Міністерство освіти і науки, молоді та спорту України
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

«До захисту допущено»
Завідувач кафедри
комп'ютерної інженерії
к.т.н., доц. О.М.Березький
_____
"____"_____20__ р.

ДИПЛОМНИЙ ПРОЕКТ
освітньо-кваліфікаційного рівня "Спеціаліст"
зі спеціальності 7.05010201 "Комп'ютерні системи та мережі"
на тему:

Програмно-апаратна реалізація алгоритму експоненціювання на еліптичних кривих

Студент групи КСМ$_C$-51_____Пекельний Н.М.
(підпис)

Керівник:
викладач _____ Якименко І.З.
(підпис)

Нормоконтроль:
к.т.н., доцент _____Васильків Н.М.
(підпис)

Консультант
з охорони праці
доцент _____Сапожник Г.В.
(підпис)

2012

Міністерство освіти і науки, молоді та спорту України
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії
спеціальність 7.05010201 – "Комп'ютерні системи та мережі"

"Затверджую"
завідувач кафедри
комп'ютерної інженерії
к.т.н., доц. О.М.Березький
_____
"____"_____20__ р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТА
Пекельного Назара Мар'яновича

1. Тема проекту: "Програмно-апаратна реалізація алгоритму експоненціювання на еліптичних кривих" затверджена наказом університету № 475 від 14 жовтня 2011 р.

2. Термін здачі студентом закінченого проекту "___"_____20__ р.

3. Вихідні дані для проекту: Технічне завдання.

4. Перелік задач, які мають бути вирішені:
- провести аналіз систем захисту інформаційних потоків з використанням математичного апарату еліптичних кривих;
- встановити основні задачі ефективної реалізації криптографії на еліптичних кривих;
- вказати основні переваги апаратної реалізації алгоритмів на еліптичних кривих;
- визначити стійкість та продуктивність алгоритмів експоненціювання точок на еліптичних кривих;
- програмно-апаратно реалізувати алгоритм експоненціювання точок на еліптичних кривих;
- дослідити часові та продуктивні параметри апаратної та програмної реалізації.

5. Перелік графічного матеріалу (з точним вказанням обов'язкових креслень)
- Структурна схема організації захисту інформаційних потоків з використанням математичного апарату еліптичних кривих;
- блок-схема алгоритму генерування параметрів еліптичної кривої

еліптичних кривих;

– блок-схема алгоритму подвоєння-додавання віднімання точок на еліптичних кривих;

– класифікація застосування криптографії еліптичних кривих;

6.Консультанти по проекту (із зазначенням розділів):

Розділ	Консультант	Підпис
Охорона праці	Сапожник Г.В.	

КАЛЕНДАРНИЙ ПЛАН

№	Назва розділів дипломного проекту	Термін виконання	Позначки керівника про виконання завдань
1	Криптографія еліптичних кривих	15.09.2011 – 5.11.2011	
2	Методи експоненціювання точок на еліптичних кривих	6.11.2011 – 31.01.2012	
3	Програмно-апаратна реалізація операцій експоненціювання точок на еліптичних кривих	1.02.2012 – 14.04.2012	
4	Охорона праці	15.04.2012 – 23.04.2012	

Завдання прийняв до виконання _____
(підпис)

Керівник дипломного проекту _____
(підпис)

Технічне завдання
на дипломний проект
“Програмно-апаратна реалізація алгоритмів експоненціювання на еліптичних кривих”

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

1.1. Програмно-апаратна реалізація алгоритму експоненціювання на еліптичних кривих.

1.2. Область застосування – підприємства які використовують систем захисту інформаційних потоків з використанням асиметричних криптоалгоритмів.

2. ОСНОВА ДЛЯ РОЗРОБКИ

Основою для розробки є індивідуальне завдання на дипломний проект, затверджене кафедрою комп’ютерної інженерії (КІ) факультету комп’ютерних інформаційних технологій Тернопільського національного економічного університету.

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даної розробки є розробка програмно – апаратного комплексу базової операції медулярного експоненціювання алгоритму шифрування на основі використання математичного апарату еліптичних кривих.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами даної розробки є матеріали навчальної і реферативної літератури, технічна документація, науково-дослідні роботи, журнали.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до структури і функцій

5.1.1. Робочий діапазон частот 310 МГц – 800 МГц;

5.1.2. Швидкість передачі даних до 100 Мбіт/с;

5.1.3. Потужність споживання не більше 100 Вт;

5.1.4. Час встановлення робочого режиму не перевищує 3 с.

5.2. Вимоги до апаратної сумісності

5.2.1. IBM сумісні комп'ютери.

5.2.2. Стандартні інтерфейси

5.3. Вимоги до надійності

5.3.1. Час безвідмовної роботи 6000 год.

5.3.2. Надійна робота при дотримані умов експлуатації

5.4. Вимоги безпеки

5.4.1. Відповідність вимогам електробезпеки та пожежної безпеки згідно стандартів.

5.5. Умови експлуатації

5.5.1. Експлуатувати в закритих приміщеннях

5.5.2. Для нормальної роботи системи необхідно підтримувати (по ГОСТ 23.865–85):

- температуру повітря в межах від +5^{o}C до +40^{o}C;
- відносну вологість повітря при +25^{o}C в межах від 30% до 80%;
- атмосферний тиск – 760±25 мм.рт.ст.

6. ВИМОГИ ОХОРОНИ ПРАЦІ

6.1. В розділі "Охорона праці" дипломного проекту повинен бути даний аналіз умов праці при експлуатації автоматизованої системи диспетчерського управління енергопостачанням промислового підприємства..

7. ПОРЯДОК КОНТРОЛЮ І ПРИЙОМКИ

7.1. Представлення дипломного проекту на попередній захист.

7.2. Представлення дипломного проекту на захист.

Анотація

Дана робота виконана на 135 сторінках основного тексту, містить 19 ілюстрацій, 5 таблиць, 60 джерел за переліком посилань.

Мета роботи. Метою досліджень є дослідження ефективності застосування реалізацій алгоритмів на ЕК для засобів захисту інформації в комп'ютерних системах. Дана робота являє собою комплексний теоретичний аналіз і створення нової сучасної обчислювальної технології оцінки часових та продуктивних характеристик криптографічних алгоритмів з використанням математичного апарату ЕК. Розробка алгоритмів шифрування є одним з аспектів створення надійних систем захисту інформації.

Методи дослідження і апаратура. Для проведення описаних в даній роботі досліджень, використовуються результати з таких областей знань: для дослідження та аналізу криптографічних перетворень алгоритмів ЕК - теорія чисел та алгебра Евкліда; при досліджені методів криптоаналізу - лінійна алгебра, теорія складності та теорія Галуа.

Результати роботи та їх новизна. Теоретична цінність даної роботи полягає у розробленій методиці оцінки стійкості сучасних алгоритмів модулярного експоненціювання до атак спеціальних впливів. Практична цінність полягає у розроблених рекомендаціях щодо використання сучасних алгоритмів модулярного експоненціювання для розробки високопродуктивних засобів захисту інформації, стійких до атак спеціальних впливів.

Рекомендації по використанню результатів роботи. Результати даної роботи можуть бути використаними при застосуванні існуючих та розробці нових алгоритмів модулярного експоненціювання для задач захисту інформації, що базуються на асиметричній криптографії.

Значущість роботи та висновки. Результати даної роботи розширюють область проектування алгоритмів реалізації базових операцій засобів захисту інформації, стійких до атак спеціальних впливів.

Напрямки продовження досліджень. Подальші дослідження полягають в аналізі стійкості алгоритмів шифрування до часової атаки, атаки апаратних помилок.

Ключові слова: криптографія, асиметричні криптосистеми, еліптичні криві(ЕК), алгоритм експоненціювання точок на Еліптичних Кривих, продуктивність алгоритму, вага Хемінга, стійкість криптосистеми.

Annotation

This work is done on 135 pages of main text contains 19 artwork 5 tables 60 sources for references.

Purpose. The aim of research is to study the effectiveness of implementations of algorithms for EC for information security in computer systems. This work is a comprehensive theoretical analysis and a new modern computer technology evaluation time and productive characteristics of cryptographic algorithms using mathematical tools EC. Development of algorithms for encryption is one aspect of a robust information security systems.

Methods and apparatus. Facilities described in this paper, research results are used in the fields of: for research and analysis of cryptographic algorithms EC - Number Theory and Algebra Euclid, with the methods of cryptanalysis - linear algebra, complexity theory and Galois theory.

The results and their novelty. The theoretical value of this work is developed technique estimates the stability of modern algorithms for modular eksponentsiyuvannya to attack special effects. The practical value lies in the recommendations for use of modern algorithms for modular eksponentsiyuvannya to develop high-performance data protection, resistant to attack by special effects.

Recommendations for use of the results. The results of this work may be used in applying existing and developing new algorithms for modular eksponentsiyuvannya problems of information security based on asymmetric cryptography.

The significance of the work and conclusions. The results of this work extend the design algorithm of the basic operations of information security, resistant to attack by special effects.

Areas of continued research. Further investigation is to analyze the stability of encryption algorithms for time attack, attack of hardware errors.

Keywords: cryptography, asymmetric cryptosystem, elliptic curves, the algorithm eksponentsiyuvannya points on the EC, the performance of the algorithm, the weight Heminha, stability cryptosystem.

ЗМІСТ

ВСТУП

Безумовним є той факт, що людство переходить до нової фази свого розвитку – інформаційної, що характеризується широким впровадженням інформаційних технологій у всі сфери його діяльності. Основною особливістю цієї фази є створення та колективне використання масивів та баз даних, баз знань, інтенсивний обмін повідомленнями, передача команд управління, широке використання та розповсюдження різноманітного програмного забезпечення. При цьому прийнятну якість функціонування інформаційних систем можна досягнути тільки за умови забезпечення необхідних цілісності та аутентичності (справжності) інформації на всіх етапах її життєвого циклу – від створення до знищення. Під цілісністю інформації необхідно розуміти умови, при яких дані (повідомлення, команди, програмне забезпечення) зберігаються для використання за призначенням, а під аутентифікацією розуміється встановлення справжності інформації (повідомлень, команд, даних, програмного забезпечення), джерела та приймача даних винятково на основі внутрішньої структури самої інформації. По суті основною кінцевою метою аутентифікації інформації є забезпечення захисту учасників інформаційного обміну від навмисного або випадкового нав'язування неправдивої інформації. Аутентифікацію як процедуру встановлення справжності інформації можна реалізовувати для однієї з моделей взаємодії учасників (абонентів, прикладних процесів) інформаційного обміну – взаємної довіри та захисту, а також взаємної недовіри та взаємного захисту.

Інформаційна безпека відіграє важливу роль у забезпеченні життєво важливих інтересів будь-якої держави. Створення розвиненого і захищеного інформаційного розвитку суспільства є невід’ємною умовою розвитку суспільства та держави. Глибока структура та технологічна реформа, що проходить сьогодні в Україні, спрямована на впровадження

низки важливих для держави автоматизованих інформаційних систем і мереж зв’язку та передачі даних, систем прийняття рішень.

Аналіз стану захисту інформації в інформаційних системах України свідчить, що в цілому стан розв’язання цієї задачі в державі далекий від досконалості. Тим більше, що виникає потреба в побудові стійких і продуктивних криптографічних алгоритмів. До них можна віднести криптографічні алгоритми з використанням математичного апарату еліптичних кривих (ЕК), які потребують менших витрат ресурсів для забезпечення високого рівня захисту.

Одним з найосновніших завдань криптографії еліптичних кривих є скалярне експоненціювання точки на кривій. Адже саме ця процедура і є шифруванням інформації. Наприклад, ключем може виступати базова точка кривої, інформацію, яку потрібно зашифрувати, можна використати як множник. А результат множення базової точки на множник і буде зашифрованою інформацією.

З огляду на все ширше впровадження КЕК в сучасних системах було вирішено реалізувати програмний продукт, який би здійснював операцію скалярного множення точки на еліптичній кривій. Цей продукт можна використати не тільки для шифрування, але й для дослідження швидкодії та загальної продуктивності криптографічних алгоритмів криптографії еліптичних кривих, оскільки всі ці алгоритми використовують експоненціювання точок на ЕК.

1 КРИПТОГРАФІЯ ЕЛІПТИЧНИХ КРИВИХ

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 електронно-цифрового підпису (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 комп’ютерної системи (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 (Рисунок 1.1).

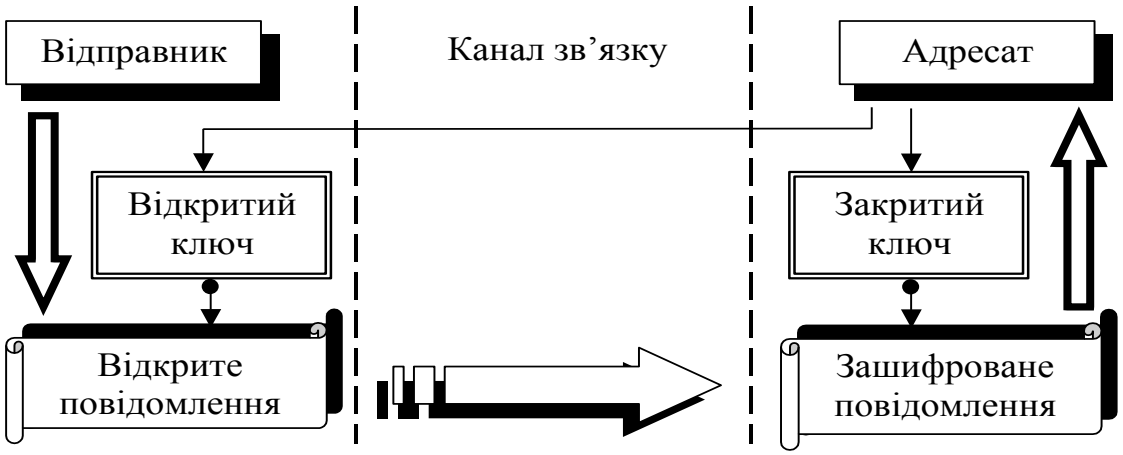


Рисунок 1.1 – Процедура шифрування і дешифрування з відкритим ключем

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

У таблиці 1.1 наведені довжини ключів симетричної і асиметричної криптосистем, що забезпечують однакову стійкість до атаки повного перебору, “брутальної атаки”.

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 $y = f(x)$, то немає простого шляху для обчислення значення x [22].

Велика кількість класів необоротних функцій і породжує всю розмаїтість 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 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 інформаційних системах (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 та обчислення дискретних логарифмів.

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 Ель-Гамаля, алгоритм Поліга-Хеллмана, криптосистеми на основі еліптичних кривих (ЕК).

Криптографія з відкритим ключем може застосовуватися як для шифрування повідомлень, так і для аутентифікації (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. Тобто ключова система шифрування заснована на цій проблемі.

Еліптичні криві - математичні конструкції, що вивчилися 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 [19].

Безпека криптосистем на основі еліптичних кривих, як правило, заснована на складності рішення задачі дискретного логарифмування в групі точок на еліптичній кривій над кінцевим полем. Цим і обумовлена їх висока криптостійкість у порівнянні з іншими алгоритмами. Існують стійкі криптоалгоритми на ЕК, засновані на складності розкладання великих цілих чисел, коли ЕК задається над кінцевим 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 до криптосистем на ЕК це алгоритми Сільвера-Поліга-Хеллмана, Полларда і інші.

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 міняється, 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. Але зате їх можна ефективно використовувати для систем цифрового підпису і обміну ключовою інформацією. Вже існує попередній стандарт ANSI X9.62, що пропонує розробникам принципи створення могутніх систем цифрового підпису на основі використання кривих. Безпека таких систем цифрового підпису спирається не тільки на стійкість алгоритму на ЕК, але і на стійкість 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 ведучими 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 і інших відомих асиметричних та симетричних криптосистем захисту інформації [18].

Департамент стандартизації IEEE в 1999 році прийняв стандарт P1363, в якому описані деякі розділи модулярної арифметики, а також алгоритми генерування параметрів, додавання і множення точок та інше, що стосується еліптичних кривих [28].

1.2 Задачі ефективної реалізації криптографії ЕК

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

У сучасних криптосистемах на основі еліптичних кривих бінарної розмірності в діапазоні від 150 до 350 забезпечується рівень криптографічної стійкості, який потрібно використовувати у відомих криптографічних системах бінарної розмірності від 600 до 1400 і більше [23].

У наведеній нижче Таблиці 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 (базова точка P)	RSA (довжина модуля n)
1024 біт	3084 біт
3250 біт	9750 біт
15500 біт	46500 біт

Проаналізувавши таблицю видно, що використання еліптичних кривих дозволяє будувати стійкі системи з ключами значно менших розмірів у порівнянні до традиційних асиметричних криптоалгоритмів. 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 побудову 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:

- генерування параметрів еліптичної кривої;
- обчислення точок еліптичної кривої;
- проблема дискретного логарифму.

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 в безкінечності” (“безкінечним 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 [14].

У випадку криптографії з використанням еліптичних кривих доводиться мати справу зі зміненою формою еліптичної кривої, яка визначається над кінцевим полем. Особливий інтерес для криптографії представляє об'єкт, який називається еліптичною групою по модулю p, де p є простим числом [5].

В криптографії з використанням еліптичних кривих над кінцевим 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 $Ep(a,b)$. 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 $0 \leq x < p$, 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 $Ep(a,b)$ нема 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 $Ep(a,b)$ 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 $Ep(a,b)$;

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

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

(1.4)

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

(1.5)

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

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

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

(1.6)

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

(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 (Elliptic Curve Discrete Logarithm Problem – задача дискретного логарифму на еліптичній кривій), 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 Переваги апаратної реалізації еліптичних кривих для алгоритмів шифрування

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.

Багато засобів криптографічного захисту даних реалізовано у виді спеціалізованих апаратних пристроїв. Ці пристрої вбудовуються в лінію зв'язку і здійснюють шифрування всієї переданої по ній інформації. Перевага апаратного шифрування над програмним обумовлено декількома причинами [13].

По-перше, апаратне шифрування має більшу швидкість. Криптографічні алгоритми складаються з величезного числа складних операцій, виконуваних над бітами відкритого тексту. Сучасні універсальні комп'ютери погано пристосовані для ефективного виконання цих операцій. Спеціалізоване устаткування вміє робити їх набагато швидше.

По-друге, апаратуру легше фізично захистити від проникнення ззовні. Програма, виконувана на персональному комп'ютері, практично беззахисна. Озброївшись відладчиком, зловмисник може потай внести в неї зміни, щоб понизити стійкість використовуваного криптографічного алгоритму, і ніхто нічого не помітить. Що ж стосується апаратури, то вона зазвичай міститься в особливі контейнери, що унеможливлює зміну схеми її функціонування.

Чіп покривається зверху спеціальним хімічним складом, і в результаті будь-яка спроба перебороти захисний шар цієї схеми приводить до самознищення його внутрішньої логічної структури. І хоча іноді електромагнітне випромінювання може служити гарним джерелом інформації про те, що відбувається усередині мікросхеми, від цього випромінювання легко позбутися, екранувавши мікросхему. Аналогічним чином можна екранувати і комп'ютер, однак зробити це набагато складніше, ніж у випадку мініатюрної мікросхеми.

І по-третє, апаратура шифрування більш проста в установці. Дуже часте шифрування потрібно там, де додаткове комп'ютерне устаткування є зовсім зайвим. Телефони, факсимільні апарати і модеми значно дешевше обладнати пристроями апаратного шифрування, чим вбудовувати в них мікрокомп'ютери з відповідним програмним забезпеченням.

Навіть у комп'ютерах установка спеціалізованого шифрувального устаткування створює менше проблем, чим модернізація системного програмного забезпечення з метою додавання в нього функцій шифрування даних. В ідеалі шифрування повинне здійснюватися непомітно для користувача.

Щоб домогтися цього за допомогою програмних засобів, шифрування повинне бути сховане глибоко в надра операційної системи. З готовою і налагодженою операційною системою безболісно проробити це не так вже просто. Але навіть будь-який непрофесіонал зможе приєднати шифрувальний блок з однієї сторони до персонального комп'ютера і до зовнішнього модему з іншої.

Сучасний ринок апаратних засобів шифрування інформації пропонує потенційним покупцям три різновиди таких засобів:

- самодостатні шифрувальні модулі (вони самостійно виконують усю роботу з ключами);
- блоки шифрування в каналах зв'язку;
- шифрувальні плати розширення для установки в персональні комп'ютери.

Більшість пристроїв першого і другого типу є вузько спеціалізованими, і тому, перш ніж приймати остаточне і безповоротне рішення про їхнє придбання, необхідно досконально вивчити обмеження, які при установці накладають ці пристрої на відповідну апаратуру, операційні системи і прикладне програмне забезпечення. А інакше можна витрачати значні кошти, ні на крок не наблизившись до бажаної мети. Правда, іноді вибір полегшується тим, що деякі компанії торгують

комунікаційним устаткуванням, що вже має у своєму складі вмонтовану апаратуру шифрування даних.

Плати розширення для персональних комп'ютерів є більш універсальним засобом апаратного шифрування і звичайно можуть бути легко зконфігуровані таким чином, щоб шифрувати всю інформацію, що записується на жорсткий диск комп'ютера, а також усі дані, що пересилаються на його гнучкий диск і в послідовні порти. Як правило, захист від електромагнітного випромінювання в шифрувальних платах розширення відсутній, оскільки нема потреби захищати ці плати, якщо аналогічні міри не починаються у відношенні всього комп'ютера в цілому.

Криптографічні системи захисту інформації на основі еліптичних кривих можуть бути реалізовані як програмно, так і апаратно.

Перевага вище згаданих систем в порівнянні з іншими асиметричними криптосистемами полягає в тому, що в даному випадку забезпечується еквівалентний рівень захисту при використанні меншої довжини ключа.

1.4 Вибір мікроконтроллера для ефективної апаратної реалізації систем захисту на ЕК

На даний час існує дві потужні мікроконтролерні фірми, які випускають мікроконтролери загального призначення – це фірми Microchip (мікроконтролери серії PIC) та Atmel (серія AVR). Для правильного вибору з точку зору продуктивності, периферії, якості та вартості потрібно провести порівняння продукції обох фірм.

Перші мікроконтролери компанії MICROCHIP PIC16C5x з'явилися в кінці 80-х років і завдяки своїй високій продуктивності і низькій

вартості склали серйозну конкуренцію вироблюваним у той час 8-розрядним МК з CISC-архітектурою.

Перше, що привертає увагу в PIC-контролерах – це простота і ефективність. У основу концепції PIC, єдину для всіх сімейств, що випускаються, була покладена RISC-архітектура з системою простих однослівних команд, застосування вбудованої пам'яті програм і даних і мале енергоспоживання.

Система команд базового сімейства PIC165x містить тільки 33 команди. Як не дивно, і це зіграло свою роль в популяризації PIC-контролерів. Всі команди (окрім команд переходу) виконуються за один машинний цикл (або чотири машинні такти) з перекриттям за часом вибірок команд і їх виконання, що дозволяє досягти продуктивності 5 MIPS при тактовій частоті 20 Мгц.

Мікроконтролери PIC мають симетричну систему команд, що дозволяє виконувати операції з будь-яким регістром, використовуючи будь-який метод адресації. Правда, розробники MICROCHIP так і не змогли відмовитися від коханій всіма структури з регістром-акумулятором, необхідним учасником всіх операцій з двома операндами. Зате тепер користувач може зберігати результат операції на вибір, де побажає, в самому регістрі-акумуляторі або в другому регістрі, використовуваному для операції. В даний час MICROCHIP випускає чотири основні сімейства 8-розрядних RISC-мікроконтролерів, сумісних від низу до верху за програмним кодом:

- базове сімейство PIC15Cx з 12-розрядними командами, прості недорогі мікроконтролери з мінімальною периферією;
- PIC12Cxxx з 12-розрядними командами та вбудованим тактовим генератором, що випускаються в мініатюрного 8-вивідного виконання. Не так давно був анонсований черговою такий "малюк" з внутрішнім 8-розрядним 4-канальним АЦП;
- Mid-range PIC16x/7x/8x/9x з 14-розрядними командами. Найбільш численне сімейство, що об'єднує мікроконтролери з

різноманітними

периферійними пристроями, до числа яких входять аналогові компаратори, аналогово-цифрові перетворювачі, контролери послідовних інтерфейсів SPI, USART і I2C, таймери-лічильники, модулі захоплення/порівняння, широко-імпульсні модулятори, сторожові таймери, супервізорні схеми і так далі;

- High-end PIC17C4x/5xx високопродуктивні мікроконтролери з розширеною системою команд 16-розрядного формату, що працюють на частоті до 33 Мгц, з об'ємом пам'яті програм до 16 Кслів. Окрім обширної периферії майже всі мікроконтролери цього сімейства мають вбудований апаратний помножувач 8x8, що виконує операцію множення за один машинний цикл.

