Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

на тему:

РОЗРОБКА ПРОГРАМИ РОЗПОДІЛУ СЕКРЕТУ МІЖ ГРУПОЮ КОРИСТУВАЧІВ

Виконав: студент 4 курсу, гр.1КІТС-20б
спеціальності 125– Кібербезпека
Освітня програма – Кібербезпека
інформаційних технологій та систем
(шифр і назва напряму підготовки, спеціальності)

Шкробот Є. С.
(прізвище та ініціали)

Керівник: д.т.н., проф., каф. МБІС
Яремчук Ю. Є .
(прізвище та ініціали)
«____»_____2024 р.

Рецензент: к.т.н., доц., каф. ОТ
Тарновський М. Г.
(прізвище та ініціали)
«____»_____2024 р.

Допущено до захисту
Голова секції УБ, кафедри МБІС
д.т.н., проф. Яремчук Ю.Є.
“_____”_____2024р.

Вінниця ВНТУ – 2024

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти І-й (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС

д.т.н., проф. Яремчук Ю.Є.
" 22 " березня 2024 р.

З А В Д А Н Н Я
на бакалаврську дипломну роботу студенту
Шкробот Єгор Сергійович
(прізвище, ім`я, по-батькові)

1. Тема роботи Розробка програми розподілу секрету між групою користувачів
Керівник роботи Яремчу Юрій Євгенович д.т.н., професор кафедри МБІС.
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "11" березня 2024року
№ 80
2. Термін подання студентом роботи_____
3. Вихідні дані до роботи Електронні джерела, наукові статті, підручники та офіційні документи по темі роботи. Тип даних – секрет, схема розподілу секрету – порогова.
4. Зміст текстової частини:
Для успішного виконання роботи: Дослідити предметну область розподілу секрету та порогових криптосистем. Здійснити аналіз сучасних методів розподілу секрету, включаючи схему Шаміра. Здійснити вибір найбільш підходящих методів програмування для реалізації розподілу секрету. Розробити програмне забезпечення для розподілу секрету між групою користувачів на основі схеми Шаміра. Тестувати розроблене програмне забезпечення та провести аналіз його ефективності у різних умовах використання.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень) Ілюстрація схеми розподілу секрету Шамира, Детальний алгоритм роботи схеми Шамира, Ілюстрація схеми порогового розподілу секрету Шаміра,
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ I	Яремчу Ю. Є. д.т.н., проф. кафедри МБІС		
Розділ II	Яремчу Ю. Є. д.т.н., проф. кафедри МБІС		
Розділ III	Яремчу Ю. Є. д.т.н., проф. кафедри МБІС		

7.Дата видачі завдання 22 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	23.03.2024	05.04.2024	
2	Проектування схеми розподілу секрету	05.04.2024	16.04.2024	
3	Проектування схеми відновлення секрету	16.04.2024	28.04.2024	
4	Тестування розробленої програми	28.04.2024	20.05.2024	
5	Попередній захист БДР	24.05.2024	02.06.2024	
6	Виправлення роботи	03.06.2024	11.06.2024	
7	Захист БДР	13.06.2024	17.06.2024	

Студент _____ Шкробот Є. С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Яремчук Ю. Є.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська робота складається з 55 сторінки, 18 ілюстрацій, 4 додатків та 36 джерел інформації.

Дана бакалаврська дипломна робота присвячена розробці програми для розподілу секрету між групою користувачів на основі схеми Шаміра. Метою дослідження є створення ефективногомеханізму захисту конфіденційної інформації, який забезпечить високий рівень безпеки та гнучкість у використанні.

Дипломна робота включає аналіз сучасних порогових криптосистем та їх застосування для розподілу секрету. У роботі представлено детальний огляд схеми Шаміра, її принципи роботи та переваги над іншими методами. Розроблена програма реалізує алгоритм схеми Шаміра, дозволяючи безпечно розподіляти секретну інформацію між декількома користувачами і відновлювати її за наявності необхідної кількості частин. Проведено тестування програми та аналіз її ефективності у різних умовах використання.

Програма забезпечує надійний захист даних, що робить її незамінною у багатьох сучасних додатках, таких як розподілені системи зберігання даних, аутентифікація та контроль доступу, а також безпека виборчих систем.

Ключові слова: розподіл секрету, порогові криптосистеми, схема Шаміра, інформаційна безпека, аутентифікація, контроль доступу, розподілені системи зберігання даних, захист даних, конфіденційність, цілісність.

ABSTRACT

The bachelor's thesis consists of 55 A4 pages, which have 18 illustrations, 4 Appendices and the list of sources used contains 36 items.

This bachelor's thesis is dedicated to the development of a program for secret sharing among a group of users based on Shamir's scheme. The aim of the research is to create an effective mechanism for protecting confidential information, which will ensure a high level of security and flexibility in use.

The thesis includes an analysis of modern threshold cryptosystems and their application for secret sharing. The work presents a detailed overview of Shamir's scheme, its working principles, and advantages over other methods. The developed program implements the Shamir's scheme algorithm, allowing secure distribution of secret information among multiple users and its recovery when the required number of shares is available. The program's effectiveness has been tested and analyzed under various conditions of use.

The program provides reliable data protection, making it indispensable in many modern applications, such as distributed data storage systems, authentication and access control, as well as election system security.

Keywords: secret sharing, threshold cryptosystems, Shamir's scheme, information security, authentication, access control, distributed data storage systems, data protection, confidentiality, integrity.

ЗМІСТ

ВСТУП

Актуальність задачі. В умовах стрімкого розвитку інформаційних технологій та зростання кількості конфіденційних даних, які потребують надійного захисту, питання забезпечення безпеки інформації стає дедалі актуальнішим. Розподіл секрету між групою користувачів є важливим інструментом у сфері інформаційної безпеки, який дозволяє захистити критичну інформацію від несанкціонованого доступу, зберігаючи її у розподіленій системі. Порогові криптосистеми, зокрема схема Шаміра, забезпечують високий рівень захисту даних, що робить їх незамінними у багатьох сучасних додатках, включаючи розподілені системи зберігання даних, аутентифікацію та контроль доступу.

Метою бакалаврської дипломної роботи є розробка програми розподілу секрету між групою користувачів на основі схеми Шаміра, яка забезпечить надійний та ефективний механізм захисту конфіденційної інформації.

Об'єктом дослідження є порогові криптосистеми, які використовуються для забезпечення безпеки інформації шляхом розподілу секрету між декількома користувачами.

Предметом дослідження є алгоритми та методи реалізації схеми Шаміра для розподілу секрету, а також їхнє застосування у програмному забезпеченні.

Новизна роботи полягає у розробці та впровадженні ефективного програмного забезпечення, яке реалізує схему Шаміра для розподілу секрету між групою користувачів, з урахуванням сучасних вимог до безпеки та продуктивності. В роботі буде представлено детальний аналіз ефективності та безпеки розробленої програми, а також її практичне застосування у різних сферах.

Для досягнення мети потрібно:

- Провести огляд та аналіз існуючих порогових криптосистем, зокрема схеми Шаміра.

- Розробити алгоритм для реалізації схеми Шаміра в програмному забезпеченні.
- Створити програмне забезпечення, яке реалізує алгоритм розподілу секрету на основі схеми Шаміра.
- Провести тестування та аналіз ефективності розробленої програми.
- Розглянути можливості та приклади застосування розробленої програми у реальних умовах.

Таким чином, дана робота спрямована на вирішення актуальної задачі забезпечення безпеки інформації шляхом розробки програми для розподілу секрету між групою користувачів, що базується на схемі Шаміра. Це забезпечить високий рівень захисту даних.

Розділ 1. АНАЛІЗ ІСНУЮЧИХ СХЕМ РОЗПОДІЛУ СЕКРЕТУ ТА ЙОГО ВІДНОВЛЕННЯ

Розподіл секрету є важливою задачею у сфері інформаційної безпеки, яка передбачає розділення секретної інформації на частини таким чином, щоб лише певна кількість частин могла відновити цей секрет. Ця концепція забезпечує захист даних від несанкціонованого доступу, оскільки окремі частини секрету самі по собі не розкривають ніякої інформації. У цьому розділі розглянуто основні схеми розподілу секрету та методи їх відновлення.

1.1 Аналіз та актуальність розподілу секрету

У сучасному світі безпека інформації є однією з ключових проблем, з якою стикаються різні організації та користувачі. Для забезпечення конфіденційності, цілісності та доступності даних використовуються різноманітні методи криптографії. Одним із таких методів є розподіл секрету (англ. secret sharing), який дозволяє захищати секретну інформацію шляхом її поділу на кілька частин і розподілу між учасниками. У цьому підрозділі буде розглянуто поняття розподілу секрету, його основні принципи та актуальність у контексті сучасних викликів інформаційної безпеки[1].

Розподіл секрету – це криптографічний метод, який дозволяє розподілити секретну інформацію на кілька частин (або шифротекстів) таким чином, що лише визначена кількість частин може бути використана для відновлення оригінального секрету. Така система називається пороговою схемою розподілу секрету[1].

Актуальність розподілу секрету

В умовах сучасних кіберзагроз та збільшення обсягів конфіденційної інформації, що передається та зберігається в електронному вигляді, розподіл секрету стає дедалі актуальнішим[1]. Основні причини, чому розподіл секрету є важливим:

- Захист від зловмисників: Розподіл секрету дозволяє забезпечити високий рівень захисту інформації, оскільки зловмисник, який отримав доступ до меншої кількості частин, ніж визначено пороговим значенням, не зможе відновити оригінальний секрет[1].
- Надійність зберігання даних: У розподілених системах зберігання даних, де інформація зберігається на декількох серверах, розподіл секрету забезпечує цілісність і доступність даних навіть у разі втрати або пошкодження окремих частин[1].
- Контроль доступу: Важливі рішення можуть прийматись лише за участі визначеної кількості уповноважених осіб, що знижує ризик несанкціонованого доступу до секретної інформації[1].
- Безпека виборчих систем: Розподіл секрету може використовуватись для забезпечення безпеки та конфіденційності в електронних виборчих системах, де важливо зберігати таємницю голосування та захистити результати виборів[1].

Розподіл секрету надає надійний спосіб захисту конфіденційної інформації, дозволяючи зберігати та передавати її без ризику компрометації. Порогові схеми, такі як схема Шаміра, забезпечують баланс між безпекою та доступністю, роблячи цей метод актуальним і ефективним у сучасному світі. Розробка програмного забезпечення для реалізації таких схем є важливим кроком у забезпеченні інформаційної безпеки[1].

Основні принципи розподілу секрету:

Розподіл секрету є ключовим механізмом криптографії, який забезпечує захист конфіденційної інформації шляхом її поділу на частини і розподілу між групою користувачів. Розглянемо докладно основні принципи цього методу[1].

Принцип 1: Поділ секрету

Поділ секрету означає, що вихідна секретна інформація розбивається на

n частин, кожна з яких передається окремому учаснику. Цей процес здійснюється таким чином, що кожна частина сама по собі не містить достатньо інформації для відновлення секрету. Тільки певна комбінація цих частин дозволяє відновити початкову інформацію[1].

Уявімо, що секрет — це число S. Це число розділяється на n частин S1, S2, ... , Sn які передаються різним учасникам. Жоден з учасників, який володіє лише однією частиною Si, не зможе дізнатися оригінальний секрет S без інших частин[1].

Принцип 1: Порогове значення

Порогове значення t визначає мінімальну кількість частин, необхідних для відновлення оригінального секрету. Це значення вибирається заздалегідь і забезпечує баланс між безпекою та доступністю[1].

Тобто, Якщо вибрано t як порогове значення, то для відновлення секрету потрібно мати щонайменше t частин. Будь-яка кількість частин, менша ніж t, не дозволить відновити секрет. Це забезпечує високий рівень безпеки, оскільки зловмисник, який має доступ лише до t−1 частин, не зможе дізнатися секрет[1].

Принцип 2: Відновлення секрету

Відновлення секрету здійснюється за допомогою об'єднання t або більше частин. Цей процес зазвичай базується на математичних методах, таких як інтерполяція поліномів, у разі схеми Шаміра, або геометричних методах, як у схемі Блеклі. У випадку схеми Шаміра, секрет може бути представлений як значення полінома в нульовій точці. Кожна частина секрету є значенням цього полінома в певній точці. Для відновлення секрету необхідно мати щонайменше t таких значень, після чого можна відновити поліном і обчислити значення в нульовій точці[1].

1.2 Аналіз існуючих систем розподілу секрету

В більшості випадків для вирішення проблеми розподілу секрету між групою користувачів використовують 3 схеми.[1]

- Схема Блеклі (Blakley's Secret Sharing)
- Схема Асада і Ітамара (Asaad and Itamar's Secret Sharing)
- Схема Шаміра (Shamir's Secret Sharing)

Першою є Схема Блеклі, вона використовує геометричний підхід до розподілу секрету, заснований на властивостях перетину гіперплощин у багатовимірному просторі[2].

Основні принципи схеми Блеклі:

Схема Блеклі заснована на використанні геометричних властивостей площин у багатовимірному просторі[2].

Параметри схеми:

- k: кількість частин секрету, необхідних для його відновлення (поріг).
- n: загальна кількість частин, на які розбивається секрет.

