl;
while(1){///Ввід параметру p та перевірка його на простоту.
   cout << "Введіть p: ";
   cin >> p;
   if(Prime(p)==1&&p>3)
      break;
   cout << "p - не просте, або p менше 3" << endl;
}

while(1){///Ввід параметру a і b та перевірка ЕК на сингулярність.
   cout << "Введіть a: ";
   cin >> a;
   cout << "Введіть b: ";
   cin >> b;
   if((4*pow(a,3)+27*pow(b,2))!=0)
      break;
   cout << "ЕК - сингулярна" << endl;
}

while(1){///Ввід координат базової точки GP та перевірка на належність її
            до заданої ЕК
   cout << "Введіть координату x точки GP: ";
   cin >> xG;
   cout << "Введіть координату y точки GP: ";
   cin >> yG;
   if(((xG*xG*xG+a*xG+b)%p)==((yG*yG)%p))
      break;
   cout << "точка GP не лежить на ЕК" << endl;
}
int yy=yG;
```

```
cout << "Введіть порядок точки GP: ";///Ввід порядку точки GP
cin >> n;

cout << endl << "Аліса генерує: " << endl;
srand(time(NULL));
sk=1+rand()%(n-1);
cout << "Секретний ключ: " << sk <<endl;///Генерація секретного ключа
Mult(pk,sk,xG,yG,a,p);///Обчислення публічного ключа за допомогою
                        секретного ключа та базової точки
if (pk[1]==(-1))
   cout << "pk = O(точка на нескінченності)" << endl;
else
   cout << "Публічний ключ: (" << pk[1] << ";" << pk[2] << ")" << endl;
while (1){
   hashM=(1+rand())%n;
   if (hashM!=0)
      break;
}
cout << "Хеш-функція повідомлення: " << hashM << endl << endl;

cout << "Аліса створює цифровий підпис: ";///Створення цифрового
                                            підпису згідно алгоритму
while(1){
   k=1+rand()%(n-1);
   Mult(PP1,k,xG,yy,a,p);
   r=PP1[1]%n;
   if(r!=0){
      s=((inv(k,n)*(hashM+r*sk))%n+n)%n;
      if(s!=0)
         break;
   }
}
cout << "(" << r << ";" << s << ")" << endl << endl;///Виведення цифрового
                                                    підпису на екран

cout << "Аліса відправляє Бобу:" << endl;///Розголошення відкритої
                                          інформації незахищеними каналами зв'язку
cout << "Публічний ключ: (" << pk[1] << ";" << pk[2] << ")" << endl;
```

```
    cout << "Хеш-функцію повідомлення: " << hashM << endl;
    cout << "Цифровий підпис: (" << r << ";" << s << ")" << endl << endl;

    cout << "Боб перевіряє підпис Аліси: ";///Перевірка цифрового підпису
                                                згідно алгоритму
    u1=((inv(s,n)*hashM)%n+n)%n;
    u2=((inv(s,n)*r)%n+n)%n;
    int x;
    Mult(PP1,u1,xG,yy,a,p);
    Mult(PP2,u2,xG,yy,a,p);
    x=SumX(PP1[1],PP1[2],PP2[1],PP2[2],p);
    if (r==x%n)///Вивід результату перевірки цифрового підпису
       cout << "TRUE" << endl;
    else
       cout << "FALSE" << endl;
    return 0;
}
```