**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**

**КАФЕДРАКОМП’ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ**

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_ С.В. Казмірчук?

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2019 р.

На правах рукопису

УДК 003.26:004.056.55

**ДИПЛОМНА РОБОТА**

**ВИПУСКНИКА ОСВІТНЬО СТУПЕНЯ**

**«БАКАЛАВР»**

**Тема**: Клас еліптичних кривих для забезпечення цілісності інформації

|  |  |
| --- | --- |
| **Автор:** | Я.В. Яценко |
| **Науковий керівник:** к.т.н., доц. | В.П. Щербина |
| **Нормоконтролер:**к.т.н., доц. | В.П. Щербина |

**Київ 2019**

**НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**

**Факультет:** Кібербезпеки комп’ютерної та програмної інженерії

**Кафедра:** Комп’ютеризованих систем захисту інформації

**Освітній ступінь:** «Бакалавр»

**Напрям:**6.170101«Безпека інформаційно-комунікаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_С.В. Казмірчук ?

«\_\_\_»\_\_\_\_\_\_\_\_2019 р.

**ЗАВДАННЯ**

**на виконання дипломної роботи**

**студенту Яценку Ярославу Валерійовичу**

Тема: *Клас еліптичних кривих для забезпечення цілісності інформації*

затверджена наказом ректора від 28.02.2019 № 481/ст.

1. Термін виконання: з 09.05.2019 р. по 26.06.2019 р.
2. Вихідні дані: ?!дослідити основні поняття випадковості, методи генерування псевдовипадкових двійкових послідовностей; на основі проведеного аналізу сформулювати методику оцінки рівня псевдовипадковості тритових послідовностей, та реалізувати її програмно у вигляді програмного комплексу оцінки рівня псевдовипадковості трійкових послідовностей для квантових систем захисту інформації.
3. Зміст пояснювальної записки (перелік питань, що підлягають розробці): процеси обміну ключами, шифрування/дешифрування та створення/перевірка електронно-цифрового підпису у еліптичних криптосистемах.

**КАЛЕНДДРНИЙ ПЛАН**

**виконання дипломної роботи**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п/п** | **Етапи виконання дипломної роботи** | **Термін виконання етапів** | **Примітка** |
|  | Уточнення постановки задачі | 24.12.2012 | Виконано |
|  | Аналіз літературних джерел | 02.01.2013 |  |
|  | Обґрунтування рішення | 11.01.2013 |  |
|  | Збір інформації | 16.01.2013 |  |
|  | Аналіз понять випадковості та існуючих методів генерації псевдовипадкових бінарних послідовностей | 12.02.2013 |  |
|  | Аналіз та дослідження принципів оцінки псевдовипадковості бінарних послідовностей | 30.03.2013 |  |
|  | Розробка та опис методики оцінки ступеня псевдовипадковості тритових послідовностей | 24.04.2013 |  |
|  | Розробка та дослідження програмного комплексу оцінки рівня псевдовипадковості тритових послідовностей | 26.04.2013 |  |
|  | Апробація роботи на IV науково-практичній конференції «Інтегровані інтелектуальні робото-технічні комплекси» | 28.04.2013 |  |
|  | Оформлення і друк пояснювальної записки | 29.04.2013 |  |
|  | Оформлення презентації | 30.04.2013 |  |
|  | Отримання рецензій від опонентів | 10.05.2013 |  |
|  | Захист в ЕК | 17.05.2013 |  |

Дипломник Т.О. Жмурко

(підпис, дата)

Дипломний керівник С.В. Карпенко

(підпис, дата)

**РЕФЕРАТ**

**Актуальність теми**: На сьогоднішній день галузь шифрування та

захисту інформації відіграє важливу роль в царині інформатики. Це

зумовлено масштабним використанням автоматизованих методів обробки та

передачі даних та широким розповсюдженням методів та засобів несанкціонованого доступу до інформації, що пересилається незахищеними

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

Одним з яскравих представників сімейства криптографічних алгоритмів з

використанням еліптичних кривих є алгоритм електронно-цифрового підпису

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

еліптичної кривої на число. Таким чином, актуальними є дослідження способів та методів оптимізації даного обчислення.

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

**ЗМІСТ**

**СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ**

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

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

Відкритий (вихідний) текст – дані, що зашифровуються.

Шифротекст – дані, отримані після застосування шифрування.

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

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

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

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

ЕК – еліптична крива.

RAII-

NAF – non-adjacent form, неспарена форма.

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

ECDSA – elliptic curve digital signature algorithm, алгоритм електронно-

цифрового підпису на еліптичних кривих.

ECDLP – elliptic curve digital signature problem, проблема дискретного

логарифму на еліптичних кривих.

DBNS – double-based number system, двобазисна система.

**ВСТУП**

У сучасному світі нестримними і колосальними темпами відбувається процес інформатизіції, диджиталізації або ж оцифровки уього, що можна оцифрувати.

Інформація змінює ареал свого існування, невпинно перебираючись із сторінок підручників, шпальт газет або ж аналогових фотокарток в царину кремнієвих транзисторів, в безкінечну послідовність із нулів та одиниць.

Ця зміна породжує доволі велику кількість нових вимог до інформації та її електронних носіїв. Однією із найважливіших цілей є забезпечення захищеності інформації від неавтенифікованої зміни та доступу до неї, адже як відомо, хто володіє інформацією, той володіє світом. Більше того надзвичайно важливим стає питання дотримання вищезазначених принципів при передачі інформації через мережу Інтернет. У зв'язку з цим використовуються алгритми шифруання данних і така наука, як криптграфія сьогодні набула великої популярності.

Не варто забувати про сфери людської діяльності,які оперують із персональними даними. Основною проблемою у цих сферах є засвідчення оригінальності даних.

Для вирішення цих важливих питань використовуються симетричні та асиметричні алгоритми шифрування даних. У рамках цієї роботи буде розглянуто алгоритми, що базуються на операціях над еліптичними кривими.