Більшість PIC-контролерів випускаються з одноразово програмованою пам'яттю програм OTP з можливістю внутрішньосхемного програмування або масочною ROM. Для цілей відладки пропонуються версії з ультрафіолетовим стиранням, треба визнати, не дуже дешеві. Проте останнім часом фірма Microchip випустила серію мікроконтролерів з використанням Flash-пам’яті і досить доступною ціною. Повна кількість модифікацій PIC-контролерів, що випускаються, складає близько п'ятисот найменувань. Як небезпідставно затверджує MICROCHIP, продукція компанії перекриває весь діапазон застосувань мікроконтролерів.

Особливий акцент MICROCHIP робить на максимально можливе зниження енергоспоживання для мікроконтролерів, що випускаються. При роботі на частоті 4 Мгц PIC-контролери, залежно від моделі, мають струм споживання менше 1,5 мА, а при роботі на частоті 32,768 КГц нижче 15 мкА. Підтримується "сплячий" режим роботи. Діапазон напруги живлення PIC-контролерів складає 2,0...6,0 В.

В даний час готується до запуску у виробництво нове п'яте сімейство PIC-контролерів PIC18Cxxx. Нові мікроконтролери матимуть

розширене RISC-ядро, оптимізоване під використання нового Сі-компілятора, адресний простір програм до 2 Мбайт, до 4 Кбайт вбудованої пам'яті даних і продуктивності 10 MIPS.

З програмних засобів відладки найбільш відомі і доступні різні версії асемблерів, а також інтегроване програмне середовище MPLAB.

На відміну від MICROCHIP, компанія ATMEL Corp., один з світових лідерів у виробництві широкого спектру мікросхем незалежної пам'яті, FLASH-мікроконтролерів і мікросхем програмованої логіки, узяла старт по розробці RISC-мікроконтролерів у середині 90-х років, використовуючи всі свої технічні рішення, накопичені до цього часу.

Концепція нових швидкісних мікроконтролерів була розроблена групою розробників дослідницького центру ATMEL в Норвегії, ініціали яких потім сформували марку AVR. Перші мікроконтролери AVR AT90S1200 з'явилися у середині 1997 р. і швидко здобули розташування споживачів.

AVR-архітектура, на основі якої побудовані мікроконтролери сімейства AT90S, об'єднує могутній гарвардський RISC-процесор з роздільним доступом до пам'яті програм і даних, 32 регістри загального призначення, кожний з яких може працювати як регістр- акумулятор, і розвинену систему команд фіксованої 16-бітної довжини. Більшість команд виконуються за один машинний такт з одночасного виконання поточною і вибіркою наступної команди, що забезпечує продуктивність до 1 MIPS на кожен Мгц тактової частоти.

32 регістри загального призначення утворюють регістровий файл швидкого доступу, де кожен регістр безпосередньо пов'язаний з АЛУ. За один такт з регістрового файлу вибираються два операнди, виконується операція, і результат повертається в регістровий файл. АЛУ підтримує арифметичні і логічні операції з регістрами, між регістром і константою або безпосередньо з регістром.

Регістровий файл також доступний як частина пам'яті даних. 6 з 32-х регістрів можуть використовуватися як три 16-розрядні регістри-покажчики для непрямої адресації. Старші мікроконтролери сімейства AVR мають у складі АЛУ апаратний помножувач.

Базовий набір команд AVR містить 120 інструкцій. Інструкції бітових операцій включають інструкції установки, очищення і тестування бітів.

Всі мікроконтролери AVR мають вбудовану FLASH ROM з можливістю внутрішньосхемного програмування через послідовний 4-провідниковий інтерфейс.

Периферія МК AVR включає: таймери-лічильники, широко-імпульсні модулятори, підтримку зовнішніх переривань, аналогові компаратори, 10-розрядний 8-канальний АЦП, паралельні порти (від 3 до 48 ліній вводу і виводу), інтерфейси UART і SPI, сторожовий таймер і пристрій скидання по включенню живлення. Всі ці якості перетворюють AVR-мікроконтролери на могутній інструмент для побудови сучасних, високопродуктивних і економічних контролерів різного призначення.

В рамках єдиної базової архітектури AVR-мікроконтролери підрозділяються на три підродини:

- Classic AVR основна лінія мікроконтролерів з продуктивністю окремих модифікацій до 16 MIPS, FLASH ROM програм 28 Кбайт, EEPROM даних 64512 байт, SRAM 128512 байт;
- mega AVR з продуктивністю 46 MIPS для складних додатків, що вимагають великого об’єму пам'яті, FLASH ROM програм 64128 Кбайт, EEPROM даних 64512 байт, SRAM 24 Кбайт, SRAM 4 Кбайт, вбудований 10-розрядний 8-канальний АЦП, апаратний помножувач 8x8;
- tiny AVR мікроконтролери низької вартості 8-вивідного виконання мають вбудовану схему контролю напруги живлення, що дозволяє обійтися без зовнішніх супервізорних мікросхем.

AVR-мікроконтролери підтримують сплячий режим і режим мікроспоживання. У сплячому режимі зупиняється центральне

процесорне
ядро, тоді як регістри, таймери-лічильники, сторожовий таймер і система переривань продовжують функціонувати. У режимі мікроспоживання зберігається вміст всіх регістрів, зупиняється тактовий генератор, забороняються всі функції мікроконтролера, поки не поступить сигнал зовнішнього переривання або апаратного скидання. Залежно від моделі, AVR-мікроконтролери працюють в діапазоні напруги 2,76 В або 46 В (виключення складає ATtiny12V з напругою живлення 1,2 В).

Засоби відладки. ATMEL пропонує програмне середовище AVR-studio для відладки програм в режимі симуляції на програмному відладчику, а також для роботи безпосередньо з внутрішньосхемним емулятором. AVR-studio доступний з WEB-сторінки ATMEL, містить асемблер і призначений для роботи з емуляторами ICEPRO і MEGAICE. Ряд компаній пропонують свої версії Сі-компіляторів, асемблерів, лінкувальників і завантажувачів для роботи з мікроконтролерами сімейства AVR.

Як видно з проведеного порівняння, мікроконтролери AVR фірми Atmel призначені для вирішення широкого профілю задач. Вони містять розвинену периферію, що дозволяє вирішувати багато різних задач на одній і тій самій моделі мікроконтролера. Проте чи потрібне це при вирішення поставленої задачі? Завдяки дуже широкому виборі моделей мікроконтролерів серії PIC фірми Microchip можна підібрати саме таку апаратну конфігурацію, яка найкраще підійде для вирішення саме цієї задачі. Завдяки цьому значно знизиться ціна вибраного контролера. Крім того, PIC-контролери мають знижене енергоспоживання. Звичайно, для створення і відладки створюваної системи потрібно мати контролер з пере записуваною пам’яттю програм. Проте останнім часом фірма Microchip випустила серію PIC-контролерів на основі саме Flash-пам’яті, що додає ще одну перевагу мікроконтролерам саме цієї серії.

В результаті проведеного дослідження було вибрано мікроконтролер фірми Microchip, марки PIC16F628. Його основні

характеристики: обсяг

пам’яті програм – 2048 байт, пам’ять даних EEPROM – 128 байт, обсяг оперативної пам’яті – 224 байти.

2 АЛГОРИТМ СКАЛЯРНОГО ЕКСПОНЕНЦІЮВАННЯ ТОЧОК НА ЕК

2.1 Теоретичні основи захисту інформації на основі еліптичних кривих

Нехай К – поле (наприклад, R – поле дійсних чисел, Q – поле раціональних чисел, C – поле комплексних чисел, F_q - кінцеве поле, $q=p^m$, де p — просте число).

Означення. Еліптичною кривою E над полем K називається множина точок $(x,y) \in K*K$, які задовільняють рівнянню (в афінних координатах)

$$E: \quad y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, ai \in K, \qquad (2.1)$$

разом із точкою O, що називають «точкою на нескінченності».

Іноді замість (2.1) зручно користуватися алгебраїчним рівнянням для функції двох змінних

$$F(x,y) = y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 = 0. \qquad (2.2)$$

Введення однієї операції (наприклад, додавання) над парами точок дозволяє побудувати структуру абелевої групи точок, якщо всі точки кривої неособливі, тобто мають однозначні похідні. Таку криву ще називають гладкою, або несингулярною кривою.

Означення. Крива E називається сингулярною (особливою), якщо існує хоча б одна точка (x, y), у якій часткові похідні функції (2.2) одночасно перетворюються в 0, тобто

$$\partial F / \partial x = \partial F / \partial y = 0. \qquad (2.3)$$

У протилежному випадку крива E називається несингулярною (неособливою).

Замість загального запису рівняння (2.1) часто розглядають канонічні рівняння трьох типів кривих:

$$\text{E:} \quad y^2 = x^3 + ax + b,\ p \neq 2,3; \qquad (2.4)$$

$$\text{E}_S\text{:} \quad y^2 + y = x^3 + ax + b,\ p = 2; \qquad (2.5)$$

$$\text{E}_N\text{:} \quad y^2 + xy = x^3 + ax^2 + b, b \neq 0,\ p = 2, \qquad (2.6)$$

перша з них описує всі криві над полями характеристики, які не дорівнюють 2 й 3, друга й третя - криві над полями характеристики 2. До канонічної форми несингулярної кривої з певними властивостями можна прийти заміною змінних в (2.1).

Розглянемо криві (2.4) над полями R й Q. Умова несингулярності цих кривих виявляється рівносильною умові відсутності кратних коренів у кубі в правій частині рівняння. Дійсно, виразимо куб у рівнянні (2.4) через корінь α_i,

$$f(x) = x^3 + ax + b = (x - \alpha_1)(x - \alpha_2)(x - \alpha_3),$$

причому слід кубика $\alpha_1 + \alpha_2 + \alpha_3 = 0$. Остання рівність, зокрема, дає простий шлях переходу до канонічної форми кривої зсувом всіх корінь. Згідно (2.3) у сингулярній точці

$$\partial F / \partial y = 2y = 0, \qquad (x - \alpha_1)(x - \alpha_2)(x - \alpha_3) = 0,$$

$$-\partial F/\partial x=(x-\alpha_2)(x-\alpha_3)+(x-\alpha_1)(x-\alpha_3)+(x-\alpha_1)(x-\alpha_2)=0.$$

Звідси випливає, що умовою існування сингулярної точки є збіг хоча б двох коренів кубика. Несингулярна крива (2.4), таким чином, не має кратних коренів куба $f(x)$ (тобто кубічне рівняння має три різних корені, або один раціональний і два комплексно-сполучених корені).

Співвідношення між параметрами a й b сингулярної точки легко одержати з (2.3) і (2.4)

$$\partial F/\partial y=2y=0, \qquad \Rightarrow \qquad x^3+ax+b=0,$$

$$\partial F/\partial x=3x^2+a=0.$$

Звідси

$$2x^3=b,$$

$$2xa+3b=0, \Rightarrow \qquad 4a^3+27b^2=0.$$

Для несингулярної кривої (2.4), повинна виконуватися умова:

$$\Delta=-(4a^3+27b^2)\neq 0. \qquad (2.7)$$

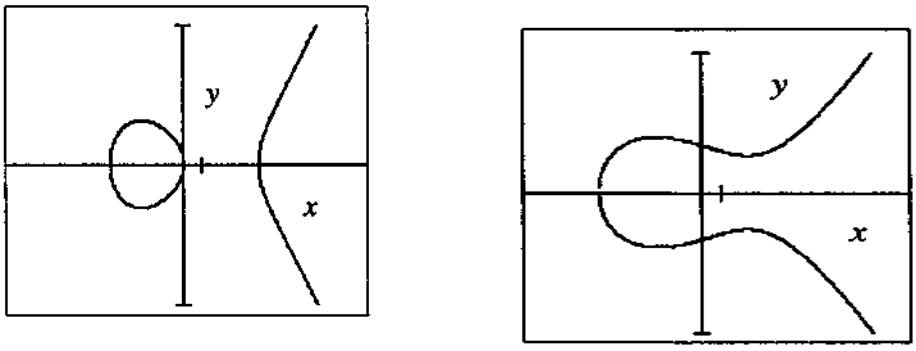
Величину Δ називають дискримінантом кривої.

Характерні графіки несингулярних кривих показані на рисунок 2.1, а, б. Криві є всюди гладкими, тобто в будь-якій їхній точці можна побудувати дотичну. Для порівняння на Рисунок 2.2, а, б наведені сингулярні криві $y^2=x^3-3x+2=(x+2)(x-1)^2$ й $y^2=x^3$ із сингулярними точками (1,0) і (0,0) відповідно (особливості типу «вузол» й «пік»). У цих точках не визначені похідні, що виключає можливість побудови абелевої групи точок.

Важливою властивістю сингулярних кривих, є те, що будь-яка пряма, що проходить через дві різні точки кривої E, перетинає цю криву в єдиній третій точці. Крім того, дотична в будь-якій точці кривої (крім

точки перегину) перетинає її ще в одній точці. Симетрія кривою щодо осі x дозволяє дати наочне визначення зворотної до точки $P = (x_1, y_1)$ точки:

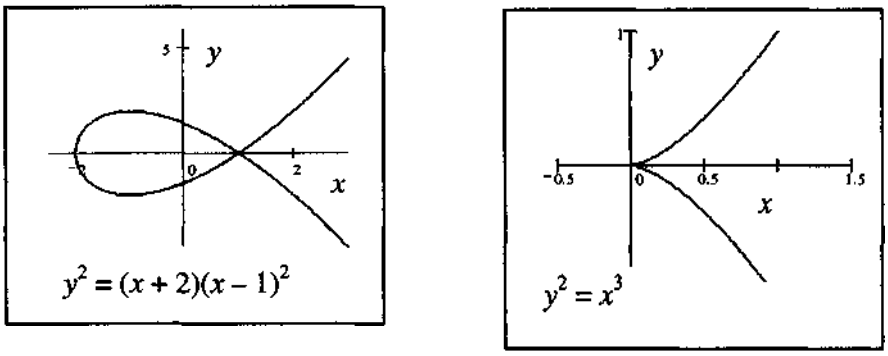
$$-P=(x_1,-y_1). \qquad (2.8)$$



а б

Рисунок 2.1 – Графіки несингулярних еліптичних кривих над полем Q:

а - для трьох раціональних коренів кубічного рівняння; б - для одного раціонального кореня



а б

Рисунок 2.2 – Графіки сингулярних еліптичних кривих над полем Q:

а - для двох кратних дійсних коренів кубічного рівняння(особливість типу «вузол»); б - для трьох кратних дійсних коренів (особливість типу «пік»).

Відзначені властивості приводять до визначення групової операції,

яку традиційно називають додаванням точок еліптичної кривої.

Означення. Сумою двох точок P = (x_1, y_1) і Q = (x_2, y_2) називається точка R = P + Q = (x_3, y_3), зворотня третій точці перетинання ЕК прямою лінією, що проходить через точки P й Q.

Знайдемо координати точки R = P + Q = (x_3, y_3), виразивши їх через координати точок P и Q. При цьому точки P и Q можуть бути різними (Рисунок 2.3, а) або співпадаючими (Рисунок 2.3, б). Відповідно до цього мають місце два випадки.

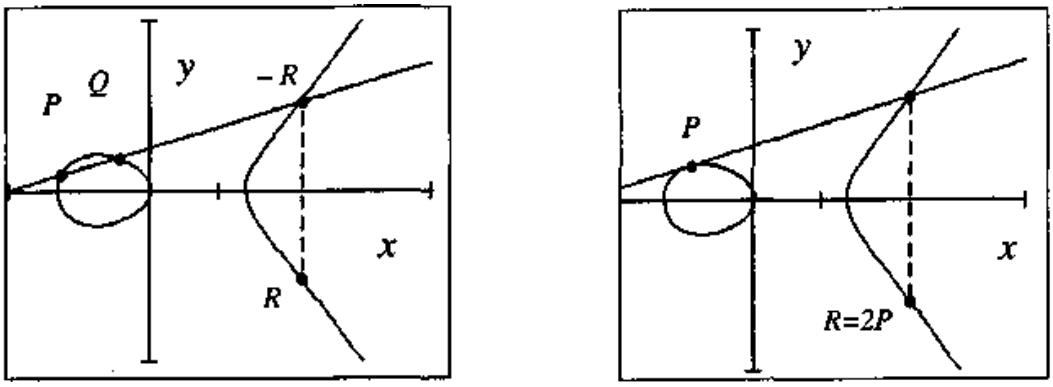
1. P≠±Q. Рівняння прямої лінії, що проходить через точки P и Q (мал. 2.3, а), має вигляд

$$y = \lambda x + \beta; \qquad \lambda = \frac{y_2 - y_1}{x_2 - x_1}; \qquad \beta = y_1 - \lambda x_1. \tag{2.9}$$

Рівняння (2.2) у канонічній формі (2.4) можна переписати

$$f(x, y) = y^2 - x^3 - ax - b = 0. \tag{2.10}$$

Точки перетинання кривої E и прямої (2.9) мають по осі *x*



а б

Рисунок 2.3 – Геометрична інтерпретація додавання двох різних точок P и Q (а) і подвоєння точки P (б)координати x_1, x_2, x_3 точок P, Q й R відповідно.

Оскільки вони є загальними для функцій (2.9) і (2.10), останнє рівняння можна записати у вигляді:

$$(\lambda x+\beta)^2 - x^3 - ax - b = 0\text{, або } -(x-x_1)(x-x_2)(x-x_3) = 0\ .$$

Прирівнюючи в цих кубічних рівняннях коефіцієнти при змінних x^2, одержимо

$$\lambda^2 = x_1 + x_2 + x_3\,. \tag{2.11}$$

Параметр λ прямої (2.9) можна також виразити

$$\lambda = \frac{-y_3 - y_1}{x_3 - x_1}\,.$$

З (2.11) і останнього співвідношення остаточно маємо координати точки $R=P+Q=(x_3,y_3)$:

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2, P \neq Q; \\ y_3 = -y_1 - \lambda(x_3 - x_1), \lambda = \dfrac{y_2 - y_1}{x_2 - x_1} \end{cases} \tag{2.12}$$

2. P=Q, R=2P. У цьому випадку $x_1=x_2$ і параметр λ в (2.12) не визначений. Диференціал функції (2.4)

$$2ydy = 3x^2dx + adx\,,$$

тоді при $x=x_1$ похідна дорівнює параметру ν дотичної $y=\nu x+\beta$ (до кривої в точці P (Рисунок 2.3, б)

$$\nu = \frac{dy}{dx}\Big|_{X=X_1} = \frac{3x_1^2 + a}{2y_1}.$$

Замість (2.12) тепер можна записати координати точки R=2P=(x_3,$в_3$):

$$\begin{cases} x_3 = \nu^2 - 2x_1, P = Q; \\ y_3 = -y_1 - \nu(x_3 - x_1), \nu = \dfrac{3x_1^2 + a}{2y_1} \cdot \end{cases} \quad (2.13)$$

Відзначимо, що формули додавання (2.12) і подвоєння (2.13) справедливі для кривих E над всіма полями, у тому числі й кінцевими, крім полів характеристик 2 й 3. В останньому випадку, як видно з (2.13), редукція по модулю 2 або 3 веде до некоректності формул подвоєння й варто використати інші канонічні рівняння кривих. Крім того, помітимо, що координати додавання й подвоєння точок визначаються за допомогою всіх операцій у поле, тобто додавання, множення й ділення.

Для побудови абелевої групи точок E після введення операції додавання й зворотнього елемента залишилося визначити O групи як

$$P + (-P) = O, \forall P \in E .$$

Якщо провести пряму через точки P и –P, то третя точка перетинання прямо їй *EC* іде в нескінченну точку вздовж осі y (Рисунок 2.3, а). Тому O групи E ще називають «точкою на нескінченності».

Смисл переходу до зворотної точки перетинання прямої і кривої E при визначенні суми R=P+Q стає зрозумілим, якщо виразити, наприклад, точку P як P=R-Q. У цьому випадку пряма проходить через точки R, –Q й –P, а зворотною до цієї третьої точки є точка P. Нескладно переконатися в асоціативності додавання P+(Q+S)=(P+Q)+S. Таким чином, множина точок E замкнута щодо операції додавання, задовільняє властивостям

асоціативності, комутативності, має зворотний елемент і O (одиниця групи), тобто задовольняє всім умовам адитивної абелевої групи.

Приклад 2.1. Нехай $y^2=x^3+1=(x+1)(x^{2-x}+1)$ над полем дійсних чисел. Знайдемо точки R=P+Q й S=2P при P=(0,1) і Q=(-1,0) (Рисунок 2.4).

Згідно (2.12) маємо:

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{0-1}{-1-0} = 1;$$

$$x_3 = 1-(-1)-0 = 2, \quad y_3 = -1-1(2-0) = -3.$$

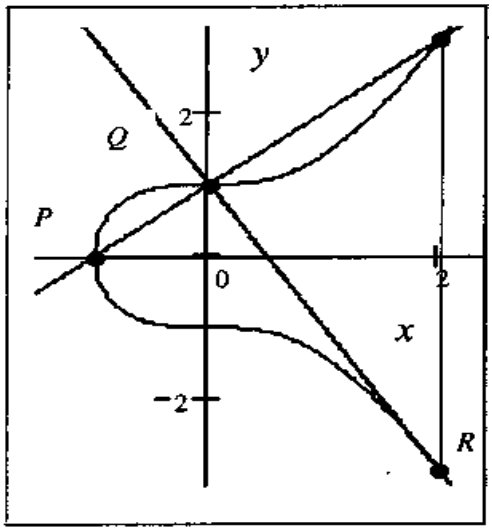


Рисунок 2.4 – Деякі точки кривої $y^2=x^3+1$

Таким чином, точка R=P+Q=(2,-3) (Рисунок 2.4). Тепер за допомогою (2.13) одержимо:

$$\nu = \frac{3x_1^2 + a}{2y_1} = \frac{3\cdot 0 + 0}{2\cdot 1} = 0;$$

$$x_3 = 0-2\cdot 0 = 0; y_3 = -1-0\cdot(0-0) = -1.$$

Отже, точка S=2P=-P=(0,-1).

Розглянемо несингулярну криву:

$$\text{E(Q): } y^2 = x^3 + ax + b,\ \Delta = -(4a^3 + 27b^2) \neq 0, \quad a, b \in Q. \qquad (2.16)$$

Без обмеження загальних положень тут раціональні a й b можна вважати цілими числами. Дійсно, нехай

$$a = \frac{m}{n}; \quad b = \frac{k}{n}.$$

Тоді рівняння (2.16) після множення його на n^6 прийме вигляд:

$$(n^3 y)^2 = (n^2 x)^3 + mn^3 (n^2 x) + mn^5.$$

Позначаючи $Y = n^3 y$, $X = n^2 x$, $A = mn^3$, $B = mn^5$, одержимо рівняння:

$$Y^2 = X^3 + AX + B,$$

аналогічне (2.16), але із цілими коефіцієнтами A и B. Вважатимемо надалі a й b у рівнянні (2.16) цілими числами. Поліноміальні рівняння із цілими коефіцієнтами часто називають діофантовими. Розкладемо поліном у правій частині рівняння на лінійні множники, тобто виразимо куб через корінь α_i

$$y^2 = x^3 + ax + b = (x - \alpha_1)(x - \alpha_2)(x - \alpha_3), \qquad (2.17)$$

причому всі корені різні й справедливі співвідношення:

$$\alpha_1 + \alpha_2 + \alpha_3 = 0, \qquad \alpha_1\alpha_2 + \alpha_1\alpha_3 + \alpha_2\alpha_3 = a,\ \alpha_1\alpha_2\alpha_3 = b.$$

Якщо всі 3 корені куба раціональні, то вони є цілими. Випадок одного раціонального й двох комплексно-сполучених коренів приводить

при цілих a й b до кореню α_i, які називають цілими алгебраїчними числами. З (2.17) випливає, що в раціональних точках E(Q) всі двочлени x-α_i повинні бути квадратами в полі Q.

Згідно (2.12), (2.13) додавання двох раціональних точок дає раціональну точку. Якщо знайдена хоча б одна раціональна точка, то дотична до кривої в цій точці перетинає криву в раціональній точці. Тому будь-яка раціональна точка P може породжувати множину раціональних точок {P,2P,3P,..., k} скінченного або нескінченного порядку. Відповідно до відомої теореми Мордела [1] всі раціональні точки E(Q) породжуються деякою кінцевою множиною таких точок [2, 3, 4]. Для точок кінцевого порядку з E(Q) справедлива теорема Нагеля [2].

Теорема 2.1. Нехай (x_1,y_1) - точка скінченного порядку кривої

$$y^2 = x^3 + ax + b,\ a, b \in Z.$$

Тоді: 1) $x_1,y_1 \in Z$, 2) $y_1 = 0$, 3) $y_1 \mid (4a^3 + 27b^2)$.