Геометричний підхід:

- Секрет представляється як координати точки у (k−1)-вимірному просторі.
- Для кожного учасника генерується гіперплощина у цьому просторі, причому всі ці гіперплощини перетинаються в одній точці, що відповідає секрету.

Розподіл частин секрету:

- Генерується довільна точка S у $k-1$ вимірному просторі, яка є секретом.
- Кожному з n учасників видається гіперплощина, що проходить через точку S.
- Кожна гіперплощина визначається лінійним рівнянням.

Відновлення секрету

Для відновлення секрету потрібно як мінімум kучасників, оскільки k гіперплощин у (k−1)-вимірному просторі перетинаються лише в одній точці. Ця точка і є шуканим секретом. Приклад схеми Блекли наведено на (рисунок 1.1)[2]

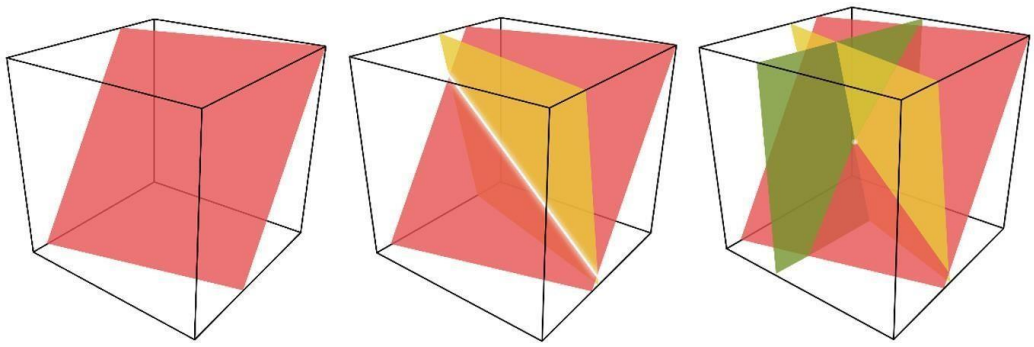


Рисунок 1.1 – Графічний приклад реалізації схеми Блекли в трьох вимірах

Приклад:

Розглянемо простий приклад для схеми з порогом k=2 у двовимірному просторі[2].

Припустимо, що Секрет S представляється точкою(x_0, y_0) на площині. Двом учасникам A та B видаються рівняння прямих, що проходять через точку[2]

A отримує рівняння прямої $y = m_1x + c_1$

B отримує рівняння прямої $y = m_2x + c_2$

Відновлення секрету:

Учасники A та B діляться своїми рівняннями. Точка перетину цих двох прямих(x_0, y_0) і є секретом[2].

Схема Блеклі є важливим інструментом у криптографії, зокрема у ситуаціях, де потрібно забезпечити спільний доступ до конфіденційної інформації за умов суворого дотримання безпеки[2].

Другою схемою для аналізу була **схема розподілу секрету Асада і Ітамара**, вона базується на використанні лінійних перетворень та інтерполяції поліномів. Основною ідеєю є представлення секрету у вигляді вектора та його розподіл між учасниками за допомогою матриць. Для кращого розуміння схему було розділено на 4 етапи[2].

1. Генерація секрету

Нехай секрет S представляється як вектор у просторі Rn. Для простоти, будемо вважати, що секрет S є скалярним значенням. Далі, використовуючи секрет, генеруємо поліном ступеня $k-1$:

$$P(x) = S + a_1x + a_2x^2 + \cdots + a_{k-1}x^{k-1}$$

Де $a_1, a_2, \ldots, a_{k-1}$ є випадковими коефіцієнтами.

2. Створення часток

Для створення часток вибираємо n різних значень $x_1, x_2, \ldots, x_n$ Кожному учаснику і призначається частка $(x_i, P(x_i))$.Таким чином, кожен учасник отримує пару значень: x_i та $P(x_i)$

3. Розподіл часток

Частки розподіляються між учасниками. Кожен учасник отримує своє унікальне значення $P(x_i)$ що є обчисленим значенням полінома в точці x_i

4. Відновлення секрету

Для відновлення секрету необхідно зібрати k або більше часток. Припустимо, що зібрано k часток:$(x_1, P(x_1))$, $(x_2, P(x_2))$, ... , $(x_k, P(x_k))$. Використовуючи ці частки, можна відновити поліном P(x) за допомогою інтерполяції Лагранжа:

$$P(x) = \sum_{i=1}^{k} P(x_i) \prod_{\substack{1 \le j \le k \\ j \ne i}} \frac{x - x_j}{x_i - x_j}$$

Після відновлення полінома $P(x)$, секретом S буде його значення в точці x=0:

$$S = P(0)$$

Приклад:

Розглянемо приклад для секрету S=123 та порогу k=3. Генеруємо випадкові коефіцієнти a_1=45 та a_2=67. Поліном виглядатиме наступним чином:

$$P(x) = 123 + 45x + 67x^2$$

Вибираємо значення x: $x_1 = 1, x_2 = 2 x_3 = 3, x_4 = 4$ Обчислюємо частки:

$$P(1) = 123 + 45 \cdot 1 + 67 \cdot 1^2 = 235$$

$$P(2) = 123 + 45 \cdot 2 + 67 \cdot 2^2 = 469$$

$$P(3) = 123 + 45 \cdot 3 + 67 \cdot 3^2 = 825$$

$$P(4) = 123 + 45 \cdot 4 + 67 \cdot 4^2 = 1303$$

Розподіляємо частки: (1, 235), (2, 469), (3, 825), (4, 1303).

Для відновлення секрету зберемо три частки, наприклад, (1,235),(2,469),(3,825), та виконаємо інтерполяцію Лагранжа для відновлення полінома:

$$P(x) = 235\frac{(x-2)(x-3)}{(1-2)(1-3)} + 469\frac{(x-1)(x-3)}{(2-1)(2-3)} + 825\frac{(x-1)(x-2)}{(3-1)(3-2)}$$

Відновлений поліном P(x) дозволяє обчислити секрет S=P(0).

P(0)=705−1407+825=705+825−1407=1530−1407=123

Таким чином, секрет S дорівнює 123, що співпадає з початковим значенням секрету.

Третьою схемою для аналізу була **схема Шамира**

Схема інтерполяції поліномів Лагранжа (також відома як схема поділу секрету Шаміра) – це метод розподілу секрету, який широко використовується в криптографії. Вона дозволяє здійснювати (k,n) - пороговий поділ секретного повідомлення між сторонами таким чином, щоб будь-які k і більше сторін (де k≤n) могли відновити секрет. При цьому будь-які k−1 або менше сторін не зможуть відновити секрет[3].

Для інтерполяції багаточлена ступеня k−1 потрібно k точок. Наприклад, для знаходження прямої достатньо двох точок, для знаходження параболи[3] – трьох точок як зображено на рисунку 1.2.

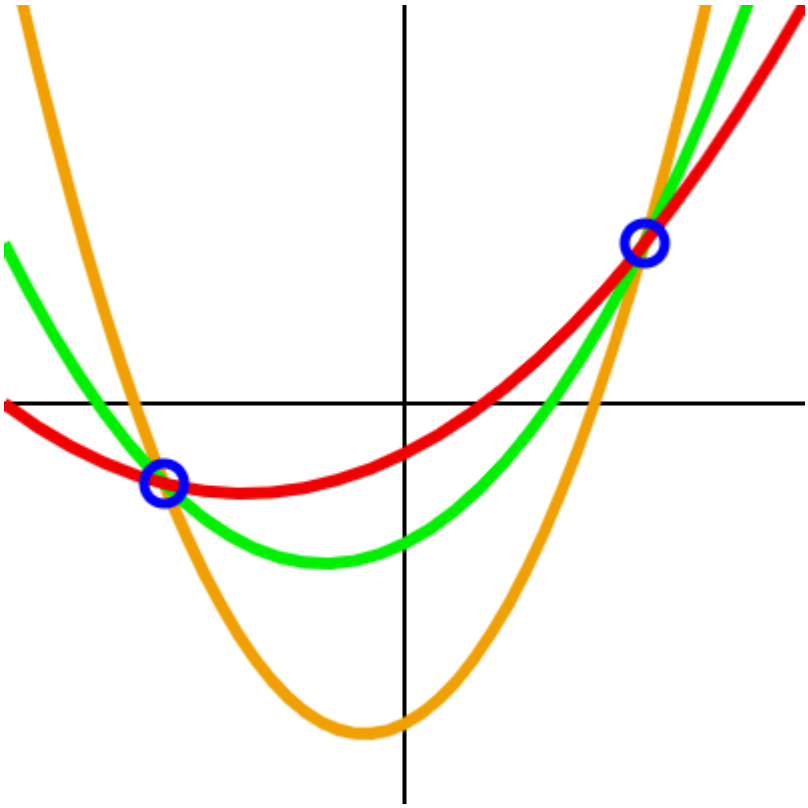


Рисунок 1.2 Ілюстрація ідеї Полімерної Інтерполяції порогової схеми Шаміра.

Основна ідея цієї схеми полягає в тому, що інтерполяція неможлива, якщо відомо менше k точок[3].

Якщо потрібно розділити секрет між n людьми так, щоб відновити його могли лише ті, у кого є k або більше частин секрету, то ми формуємо багаточлен ступеня k−1. Відновити цей багаточлен і вихідний секрет можна тільки за k точками. Кількість різних точок множини не обмежена (на практиці воно обмежується розміром числового поля, в якому ведуться розрахунки)[3].

Порогова схема Шаміра (k,n) базується на концепції поліноміальної інтерполяції. Якщо ви не знайомі з цією концепцією, вона насправді є досить простою[3].

Приклад створення

Щоб зрозуміти порогову схему Шаміра, уявімо, що наш секрет S дорівнює 42. Ми можемо перетворити S на точку на графіку (0,42) та створити

поліноміальну функцію ступеня k−1, яка проходить через цю точку. Зазначимо, що k - це мінімальна кількість частин секрету, необхідна для його відновлення. Наприклад, якщо встановити поріг k=3, то ми обираємо поліном ступеня 2 [3].

Поліном виглядає так: $f(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{k-1}x^{k-1}$, де $a_0 = S$, та $a_1, \ldots, a_k$ - випадково обрані позитивні цілі числа. Ми шукатимемо поліном ступеня k−1, де вільний коефіцієнт a_0 є нашим секретом S, а кожен з наступних k−1 коефіцієнтів обирається випадковим чином серед позитивних цілих чисел [3].

Припустимо, що S=42, k=3, а a_1 і a_2 обрані як 3 та 5 відповідно. Тоді поліномічна функція матиме вигляд $f(x) = 42 + 3x + 5x^2$[3].

На цьому етапі можна згенерувати унікальні фрагменти, підставляючи цілі значення x у функцію $f(x) = 42 + 3x + 5x^2$. Для x=0 (наш секрет), і для інших значень x=1,2,3 обчислюються значення функції, що стають фрагментами секрету[3].

Припустимо, ми хочемо розподілити чотири фрагменти з порогом три. Тому випадково генеруємо точки (18, 1716), (27, 3768), (31, 4940) та (35, 6272) і передаємо їх чотирьом довіреним особам, зберігаючи ключ. Також повідомляємо людям, що k=3, тобто потрібно три фрагменти для відновлення секрету[3].

Приклад відновлення

Основою порогової схеми Шаміра (k,n) є концепція поліноміальної інтерполяції. Коли три з чотирьох довірених осіб хочуть відновити секрет S, їм потрібно інтерполювати f(x) за допомогою їхніх унікальних точок. Відомі точки $(x_1, y_2), (x_2, y_2), \ldots, (x_k, y_k)$ дозволяють визначити свої координати та обчислити інтерполяційний поліном Лагранжа, використовуючи формулу [2]:

$$P(x) = \sum_{j=1}^{k} P_j(x)$$

Де

$$P_j(x) = y_j \prod_{\substack{m=1 \\ m \neq j}}^{k} \frac{x - x_m}{x_j - x_m}$$

При k=3 можемо вирішити це та відновити наш вихідний поліном:

$$P(x) = y_1 \left(\frac{x - x_2}{x_1 - x_2} \cdot \frac{x - x_3}{x_1 - x_3} \right) + y_2 \left(\frac{x - x_1}{x_2 - x_1} \cdot \frac{x - x_3}{x_2 - x_3} \right) + y_3 \left(\frac{x - x_1}{x_3 - x_1} \cdot \frac{x - x_2}{x_3 - x_2} \right)$$

Знаючи точки:

$$P(x) = 1716 \left(\frac{x - 27}{18 - 27} \cdot \frac{x - 31}{18 - 31} \right) + 3768 \left(\frac{x - 18}{27 - 18} \cdot \frac{x - 31}{27 - 31} \right) + 4940 \left(\frac{x - 18}{31 - 18} \cdot \frac{x - 27}{31 - 27} \right)$$

Отримуємо:

$$P(x) = 42 + 3x + 5x^2$$

Оскільки P(0)=S=42, відновлення S:

$$P(0) = 42 + 3(0) + 5(0)^2$$

$$P(0) = 42 \text{ [2]}$$

Для відновлення секрету потрібно хоча б 3 фрагменти, використовуючи інтерполяцію Лагранжа для відновлення початкового полінома та отримання секрету S=42S = 42 [2].

