Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(освітній рівень)

на тему: *"Програмна реалізація криптографічного протоколу розділення секрету"*

Виконав: студент (ка)
Спеціальності:
125 «Кібербезпека»
(шифр і назва напряму підготовки, спеціальності)
Кмиць В.Р.
підпис (прізвище та ініціали)

Керівник *Муж В.В.*
підпис (прізвище та ініціали)

Нормоконтроль
підпис (прізвище та ініціали)

Завідувач кафедри *Загородна Н.В.*
підпис (прізвище та ініціали)

Рецензент
підпис (прізвище та ініціали)

м. Тернопіль – 2021

АНОТАЦІЯ

Програмна реалізація криптографічного протоколу розділення секрету // Кваліфікаційна робота ОР «Бакалавр» //Кмиць Володимир Романович// Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБс-42 // Тернопіль, 2021 // С. 49 , рис. – 4, табл. – 4 , кресл. – , додат. – 6.

Ключові слова: КРИПТОГРАФІЧНИЙ ПРОТОКОЛ, РОЗДІЛЕННЯ СЕКРЕТУ, ПОЛІНОМ ЛАГРАНЖА, КИТАЙСЬКА ТЕОРЕМА ПРО ОСТАЧІ, СХЕМА АСМУТА БЛЮМА.

Кваліфікаційна робота присвячена розробці програмного забезпечення для криптографічного протоколу розподілу секрету. В роботі обґрунтовано вибір програмного середовища розробки та вибір методів порогових схем розділення секрету. Протестовано програмне забезпечення, в якому реалізовано дві схеми розділення секрету: Шаміра та Асмута-Блюма. Показано роботу програмного забезпечення при коректному використанні параметрів схеми та можливість відновлення секрету за частками. Також показано, що при недостатній кількості часток секрету, таємна інформація не буде відновлена.

Дану розробку можна використовувати для зберігання особливо важливої інформації в державницьких органах, або ж для надійного зберігання ключа шифрування.

В першому розділі описано сутність криптографічних протоколів та алгоритми відновлення секрету. В другому розділі обґрунтовано вибір програмного середовища та описано реалізацію розробки. В третьому розділі висвітлено результати тестування програмної розробки.

ANNOTATION

Software implementation of secret sharing cryptographic protocol // Thesis of educational level "Bachelor" // Kmyts Volodymyr Romanovych // Ternopil National Technical University named after Ivan Pulyuy, Faculty of Computer Information Systems and software engineering, Department of Cybersecurity, СБс-42 group // Ternopil, 2021 // P. 49 , fig. -4, table. - 1 , chair. - , added. -6.

Keywords: CRYPTOGRAPHIC PROTOCOL, SECRET SHARING, LAGRANGE POLYNOM, CHINESE REMAINING THEOREM, ASMUTH-BLOOM S SCHEME.

Qualification thesis is devoted to the development of software for cryptographic protocol of secret sharing. The choice of software development environment and the choice of methods of threshold schemes for sharing secrets are substantiated in the paper. The software was tested, in which two schemes of secret sharing were implemented: Shamir and Asmuth-Bloom. The work of the software with the correct use of the parameters of the scheme and the possibility of restoring the secret by shares is shown. It is also shown that if there are not enough shares of the secret, the secret information will not be restored.

This development can be used to store particularly important information in government organizations, or to securely store the encryption key.

The first section describes the essence of cryptographic protocols and algorithms for the secret sharing. The second section substantiates the choice of software environment and describes the implementation of development. The third section highlights the results of software development testing.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API	Application Programming Interface
IDE	Integrated Development Environment
GUI	Graphical User Interface
https	Hypertext Transfer Protocol Secure
SSS	Secret Sharing Scheme
ZKP	Zero Knowledge Password Proof
ІТ	Інформаційні технології
КП	Криптографічний протокол
НСК	Найменше спільне кратне
ООП	Об’єктно-орієнтоване програмування

ВСТУП

Стрімкі темпи цифрової трансформації суспільства, освіти, бізнесу, урядових структур та впровадження сучасних інформаційних технологій в інших сфери життя людини з одного боку дозволяють автоматизувати багато процесів, підвищити продуктивність роботи людини, створити нові можливості для розвитку людини, підприємства держави. З іншої сторони такий перехід у віртуальний світ створює додаткові ризики у сфері захисту інформації, персональних даних та особливо важливих даних. Серед різних методів захисту інформації (технічних, правових, організаційних та інших) почесне місце займають криптографічні методи. Криптографія - це математичний інструмент, який відіграє життєво важливу роль в мережевій безпеці. Криптографія використовується для забезпечення конфіденційності та цілісності даних, а також забезпечує автентифікацію користувачів та невідмовність. Часто криптографію згадую в аспекті захищеного зберігання даних. Проте, попри уявлення більшості сфера застосування криптографії є значно ширшою. Криптографічні методи захисту інформації застосовуються також для розв'язання специфічних задач з кількома учасниками. Такі спеціальні алгоритми називають криптографічними протоколами. До таких протоколів відносять: протокол розподілу таємниці, протокол підкидання чесної монети, протокол сліпого підпису, доведення з нульовим розголошенням, протоколи анонімної електронної готівки, протоколи підписання контракту, протокол голосування та інші. Часто протоколи комбінують зі стандартними схемами шифрування для досягнення тих чи інших цілей. Для прикладу стандартна задача шифрування даних вимагає зберігання і захисту секретних ключів учасників. Проте можна собі уявити ситуацію, коли важливу секретну інформацію зашифрували з допомогою секретного ключа і з певних причин ключ був втрачений. Тоді назавжди втраченою виявиться і уся важлива інформація. Таким чином, потрібні безпечні та ефективні механізми управління ключами. Однією з них є схема розподілу секрету (SSS - Secret Sharing Scheme), яка дозволяє розділити секретний ключ, що використовується в схемі шифрування на кілька частин і розподілити їх серед вибраних сторін. Секрет

можна відновити, коли ці сторони певним чином співпрацюють. Існують схеми, у яких для відновлення секретного ключа необхідно участь всіх сторін. Існують схеми, в яких для відновлення паролю необхідна лише частина учасників. У цій роботі буде розглянуто деякі алгоритми схем розділення секрету, що варіюються від тривіальних схем до порогових.

Метою даної роботи є розробка програмного забезпечення для реалізації криптографічного протоколу розділення секрету, яке може бути використане для зберігання важливої інформації між кількома учасниками, в тому числі і ключа шифрування.

Для досягнення поставленої мети потрібно розв'язати наступні завдання:

- Провести огляд літературних джерел в предметній області.
- Проаналізувати основні відомі алгоритми протоколу розділення секрету.
- Обрати середовище розробки.
- Розробити програмне забезпечення.
- Протестувати розроблене програмне рішення.

1 ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1.1 Криптографічні протоколи

Під протоколом загалом розуміють розподілений алгоритм (порядок дій) між декількома учасниками, призначений для розв'язання певної задачі [1]. Порядок дій означає, що існує певна послідовність дій, яка повинна бути дотримано при виконанні протоколу. Компонентами протоколу вважаються його учасники, алгоритми, що використовуються учасниками чи формулювання задачі, яку призначений розв'язувати цей протокол. Учасниками протоколу можуть виступати:

- люди;
- комп'ютерні вузли;
- сервери;
- комп'ютерні програми;
- канали зв'язку;
- бази даних і т.п.

Якщо призначення протоколу, пов'язане з забезпеченням конфіденційності, цілісності, невідмовності чи іншої задачі криптографії, то такий протокол називають криптографічним. Все ж потрібно розуміти, що протокол можна назвати криптографічним (КП) лише у випадку, якщо в його основі лежить певна криптографічна задача. Якщо ж протокол (наприклад https) використовує криптографічні елементи, які є допоміжними, то його в такому випадку не називають криптографічним. Безумовно, криптографічний протокол теж включає в себе відомі стандартні криптографічні примітиви, але за звичай призначення такого протоколу виходить за рамки простої інформаційної безпеки. Існують базові криптографічні протоколи, такі як: обмін ключами, аутентифікація особи, управління ключами, їх поєднання та різновиди. Проте, існує ряд протоколів, невідомі широкому загалу, які, проте, мають важливе значення. Так, для прикладу учасники протоколу можуть розділити секретну інформацію між собою, щоб

застрахуватись від її втрати або розкриття одним ненадійним учасником. Учасники також можуть підписати контракт в один і той самий час або ж згенерувати спільний випадковий біт, перебуваючи на відстані. Завдання криптографії в протоколі – запобігання і попередження шахрайства.

Відомими спеціальними криптографічними протоколами є [3]:

- Протокол розподілу секрету.
- Протокол підкидання чесної монети.
- Доведення з нульовим знанням.
- Мітки часу.
- Підписання контракту.
- Сліпі підписи.
- Електронна готівка
- Протокол голосування та інші.

Розглянемо детальніше частину з цих протоколів[5].

1.1.1 Протокол розподілу секрету

Розподіл секрету стосується криптографічних методів розбиття секрету на декілька частин та розподілу цих часток між кількома сторонами, так що лише тоді, коли сторони зберуть свої відповідні частки, секрет може бути відновлений. Більш конкретно, власник секрету, який іноді називають дилером, створює n часток секрету і визначає поріг m для кількості часток, необхідних для відновлення секрету. Потім дилер розподіляє частки, щоб вони контролювалися різними сторонами [6].

У схемах безпечного обміну таємницею зловмисник, який отримує доступ до меншої кількості поділ таємниці, ніж визначено пороговим значенням, не отримує інформації про таємницю.

Схеми секретного обміну корисні, оскільки вони дозволяють забезпечити більш безпечне зберігання високочутливих даних, включаючи ключі шифрування, коди запуску ракети та банківські рахунки. Відстуня єдина точка вразливості, яка може призвести до втрати даних.

Схеми спільного використання секретів важливі в середовищах хмарних обчислень, оскільки вони забезпечують метод для забезпечення високого рівня безпеки секретної інформації за допомогою алгоритмів, реалізованих у програмному забезпеченні (без необхідності спеціалізованого обладнання).

1.1.2 Протокол підкидання чесної монети.