З метою підтвердження оригінальності даних, що пересилаються мережею інтернет в сучасному світі часто використовується технологія електронно-цифрового підпису. Ця технологія є певним аналогом реального підпису, коли люди, між якими здійснюється певний договір підписуються в документах і таким ином засвідчують свою особистість.

Використання еліптичних кривих для створення криптосистем було незалежно запропоновано Нілом Коблицем та Віктором Міллером у 1985 році. Ключова перевага даної методології ґрунтується на двох основних особливостях точок еліптичної кривої: розподіл результату додавання точок еліптичної кривої є у великій мірі рівномірним, тобто покриває всю область значень даної кривої; наслідком цієї особливості є проблема дискретногологарифмування для еліптичних кривих (ECDLP). Ці особливості і обумовлюють високу криптографічну стійкість систем з використанням еліптичних кривих. З іншого боку, слід зазначити, що криптостійкість забезпечена відсутністю субекспоненційних алгоритмів вирішення проблеми дискретного логарифмування. Таким чином, знаходження субекспоненційного підходу унеможливить використання еліптичних кривих криптографії, хоча це і стосується в однаковій мірі і інших схем шифрування.

Предмет дослідження: операції над еліптичними кривими у полі GF( p ) та GF( 2m ).

Мета дослідження: розробити клас для роботи із еліптичними кривими.

Методи дослідження: методи теорії чисел, аналітичної геометрії,

методи абстрактної алгебри, дискретної математики та криптографії.

Наукова новизна полягає в наступному:

1. Створено модифікацію методу множення точки еліптичної кривої на число, яка полягає у використанні аспеціального алгоритму множення чисел, що був запропонований у 1960 році Карацубою. Принципи функціонування алгоритму будуть наведенні нище за текстом.

2. Розроблено програмну бібліотеку, що надає можливість розширити функціональність мови С++ для роботи із надвеликими числами та дозволяє працювати із еліптичними кривими, що в свою чергу, дає можливість використовувати мову програмування С++ для написання складних алгоритмів шифрування і дешифрування даних на основі еліптичних кривих.

Практична цінність результатів полягає в наступному: запропоновані математичні та програмні алгоритми роблять можливим пришвидшння виконання операції множення числа нп точку еліптичної кривої на 25% за умови, що точка еліптичної має розрядність більше 2^100. Це є надзвичайно корисно у подальшому вивчені та дослідженні криптографії на основі еліптичних кривих, адже ми можем зекономити дорогоцінний час.

Структура пояснювальної записки:

У розділі 1 розглядається загальний контекст досліджень, а саме –

основи еліптичної криптографії та математичного інструменту, який робить еліптичну криптографію можливою, популярні алгоритми, створені на основі еліптичних кривих.

У розділі 2 розглядаються принципи реалізації операцій над точками

еліптичної кривої, розглядаються основні алгоритми виконання операцій додавання точок та множення числа на точку еліптичноїкривої.

У розділі 3 розглядається метод пришвидшення множення Карацуби,

проаналізовано суть алгоритму так само було порівняно його із іншими алгоритмами пришвидження операцій над числами.

У розділі 4 розроблено програмне забезпечення , що дозволяє прицювати із числами великої розрядності, виконувати операції над еліптичними кривими і дозволяє досягти пришвидшення до 25% при виконання множення числа на точку еліптичної кривої.

Ключові слова: еліптична криптографія, електронно-цифровий підпис,

ECDSA, багатократне множення точок еліптичної кривої на число, об’єктно-

орієнтована архітектура програмного забезпечення.

1. **АНАЛІЗ КРИПТОСИСТЕМ, НА ОСНОВІ ЕЛІПТИЧНИХ КРИВИХ**

**1.1.** **Принципи еліптичної криптографії**

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

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

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *y*2 *+ a xy + a* | | *y = x*3 *+ a* | 2 | *x*2 *+ a* | 4 | *x + a* | 6 | (1) |  |
| 1 | 3 |  |  |  |  |  |

де {*a*1 , *a*2 , *a*3 , *a*4 , *a*5 , *a*6 }*K*, 0.

* називається дискримінантом *Е* і визначається як:
  +  *d*22 *d*88*d*4327*d*629*d*2 *d*4 *d*6

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *d* | | 2 | | *= a*2 | *+* 4*a* | | 2 | | |  |  |  |  |  |  |
|  |  |  |  |  |  |  | 1 |  |  |  |  |  |  |  |  |
|  |  |  |  |  | *d*4 *=* 2*a*4 *+ a*1*a*3 | | | | | | | | | |  |  |  |  |  |  |
|  |  |  |  |  | *d* | 6 | | | *= a*2 | *+* 4*a* | | | 6 | |  |  |  |  |  |  |
|  |  |  |  |  |  | 3 |  |  |  |  |  |  |  |  |  |
| *d* | 8 | *= a*2 *a* | | 6 | *+* 4*a* | | | *a*  *a a a* | | | | | | 4 | *+ a* | *a*2 | |  *a* | 2 . |  |
|  | 1 |  |  |  |  | 2 | 6 | 1 | 3 |  |  | 2 |  | 3 |  | 4 |  |
| Якщо скінченне поле | | | *K* є простим– | | | | | | | | *GF* ( *p*), | | | | | | *E* трансформується у | | |  |
| криву, що описується таким рівнянням: | | | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |
|  |  | *y*2 *= x*3 *+ ax*2 *+ b*. | | | | | | | | | |  |  |  |  |  |  |  | (2) |  |
| У випадку *K = GF* (2*m* ) | | | | | *E* |  |  |  | трансформується | | | | | | | | | у | криву, описану |  |
| рівнянням: |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 2 *+ xy = x*3 *+ ax*2 *+ b* , | | | | | | | | | | | |  |  |  |  |  | (3) |  |

де *a*, *b*  *K* .