Друга властивість таких точок очевидна, тому як при $y_1 = 0$ маємо точки другого порядку. Перша й третя властивості теореми можна проілюструвати на прикладах. Були знайдені точки 2, 3 й 6-го порядків кривої $y^2 = x^3 + 1$ із приклада 2.1. Це точки Q=(-1, 0), P=(0, 1) і R=(2, 3). Як коефіцієнти кубічного рівняння, так і координати точок - цілі числа. Тому що дискримінант рівняння $\Delta = -(4a^3 + 27b^2) = -27,$ y-координати точок P и R ділять Δ, а y-координата точки Q дорівнює 0.

З іншого боку, невиконання кожного із властивостей теореми Нагеля веде до точки нескінченного порядку. Візьмемо точку P=(3,5) із цілими координатами на кривій $y^2=x^3-2$. Тут $\Delta = -(4a^3 + 27b^2) = -108,$ не має в якості дільника y-координату точки P. Послідовне додавання точок P дає x-координати точок $k=(x_k,y_k)$:

$$x1 = 3; x2 = \frac{129}{100}; x3 = \frac{164323}{29241}; x4 = \frac{2340922881}{58675600}; x5 = \frac{307326105747363}{160280942564521}$$

.

Ця послідовність неперіодична й, тому, нескінченна.

Діофантово рівняння $y^2=x^3-2$ вивчалося ще Ферма й Баше в XVII столітті. Вони стверджували (без доказу), що якщо встановлено одне раціональне рішення (x,y), xy≠0, то метод дотичної веде до нескінченного числа рішень.

В 1966 році доказ цього в більш загальному випадку було отримано Л. Морделом [10]. Він довів наступну теорему.

Теорема 2.2. Якщо рівняння $y^2=x^3+b$, де b - вільне від шостих ступенів ціле число, має деяке раціональне рішення (x,y),xy≠0, то при b≠1,-432 існує нескінченне число раціональних рішень цього рівняння.

Число утворюючих точок нескінченного порядку (або число підгруп E нескінченного порядку) називається рангом r еліптичної кривої. Наприклад, розглянута вище крива $y^2 = x^3 + 1$ над полем Q не має раціональних точок, крім точок кінцевого порядку (-1,0), (0,±1) і (2,±3). Таким чином, вона має ранг r=0. Крива $y^2=x^3-2$ над полем Q має одну утворюючу точку нескінченного порядку (3,5) і, відповідно, ранг r=1. У більшості випадків ранг кривих малий і рідко перевищує r =3 (хоча відомі криві з рангом 14).

При обчисленні точок з багаторазовими операціями додавання і подвоєння часто більш продуктивні групові операції не в афінних координатах, а в різного роду проективних координатах. Це дозволяє уникнути обчислення зворотного елемента в полі як самої трудомісткої операції й заощадити тимчасові обчислювальні ресурси.

У стандартних проективних координатах, проективна точка (X:Y:Z), Z≠0, відповідає афінній точці (x=X/Z,y=Y/Z). Однорідне рівняння кривої після заміни змінних і множення на Z^3 приймає вид:

$$E_p:\ Y^2Z = X^3 + aXZ^2 + bZ^3 (\mathrm{mod}\ p) \qquad (2.18)$$

Точка на нескінченності O=(0:1:0) є вже одним з рішень даного рівняння. Зворотна точка тут, як і раніше, визначається інверсією знака Y-координати: -P=(X:-Y:Z). Подібно тому, як в афінних координатах, сумою точок $P_1=(X_1:Y_1:Z_1)$ і $P_2=(X_2:Y_2:Z_2)$ при $P_1 \neq \pm P_2$ називається точка $P_3=P_1+P_2=(X_3:B_3:Z_3)$ координати якої (позначення (mod p) надалі опускаємо для скорочення запису) рівні:

$$\begin{aligned} & X_3 = \nu g; \\ P_1 \neq \pm P_2: \quad & Y_3 = u(v^2 X_1 Z_2 - g) - v^3 Y_1 Z_2; \\ & Z_3 = v^3 Z_1 Z_2, \end{aligned}$$

$$\text{де } v = X_2 Z_1 - X_1 Z_2;\ u = Y_2 Z_1 - Y_1 Z_2;\ g = u^2 Z_1 Z_2 - v^3 - 2v^2 X_1 Z_2.$$

Операцію сумовування однакових точок $P_1=P_2$ називають подвоєнням, а координати точки $P_3=2P_1=(X_3:B_3:Z_3)$ обчислюються наступним чином:

$$\begin{aligned} & X_3 = 2hs; \\ P_1 = P_2: \quad & Y_3 = w(4d - h) - 8s^2 Y_1^2; \\ & Z_3 = 8s^3, \end{aligned}$$

$$\text{де } w = aZ_1^2 + 3X_1^2;\ s = Y_1 Z_1;\ d = sX_1 Y_1;\ h = w^2 - 8d.$$

Наступний вигляд проективних координат - якобіанові координати. До них можна перейти ізоморфним перетворенням координат, при цьому одержимо:

$$(yZ^3)^2 = (xZ^2)^3 + aZ^4(xZ^2) + bZ^6$$

або

$$E_j:\quad Y^2 = X^3 + aXZ^4 + bZ^6 (\mathrm{mod}\ p), \qquad (2.19)$$

де $y=Y/Z^3$, $x=X/Z^2$. Сумою точок $P_1=(X_1:Y_1:Z_1)$ і $P_2=(X_2:Y_2:Z_2)$ при $P_1 \neq \pm P_2$ називається точка $P_3=P_1+P_2=(X_3:B_3:Z_3)$, координати якої визначаються як:

$$
\begin{aligned}
& & X_3 &= -H^3 - 2U_1H^2 + r^2; \\
P_1 \neq \pm P_2: & & Y_3 &= -S_1H^3 + r(U_1H^2 - X_3); \\
& & Z_3 &= HZ_1Z_2,
\end{aligned}
$$

$$
\text{де } U_1 = X_1Z_2^{\,2};\ U_2 = X_2Z_1^{\,2};\ S_1 = Y_1Z_2^{\,3};\ S_2 = Y_2Z_1^{\,3};\ H = U_2 - U_1;
$$

$$
r = S_2 - S_1.
$$

При подвоєнні точки кривої (2.17) одержимо $P_3=2(X_1:Y_1:Z_1)=(X_3:B_3:Z_3)$:

$$
\begin{aligned}
& & X_3 &= T; \\
P_1 = P_2: & & Y_3 &= -8Y_1^{\,4} + M(S - T); \\
& & Z_3 &= 2Y_1Z_1,
\end{aligned}
$$

$$
\text{де } S_1 = 4X_1Y_1^{\,2};\ M = 3X_1^{\,2} + aZ_1^{\,4};\ T = -2S + M^2.
$$

Замість трьох якобіанових координат точки Чудновський в [22] запропонував використати п'ять: $(X:Y:Z:Z^2:Z^3)$. Рівняння кривої описується формулою (2.19), а сума точок $P_1=(X_1:Y_1:Z_1:Z_1^{\,2}:Z_1^{\,3})$ і $P_2=(X_2:Y_2:Z_2:Z_2^{\,2}:Z_2^{\,3})$ при $P_1 \neq \pm P2$ визначається як точка $P_3=P_1+P_2=(X_3:Y_3:Z_3:Z_3^{\,2}:Z_3^{\,3})$, координати Чудновського якої рівні:

$$
\begin{aligned}
& & X_3 &= -H^3 - 2U_1H^2 + r^2; \\
& & Y_3 &= -S_1H^3 + r(U_1H^2 - X_3); \\
P_1 \neq \pm P_2: & & Z_3 &= HZ_1Z_2; \\
& & Z_3^{\,2} &= Z_3^{\,2}; \\
& & Z_3^{\,3} &= Z_3^{\,3},
\end{aligned}
$$

де $U_1 = X_1 Z_2^{\,2};\ U_2 = X_2 Z_1^{\,2};\ S_1 = Y_1 Z_2^{\,3};\ S_2 = Y_2 Z_1^{\,3};\ H = U_2 - U_1;$

$$r = S_2 - S_1.$$

При подвоєнні точки кривої (2.19) одержимо $P_3=2(X_1:Y_1:Z_1:Z_1^{\,2}:Z_1^{\,3})=(X_3:Y_3:Z_3:Z_3^{\,2}:Z_3^{\,3})$:

$$\mathrm{P_1=P_2:}\quad \begin{aligned} X_3 &= T;\\ Y_3 &= -8Y_1^{\,4} + M(S-T);\\ Z_3 &= 2Y_1 Z_1;\\ Z_3^{\,2} &= Z_3^{\,2};\\ Z_3^{\,3} &= Z_3^{\,3}, \end{aligned}$$

де $S_1 = 4X_1 Y_1^{\,2};\ M = 3X_1^{\,2} + aZ_1^{\,4};\ T = -2S + M^2.$

Модифіковані якобіанові координати для рівняння кривої (2.19) містять чотири координати $(X:Y:Z:a^4)$ точки [10]. Сума точок $P_1=(X_1:Y_1:Z_1:a_1^{\,4})$ і $P_2=(X_2:Y_2:Z_2:a_2^{\,4})$ при $P_1 \neq \pm P2$ визначається як точка $P_3=P_1+P_2=(X_3:B_3:Z_3:a_3^{\,4})$, модифіковані якобіанові координати якої рівні:

$$\mathrm{P_1 \neq \pm P_2:}\quad \begin{aligned} X_3 &= -H^3 - 2U_1 H^2 + r^2;\\ Y_3 &= -S_1 H^3 + r(U_1 H^2 - X_3);\\ Z_3 &= HZ_1 Z_2;\\ aZ_3^{\,4} &= aZ_3^{\,4}, \end{aligned}$$

де $U_1 = X_1 Z_2^{\,2};\ U_2 = X_2 Z_1^{\,2};\ S_1 = Y_1 Z_2^{\,3};\ S_2 = Y_2 Z_1^{\,3};\ H = U_2 - U_1;$

$r = S_2 - S_1.$

При подвоєнні точки кривої (2.19) одержимо $P_3=2(X_1:Y_1:Z_1:a_1^{\,4})=(X_3:B_3:Z_3:a_3^{\,4})$:

$$\mathrm{P}_1 = \mathrm{P}_2: \quad \begin{aligned} X_3 &= T; \\ Y_3 &= M(S-T) - U; \\ Z_3 &= 2Y_1 Z_1; \\ aZ_3{}^4 &= 2U(aZ_1{}^4), \end{aligned}$$

де $S_1 = 4X_1 Y_1{}^2; U = 8Y_1{}^4; M = 3X_1{}^2 + aZ_1{}^4; T = -2S + M^2.$

2.2 Оптимальне проектування систем захисту на основі еліптичних кривих

Для оптимального проектування на першому етапі здійснюється огляд з яких компонентів складається система, як вони функціонують. Яким чином здійснюється взаємодія між компонентами системи: загальнодоступні або закриті обчислювальні мережі, які мережні протоколи використовуються і т.д., які системні (у тому числі операційні системи) і прикладні програми використовуються. Словесний опис політики безпеки полягає в завданні простих правил, виконання яких дозволяє контролювати безпеку в системі захисту інформації.

На основі аналізу існуючих формальних моделей безпеки і стандартів безпеки, використовуючи доказовий підхід, необхідно створити адекватну модель розподіленої обчислювальної мережі, що забезпечує необхідний рівень захищеності, а також усуває виявлені недоліки в досліджених моделях і стандартах.

Для даної предметної області, користуючись складеним неформальним описом системи і політикою керування безпекою, виробляється формалізація цього опису і правил безпеки у термінах формальної моделі безпеки.

Виявляються об'єкти і суб'єкти системи. Об'єкти групуються по приналежності до суб'єктів. Формальна модель дозволяють обґрунтувати практичну придатність системи, визначаючи її базову архітектуру і використовувані технологічні рішення при її побудові. У термінах формальної моделі задаються словесні твердження політики безпеки. Таким чином, будується повна і несуперечлива формальна модель системи.

Умови виконання політики інтерпретуються для реальної захищеної системи і реалізуються у виді використання засобів і механізмів інформаційної безпеки і їхнього відповідного настроювання. Для того, щоб система реально була безпечної, необхідне виконання двох умов:

- модель інформаційної взаємодії, що лежить в основі розроблювальної системи захисту – забезпечує гарантовану безпеку;
- інтерпретація моделі безпеки виробляється коректно, необхідно взаємно однозначну відповідність.

Ґрунтуючись на результатах попереднього етапу, виробляється розподіл функцій забезпечення інформаційної безпеки між суб'єктами системи. Для кожного суб'єкта формалізуються правила доступу до приналежним йому об'єктам, далі суб'єкт, користуючись даною інформацією, буде здійснювати контроль виконання правил розмежування доступу.

Суб'єкт виконує наступні операції в системі:

- розмежування (контроль) доступу, фільтрація інформаційних потоків на L і N;
- авторизація, аутентификация, ідентифікація суб'єктів;
- створення захищеного каналу передачі даних;
- авторизована, захищена від модифікації і перехоплення передача даних (шифрування, хеш-функция, цифровий підпис);
- захист від DOS атак (“відмовлення в обслуговуванні”);
- аудит і реєстрацію подій.

Для здійснення обов'язків, що пропонуються, суб'єкт повинний уміти виконувати наступні функції:

- робити обчислення хеш-функции;
- здійснювати симетричне й асиметричне шифрування;
- проставляти електронний підпис;
- генерувати випадкові числа і ключі шифрування;
- фільтрувати запити і спроби атак типу «відмовлення в доступі».

На основі базових будуються похідні функції суб'єктів у системі:

- аутентифікація й авторизація суб'єктів системи, що надсилають запити на доступ до приналежних об'єктів;
- фільтрація запитів на дозволені і заборонені;
- організація безпечної передачі даних від об'єктів до інших суб'єктів системи.

Для реалізації всіх необхідних вимог щодо захисту інформації, можна використати математичний апарат еліптичних кривих.

Незважаючи на вагомі переваги ЕК в криптографічних системах захисту інформації, поряд з цим наявні певні труднощі, які стримують практичне застосування ЕК в цих системах. Вони зумовлені тим, що потребують таких актуальних нижче наведених класів задач:

1) задання простого числа p – модуля перетворення груп точок еліптичної кривої;
2) генерування параметрів ЕК;
3) генерування базових точок на ЕК;
4) знаходження порядку ЕК.

Для успішного генерування параметрів на ЕК над простим полем пропонується нижче наведений підхід, виходячи з необхідності знаходження насамперед відповідних параметрів в залежності від класу задач, а саме:

а) задання простого числа p – модуля перетворення груп точок еліптичної кривої. При цьому беремо до уваги, що просте число p можна розкласти на множники в кільці цілих чисел квадратичного (чи

квадратного?) поля $K = Q(\sqrt{-D})$. Істотно спрощені формули для обчислення кількості точок та їх чисельних значень координат отримуємо при використанні дискримінантів D, які дають кількість класів поля K, що дорівнює одиниці. Доведено, що таке кількість класів забезпечують дискримінанти: $D \in \{2, 3, 7, 11, 19, 43, 67, 163\}$.

Розкласти прості числа можна наступним чином та за умови дотримання таких умов:

$$f_1 = \begin{cases} 2^{255} < p < 2^{256} \\ p = (a + b\sqrt{-D}) * (a - b\sqrt{-D}) = a^2 + b^2 * D, \text{ де } D - \text{квадратичний} \\ \text{лишок по модулю } p \end{cases};$$

(2.20)

б) генерування параметрів ЕК – проведено на підставі алгоритму А.12.4 стандарту IEEE P1363. Ключовим моментом відбору параметрів є визначення критерію відбору, тобто задання цільової функції. При цьому застосовано критерій оцінки гладкості кривої [2,3]:

$$f_2 = \begin{cases} c \cdot b^2 (\mathrm{mod}\, p) = a^3 (\mathrm{mod}\, p) \\ 4a^3 + 27b^2 \neq 0 (\mathrm{mod}\, p) \end{cases}, \qquad (2.21)$$

де $a, b \in GF(p)$;

в) генерування базових точок на ЕК – здійснено згідно з алгоритмом А.11.1 «Знаходження випадкової точки на ЕК» стандарту IEEE P1363. Авторами ж запропоновано критерій відбору базових точок на ЕК з врахуванням наступних вимог:

$$f_3 = \begin{cases} \alpha = x^3 + ax + b (\mathrm{mod}\, p), \text{ де } 0 < x < p \\ \beta^2 \equiv \alpha (\mathrm{mod}\, p) \\ y = (-1)^{\mu} \beta, \text{ де } \mu - \text{випадково згенерована бітова послідовність} \end{cases};$$

(2.22)

г) знаходження порядку ЕК – виконано з врахуванням того, що використовувані у криптографії ЕК вимагають великого простого порядку $\#E(F_p)$ підгрупи та повинні задовольняти умові теореми Хассе $p+1-2\sqrt{p} \le \#E(F_p) \le p+1+2\sqrt{p}$. Згідно з вимогами криптографічних протоколів використано точки порядку n, причому ціле число $h = \frac{\#E(F_p)}{n}$ - кофактор, повинен бути малим $h \le 4$ [32].

Узагальнюючи вище наведене, пропонується такий критерій ефективного відбору порядку на ЕК:

$$f_4 = \begin{cases} p+1-2\sqrt{p} \le \#E(F_p) \le p+1+2\sqrt{p} \\ h = \frac{\#E(F_p)}{n}, h \le 4 \\ \#E(F_p) \ne p \end{cases} . \qquad (2.23)$$

Нехай складена ідеальна система захисту на ЕК, яка задовольняє переліку вимог і критеріям, та в якої рівень відповідності заданої політики безпеки дорівнює 100%, що рівноцінно одиничному рівню забезпечуваного захисту. На практиці подібні системи будуються на підставі наявних механізмів та засобів. В дійсності показники надійності та захищеності системи ніколи не зможуть досягнути 100%. Для виконання заданих вимог використовуються відповідні технології. Ці технології безпеки (криптографічні, тощо) можуть реалізовуватися деякими пристроями, апаратно або програмно, з певною мірою достеменності.

Кожному криптографічному засобу захисту інформації, зокрема і такому, принцип дії якого грунтується на застосуванні ЕК, притаманна низка важливих параметрів. Це, перш за все, вартість і рівень

захищеності реалізованої технології, що реалізується. Серед другорядних критеріїв можна виділити вартість експлуатації, часові характеристики, необхідний обсяг пам’яті та інші.

Завдання підвищення ефективності систем захисту на ЕК в основному пов’язане з продуктивністю криптографічних перетворень, тобто оптимізацією швидкодії обчислення в групі точок на ЕК. Ці технологічні аспекти не регламентуються стандартами і потребують високої кваліфікації розробника як в програмній, так і в апаратній реалізації криптомодулів. В залежності від того, які методи використовуються при виконанні операцій в полі і в групі точок ЕК, швидкість перетворень може змінюватися на один-два порядки. Потрібно також відзначити, що вимоги щодо безпеки та довжини ключа повинні бути достатніми, оскільки швидкодія системи знижується з ростом довжини ключа.

Для оптимального проектування спочатку формується цільова функція безпеки, яка враховує відповідні показники та критерії. В подальшому, на підставі розв’язку оптимізаційної задачі з врахуванням цільової функції безпеки, вибирається такий набір засобів захисту, який оптимально відповідає заданим обмеженням. На практиці оптимальне проектування найчастіше зводиться до розв’язання двох видів задач:

1) знаходження максимально досягнутого рівня захищеності за заданої вартості;

2) вибір системи з мінімальними фінансовими витратами, що забезпечує потрібний рівень захищеності.

Модель прямої задачі вибору варіанту системи, яка забезпечує потрібний рівень захищеності, можна подати у вигляді:

$$\begin{cases} P(f_1, f_2, f_3, f_4) \to \max \\ \sum_{i=1}^{4} c_i \le c_{доп} \\ \sum_{i=1}^{4} t_i \le t_{потр} \\ \sum_{i=1}^{4} M_i \le M_{потр} \\ \sum_{i=1}^{4} W_i \le W_{потр} \end{cases} \quad (2.24)$$

де c_i - вартість здійснення кожної трансакції,

t_i - час, витрачений на одну трансакцію,

$м_i$ - обсяг пам'яті, задіяної для виконання одної трансакції,

W_i - кількість енергоресурсів, витрачених на одну трансакцію.

Модель задачі вибору системи з мінімальною вартістю при досягненні потрібного рівня захищеності заданого параметру можна описати системою рівнянь:

$$\begin{cases} P(f_1, f_2, f_3, f_4) \ge P_{доп} \\ \sum_{i=1}^{4} c_i \to \min \\ \sum_{i=1}^{4} t_i \le t_{потр} \\ \sum_{i=1}^{4} M_i \le M_{потр} \\ \sum_{i=1}^{4} W_i \le W_{потр} \end{cases} . \quad (2.25)$$

В результаті розв'язку задачі оптимального проектування, частіш за все, отримуємо деяку підмножину приблизно рівнозначних за якістю варіантів. Для уточнення цих результатів застосовуємо відповідні методики вибору варіантів розв'язку всередині області компромісів:

а) принцип рівномірності:

- принцип повторення;
- принцип квазірівності,

- принцип максимізму;

б) принцип справедливої поступливості:

- принцип абсолютної поступливості,
- принцип відносної поступливості;

в) принцип виділення одного критерію, яким оптимізуємо.

2.3 Визначення стійкості та продуктивності алгоритмів експоненціювання точок на еліптичних кривих

Найпоширенішою операцією у всіх мережних криптографічних алгоритмах на ЕК є k - кратне додавання точки P, яке позначається kP. Загалом цю операцію можна віднести до скалярного множення, або, звертаючись до термінології мультиплікативної групи, експоненціювання точки ЕК [81].

Для підвищення продуктивності обчислення точки kP багатьма авторами запропоновані різні методи [26]. Підхід до знаходження точки kP може відрізнятися залежно від того, чи є точка P заздалегідь відомою чи невідомою, а також її випадковість. У першому випадку завжди можна користуватися попереднім обчисленням точок, наприклад, 2^iP, які зберігаються в пам'яті. Двійкове подання числа k дозволяє вибрати його з отриманого набору чисел, які в результаті підсумовування утворять точку kP. У другому, більш загальному випадку, всі обчислення доводиться проводити у реальному часі.

Нехай порядок Ord P=r, $[\log_2 r]=L$ і число k подане у двійковій системі:

$$k = \sum_{i=1}^{L-1} k_i 2^i . \qquad (2.26)$$

Розглянемо спочатку основні алгоритми експоненціювання для

невідомої заздалегідь точки P.

Згідно з найпростішим методом, обчислення здійснюються за формулою [26] :

$$kP = \sum_{i=1}^{L-1} k_i 2^i P = \sum_{i=1}^{L-1} k_i P_i , \qquad P_i = 2^i P . \qquad (2.27)$$

Тоді можна застосувати розрахунок зліва направо за допомогою одного з двох нижче наведених алгоритмів [26].

Алгоритм 1.

Вхід: $k=(k_{L-1}, k_{L-2},..., k_0)$, $P \in E$.

Вихід: kP.

1. Присвоюємо початкове значення R=0.

2. for i=L-1 to 0 do

2.1. P←2P.

2.2. if k_i=1 then R←R+P.

3. return R.

Реалізація методу вимагає (L-1) операцій D подвоєння точки й W(k)-1 додавань A, де W(k) - вага Хемінга двійкового вектора, k - кількість одиниць цього вектора. Беручи до уваги, що в середньому кількість одиниць випадкового вектора k дорівнює 1/2, то загальна кількість групових операцій оцінюється значенням 0,5LA+LD.

Цей алгоритм пропонується вдосконалити, якщо ввести додаткову операцію - віднімання точки. Нехай число k=31 у двійковій системі має вагу W(k)=5. Тоді його можна подати у вигляді 2^5-1 з вагою 2. При цьому використано підхід до зниження ваги Хемінга (і, відповідно, кількості групових операцій), що реалізується переходом від двійкового подання числа k до потрійного NAF(k) з коефіцієнтами $k \in \{0,1,-1\}$ (NAF - non-adjacent form). Одна з суттєвих переваг подання NAF(k) полягає у відсутності суміжних пар ненульових елементів, завдяки чому зростає питома вага нульових елементів k_i.

Для розрахунку NAF(k) використовується додатковий проміжний алгоритм.

Проміжний алгоритм.

Вхід: позитивне ціле число k.

Вихід: NAF(k).

1. $i \leftarrow 0$.
2. while $k \geq 1$ do

2.1. if k is odd then: $k_i \leftarrow 2-(k \bmod 4)$, $k \leftarrow k-k_i$;

2.2. else $k_i \leftarrow 0$;

2.3. $k_i \leftarrow k/2, i \leftarrow i+1$.