1.3 Аналіз криптосистем с пороговим розподілом секрету засновані на китайскій теоремі про остачі

Криптосистеми з пороговим розподілом секрету, засновані на китайській теоремі про остачі (КТО), є одним з підходів до реалізації безпечного розподілу секретної інформації серед групи учасників. Основна ідея полягає в тому, що секрет ділиться на частини таким чином, що лише певна кількість (поріг) цих частин дозволяє відновити початковий секрет[23]. Ось як це працює:

КТО стверджує, що для будь-якої системи конгруенцій виду:

$$x \equiv a_1 \pmod{m_1}$$

$$x \equiv a_2 \pmod{m_2}$$
$$\vdots$$
$$x \equiv a_k \pmod{m_k}$$

Де $m_1, m_2, \ldots, m_k$ взаємно прості числа, існує єдине рішення x в межах модуля $M = m_1 \cdot m_2 \cdot \ldots \cdot m_k$ яке задовольняє всі ці конгруенції.

Порогова криптосистема t-з-п-n означає, що для відновлення секрету необхідно принаймні t частин з n можливих. Жодна менша кількість частин не дозволить відновити секрет[23].

Використання КТО для порогового розподілу секрету

- Нехай секретом є число S.
- Обираються взаємно прості числа $m_1, m_2, \ldots, m_n$ так, щоб кожне з них було достатньо великим.
- Для кожного учасника i (де i від 1 до n) обчислюється частка секрету S_i На m_i тобто:

$$S_i = S \pmod{m_i}$$

- Кожному учаснику i передається пара(S_i, m_i).

Відновлення секрету:

- Для відновлення секрету зібрані принаймні t часток:

$$(S_{i_1}, m_{i_1}), (S_{i_2}, m_{i_2}), \ldots, (S_{i_t}, m_{i_t})$$

- Використання КТО для розв'язання системи конгруенцій:

$$x \equiv S_{i_1} (\mathrm{mod} m_{i_1})$$
$$x \equiv S_{i_2} (\mathrm{mod} m_{i_2})$$
$$\vdots$$
$$x \equiv S_{i_t} (\mathrm{mod} m_{i_t})$$

- Єдине рішення x цієї системи (в межах модуля $M = m_{i_1} \cdot m_{i_2} \cdot \ldots \cdot m_{i_t}$) є відновленим секретом S.

Переваги та недоліки

Переваги:

До переваг можна віднести високий рівень безпеки: знання менш ніж t часток не дає жодної інформації про секрет, а токож ефективність у відновленні секрету за допомогою КТО[23].

Недоліки:

Недоліками може бути вибір взаємно простих чисел може бути складним для великої кількості учасників, а також може вимагати великого обсягу пам'яті для зберігання часток, якщо дуже великі[23].

Таким чином, криптосистеми з пороговим розподілом секрету, засновані на китайській теоремі про остачі, є потужним інструментом для забезпечення безпеки секретної інформації в різних системах, де потрібно розподілити довіру серед багатьох учасників. На даний час існують всього дві основні схеми які використовують КТО[23], це:

- Схема Миньотта
- Схема Асмута та Блума

Для кращого розуміння як працює схема Міньотта було наведено приклад використання нище.

Припустимо, що секрет S=123456.

Для 5 учасників (n=5) з порогом 3 (t=3) обираємо числа m1=101, m2=103, m3=107, m4=109, m5 =113.

Обчислюються частки секрету для кожного числа:

$$S1 = 12345(mod\ 101) = 90$$
$$S2 = 12345(mod\ 103) = 12$$
$$S3 = 12345(mod\ 107) = 43$$
$$S4 = 12345(mod\ 109) = 14$$
$$S5 = 12345(mod\ 113) = 47$$

Уасники отримують свої частки у вигляді пар (Si,mi).

Відновлення секрету:

Зібрані частки від трьох учасників, наприклад, 1, 2 і 3: (90,101),(12,103),(43,107).

Використовується КТО для розв'язання системи конгруенцій:

$$x \equiv 90 \ (mod\ 101)$$

$$x \equiv 12 \ (mod\ 103)$$

$$x \equiv 43 \ (mod\ 107)$$

Розв'язок цієї системи дає відновлений секрет S=123456

Переваги та недоліки схеми Міньотта

Переваги:

- Безпека: Знання менш ніж t часток не дає жодної інформації про секрет.
- Простота: Відновлення секрету базується на ефективному алгоритмі КТО.

Недоліки:

- Вибір чисел: Потреба у великих взаємно простих числах може ускладнювати реалізацію для великої кількості учасників.
- Обчислювальні ресурси: Великі числа можуть вимагати значних обчислювальних ресурсів для відновлення секрету.

Схема Міньотта є ефективним і надійним методом порогового розділення секрету, який забезпечує високу безпеку та простоту використання завдяки китайській теоремі про остачі[23].

Також наведемо приклад реалізації схеми Асмута та Блума

Припустимо, секрет S=123, поріг t=3, і p=101.

Обираємо числа $m1$=2,$m2$=3, $m3$=5, $m4$=7

Розподіл секрету**:**

Випадкове число r обирається, наприклад, r=10.

Обчислюється доповнений секрет:

$$S = S + p \cdot r = 123 + 101 \cdot 10 = 1133$$

Обчислюються частки для кожного числа mi:

$$x \equiv 1 \ (mod\ 2)$$

$$x \equiv 2 \ (mod\ 3)$$

$$x \equiv 3 \ (mod\ 5)$$

Розв'язок: x=23 відповідає значенню S′ модулю $\prod_{i=1}^{3} mi = 30$ Оскільки x=1133:

$$1133 \ (mod\ 30) = 23$$

Остаточний секрет відновлюється як:

$$\mathrm{S} = 1133 \ (\mathrm{mod}\ 101) = 123$$

Переваги та недоліки схеми Асмута та Блума:

Переваги:

- Безпека: Знання менш ніж t часток не дає жодної інформації про секрет.
- Гнучкість: Схема легко масштабується для різної кількості учасників і порогів.

Недоліки:

- Обчислювальні ресурси: Використання великих чисел може вимагати значних обчислювальних ресурсів.
- Складність вибору чисел: Потребує обережного вибору чисел, щоб забезпечити виконання умов для КТО.

Схема Асмута та Блума є потужним методом порогового розділення секрету, який забезпечує високу безпеку та коректність відновлення секрету завдяки використанню китайської теореми про остачі[23].

1.4 Висновки та постановка задач

У першому розділі було розглянуто основні аспекти розподілу секрету, включаючи поняття, принципи та актуальність цього методу в сучасних умовах.

Найпоширеніші схеми розподілу секрету включають схему Шаміра та схему Блеклі. Схема Шаміра базується на інтерполяції поліномів і забезпечує високий рівень безпеки та гнучкість у налаштуванні порогових значень. Схема Блеклі базується на геометричному підході. Розподіл секрету є важливим у контексті сучасних викликів інформаційної безпеки. Він забезпечує надійний захист конфіденційної інформації, підвищує надійність зберігання даних у розподілених системах, допомагає в аутентифікації та контролі доступу, а також у захисті виборчих систем.

На основі проведеного аналізу можна сформулювати основні задачі, які необхідно вирішити у рамках роботи:

- Дослідження та аналіз: Провести детальне дослідження існуючих порогових криптосистем, зокрема схеми Шаміра, та їх застосування для розподілу секрету.
- Розробка алгоритму: Розробити алгоритм для реалізації схеми Шаміра, що дозволить безпечно розподіляти секретну інформацію між групою користувачів і відновлювати її за необхідності.
- Розробка програмного забезпечення: Створити програмне забезпечення, яке реалізує розроблений алгоритм розподілу секрету. Програма має бути зручною у використанні, забезпечувати високий рівень безпеки та гнучкість у налаштуванні параметрів.
- Тестування та аналіз ефективності: Провести тестування розробленої програми у різних умовах використання та проаналізувати її ефективність. Виявити можливі недоліки та способи їх усунення.

Таким чином, дана робота спрямована на вирішення актуальної задачі забезпечення інформаційної безпеки шляхом розробки ефективної та надійної системи розподілу секрету на основі схеми Шаміра.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СХЕМИ РОЗПОДІЛУ СЕКРЕТУ ТА ЙОГО ВІДНОВЛЕННЯ

Надалі розділ буде присвячено проектуванню схеми розподілу секрету та його відновлення. Метою цього розділу є опис принципів роботи схеми Шаміра, яка була обрана для реалізації програми розподілу секрету між групою користувачів,а також представлення алгоритмів та методів, що використовуються для розробкипрограмного забезпечення.

2.1 Схема розподілу та відновлення секрету.

Поділ секрету (англ. secret sharing) - термін у криптографії, під яким розуміють будь-який із способів розподілу секрету серед групи учасників, кожному з яких дістається своя частка. Секрет може відтворити тільки коаліція учасників з початкової групи, причому входити в коаліцію має не менше деякого відомого їх числа.

Схеми поділу секрету застосовуються у випадках, коли існує значна ймовірність компрометації одного або декількох зберігачів секрету, але ймовірність недобросовісної змови значної частини учасників вважається дуже малою.

Існуючі схеми мають дві складові: поділ та відновлення секрету. До поділу відноситься формування частин секрету та розподіл їх між членами групи, що дозволяє розділити відповідальність за секрет між її учасниками. Зворотна схема має забезпечити його відновлення за умови доступності його зберігачів у певній необхідній кількості.

Нехай потрібно розділити секрет M між n сторонами таким чином, щоб будь-які k учасників могли б відновити секрет (тобто потрібно реалізувати (k,n) порогову схему).

Виберемо деяке просте число $p>M$. Це число можна відкрито повідомляти всіх учасників. Воно задає кінцеве поле розміру p. Над цим полем побудуємо багаточлен ступеня $k-1$(тобто випадково виберемо всі коефіцієнти багаточлена, крім M:

$$F(x) = \left(a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \cdots + a_1 x + M\right) \mod p$$

У цьому багаточлені M - це секрет, що розділяється, а решта коефіцієнтів $a_{k-1}, a_{k-2}, \ldots, a_1$ — деякі випадкові числа, які потрібно буде «забути» після того, як процедура поділу секрету буде завершена.

Генерація долей секрету

Тепер обчислюємо «частки» - значення побудованого вище багаточлена, в різних точках, причому ($x \neq 0$):

$$\begin{aligned}
k_1 &= F(1) = \left(a_{k-1}\cdot 1^{k-1} + a_{k-2}\cdot 1^{k-2} + \cdots + a_1\cdot 1 + M\right) \mod p \\
k_2 &= F(2) = \left(a_{k-1}\cdot 2^{k-1} + a_{k-2}\cdot 2^{k-2} + \cdots + a_1\cdot 2 + M\right) \mod p \\
&\cdots \\
k_i &= F(i) = \left(a_{k-1}\cdot i^{k-1} + a_{k-2}\cdot i^{k-2} + \cdots + a_1\cdot i + M\right) \mod p \\
&\cdots \\
k_n &= F(n) = \left(a_{k-1}\cdot n^{k-1} + a_{k-2}\cdot n^{k-2} + \cdots + a_1\cdot n + M\right) \mod p
\end{aligned}$$

Аргументи багаточлена (номери секретів) не обов'язково повинні йти по порядку, головне — щоб усі вони були різні за модулем p

Після цього кожній стороні, що бере участь у розділенні секрету, видається частка секрету ki разом з номером i Крім цього, всім сторонам повідомляється ступінь багаточлена $k-1$ та розмір поля p. Випадкові коефіцієнти $a_{k-1}, a_{k-2}, \ldots, a_1$ і сам секрет M.

Відновлення секрету

Тепер будь-які k учасників, знаючи координати k різних точок багаточлена, зможуть відновити багаточлен і всі його коефіцієнти, включаючи останній з них - секрет, що розділяється.

Особливістю схеми є те, що ймовірність розкриття секрету у разі довільних ?часть оцінюється як ?. Тобто в результаті інтерполяції по точці $k-1$ секретом може бути будь-який елемент поля з рівною ймовірністю. При цьому спроба повного перебору всіх можливих тіней не дозволить зловмисникам отримати додаткову інформацію про секрет.

Прямолінійне відновлення коефіцієнтів многочлена через рішення системи рівнянь можна замінити на обчислення інтерполяційного багаточлена Лагранжа (звідси одна з назв методу). Формула многочлена виглядатиме так:

$$F(x) = \sum_i l_i(x)\, y_i \mod p$$

$$l_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} \mod p$$

де (xi,yi) - координати точок багаточлена. Усі операції виконуються також у кінцевому полі p.