Протокол підкидання чесної монети по телефону дозволяє сторонам, які не взаємно не довіряють одна одній, генерувати загальний неупереджений випадковий біт. Причому така взаємодія відбувається по каналу зв'язку ("по телефону") і передбачає момент, що два учасники не бачать одне одного. Відбувається протокол приблизно таким чином: абонент А підкидає монету, а інший абонент вгадує результат. Проте, при такому спрощеному підході першому абоненту дуже легко вийти на бажаний для нього результат. Тому спрощено цей протокол можна продемонструвати наступним чином. Абонент А підкидає самостійно монетку і записує на листочку біт, що відповідає результату підкиданн (наприклад, 0 – це решта, 1 – це орел). Потім запаковує цей листочок в конверт і надсилає його другому учаснику протоколу звичайною поштою. Одразу після абонент А телефонує абоненту Б (поки лист ще не дійшов) і просить назвати свій біт, після чого називає свій. В такому випадку, перший абонент не може схитрувати, бо вже зафіксувала свій біт у відправленому листі. Результатом підкидання монети є XOR двох бітів учасників. Такий протокол повинен задовольняти двом властивостям. По-перше, коли всі сторони чесні і дотримуються інструкцій протоколу, на виході протоколу отримаємо рівномірно розподілений біт. По-друге, навіть якщо деякі сторони змовляються та відхиляються від інструкцій протоколу, вони не повинні мати можливості суттєво вплинути на загальний результат чесних учасників.

1.1.3 Доведення з нульовим знанням (Zero Knowledge Password Proof - ZKP)

ZKP дозволяє вам довести, що ви знаєте якусь таємницю (або багато таємниць) іншому абоненту, фактично не розкриваючи її. Сам термін "нульові

знання" походить від того, що не розголошується жодної ("нульової") інформації про таємницю , але друга сторона (яка називається "Верифікатор") переконана, що перша сторона ("Довіритель") знає секретну відповідь на запитання. Цей протокол виглядає дещо дитячим і беззмістовним. Навіщо нам потрібно доводити, що знаємо таємницю, не відкриваючи її? Як виглядає процес доведення знання, коли ви не довіряєте іншій людині, але все одно потрібно переконати її, що ви знаєте таємницю.

Проте цей протокол має доволі прикладне застосування, для прикладу протокол доведення з нульовим знанням може бути використаний як спосіб автентифікації, коли користувачі не обмінюються паролями, а значить, їх не можна вкрасти. Це круто, оскільки такий спосіб робить ваше спілкування настільки захищеним і захищеним, що ніхто інший не може дізнатися, про що ви спілкуєтесь або якими файлами ви ділитесь один з одним.

Крім цього, можна навести ще такий приклад застосування цього протоколу. Нехай існує програмний застосунок, який перевіряє, чи достатньо грошей на вашому банківському рахунку, щоб здійснити транзакцію, при цьому не відкриваючи інформацію про баланс рахунку. Або додаток, що підтверджує дійсність пароля, без необхідності його безпосередньої обробки. Таким чином, підтвердження нульових знань може допомогти проводити всілякі конфіденційні угоди, транзакції та взаємодії більш приватним та безпечним способом.

1.1.4 Протокол сліпого підпису

Схема сліпого підпису - це тип цифрового підпису, що приховує вміст повідомлення та відправника. У цих схемах повідомлення відправника приховується (або засліплюється) до того, як одержувач підпише його. Схеми сліпого підпису корисні в програмах, де інформація про відправника є важливою особливістю спілкування. Коротко алгоритм сліпого підпису можна описати наступним чином: Користувач А засліплює повідомлення з використанням деякого випадкового числа. Користувач Б підписує це засліплене повідомлення своїм секретним підписом і надсилаю користувачу А назад. Користувач А виконує

перетворення над підписаним документом з використанням того ж випадкового числа і отримує своє повідомлення, підписане користувачем Б.

Електронна система голосування є реальним прикладом застосування схеми сліпого підпису. У цьому випадку відправник (виборець) та підписант (одержувач, виборчий орган) не пов’язані між собою, а особиста конфіденційність відправника та переваги голосування є першорядними. Підпис вважається достатньо чинним, щоб гарантувати, що голосування буде записано з довірою, і виборець залишається анонімним.

В подальшому, якщо виборчий орган буде покликаний перевірити отриману інформацію, він зможе перевірити достовірність повідомлення, але не зможе зв’язати його з відправником (що називається неприв’язаністю).

1.1.5 Схема міток часу

В багатьох випадках виникає необхідність довести, що певний документ вже існував у визначений момент часу. Для прикладу, особливо гостро це питання стоїть при доведенні авторських прав. Паперові документи, завірені нотаріусами містять дату завірення, яка і є доказом того, що даний документ існував у відповідний момент. З електронними документами все набагато складніше. Їх можна безкінечно копіювати, модифікувати, не залишаючи слідів. Якщо з методи забезпечення цілісності документа є більш-менш відомими, то з датою все складніше. Дату створення документу не так то важко і поміняти. Тому протокол мітки часу повинен володіти наступними властивостями:

- мітка часу повинна існувати сама по собі і не залежати від середовища, де вона зберігається;
- мітка часу також повинна забезпечувати цілісність документу (неможливість модифікації)
- повинно бути неможливим поміняти певним чином мітку часу, або ж задати мітку відмінну від поточного часу (1 січня створити документ з датою 31 січня, або ж навпаки).

1.2 Протокол розподілу секрету, його застосування

Останнім часом в криптографії інтенсивно розвиваються специфічні протоколи, які отримали назву схем (протоколів) розподілу таємниці. Спочатку елементи розподілу секрету застосовувалися для створення резервних копій ключів та забезпечення криптографічної стійкості систем. Схеми розподілу секрету знайшли також застосування і для сумісного управління критичними технологіями та процесами (для зниження ризиків компрометації деяких критичних даних).

Ідея розподілу таємниці полягає в тому, що загальна таємниця ділиться на n-частин, які називають долями (тінями, частками) таємниці. Для відновлення секрету потрібно зібрати всі або певну кількість часток. В першому випадку говорять про розбиття (англ. splitting), а в другому про розподілення (англ. sharing) секрету (спільне володіння). Іншою відмінністю протоколів розбиття від протоколів розподілення є розподіл часток. В першому випадку долі передаються різним людям, а при розподіленні таємниці – одна людина може володіти декількома частками секрету.

Розроблені на сьогодні протоколи спільного володіння таємницею можна класифікувати наступним чином:

- Порогові протоколи розподілу таємниці (граничні схеми)
- Спільне використання секрету з шахраями (Спільне використання секрету з можливістю виявлення фальшивих часток, Розподіл таємниці з недовірою)
- Спільне використання секрету без власника секрету
- Спільне використання секрету без розкриття часток при встановленні секрету
- Спільне використання секрету з можливістю перевірки окремих часток
- Спільне використання секрету з можливістю блокування окремих часток

Однією з найпростіших відомих схем розбиття таємниці є використання схеми так званого гамування, яка працює як одноразовий шифр-блокнот.

1.2.1 Схема гамування

Просту і в той же час ефективну схему розбиття секрету можна побудувати на базі додавання по модулю 2. Нехай задано деякий секрет S, закодований у двійковому вигляді. Для його розбиття власник генерує кілька бітових рядків (гам) $G_1, G_2, \dots, G_{n-1}$, які окремо передає кожному з учасників протоколу. Крім цього власник секрету (Трент) виконує операцію побітового додавання за модулем 2 секрету з усіма гамами:

$$C = S \oplus G_1 \oplus G_2 \oplus \dots \oplus G_{n-1} \quad (1.1)$$

Результат шифрування (шифрограму) *C* власник викладає в доступне для учасників місце, або ж віддає останньому учаснику розбиття. Для відновлення секрету необхідно скласти шифрограму з усіма виданими гаммами, при чому не важливо, в якому порядку.

На відміну від розбиття секрету учаснику (суб'єкту) може передаватися відразу кілька рівних часток і для відновлення секрету необов'язково мати всі частки. Така схема, де секрет ділиться на n часток, а для його відновлення необхідно зібрати не менше ніж m часток, де m <n, називається (m, n) -пороговою схемою. Розглянемо дві найбільш відомі порогові схеми

1.2.2 Схема інтерполяційних многочленів Лагранжа

Для створення схеми Аді Шамір в 1979р.використав рівняння для многочленів в полі лишків.

Загалом суть інтерполяційних поліномів Лагранжа полягає в наступному: якщо деяка функція *f(x)* задана з допомогою m точок (x_i, y_i), $i = \overline{0, m-1}$, то її можна інтерполювати з допомогою полінома ступеня ступеня *m-1*, який буде проходити через всі точки і максимально близько описувати задану функцію. Всім відомо, що пряму можна задати двома точками, отже поліном буде третього степеня. Квадратичну функцію другого степеня можна задати трьома точками і т.д.

Інтерполяційний поліном задається формулою:

$$f(x) \approx L(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_2x^2 + a_1x^1 + a_0 = \sum_{i=0}^{m-1} y_i l_i(x), \quad (1.2)$$

де a_i – коефіцієнти інтерполяційного поліному

y_i – значення вихідної функції в *i*-й точці

$l_i(x)$ – базисні поліноми, що визначаються за формулою:

$$l_i(x) = \prod_{j=0, j\neq j}^{m-1} \frac{x - x_j}{x_i - x_j} \quad (1.3)$$

Для розділення секрету *S*, що може бути відновлений на основі *m* часток на основі інтерполяційних поліномів Лагранжа необхідно взяти поліном *m-1* степені за модулем *p*:

$$L(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_2x^2 + a_1x^1 + a_0$$

Число p повинно бути достатньо великим простим числом (більшим за максимально можливий секрет, коефіцієнти поліному та загальну кількість можливих часток *n*). Коефіцієнти многочлену обираються довільно випадковим чином, за винятком $a_0 = S$ і зберігаються в таємниці. Число p повинно бути опубліковане.

Власник секрету обчислює значення поліномів $y_i = L(x_i), i = \overline{0, n-1}$ і передає значення точок (x_i, y_i) відповідно до визначеної кількості часток для кожного з учасників протоколу. Для встановлення секрету необхідно зібрати *m* часток і знайти коефіцієнти інтерполяційного поліному, включаючи *S*. Для цього необхідно розв'язати систему з *m* лінійних рівнянь з *m* невідомими. Складність розв'язку такої системи полягає в тому, що її необхідно розв'язати в полі лишків Z_p. Для цього необхідно для кожної з m часток (точок) обчислити базисний поліном за формулою (1.3). Далі на основні визначених m базисних поліномів визначити інтерполяційний поліном Лагранжа за формулою (1.2), знайти та привести значення всіх коефіцієнтів полінома до значень поля Z_p. Як відомо, значення секрету і є одним з коефіцієнтів.

1.2.3 Спільне використання секрету за схемою Асмута-Блума (Asmuth-Bloom)

В (m,n)-пороговій схемі Асмута-Блума для розподілу часток використовуються прості і взаємно прості числа, а для відновлення китайська теорема про остачі.