3. return $(k_0,k_1,...,k_{L-1})$.

Після розрахунку NAF(k) обчислюється точка kP методом зліва на право за допомогою алгоритму 2.

Алгоритм 2.

Вхід: NAF(k), $P \in E$.

Вихід: kP.

1. $R \leftarrow O$.
2. for i=L-1 to 0 do

2.1. $P \leftarrow 2P$.

2.2. if $k_i=1$ then $R \leftarrow R+P$.

2.3. if $k_i=-1$ then $R \leftarrow R-P$.

3. return R.

NAF-подання числа k може виявитися на один біт більше двійкового. У той же час для випадкового k імовірність появи ненульових елементів 1 та -1 знижується від 1/2 до 1/3, тобто в середньому для L-розрядного числа їхня кількість оцінюється значенням L/3. Тоді загальну середню кількість групових операцій додавання A та подвоєння D в алгоритмі 2 можна оцінити сумою (L/3)A + LD.

До однієї з найважливіших характеристик алгоритму належить швидкодія. Як відомо, для виконання різних операцій процесор витрачає різний час. В роботах Черкаського М.В. запропоновано п’ять

характеристик складності алгоритмів для аналізу, синтезу і оптимізації Software/Hardware(SH) моделей, а саме: апаратна, часова, ємнісна, програмна та структурна (таблиця 2.1).

Таблиця 2.1 – Характеристики складності SH – моделей

№ п/п	Характеристика складностей SH – моделей	Аналітичне представлення складностей SH – моделей
1.	Апаратна складність	$A = \lvert X \rvert$
2.	Часова складність	$L = \lvert \max X_i \rvert$
3.	Програмна складність	$P = -F \log_2 \frac{F}{n \cdot m}$
4.	Структурна складність	$S = -E \log_2 \frac{E}{r(r-1)}$

де, X – множина елементів схеми,

$F = \sum_{L} f_l$;

n - кількість входів управління;

m - кількість дискретів часу (часової діаграми);

f_l - кількість сигналів управління l-того фрагменту часової діаграми для вибраного рівня ієрархії побудови апаратних засобів;

L - кількість фрагментів часової діаграми, конфігурації яких не повторюються;

E – кількість елементів матриці інцидентності системи;

R – розмір матриці.

Згідно ємнісного критерію, запропонованого Черкаським М.В., ємнісна складність визначається необхідним об’ємом пам’яті для реалізації алгоритму. Аналіз типових програмних продуктів, які призначені для опрацювання ІП даних, показує, що загальний об’єм пам’яті згідно експертних оцінок може бути виражений аналітично через адитивну функцію:

$$M_з = \sum_{i=1}^{k} M_i \,, \tag{2.28}$$

де M_i - об'єм пам'яті, яка виділяється для реалізації компонентів та окремих модулів алгоритму.

Згідно систематизації: M_1 - об'єм вхідних даних, яке визначається числом вхідних даних та їхньою розрядністю:

$$M_1 = N_1 \cdot n_1, \tag{2.29}$$

M_2 - об'єм пам'яті необхідний для зберігання матриць проміжних результатів:

$$M_2 = N_2 \cdot n_2, \tag{2.30}$$

де N_2 - розмірність матриці,

n_2 - розрядність компонентів матриці.

Аналогічно можна оцінити ємнісну складність інших компонентів алгоритму, включаючи вихідні дані, які можна представити у вигляді діалогових та телекомунікаційних фреймів.

Таким чином, в загальному випадку оцінку ємності алгоритму можна представити у вигляді наступного аналітичного виразу:

$$M_з = \sum_{i=1}^{k} \prod_{l=1}^{m} M_{il} \cdot n_{il}. \tag{2.31}$$

Крім цього, час виконання будь-якого алгоритму можна подати через суму інтервалів часу, що витрачаються на виконання кожної операції розглянутих алгоритмів. Так, подвоєння-додавання в алгоритмі 1 та подвоєння-додавання-віднімання в алгоритмі 2 супроводжуються наведеними в таблиці 2.2 витратами часу на виконання кожної з основних операцій [69].

Таблиця 2.2 – Витрати часу на виконання основних операцій алгоритмів експоненціювання точки кривої

Зміст операції	Витрачений час (такти)
присвоєння ($a = b$)	t_1
присвоєння за модулем ($a = b \bmod c$)	t_2
переведення числа в бінарну систему числення ($k = (k_{L-1}, k_{L-2}, ..., k_0)$)	t_3
додавання ($a = b + c$)	t_4
множення ($a = b \cdot c$)	t_5
ділення ($a = \frac{b}{c}$)	t_6

Загалом можна прийняти таке співвідношення між цими часами:

$$t_1 \le t_2 \le t_3 \le t_4 \le t_5 \le t_6. \qquad (2.22)$$

На підставі наведених у таблиці 2.2 даних та співвідношення (2.32) можна побудувати математичні моделі загального часу виконання кожного алгоритму.

Для пошуку кратної точки на основі розрахунку зліва направо згідно з алгоритмом 1 використано бінарне зображення k. Тоді відбувається побітове зчитування послідовності $k = (k_{L-1}, k_{L-2}, ..., k_0)$, де L - довжина даної послідовності. Крім цього, в даному алгоритмі на кроці 2.1 виконується операція подвоєння точки. Необхідно зазначити, що при додаванні двох точок $P(x_1, y_1)$ та $Q(x_2, y_2)$ координати суми визначаються так:

$$x = \left(\frac{y_1 - y_2}{x_1 - x_2} \right) - x_1 - x_2, \qquad (2.33)$$

$$y = -y_1 + \left(\frac{y_1 - y_2}{x_1 - x_2} \right) \cdot \left(x_1 - x_2 \right).$$

Беручи до уваги, що операція віднімання вимагає стільки ж часу, що і додавання, то час, витрачений на виконання операції додавання двох точок, можна подати у вигляді [69]:

$$t_0 = t_4 + t_4 + t_6 + t_4 + t_4 + t_4 + t_4 + t_6 + t_4 + t_5 + t_4 = 8 \cdot t_4 + t_5 + 2 \cdot t_6. \qquad (2.34)$$

Звідси отримуємо, що для виконання алгоритму 1 потрібно витратити час [69]:

$$T_1(k) = t_3 + t_1 + \sum_{i=L-1}^{0} t_0 + \sum_{\substack{i=L-1 \\ k_i=1}}^{0} t_0 = t_3 + t_1 + L \cdot t_0 + W(k) \cdot t_0, \qquad (2.35)$$

де $W(k)$ - вага Хемінга двійкового вектора k.

Оскільки L - довжина бінарної послідовності $k = (k_{L-1}, k_{L-2}, ..., k_0)$, то $L = \lceil \log_2 k \rceil$. Крім цього, в загальному випадку можна вважати, що вага Хемінга $W(k) = \frac{L}{2}$ для найсприятливішого випадку для проведення атаки. Звідси та зі співвідношень (2.32) і (2.33) випливає [69]:

$$T_1(k) = t_3 + t_1 + \lceil \log_2 k \rceil \cdot t_0 + \frac{\lceil \log_2 k \rceil}{2} t_0 = t_3 + t_1 + \frac{3\lceil \log_2 k \rceil}{2} t_0 = \\ = t_3 + t_1 + \frac{3}{2} \lceil \log_2 k \rceil \cdot \left(t_4 + t_5 + 2t_6 \right) \qquad (2.36)$$

Для алгоритму 2 подвоєння-додавання-віднімання використовується потрійне подання числа k NAF(k) з коефіцієнтами k$\in$ {0,1,-1}. Перехід від

бінарного до NAF-подання здійснюється за проміжним алгоритмом. Додатково в проміжному алгоритмі на кроці 2.1 перевіряється парність числа k. Тому загальний час виконання даного алгоритму буде залежати від k. Витрачений на NAF-подання час можна обчислити так [69]:

$$\begin{aligned} T_2(k) &= t_1 + \sum_{\substack{i=k \\ k_i \neq 2}}^{1}(t_2 + t_4 + t_4) + \sum_{\substack{i=k \\ k_i = 2}}^{1} t_1 + t_6 + t_4 = \\ &= t_1 + \frac{k}{2}(t_2 + 2t_4) + \frac{k}{2}t_1 + t_6 + t_4 = \\ &= (\frac{k}{2} + 1)t_1 + \frac{k}{2}t_2 + (k+1)t_4 + t_6 \end{aligned} \quad . \qquad (2.37)$$

На підставі співвідношень (2.32) та (2.35), а також імовірності появи ненульових елементів у $NAF(k)$, час, витрачений на виконання алгоритму подвоєння-додавання-віднімання (алгоритм 2), такий [69]:

$$\begin{aligned} T_3(k) &= T_2(k) + t_1 + \sum_{i=L-1}^{0} t_0 + \sum_{\substack{i=L_{NAF}-1 \\ k_i = 1}}^{0} t_0 + \sum_{\substack{i=L_{NAF}-1 \\ k_i = -1}}^{0} t_0 = \\ &= t_1 + \frac{k}{2}(t_2 + 2t_4) + \frac{k}{2}t_1 + t_4 + t_6 + t_1 + L_{NAF} \cdot t_0 + \frac{L_{NAF}}{3}t_0 + \frac{L_{NAF}}{3}t_0 = \\ &= \left(\frac{k}{2} + 2\right) \cdot t_1 + \frac{k}{2} \cdot t_2 + \left(k + 1 + \frac{40}{3}L_{NAF}\right) \cdot t_4 + \frac{5}{3}L_{NAF} \cdot t_5 + \\ &+ \left(\frac{10}{3}L_{NAF} + 1\right) \cdot t_6 \end{aligned} \quad , \qquad (2.38)$$

де L_{NAF} - довжина $NAF(k)$.

З аналізу співвідношень (2.34) та (2.36) випливає, що загальний час виконання алгоритмів 1 та 2 залежить лише від значення числа k. Тому можна оцінити продуктивні характеристики алгоритмів, що дозволяє визначити кращий з них для застосування з точки зору швидкодії.

На рисунку 2.5 зображено залежність T_1 та T_3 від k для довжини

бітової послідовності $L = 1..256$. Слід зазначити, що для даної оцінки використовуються значення часів $t_1 = 1$, $t_2 = 1{,}5$, $t_3 = 1{,}6$, $t_4 = 1{,}8$, $t_5 = 10$, $t_6 = 12$ (в тактах).

Аналіз наведених на рисунку 2.5 залежностей показує, що вища продуктивність притаманна алгоритму 1. Проте і продуктивність алгоритму подвоєння-додавання-віднімання (алгоритм 2) цілком задовольняє вимогам сучасного користувача.

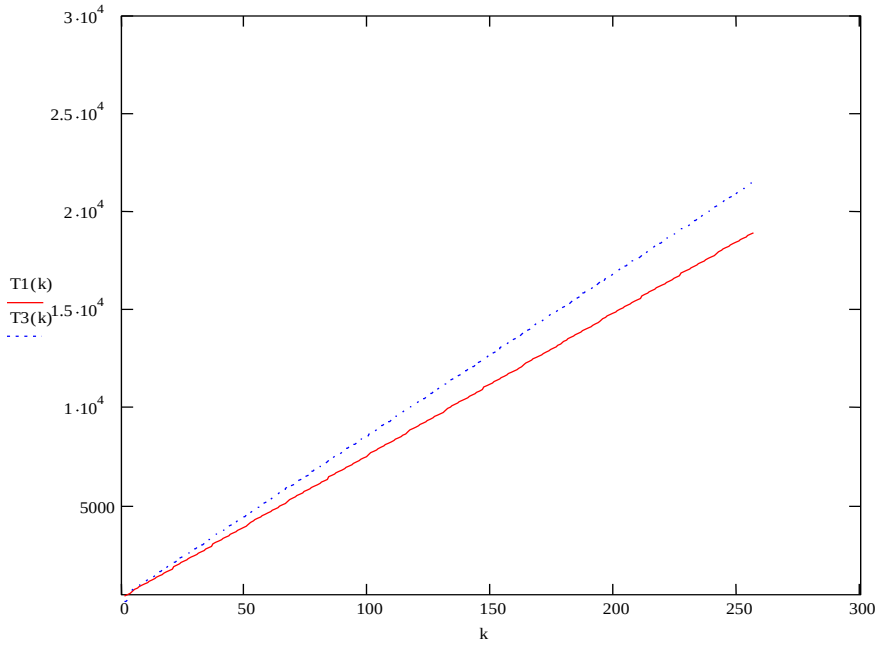


Рисунок 2.5 – Оцінка продуктивності досліджуваних алгоритмів 1 та 2

Час виконання криптографічних операцій залежить не лише від ефективності реалізації конкретного алгоритму, але й також, інколи суттєво, від вхідних даних. Особливо така кореляція сильно проявляється для додавання чи множення точок на ЕК. Загалом, ці криптографічні операції є обчислювально складними і для підвищення продуктивності виконання процедури шифрування повідомлення чи формування ЕЦП доцільно застосувати спеціальні алгоритми, які базуються на використанні інформації про кількість бітів у ключі шифрування. Такий підхід дозволяє пришвидшити виконання криптографічних операцій завдяки обходу виконання деяких операцій алгоритму для нульових бітів ключа. Звідси очевидно, що завжди можна виявити певну кореляцію між

кількістю одиничних бітів ключа та часом виконання такого алгоритму. Володіння криптоаналітиком саме такою інформацією дозволяє висунути гіпотезу щодо кількості одиничних та нульових бітів у приватному ключі, кількісним еквівалентом якого може бути вага Хемінга. В подальшому це дозволяє здійснити атаку повного перебору в певному піддіапазоні ключового простору, для чого вимагаються значно менші обчислювальні ресурси. Таким чином, оцінка кореляції часу виконання криптографічних операцій з вагою Хемінга щодо ключа дозволяє криптоаналітику зменшити обчислювальну складність атаки часового аналізу на систему захисту інформаційних ресурсів.

Отже, для дослідження стійкості алгоритмів 1 та 2 необхідно встановити залежність часу виконання відповідного алгоритму від ваги Хемінга. Тоді слід задати залежність згаданого часу від ваги Хемінга.

З рівності (2.33) та аналізу алгоритму 1 випливає, що:

$$T_1(k) = t_3 + t_1 + \left(\log_2 k - W(k)\right)\left(t_4 + t_5 + 2t_6\right). \qquad (2.39)$$

На основі результатів проведеного моделювання отримано залежність часу виконання алгоритму 1 від ваги Хемінга (рисунок 2.6) відповідно для довжин ключа $L = 64$ біт, $L = 128$ біт та $L = 256$ біт.

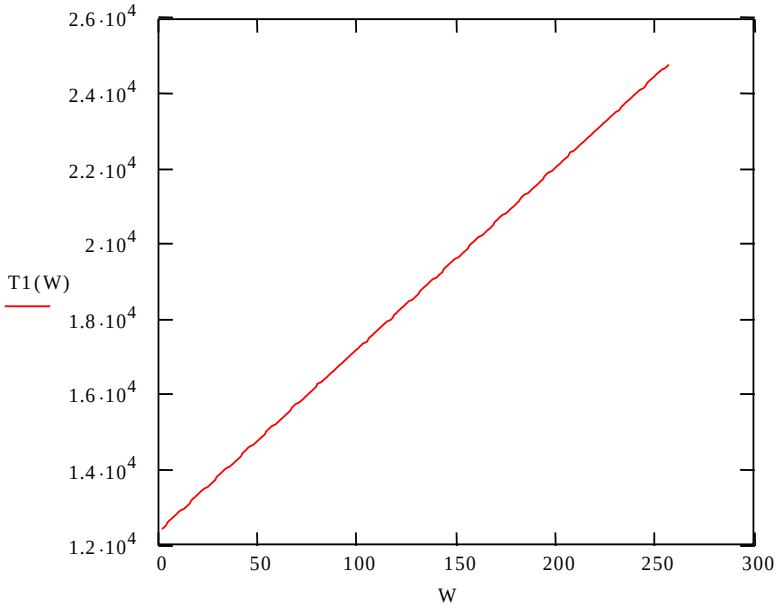


Рисунок 2.6 – Залежність часу виконання алгоритму подвоєння-додавання від ваги Хемінга для $L = 256$ біт

З аналізу зображеної на рисунку 2.6 залежності випливає, що алгоритм подвоєння-додавання (алгоритм 1) характеризується лінійною залежністю часу від ваги Хемінга. Це означає, що вимірявши час здійснення множення точки ЕК, криптоаналітик може знайти вагу Хемінга – а цього є достатньо для знаходження значення k.

Залежність часу виконання алгоритму 2 від ваги Хемінга пропонується задати таким чином [66]:

$$T_3(k) = \left(\frac{k}{2} + 2\right) \cdot t_1 + \frac{k}{2} \cdot t_2 + \left(\frac{k}{2} + 1\right) \cdot t_4 + L_{NAF} \cdot t_0 + W_1(k) \cdot t_0 + W_{-1}(k) \cdot t_0, \quad (2.40)$$

де $W_1(k)$ - кількість одиниць у $NAF(k)$-поданні, а $W_{-1}(k)$ - кількість одиниць у даній рівності. Враховуючи імовірність появи нулів у $NAF(k)$, що дорівнює $\frac{1}{3} L_{NAF}$, можна в загальному вважати, що:

$$W_1(k) = \frac{2}{3} L_{NAF} - W_{-1}(k). \quad (2.41)$$

На підставі співвідношень (2.39) та (2.40) можна побудувати графік залежності часу виконання алгоритму подвоєння-додавання-віднімання від кількості одиниць у $NAF(k)$, відповідно для $L_{NAF} = 64$ біт, $L_{NAF} = 128$ біт та
$L_{NAF} = 256$ біт (Рисунок 2.7).

Слід зазначити, що кількість одиниць у $NAF(k)$, аналогічно як і вага Хемінга в бінарному поданні числа k, дає достатньо інформації криптоаналітику щодо заповнення $NAF(k)$, тобто дозволяє правильно визначити значення k. Звідси та з аналізу залежностей, наведених на

рисунку 2.7, можна зробити висновок, що алгоритм подвоєння-додавання-віднімання (алгоритм 2) є абсолютно стійким до часового аналізу.

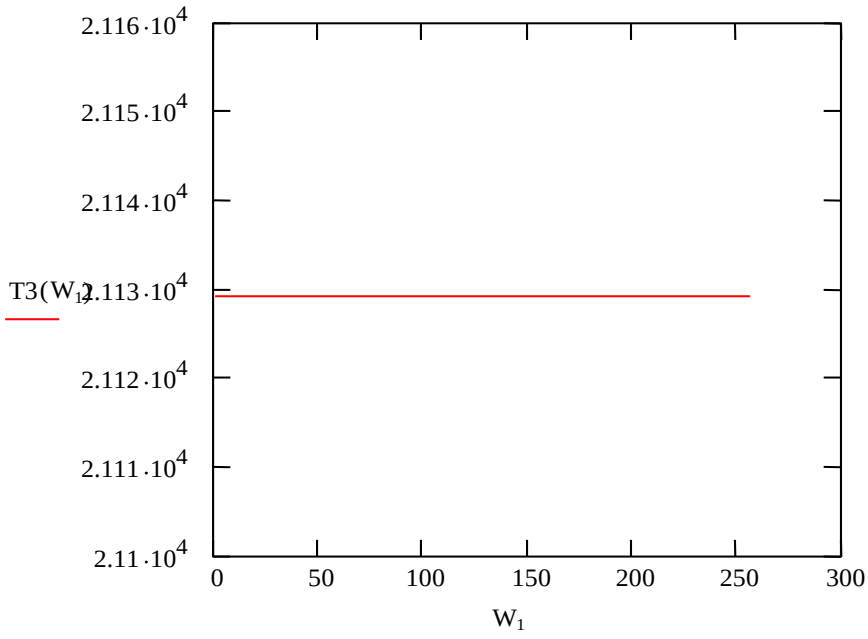


Рисунок 2.7 – Залежність часу виконання алгоритму подвоєння-додавання-віднімання від кількості одиниць при $L_{NAF} = 256$ біт

Отже, враховуючи продуктивність та стійкість алгоритму 2 до часового аналізу, рекомендується застосовувати його на практиці для скалярного множення базових точок ЕК.

3 ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ АПАРАТНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ ЗАХИСТУ НА ЕЛІПТИЧНИХ КРИВИХ

3.1 Елементи апаратної реалізації алгоритму експоненціювання точок на ЕК

3.1.1 Реалізація на мікроконтролері алгоритму експоненціювання точок на еліптичних кривих

Як видно з опису алгоритму, для пошуку кратної точки потрібно виконувати додавання, віднімання та подвоєння точки на ЕК. Додавання точки реалізовується згідно з алгоритмом додавання точок на ЕК в афінних координатах по модулю. Це алгоритм Р1363 А 10.1 [60]. Подвоєння точки – це фактично додавання її самої до себе ($P = P + P$). Згідно з вищенаведеним стандартом, у випадку використання модулярного представлення еліптичної кривої віднімання точки виглядає як додавання точки з зміною знаку біля ординати: $R = R - P(x, y) = R + P(x, -y)$. Алгоритм додавання точки згідно з стандартом Р1363 А.10.1:

Вхід: просте $p > 3$; коефіцієнти a, b для еліптичної кривої E: $y^2 = x^3 + ax + b$ по модулю p; точки $P_0 = (x_0, y_0)$ і $P_1 = (x_1, y_1)$ на E.

Вихід: точка $P_2 := P_0 + P_1$.

1. If P_0 = O then output $P_2 \leftarrow P_1$ and stop
2. If P_1 = O then output $P_2 \leftarrow P_0$ and stop
3. If $x_0 \neq x_1$ then

3.1 set $\lambda \leftarrow (y_0 - y_1) / (x_0 - x_1) \bmod p$

3.2 go to step 7

4. If $y_0 \neq y_1$ then output $P_2 \leftarrow$ O and stop
5. If $y_1 = 0$ then output $P_2 \leftarrow$ O and stop

6 Set $\lambda \leftarrow (3x_1^2 + a) / (2y_1) \bmod p$

7. Set $x_2 \leftarrow \lambda^2 - x_0 - x_1 \bmod p$
8. Set $y_2 \leftarrow (x_1 - x_2)\ \lambda - y_1 \bmod p$
9. Output $P_2 \leftarrow (x_2, y_2)$

Дані алгоритми було реалізовано в вигляді програми, написаної на мові С. Для прошивки в мікроконтролер використовувся компілятор з мови С PICC.exe в складі програмного пакету MPLAB IDE v7.50. Цей продукт дозволяє компілювати програми, прошивати їх код в мікроконтролер (при наявності спеціального апаратного забезпечення) та проводити симуляцію роботи пристроїв.

Текст програми мікроконтролера:

```
#include <16f628.h>
#use delay(clock=4000000)
#define startpin 48

typedef struct {signed long x; signed long y;} ecpoint;

unsigned long k, i = 0;
signed long a, mp, lambda;
ecpoint R, res, O, p2inv, t, PointEC;
signed long tmp[3];
byte kbin[16];
int odd(long a)
{
    return (a % 2);

}
```

```
int ecequal(ecpoint p1, ecpoint p2)
{
  return !((p1.x == p2.x) || (p1.y == p2.y));
}

void eccopy(ecpoint p1, ecpoint p2)
{
  p1.x = p2.x;
  p1.x = p2.y;
}

void ecaddpoint(ecpoint p1, ecpoint p2)   //res = p1 + p2
{
  if (ecequal(p1, O))
  {
    eccopy(p2, res);
    return;
      }

      if (ecequal(p2, O))
      {
              eccopy(p1, res);
              return;
      }
```

```
    if (p1.x == p2.x)
    {
        lambda = ((p1.y - p2.y)/(p1.x-p2.x))%mp;
        goto step7;
    }

    if ((p1.y == p2.y) || (p2.y == 0))
    {
        eccopy(O, res);
        return;
    }

    lambda = ((3*p2.x*p2.x+a)/(2*p2.y))%mp;

step7:;
    res.x = (lambda*lambda-p1.x-p2.x)%mp;
    res.y = ((p2.x-res.x)*lambda-p2.y)%mp;
}

void ecsubpoint(ecpoint p1, ecpoint p2)   //res = p1 - p2
{                                 //res = p1 + p2(x, -y)
    eccopy(p2, p2inv);
    p2inv.x = -p2inv.x;
    ecaddpoint(p1, p2inv);
```

```
}

void NAF(long k)                    //kbin = NAF(k)
{
    i = 1;
    kbin[0] = 0;
    tmp[2] = k;
    while (tmp[2] > 0)
    {
        kbin[0]++;
        if (odd(k))
        {
            kbin[i] = tmp[2] % 4;
            kbin[i] = 2 - kbin[i];
            tmp[2] -= kbin[i];
        }
        else
            kbin[i] = 0;
        tmp[2] /= 2;
        i++;
    }
}

void ecmulpoint(long k, ecpoint p1)
{
```

```
    eccopy(O, R);
    eccopy(p1, t);
    NAF(k);
    if (kbin[0] == 0) return;

    for (i = kbin[0] - 1; i != 0; i--)
    {
        ecaddpoint(t, t);                    //res = 2*p1
        eccopy(res, t);                      //p1 = 2*p1
        if (kbin[i] == 1)
            ecaddpoint(R, t);
        else
        if (kbin[i] == -1)
            ecsubpoint(R, t);
        eccopy(res, R);
    }
}

void output_long(long a)
{
    short int b, c;
    for (i = 0; i < 15; i++)
    {
        b = a % 2;
        a /= 2;
```

```
            c = i+startpin;
            if (b == 1) output_high(c);
            else output_low(c);
        }
        c = i+startpin;
        if (a > 0) output_low(c);
        else output_high(c);
    }

    void main(void)
    {
        mp = 32749;
        a = 2364;
        PointEC.x = 7899;
        PointEC.y = 2686;
        k = 12;
        ecmulpoint(k, PointEC);
        output_long(res.x);
        delay_ms(2);
        output_long(res.y);
        sleep();
    }
```

В даній програмі використовується побітовий вивід 16-бітного числа з використанням обох портів вводу-виводу мікроконтролера

(функція output_long). При настанні певної події мікроконтролер «просинається» і виконується функція main. Як видно з тексту програми, ця функція здійснює одну операцію множення точки, виводить координати результуючої точки та переводить мікроконтролер в сплячий режим. Вхідні дані зафіксовано для того, щоб отримати реальну картину швидкодії мікроконтролера. Затримку між виводом координати x та y введено для того, щоб дані не втратилися (пристрій, який зчитує дані з мікроконтролера має дві мікросекунди для того, щоб зчитати інформацію).