Розглянемо простий випадок, коли відновлення секрету необхідно дві тіні. Багаточлен, у цьому випадку, задаватиме пряму, що перетинається з віссю k у точці S(секрет). Кожна частка – точка на прямій. Секрет може бути відновлений за двома довільними тінями. Однак, у разі завдання лише однієї тіні як шуканий секрет може бути обрана будь-яка точка на осі k, тому що через одну точку можна провести безліч різних прямих, що перетинаються з віссю k в довільних точках.Ілюстрацію циз точок зображено на (рис 2.1)

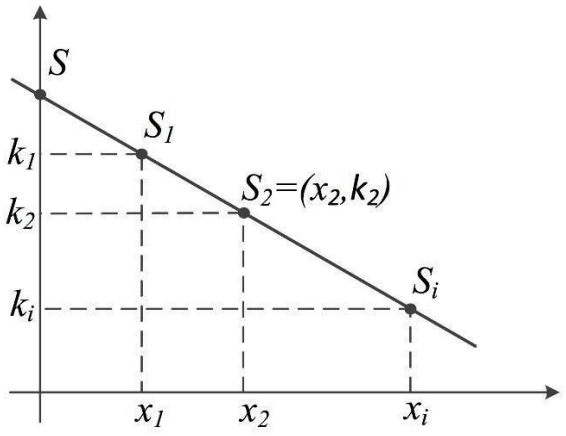


Рисунок 2.1 – Ілюстрація схеми розподілу секрету Шамира

До переваг цієї схеми поділу секрету відносять[1]:

Ідеальність: відсутня надмірність - розмір кожної з часток секрету дорівнює розміру секрету.

Масштабованість: в умовах схеми (k,n) число власників часток секрету nможе додатково збільшитися аж до p, де p - розмір поля, в якому ведуться обчислення. При цьому кількість часток, необхідних для отримання секрету, залишиться незмінною.

Динамічність: можна періодично змінювати багаточлен, що використовується, і перераховувати тіні, зберігаючи секрет (вільний член) незмінним. При цьому ймовірність порушення захисту шляхом витоку тіней зменшиться, тому що для отримання секрету потрібно тіней, отриманих на одній версії багаточлена.

Гнучкість: у випадках, коли боку є рівними між собою, схема дозволяє це врахувати шляхом видачі відразу кількох тіней одній стороні. Наприклад, пусковий код балістичної ракети може бути розділений за схемою (3,5) так, щоб ракету могли запустити лише три генерали, які зберуться разом, або будь-який з генералів і президент, якому під час поділу секрету було видано одразу дві тіні.

2.2 Розробка алгоритму програми для розподілення секрету між групою користувачів

Відповідно до проаналізованої інформації в попередніх підрозділах, буде розроблено структурний алгоритм роботи схеми. Буде розібрано покроково кожний етап процедури. Для початку було створено схематичний алгоритм роботи алгоритму схеми Шамира. (рис 2.2)

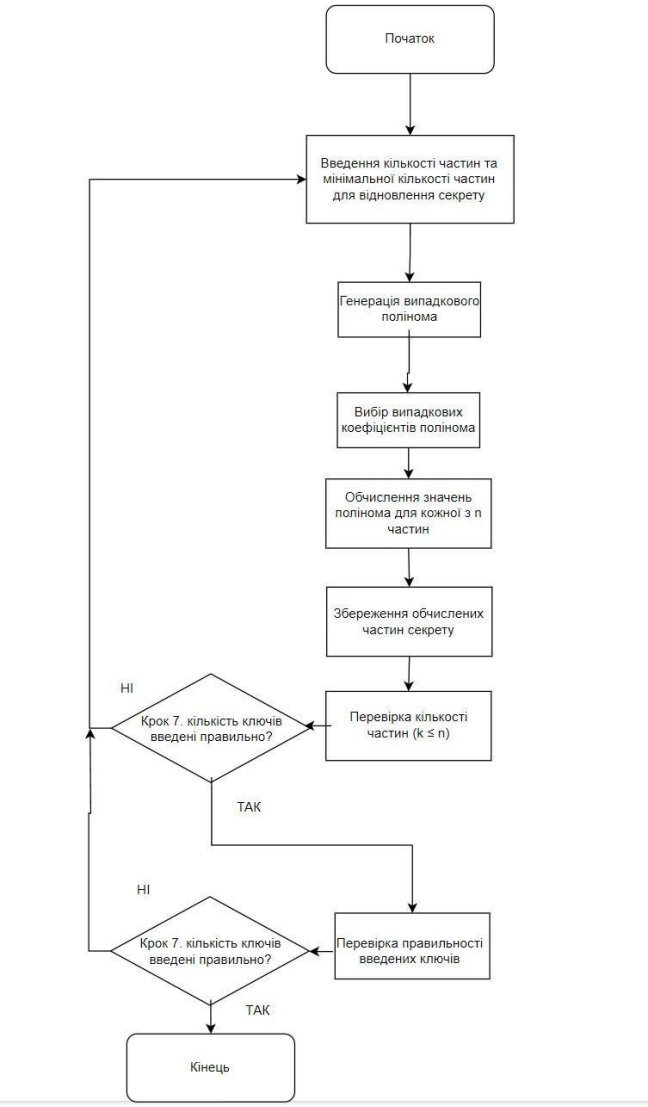


Рисунок 2.3 – Детальний алгоритм роботи схеми Шамира

Алгоритм розподілу секрету за допомогою схеми Шаміра:

Крок 1. Початок:

Процес розподілу секрету починається з ініціації програми. Це стартовий крок, де користувач готується до введення необхідних параметрів.

Крок 2. Введення параметрів (n, k):

Користувач вводить кількість частин (n) та мінімальну кількість частин (k), необхідних для відновлення секрету. Це ключові параметри, які визначають, як буде здійснюватися розподіл секрету та його подальше відновлення.

Крок 3. Генерація полінома:

Генерується випадковий поліном ступеня k−1, де секрет є вільним членом цього полінома. Це основний математичний крок, який забезпечує безпеку алгоритму.

Крок 4. Вибір коефіцієнтів:

Для полінома вибираються випадкові коефіцієнти. Ці коефіцієнти визначають поведінку полінома та впливають на значення частин секрету.

Крок 5. Обчислення значень:

Обчислюються значення полінома для кожної з n частин. Це створює конкретні частини секрету, які потім будуть розподілені між учасниками.

Крок 6. Збереження частин:

Обчислені частини секрету зберігаються. Це дозволяє зберегти розподілені частини для подальшого використання.

Крок 7. Перевірка кількості частин ($k \leq n$):

Виконується перевірка, чи кількість введених частин (n) є достатньою для відновлення секрету ($k \leq n$). Це важливий крок, який забезпечує правильність і можливість відновлення секрету.

Крок 7.1. НІ:

Якщо $k > n$, система виводить повідомлення про помилку. Користувач отримує інструкцію повернутися до кроку 2 для введення коректних параметрів.

Крок 7.2. ТАК:

Якщо $k \leq n$, система переходить до наступного кроку, забезпечуючи можливість продовження процесу.

Крок 8. Відновлення секрету:

Використовуючи необхідну кількість частин, система відновлює секрет. Це головна мета алгоритму - забезпечити можливість відновлення секрету при наявності достатньої кількості частин.

Крок 9. Перевірка правильності відновлення:

Перевіряється правильність відновленого секрету. Цей крок гарантує, що відновлений секрет є правильним і відповідає початковому секрету.

Крок 9.1. НІ:

Якщо секрет не відновлено правильно, система виводить повідомлення про помилку і повертається до кроку 8 для повторної спроби.

Крок 9.2. ТАК:

Якщо секрет відновлено правильно, система переходить до наступного кроку.

Крок 10. Завершення:

Завершення процесу розподілу секрету. Програма успішно завершує свою роботу, забезпечуючи успішний розподіл і можливість відновлення секрету.

Алгоритм розподілу секрету за допомогою схеми Шаміра є ретельно продуманим і структурованим процесом, який забезпечує високий рівень безпеки і надійності. Використання математичних основ, таких як поліноми і випадкові коефіцієнти, гарантує, що секрет може бути відновлений тільки за наявності достатньої кількості частин, що забезпечує захист від несанкціонованого доступу. Перевірки та можливість повернення до попередніх кроків забезпечують гнучкість і надійність алгоритму.

2.3 Розробка алгоритму відновлення секрету

Після розподілення та шифрування секретного вмісту отримується інформація, яка не підлягає прочитанню чи взаємодії. Проте власник інформації повинен мати можливість відновити цілісність даних або їх редагувати. Для цього

пропонується використання алгоритму відновлення інформації, зображеного на (рис. 2.2).

З цих даних зрозуміло, що зловмисник, маючи лише один файл, не зможе отримати ніякої корисної інформації, оскільки файл залишається непридатним для читання без відновлення. Те ж саме стосується ситуації, коли зловмисник володіє всіма файлами одночасно: без відновлення вони не містять цінної інформації, оскільки файл залишається побітно розбитим.

Для підвищення безпеки зберігання та ускладнення несанкціонованого відновлення рекомендовано зберігати засіб для відновлення на окремому пристрої. Ідеально виконувати розбивання секрету на комп'ютері, що не підключений до мережі. Це вирішить ряд питань з безпеки при зберіганні інформації та забезпечить додатковий фізичний захист, оскільки зловмиснику потрібно буде мати безпосередній доступ до носія інформації та програмного забезпечення, яке зберігатиметься окремо або самознищуватиметься після виконання своїх завдань.

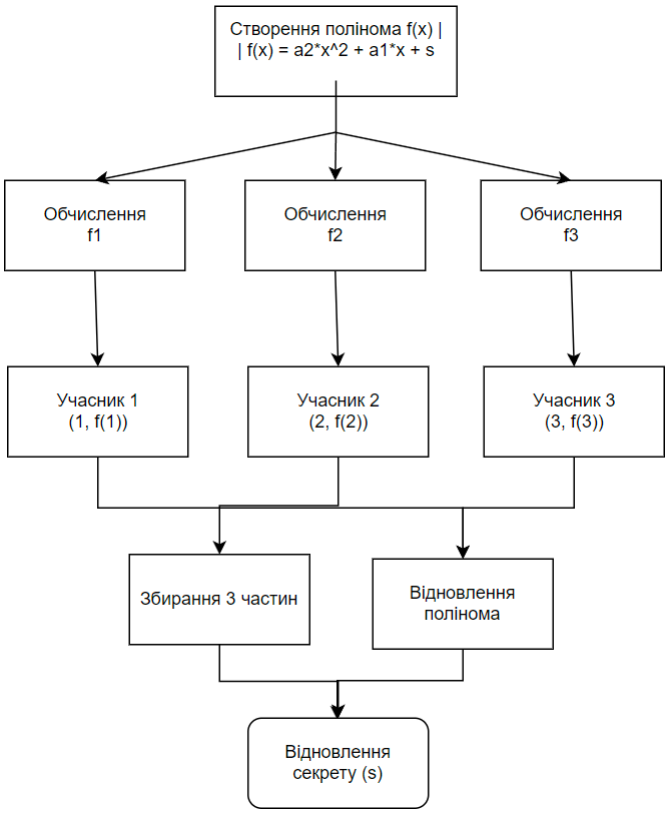


Рисунок 2.2 – Ілюстрація схеми порогового розподілу секрету Шаміра

Ця схема описує процес створення, розподілу та відновлення секрету за допомогою поліномів. Ось кроки цієї схеми:

Створення полінома f(x): Спочатку генерується поліном f(x), де $a2$, $a1$ - коефіцієнти полінома, s - секретне значення, яке потрібно розподілити.

Обчислення значень f(x) для різних точок x: Значення полінома f(x) обчислюються для декількох точок x.

Розподіл значень f(x) між учасниками: Кожному учаснику надається пара значень (x, f(x)), де x - це точка, а f(x) - відповідне значення полінома у цій точці.

Збирання частин секрету: Певна кількість учасників обмінюються своїми частинами (x, f(x)), щоб отримати досить інформації для відновлення секретного значення.

Відновлення полінома f(x): Зібрані частини використовуються для відновлення початкового полінома f(x).

Відновлення секрету s: Одним із варіантів використання відновленого полінома є отримання початкового секретного значення s.

Ця схема може бути використана для розподілу секретних даних між декількома учасниками таким чином, щоб для відновлення секрету потрібно було об'єднати частини, які у кожного учасника.

2.4 Висновки до розділу 2

Цей розділ включає розглянуто схему розподілу та відновлення секрету за допомогою методу поділу секрету, зокрема за схемою Шаміра. Ця схема дозволяє розділити секрет між групою учасників таким чином, що для його відновлення необхідно зібрати мінімальну кількість з них.

Перш ніж виконувати розподіл секрету, необхідно зазначити параметри схеми, такі як загальна кількість учасників (n) і мінімальна кількість, необхідних для відновлення секрету (k). Потім генерується випадковий поліном ступеня k-1, де секрет є вільним членом, і визначаються випадкові коефіцієнти. Значення полінома обчислюються для кожної частини секрету, які потім розподіляються між учасниками.

Крім того, було розроблено алгоритм програми для розподілення секрету між групою користувачів та алгоритм відновлення секрету. Ці алгоритми дозволяють ефективно виконувати процес розподілу та відновлення секрету, забезпечуючи безпеку та правильність процедур.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ ТА ЙОГО ТЕСТУВАННЯ

У цьому розділі буде розглянуто процес розробки програмного застосунку, призначеного для реалізації схеми розподілу секрету Шаміра, а також його тестування для забезпечення коректності роботи. Розробка програмного застосунку є важливим етапом, який включає в себе вибір засобів програмування, середовища розробки та мови програмування. Цей процес потребує ретельного планування і аналізу, щоб забезпечити ефективну, надійну та зручну у використанні програму.