Для *створення часток секрету* необхідно:

1. Вибрати велике просте число p (більше за секрет S).
2. Вибрати n взаємо-простих чисел $d_i, = \overline{1,n}$, таких що
   a. $d_i > p$
   b. $d_i < d_{i+1}$ (числа впорядковані за зростанням)
   c. $d_1 \cdot d_2 \cdot \ldots \cdot d_m < p \cdot d_{n-m+2} \cdot d_{n-m+3} \cdot \ldots \cdot d_n$
3. Вибір випадкового числа r, що задовольняє умові $r < \frac{\prod_{i=1}^{m} d_i - S}{p}$.
4. Обчислити $S' = S + rp$
5. Визначити частки (d_i, k_i), де $k_i = S' \, mod \, d_i$
6. Опублікувати p та розподілити частки (d_i, k_i) між учасниками.

Схема Асмута-Блюма використовує для відновлення секрету китайську теорему про остачі. Авторство формулювання та доведення теореми про остачі приписують китайському математику першого століття Сун Це. Це один з елементарних результатів теорії чисел. Згідно з цією теоремою, будь-яке невід'ємне число, яке не перевищує добутку модулів можна однозначно відновити, якщо відомі його лишки за цими модулями.

Теорема 1. Нехай числа M_s и M'_s визначені з умов $m_1 m_2 \cdots m_k =$ $= M_S m_S$, при чому, $M_s M'_s \equiv 1 (\text{mod } m_S)$, s = 1,2,...,k і цілі числа $m_1, m_2, \ldots, m_k$ – попарно взаємо-прості, тобто. $(m_i, m_j) = 1, \; i, j = 1, 2, \ldots, k$.

Нехай $b_1, b_2, \ldots, b_k$ – теж цілі , такі, що $0 \le b_i \le m_i, \; i = 1, 2, \ldots, k$.

Тоді система рівнянь

$$x \equiv b_1 (\text{mod } m_1), \qquad (1.4)$$

$$x \equiv b_2 (\text{mod } m_2),$$

. . . . . .

$$x \equiv b_k (\mathrm{mod}\ m_k)$$

має єдиний розв'язок в інтервалі $[0,\ M-1]$, де $\mathrm{M} = \mathrm{m}_1 \cdot \mathrm{m}_2 \cdot \cdots \cdot \mathrm{m}_k$

$$x_0 = M_1 M_1' b_1 + M_2 M_2' b_2 + \ldots + M_k M_k' b_k (\mathrm{mod}\, M). \quad (1.5)$$

Для *відновлення секрету* учасники протоколу повинні:

1. Зібрати m часток (d_j, k_j).

2. Визначити добуток D взаємно простих чисел d_j.

3. Обчислити коефіцієнти $D_j = \frac{D}{d_j}$.

4. Обчислити обернені елементи D_j^{-1} до D_j за модулем d_j (за розширеним алгоритмом Евкліда наприклад).

5. Обчислити $S' = \sum_j k_j D_j D_j^{-1} \ \ mod\ D$

6. Визначити секрет, як $S = S'\ mod\ p$

1.3 Постановка задачі

Отже необхідно розробити програмне забезпечення, в якому реалізувати схему Шаміра(інтерполяційних многочленів Лагранжа) та схема Асмута-Блюма.

Програмна реалізація схеми Шаміра повинна складатись з двох модулів, які повинні задовольняти наступним вимогам

1. Перший модуль:

   i. На вхід першого модуля подаються значення m, n, S, p. Необхідно виконати перевірку коректності введеного значення p.

   ii. Значення коефіцієнту a_0 присвоїти S. Інші коефіцієнти інтерполяційного поліному Лагранжа $a_i, i = \overline{1, m-1}$ слід вибрати випадково.

   iii. Випадковим чином сформувати n значень x_i в полі Z_p та обчислити відповідні їм значення y_i.

iv. На виході першого модуля необхідно сформувати n текстових файлів з відповідними значеннями часток (x_i, y_i)

2. Другий модуль

i. На вхід другого модуля подається значення m – кількості часток для відновлення секрету та p.

ii. Користувач повинен зробити вибір m файлів з частками.

iii. Програма повинна відновити початковий секрет S.

Протокол розподілення секрету за схемою Асмута-Блюма повинен бути реалізований з врахуванням наступних вимог. Програмна реалізація повинна складатись з двох модулів.

1. Перший модуль

i. На вхід першого модуля подаються значення m, n, S, p та n взаємо-простих чисел $d_i, = \overline{1, n}$. Необхідно виконати перевірку коректності введених значень чисел p та d_i.

ii. Випадковим чином здійснити вибір параметра r та здійснити перевірку коректності вибраного значення.

iii. Обчислити значення S'.

iv. За відомими значеннями d_i обчислити k_i.

v. На виході першого модуля необхідно сформувати n текстових файлів з відповідними значеннями часток (d_i, k_i)

2. Другий модуль

v. На вхід другого модуля подається значення m – кількості часток для відновлення секрету.

vi. Користувач повинен зробити вибір m файлів з частками.

vii. Програма повинна відновити початковий секрет S

2 ПРОГРАМНА РЕАЛІЗАЦІЯ КРИПТОПРОТОКОЛУ РОЗПОДІЛУ СЕКРЕТУ

Розробка будь-якого програмного забезпечення вимагає насамперед вибору і обґрунтування середовища розробки

2.1 Вибір середовища розробки

Об’єктно-орієнтоване програмування (ООП) - одна з парадигм програмування, яка розглядає програму як множину “об’єктів”, що взаємодіють між собою. ООП є найпопулярнішою парадигмою програмування і подається як стандартний спосіб програмування для більшості програмістів. Об'єктно-орієнтоване програмування (ООП) - це парадигма програмування, яка спирається на концепцію класів та об'єктів. Застосовується для структурування програмного забезпечення у прості шматки (зазвичай їх називають класами), які використовуються для створення окремих екземплярів об’єктів. Існує багато об’єктно-орієнтованих мов програмування, включаючи, C++, C#, Java та Python.

C # - це відповідь Microsoft на C ++ та JAVA. В міру того, як парадигма мови програмування перейшла у об'єкто-орієнтовану розробку, Microsoft відстала від VB/VBA мов програмування і їм потрібно було повернутися до гри. Отже, вони зробили те, що їм вдається найкраще ... взяли найкращі чужі ідеї C ++ та Java і поєднали їх у спосіб Microsoft ... ergo C#. Це чиста об'єкто-орінтована мова програмування, як Java та C ++, і вона використовує найкраще інтегроване середовище розробки (integrated development environment - IDE) Visual Studio. Ці продукти дають змогу розробляти як консольні програми, так і програми з графічним інтерфейсом. Microsoft і Visual Studio пропонують будь-яку з декількох мов програмування в Visual Studio, в яких можна розробляти додатки для настільних комп'ютерів Windows, веб-додатки, вікна та веб-сервіси, javascript / jquery, програми MVC, навіть мобільні застосунки, і все це в межах Visual Studio..

Переваги ООП:

- ООП моделює складні речі як відтворювані, прості структури.

- Об'єкти ООП можна використовувати багаторазово в усіх програмах.
- Дозволяє специфічну для класу поведінку за допомогою поліморфізму.
- Простіше налагоджувати, класи часто містять всю відповідну для них інформацію.
- Захищений підхід, захищає інформацію шляхом інкапсуляції.

2.1.1 Мова програмування C#

Мова програмування C # (C sharp) - це дуже проста мова для вивчення. Вона повністю заснована на мовах C та C ++. Синтаксис мови C# є дуже виразним, але він також простий і легкий у вивченні. Кожен, хто знайомий з C, C ++ або Java може миттєво розпізнати синтаксис фігурних дужок C #. Використання C sharp спрощує багато складностей C++ і забезпечує потужні функції, такі як нульові типи значень, делегати, перелічення, лямбда-вирази та інші, яких немає в Java.

У січні 1999 року Андерс Хейлсберг створив групу для розробки нових мов програмування, на той час цю нову мову назвали крутою. Але через маркетингов причини Андерс Хейлсберг змінив цю мовну назву Cool на C #.

Хейлсберг є головним дизайнером C # у компанії Microsoft, також він створив різні мови, такі як Turbo Pascal, Embarcadero Delphi та Visual J ++. Тоді він сказав в одному з інтерв'ю, що мови C ++ та Java Pascal не мають основ загальної роботи, тому ми створили мову C #.

В таблиці 2.1. наведено історію розвитку мови C# [7]

Таблиця 2.1 – Історія версій C#

№ за/п	Версія C#	Microsoft Visual Studio	.Net Framework	Особливості
1	C# версія 1.0	Microsoft Visual Studio 2002	MS.Net Framework 1.0	Перший реліз
2	C# версія 2.0	Microsoft Visual Studio 2005	MS.Net Framework 2.0	- перетворення груп методів (делегати) - підтримка типів null-able - ітератори

Продовження таблиці 2.1

3	C# версія 3.0	Microsoft Visual Studio 2008	MS.Net Framework 3.0-3.5	- автоматично реалізовані властивості - анонімні типи - вирази запитів - лямбда-вирази - дерева виразів - методи розширення - неявно типізовані локальні змінні - спільні методи - ініціалізатор об'єктів і колекцій
4	C# версія 4.0	Microsoft Visual Studio 2010	MS.Net Framework 4.0	- динамічна прив'язка - іменовані / додаткові аргументи - універсальна коваріантність і контраваріантних - впроваджені типи взаємодії
5	C# версія 5.0	Microsoft Visual Studio 2012	MS.Net Framework 4.5	-асинхронні члени -інформаційні атрибути об'єкта, що викликається
6	C# версія 6.0	Microsoft Visual Studio 2015	MS.Net Framework 4.6	- статичні імпорти - фільтри винятків - ініціалізатор автовластивостей - елементи, що втілюють вираз - Null-розповсюджувач - інтерполяція рядків - оператор nameof

Закінчення таблиці 2.1

7	C# версія 7.0	Microsoft Visual Studio 2017	Net.Core	- змінні Out - кортежі і деконструкція - зіставлення шаблонів - локальні функції - розширені елементи, що втілюють вираз - Локальні змінні і значення Ref
8	C# версія 8.0	Microsoft Visual Studio 2019	MS.Net.Core 3.0	- члени тільки для читання - методи інтерфейсу за замовчуванням - покращення зіставлення шаблонів - оголошення using - статичні локальні функції - довідкові типи, що допускають значення NULL -асинхронні потоки. - індекси і діапазони - присвоєння об'єднання зі значенням NULL

Основними перевагами C sharp є:

- Швидкодія. Існує багато важливих особливостей мови C #, які роблять її мовою C # дуже швидкою, її компіляція та час виконання дуже швидкі
- Простота. C # - це проста мова. Це дає структурований підхід до розбиття проблеми на частини. Крім того, він має багатий набір функцій бібліотеки та типів даних. Код мови C # не вимагає файлів заголовків. Її код пишеться вбудовано.
- Об’єктно-орієнтованість. Мова C # - це об'єктно-орієнтована мова програмування, що полегшує розробку та обслуговування, порівняно з мовою

програмування, орієнтованою на процедури. Однак надто складно управляти, якщо код зростає із збільшенням розміру проекту. Крім того, програмування на C # підтримує інкапсуляцію даних, успадкування, поліморфізм, інтерфейси.