З допомогою симулятора в складі програмного пакету MPLAB IDE v7.50 було проведено дослідження роботи віртуального мікроконтролерного пристрою, які показали правильність його роботи. Результати досліджень наведено в п.3.3.

3.2 Програмна реалізація алгоритму експоненціювання точок на ЕК

Згідно з вищенаведеними алгоритмами (див. 3.1) був розроблений програмний продукт Multiplication, який і здійснює пошук кратних точок на еліптичній кривій з заданими параметрами. Головне вікно програми наведено на рисунку 3.1, текст програми – в додатку А.

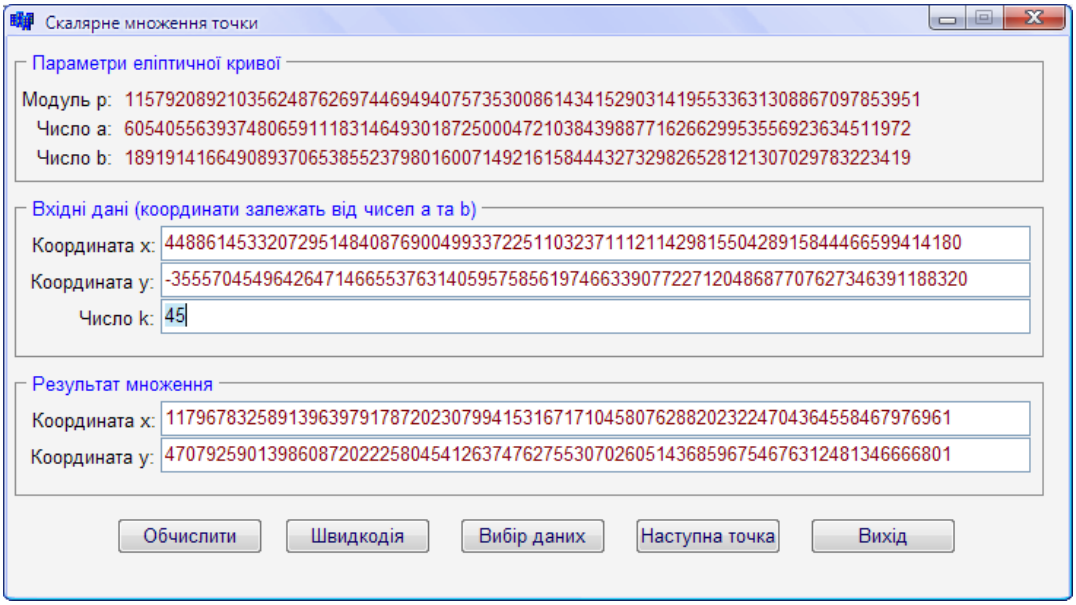


Рисунок 3.1 – Головне вікно програми скалярного множення точки на ЕК

Параметри еліптичної кривої зберігаються в файлі Numbers.txt. Модуль p еліптичної кривої однаковий для всіх параметрів та зберігається в файлі ModulusP.txt. В файлі Numbers.txt також зберігаються координати базової точки на еліптичній кривій з параметрами *a* та *b*, ці координати однозначно визначаються параметрами еліптичної кривої, тому користувач не може їх змінити. Фактично єдине, що задає користувач програмі – це множник *k*. Після натиснення кнопки «Обчислити» програма шукає скалярну точку і видає афінні координати результуючої точки. Як видно з рисунку 3.1, головне вікно програми має ще кілька кнопок:

- Швидкодія – визначення часових характеристик роботи програми.
- Вибір даних – вибір вхідних параметрів *a, b, P(x, y)* (рисунку 3.2).
- Наступна точка – прочитати наступний запис з бази даних параметрів.
- Вихід – завершення роботи програми.

Тепер розглянемо можну кнопку детальніше. Натиснення на кнопку «Швидкодія» покаже вікно визначення часових характеристик програми (Рисунок 3.3).

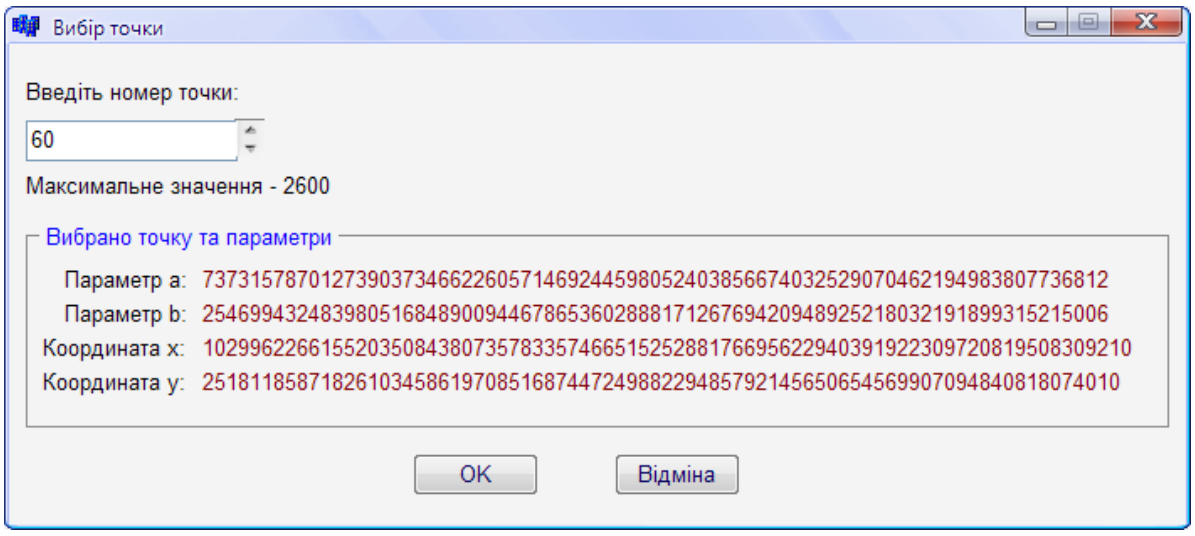


Рисунок 3.2 – Вікно вибору вхідних даних

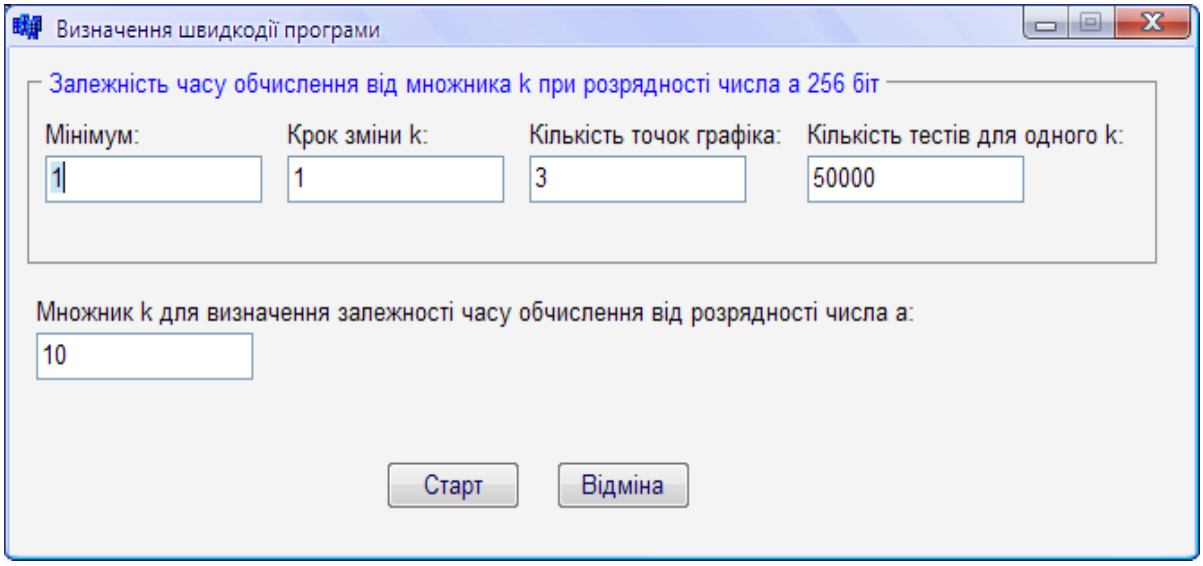


Рисунок 3.3 – Вікно визначення часових характеристик програми

Як видно з рисунку, досліджується зміна часових характеристик роботи програми в залежності від множника *k* та розрядності (бітової довжини) параметра еліптичної кривої *a*. Параметр *b* в дослідженні участі не бере, оскільки, як видно з вищенаведених алгоритмів, він є неважливим для пошуку скалярної точки. Варто зазначити, що час виконання операції множення дуже малий (кілька мікросекунд). Тому для визначення часу виконання операції множення використовується

механізм накопичення часу, тобто для кожного тестового набору параметрів проводиться багато операцій множення (наприклад, 50 тисяч), і отримана часова затримка ділиться на кількість операцій. Це дозволяє досить точно визначити час виконання однієї операції множення. Крім того, такий механізм дозволяє нівелювати неточності, які вносить сама операційна система. Наприклад, при виконанні якоїсь операції множення операційна система може зайняти процесор і час виконання множення буде більший, ніж реальний. Проте, якщо множення виконується 50000 раз, то така неточність буде практично зведена до нуля. Отже, розглянемо рисунок 3.3. Як видно з цього рисунку, для визначення залежності часу множення від множника *k* потрібно ввести наступні параметри:

- Мінімум – початкове значення параметра *k*.
- Крок зміни – число, яке додається до множника на кожному кроці табулювання.
- Кількість точок графіка – визначає скільки раз потрібно збільшувати множник. Бажано задавати не більше 50 точок, інакше графік може бути нечітким (кілька точок можуть злитися в одну і т.п.).
- Кількість тестів для одного *k* – скільки операцій множення буде проведено для кожного *k* та для кожного *a*. Тобто, цей параметр є спільним для визначення обох часових залежностей.

Після вводу всіх необхідних параметрів потрібно натиснути кнопку «Старт». Це запустить на виконання серію тестів. Прогрес їх виконання виводиться на екран (Рисунок 3.4). Після виконання всіх тестів програма записує часові характеристики своєї роботи у файли t_of_k.m та t_of_bitlen.m (Рисунок 3.5). Потім ці файли потрібно відкрити в середовищі Matlab та виконати їх. При проведенні тесту на залежність часу виконання від розрядності коефіцієнту a дані беруться з файлу SpeedTest.txt.

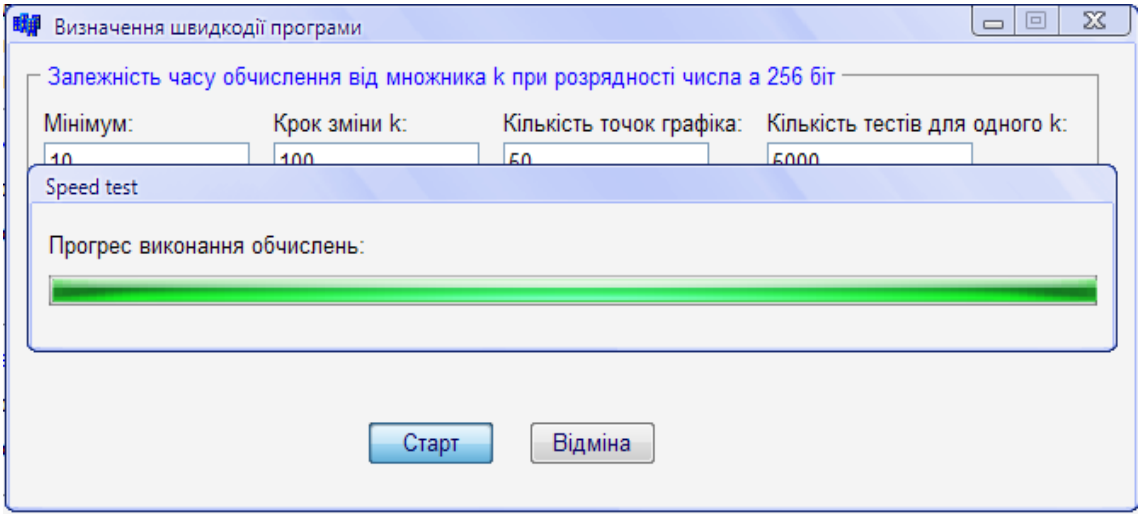


Рисунок 3.4 – Прогрес виконання обчислень

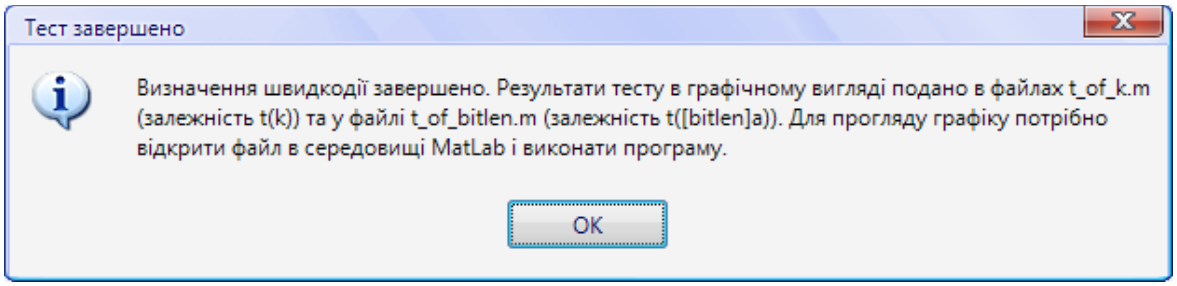


Рисунок 3.5 – Повідомлення про завершення тесту

Текст основних модулів програми подано в додатку А. При створенні програмного продукту було використано бібліотеку *lip.c* для роботи з великими числами. Змінна, яка містить число в форматі цієї бібліотеки оголошується так: verylong a. Нижче подано опис функцій цієї бібліотеки, які використовувались при написанні текстів програми:

- zsread(char *str, verylong *a) – зчитування числа з стрічки;
- zswrite(char *str, verylong a) – запис числа в стрічку (в десятковому форматі);
- zcompare(verylong a, verylong b) – порівняння двох чисел. Результат – 0, якщо числа рівні, 1 якщо a>b та -1 якщо a<b;
- zcopy(verylong a, verylong *b) – копіює число a в число b;
- zsub(verylong a, verylong b, verylong *r) – виконує операцію r=a-b;

- zdiv(verylong a, verylong b, verylong *r, verylong *q) – операція ділення. Тут r=a/b, а в q записується остача від ділення;
- zmod(verylong a, verylong p, verylong *r) – операція взяття модуля p з числа a. Результат заноситься в змінну r;
- zsq(verylong a, verylong *r) – операція r=a*a;
- zsmul(verylong a, long b, verylong *r) – виконує те саме що й zmul, але один з множників задається як звичайне число;
- znegate(verylong *a) – змінює знак числа a на протилежний (a = -a);
- zodd(verylong *a) – перевіряє непарність числа a. Якщо воно непарне, то результат – 1, інакше – 0;
- zsmod(verylong a, long p, verylong *r) – те саме, що й zmod, але модуль – звичайне число;
- zintoz(long a, verylong *r) – переводить звичайне число в формат verylong;
- zsdiv(verylong a, long b, verylong *r) - те саме, що й zdiv, але дільник – звичайне число, результат функції – остача від ділення;
- zstart() – ініціалізація бібліотеки lip.c;
- zzero(verylong *a) – запис в число a нульового значення.

Для вирішення задачі пошуку кратних точок потрібно реалізувати три операції – додавання (віднімання) двох точок на еліптичній кривій, множення точок та пошук NAF(k). Для полегшення роботи в програмі було введено новий тип даних – ecpoint. Цей тип описується наступним чином:

typedef struct {verylong x; verylong y;} ecpoint;

Як видно з запису, тип ecpoint є структурою з двома полями – x та y, кожне типу verylong. Цей тип визначає точку на еліптичні кривій і всі функції, які оперують точками, використовують саме тип ecpoint в якості аргументів. Крім вище вказаних функцій для реалізації математичних операцій з точками еліптичних кривих, було введено цілий ряд

допоміжних функцій. Список функцій, які описані в програмі, подано нижче:

- ansitoz(AnsiString astr, verylong &num) – переведення числа з стрічки формату AnsiString в числовий формат verylong;
- refreshdata() – оновлення інформації на екрані;
- closedata(FILE *fstream) – закриває базу даних;
- reopendata(FILE *fstream) – відкриває базу даних і встановлює її покажчик на останній запис, що використовувався;
- nextrec(FILE *fstream) – встановлює покажчик бази даних на наступний запис;
- getnextdata(FILE *fstream, verylong &coef_a, verylong &coef_b, verylong &coor_x, verylong &coor_y) – зчитує з бази даних наступний запис в змінні-аргументи;
- ecequal(ecpoint p1, ecpoint p2) – вертає 0, якщо точки p1 та p2 рівні (мають однакові координати), інакше результат – 1;
- eccopy(ecpoint p1, ecpoint &p2) – копіює точку p1 в точку p2;
- ecaddpoint(ecpoint p1, ecpoint p2) – здійснює додавання точок p1 та p2. Результат додавання заноситься в глобальну змінну res;
- ecsubpoint(ecpoint p1, ecpoint p2) – здійснює операцію res = p1(x1,y1)+p2(x2, -y2);
- NAF(verylong k) – переводить число k в NAF-формат. Результат заноситься в масив kbin;
- ecmulpoint(verylong k, ecpoint p1) – здійснює скалярне множення точки на число (res = k*p1).

Крім модуля main.cpp, який було описано вище, програма містить ще ряд модулів. Це: DataChoose.cpp (відповідає за вікно вибору даних); Progress.cpp (виводить прогрес виконання тестів при визначенні часових характеристик); Speed.cpp (визначення часових характеристик роботи програми).

3.3 Дослідження часових та продуктивних параметрів апаратної та програмної реалізації на ЕК

В даній роботі досліджувалася швидкодія програмної та апаратної реалізації пошуку скалярної точки на еліптичній кривій. Програмна та апаратна реалізація відрізняються розрядністю чисел, з якими вони оперують – мікроконтролер обробляє числа розрядністю не більше 16 біт, в той час як програма оперує 256-бітними числами. З іншого боку, час виконання програми замірювався на комп'ютері з процесором AMD Sempron 2400+ з тактовою частотою 1659 MHz. Частота роботи мікроконтролера – 40 мегагерц. Тобто продуктивність апаратної реалізації є завідомо нижчою, ніж програмною. Проте, як було сказано в п. 1.2, апаратна реалізація шифрування має багато переваг над програмною.

Після запуску програми Multiplication.exe було запущено підпрограму визначення її швидкодії. Часові характеристики роботи програмної реалізації подано на рисунку 3.6 (залежність часу виконання від множника k) та на рисунок 3.7 (залежність часу виконання від розрядності числа a). З огляду на те, що в апаратній реалізації всі числа 16-бітні, немає змісту досліджувати залежність часу виконання операції множення мікроконтролером від розрядності коефіцієнту a.

Після визначення продуктивності роботи мікроконтролера було систематизовано отримані дані і побудовано графік залежності часу виконання операції множення мікроконтролером від множника k (рисунок 3.8).

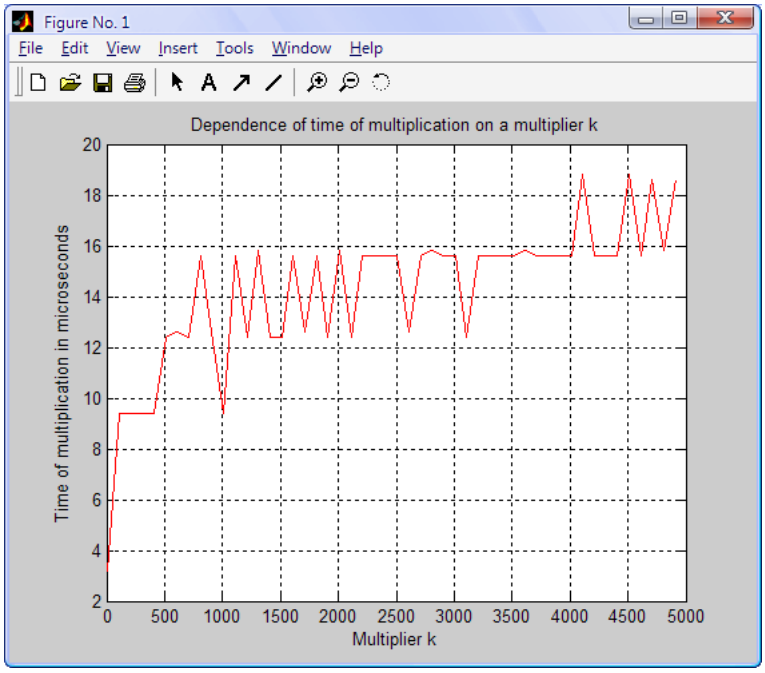


Рисунок 3.6 – Графічне подання залежності часу виконання множення від множника *k*

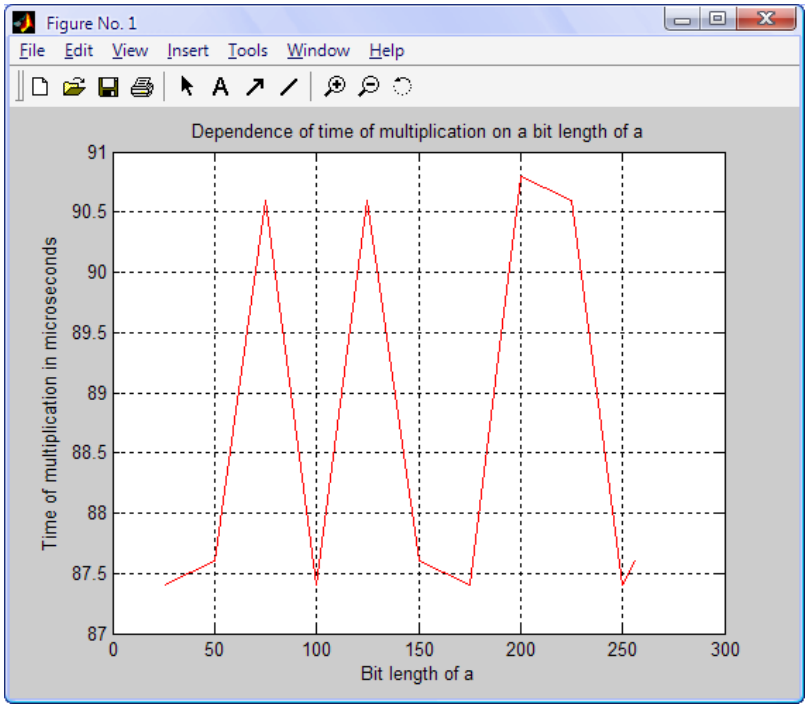


Рисунок 3.7 – Графічне подання залежності часу виконання множення від розрядності коефіцієнта *a*

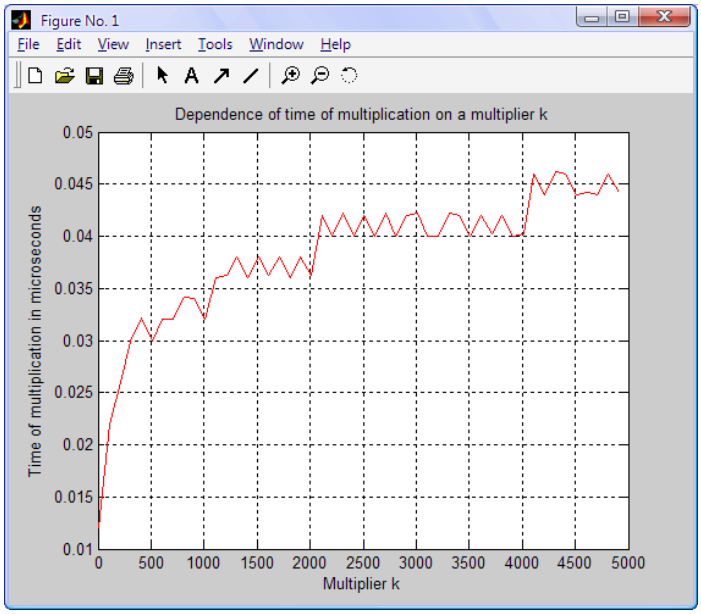


Рисунок 3.8 – Графічне подання залежності часу виконання апаратної операції множення від множника *k*

По отриманим даним було побудовано порівняльний графік продуктивності програмної та апаратної реалізацій операції пошуку скалярної точки на ЕК. Ці дані подано на рисунку 3.9.