3.1 Вибір засобів програмування, середовища розробки та мови програмування

При виборі засобів програмування для розробки сучасних програмних продуктів важливо враховувати різні аспекти, такі як функціональність, зручність використання, підтримка різних мов програмування, інтеграція з іншими інструментами та спільнотою розробників. Одним з найбільш популярних та універсальних інструментів на сьогодні є Visual Studio Code (VS Code).

Visual Studio Code (VS Code) — це безкоштовний редактор вихідного коду, розроблений компанією Microsoft. Він став дуже популярним серед розробників завдяки своїй гнучкості, швидкодії та широким можливостям налаштування. До переваг можна віднести[12]:

VS Code відомий своєю високою швидкодією та низьким споживанням ресурсів, що дозволяє його використовувати навіть на менш потужних машинах[12].

Редактор доступний на різних операційних системах, включаючи Windows, macOS та Linux. Це дозволяє розробникам використовувати один і той же

інструмент на різних платформах без необхідності адаптації до нових середовищ[12].

VS Code має потужну інтеграцію з різними хмарними сервісами, такими як Azure, що дозволяє розробникам легко розгортати свої проекти та управляти ними безпосередньо з редактор.

Visual Studio Code підтримує різні мови програмування, такі як Java, Python та інші. Крім того, існує можливість додавати розширення для редактора, наприклад, інструменти для налагодження коду, засоби для роботи в хмарі та веб-розробки[12].

Інтуїтивно зрозумілий інтерфейс редактора також є значною перевагою. Завдяки чіткій структурі та організації, програмісти знаходять його зручним для написання зрозумілого коду та виправлення помилок.

Розширення та плагіни: Можливість додавати функціональність через велику кількість розширень.

Інтеграція з Git: Вбудована підтримка Git для контролю версій.

Автодоповнення коду: Інтелектуальне автодоповнення, яке підвищує продуктивність.Вбудований термінал: Дозволяє запускати команди безпосередньо з редактора.

З середовищем все зрозуміло, наступним кроком було вибрано могу програмування яка найбільше підходить для реалізування програми розподілу секрету між користувачами. В даному випадку вибір пав на мову JavaScript.

JavaScript - це динамічна мова програмування, яка в основному використовується для створення інтерактивних веб-сторінок і веб-додатків. Вона була створена Бренданом Ейхом у 1995 році і стала однією з основних технологій веб-розробки поряд з HTML та CSS[6].

Основні особливості JavaScript

Динамічна типізація: JavaScript використовує динамічну типізацію, що означає, що типи змінних визначаються під час виконання програми, а не під час

компіляції. Це дозволяє гнучко маніпулювати даними, але вимагає додаткової уваги до помилок типізації[9].

Об'єктно-орієнтована мова: JavaScript підтримує об'єктно-орієнтоване програмування (ООП), що дозволяє створювати об'єкти і класи, а також використовувати наслідування. У сучасній версії ECMAScript 6 (ES6) була введена підтримка класів, що спрощує роботу з ООП[9].

Асинхронність: JavaScript має вбудовані механізми для роботи з асинхронними операціями, такими як колбеки, проміси та async/await. Це дозволяє ефективно управляти операціями введення-виведення, такими як запити до серверів або робота з файлами[]6.

Подієво-орієнтована модель: JavaScript працює на основі подій, що дозволяє створювати інтерактивні додатки, які реагують на дії користувачів, такі як кліки, наведення миші, натискання клавіш тощо[6].

Широка підтримка в браузерах: JavaScript підтримується всіма сучасними веб-браузерами, що дозволяє створювати кросплатформні веб-додатки. Браузери мають вбудовані рушії для виконання JavaScript, такі як V8 в Google Chrome або SpiderMonkey в Mozilla Firefox[6].

Велика екосистема: JavaScript має величезну екосистему бібліотек і фреймворків, таких як React, Angular, Vue.js, Node.js, Express.js та багато інших, що значно спрощує і пришвидшує розробку складних додатків.

Переваги JavaScript:

Універсальність: JavaScript підтримується всіма сучасними веб-браузерами, що дозволяє створювати кросплатформні додатки без необхідності додаткових інструментів або налаштувань.

Інтерактивність: Можливість створення динамічних і інтерактивних елементів на веб-сторінках значно підвищує користувацький досвід.

Велика спільнота: Завдяки величезній спільноті розробників, JavaScript має велику кількість бібліотек та фреймворків (таких як React, Angular, Vue.js), які спрощують розробку складних додатків.

Асинхронність: Вбудована підтримка асинхронних операцій (через колбеки, проміси та async/await) дозволяє ефективно керувати часозатратними операціями без блокування основного потоку виконання.

JavaScript є потужним та універсальним інструментом для створення сучасних веб-додатків. Його широкі можливості, велика спільнота та активний розвиток роблять його одним з найбільш популярних мов програмування у світі. Незважаючи на деякі недоліки, його переваги значно переважають, що забезпечує подальше зростання і розвиток JavaScript у майбутньому.

В додаток було встановленно середовище виконання JavaScript, яке дозволяє запускати JavaScript-код поза межами веб-браузера.

Node.js значно розширив можливості JavaScript, зробивши його потужним інструментом для серверної розробки. Завдяки використанню двигуна V8 від Google, Node.js забезпечує високу продуктивність і ефективність роботи додатків.

Основні особливості Node.js[7].

Подієво-орієнтована архітектура: Node.js працює на основі подій і використовує неблокуючий ввід-вивід, що дозволяє обробляти велику кількість одночасних з'єднань з високою продуктивністю[7].

Однопоточність з підтримкою асинхронності: Node.js працює в одному потоці, що спрощує управління ресурсами та уникнення проблем, пов'язаних із багатопоточністю, таких як стан гонок. Асинхронність забезпечується за допомогою колбеків, промісів та async/await.

NPM (Node Package Manager): NPM є стандартним менеджером пакетів для Node.js. Він надає доступ до величезної кількості бібліотек і модулів, які можна легко встановити і використовувати у своїх проектах. Це значно спрощує розробку, дозволяючи використовувати готові рішення для типових задач[7].

Модульна система: Node.js використовує власну модульну систему (CommonJS), яка дозволяє організовувати код у вигляді модулів. Це полегшує підтримку та повторне використання коду[7].

Швидкість і продуктивність: Завдяки використанню двигуна V8, Node.js забезпечує високу швидкість виконання коду. V8 компілює JavaScript у машинний код перед виконанням, що значно підвищує продуктивність.

Переваги Node.js

Висока продуктивність: Використання V8 та неблокуючий ввід-вивід забезпечують високу продуктивність додатків.

Масштабованість: Node.js дозволяє легко масштабувати додатки, як горизонтально (додавання нових серверів), так і вертикально (додавання нових процесів).

Єдина мова програмування: Використання JavaScript як для клієнтської, так і для серверної частин спрощує розробку і підтримку додатків.

Велика спільнота: Широка спільнота розробників і великий вибір модулів та бібліотек роблять розробку на Node.js швидкою і ефективною.

Node.js є потужним інструментом для розробки серверних додатків, який забезпечує високу продуктивність і масштабованість завдяки своїй подієво-орієнтованій архітектурі та неблокуючому вводу-виводу. Використання JavaScript для серверної розробки спрощує процес створення веб-додатків, дозволяючи розробникам використовувати одну мову програмування як на клієнтській, так і на серверній стороні[7].

В додаток до JavaScript також був використаний потужний JavaScript-фреймворк Vue.js який використовується для створення інтерфейсів користувача та односторінкових додатків. Він був розроблений Єваном Ю та вперше був випущений в 2014 році. Vue.js часто хвалять за його простоту, гнучкість та легкість інтеграції з іншими бібліотеками та проектами.

Одна з ключових особливостей Vue.js - це його архітектура на базі компонентів, що дозволяє розробникам створювати перевикористовувані та інкапсульовані елементи інтерфейсу. Vue.js також надає потужну екосистему з інструментами та бібліотеками для різноманітних задач, таких як керування станом, маршрутизація та рендеринг на стороні сервера[36].

Vue.js здобув значну популярність у спільноті розробників веб-додатків завдяки його простому процесу навчання, зрозумілій документації та активній підтримці спільноти. Він конкурує з іншими популярними фреймворками JavaScript, такими як React та Angular, пропонуючи привабливу альтернативу для розробників, які шукають легку та універсальну рішення для створення сучасних веб-додатків[36].

А також в для реалізації програми було використано Electron.Це фреймворк для створення кросплатформенних настільних додатків з використанням веб-технологій, таких як HTML, CSS та JavaScript. Розроблений спочатку компанією GitHub, Electron значно спрощує процес створення настільних додатків, дозволяючи використовувати знання та інструменти, знайомі веб-розробникам.

Також в даній роботі було використно «shamirs-secret-sharing» - це бібліотека для JavaScript, яка надає можливість реалізації схеми розподілу секрету Шамира. Ця схема дозволяє розділити секрет на кілька частин (так званих "часток") і розподілити їх між учасниками, при цьому зазначеної кількості учасників достатньо буде будь-яка частина, щоб відновити вихідний секрет.

Використання цієї бібліотеки дозволяє легко виконувати наступні операції:

Розділення секрету: Створення секрету і розділення його на задану кількість часток.

Об'єднання часток: Об'єднання достатньої кількості часток для відновлення початкового секрету.

Ця бібліотека корисна для забезпечення безпеки даних шляхом розподілу секретної інформації між декількома учасниками, забезпечуючи таким чином безпеку від втрати чи доступу до даних неповноважних осіб.

3.2 Програмна реалізація створеного додатку

Впровадження програмного забезпечення є невід'ємною частиною процесу його розробки. Це включає створення алгоритмів, структур даних, функцій та модулів, які дозволяють програмі виконувати необхідні завдання.

Цей код налаштовує базовий вікно за допомогою Electron. Ось його основні етапи: необхідні модулі Electron та node.js:

electron: модуль, необхідний для створення десктопних додатків.

node:path: модуль node.js для роботи з файловою системою та шляхами.

shamirs-secret-sharing: модуль для реалізації схеми секретного ділення Шаміра.

Створюється вікно за допомогою класу BrowserWindow, вказуючи його параметри, такі як ширину, висоту та параметри веб-переглядача. налаштовуєте параметр webPreferences, вказуючи шлях до файлу preload.js. Це використовується для завантаження скриптів до веб-сторінки, щоб мати доступ до API Node.js.

Нарешті, функція createWindow створює новий екземпляр BrowserWindow.

```
const { app, BrowserWindow, ipcMain } = require("electron");
const path = require("node:path");
const sss = require("shamirs-secret-sharing");
const createWindow = () => {
  const win = new BrowserWindow({
    width: 800,
    height: 600,
    webPreferences: {
      preload: path.join(__dirname, "preload.js"),
    },
  });
```

Далі наведено код який очікує, коли додаток Electron буде готовий за допомогою методу app.whenReady(). Після того, як додаток готовий, викликається

функція createWindow(), яка створює вікно. Також, встановлюється обробник події "activate", який викликається, коли користувач натискає на піктограму програми у доку або панелі завдань

```
app.whenReady().then(() => {
 createWindow();

 app.on("activate", () => {
  if (BrowserWindow.getAllWindows().length === 0) createWindow();
 });
});
```

Далі іде дуже важлива частина коду, тому що ipcMain.on("encode-secret", ...) є обробником подій який активується, коли відображення відправляє повідомлення з ім'ям "encode-secret". Він приймає параметри:

event: об'єкт події IPC, який дозволяє відповісти на відправлену подію.

secretKey: секретний ключ, який потрібно закодувати.

Об'єкт з параметрами sharesCount і thresholdCount, які вказують кількість акцій та поріг, необхідний для відновлення секрету.

Об'єкт з параметрами sharesCount і thresholdCount, які вказують кількість акцій та поріг, необхідний для відновлення секрету.

Параметри sharesCount та thresholdCount виводяться в консоль.

Секретний ключ перетворюється в буфер за допомогою Buffer.from(secretKey).

Функція sss.split з бібліотеки shamirs-secret-sharing використовується для розбиття секрету на фрагменти за вказаними параметрами shares та threshold.

Результат (фрагменти) перетворюються в рядок у форматі base64 за допомогою методу chunk.toString("base64").

Закодовані фрагменти відправляються назад до відображення за допомогою event.reply("encode-secret-result", ...).

```
ipcMain.on(
 "encode-secret",
 (event, secretKey, { sharesCount, thresholdCount }) => {
  console.log(sharesCount, thresholdCount);
```

```
  const secret = Buffer.from(secretKey);
  const shares = sss.split(secret, {
  shares: +sharesCount,
    threshold: +thresholdCount,
  });
  event.reply(
    "encode-secret-result",
    shares.map((chunk) => {
      return chunk.toString("base64");
    })
  );
 }
);
```

Наступна частина коду є також дуже важлива адже ipcMain.on("decode-secret", ...) є обробником подій який активується коли відображення відправляє повідомлення з ім'ям "decode-secret".

Приймає два параметри:

event: об'єкт події IPC, який дозволяє відповісти на відправлену подію.

shares: масив фрагментів, які потрібно об'єднати та відновити секрет.

Фрагменти перетворюються в буфер та фільтруються, щоб видалити порожні значення за допомогою методу shares.filter(chunk => chunk.length).

Потім фрагменти декодуються з base64 назад у буфери за допомогою методу Buffer.from(chunk, "base64").