Розглянемо арифметику над точками еліптичної кривої.

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

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

Подвоєння точки *P* графічно визначається наступним чином.

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

***Вимоги до еліптичної кривої***

Для того, щоб уникнути відомі атаки, що засновані на проблемі дискретного логарифма у групі точок ЕК, необхідно, щоб кількість точок ЕК ділилась на достатньо велике просте число *n* . Стандарт ANSI X9.62 вимагає *n* 2160.Рівняння ЕК будується специфічним способом,використовуючивипадкові/псевдовипадкові коефіцієнти.

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

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

|  |  |
| --- | --- |
| *y* 2 *x* 3 *ax*  *b* , | (4) |

де *a* ,*b* *GF*  *p*  та 4 *a* 3  27 *b* 2  0 mod *p* .

1. Два елементи поля *GF*  *p*   *xG* та *yG* , які визначають кінцеву точку *G*  *xG* , *yG*   генератор групи.
2. Порядок *q* точки *G* , де *q*  2160 та *q*  4 *p* .
3. Співмножник *h*  # *E* / *q* , де # *E* означає порядок групи точок ЕК.



***\\Корекція формату \\Генерація основних параметрів***

Один із способів генерації криптографічно надійних параметрів полягає в наступному:

1. Обираємо коефіцієнти *a* та *b* специфічним чином, використовуючи в обчисленнях випадкові/псевдовипадкові числа. Нехай *E*  еліптична крива *y* 2  *x* 3  *ax*  *b* [3].
2. Обчислюємо *N*  # *E* .
3. Перевіряємо, що *N* має дільник, який є великим простим числом

*q* *q* 2160, *q* 4 *p* .Якщо не виконується,то необхідноповернутись на крок 1

1. Перевіряємо, що *q* не ділить *pk* 1 для кожного *k* , 1  *k* 100 . Якщо не виконується, то необхідно повернутись на крок 1.
2. Перевірити, що *q*  *p* . Якщо не виконується, то необхідно повернутись на крок 1.

6. Обрати випадкову точку *G*  *E* і покласти *G* *N* / *q* *G*.

Повторювати, доки *G*  *O* .

**1.2.** **Методи шифрування в еліптичній криптографії**

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

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

та асиметричний (існує пара ключів – приватний та публічний, що може бути переданий незахищеним каналом). В межах даної роботи детально розглядається другий варіант, а саме асиметричне шифрування.

***1.2.1 Шифрування з відкритим ключем***

***Еліптичний варіант схеми обміну ключами Діффі-Хелмана*** Протокол Діффі-Хелмана – це метод обміну криптографічними

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

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

*y* 2 *x* 3 *ax*  *b*

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

1. Сторона *A* обирає ціле число *Priv* *A*  *n* . Дане число називається

приватним ключем учасника, а точка еліптичної кривої *PubA* Pr*iv A* *G* називається публічним ключем.

1. Сторона *B* обирає аналогічно секретний ключ Pr *ivB* та обчислює відкритий ключ *PubB*  Pr*ivB* *G* .
2. Учасник *A* генерує секретний ключ *K*  *k* *A*  *PB* , а учасник *B* генерує секретний ключ *K*  *k* *B*  *PA* .

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

1. *A*  *PB*  *k A*  *k B*  *G*  *k B*  *k A*  *G*  *k B*  *PA* .

Проблема, що стоїть перед сторонніми спостерігачами, які мають намір дізнатись секретний ключ, полягає в обчисленні *k* *A*  *k* *B*  *G* за відомими *G* , *k A*  *G* , *k B*  *G* ,але не знаючи *kA* та *kB* .Це і є проблемою Діффі-Хеллманадля еліптичних кривих.

***Протокол Мессі-Омура (Massey-Omura)***

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

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

1.  *e* 1mod *N*

За визначенням,

*e*  *d*  *j*  *N* 1

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

( *e*  *d* )  *Q*  ( *j*  *N*  1) *P*  ( *j*  *N* )  *P*  *P*  *P*  *O*  *P*

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

1.  *e*  *P P*  *d* *Q*

Де *P*  *R* . Обчислення точки *P* по *e*  *P* еквівалентно вирішенню завдання дискретного логарифма для еліптичної кривої.

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

1. Сторони *A* та *B* генерують пари ключів *( PubA* *,Pr ivA* *)* для сторони
2. та *( PubB* *,Pr ivB* *)* для сторони *B* , де *( Pub,Pr iv )* – публічний та

приватний ключі.

1. Сторона *B* надсилає незахищеним каналом стороні *A* свій публічний ключ – *PubB* .
2. Сторона *A* виконує шифрування повідомлення *m* за допомогою

деякої функції *Enc* , що приймає аргументи *Pub* та m. Результатом виконання даної функції є деяке значення *e*  *Enc( PubB* *,m )* .

1. Сторона *A* надсилає стороні *B* незахищеним каналом зашифроване повідомлення *e* .
2. Сторона *B* отримує зашифроване повідомлення *e* , та за допомогою

деякої функції *Dec* , що приймає аргументи *Pr iv* та *e* , отримує вихідне повідомлення *m*  *Dec(Pr ivB* *,e )* .

Для зворотнього зв’язку виконуються аналогічні дії, де відправником є сторона *B* , отримувачем – сторона *A*, і замість *( PubB* *,Pr ivB* *)* виконуються маніпуляції з парою ключів *( PubA* *,Pr ivA* *)* .

***2.3 Опис алгоритму електронно-цифрового підпису***

Ще однією сферою, де активно застосовується еліптична криптографія,

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

1. Сторона *A* обирає деяке значення *Pr iv* *A* , що є числом великої

розрядності. Обчислюється точка еліптичної кривої *PubA*  *Pr ivA**G* ,де *G* –генеруюча точка еліптичної кривої,тобтоточка з великим порядком. Точка *PubA* називається електронно-цифровим підписом відправника *A* .