- Сучасна мова програмування. Мова C # є однією з сучасних мов програмування, оскільки вона базується на сучасній тенденції. Однак вона дуже проста, потужна для створення масштабованих, сумісних та надійних додатків.

- Безпека типів. Мова C # - це безпечний для типів код, який може отримати доступ лише до місця пам'яті та має дозвіл на виконання. Отже, це покращує безпеку програми. Мовою C # ви не можете виконувати небезпечні трансляції, такі як перетворення double у логічну форму. Його типи значень (примітивні типи) ініціалізуються нулями, а посилальні типи (об'єкти та класи) компілятором автоматично ініціалізуються до нуля.

- Сумісність. Сумісність - це процес, який дозволяє програмам C # робити майже все, що може робити власний додаток C ++. Коротше кажучи, сумісність мови - це здатність коду взаємодіяти з кодом, який написаний за допомогою іншої мови програмування. Це може допомогти максимізувати повторне використання коду і, отже, підвищити ефективність процесу розробки.

- Масштабованість та оновлюваність. Мова C # - це комп’ютеризована, масштабована та оновлювана мова програмування. Однак одна важлива річ полягає в тому, щоб оновити ваш .Net framework. вам доведеться вбити ваші старі файли та оновити їх новими

- Мова структурованого програмування. Для вирішення великих проблем програмування C # розділяє проблему на менші модулі, які називаються функціями або процедурами, кожна з яких несе певну відповідальність, тому мова C # називається структурованою мовою програмування.

- Багата бібліотека. Мова C # багата на бібліотеку. Таким чином, вона забезпечує безліч вбудованих функцій, які пришвидшують розробку.

- Орієнтованість на компоненти. Мова C # є мовою програмування, орієнтованою на компоненти, і підтримує компонентно-орієнтоване програмування через концепції методів, властивостей, подій та атрибутів (або

метаданих), дозволяючи самостійні та самоописуючі компоненти функціональних можливостей, які називаються ансамблями.

2.1.2 Visual studio

Visual Studio, створений Microsoft, є інтегрованим середовищем розробки (IDE), яке використовується для розробки комп'ютерних програм для Windows.

Інтегроване середовище розробки (IDE) - це програмне забезпечення для побудови програм, що поєднує загальні інструменти розробника в єдиний графічний інтерфейс користувача (GUI). IDE зазвичай складається з:

- Редактор вихідного коду: текстовий редактор, який може допомогти у написанні програмного коду з такими функціями, як підсвічування синтаксису візуальними сигналами, забезпечення автоматичного заповнення мови та перевірка на наявність помилок під час написання коду.
- Локальна автоматизація побудови: Службові програми, що автоматизують прості, повторювані завдання в рамках створення локальної побудови програмного забезпечення для використання розробником, як-от компіляція вихідного коду комп’ютера у двійковий код, упаковка двійкового коду та запуск автоматизованих тестів.
- Налагоджувач: програма для тестування інших програм, яка може графічно відображати розташування помилки в оригінальному коді.

Visual studio також може використовуватися для розробки веб-сайтів, веб-додатків та веб-сервісів. IT використовує платформи розробки програмного забезпечення Microsoft, такі як Windows API, Windows Forms, Windows Store, Windows Presentation Foundation та Microsoft Silverlight,

Включаючи редактор коду, який підтримує IntelliSense, Visual Studio написаний на C ++ та C # та пропонує інтегрований дебагер, який працює як дебагер вихідного рівня, так і дебагер машинного рівня.

2.2 Написання програмного забезпечення

2.2.1 Реалізація схеми многочленів Лагранжа

Згідно вимог до програмного забезпечення значення секрету *S*, загальної кількості часток *n*, кількості часток для відновлення секрету *m* та параметр *p*, . Необхідно виконати перевірку коректності введеного значення *p*. А значення коефіцієнтів поліному Лагранжа слід було вибрати самостійно випадковим чином. У лістингу 3.1 наведено код для випадкового вибору коефіцієнтів інтерполяційного поліному Лагранжа $a_i, i = \overline{1, m-1}$

Лістинг 3.1 – Вибір коефіцієнтів інтерполяційного поліному Лагранжа $a_i, i = \overline{1, m-1}$

```
for (int i = 0; i < m; i++) //вибір випадкових значень A
            {
                a.Add(rand.Next(0, 100));
                rtb_result.Text += i == 0 ? "" : " A" + i + "=" +
a[i] + ";";
            }
```

Наступною вимогою було випадковим чином сформувати *n* значень x_i в полі Z_p та обчислити відповідні їм значення y_i. В лістингу 3.2 наведено код виконання програми, що реалізовує цю вимогу. Правда, необхідно зазначити, що значення x вибираються невипадково, а від 1 до n, проте такий вибір x-ів не впливає на стійкість самого алгоритму

Лістинг 3.2 – Код для обчислення n часток інтерполяційного поліному Лагранжа для значень *x*-ів від 1 до n

```
for (int x= n; x>0; x--)//генеруємо x який дорівнює послідовному
номері частки ключа
            {
                int k=0; //к - добуток aj*xi
                for (int j=(m-1); j > 0; j--)//к - добуток
                {

                    k += a[j] * Convert.ToInt32(Math.Pow(x, j));
//генерація значень многочлена aj*xi
                    rtb_result.Text += "\nK=" + k.ToString();
                }
```

```
                y = (k + Convert.ToInt32(nud_var_S.Value)) %
Convert.ToInt32(nud_var_p.Value);
```

Далі згідно з вимог, на виході першого модуля необхідно сформувати n значень часток (x_i, y_i), що наведено у лістингу 3.3

Лістинг 3.3 - Вивід значень часток секрету на інтерфейс

```
                while(y<0) //якщо значення під модулем менше 0 то
додаємо значення модуля доки воно не стане більше або рівне 0
                {
                    y += Convert.ToInt32(nud_var_S.Value);
                }

                result += "\nX" + (x - 1).ToString() + "=" +
x.ToString() + "   Y" + (x - 1).ToString() + "=" + y.ToString();
            }
            rtb_Lagrang_m1.Text = result; //вивід значень часток
секрету на інтерфейс
```

Для відновлення секрету, необхідно зібрати m часток і обчислити для них базисні поліноми. Базисні поліноми – це фактично дроби, тому утворюємо два списки – один для чисельників, інший для знаменників (лістинг 3.4):

Лістинг 3.4 – Створення списків для обчислення базисних поліномів

```
public int interpol_polinom_Lagrange(List<int> x, List<int> y, int
m, int p) //інтерполяційний метод Лагранжа
        {
            int l_up = 0, l_down = 1, l = 0, b = 0, d = 0;
            List<int> x_up = new List<int>(); //масив чисельників
            List<int> x_down = new List<int>(); //масив знаменників

            x_up.Clear();
            x_down.Clear();
```

В кінці лістингу 3.4 використовуються функції для очищення списків, щоб не накопичувати дані при багаторазовому використанню програми.

В лістингу 3.5 наведено код для обчислення чисельників та знаменників базисних поліномів

Лістинг 3.5 - Код для обчислення чисельників та знаменників базисних поліномів

```
for (int i = 0; i < m; i++)//
{
    int x_basis_chis = 1, x_basis_znam = 1;

    for (int j = 0; j < x.Count; j++)
    {
        if (i != j)
        {
            x_basis_chis *= -x[j]; //добуток
чисельників
            x_basis_znam *= (x[i] - x[j]);//добуток
знаменників
        }
    }
    x_up.Add(x_basis_chis);
    x_down.Add(x_basis_znam);
```

В лістингу 3.6 приведено код для зведення до спільного знаменника всіх базисних поліномів та пошуку оберненого елемента до спільного знаменника

Лістинг 3.6 – Обчислення спільного знаменника та оберненого елемента до нього

```
{
    int k = x_up[h] * y[h] * (LCM(x_down) / x_down[h]);
    l_up += k;
}
l_down = LCM(x_down);
do
{
    b++;
} while (b * Math.Abs(l_down) % p != 1);
```

В лістингу 3.7 приведено код для відновлення секрету

Лістинг 3.7 – Відновлення секрету

```
l = ((b * l_up) % p);//знаходження значення L(x)
while (l < 0)//
{
    l += p;
}
return l; //
```

l - утворене число яке рівне секрету

Повний код програми приведено в додатку А.

2.2.2 Реалізація схеми розподілу секрету Асмута-Блюма

Згідно з вимогами до програмного забезпечення програма повинна передбачати введення секрету S, загальної кількості учасників протоколу n, кількості учасників для відновлення секрету m, і параметр системи p. Також потрібно передбачити можливість задати *n* взаємо-простих чисел $d_i, = \overline{1,n}$

В лістингу 3.8 наведено модуль для введення та перевірки параметрів схеми Асмута-Блюма.