Функція sss.combine з бібліотеки shamirs-secret-sharing використовується для об'єднання буферів та відновлення секрету.

Результат (відновлений секрет) перетворюється в рядок і відправляється назад до відображення за допомогою event.reply("decode-secret-result", ...).

```
ipcMain.on("decode-secret", (event, shares) => {
  const bufferedShares = shares.filter(chunk => chunk.length).map((chunk) => Buffer.from(chunk,
"base64"));
  console.log(shares.filter(chunk => chunk.length));
  const recovered = sss.combine(bufferedShares);
  event.reply("decode-secret-result",  recovered.toString());
});
```

Далі розглянемо як зроблене оформлення сайту та його інтерфейс. HTML-код який наведено нище представляє собою базовий шаблон сторінки веб-

додатка, який використовує Vue.js. Ось опис його складових: <head>...</head>: Цей тег містить метадані та посилання на зовнішні ресурси, такі як стилі та скрипти. title>...</title>:Вказує заголовок сторінки, який буде відображатися у вкладці браузера. <style>...</style>:В цьому розділі містяться стилі для відображення сторінки.Задаються стилі для елементів, таких як <body>, , <button>, <input>, <ul>, <li> та інші.

Налаштовується оформлення тексту, колір фону, ширини, відступи, рамки, курсори миші тощо.Зокрема, використовуються класи button, form-group, tabBar, addFieldButtonContainer для стилізації елементів.

```
<script src="https://cdn.jsdelivr.net/npm/vue@2"></script>
    font-family: Arial, sans-serif;
  background-color: hsl(0, 8%, 12%);
}
#app {
  max-width:  600px;
  margin: 50px auto;
  padding: 20px;
  border: 1px solid #ffffff;
  background-color: #beb6b6;
  text-align: center;
}
.form-group {
  margin: 15px 0;
}
label {
  display: block;
  margin-bottom: 5px;
}
input {
  padding: 8px;
  width: calc(100% - 16px);
}
.tabBar {
  display: flex;
}
.addFieldButtonContainer {
  display: flex;
  width: 100%;
  justify-content: flex-end;
}
```

Наступний є HTML-код який є частиною веб-сторінки, яка використовує Vue.js для динамічного взаємодії з користувачем. ... :Це головний контейнер для Vue.js додатка. Всі інші елементи розміщуються всередині цього контейнера. ... :В цьому блоку знаходяться дві кнопки, які використовуються для перемикання між режимами "Encode" та "Decode". При кліку на кнопку викликається метод changePage(index), який змінює pageIndex.<h1>...</h1>:Заголовок сторінки, який відображає текст "Shamir's secret sharing". ... :Цей блок відображається тільки тоді, коли pageIndex має значення 0, тобто коли активований режим "Encode".Містить поле введення для секретного ключа, кнопку "Encode", список для відображення фрагментів секрету, поля для введення кількості акцій та порогу. ... :Цей блок відображається тільки тоді, коли pageIndex має значення 1, тобто коли активований режим "Decode".Містить поля для введення фрагментів секрету для декодування, кнопку "Submit" та відображення розкодованого секрету.

Цей код використовує директиви Vue.js, такі як v-if, v-for, v-model, а також обробники подій, щоб реалізувати веб-інтерфейс для кодування та декодування секретів за допомогою схеми секретного ділення Шаміра.

```
<body>
 <div id="app">
  <div class="tabBar">
   <button class="button" @click="changePage(0)">Encode</button>
   <button class="button" @click="changePage(1)">Decode</button>
  </div>
  <h1>Shamir's secret sharing</h1>
  <div v-if="pageIndex === 0">
   <div class="form-group">
    <label for="secretKey"></label>
    <input v-model="secretKey" id="secretKey" />
   </div>
   <button class="button" @click="encode">Encode</button>
   <div class="form-group">
    <h2>secret parts</h2>
    <ul>
```

```
      <li v-for="(chunk, index) in secretChunks" :key="index">
       {{ chunk }}
      </li>
     </ul>
    </div>
    <div class="form-group">
     <label for="sharesCount">input amount of Keys to split secret</label>
     <input type="number" v-model="sharesCount" id="sharesCount" />
    </div>
    <div class="form-group">
     <label for="thresholdCount">
      Input amount of Keys to decode secret
     </label>
     <input type="number" v-model="thresholdCount" id="thresholdCount" />
    </div>
   </div>
   <div v-if="pageIndex === 1">
    <div class="form-group">
     <label for="secretKey">Input keys to decript</label>

     <input
      v-for="(chunk, index) in secretChunksToDecode"
      :key="index"
      v-model="secretChunksToDecode[index]"
     />
     <div class="addFieldButtonContainer">
      <button class="button" @click="addChuknFieldToDecode">
       Add field
      </button>
     </div>
     <input disabled v-model="decodedSecret" />
     <button class="button" @click="decode">Submit</button>
    </div>
   </div>
  </div>
```

Наступним є JavaScript-код який використовує Vue.js для створення екземпляра Vue та визначення його даних (data), які використовуються для управління станом додатку. data: { ... }:

Об'єкт data містить дані, які будуть використовуватися в шаблоні Vue.

secretKey: рядок, який представляє секретний ключ.

sharesCount: кількість акцій для розподілення секрету.

thresholdCount: порігова кількість акцій, необхідних для відновлення секрету.

keysForSecret: кількість ключів для секрету (якщо використовується).

selectedSecret: індекс вибраного секрету (якщо використовується).

secretChunks: масив фрагментів секрету.

pageIndex: індекс поточної сторінки (0 - режим кодування, 1 - режим декодування).

secretChunksToDecode: масив фрагментів секрету для декодування.

decodedSecret: розкодований секрет (якщо використовується).

```
<script>
  new Vue({
  el: "#app",
  data: {
     secretKey: "",
     sharesCount: 10,
     thresholdCount: 4,
     keysForSecret: 5,
     selectedSecret: 0,
     secretChunks: [],
     pageIndex: 0,
     secretChunksToDecode: [""],
     decodedSecret: "",
   },
```

Наступний код визначає метод mounted() в екземплярі Vue. Метод mounted() викликається автоматично після того, як екземпляр Vue був вміщений у DOM елемент, що вказано властивістю el. У цьому методі встановлюються обробники подій для результатів операцій кодування та декодування. Якщо говорити коромко то цей код цей код встановлює обробники подій для результатів операцій кодування та декодування, які отримані через window.electronAPI.

```
mounted() {
    window.electronAPI.onEncodeSecretResult((event, result) => {
      console.log("Результат операции:", result);
      this.secretChunks = result;
    });
    window.electronAPI.onDecodeSecretResult((event, result) => {
      console.log("Результат операции:", result);
      this.decodedSecret = result;
```

```
    });
  },
```

Далі було створено функцію яка містить методи, які визначені в екземплярі Vue і використовуються для взаємодії з користувачем та виклику функцій зовнішнього API. decode(), addChuknFieldToDecode(), changePage(index)**,** encode().Ці методи виконують важливі операції у додатку, такі як додавання нових полів введення, зміна сторінок і виклик функцій зовнішнього API для кодування та декодування секретів.

```
methods: {
    decode() {
      window.electronAPI.decodeSecret(this.secretChunksToDecode);
    },
    addChuknFieldToDecode() {
      this.secretChunksToDecode.push("");
    },
    changePage(index) {
      this.pageIndex = index;
    },
    encode() {
      if (!this.secretKey.length) {
        return;
      }
      window.electronAPI.encodeSecret(this.secretKey, {
        sharesCount: this.sharesCount,
        thresholdCount: this.thresholdCount,
      });
    },
  },
```

Розробка програмного забезпечення має ключове значення в сучасному світі, де програми та додатки стають все більш складними та функціональними. Вона дозволяє перетворювати ідеї та концепції в реальність, створюючи програми, які надають користувачам потрібні функції та можливість взаємодії з інформацією.

3.3 Інструкція користування а також тестування розробленої програми

Тестування програмного застосунку є невід'ємною частиною процесу розробки, оскільки дозволяє перевірити його працездатність, надійність та відповідність вимогам.

Насамперед користувач відкривши програму побачить головне меню в якому є 3 кнопки:

2 кнопки зверху дають змогу переключатись між сторінками залежно від того чи користувач хоче закодувати якесь повідомлення, чи розкодувати його.

По середині меню є кнопка закодувати (рис 3.1), ця кнопка відподіває за те щоб при вводі в поле якогось повідомлення можна було закодувати це повідомлення.

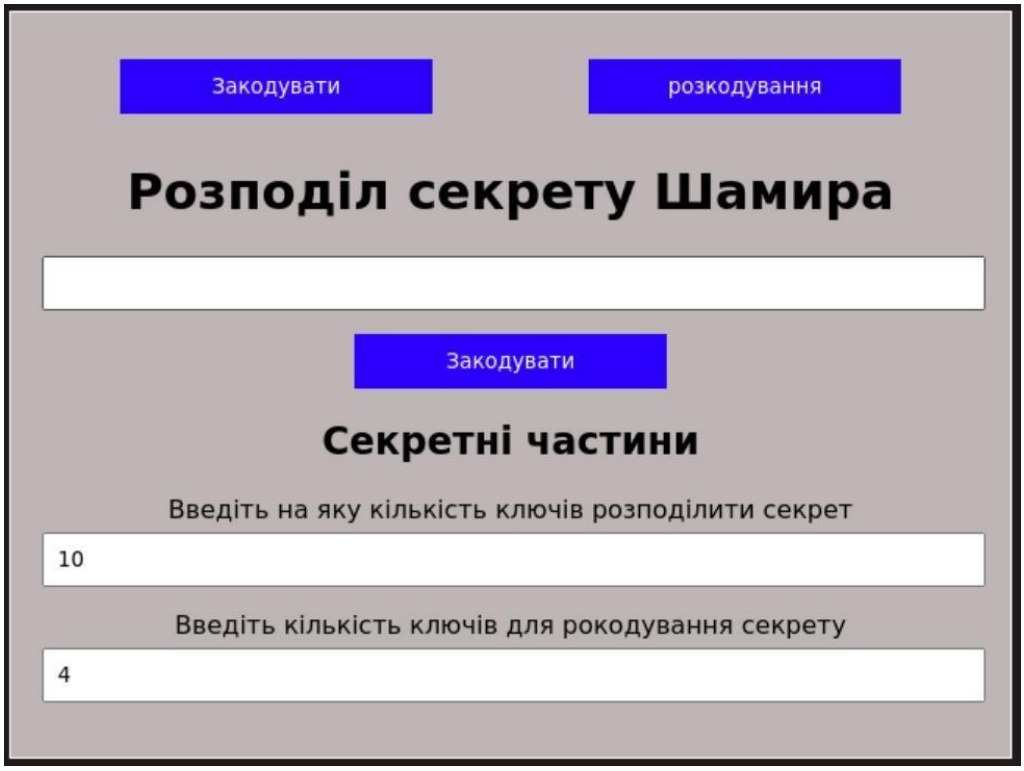


Рисунок 3.1 – Головне меню реалізованї програми

Далі чуть нище є поле в якому можна вибрати кількість ключів на яку потрібно розподілити повідомлення. (рис. 3.2). Важливий момент, ключів не може бути меньше ніж 2, адже тоді вся суть розподілу секрету втрачається.

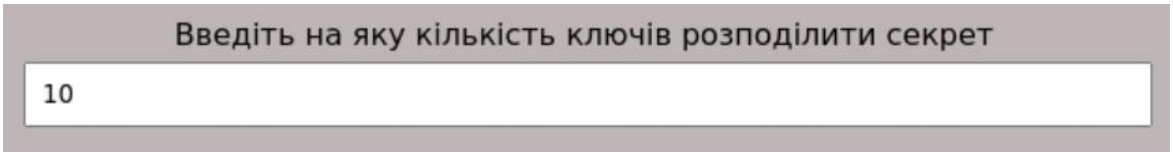


Рисунок 3.2 – Поле вибору на яку кількість ключів повідомлення поділиться

Наступним чуть нище йде таке саме поле вибору як щойно було розглянуто, але це поле відповідає за кількість ключів яка потрібна для того, щоб розкодувати повідомлення.(рис. 3.3)

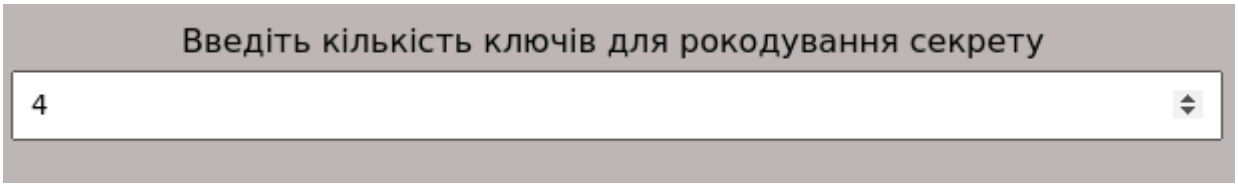


Рисунок 3.3 – Поле вибору яку кількість ключів необхідно для розкодування повідомлення

Попередньо було розглянуто функціонал першою сторінки, тобто розкодування. Далі розглянемо другу сторінку, яка відповідає за розкодування того повідомлення яке було закодовано на попередній сторінці. (Рис 3.4)