1. Обчислюється хеш-сума повідомлення використовуючи деяку функцію *Hash* , *h*  *Hash( m )*.
2. Використовуючи електронно-цифровий підпис *PubA* , виконується підпис хеш-суми повідомлення, використовуючи деяку функцію

*Sign* ,що приймає в якості параметрів хеш та електронно-цифровийпідпис, *signature*  *Sign( h,PubA* *)* .

1. Відправнику *B* надсилається *( m,signatre,PubA* *)* .

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

1. Обчислюється хеш повідомлення *h*  *Hash( m )*.
2. Використовується деяка функція *Verify* , що приймає параметрами хеш-суму *h* , підпис повідомлення *signature*, електронно-цифровий підпис *PubA* , та повертає булеве значення

*verified* *Verify( h,signature,PubA )* .

1. Якщо значення *verified*  *true* , то повідомлення є оригінальним та не було змінене в процесі передачі незахищеним каналом.

**1.3.** **Алгоритми генерування та перевірки цифрового підпису**

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

***1.3.1. Схема цифрового підпису Ель-Гамаля***

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

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

Генерація ключів:

1. Генерується випадкове просте число довжиною *p* бітів.
2. Вибирається випадковий примітивний елемент *g* *p* .
3. Вибирається випадкове ціле *x* таке що 1  *x*  *p* 1.
4. Обчислюється *y*  *g* *x* mod *p* .
5. Відкритим ключем є трійка ( *p* , *g* , *y*) , а закритим число *x* .

Алгоритм підпису повідомлення *M* :

1. Обчислити хеш повідомлення: *m*  *h* ( *M* ) .
2. Вибрати випадкове число 1  *k*  *p* 1 взаємно просте з *p* 1 і обчислюється *r*  *g* *k* mod *p* .
3. Обчислити число *s*  ( *m*  *xr* ) *k* 1 mod *p* 1.
4. Підписом повідомлення *M* є пара ( *r* , *s*) .

Перевірка підпису:

1. Перевіряється справедливість умов: 0  *r*  *p* і 0  *s*  *p* 1. Якщо хоча б одна з них не виконується, то підпис вважається невірним.
2. Обчислюється хеш повідомлення *m*  *h* ( *M* ) .

3. Підпис вважається вірним, якщо виконується порівняння:

*y r r s*  *g m* mod *p*

***1.3.2. Алгоритм ECDSA***

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

*GF*  *p* .

***Генерація ключів ECDSA***

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

1. Обирає випадкове ціле число *x* з інтервалу 1, *q* 1.
2. Обчислює добуток *Q*  *x*  *P* .

Відкритим ключем відправника *A* є точка *Q* , а закритим  *x* .

***Обчислення цифрового підпису***

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

1. Обрати випадкове ціле число *k*  1, *q* 1.

2. Обчислити *k*  *P* *x*1, *y*1 та *r*  *x*1mod *q* .Якщо *r* 0mod *q* , то

обирають нове випадкове число *k*  1, *q* 1.

3. Обчислити *k* 1mod *q*  та *s*  *k* 1*h*  *x*  *r* mod *q* ,де *h* значення

хеш-функції повідомлення, що підписується. Якщо *s* 0, то

значення *s* 1 mod *q* не існує, тому необхідно повернутись до п.1.

Підписом для повідомлення є пара цілих чисел *r* , *s* .

***Перевірка цифрового підпису***

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

1. Отримати копію відкритого ключа *Q* відправника *A*.
2. Перевірити, що числа *r* та *s* є цілими числами з інтервалу 1, *q* 1 та обчислити значення хеш-функції *h* від повідомлення.
3. Обчислити *u*1  *s* 1*h* mod *q* та *u* 2  *s* 1*r* mod *q* .
4. Обчислити *u*1  *P*  *u* 2  *Q*  *x*0 , *y*0  та *v*  *x*0 mod *q* .
5. Прийняти підпис, якщо *v*  *r* .

Можна легко показати, що даний алгоритм дійсно успішно засвідчує цифровий підпис. Якщо підпис *r* , *s*  повідомлення *m* було дійсно згенеровано з використанням секретного ключа, то *s*  *k* 1*h*  *xr* mod *q* .

Розкриваючи дужки отримаємо:

*k*  *s* 1*h*  *xr*  *s* 1*h*  *s* 1*rx*  *u*1 *u* 2 *x* mod *q* .

Тоді *u*1  *P*  *u* 2  *Q*   *u*1  *u* 2 *x*  *P*  *kP* і значить *v*  *r* , що і вимагається для підтвердження підпису.

**1.4.** **Висновки до розділу 1**

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

* методах шифрування еліптичної криптографії та схемах цифрового підпису, що ґрунтуються на властивостях адитивної абелевої групи,

утвореної точками ЕК, так само, найбільш вживаною є операція множення точки ЕК на число.

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

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

**АНАЛІЗ ОПЕРАЦІЙ НАД ТОЧКАМИ ЕЛІПТИЧНИХ КРИВИХ**

межах криптографії, активне застосування знаходять еліптичні криві

параметрами, визначеними над скінченними полями, а саме полями з

простою характеристикою – *GF*  *p*  та бінарним розширенням – *GF* (2 *m* ) . Це в свою чергу означає, що координати точок еліптичної кривої належать деякому скінченному полю, що знімає проблему округлення значень при виконанні операцій над ними.

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

**2.1.** **Алгебраїчні операції в скінченних полях**

даному розділі розглядаються особливості побудови скінченних полів, або полів Галуа, зокрема полів *GF*  *p*  та *GF* (2 *m* ) . Вводяться поняття

характеристики поля, мультиплікативної групи, незвідного полінома, додавання, віднімання, множення елементів поля, знаходження оберненого елемента тощо. Для подання однозначного і вичерпного визначення скінченного поля необхідно дати декілька математичних визначень, що використовуються для формулювання означення поля.