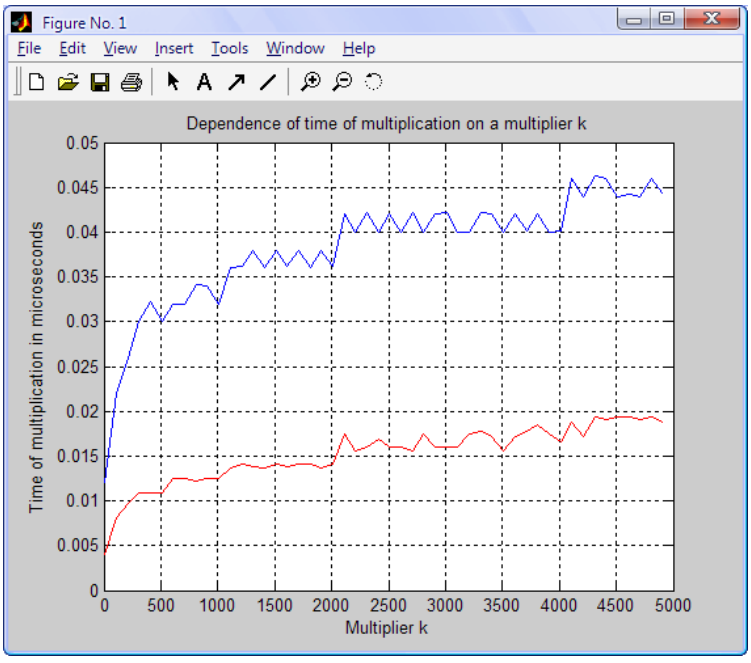


Рисунок 3.9 – Порівняння продуктивності апаратної та програмної реалізацій операції пошуку скалярної точки

На цьому рисунку синьою лінією відображено продуктивність апаратної реалізації, червоною – продуктивність програмної реалізації. З

Рисунку 3.9 видно, що різниця між графіками полягає тільки в швидкості виконання множення. В обох графіках чітко проглядається однакова залежність, яка пов’язана з тим, що в NAF(k) виникає новий розряд (це збільшує кількість додавань/віднімань точки). Також в межах кожної «сходинки» проглядається пилкоподібність графіка, що пояснюється неоднорідністю NAF(k) (змінюється кількість нульових бітів, що міняє кількість операцій додавання та віднімання).

4 ОХОРОНА ПРАЦІ

Основною метою розділу охорони праці є уникнення можливості виробничого травматизму, професійних отруєнь і захворювань, пожеж і вибухів, аварій, забруднення довкілля при будівництві та використанні об'єкта проектування [16].

Даний дипломний проект передбачає програмно-апаратну реалізацію алгоритму експоненціювання на еліптичних кривих.

В розділі охорона праці проводиться розрахунок безпечних умов праці для приміщення з комп'ютерами [17].

Обслуговування апаратури виконується в кімнаті розміщення обладнання контролера базових станцій. Обслуговуючий персонал займається контролюванням роботи апаратури, виявленням аварій та їх усуненням.

Контроль за роботою здійснюється за допомогою комп'ютерного обладнання, тому ця робота відноситься до категорії легких, які виконуються в сидячому, стоячому положенні або пов'язані з незначним рухом, але вона не відноситься до систематичної фізнчної роботи або до перенесення важких предметів.

Виходячи зі СН 245–71 і ГОСТ 12.1.005–88, а також, беручи до уваги характер робіт, відповідно до яких, площа приміщення на одного працівника в прbиміщенні дорівнює (6 м^2), приймаємо:

$$Sn=n\,S_0, \tag{4.1}$$

де $S0$ – площа приміщення, що виводиться на одного працівника;

n – кількість працівників.

Оскільки в приміщенні працює 6 чоловік, тоді необхідна площа для роботи повинна становити: $Sn = 6 \cdot 6 = 36$ м^2.

Реальна площа приміщення становить 48 м^2, тобто відповідає вимогам санітарних норм.

Згідно ГОСТ 12.1.005-88, в приміщенні повинні підтримуватися певні метеорологічні умови, що визначаються температурою відносною вологістю повітря, тиском та швидкістю руху повітря. Ці фактори впливають на термо-регуляцію, тобто спроможністю організму людини підтримувати нормальну температуру тіла (в межах 36 – 37 °С).

Тепловіддача від організму може здійснюватись шляхом тепловипромінення, конвекції і випаровування. При підвищеній температурі навколишнього середовища тепловіддача здійснюється лише за рахунок випаровування поту. Перегрівання тіла до 40 – 41 °С приводить до порушення водно – сольового обміну, виникнення судорожної хвороби і теплового удару з втратою свідомості.

Робота в умовах пониженої температури повітря, особливо при підвищеній вологості і швидкості руху, призводить до переохолодження тіла, що супроводжується виникненням простудних захворювань. Мінусова температура повітря призводить до обморожування, що розглядається як виробнича травма.

Для робочої зони нашого приміщення оптимальні і допустимі значення температури, відносної вологості і швидкості руху повітря встановлюються з врахуванням трудоємності і складності роботи, яка виконується. Користувачі персональних комп'ютерів належать до групи 1а – легкі роботи.

Відповідно з цим і ГОСТ 12.1.005–88 вибираємо необхідні метеорологічні умови (таблиця 4.1).

Для підтримання відповідних метеорологічних умов в приміщенні встановлено обладнання системи центрального опалення, але в зимовий період його тепловіддача є недостатньою. Доцільним є

проведення ущільнення конструктивів вікон і дверей, щоб припинити втрати тепла.

Решту метеорологічних умов забезпечує обладнання повного конди-ціонування повітря. Воно забезпечуе постійність температури, вологості, руху і чистоти повітря.

Таблиця 4.1 – Оптимальні та допустимі метеоумови

Період року	Категорія робіт	Температура, °С		Відносна вологість повітря		Швидкість повітря	
		опти-мальна	допус-тима	опти-мальна	допус-тима	опти-мальна	допу-стима
Холодний	Легка 1а	22–24	21–25	40–60	35–75	0,1	0,05–0,2
Теплий	Легка 1а	23–25	22–8	40–60	35–75	0,1	0,05–0,2

Сприятливі умови роботи забезпечують як високу продуктивність праці, так і позитивно впливають на психологічний стан людини, на її працездатність і здоров'я. Особливо важливе біологічне і гігієнічне значения для людини має природне освітлення, тому при проектуванні виробничих приміщень важливо передбачити наявність природного освітлення СНиП II–4–79.

Проведемо розрахунок природного освітлення згідно зі СНиП II–4–79 «Природне і штучне освітлення. Норми проектування», а при необхідності розрахуємо додаткове штучне освітлення приміщення.

Розрізняють три системи природного освітлення: бокове, верхнє, комбіноване. Для кількісної оцінки виробничого освітлення важливою технічною характеристикою є освітленість робочої поверхні. Густина світлової енергії на площі E (лк) визначається за формулою:

$$E = dF/dS,$$

(4.2)

де *dF* – світловий потік, який характеризує потужність світлового випромінювача (лм), рівномірно розподілений по площі *dS* (м^2).

Коефіцієнт природного освітлення, який являє собою відношення освітленості в даній точці середини приміщення *Ев* до зовнішнього горизонтального освітлення *Ез* визначаємо за формулою:

$$I = Ев/Ез. \quad (4.3)$$

Заміри натурного освітлення проводяться люксметром 10116.

Розміри приміщення становлять:

En · *B* = 6 × 8 м^2; висота приміщення *h* = 3 м, *S* – світловий отвір вікон 1 – 1,9 м^2. Віконне скло подвійне. Характеристика зорової роботи відноситься до високої точності. Це відповідає нормі природного освітлення КПО $I_н$= 2 % при боковому освітленні.

При боковому освітленні використовується формула:

$$100\frac{S_0}{S_n} = \frac{I_н \cdot K_з \cdot \eta_{10}}{\tau_0 \cdot VI} K_б; \quad (4.4)$$

де *S0* – площа світлових отворів, м^2;

Sn – площа підлоги, м ;

$K_з$ – коефіцієнт світлопроникнення;

η_{10} – світлова характеристика вікон;

τ_0 – загальний коефіцієнт світлопроникності;

VI – коефіцієнт, який враховує відбивання світла від поверхні;

$K_б$ – коефіцієнт, який враховує затемнення будинками, що стоять навпроти.

Для приміщення розмірами 6 · 8 · 3 площа *S* = 48 м^2; для *Ln/B* = 8/6 =1,33; *B/H* = 6/3 = 2; η_{10} = 16.

Для середньозваженого коефіцієнта відображення стелі, стін і підлоги, який дорівнює 0,4, коефіцієнт *VI* становить 2,4, $K_б$ приймаємо – 1,4.

Для приміщень з повітряним середовищем, в якому концентрація пилу менше 1 мг/м^3 $Кз$ =1,4; оскільки $I_н$= 2 % , коефіцієнт τ_0 визначаємо за формулою:

$$\tau_{0=}\tau_1 \cdot \tau_2 \cdot \tau_3 \cdot \tau_4 \cdot \tau_5; \qquad (4.5)$$

де τ_1, τ_2, τ_3 – коефіцієнти світлопропускання матеріалу вікна, виду вікна і його конструкції: для віконного, листового, подвійного скла τ_1=0,8; для дерев'яних подвійних роздільних оправ до вікон τ_2 =0,6; для залізобетонних конструкцій τ_3=0,8;

τ_4 – коефіцієнт, який враховує витрати світла в сонцезахисних конструкціях: для жалюзі і штор, що регулюються, дорівнює 1;

τ_5 – коефіцієнт, який враховує втрати світла в захисній сітці, що встановлюється під світильником — дорівнює 0,9.

Отже: τ_0= 0,8 · 0,6 · 0,8 · 1 · 0,9 = 0,35.

Визначаємо площу світлових отворів *S0*:

$$S_0 = \frac{I_н \cdot K_з \cdot \eta_{10} \cdot S_n}{100 \cdot \tau_0} = K_б; \qquad (4.6)$$

де *Sn* — стандартна площа вікна.

Кількість вікон визначаємо за формулою:

$$S_0 = \frac{2 \cdot 1{,}4 \cdot 16 \cdot 1{,}4 \cdot 48}{100 \cdot 0{,}35 \cdot 24} = 3{,}47(м^2) \qquad (4.7)$$

Відповідно: n =3,47/1,9 = 1,83 = 2 вікна. Таким чином, для забезпечення КПО $I_н$= 2 % у приміщенні повинно бути два вікна площею 1 ,9 м^2.

Для освітлення приміщення, коли природного освітлення недостатньо або взагалі немає, використовується штучне освітлення.

Світловий потік $Ф$ – це потужність світлової енергії, що оцінюється за світловим відчуттям, яке воно справляє на органи зору людини:

$$Ф = dQ/dt. \tag{4.8}$$

Сила світла I – це відношення світлового потоку до величини тілесного кута, в якому рівномірно розподілено випромінювання:

$$I = dF/d\omega. \tag{4.9}$$

Освітленість E – густина світлового потоку на освітлюваній поверхні:

$$E = dФ/dS. \tag{4.10}$$

Яскравість L – поверхнева густина сили світла у заданому напрямку:

$$L = dl/dS \cdot cos(\alpha). \tag{4.11}$$

Коефіцієнт відбиття β – відношення відбитого світлового потоку до падаючого: $\beta = Ф$відб/$Ф$пад.

Фон – поверхня, що прилягає безпосередньо до об'єкта розпізнавання, на який цей об'єкт сприймається. Фон характеризує коефіцієнт відбиття

(залежить від кольору поверхні та від її фактури). Фон світлий $Ф$>0,4; середній – $Ф$ = =0,2 – 0,4; темний $Ф$<0,2.

Контраст – ступінь розпізнавання яскравості об'єкта і фону:

$$K=(L_0-L_ф) / L_0. \quad (4.12)$$

Контраст великий, то K>0,5; середній – K = 0,2 – 0,5; маленький – K<0,2.

Коефіцієнт пульсацій $K_п$ – критерій оцінки відносної глибини коливань освтленості в результаті зміни в часі світлового потоку газорозрядних ламп при живленні їх змінним струмом:

$$K_п = (E_{макс} - E_{мін}) \cdot 100\% / (2 \cdot E_{сер}) \quad (4.13)$$

де $E_{сер}$ – значения освітленості за період.

Розміри приміщення: A = 8 м, B = 6 м, H = 3 м. Нормована освітленість 300 лк. Показник приміщення: $i = A \cdot B/(H \cdot (A+B)) = 8 \cdot 6/(3 \cdot (8+6)) = 1,14$.

Вибираємо світильник НОДЛ з коефіцієнтом використання світлового потоку η = 49%. Сумарний світловий потік:

$$Ф = ((E_н \cdot S \cdot k \cdot Z)/ \eta) \cdot 100\%, \quad (4.14)$$

де $E_н$ – нормована освітленість, лк;

S – площа приміщення, м^2;

k – коефіцієнт запасу;

Z – коефіцієнт мінімальної освітленості;

η – коефіцієнт використання світлового потоку.

$Ф$ = ((300·48· 1,75·1,1)/49)·100% = 56 572 лм.

Вибираємо лампи ЛТБ-80 р ,$Ф_л$— 4300 лм, тоді кількість ламп дорівнює: $N=Ф/Ф_л$ = 56572/4300 = 14 шт. Кількість світильників: $N_c=N/2$=7 шт.

Перерахуємо значення E:

$$E=\frac{N\cdot Ф_Л\cdot\eta}{S\cdot k\cdot Z\cdot 100\%}=\frac{14\cdot 4300\cdot 49}{48\cdot 1{,}75\cdot 1{,}1\cdot 100\%}=319{,}3. \quad (4.15)$$

Отже, штучне освітлення забезпечує освітленість E = 319 лк, що є більшим за $E_н/E_н$= 300 лк, тобто розрахунок проведений правильно.

Рівень шуму дорівнює 75 дБ, що відповідає вимогам ГОСТу, тому захисних заходів не передбачається.

Електричний струм при дії на людину може викликати як місцеві, так і загальні пошкодження. Місцеві електротравми – це опіки, нагрівання внутрішніх органів, механічні пошкодження (розрив тканин м'язів), порушення біоелектричних процесів у організмі, електроліз органічних рідин. Зовнішніми проявами електротравм можуть бути термічні опіки, електричні ознаки на шкірі, металізація поверхні шкіри, електроофтальмія (ураження зору під дією ультрафіолетових променів при іскровому розряді). Загальне ураження струмом відбувається при проходженні струму через нервові центри, центри дихання і роботи серця (електричний удар).

Небезпека ураження тим більша, чим більший струм проходить через людину, але крім цього, впливають: тривалість і шлях проходження струму, його вид, частота і виробничі умови.

Умови ураження людини електричним струмом такі:

– двофазне дотикання (двофазне включения людини в мережу);

– однофазне дотикання, наближення на небезпечну віддаль до неізольованих дротів з напругою більше 1000 В;

– дотик до корпуса обладнання, що не проводить струм, але опинилося під напругою;

– перебування в зоні дії атмосферної електрнки;

– вхід у зону дії електромагнітного поля.

Згідно класифікації приміщень за ступенем небезпеки ураження електричним струмом (ПУЕ 1.1.6) приміщення роботи системи відноситься до першого (без підвищеної небезпеки).

Електричні установки, до яких відноситься переважна більшість обладнання системи, вимагають дотримання правил електробезпеки, оскільки в процесі експлуатації або проведения профілактичних робіт людина може доторкнутись до частин, що знаходяться під напругою 220 В, тому виникає необхідність у захисті персоналу від ураження електричним струмом. Дуже велике значения для запобігання електротравматизму має правильна організація експлуатації, обслуговування системи. Під цим розуміється точне виконання ряду організаційних та технічних заходів, які встановлені діючими «Правилами технічної експлуатації електроустановок споживачів і правил техніки безпеки при експлуатації електроустановок споживачів» (ППЕ і ПТБ споживачів) і «Правилами побудови електропристроїв» (ППЕ). Основними технічними засобами. які забезпечують безпеку робіт в електроустановках, є: захисне заземления, занулення, вирівнювання потенціалів, захисне включення, електричний розподіл мереж, мала напрута, подвійна ізоляція. Використання цих засобів у різноманітних поєднаннях дозволяє захистити людину від ураження струмом.

Захисне заземлення – це навмисне електричне з'єднання з землею або її еквівалентом металевих неструмопровідних частин, які можуть бути під напругою. У приміщенні розміщення контролера базових станцій заземлено всі шафи з обладнанням, а також вся комп’ютерна техніка. Приміщення, де знаходиться система, обладнується контуром-шиною захисного заземлення, яка з'єднується із заземлювачем. Контур-шина виготовляється з мідного дроту діаметром 6 мм у перерізі і вкладається по периметру приміщення. Місця перетину дротів пропаюються з застосуванням бікислотного флюсу. Для під’єнання заземлювальних

провівників на шину наварюються гвинти М8. У дипломній роботі проведу розрахунок захисного заземления згідно порядку, встановленого ПУЕ.

Згідно вимог ПУЕ 1.7.65 в електроустановках з напругами до 1 кВ при потужності трансформатора менше 100 кВт опір заземлювача повинен бути не більше 10 Ом.

1. Визначаємо розрахунковий опір землі: $ro_{р.з.} = Ф\ ro_з$, де $Ф$ – коефіцієнт сезонності, який враховує коливання питомого опору при зміні вологості грунту протягом року; використовується стержневий заземлювач (рисунок 4.1) довжиною l = 2 м при глибині закладання від вершини h = 0,5 м, $Ф$ = 1,1 для четвертої кліматичної зони. Питомий опір грунту: $ro_З$ = 300 Ом·м - для піску; $ro_{р.з.}$ = 1,1 ·300 = 330 Ом·м.

2. Визначаємо опір R, розтікання струму в землі від одного вертикального заземлювача:

$$R_B = \frac{ro_{п.з.}}{2 \cdot 3{,}14 \cdot l}\left(ln\frac{2 \cdot 1}{d} + \frac{1}{2} ln\frac{4 \cdot t + l}{4 \cdot t - l}\right), \qquad (4.16)$$

де l – довжина заземлювача (l = 2 м);

d=0,05 м – діаметр заземлювача за таблицею при U<1 кВ та при S<100 кВА;

t –відстань від поверхні землі до середини заземлювача,

$t=h+l/2=0{,}5+2/2=$ 1,5 м; $R_B = \frac{330}{2 \cdot 3{,}14 \cdot 2}\left(ln\frac{2 \cdot 2}{0{,}05} + \frac{1}{2} ln\frac{4 \cdot 1{,}5 + 2}{4 \cdot 1{,}5 - 2}\right) = 133{,}3\ Ом.$

3. Приблизна кількість заземлювачів: $n = \frac{R_в}{R_{в.нтмМ}} = \frac{133{,}3}{10} = 13{,}3 \approx 14.$

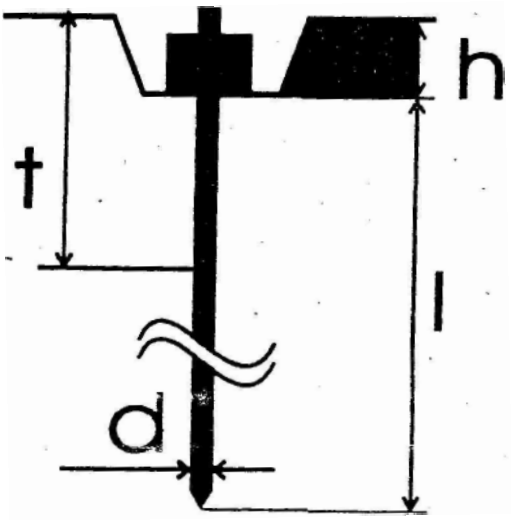


Рисунок 4.1 – Схема розташування одиничного заземлювача в грунті

4. Знаходимо із таблиць коефіцієнт використання вертикальних заземлювачів, який враховує ефект екранування при вибраному значенні $k=a/l$, де a — віддаль між заземлювачами, м; k = 1,2 при а = 2,4 м; отже коефіцієнт використання вертикального заземлювача за таблицями дорівнює η_e= 0,56.

5. Кількість вертикальних заземлювачів з урахуванням η_e обчислюємо за формулою $n=\frac{R_в}{R_{в.норм}\cdot\eta_в}=\frac{133,277}{10\cdot0,56}=23,799=24.$

6. Довжина горизонтального заземлювача для розміщення по контуру $L=a\cdot n$ = 2,4 м · 24 = 57,6 м.

7. Опір горизонтального заземлювача $R_г$ (Ом), прокладеного на глибині h = 0,5 м від поверхні землі:

$$R_г=\frac{ro_{р.к.}}{2\cdot3,14\cdot L}ln\frac{2\cdot L}{b\cdot h}=\frac{330}{2\cdot3,14\cdot57,6}ln\frac{2\cdot57,6}{0,04\cdot0,5}=7,3 \quad (4.17)$$

де b = 0,04 м — ширина штабової сталі, з якої виготовлений заземлювач.

8. Обчислюємо загальний опір:

$$R_к = \frac{R_в \cdot R_д}{n \cdot R_д \eta_в + R_в \eta_д} = \frac{133{,}3 \cdot 7{,}3}{24 \cdot 7{,}3 \cdot 0{,}56 + 133{,}3 \cdot 0{,}27} = 7{,}5 \text{ Ом}.$$

(4.18)

Результат є менше 10 Ом, тобто виконується нормуюча умова $R_{з.} < R_{з.норм.}$.

Велика увага приділяється дотриманню обслуговуючим персоналом правил роботи в приміщенні, яке призначене для експлуатації системи. У приміщенні не повинно бути сторонніх людей. Працівники повинні використовувати спецодяг. Безпека роботи обслуговуючого персоналу в приміщенні забезпечується:

– наявністю нормальних проходів між обладнанням;

– використанням спеціальних технічних меблів;

– використанням електрозахисних засобів (діелектричних килимків, гумових рукавиць);

– наявністю аварійного освітлення (E=2 лк);

– обладнанням розеток з напругою 220 В;

– заземленням корпусів обладнання і апаратури освітлювальних пристроїв.

Одне з основних місць в охороні праці займає пожежна безпека.

Першочергове завдання пожежної профілактики — це запобігання пожеж. Під пожежною профілактикою розуміють комплекс організаційних і технічних заходів, спрямованих на забезпечення безпеки людей, на запобігання пожеж, обмеження їх розповсюдження, а також на створення умов для успішного гасіння пожеж. Пожежно-профілактичні заходи розробляються та виконуються разом, в тісному взаємозв'язку з усіма проектними, будівельними та експлуатаційними роботами.

Приміщення чергування технічного персоналу забезпечуеться протипожежним івентарем (вуглекислотними вогнегасниками типу ВВ–

2). Проходи між рядами і вихід не повинні загромаджуватись. У випадку виникнення пожежі перш за все потрібно виключити джерело живлення, сповістити про пожежу в пожежну частину. Евакуювати сторонніх людей, які могли опинитися в небезпечшй зоні і лише після цього приступити до гасіння пожежі і рятування цінного обладнання.

Один вуглекислотний вогнегасник ВВ–2 розрахований на 40–50 м^2 примі-щення. Для ліквідації невеликих пожеж можна використовувати деякі порошкові матеріали (хлориди лужних металів, соду, пісок і т. д.), що подаються в зону горіння порошковими вогнегасниками.