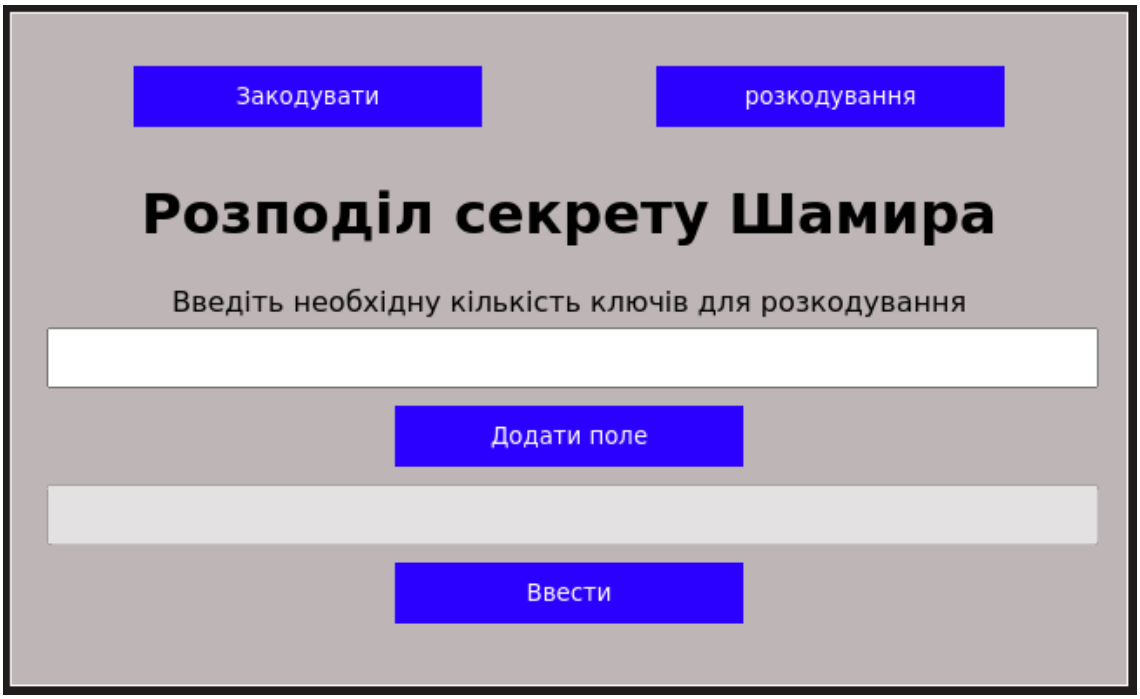


Рисунок 3.4 – Меню розкодування

Дане меню має лише два поля. Поле яке знаходиться посередині програми відповідає за введення в нього ключів необхідних для розкодування повідомлення. (рис 3.5)

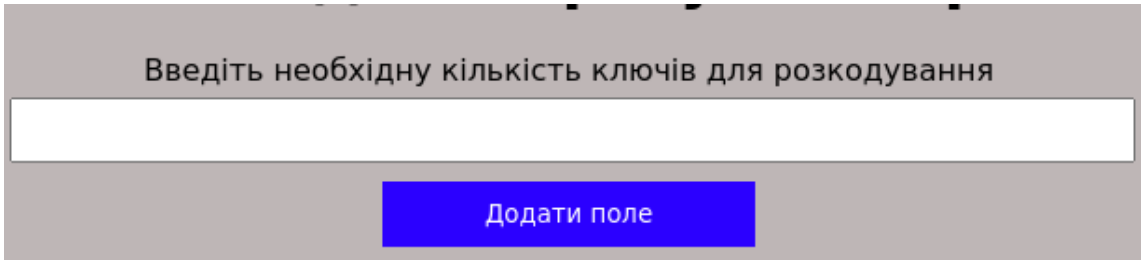


Рисунок 3.5 – Поле вводу ключів для розкодування.

Якщо нажати на кнопку додати поле, то буде додано додаткове поле вводу ключа, як показано на (рисунку 3.6)

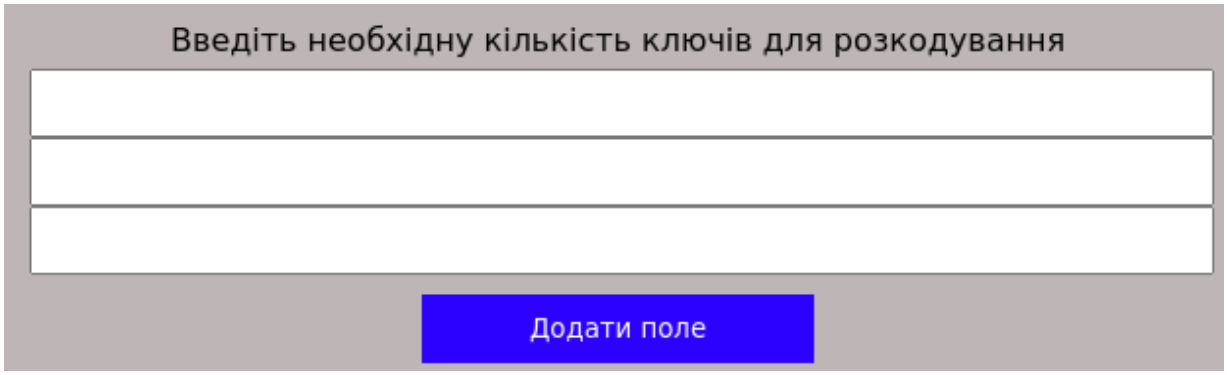


Рисунок 3.6 – Додавання полей вводу для ключів розкодування

Вище було описано функціонал програми, теперь потрібно протестувати її працездатність на практиці.

Скажімо що нам потрібно закодувати пароль наприклід від сейфу, нам потрібно щоб про цей пароль знали лише 5 робітників, але для того щоб відкрити цей сейф задамо лише двох робітників.(рис 3.7)

Паролем буде: 154586

Робітників 5

Кількість ключів для рокодування 2

Для початку потрібно додати ключ у поле для кодування пароля. (Рис 3.7)

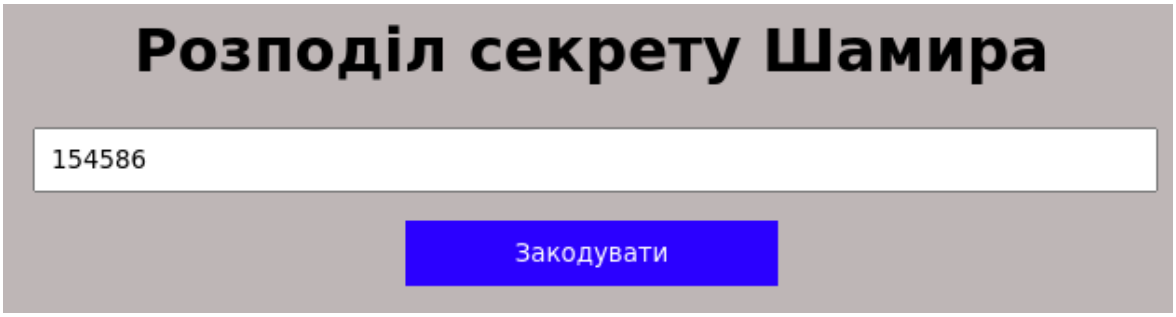


Рисунок 3.7 – Додавання секрету у поле вводу

Наступним кроком потрібно розподілити цей секрет між п’ятьма робітниками, для цього вводимо потрібну кількість у полі для вводу розподілення ключів. (рис 3.8)

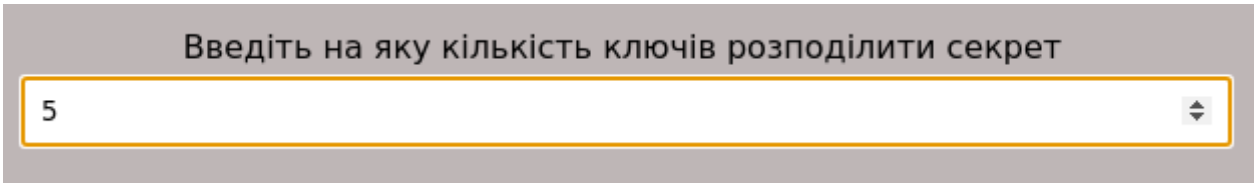


Рисунок 3.8 – Додавання кількость робітників для розподілу секрету

І нарешті для кодування повідомлення потрібно ввести кількісь ключів для розкодування. (рис 3.9)

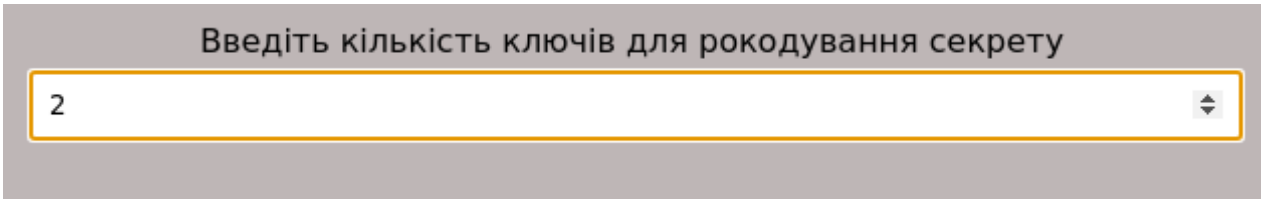


Рисунок 3.9 – Додавання кількості ключів для розкодування

Після чого натиснути закодувати. Якщо все зробити правильно, то на екрані з’явиться 5 ключів які потрібно розподілити між робітниками. (рис 3.10)

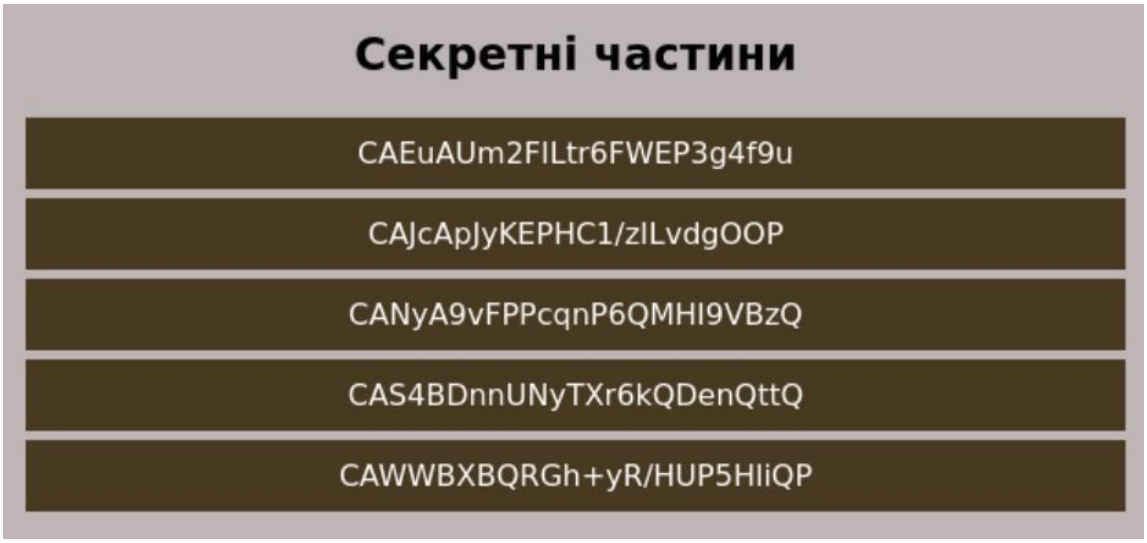


Рисунок 3.10 – Результат кодування секрету

Теперь для того щоб розкодувати секрет, потрібно перейти на сторінку розкодування. (3.4).Після чого додати необхідну кількість полів для вводу ключів, в нашому випадку їх 2.(рис 3.11)

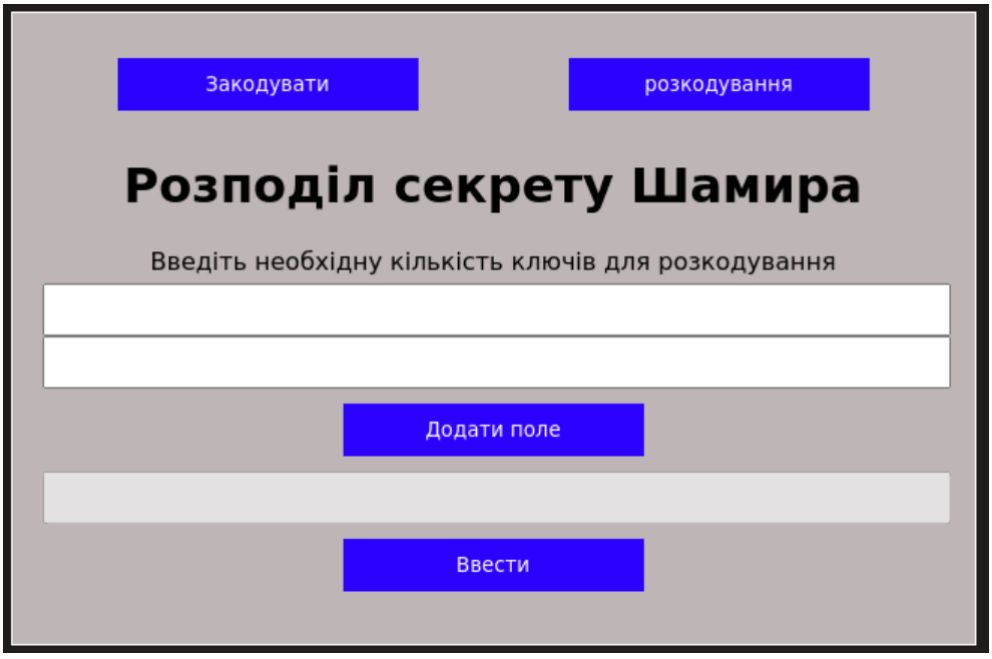


Рисунок 3.11– Сторінка розкодування з доданим полем

Далі нам потрібні будь яких 2 дійсних ключа які були розподіленні між п'ятьма робітниками. (рис 3.12)