***Група***

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

Асоціативність: для довільних елементів *a* , *b* , *c* групи *G* виконується правило ( *ab* ) *c*  *a* (*bc*) .

Існування нейтрального елемента: існує елемент *e* такий, що для кожного елемента *a* групи *G* виконується *ae*  *ea*  *a* .

Існування оберненого елемента: для кожного елемента *a* групи *G* існує елемент *a*1 такий, що *aa* 1  *e* .

***Кільце***

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

Дистрибутивність: для довільних елементів *a* , *b* , *c* групи *G* виконується правило ( *a*  *b* )*c*  *ac*  *bc* .

Існування нейтрального елемента: існує елемент *e* такий, що для кожного елемента *a* групи *G* виконується *ae*  *ea*  *a* .

Існування адитивної групи кільця і нейтральний елемент в ній позначають як 0.

***Скінченне поле***

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

Для будь-якого простого числа *p* кільце лишків mod( *p*) – це скінчене

поле з елементів, яке позначається *GF*  *p*  / *p* Елементи цього поля



можуть бути представлені цілими числами, для них визначено операції додавання та множення за модулем *p* . Будь-яке скінчене поле містить *pn* елементів і однозначно задається своєю характеристикою *p* і степенем *n* . Загалом, все описане вище, можна сформулювати наступним чином. Полем *F* (,\*)є множина *F* ,на якій визначені дві операції:додавання та множення,для яких виконуються наступні аксіоми:

Комутативність додавання *a*, *b*  *F* : *a*  *b*  *b*  *a* .

Асоціативність додавання *a* , *b*, *c*  *F* : ( *a*  *b* )  *c*  (*b*  *c* )  *a* .

Існування нульового елемента 0  *F* : *a*  *F a*  0  *a* .

4. Існування оберненого елемента відносно додавання

*a*  *F*  *a*  *F* : *a*  *a* 0.

Комутативність множення *a*, *b*  *F* : *a* \* *b*  *b* \* *a* .

Асоціативність множення *a* , *b*, *c*  *F* : ( *a* \* *b* ) \* *c*  (*b* \* *c* ) \* *a* .

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 7. | Існування нейтрального елемента *e*  *F* : *a*  *F a* \* *e*  *a* . | | | | | |
| 8. | Існування | нейтрального | | елемента | відносно | множення |
|  | *a*  *F* *a* 1 *F* : *a* \* *a* 1 *e* . | | |  |  |  |
| 9. | Дистрибутивність | | додавання | | відносно | множення |
|  | *a* , *b*, *c*  *F* : ( *a*  *b* ) \* *c*  *a* \* *c*  *b* \* *c* . | | | |  |  |

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

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

Табл. 1. Елементи поля *GF* (2 4 )

|  |  |  |
| --- | --- | --- |
| **Многочлен** | **Степінь** | 1**,** *x* , *x*2 , *x*3 |
| ** | ** | (0,1,0,0) |
|  |  |  |
| **2 | **2 | (0,0,1,0) |
| **3 | **3 | (0,0,0,1) |
| 1  ** | **4 | (1,1,0,0) |
| ****2 | **5 | (0,1,1,0) |
| **2****3 | **6 | (0,0,1,1) |
| ** 3  ** 1 | **7 | (1,1,0,1) |
| ** 2 1 | **8 | (1,0,1,0) |
| **3**** | ** 9 | (0,1,0,1) |
| ** 2  ** 1 | **10 | (1,1,1,0) |
| **3****2**** | **11 | (0,1,1,1) |
| ** 3  ** 2  ** 1 | **12 | (1,1,1,1) |
| ** 3  **2 1 | **13 | (1,0,1,1) |
| **3 1 | **14 | (1,0,0,1) |
| 1 | **15 | (1,0,0,0) |

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

1. *u* : *a* ; *v* : *b* ; *A* :1; *B* :0; *C* :0; *D* :1.

Поки *v*  0

2.1. *q* :  *u*  ;

*v* 

2.2. *t*1: *u*  *q*  *v* ; // *u* mod *v*

*t* 2: *A*  *q*  *C* ;

*t*3: *B*  *q*  *D* ;

2.3. *u* : *v* ;

*A* : *C* ;

*B* : *D* .

2.4. *v* : *t*1;

: *t*2 ; *D* : *t*3.

*d* : *u* ; *x* : *A* ; *y* : *B* .

Результат: *d* , *x* , *y* , *inv* .

**2.2. Додавання та подвоєння точок еліптичної кривої** Введемо наступні правила додавання точок ЕК:

Точка *O* виступає в ролі нульового елементу. Таким чином, *O* *O* та для будь-якої точки *P* на еліптичній кривій *P*  *O*  *P* .

Вертикальні лінія перетинає криву в двох точках з однією й тією ж

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| координатою *x* . | | Наприклад, *S*  *x* , *y*  | | та *T*   *x* , *y* . Ця | пряма |
| перетинає криву | | і | в нескінченно | віддаленій точці. | Тому |
| *P*  *P*  *O*  *O* та | | *P* *P* . | |  |  |
| 1 | 2 | 1 | 2 |  |  |

Для того, щоб додати дві точки *P* та *Q* з різними координатами *x* , необхідно провести через ці точки пряму та знайти точку перетину її з ЕК. Якщо пряма не є дотичною до кривої в точках *P* або *Q* , то

існує тільки одна така точка, позначимо її *S* . Відповідно до нашого припущення, *P*  *Q*  *S*  0 *P*  *Q* *S* або *P*  *Q*  *T* . Якщо пряма є дотичною до кривої в якій-небудь з точок *P* або *Q* , то в цьому випадку варто покласти *S*  *P* або *S*  *Q* відповідно.