Будівля, в якій знаходиться наше приміщення, обов'язково має резервний вихід на випадок екстреної евакуації працівників і неможливості використання основного виходу.

За вибухопожежною і пожежною небезпекою приміщення і будівлі згідно ОНТП–24–86 і СНТП 2.09, СНТП 02–85 діляться на категорії А, Б, В, Г, Д.

Для приміщення чергування персоналу встановлена категорія пожежної небезпеки Д (СНІП 2.09.02–85) при ступені вогнестійкості (СНІП Н–90–81), що означає наявність у приміщенні негорючих речовин та матеріалів у холодному стані.

Для швидкого сповіщення пожежної схорони при виникненні пожежі приміщенні використовується електрична пожежна сигналізащя. Система електричної пожежної сигналізації виявляє пожежу на початковій стадії і сповіщае про місце її виникнення, а також автоматично включає стаціонарні установки гасіння пожеж.

Автоматичні сповіщувачі при ознаках пожежі здійснюють посилання сигналу. Сповіщувачі типу АТИП–1, АТИП–3 і АТИП–3М спрацьовують внаслідок теплової деформації (при 80–100 °С) біметалічних пластинок і мають розраховану площу обслуговування в приміщеннях до 15 м^2. Комбіновані теплові і димові сповіщувачі типу КИ–1 мають чутливий елемент у вигляді іонізуючої камери (реагування на дим) і терморезистори

(реагування на тепло). Температура спрацювання цих сповіщувачів 50–80°С, площа обслуговування 100 м^2.

Передбачені нами заходи з охорони праці в першу чергу призначені для уникнення нещасних випадків, що можуть виникнути на підприємстві.

В іншому передбачені заходи з охорони праці відповідають вимогам нормативних документів та актів та забезпечують нормальну, ефективну і безпечну для здоров'я людини виробничу діяльність.

ВИСНОВКИ

В даному дипломному проекті було 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;
- використання малих ключів дозволяє використовувати математичний апарат ЕК в мобільних пристроях (телефонах, старт-картках, платіжних картках та ін.);
- 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 електронно-цифрового підпису (де в принципі не використовується розшифровка даних) та в процедурах розповсюдження ключів, де не потрібно розшифровувати великі обсяги інформації. Також суттєвою проблемою КЕК є генерація параметрів кривої, які б забезпечили необхідну криптографічну стійкість системи.

Було розглянуто алгоритм скалярного експоненціювання точок на еліптичних кривих. Цей алгоритм було реалізовано в вигляді програмного продукту, який дозволяє здійснювати процедуру множення точок на число та аналізувати швидкодію цього процесу.

Згідно з експериментальними даними було побудовано графіки, які ілюструють залежність часу виконання множення точки на скаляр від множника та параметрів еліптичної кривої.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Elliptic Curve Cryptography. Certicom Research, 1999. Working Draft.
2. IEEE P1363 / D8(Draft Version 8). Standard Specifications for Public Key Cryptography.
3. Исаев Сергей. Генетические алгоритмы – эволюционные методы поиска, http://ai-online.fromru.com/documents-genetic_algorithms.html.
4. Операційні системи. Внутрішня будова і принципи проектуваня. 4-е видання.: Пер.с англ. – М.: Видавничий дім “Вільямс”, 2002. – 848 с.
5. A. Menezes, Y.-H. Wu, and R. Zuccherato. An Elementary Introduction to Hyperelliptic Curves. In Algebraic Aspects of Cryptography.
6. N. Koblitz. Hyperelliptic cryptosystems. J. of Cryptology, 1:139-150, 1989.
7. D. Lorenzini. An Invitation to Arithmetic Geometry, volume 9 of Graduate studies in mathematics. AMS, 1996.
8. T. Lange. Efficient Arithmetic on Genus 2 Hyperelliptic Curves over Finite Fields via Explicit Formulae. Report 2003. http://eprint.iacr.org/ or http://www.itsc.ruhr-uni-bochum.de/tanja.
9. “Еліптична крива для асиметричної криптографічної системи”, Вісник Тернопільського державного технічного університету імені Ів.Пулюя, - Том 6, - №3 - 2001, ст.91-95.
10. М.Карпінський, І.Васильцов, І.Якименко, Я.Кінах, “Метод генерування параметрів еліптичних кривих”, Правове, нормативне, та метрологічне забезпечення системи захисту інформації в Україні, Київ, випуск 6, с.74, 2003.
11. М. Карпінський, І.Васильцов.,І.Якименко, А.Гончарик. Elliptic curve Parameters Generetion // Proceedings of the Integrational Conference TCSET’2004 “Modern problems of radio engineering, Telecomunications and computer science”, february 24-28, 2004, p. 294-295.
12. М.Карпінський, І.Васильцов, І.Якименко, “Показники оцінки ефективності алгоритмів шифрування на еліптичних кривих”,

Правове, нормативне, та метрологічне забезпечення системи захисту інформації в Україні, Київ, випуск 8, с.121-124, 2004.

13. Карпінський М.П., Якименко І.З., Чайківська Ю.М. Класифікація атак на апаратно-програмні засоби криптосистем // Матеріали II міжнар. наук.-техн. конф. «Світлотехніка й електротехніка: історія, проблеми й перспективи» приуроченої 160-річчю видатного українського фізика, піонера в галузі світлотехніки і електротехніки професора Івана Пулюя (24-27 травня 2005 р., Тернопіль, Україна). – Тернопіль: Тернопільський державний технічний університет імені Івана Пулюя. – 2005. – С. 70-71.
14. Karpinsky M.P., Yakymenko I.Z., Chajkivska J.M. Formalization Assessment Criterion Attacks on Cryptosystems Using Elliptic Curves// Proceedings of the Third IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications “IDAACS’2005” (September 5-7, 2005, Sofia, Bulgaria). – Sofia. – 2005. – Pp. 399-402.
15. Heinlein A., Gharachorloo R., Dresser M., Gupta F. Integration of Message Passing and Shred Memory in the Stanford FLASH Multiprocessor // Proc. International Conf. on Architectural Support for Programming Languages and Operating Systems. XI. - San Jose (Canada). - 1994. - P. 38 - 50.
16. Abramson D., Sosic. R., Debugging A. Tool for Software Evolution // Proc. International Conf. Workshop on Computer - Aided Software Engineering. - Toronto (Canada). - 1995. - P. 115 - 127.
17. Brands S. Rethinking Public Key Infrastructures And Digital Certificates - Building In Privacy. - Fatbrain, 1999. - P. 9 - 30.
18. ДСТУ 3396.2-97. Захист інформації. Технічний захист інформації. Терміни та визначення. – Введ. 01.01.98. – К.: Держстандарт України, 1997. – 11 с.
19. 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.

20. Хоффман Л. Современные методы защиты информации: Пер. с англ. - М.: Сов. радио, 1980. - С. 12 - 34.
21. 0тчет Министерства внутренних дел РФ перед гражданами России // Российская газета. - 1994. - 11 марта.
22. Бияшев Р. Г., Диев С. И., Размахнин М. К. Основные направления развития и совершенствования криптографического закрытия информации // Зарубежная радиоэлектроника. - 1989. - № 12. - С. 35 - 39.
23. Диффи У. Первые десять лет криптографии с открытым ключом // ТИИЭР. - 1988. - Т. 76, №5. - С. 54 - 63.
24. Кнут Д. Искусство программирования на ЭВМ: В 4т. / Мир. - М., 1977. - Т.2: Получисленные алгоритмы С. 22 - 34.
25. Riesel H. Prime numbers and computer methods for factorization. - Birkhauser, 1985. - P. 51 - 55.
26. Cohen H. A course in computational algebraic number theory. GraduateTexts in Math // Proc. International Conf. on Computer sciences. - New - York (USA). - 1993. P. 132 - 135.
27. Gordon D.M. Discrete logarithms in GF(p) , using the number field sieve // SIAM J. Disc. Math. 1993. - №1. P. 124 - 138.
28. Lenstra A., Lenstra H. The Development of the Number Field Sieve // Lect. Notes in Math. - 1993. - Vol. 1554, №2. - P. 155 - 157.
29. Tuminaro R., Heroux M., Hutchinson S., Shadid J. Official Aztec User's Guide: Sandia National Laboratories, 1999. - P. 6 - 8.
30. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анаализ вычислительных алгоритмов. М.: Мир, 1979. - С. 37 - 40.
31. Варновский Н. П. Криптография и теория сложности // Математическое просвещение. - 1998. - №2. - С. 71 - 86.
32. Dfrett P. D. Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor // Advances in Cryptology, Eurocrypt'86, Lect.Notes Comput.Sci.–1987.-№263.–P.311–323.

33. Горбенко И. Д., Долгов В. И., Рублинецкий В. И., Коровкин К. В. Методы защиты информации в системах телекомуникаций и методы их криптоанализа // Радиотехника. – 1997. – Вып. 104. – С.138 – 150.
34. El - Gamal T. On Computing Logarithms Over Finite Fields. // Proc. International Conf. CRYPTO. - Chikago (USA). - 1985. - P. 207 - 210.
35. El - Gamal T. A Subexponential - Time Algorithm for Computing DiscreteLogarithms over GF(pI) // Proc. International Conf CRYPTO. - London (England). -1983. - P. 93 - 110.
36. Лотохов А.В., Мельникова О.А. Исследование зависимости времени работы алгоритма "квадратичное решето" от размера базы // Программа 2-ого международного молодёжного форума "Электроника и молодёжь в XXI веке". - Харьков: ОНТИ ХТУРЭ, 1998. - С. 67.
37. Кнут. Д. Искусство программирования для ЭВМ: В 4 т. / Мир. - М., 1976. - Т. 1: Основные алгоритмы. - С. 25 - 29.
38. Pohlig S., Hellman M. An improved algorithm for computing logarithms in GF(p) and its cryptographic significance // IEEE trans. on inform. theory. – 1978. – Vol. IT – 24. -№24. – P. 106 – 110.
39. Мур. Дж. Несостоятельность протоколов криптосистем // ТИИЭР. - 1888. - №5. - С 40 - 42.
40. Горбенко И.Д., Бондаренко А.С., Качко Е.Г., Мельникова О.А. Сравнительный анализ стандартов шифрования и цифровой подписи // Тезисы докладов научно - методической конференции "Использование компьютерных технологий в учебном процессе". - Харьков: ОНТИ ХТУРЭ, 1997. - С. 90.
41. Горбенко И.Д., Долгов В.И.., Качко Е.Г., Коровкин К.В., Свинарёв А.В., Мельникова О.А. Системы с открытыми ключами и открытым распространением ключей: мифы и реальность // Материалы II международной научно-практической конференции "Безопасность информации в компьютерных системах и связи". - Партенит, 1996. - С. 32.

42. Вербіцький О.В. Вступ до криптології. – Львів: ВНТЛ, 1998. – 247 с.
43. Романєц Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерніх системах и сетях. - М.: Радио и связь, 1999. - 328с.
44. Lenstra A.,Manasse M. Factoring with two large primes // Advances in cryptology: Proc. of Eurocrypt'90. – Berlin (Gemeny), 1991. – P. 72 – 82.
45. Montgomery P. Modular Multiplication without Trial Division // Mathematics of Computation. - 1985. - Vol. 44, № 170. - P. 519 - 521.
46. Уильямс Х. Проверка чисел на простоту с помощью вычислительных машин. Пер. с англ. // Кибернетический сборник. Вып. 23.-М.:Мир,1986. С.51 - 59.
47. Жуков І.А. Теорія та принципи організації паралельних обчислювальних структур і систем для розв'язання задач великої розмірності: Дис...д-ра техн. наук: 05.13.13. - К., 1997. - 297с.
48. Кофто А.Г. Алгоритмы распараллеливания решения систем уравнений большой размерности, имеющих блочно-диагональные с окаймлением матрицы коэффициентов, на многопроцессорных вычислительных системах // Электронное моделирование. - 1982. - №2. С. 90 - 92.
49. Нагорный Л.Я. Жуков И.А. Метод решения систем больших размерностей на многопроцессорных структурах // Автоматизация проектирования в електронике. - Киев: Техніка.-1979. - Вып.20. - С. 76 - 80.
44. Нагорный Л.Я. Методы распаллеливания систем уравнений большой размерности для решения их на многпроцессорных структурах // Электронное моделирование. - 1980. - №1. - С. 28 - 32.
45. Jean-Yves Chouinard. Notes on Elliptic Curve Cryptography, 2002.
47. [Електронний ресурс] http://www.bezpeka.net

48. [Електронний ресурс] http://www.certicom.com
51. [Електронний ресурс] http://www.domarev.kiev.ua
52. [Електронний ресурс] http://www.haker.ru
53. [Електронний ресурс] http://www.ib.kgtu.runnet.ru
54. [Електронний ресурс] http://www.ict.nsc.ru
55. [Електронний ресурс] http://www.iis.ee.ethz.ch
56. [Електронний ресурс] http://www.kiev-security.org.ua
57. [Електронний ресурс] http://www.ods.com.ua
58. [Електронний ресурс] http://www.stdsbbs.ieee.org/groups/1363
59. Карпінський М.П., Гіжицкі М., Якименко І.З. Оцінка продуктивності та стійкості до часового аналізу алгоритмів експоненціювання точки еліптичної кривої // Вісник Хмельницького національного університету. – 2006. – № 5. – С. 23-30.
60. IEEE P1363 / D8(Draft Version 8). Standard Specifications for Public Key Cryptography.

Додаток А.

Текст програми пошуку скалярної точки на еліптичних кривих

Модуль Main.cpp:

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Main.h"
#include "DataChoose.h"
#include "Speed.h"
#include "lip.c"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"

/*

Після виконання функцій з префіксом ec
результат заноситься в точку res.

Виняток: eccopy і ecequal

В файлі Numbers.txt позиції значать:
   - 90 -- число A
   - 178 -- число B
   - 273 -- координата X базової точки на ЕК
*/

typedef struct {verylong x; verylong y;} ecpoint;
```

```
const long anum = 90;
                long bnum = 90;
                long xcoor = 273;
                long start = 0;
                long curr = 1;
                long end = 2;

TFMain *FMain;
unsigned long k, i = 0, numstr;          //numstr - кількість рядків в файлі
Numbers.txt
verylong a, b, p, lambda, k_coef;
ecpoint R, P, PointEC, res, O, p2inv, t;
FILE *f;
verylong tmp[3];
DynamicArray <signed short> kbin;    //k in binary view "01010001" for ex.
//int BitLen;                                                    //Length    of
kbin
char str[256];
fpos_t fpos;

void ansitoz(AnsiString astr, verylong &num)
{
   for (i = 0; i < astr.Length(); i++)
         str[i] = astr[i+1];
   str[i] = 0;
   zsread(str, &num);
}

void refreshdata()
{
   zswrite(str, a);
```

```
    FMain->CoefA->Caption = str;
    zswrite(str, b);
    FMain->CoefB->Caption = str;
    zswrite(str, PointEC.x);
    FMain->XCoor->Text = str;
    zswrite(str, PointEC.y);
    FMain->YCoor->Text = str;
    if (FMain->Visible) FMain->KNum->SetFocus();
}

void closedata(FILE *fstream)
{
    fgetpos(fstream, &fpos);
    fclose(fstream);
}

int reopendata(FILE *fstream)
{
    fstream = fopen("Numbers.txt", "r");
    return !(fsetpos(fstream, &fpos));
}

void nextrec(FILE *fstream)
{
    char s[256];
    while (fscanf(fstream, "%s", s) != EOF)
    {
            if (s[0] == 'A') break;
    }
}
```

```
bool getnextdata(FILE *fstream, verylong &coef_a,
                 verylong &coef_b, verylong &coor_x, verylong &coor_y)
{
   char s[256];
   nextrec(fstream);

   if (fscanf(fstream, "%s", s) == EOF) return false;
   zsread(s, &coef_a);

   fscanf(fstream, "%s %s", s, s);  //B number
   zsread(s, &coef_b);

   fseek(fstream, 9, curr);
   fscanf(fstream, "%s", s);                    //X coordinate
   zsread(s, &coor_x);
   fscanf(fstream, "%s", s);                    //Y coordinate
   zsread(s, &coor_y);
   return true;
}

bool ecequal(ecpoint p1, ecpoint p2)
{
  return !((zcompare(p1.x, p2.x) || zcompare(p1.y, p2.y)));
}

void eccopy(ecpoint p1, ecpoint &p2)
{
  zcopy(p1.x, &p2.x);
  zcopy(p1.y, &p2.y);
}
```

```
void ecaddpoint(ecpoint p1, ecpoint p2) //res = p1 + p2
{
  if (ecequal(p1, O))
  {
    eccopy(p2, res);
    return;
    }

    if (ecequal(p2, O))
    {
            eccopy(p1, res);
            return;
    }

    if (zcompare(p1.x, p2.x))
    {
            zsub(p1.y, p2.y, &tmp[0]);
            zsub(p1.x, p2.x, &tmp[1]);
            zdiv(tmp[0], tmp[1], &tmp[2], &lambda);
            zmod(tmp[2], p, &lambda);
            goto step7;
    }

    if (zcompare(p1.y, p2.y) || ziszero(p2.y))
    {
            eccopy(O, res);
            return;
    }

    zsq(p2.x, &tmp[0]);
    zsmul(tmp[0], 3, &tmp[1]);
```

```
    zadd(tmp[1], a, &tmp[0]);                //tmp[0] = 3x1^2+a
    zsmul(p2.y, 2, &tmp[1]);          //tmp[1] = 2y1
    zdiv(tmp[0], tmp[1], &tmp[2], &lambda);
   zmod(tmp[2], p, &lambda);

step7::;
    zsq(lambda, &tmp[0]);                                    //tmp[0] = lambda^2
    zsub(tmp[0], p1.x, &tmp[1]);             //tmp[1] = lambda^2 - x0
    zsub(tmp[1], p2.x, &tmp[0]);
    zmod(tmp[0], p, &res.x);
    zsub(p2.x, res.x, &tmp[0]);
    zmul(tmp[0], lambda, &tmp[1]);        //tmp[1] = (x1-x2)lambda
    zsub(tmp[1], p2.y, &tmp[0]);
    zmod(tmp[0], p, &res.y);
}

void ecsubpoint(ecpoint p1, ecpoint p2) //res = p1 - p2
{                              //res = p1 + p2(x, -y)
    eccopy(p2, p2inv);
    znegate(&p2inv.y);
    ecaddpoint(p1, p2inv);
}

void NAF(verylong k)                        //kbin = NAF(k)
{
    i = 0;
    kbin.Length = 0;
    zcopy(k, &tmp[2]);
    while (zcompare(tmp[2], O.x) == 1)
    {
            kbin.Length++;
```

```
            if (zodd(k))
            {
                kbin[i] = zsmod(tmp[0], 4);
                kbin[i] = 2 - kbin[i];
                zintoz(kbin[i], &tmp[0]);
                zsub(tmp[2], tmp[0], &tmp[1]);
                zcopy(tmp[1], &tmp[2]);
            }
            else
                kbin[i] = 0;
            zsdiv(tmp[2], 2, &tmp[0]);
            zcopy(tmp[0], &tmp[2]);
            i++;
    }
//  kbin[i] = 0;
//  BitLen = i;
}

void ecmulpoint(verylong k, ecpoint p1)
{
    eccopy(O, R);
    eccopy(p1, t);
    NAF(k);
    if (kbin.Length == 0) return;

    for (i = kbin.Length - 1; i != 0; i--)
    {
            ecaddpoint(t, t);                                   //res = 2*p1
            eccopy(res, t);                                         //p1 = 2*p1
            if (kbin[i] == 1)
                ecaddpoint(R, t);
```

```
            else
            if (kbin[i] == -1)
                    ecsubpoint(R, t);
            eccopy(res, R);
    }
}

//---------------------------------------------------------------------------
__fastcall TFMain::TFMain(TComponent* Owner)
                        : TForm(Owner)
{
}
//---------------------------------------------------------------------------


void __fastcall TFMain::FormCreate(TObject *Sender)
{
    zstart();
    char str[256];
    zzero(&O.x);
    zzero(&O.y);
    if (!(f = fopen("Numbers.txt", "r")))
    {
            MessageBox(FMain->Handle, "Error opening file Numbers.txt!",
"Fatal Error", MB_ICONSTOP);
            Application->Terminate();
            return;
    }
    numstr = 0;
    while (fscanf(f, "%s\n", str) != EOF)
    {
```

```
            if (str[0] == 'A') numstr++;
    }
    fseek(f, 0, start);
    getnextdata(f, a, b, PointEC.x, PointEC.y);
    refreshdata();
    closedata(f);
    reopendata(f);
    getnextdata(f, a, b, PointEC.x, PointEC.y);
    if (!(f = fopen("ModulusP.txt", "r")))
    {
            MessageBox(FMain->Handle, "Error opening file ModulusP.txt!",
"Fatal Error", MB_ICONSTOP);
            Application->Terminate();
            return;
    }
    fscanf(f, "%s", str);
    zsread(str, &p);
    ModP->Caption = str;
    fclose(f);
}
//---------------------------------------------------------------------------

void __fastcall TFMain::QuitClick(TObject *Sender)
{
    Application->Terminate();
}
//---------------------------------------------------------------------------

void __fastcall TFMain::OkBClick(TObject *Sender)
{
    if (XCoor->Text == "")
```

```
        XCoor->Text = "0";
    if (YCoor->Text == "")
        YCoor->Text = "0";
    if (KNum->Text == "")
        KNum->Text = "0";
    for (i = 1; i <= XCoor->Text.Length(); i++)
        str[i-1] = XCoor->Text[i];
    str[i-1] = 0;
    zsread(str, &PointEC.x);

    for (i = 1; i <= YCoor->Text.Length(); i++)
        str[i-1] = YCoor->Text[i];
    str[i-1] = 0;
    zsread(str, &PointEC.y);

    for (i = 1; i <= KNum->Text.Length(); i++)
        str[i-1] = KNum->Text[i];
    str[i-1] = 0;
    zsread(str, &k_coef);

    ecmulpoint(k_coef, PointEC);
    zswrite(str, res.x);
    ResX->Text = str;
    zswrite(str, res.y);
    ResY->Text = str;
    refreshdata();
}
//---------------------------------------------------------------------------


void __fastcall TFMain::XCoorKeyPress(TObject *Sender, char &Key)
```

```
{
   if (Key == '\b') return;
   if (Key == '\r')
         OkB->Click();
   if ((Key < '0') || (Key > '9'))
         Key = 0;
}
//---------------------------------------------------------------------------

void __fastcall TFMain::KNumKeyPress(TObject *Sender, char &Key)
{
   if (Key == '\b') return;
   if (Key == '\r')
         OkB->Click();
   if ((Key < '0') || (Key > '9'))
         Key = 0;
}
//---------------------------------------------------------------------------

void __fastcall TFMain::YCoorKeyPress(TObject *Sender, char &Key)
{
   if (Key == '\b') return;
   if (Key == '\r')
         OkB->Click();
   if ((Key < '0') || (Key > '9'))
         Key = 0;
}
//---------------------------------------------------------------------------

void __fastcall TFMain::DataChooseClick(TObject *Sender)
{
```

```
    FDataChoose->Scroll->Max = numstr;
    FDataChoose->ShowModal();
    ansitoz(FDataChoose->ACoef->Caption, a);
    ansitoz(FDataChoose->BCoef->Caption, b);
    ansitoz(FDataChoose->XCoor->Caption, PointEC.x);
    ansitoz(FDataChoose->YCoor->Caption, PointEC.y);
    refreshdata();
    KNum->SetFocus();
  KNum->SelectAll();
}
//---------------------------------------------------------------------------
void __fastcall TFMain::Button1Click(TObject *Sender)
{
    reopendata(f);
again:;
    if (!getnextdata(f, a, b, PointEC.x, PointEC.y))
    {
            fclose(f);
            f = fopen("Numbers.txt", "r");
            fgetpos(f, &fpos);
            goto again;
    }
    ;
    refreshdata();
}
//---------------------------------------------------------------------------

void __fastcall TFMain::XCoorChange(TObject *Sender)
{
    ResX->Text = "";
    ResY->Text = "";
```