Лістинг 3.8 - Модуль для введення та перевірки параметрів схеми Асмута-Блюма

```
private void btn_blum_m1_Click(object sender, EventArgs e) //виклик
функцій розбиття секрету
        {
            m1_Blum_Check();//перевірка введених значень

            RandomNumberSample LCM = new RandomNumberSample();
            Random rand = new Random();
            int d_sum = 1, r, S_revers;
            List<int> k = new List<int>();
            big_list_d.Clear();
            k.Clear();
            rtb_Blum_m1_result.Text = "";

            foreach (Control item in p_blum_add_d_m1.Controls)
//внесення значень di в масив
            {
                for(int i=0; i<nud_Blum_m_m1.Value; i++)
                {
                    if(item is NumericUpDown)
                    {

add_to_list_d(Convert.ToInt32(item.Name.Substring(item.Name.Length
- 1)), Convert.ToInt32(item.Text));
                    }
                }
            }
```

Частки $d_i, = \overline{1,n}$ повинні згідно вимог задовольняти умову $d_1 \cdot d_2 \cdot \ldots \cdot d_m < p \cdot d_{n-m+2} \cdot d_{n-m+3} \cdot \ldots \cdot d_n$. В лістингу 3.9 наведено код для перевірки умови:

Лістинг 3.9 - Перевірка часток $d_i, = \overline{1,n}$ на валідність

```
        if (LCM.LCM(big_list_d) != 1)
            {
                MessageBox.Show("Перевірте значення d!!!");
            }

            for (int i=0; i<nud_Blum_m_m1.Value; i++)
            {
                d_sum *= big_list_d[i]; //утворення суми масиву
значень d
            }
```

Далі необхідно було випадковим чином здійснити вибір параметра r та здійснити перевірку коректності вибраного значення, а саме відповідність умові

$$r < \frac{\prod_{i=1}^{m} d_i - S}{p}$$

В лістингу 3.10 наведено генерація та перевірка числа r

Лістинг 3.10 - Генерація та перевірка числа r

```
            do
            {
                r = rand.Next(0, 1000); //генерація випадкових
значень r
            } while (r < (d_sum - nud_Blum_S_m1.Value) /
nud_Blum_p_m1.Value);
```

Для обчислення других параметрів часток секрету можна використати Лістинг 3.11

Лістинг 3.11 – Формування k_i

```
S_revers = Convert.ToInt32(nud_Blum_S_m1.Value) + (r *
Convert.ToInt32(nud_Blum_p_m1.Value)); //утворення значення S

            for(int i=0; i<nud_Blum_n_m1.Value; i++)//виведення
результату
```

```
            {
                k.Add(S_revers % big_list_d[i]);
                rtb_Blum_m1_result.Text += "\nd" + i + "= " +
big_list_d[i] + "  k" + i + "= " + k[i];
            }
```

В лістингу 3.12 наведено код, що застосовує китайську теорему про остачі для відновлення проміжного результату S' з допомогою обчислених чисел D_j та обернених D_j^{-1}. Для обчислення обернених елементів в коді використовується перебір елементів відповідного поля та перевірка умови $D_j D_j^{-1} = 1 \; mod \; n$

Лістинг 3.12 – Обчислення проміжного результату S'

```
            for(int i=0; i < nud_Blum_m_m2.Value; i++)
            {
                D *= big_list_d[i];
            }

            for(int j=0; j < nud_Blum_m_m2.Value; j++)
            {
                D_list.Add(D / big_list_d[j]);
                int c = 0;

                do//пошук числа оберненого до D за допомогою
простого лічильника
            {

                {
                    c++;
                } while (D_list[j] * c % big_list_d[j]!=1);

                D_revers_list.Add(c);

                S_revers += (big_list_k[j] * D_list[j] *
D_revers_list[j]);
            }
```

Для відновлення початкового секрету використовує код `S_revers %= D`

Повний код програми наведено у додатку Б.

3 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Тестування схеми інтерполяційних многочленів Лагранжа

Здійснимо перевірку роботи програми для наступних даних: нехай S= 15, n=10, m=4, p=17.

Тоді нам потрібно побудувати поліном Лагранжа 3-го степеня згідно з формулою (1.1)

$$L(x) = a_3x^3 + a_2x^2 + a_1x^1 + a_0 \ \ mod\ p$$

Згідно з алгоритмом, покладаємо значення коефіцієнта a_0 рівним S, тобто $a_0 = 15$, а інші коефіцієнти вибираємо випадково. Нехай $a_1 = 38, a_2 = 27, a_3 = 84$. Тоді

$$L(x) = 84x^3 + 27x^2 + 38x^1 + 15\ mod\ 17$$

Обчислимо частки:

$$L(1) = (84 \cdot 1^3 + 27 \cdot 1^2 + 38 \cdot 1^1 + 15)\ mod\ 17 = 11$$
$$L(2) = (84 \cdot 2^3 + 27 \cdot 2^2 + 38 \cdot 2^1 + 15)\ mod\ 17 = 4$$
$$L(3) = (84 \cdot 3^3 + 27 \cdot 3^2 + 38 \cdot 3^1 + 15)\ mod\ 17 = 5$$
$$L(4) = (84 \cdot 4^3 + 27 \cdot 4^2 + 38 \cdot 4^1 + 15)\ mod\ 17 = 8$$
$$L(5) = (84 \cdot 5^3 + 27 \cdot 5^2 + 38 \cdot 5^1 + 15)\ mod\ 17 = 7$$
$$L(6) = (84 \cdot 6^3 + 27 \cdot 6^2 + 38 \cdot 6^1 + 15)\ mod\ 17 = 13$$
$$L(7) = (84 \cdot 7^3 + 27 \cdot 7^2 + 38 \cdot 7^1 + 15)\ mod\ 17 = 3$$
$$L(8) = (84 \cdot 8^3 + 27 \cdot 8^2 + 38 \cdot 8^1 + 15)\ mod\ 17 = 5$$
$$L(9) = (84 \cdot 9^3 + 27 \cdot 9^2 + 38 \cdot 9^1 + 15)\ mod\ 17 = 13$$
$$L(10) = (84 \cdot 10^3 + 27 \cdot 10^2 + 38 \cdot 10^1 + 15)\ mod\ 17 = 4$$

Тепер спробуємо відновити секрет за 4-ма відомими частками:

$$L(1) = 11, \qquad L(5) = 7, \qquad L(7) = 3, \qquad L(10) = 4$$

Обчислимо базисні поліноми для кожної з часток за формулою (1.3):

$$l_i(x) = \prod_{j=0, j\neq j}^{m-1} \frac{x - x_j}{x_i - x_j}$$

$$l_1(1) = \frac{x-5}{1-5} \cdot \frac{x-7}{1-7} \cdot \frac{x-10}{1-10} = \frac{1}{-216}(x^3 - 22x^2 + 155x - 350)$$

$$l_2(5) = \frac{x-1}{5-1} \cdot \frac{x-7}{5-7} \cdot \frac{x-10}{5-10} = \frac{1}{40}(x^3 - 18x^2 + 87x - 70)$$

$$l_3(7) = \frac{x-1}{7-1} \cdot \frac{x-5}{7-5} \cdot \frac{x-10}{7-10} = \frac{1}{-36}(x^3 - 16x^2 + 65x - 50)$$

$$l_4(10) = \frac{x-1}{10-1} \cdot \frac{x-5}{10-5} \cdot \frac{x-7}{10-7} = \frac{1}{135}(x^3 - 13x^2 + 47x - 35)$$

Після цього можна сконструювати многочлен Лагранжа за формулою (1.2), домноживши відповідне значення базисного полінома на y:

$$L(x) = -\frac{11}{216}(x^3 - 22x^2 + 155x - 350) + \frac{7}{40}(x^3 - 18x^2 + 87x - 70)$$
$$-\frac{3}{36}(x^3 - 16x^2 + 65x - 50) + \frac{4}{135}(x^3 - 13x^2 + 47x - 35)$$

Зведемо до спільного знаменника вираз $L(x)$. Найменшим спільним кратним чисел 216, 40, 36 і 135 буде 1080

$$L(x) = \left(-\frac{11}{216}(x^3 - 22x^2 + 155x - 350) + \frac{7}{40}(x^3 - 18x^2 + 87x - 70)\right.$$
$$-\frac{3}{36}(x^3 - 16x^2 + 65x - 50)$$
$$\left.+\frac{4}{135}(x^3 - 13x^2 + 47x - 35)\right) mod 17 =$$

$$= \left(-\frac{55}{1080}(x^3 - 22x^2 + 155x - 350) + \frac{189}{1080}(x^3 - 18x^2 + 87x - 70) \right.$$
$$- \frac{90}{1080}(x^3 - 16x^2 + 65x - 50)$$
$$\left. + \frac{32}{1080}(x^3 - 13x^2 + 47x - 35) \right) mod 17$$

Зведемо спільні доданки

$$L(x) = \left(\frac{1}{1080}(-55x^3 + 1210x^2 - 8525x + 19250 + 189x^3 - 3402x^2 + 16443x - 13230 - 90x^3 + 1440x^2 - 5850x + 4500 + 32x^3 - 416x^2 + 1504x - 1120) \right) mod 17 = \left(\frac{1}{1080}(76x^3 - 1168x^2 + 3572x + 9400) \right) mod 17$$

Тепер потрібно знайти обернений елемент до 1080 за модулем 17. Знайдемо спершу 1080 mod 17 =9, тому що $1080 = 17 \cdot 63 + 9$

Тепер виконаємо алгоритм Евкліда

$$17 = 9 \cdot 1 + 8$$
$$9 = 8 \cdot 1 + 1$$

Звідси виразимо остачі:

$$8 = 17 - 9 \cdot 1$$
$$1 = 9 - 8 \cdot 1$$

І підставимо значення першого виразу в другий

$$1 = 9 - 8 \cdot 1 = 9 - (17 - 9 \cdot 1) \cdot 1 = -1 \cdot 17 \cdot +2 \cdot 9$$

Отримали вираз для розширеного алгоритму Евкліда. Отже, оберненим елементом до 9 буде 2, адже $9 \cdot 2 \; mod \; 17 = 1$

Для відновлення секрету нам потрібно вільний член многочлену 9400 помножити на отриманий обернений елемент 2 і виконати операцію mod 17

$$9400 \cdot 2 \; mod \; 17 = 15$$

Очевидно, що нам вдалося успішно відновити секрет. На рисунку 3.1 наведено результат тестування схеми розділення секрету з допомогою поліному Лагранжа для вище наведеного прикладу:

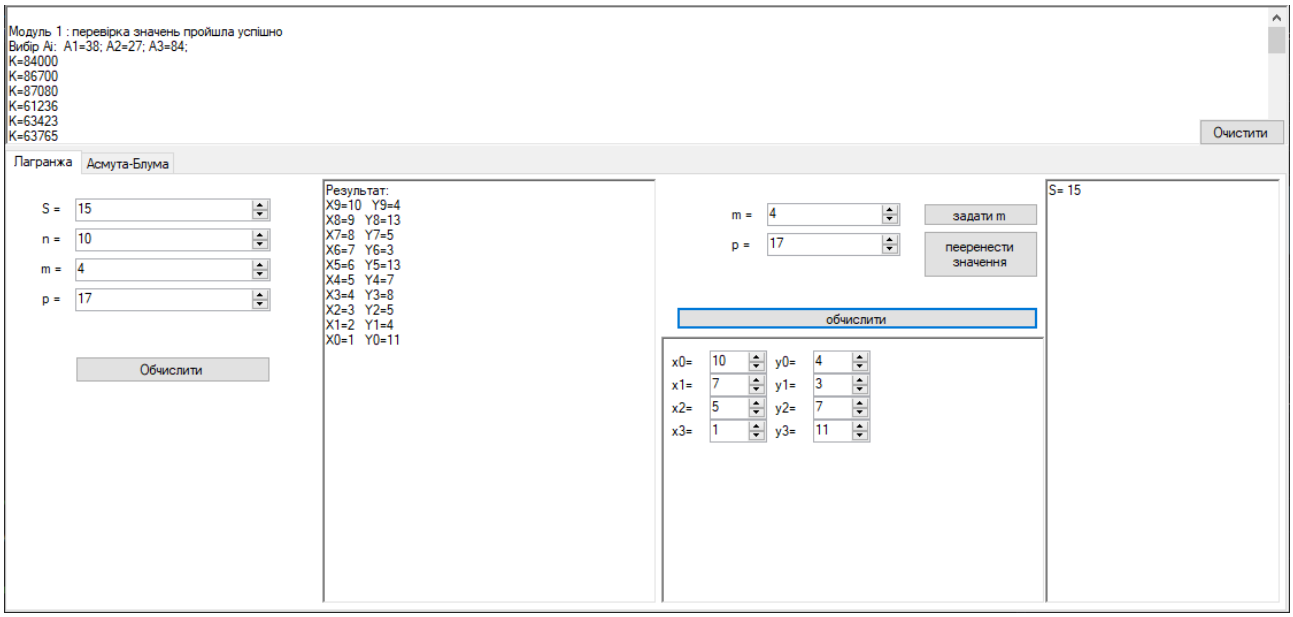

Рисунок 3.1 – Результат тестування програми, що реалізовує схему інтерполяційних поліномів Лагранжа

Як бачимо результати, отримані в програмі відповідають обчисленням, які ми провели.