Щоб подвоїти точку *Q* , варто провести дотичну в точці *Q* та

знайти іншу точку перетину *S* з ЕК. Тоді *Q*  *Q*  2*Q* *S*.

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

Множина точок ЕК має такі властивості:

*P*  *O*  *P*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2. | Якщо *P*  *x* , *y* , | то | *P*  *x* , *y*  *O*. | Точка *x* ,  *y*  є від’ємним |
|  | значенням точки *P* і позначається *P*. Зазначимо, що *x* ,  *y*  також | | | |
|  | лежить на ЕК. |  |  |  |
| 3. | Якщо *P*  *x*1 , *y*1  | та | *Q* *x*2, *y*2,де | *P*  *Q* ,то *P*  *Q* *x*3, *y*3 |

визначається за наступними формулами:

*x*3 **2 *x*1 *x*2mod *p* 

*y*3 ** *x*1 *x*3 *y*1mod *p* 

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  *y* 2 | |  *y*1 | | , | | *якщо P*  *Q* |  |
|  |  |  |  |  |
| *x* |  *x* | |  |
|  | 2 | 1 |  |  |  |  |  |
| де **  | 3*x* 2  *a* | | |  |  |  |  |
|  |  |  |  |  |
| 1 | |  |  | , | *якщо P*  *Q*. |  |
|  |  |  |  |  |  |
| 2 *y* | | |  |  |
|  |  |  |  |
|  |  | 1 |  |  |  |  |  |

Число ** є кутовим коефіцієнтом січної [13], проведеної через точки *P*

та *Q*

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

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

**2.3. Особливості програмної реалізації операцій над точками еліптичної кривої**

Еліптичні криві з параметрами, що визначені над скінченними полями мають ряд істотних переваг у порівнянні з тими, що визначені над раціональними числами, а саме – теоретична ефективність реалізації в обчислювальній техніці. Це обумовлено тим, що у випадку поля *GF* ( *p*) , числа можуть бути легко поданими у вигляді бітових слів, адже обчислювальні пристрої оперують саме з числами у двійковому поданні, а операція обчислення за модулем може бути представлена як звичайна булева функція XOR. Для поля *GF* (2 *m* ) , многочлени, що є елементами поля можуть бути представлені у вигляді кодових двійкових слів, де кожен розряд позначає коефіцієнт при доданку-степені. Наприклад, многочлен ** 3  **2 1 може бути представлений на апаратному рівні у вигляді кодового слова **«**1011»або1\***31\***20 \***11\***0**3**21**.**

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

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

4  *P*.

4  *P*,

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

*O* *O*

*O*  *P*  *P*

*k* \**O*  *O*

точки зору об’єктного проектування, операції над точками еліптичної кривої є за своєю суттю надбудовою над операціями в скінченних полях.

Деталі програмної реалізації даних модулів будуть детальніше описані в розділі 4.

***Множення точок еліптичної кривої на 3 та 4***

Розглянемо докладніше алгоритми множення точки ЕК на 3 та 4 для ЕК у формі Вейерштраса. Найпростішим способом виконання обчислення 3  *P* та 4  *P*[14], є подвійне подвоєння та додавання плюс подвоєння. В той же час

існує інший підхід, який ґрунтується на обчисленні 3  *P* та використовуючи тільки координати точки *P*.

Існують три найкращі способи обчислення 3  *P* та Спосіб 1:

*t*12 *y*12;

*t* 23 *x*12 *a*;

*t*3 *t*22;

 *t*13 *x*1 *t*3;

*t* 4 *d* 2 *y*1; *inv*  *t*41;

**1 *d*  *inv* *t*2;

**2 *t*12 *inv*  **1;

*y*3 *x*1 *x*3**2 *y*1.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Спосіб 2: | |  |  |  |  |  |  |  |
| *inv*12 *y*11; | | |  |  |  |  |  |  |
| **1   3*x*12  *a* *inv*1; | | | | | |  |  |  |
|  |  |  *a*  | | 2 | 12 *x*1 *y*12 | 1 | |  |
| *inv*23 *x*12 | |  |  | ; |  |
|  |  |  |  |  |  |  |  |  |
| ** |  **  8 *y* 3 | | *inv* ; | | |  |  |  |
| 2 | 1 | 1 |  |  | 2 |  |  |  |

*x*3 **2 **1 **2 **1 *x*1;

*y*3 **2**2 **1 **2 **1 *y*1.

даних алгоритмах, при обчисленні мультиплікативно оберненого

елемента необхідно перевіряти аргумент на рівність нулю. Нехай *P*  *x*1 , *y*1  , і необхідно обчислити 4 *P*  *x*4 , *y*4 .

Припустимо, що:

  3 *x* 2  *a*  12 *x*1 *y*1  3 *x* 2  *a* 2   8 *y*14.



Легко побачити, що *d*  8 *y*1 *y*3 , де *y*3  це координата *y* результуючої точки. Обчислення значення *d* потребує одну операцію множення та 5 операцій піднесення до квадрату. Якщо *a* та *b* є малими, тоді *d* може бути обчислено за 4 операції піднесення до квадрату (ми вважаємо, що *a* 2 , *a*3 та

27*b*2 обчислюються попередньо та зберігаються):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *d*  *y* 4 |  3*ax* 4 |  6 *a* 2 *x* 2  18*by* 2 | | | | |  24 *abx*  *a* 3 |  27*b*2. |  |
| 1 | 1 | 1 | | | 1 | | 1 |  |  |
| Визначивши *D*  2 *y*1 *d* та *I*  *D*1 , ми отримаємо: | | | | | | | | |  |
|  |  |  | 1 |  *dI* , | 1 |  8 *y* 4*I*. | |  |  |
|  |  |  |  |  |  |  |
|  |  |  | 2 *y*1 | | 2 *y*3 | | 1 |  |  |
|  |  |  |  |  |  |

Таким чином, спосіб 3 полягає у наступному:

*t*1 *x*12;

*t* 23 *x*12 *a*;

*t*32 *y*12;

*t* 4 *t*32;

*t*5 *x*1 *t*32 *t*1 *t*4;

 *t* 23*t*5 *t* 222*t*4;

*t* 62 *y*1 *d* ;

*inv*  *t* 1;

6

**1 *d*  *inv* *t*2;

*x*3 **122*x*1;

*y*3 **1*x*1 *x*3 *y*1;

*t*73*x*32 *a*;

**2  2  *t*4  *inv* *t*7;

*x*4 **222 *x*3;

*y* 4 **2*x*3 *x*4 *y*3.