```
}
//---------------------------------------------------------------------------

void __fastcall TFMain::SpeedTestClick(TObject *Sender)
{
   FSpeed->ShowModal();
}
//---------------------------------------------------------------------------
```

Модуль DataChoose.cpp:

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "DataChoose.h"
#include "Main.cpp"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"

TFDataChoose *FDataChoose;
int readres;
verylong an, bn, xc, yc;
char s[256];
AnsiString string;

void readpoint(long num)
{
   f = fopen("Numbers.txt", "r");
   for (int i = 0; i < num-1; i++)
         nextrec(f);
   getnextdata(f, an, bn, xc, yc);
```

```
    fclose(f);
}

void setdata()
{
    zswrite(s, an);
    FDataChoose->ACoef->Caption = s;
    zswrite(s, bn);
    FDataChoose->BCoef->Caption = s;
    zswrite(s, xc);
    FDataChoose->XCoor->Caption = s;
    zswrite(s, yc);
    FDataChoose->YCoor->Caption = s;
}

//---------------------------------------------------------------------------
__fastcall TFDataChoose::TFDataChoose(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TFDataChoose::CancelClick(TObject *Sender)
{
    FDataChoose->Close();
}
//---------------------------------------------------------------------------
void __fastcall TFDataChoose::FormShow(TObject *Sender)
{
    if (!(f = fopen("Numbers.txt", "r")))
    {
```

```
        MessageBox(FMain->Handle, "Error opening file ModulusP.txt!",
"Fatal Error", MB_ICONSTOP);
        Application->Terminate();
        return;
    }
    PointNum->Text = 1;
    Scroll->Position = 1;
    Max->Caption = "Максимальне значення - "+IntToStr(Scroll->Max);
    readpoint(1);
    setdata();
    PointNum->SetFocus();
    PointNum->SelectAll();
}
//---------------------------------------------------------------------------

void __fastcall TFDataChoose::ScrollScroll(TObject *Sender,
     TScrollCode ScrollCode, int &ScrollPos)
{
    long num = StrToInt(PointNum->Text);
    switch (ScrollCode)
    {
        case scLineUp:
            if (num < Scroll->Max)
                PointNum->Text = IntToStr(num+1);
            break;
        case scLineDown:
            if (num > 1) PointNum->Text = IntToStr(num-1);
            break;
        case scEndScroll:
            return;
    }
```

```
}
//---------------------------------------------------------------------------

void __fastcall TFDataChoose::PointNumKeyPress(TObject *Sender, char
&Key)
{
   if (Key == '\b') return;
   if (Key == '\r') OkB->Click();
   if ((Key < '0') || (Key > '9')) Key = 0;
   if (PointNum->Text == "")
   {
         OkB->Enabled = true;
         Scroll->Position = StrToInt(Key);
         return;
   }
   if (StrToInt(PointNum->Text + Key) > Scroll->Max) Key = 0;
   OkB->Enabled = StrToInt(PointNum->Text) != 0;
}
//---------------------------------------------------------------------------

void __fastcall TFDataChoose::PointNumChange(TObject *Sender)
{
   if (PointNum->Text == "") return;
   readpoint(StrToInt(PointNum->Text));
   setdata();
}
//---------------------------------------------------------------------------

void __fastcall TFDataChoose::OkBClick(TObject *Sender)
{
   FDataChoose->Close();
```

```
}
//---------------------------------------------------------------------------
```

Модуль Speed.cpp:

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Speed.h"
#include "Main.cpp"
#include "Progress.cpp"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
#define endstr "Визначення швидкодії завершено. Результати тесту в графічному вигляді \
подано в файлах t_of_k.m (залежність t(k)) та у файлі t_of_bitlen.m \
(залежність t([bitlen]a)). Для прогляду графіку потрібно відкрити файл \
в середовищі MatLab і виконати програму."

double bitlen[11];
double *k_average;                    //Залежність від числа k
double temp;
long kmin, step, steps, tests, kfora, in, jn;
verylong an, bn, knum, ktmp, kstep;
ecpoint p1;
TDateTime timeinfo, timestart, timeend, timediff;
unsigned short hour, min, sec, msec;
FILE *fst;

TFSpeed *FSpeed;
```

```
void savedata()
{
   fst = fopen("t_of_k.m", "w");
   fprintf(fst, "x=[");
   long k;
   k = StrToInt(FSpeed->KMin->Text);
   for (in = 0; in < steps; in++)
   {
         fprintf(fst, "%d ", k);
         k += step;
   }
   fprintf(fst, "]\ny=[");
   for (in = 0; in < steps; in++)
         fprintf(fst, "%f ", k_average[in]);
   fprintf(fst, "]\nplot(x,y,'r-'), hold on, grid on;\n");
   fprintf(fst, "title('Dependence of time of multiplication on a multiplier k');\n");
   fprintf(fst, "xlabel('Multiplier k');ylabel('Time of multiplication in
microseconds');");
   fclose(fst);

   fst = fopen("t_of_bitlen_a.m", "w");
   fprintf(fst, "x=[256 250 225 200 175 150 125 100 75 50 25]\ny=[");
   for (in = 0; in < 11; in++)
         fprintf(fst, "%f ", bitlen[in]);
   fprintf(fst, "]\nplot(x,y,'r-'), hold on, grid on;\n");
   fprintf(fst, "title('Dependence of time of multiplication on a bit length of
a');\n");
   fprintf(fst, "xlabel('Bit length of a');ylabel('Time of multiplication in
microseconds');");
   fclose(fst);
}
```

```
void gotest()
{
   temp = 0.0;
//  zintoz(step, &kstep);
   for (in = 0; in < steps; in++)
   {
         timestart = timeinfo.CurrentDateTime();
         for (jn = 0; jn < tests; jn++)
         {
               ecmulpoint(knum, p1);
         }
         timeend = timeinfo.CurrentDateTime();
         timediff = timeend - timestart;
         timediff.DecodeTime(&hour, &min, &sec, &msec);
         temp = msec + (sec*60.0) + (min*60*60) + (hour*60*60*60);
         temp /= (double) tests;
         k_average[in] = temp * 1000;
         zsadd(knum, step, &ktmp);
         zcopy(ktmp, &knum);
         FProgress->Prog->StepBy(1);
         FProgress->Repaint();
   }

   if (!(fst = fopen("SpeedTest.txt", "r")))
   {
         MessageBox(0, "Cannot open file SpeedTest.txt!", "Fatal error",
MB_ICONSTOP);
         delete[] k_average;
         Application->Terminate();
         return;
   }
```

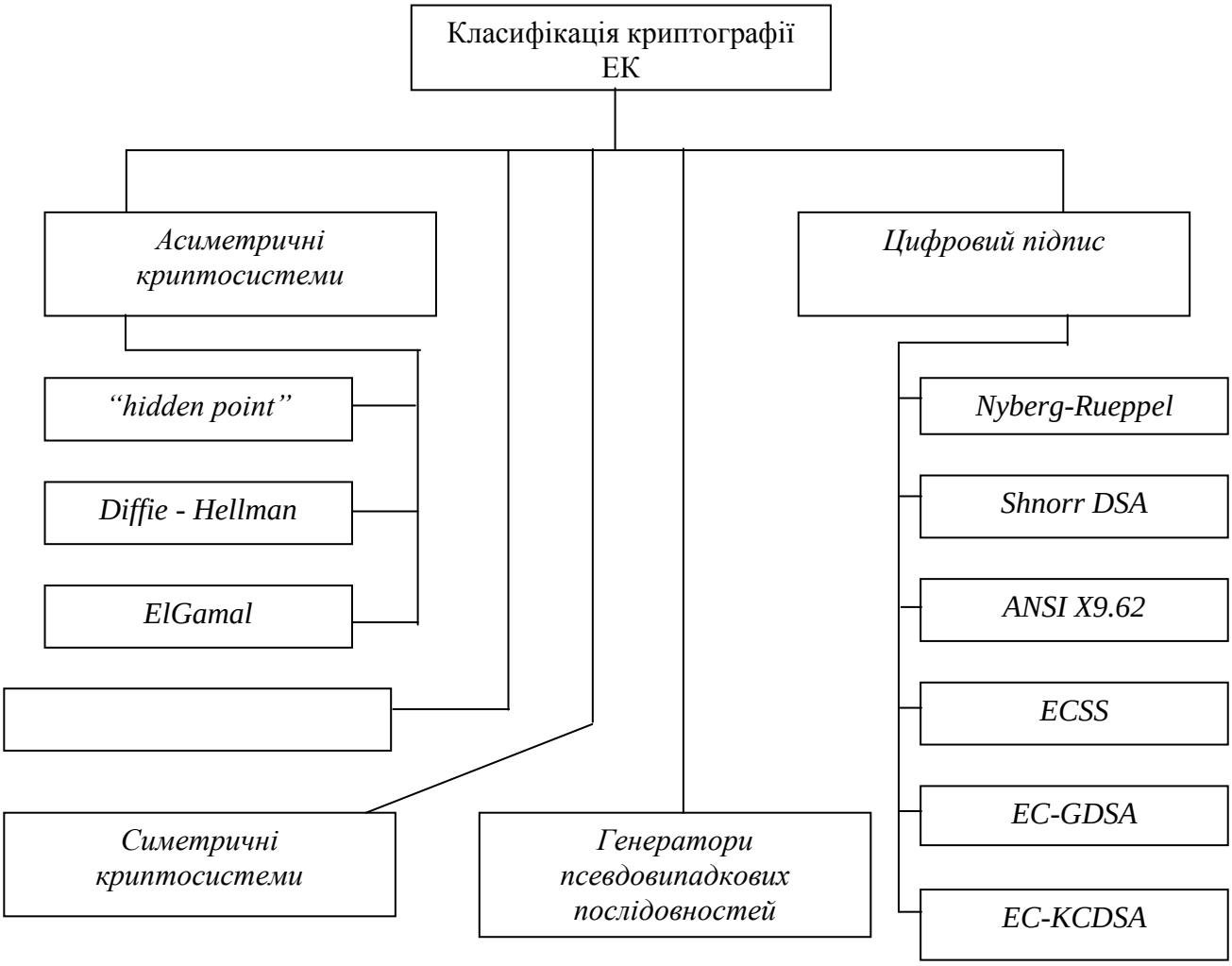
```
    ansitoz(FSpeed->KForA->Text, knum);
    for (in = 0; in < 11; in++)
    {
            getnextdata(fst, a, b, p1.x, p1.y);
            timestart = timeinfo.CurrentDateTime();
            for (jn = 0; jn < tests; jn++)
            {
                    ecmulpoint(knum, p1);
            }
            timeend = timeinfo.CurrentDateTime();
            timediff = timeend - timestart;
            timediff.DecodeTime(&hour, &min, &sec, &msec);
            temp = msec + (sec*60.0) + (min*60*60) + (hour*60*60*60);
            temp /= (double) tests;
            bitlen[in] = temp * 1000;
            FProgress->Prog->StepBy(1);
            FProgress->Repaint();
    }
    fclose(fst);
}

//---------------------------------------------------------------------------
__fastcall TFSpeed::TFSpeed(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TFSpeed::TestNumChange(TObject *Sender)
{
    if ((TestNum->Text == "") || (KPoints->Text == "")
```

```
    || (KStep == "") || (KMin == "") || (KForA->Text =="""))
    {
            OkB->Enabled = false;
            return;
    }
    else OkB->Enabled = true;

    if (StrToInt(TestNum->Text) < 20)
            TestNum->Text = "20";

    if (StrToInt(KStep->Text) < 1)
            KStep->Text = "1";

    if (StrToInt(KPoints->Text) < 3)
            KPoints->Text = "3";

    if (StrToInt(KForA->Text) < 1)
            KForA->Text = "1";
}
//---------------------------------------------------------------------------
void __fastcall TFSpeed::KMinKeyPress(TObject *Sender, char &Key)
{
    if (Key == '\b') return;
    if ((Key < '0') || (Key > '9')) Key = 0;
}
//---------------------------------------------------------------------------

void __fastcall TFSpeed::OkBClick(TObject *Sender)
{
    step = StrToInt(KStep->Text);
    steps = StrToInt(KPoints->Text);
```

```
    tests = StrToInt(TestNum->Text);
    kfora = StrToInt(KForA->Text);
    FProgress->Prog->Max = steps + 11;
    FProgress->Prog->Position = 0;
    FProgress->Show();

    k_average = new double [steps];

    zcopy(PointEC.x, &p1.x);
    zcopy(PointEC.y, &p1.y);
    ansitoz(KMin->Text, knum);

    gotest();
    savedata();

    MessageBox(0, endstr, "Тест завершено", MB_ICONINFORMATION);

    FProgress->Close();

    delete[] k_average;
}
//---------------------------------------------------------------------------

void __fastcall TFSpeed::CancelClick(TObject *Sender)
{
    FSpeed->Close();
}
//---------------------------------------------------------------------------
```

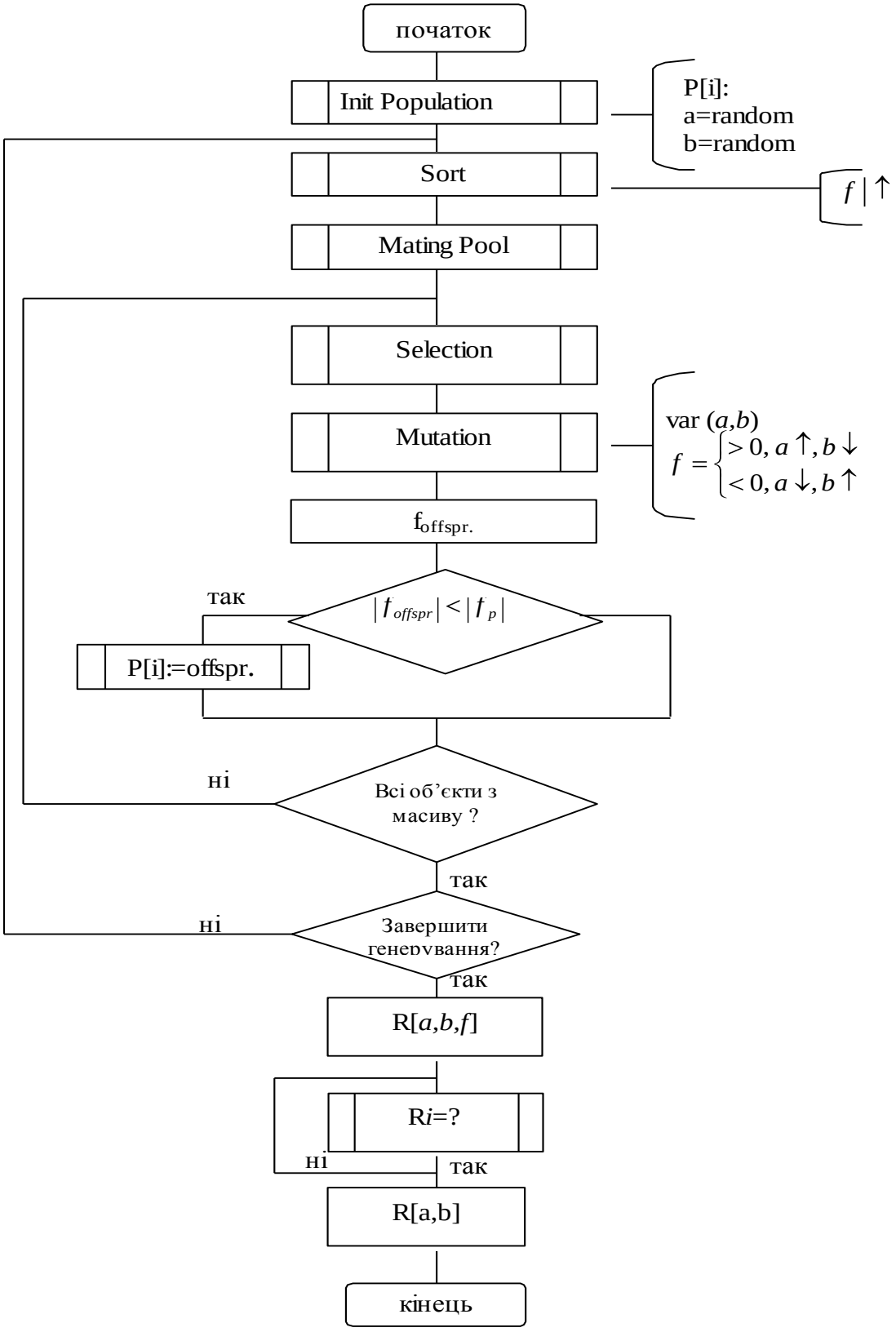
Додаток Б

Класифікація застосування криптографії еліптичних кривих



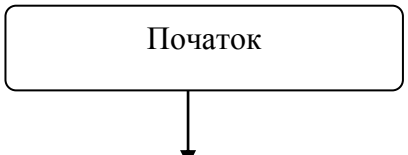
Додаток В

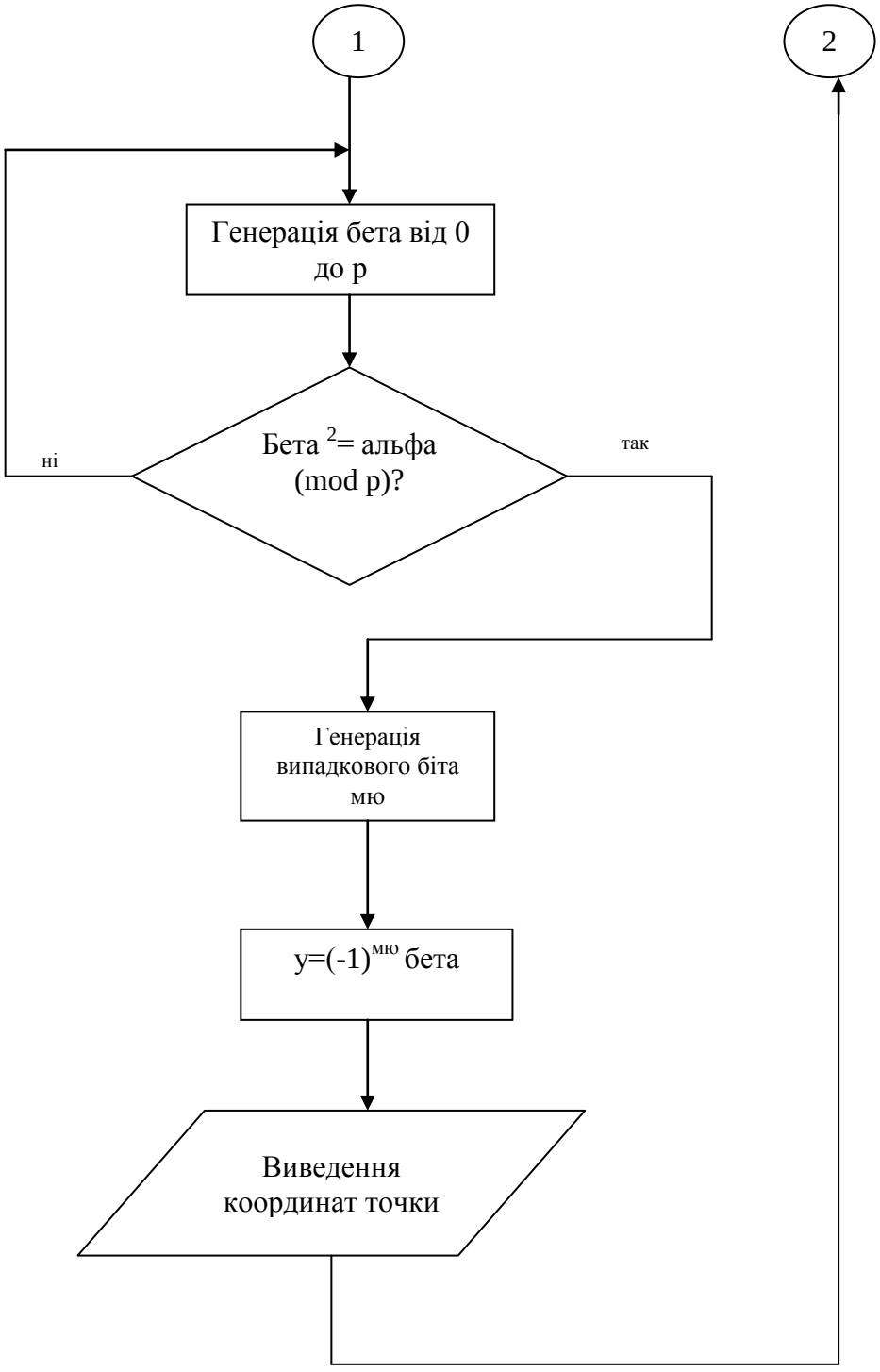
Схема алгоритму генерування параметрів еліптичної кривої



Додаток Г

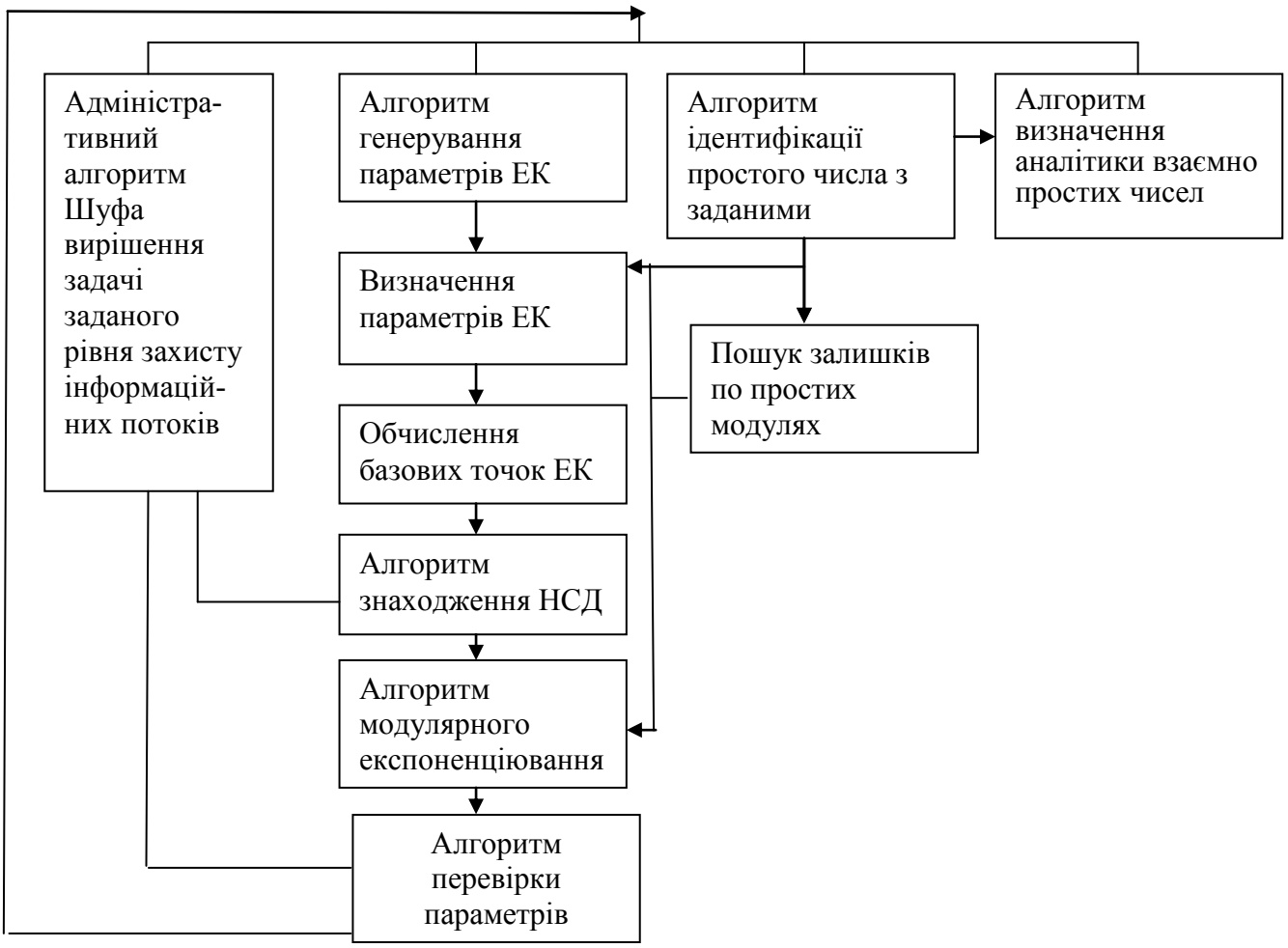
Схема алгоритму генерування випадкової точки на ЕК з відомими параметрами




1
2
Генерація бета від 0 до p
Бета ^{2}= альфа (mod p)?
ні
так
Генерація випадкового біта мю
y=(-1)^{мю} бета
Виведення координат точки

Додаток Д

Структурна схема організації захисту інформаційних потоків з використанням математичного апарату еліптичних кривих


Адміністра-
тивний
алгоритм
Шуфа
вирішення
задачі
заданого
рівня захисту
інформацій-
них потоків
Алгоритм
генерування
параметрів ЕК
Алгоритм
ідентифікації
простого числа з
заданими
Алгоритм
визначення
аналітики взаємно
простих чисел
Визначення
параметрів ЕК
Пошук залишків
по простих
модулях
Обчислення
базових точок ЕК
Алгоритм
знаходження НСД
Алгоритм
модулярного
експоненціювання
Алгоритм
перевірки
параметрів