3.2 Тестування схеми розподілу секрету Асмута-Блюма

Виконаємо перевірку роботи програми для наступних параметрів секрету S=14, загальної кількості учасників протоколу n=5, кількості учасників для відновлення секрету m=3, якщо параметр системи p=17. Задамо п'ять взаємо-простих чисел $d_1 = 2,\ d_2 = 3, d_3 = 5, d_4 = 7, d_5 = 17$.

Перевіримо вимоги до правильності введених значень. В п.1.2.3 сказано, що числа $d_i, i = \overline{1,n}$ повинні бути впрорядковані за зростанням, що в даному випадку відповідає дійсності. Крім того повинна виконуватися умова:

$$d_1 \cdot d_2 \cdot \ldots \cdot d_m < p \cdot d_{n-m+2} \cdot d_{n-m+3} \cdot \ldots \cdot d_n$$

Перевіримо виконання цієї умови:

$$2 \cdot 3 \cdot 5 < 17 \cdot 7 \cdot 17$$

Як бачимо вимога виконується.

Далі необхідно вибрати випадкове число *r*, що задовольняє умові:

$$r < \frac{\prod_{i=1}^{m} d_i - S}{p}.$$

Нехай $r = 5$ Обчислимо тоді число

$$S' = S + rp = 14 + 5 \cdot 17 = 99$$

Визначити другий параметр $k_i = S'\ mod\ d_i$для часток (d_i, k_i), отже

$$k_1 = 99\ mod\ 2 = 1$$
$$k_2 = 99\ mod\ 3 = 0$$
$$k_3 = 99\ mod\ 5 = 4$$
$$k_4 = 99\ mod\ 7 = 1$$
$$k_5 = 99\ mod\ 17 = 14$$

Для відновлення секрету використаємо частки (2;1); (3;0); (17;14). Це означає, що

$$S'\ mod\ 2 = 1$$
$$S'\ mod\ 3 = 0$$
$$S'\ mod\ 17 = 14$$

Далі необхідно обчислити добуток D взаємно простих чисел d_j:

$$D = 2 \cdot 3 \cdot 17 = 102$$

Потім необхідно обчислити коефіцієнти $D_j = \frac{D}{d_j}$

$$D_1 = \frac{102}{2} = 51, \quad D_2 = \frac{102}{3} = 34, \quad D_3 = \frac{102}{17} = 6$$

Далі необхідно обчислити обернені елементи D_j^{-1} до D_j за модулем d_j (за розширеним алгоритмом Евкліда)

$$D_1^{-1} = 51^{-1} mod\ 2 = 1$$
$$D_2^{-1} = 34^{-1} mod\ 3 = 1$$
$$D_3^{-1} = 6^{-1} mod\ 17 = 3$$

Тоді за китайською теоремою про остачі

$$S' = \sum_j k_j\, D_j D_j^{-1} \ \ mod\ D$$

Отже,

$$S' = (1 \cdot 51 \cdot 1 + 0 \cdot 34 \cdot 1 + 14 \cdot 6 \cdot 3) mod\ 102 = 99$$

Для отримання значення секрету S необхідно використати формулу

$$S = S' mod\ p = 99\ mod\ 17 = 14$$

Результати тестування програмного забезпечення для вище наведених параметрів наведено на рисунку 3.3

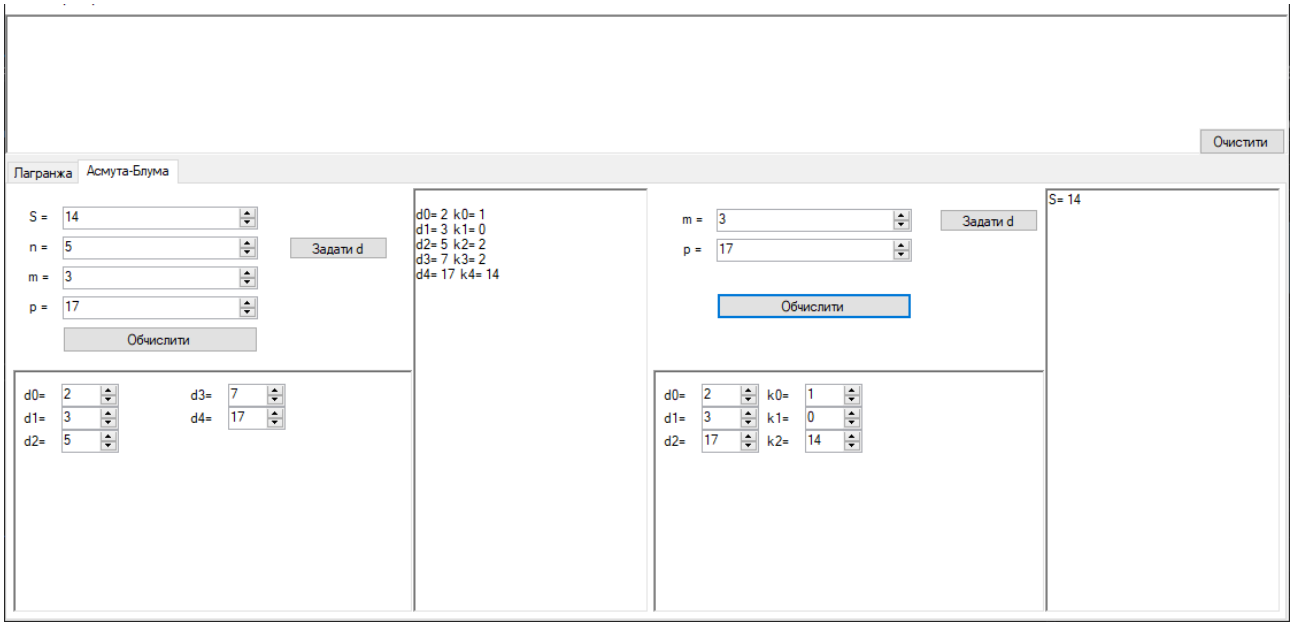

Рисунок 3.3 – Результати тестування програми для формування часток та відновлення секрету за схемою Асмута-Блюма

Як бачимо ліва частина інтерфейсу відповідає формуванню часток учасників, а права – призначена для відновлення секрету за відомими частками. При коректному застосуванні параметрів програми, нам вдалося відновити секрет.

Подивимось, що буду випадку, якщо при формуванні часток, ми передбачили 3 частки для відновлення секрету, а при відновленні зібралося лише 2 учасника. На рисунку 3.4 показано, що при меншій кількості учасників, ніж це передбачено пороговою схемою нам не вдалося коректно відновити секрет.

Спробуємо продемонструвати алгоритм відновлення

Для відновлення секрету дано частки (23;20); (25;16). Це означає, що

$$S' \; mod \; 23 = 20$$

$$S' \; mod \; 25 = 16$$

Оголошено параметр системи $p = 59$

Далі необхідно обчислити добуток D взаємно простих чисел d_j:

$$D = 23 \cdot 25 = 575$$

Потім необхідно обчислити коефіцієнти $D_j = \frac{D}{d_j}$

$$D_1 = \frac{575}{23} = 25, \quad D_2 = \frac{575}{25} = 23$$

Далі необхідно обчислити обернені елементи D_j^{-1} до D_j за модулем d_j (за розширеним алгоритмом Евкліда)

Обернений елемент знайшли шляхом обчислень

$$25 = 23 \cdot 1 + 2$$
$$23 = 2 \cdot 11 + 1$$

Звідси:

$$1 = 23 - 2 \cdot 11 = 23 - 11(25 - 23 \cdot 1) = -11 \cdot 25 + 12 \cdot 23$$

Отже,

$$D_1^{-1} = 25^{-1} mod\ 23 = -11\ mod\ 23 = 12$$
$$D_2^{-1} = 23^{-1} mod\ 25 = 12$$