**2.4.** **Висновки до розділу 2**

даному розділі були розглянуті основи математичної теорії скінченних полів, над якими визначається еліптична крива, яка використовується в підходах та методах побудови криптосистем, описаних у розділі 1. Скінченні поля представляють особливий інтерес з огляду на ефективність їх застосування в апаратних та програмних реалізаціях криптосистем заснованих на еліптичній кривій, через свою очевидну близькість до апаратного (двійкового) подання значень та виконання операцій на обчислювальних приладах.

Проаналізовані скінченні поля такі як *GF* ( *p*) та *GF* (2 *m* ) , принципи їх побудови та операції над їх елементами. Продемонстровано спосіб знаходження оберненого елемента в мультиплікативній групі поля за

допомогою розширеного алгоритму Евкліда. Продемонстровано взаємозв’язок між теорією скінченних полів та операцій над точками, описаний особливий випадок операцій з точкою на нескінченності тощо.

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

**3. Програмний комлекс для реалызації класу еліптичних кривих.**

**3.1 Вибір мови програмування**

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

Іншим важливим моментом, що був врахований на стадії проектування програмного комплексу, є наявність очевидних моделей та абстракцій, що дозволяють ефективно моделювати предметну галузь. Такими абстракціями є скінченні поля, елементи скінченних полів, еліптичні криві тощо. Дані абстракції мають різну реалізацію, як наприклад скінченні поля *GF* ( *p*) та

*GF* (2 *m* ),де в одному випадку елементи можуть бути подані у виглядіцілочисельних значень, а у іншому – у вигляді многочленів, мають різну логіку виконання операцій над елементами, тощо, але мають ряд спільностей, як скажімо, однакові структури викликів операцій над елементами. Так і еліптичні криві з параметрами, визначеними над різними типами скінченних полів мають за своєю суттю абсолютно однаковий шаблон для викликів операцій над їх точками [26].

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

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

***C++***

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

1990-х роках С++ стала однією з найуживаніших мов програмування загального призначення. Мову C++ використовують для системного програмування, розробки програмного забезпечення, написання драйверів,

потужних серверних та клієнтських програм, а також для розробки розважальних програм, наприклад, відеоігор. С++ суттєво вплинула на інші популярні сьогодні мови програмування: С# та Java. При створенні С++ прагнули зберегти сумісність з мовою С.

Нововведеннями С++ порівняно з С є:

підтримка об'єктно-орієнтованого програмування через класи.

підтримка узагальненого програмування через шаблони.

доповнення до стандартної бібліотеки.

додаткові типи даних.

обробка винятків.

простори імен.

вбудовані функції.

перевантаження операторів.

перевантаження імен функцій.

посилання і оператори управління вільно розподіленою пам'яттю.

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

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

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

***Java***

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

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

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

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

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

***Python***

Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня з строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. В мові програмування Python підтримується такі

парадигми програмування, як: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Основними її перевагами є:

Чистий та інтуїтивно зрозумілий синтаксис.

Портабельність програм пов’язана з інтерпретативнісю.

Наявність вбудованих модулів,шо надають можливість використовувати колекції, особливі структури даних, математичний пакет тощо.

Можливість програмування на Python в скрипковому режимі (з консолі).

Зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини).

Політика відкритого коду (open source).

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

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

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

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

**3.1 Інші засоби необхідні для виконання поставленого завдання**.

**make** - це системна утиліта для UNIX-подібних операційних систем або самостійно встановлюване прорамне забезпечення для операційних систем сімейста Windows , що дозволяє болегшити компіляцію і зборку проектів.

Утиліта make є доволі старим інструментом і в неї є цілком популярні на сьогоднішній день кункуренти такі, як cmake , qmake та інші. В рамках цієї роботи використовується система зборки make, з тої причини, що навіть його значно більш зручні конкуренти є надбудовами над make-ом та зрештою використовують його для створення виконуваних чи об'єктих файлів приховуючи принципи функціонування мейкфайлів та забезпечуючи програміста більш зручним інтерфейсом.

Було обрано саме цю систему, тому що знання принципів функціонування мейкфайлів дозволяє набагато краще зрозуміти особливості роботи компіляторів та безмежні можливості з роботи із змінними середовища та shell-сриптами.

Утиліта приймає як аргумент так званий Makefile , у якому вказані інструкції щодо компіляції і зборки або ж конкретну команду із мейкфайла у поточній директорії. Мейкфайл використовує специфічний синтаксис, основою якого є цілі та залежності:

*target: dependencies*

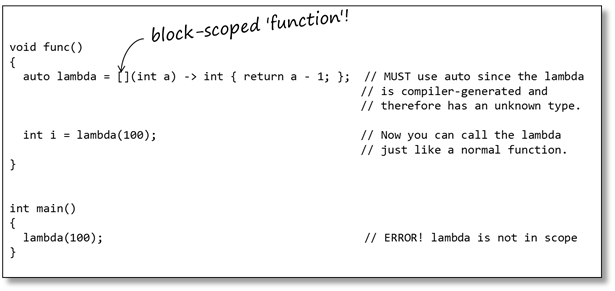
*system command(s)*

**С++11 standart**

Цей стандарт , як можна здогадатись, було прийнято у 2011 році і він сильно вдосконалив та покращив мову програмування С++.