Тоді за китайською теоремою про остачі

$$S' = \sum_j k_j\, D_j D_j^{-1} \ \ mod\ D$$

Отже,

$$S' = (20 \cdot 25 \cdot 12 + 16 \cdot 23 \cdot 12) mod\ 575 = 66$$

Для отримання значення секрету S необхідно використати формулу

$$S = S' mod\ p = 66\ mod\ 59 = 7$$

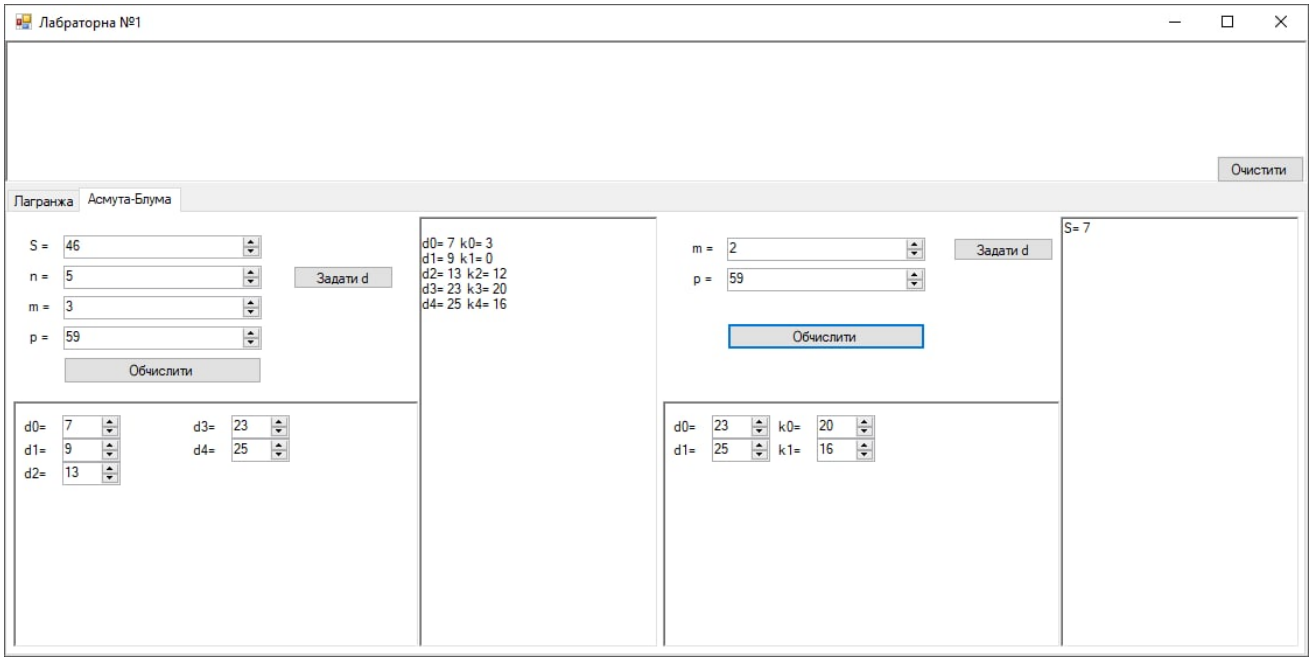

Рисунок 3.4 – Результати тестування програми при меншій кількості часток ніж передбачено пороговою схемою

Як бачимо, при використанні порогової схеми з меншою, ніж передбачено кількістю часток для відновлення секрету, програма не відновлює секрет, який був розподілений.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ХОРОНИ ПРАЦІ

4.1 Проведення інструктажів з охорони праці

Працівники, під час прийняття на роботу та періодично, повинні проходити на підприємстві інструктажі з питань охорони праці, надання першої медичної допомоги потерпілим від нещасних випадків, а також з правил поведінки та дій при виникненні аварійних ситуацій, пожеж і стихійних лих.

За характером і часом проведення інструктажі з питань охорони праці (далі інструктажі) поділяються на вступний, первинний, повторний, позаплановий та цільовий.

Вступний інструктаж проводиться:

- з усіма працівниками, які приймаються на постійну або тимчасову роботу, незалежно від їх освіти, стажу роботи та посади;
- з працівниками інших організацій, які прибули на підприємство і беруть безпосередню участь у виробничому процесі або виконують інші роботи для підприємства;
- з учнями та студентами, які прибули на підприємство для проходження трудового або професійного навчання;
- з екскурсантами у разі екскурсії на підприємство.

Вступний інструктаж проводиться спеціалістом служби охорони праці або іншим фахівцем відповідно до наказу (розпорядження) по підприємству, який в установленому Типовим положенням порядку пройшов навчання і перевірку знань з питань охорони праці.

Вступний інструктаж проводиться в кабінеті охорони праці або в приміщенні, що спеціально для цього обладнано, з використанням сучасних технічних засобів навчання, навчальних та наочних посібників за програмою, розробленою службою охорони праці з урахуванням особливостей виробництва.

Запис про проведення вступного інструктажу робиться в журналі реєстрації вступного інструктажу з питань охорони праці , який зберігається службою

охорони праці або працівником, що відповідає за проведення вступного інструктажу, а також у наказі про прийняття працівника на роботу.

Первинний інструктаж проводиться до початку роботи безпосередньо на робочому місці з працівником:

- новоприйнятим (постійно чи тимчасово) на підприємство або до фізичної особи, яка використовує найману працю;
- який переводиться з одного структурного підрозділу підприємства до іншого;
- який виконуватиме нову для нього роботу;
- відрядженим працівником іншого підприємства, який бере безпосередню участь у виробничому процесі на підприємстві.

Повторний інструктаж на робочому місці індивідуально з окремим працівником або групою працівників, які виконують однотипні роботи, за обсягом і змістом переліку питань первинного інструктажу.

Повторний інструктаж проводиться в терміни, визначені нормативно-правовими актами з охорони праці, які діють у галузі, або роботодавцем (фізичною особою, яка використовує найману працю) з урахуванням конкретних умов праці, але не рідше:

- на роботах з підвищеною небезпекою - 1 раз на 3 місяці;
- для решти робіт - 1 раз на 6 місяців.

Позаплановий інструктаж проводиться з працівниками на робочому місці або в кабінеті охорони праці:

- при введенні в дію нових або переглянутих нормативно-правових актів з охорони праці, а також при внесенні змін та доповнень до них;
- при зміні технологічного процесу, або модернізації устаткування, приладів та інструментів, вихідної сировини, матеріалів та інших факторів, що впливають на стан охорони праці;
- при порушеннях працівниками вимог нормативно-правових актів з охорони праці, що призвели до травм, аварій, пожеж тощо;

- при перерві в роботі виконавця робіт більш ніж на 30 календарних днів - для робіт з підвищеною небезпекою, а для решти робіт - понад 60 днів.

Позаплановий інструктаж з учнями, студентами, курсантами, слухачами проводиться під час проведення трудового і професійного навчання при порушеннях ними вимог нормативно - правових актів з охорони праці, що можуть призвести або призвели до травм, аварій, пожеж тощо.

Позаплановий інструктаж може проводитись індивідуально з окремим працівником або з групою працівників одного фаху.

Цільовий інструктаж проводиться з працівниками:

- при ліквідації аварії або стихійного лиха;
- при проведенні робіт, на які відповідно до законодавства оформлюються наряд-допуск, наказ або розпорядження.

Цільовий інструктаж проводиться індивідуально з окремим працівником або з групою працівників. Обсяг і зміст цільового інструктажу визначаються залежно від виду робіт, що виконуватимуться.

Первинний, повторний, позаплановий і цільовий інструктажі проводить безпосередній керівник робіт (начальник структурного підрозділу, майстер) або фізична особа, яка використовує найману працю.

Первинний, повторний, позаплановий і цільовий інструктажі завершуються перевіркою знань у вигляді усного опитування або за допомогою технічних засобів, а також перевіркою набутих навичок безпечних методів праці, особою, яка проводила інструктаж.

При незадовільних результатах перевірки знань, умінь і навичок щодо безпечного виконання робіт після первинного, повторного чи позапланового інструктажів протягом 10 днів додатково проводяться інструктаж і повторна перевірка знань.

При незадовільних результатах перевірки знань після цільового інструктажу допуск до виконання робіт не надається. Повторна перевірка знань при цьому не дозволяється.

Про проведення первинного, повторного, позапланового та цільового інструктажів та їх допуск до роботи, особа, яка проводила інструктаж, уносить запис до журналу реєстрації інструктажів з питань охорони праці на робочому місці. Сторінки журналу реєстрації інструктажів повинні бути пронумеровані, прошнуровані і скріплені печаткою.

У разі виконання робіт, що потребують оформлення наряду-допуску, цільовий інструктаж реєструється в цьому наряді-допуску, а в журналі реєстрації інструктажів не обов'язково.

Перелік професій та посад працівників, які звільняються від повторного інструктажу, затверджується роботодавцем. До цього переліку можуть бути зараховані працівники, участь у виробничому процесі яких не пов'язана з безпосереднім обслуговуванням об'єктів, машин, механізмів, устаткування; застосуванням приладів та інструментів, збереженням або переробкою сировини, матеріалів тощо.

4.2 Планування робіт щодо охорони праці

Робота з охорони праці здійснюється у відповідності з перспективнимим і поточним планів створення безпечних і нешкідливих умов праці, в яких визначені задачі підприємству в цілому і окремим структурним підрозділам, а також керівникам і спеціалістам.

Планування робіт здійснюється на основі:

- заходів, які забезпечують досягнення встановлених нормативів безпеки праці, гігієни праці та виробничого середовища; заходів, передбачених колективним договором;
- заходів по усуненню недоліків, виявлених при розслідуванні нещасних випадків, професійних захворювань і аварій.

Контроль за станом охорони праці включає:

- оцінку рівня небезпечних виробничих факторів(НВФ) і шкідливих виробничих факторів(ШВФ) на робочих місцях;

- виявлення порушення вимог законів і нормативних актів з охорони праці;
- перевірку усунення раніше виявлених порушень;
- перевірку виконання працівником обов'язків з охорони праці;
- перевірку виконання планів робіт з охорони праці;
- перевірку забезпечення працівників ЗІЗ і ЗКЗ.

Види контролю:

- зі сторони органів державного нагляду;
- зі сторони служби з охорони праці;
- оперативний контроль керівниками і іншими посадовими особами підприємства;
- громадський контроль;
- комісія підприємства, уповноваженою працівниками особою з питань охорони праці.

Оцінка стану охорони праці і результатів профілактичної роботи здійснюється за прийнятими на підприємстві показниками. Як джерело вихідної інформації використовуються: акти про нещасні випадки, звіти про виробничий травматизм; матеріали атестації робочих місць, паспорта санітарно-гігієнічного стану умов праці; журнали оперативного контролю за станом охорони праці структурного підрозділу, акти і приписи перевірок стану охорони праці.

Узагальнені дані про стан охорони праці і результатів профілактичної роботи підготовлюються службою охорони праці і підлягають обов'язковому розгляду і аналізу на всіх рівнях управління підприємства.

Стимулювання роботи з охорони праці, направлене на підвищення зацікавленості працівників у забезпеченні безпечних умов праці, здійснюється відповідно Положенню, існуючому на підприємстві, в якому визначені конкретні показники, умови, види і форми заохочення за активну участь і ініціативу в реалізації заходів з підвищення безпеки праці і за роботу без порушень правил безпеки, а також заходи впливу на порушників.

ВИСНОВКИ

В процесі виконання кваліфікаційної роботи було проаналізовано специфічні криптографічні протоколи, такі як протокол розділення секрету, підкидання чесної монети, доведення з нульовим знанням та інші. Безумовно, такі протоколи застосовуються рідше, ніж стандартні криптографічні алгоритми, проте часто вони є невід'ємною частиною безпеки системи. Метою даної роботи була розробка програмного забезпечення для криптографічного протоколу розділення секрету, зокрема було реалізовано дві порогові схеми –схема Шаміра (з використанням многочленів Лагранжа) та схема Асмута-Блюма.