Далі буде наведено невеликий список значних нововведень, що були прийняті інститутом стандартизації у 2011 році і в майбутньому визначили вектор розвитку мови, як такої.

* З'явилось ключове слово *auto* та *decltype,* що дозволяють на етапі компіляції визначати типи змінних та однозначно детермінуати тип значення, що повертає функція, зокрема лямбда-функція.
* Лямбда-функції, або ж анонімні функції, дозволяють у зручний спосіб створювати певний функціонал, без явного створення звичайної функції. Зазвичай використовується в місцях, де було доречно використовувати вказівники на функції у ранніх версіях мови.



* Розумні вказівники, що реалізовють концепцію RAII, в основі якої лежить простий, але дуже ефективний принцип: у конструкторі об'єкт захоплюється, у деструкторі звільняється. Це дозволяє реалізувати в мові програмування С++ певний аналог збирача мусора із мови програмування Java. Це осначає, що за умови, якшо вказівник вказує на область динамічно виділеної памяті і ми виходим за межі області видимості цього вказівника, динамічно виділена пам'ять автоматично звільняється, таким чином уникається протікання пам'яті. Стандарт надіє наступні види вказівників *unique\_ptr, shared\_ptr, weak\_ptr*.
* Range-based цикли - зручний механізм, що реалізує таку популярну для багатьох мов програмування функцію як foreach(). У С++11 це працюєнаступним чином:

for(int &arr\_elm: arr)

do\_something(arr\_elem);

, де arr\_elm є конкретним елементом масиву, а arr - це сам масив, по якому ми ітеруємось.

UML

git hub

Все програмне забезпечення, що розроблялось використовувало смстему контролю версій Git. Історію розробки та репозиторію можна переглянути за посиланням ...

**3.2 Клас для роботи із числами великої розрядності**.

Основним нашим пріоритетом при напсанні програмного забезпечення в рамках цієї роботи є його швидкодія та можливість застосовувати об'єктно-орієнтовану парадигму програмування для створення абстракцій з якомога меншим рівнем залежностей між ними для зручності та можливості розширення функціоналу класів в майбутньому . З огляду на ці судження для реалізації запланованих цілей в контексті написання коду було обрано мову програмування С++, оскільки вона є найшвидшою порівнянно із своїми вищезазначеними конкурентами та підтримує парадигму ООП.

Що стосується стандарту мови прорамування С++, то наш вибір - це стандарт 2011 року, у якому в мову програмування С++ було внесено значні зміни та сильно розширено її функціональні можливості. Зокрема з'явилась повноцінна підтримма багатопоточного та конкурентного програмування, що так само може бути використано в майбутньому для оптимізації роботи алгоритмів програми та поділ завдань між різними потоками. Так само з'явилась можливість явно вказувати компілятору про можливісь проведення обчислень на етапі компіляції за допомогою ключового слова constexpr та стало можливо проводити тестування результатів обчислення на етапі компіляції за допомогою функції static\_assert(); ,що є дуже корисним функціоналом для перевірки правильності роботи нашої програми.

Як зазаначалось вище по тексту використовуючи еліптичну криптографію для забеспечення захищеності інформації від третьої сторони, ми оперуємо із надзвичайно великими числами з розрядністю, що сягає до 4096 біт у числі.

Як відомо максимальна розрядність числа , яку підтримують вбудовані типи мови програмування С++ є 64 біти у певних версіях компілятора так самоє тип int128\_t , що підтримує до 128 біт у числі, але і цього нам не достатньо.

Таким чином першочергово постає питання створення класу для роботи із числами великої розрадності.

Для імплементації поставлених завдань було використано масив із беззнакових інтеджерів. В результаті кожен елемент масиву це один розряд нашого числа, таким чином маємо систему числення , в одному розряді якої можуть поміститись числа в діапазоні від 0 до 4 294 967 295. Такий масив хоча б із тисячою елементів цілком задовільняє потреби нашого завдання.

Наступним кроком є забезпечення інтерфейсу для взаємодії щойноствореного класу із іншими програними сутностями, такими як функції та класи. Для цього було перегружено основні оператори для роботи із звичайними числами такі, як +, -,\*,%. Оператор ділення не було перегружено оскільки в еліптичній криптографії не використовується дія ділення для точок еліптичної кривої, попри це реалізація цього функціоналу є досить не простим завданням і в умовах поставленої задачі є непотрібною.

Особливої уваги потребує перегрузка оператора множення. Для реалізації даної функції було використано алгоритм множення Карацуби, про який знадувалось вище. Цей алгоритм зменшує кількість ресурсозатратних операцій та дозволяє пришвидшити процес множення точки еліптичної кривої на число на 25%. Це є прекрасним результатом, зважаючи на те,що основною, найчастіше використовуваною опрацією на точками еліптичної кривої є саме множення.

Так само було розробелено інші функції, що забезпечують більш зручний інтерфейс із класом.

**3.3 Клас для роботи із еліптичними кривими.**

Тепер, коли в нас є розроблений клас для роботи із числами великої розрядность, ми маємо усе необхідне для написання класу для роботи з еліптичними кривими.

Проаналізувавши види еліптичних кривих, які використовуються в еліптичній криптографії, ми дійшли висновку, що краще починати реалізацію завдяння із створення інтерфейсу еліптичної кривої. Оскільки зазвичай використовується два види еліптичних кривих, а саме GF(p) та GF(2^m), але операції, які можна виконувати над цими кривими цілком подібні, розумно буде створити чисто віртуальний клас, у якому будуть тільки прототипи функцій та приватні змінні. Після цього ми зможемо створити два класи наслідника від даного інтерфейсу, в яких реалізуєм функціонал, що властивий для кожного із видів еліптичної кривої.

Фото ЮМЛ