В результаті виконання кваліфікаційної роботи виконано наступні завдання:

- Проведено огляд літературних джерел в предметній області.
- Вивчено досвід застосування схеми розділення секретів.
- Проаналізувано основні відомі алгоритми протоколу розділення секрету.
- Обрано середовище розробки.
- Розроблено програмне забезпечення.
- Протестовано розроблене програмне рішення.

В першому розділі описано сутність криптографічних протоколів та алгоритми відновлення секрету. В другому розділі обґрунтовано вибір програмного середовища та описано реалізацію розробки. В третьому розділі висвітлено результати тестування програмної розробки. Крім того, в розділі "Безпека життєдіяльності, основи хорони праці" висвітлено питання проведення інструктажів з охорони праці та планування робіт щодо охорони праці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бабаш А.В., Шанкин Г.П. Криптография. Москва, СОЛОН-Р, 2002, 511 с.
2. Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушки А.В. Основи криптографии. Москва, Гелиос, 2002, 480 с.
3. Миронов А.М. Криптографические протоколи Москва электронная публикация, 2016, 119с. Режим доступу до ресурсу: http://www.dut.edu.ua/uploads/l_1122_30338177.pdf
4. В.В. Ященко Введение в криптографию МЦНМО, 2012, 352 с. ISBN: 978-5-4439-0026-1
5. Брюс Шнайер: Прикладна криптографія. Протоколи, алгоритми і вихідний код на С— К:Діалектика, 2002. —— 1040 с. ISBN 978-5-9908462-4-1
6. Beimel, A.: Secure Schemes for Secret Sharing and Key Distribution. PhD thesis, Technion (1996), http://www.cs.bgu.ac.il/~beimel/pub.html
7. What Is C# Language, Advantages & Features Of C# Language [Електронний ресурс] / fyber // codexoxo. – 2019. – Режим доступу до ресурсу: https://www.codexoxo.com/advantages-c-sharp-language/..
8. Основи охорони праці: Підруч для студ вищих навч закладів За ред мп Гандзюка - К Каравела, 2004 - 408 с.
9. Охорона праці в галузі комп'ютингу: підручник / Л. А. Катренко, А. В. Катренко ; [за наук. ред. В. В. Пасічника] ; М-во освіти і науки, молоді та спорту України. — Л. : Магнолія 2006, 2012. — 544 с

ДОДАТКИ

Лістинг А1 - Код програми, що реалізує схему Шаміра

```
var rand = new Random();
            int y=0, m = Convert.ToInt32(nud_var_m.Value), n =
Convert.ToInt32(nud_var_n.Value);
            string result = "Результат:";

            rtb_result.Text += "\nВибір Ai: ";

            List<int> a = new List<int>();

            for (int i = 0; i < m; i++) //вибір випадкових значень
A
            {
                a.Add(rand.Next(0, 100));
                rtb_result.Text += i == 0 ? "" : " A" + i + "=" +
a[i] + ";";
            }

            for (int x= n; x>0; x--)//генеруємо x який дорівнює
послідовному номері частки ключа
            {
                int k=0; //к - добуток aj*xi
                for (int j=(m-1); j > 0; j--)//к - добуток
                {

                    k += a[j] * Convert.ToInt32(Math.Pow(x, j));
//генерація значень многочлена aj*xi
                    rtb_result.Text += "\nK=" + k.ToString();
                }

                y = (k + Convert.ToInt32(nud_var_S.Value)) %
Convert.ToInt32(nud_var_p.Value); //генерація числа Y[i]

                while(y<0) //якщо значення під модулем менше 0 то
додаємо значення модуля доки воно не стане більше або рівне 0
                {
                    y += Convert.ToInt32(nud_var_S.Value);
                }

                result += "\nX" + (x - 1).ToString() + "=" +
x.ToString() + "   Y" + (x - 1).ToString() + "=" + y.ToString();
            }
            rtb_Lagrang_m1.Text = result; //вивід значень часток
секрету на інтерфейс

        }
```

Лістинг А2 - Відновлення секрету за допомогою полінома Лагранжа

```
            big_list_x.Clear();
               big_list_y.Clear();//масиви із введеними частками ключа
               RandomNumberSample interpol = new RandomNumberSample();
               foreach (Control item in p_mxy_m2.Controls)
//наповнення масивів
               {
                   if (item is NumericUpDown)
                   {
                       if (item.Name.Contains("x"))
                       {

add_to_list_x(Convert.ToInt32(item.Name.Substring(item.Name.Length
- 1)),Convert.ToInt32(item.Text));
                       }
                       else if (item.Name.Contains("y"))
                       {

add_to_list_y(Convert.ToInt32(item.Name.Substring(item.Name.Length
- 1)), Convert.ToInt32(item.Text));
                       }
                   }
               }
               rtb_Lagrang_m2.Text = "S= ";
               rtb_Lagrang_m2.Text +=
interpol.interpol_polinom_Lagrange(big_list_x, big_list_y,
Convert.ToInt32(nud_m_m2.Value),
Convert.ToInt32(nud_p_m2.Value));//виклик інтерполяційного полінома

public int interpol_polinom_Lagrange(List<int> x, List<int> y, int
m, int p) //інтерполяційний метод Лагранжа
           {
               int l_up = 0, l_down = 1, l = 0, b = 0, d = 0;
               List<int> x_up = new List<int>(); //масив чисельників
               List<int> x_down = new List<int>(); //масив знаменників

               x_up.Clear();
               x_down.Clear();//обов'язкове очищення масивів аби не
накопичувати дані при багаторазовому використанню програми

               for (int i = 0; i < m; i++)//цикли для утворення
чисельника та знаменника полінома(нас цікавить лише значення
многочлена без коефіцієнта)
               {
                   int x_basis_chis = 1, x_basis_znam = 1;

                   for (int j = 0; j < x.Count; j++)
                   {
                       if (i != j)
                       {
                           x_basis_chis *= -x[j]; //добуток
чисельників
```

```
                        x_basis_znam *= (x[i] - x[j]);//добуток
знаменників
                    }
                }
                x_up.Add(x_basis_chis);
                x_down.Add(x_basis_znam);
            }

            for (int h = 0; h < x_up.Count; h++)//зведення під один
знаменник
            {
                int k = x_up[h] * y[h] * (LCM(x_down) / x_down[h]);
                l_up += k;
            }
            l_down = LCM(x_down);
            do //пошук числа оберненого до b за допомогою простого
лічильника(ніколи не помиляється))
            {
                b++;
            } while (b * Math.Abs(l_down) % p != 1);


            l = ((b * l_up) % p);//знаходження значення L(x)
            while (l < 0)//якщо число під модулем від'ємне то
додаємо значення модуля
            {
                l += p;
            }
            return l; //утворене число яке рівне секрету
        }

  static int LCM(int a, int b) //Найменьше спільне кратне двох
чисел
        {
            return (Math.Abs(a) / GCD(Math.Abs(a), Math.Abs(b))) *
Math.Abs(b);
        }
  public int LCM(List<int> numbers) //Найменше спільне кратне
масиву чисел
        {
            return numbers.Aggregate(LCM); //виклик функції вище
для всіх елементів
        }
```

Лістинг Б1 – Формування часток секрету за схемою Асмута-Блюма

```
private void btn_blum_m1_Click(object sender, EventArgs e) //виклик функцій розбиття секрету
        {
            m1_Blum_Check();//перевірка введених значень

            RandomNumberSample LCM = new RandomNumberSample();
            Random rand = new Random();
            int d_sum = 1, r, S_revers;
            List<int> k = new List<int>();
            big_list_d.Clear();
            k.Clear();
            rtb_Blum_m1_result.Text = "";

            foreach (Control item in p_blum_add_d_m1.Controls) //внесення значень dᵢ в масив
            {
                for(int i=0; i<nud_Blum_m_m1.Value; i++)
                {
                    if(item is NumericUpDown)
                    {

add_to_list_d(Convert.ToInt32(item.Name.Substring(item.Name.Length - 1)), Convert.ToInt32(item.Text));
                    }
                }
            }

            if (LCM.LCM(big_list_d) != 1) //перевірка значень d на валідність
            {
                MessageBox.Show("Перевірте значення d!!!");
            }

            for (int i=0; i<nud_Blum_m_m1.Value; i++)
            {
                d_sum *= big_list_d[i]; //утворення суми масиву значень d
            }

            do
            {
                r = rand.Next(0, 1000); //генерація випадкових значень r
            } while (r < (d_sum - nud_Blum_S_m1.Value) / nud_Blum_p_m1.Value);

            S_revers = Convert.ToInt32(nud_Blum_S_m1.Value) + (r * Convert.ToInt32(nud_Blum_p_m1.Value)); //утворення значення S

            for(int i=0; i<nud_Blum_n_m1.Value; i++)//виведення результату
```

```
            {
                k.Add(S_revers % big_list_d[i]);
                rtb_Blum_m1_result.Text += "\nd" + i + "= " +
big_list_d[i] + "  k" + i + "= " + k[i];
            }
        }
```

Лістинг Б.2 - Відновлення секрету

```
private void btn_Blum_m2_Click(object sender, EventArgs e) //виклик
функцій відновлення секрету
        {
            big_list_d.Clear();
            big_list_k.Clear();
            int D = 1;
            List<int> D_list = new List<int>();
            List<int> D_revers_list = new List<int>();
            D_list.Clear();
            D_revers_list.Clear();
            int S_revers = 0;

            foreach (Control item in p_Blum_d_k_m2.Controls)
//внесення значень часток секрету в масиви
            {
                if (item is NumericUpDown)
                {
                    if (item.Name.Contains("_d"))
                    {

add_to_list_d(Convert.ToInt32(item.Name.Substring(item.Name.Length
- 1)), Convert.ToInt32(item.Text));
                    }
                    else if (item.Name.Contains("_k"))
                    {

add_to_list_k(Convert.ToInt32(item.Name.Substring(item.Name.Length
- 1)), Convert.ToInt32(item.Text));
                    }
                }
            }

            for(int i=0; i < nud_Blum_m_m2.Value; i++)
            {
                D *= big_list_d[i];
            }

            for(int j=0; j < nud_Blum_m_m2.Value; j++)
            {
                D_list.Add(D / big_list_d[j]);
                int c = 0;

                do//пошук числа оберненого до D за допомогою
простого лічильника
            {
```

```
                {
                    c++;
                } while (D_list[j] * c % big_list_d[j]!=1);

                D_revers_list.Add(c);

                S_revers += (big_list_k[j] * D_list[j] *
D_revers_list[j]);
            }

            S_revers %= D;

            rtb_Blum_result_m2.Text += "\nS= " + (S_revers %
Convert.ToInt32(nud_blum_p_m2.Value)).ToString();
        }
```