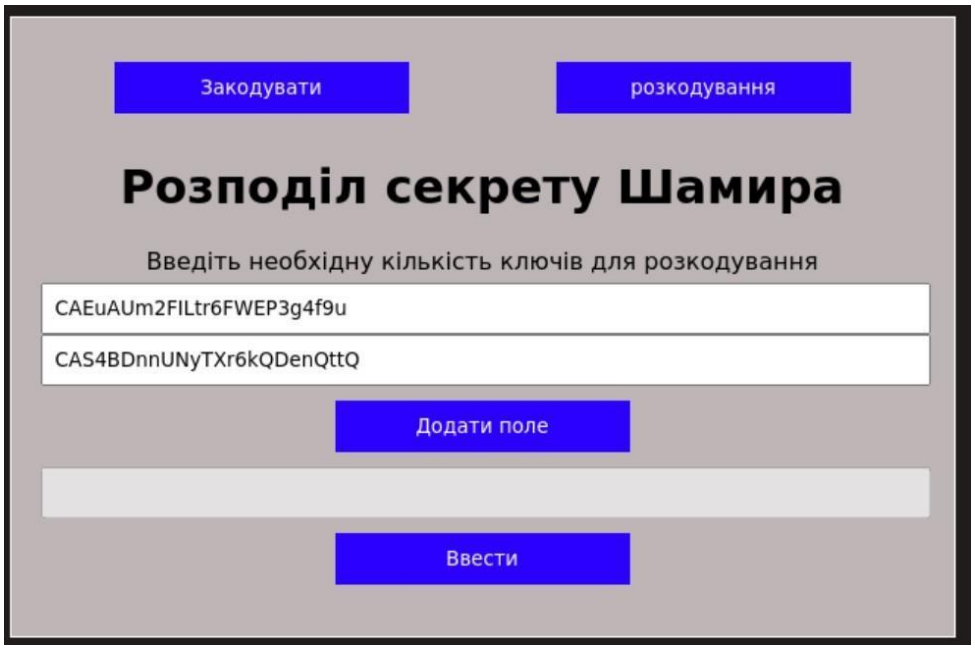


Рисунок 3.12– Ввод паролів у поле

Після чого просто натиснути на кнопку ввести, і в полі повинен з'явитись результат розкодування.(рис 3.13)

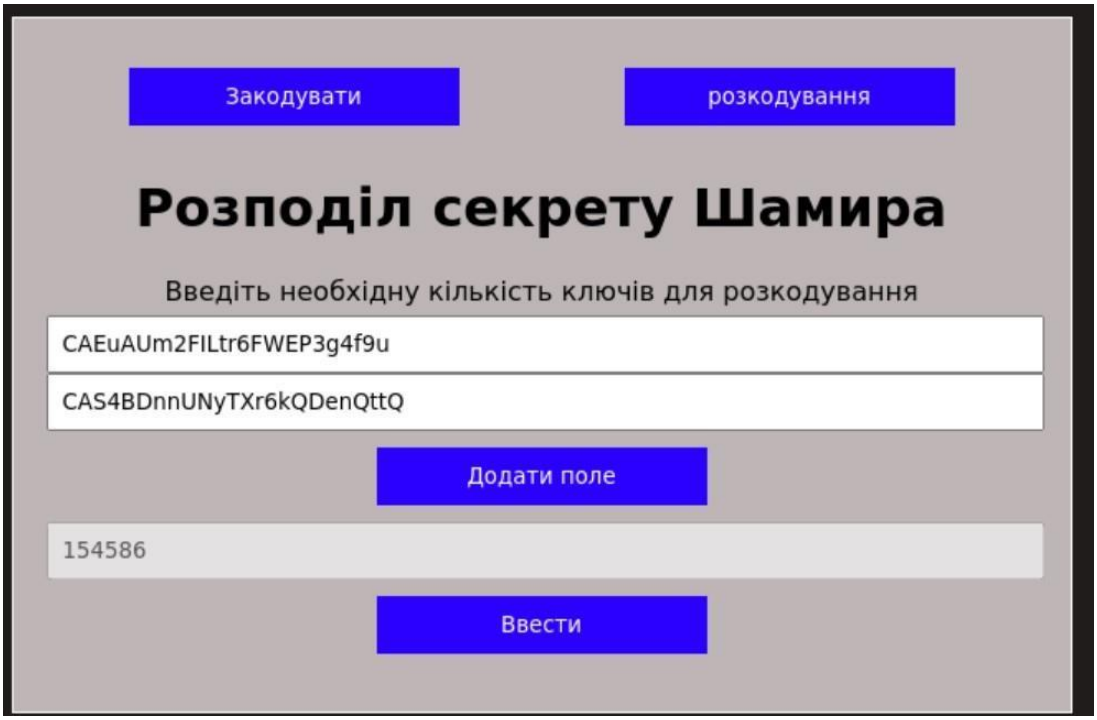


Рисунок 3.13– Результат розкодування

Отже, при демонстрації графічного інтерфейсу, було здійснено пояснення роботи користувача з інтерфейсом та всіма сторінками розробленої програми.

3.4 Висновки до розділу 3

У даному розділі було детально розглянуто процес вибору засобів програмування для розробки сучасних програмних продуктів, звертаючи увагу на різні аспекти, такі як функціональність, зручність використання, підтримка різних мов програмування, інтеграція з іншими інструментами та спільнотою розробників. Ми зосередилися на Visual Studio Code (VS Code), який визнаний своєю гнучкістю, швидкодією та широкими можливостями налаштування.

Далі ми обрали мову програмування JavaScript для реалізації програми розподілу секрету між користувачами. JavaScript відомий своєю універсальністю та широким спектром застосувань у веб-розробці, а також має велику спільноту розробників та розвинуту екосистему.

Для створення інтерфейсу користувача ми використали JavaScript-фреймворк Vue.js, який забезпечує простоту, гнучкість та ефективність в роботі з інтерфейсами користувача.

Для реалізації схеми розподілу секрету Шаміра ми використали бібліотеку "shamirs-secret-sharing", яка надає можливість ефективно розділяти секрет на частки та відновлювати секрет з достатньою кількістю часток.

У підсумку, використання цих засобів програмування та технологій дозволило нам успішно реалізувати програму розподілу секрету з гнучким інтерфейсом користувача та ефективними алгоритмами розподілу та відновлення секрету.

ВИСНОВКИ

Було проведено аналіз існуючих схем розподілу секрету та їх відновлення, включаючи криптосистеми з прогресивним розподілом секрету. Було виявлено актуальність проблеми забезпечення безпеки та відновлення конфіденційної інформації, особливо в умовах високотехнологічних загроз та широкого використання інформаційних технологій.

У розділі проектування схеми розподілу секрету та його відновлення було розроблено алгоритм відновлення секрету, який передбачає зберігання засобу для відновлення на окремому пристрої з метою підвищення безпеки зберігання та ускладнення несанкціонованого доступу. Також було розроблено алгоритм програми для розподілення секрету між групою користувачів з урахуванням принципів безпеки та ефективності.

У прикладній частині роботи було вибрано засоби програмування, середовище розробки та мову програмування для реалізації створеного додатку, який був успішно реалізований. Надалі була надана інструкція користування та проведено тестування розробленої програми, підтвердивши її функціональність та надійність.

Отже, результати дослідження та розробки показують, що використання схем розподілу секрету та відновлення може ефективно забезпечувати безпеку конфіденційної інформації та забезпечувати можливість відновлення даних в разі потреби. Дані роботи можуть бути корисними для фахівців у галузі криптографії та інформаційної безпеки, а також для розробки та впровадження систем захисту інформації в різних сферах діяльності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розділення секрету. Режим доступу до ресурсу: URL:https://bit.nmu.org.ua/ua/student/metod/cryptology/%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D1%8F23.pdf

2. Blakley G. R. Safeguarding cryptographic keys (англ.) // Proceedings of the 1979 AFIPS National Computer Conference — Montvale: AFIPS Press, 1979. P. 313-317. URL:https://www.computer.org/csdl/proceedingsarticle/afips/1979/50870313/12OmNCeK2a1.

3. Shamir A. How to share a secret Commun. ACM New York: Association for Computing Machinery, 1979. Vol. 22, Iss. 11. P. 612-613.

4. Shamir A. How to share a secret (англ.) // Commun. ACM — [New York]: Association for Computing Machinery, 1979. — Vol. 22, Iss. 11. — P. 612— 613. — ISSN 0001-0782 — doi:10.1145/359168.359176(дата звернення 19.04.2024).

5. Технології захисту інформації. URL: https://www.uzhnu.edu.ua/uk/infocentre/get/4186 (дата звернення: 17.05.2024).

6. JavaScript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 25.04.2024).

7. Node.js. Node.js. URL: https://nodejs.org/uk (дата звернення: 25.04.2024).

8. Sufiyan T. What is node.js: a comprehensive guide. Simplilearn.com. URL: https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs (дата звернення: 25.04.2024)

9. The Modern JavaScript. URL: https://javascript.info/ (дата звернення: 25.04.2024).

10. Electron. Electron. URL: https://www.electronjs.org/ (дата звернення: 26.04.2024).

11. Marcin Dryka. What is electron.js?. URL: https://brainhub.eu/library/what-iselectron-js (дата звернення: 26.04.2024).

12. Редактор коду visual studio code. Habr. URL: https://habr.com/ua/articles/490754 (дата звернення: 26.04.2024).

13. Visual studio code. Visual Studio Code. URL: https://code.visualstudio.com/ (дата звернення: 26.04.2024).

14. Лужецький В. А. Основи інформаційної безпеки: навч. посіб. / В. А. Лужецький, А. Д. Кожухівський, О. П. Войтович. - Вінниц. нац. техн. ун-т. - Вінниця: ВНТУ, 2013. - 220 с

15. What is CSS? Mozilla. URL: What is CSS? - Learn web development | MDN (mozilla.org) (дата звернення 11.04.2024).

16. Safeguarding cryptographic keys Safeguarding cryptographic keys by G. R. BLAKLEY Texas A&M University College Station, Texas URL: 50 https://web.archive.org/web/20160303193810/https://www.computer.org/csdl/proceedings/afips/1979/5087/00/50870313.pdf

17. Lim G., Beginning Node.js, Express & MongoDB Development.: Independently Published, 2019.

18. Frisbie M. Professional JavaScript for Web Developers.: Wiley. John Wiley & Sons, LTD, 2023.-

19. Бородкіна І. Л., WEB-технології та WEB-дизайн: застосування мови HTML для створення електронних ресурсів: КНУКіМ, 2017. - 329 с.

20. Глибовець М., Основи комп`ютерних алгоритмів.: Києво-Могилянська академія, 2003. - 452 с.

21. Фрейн Б., Чуйний дизайн на HTML5 та CSS3 для будь-яких пристроїв., 2002. - 336 с

22. Дудикевич В. Б., Хорошко В. О., Яремчук Ю. Є. Основи інформаційної безпеки.: навч. пос. Вінниця : ВНТУ, 2018. – 316 с

23. Adi S., How to share a secret, Communications of the ACM [Електронний ресурс] / Режим доступу: https://dl.acm.org/doi/10.1145/359168.359176 (дата звернення 26.07.2023).

24. "Secret Sharing" від Menezes, A. J. et al., "Handbook of Applied Cryptography": Режим доступу: https://crypto.stanford.edu/~dabo/cryptobook/BonehShoup_0_4.pdf (дата звернення 15.05.2024).

25. Yvo Desmedt. "Secret Sharing Schemes" Conference: Coding and Cryptology - Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings.

26 A. Beimel and Y. Ishai. On the power of nonlinear secret-sharing. SIAM J. onDiscrete Mathematics, 19(1):258–280, 2005

27. J. R. Metcalf-Burton. Improved upper bounds for the information rates ofthe secret sharing schemes induced by the Vamos matroid. Technical Reportabs/0809.3010, CoRR, 2008

28. T. Tassa and N. Dyn. Multipartite secret sharing by bivariate interpolation. InM. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, Proc. of the 33rdInternational Colloquium on Automata, Languages and Programming, volume 4052of Lecture Notes in Computer Science, pages 288–299. Springer-Verlag, 2006

29. R. W. Yeung. Information Theory and Network Coding. Springer, 2008.

30. A. Beimel. Secret-sharing schemes: A survey. In Coding and Cryptology 2011, pages 11–46, 2011.

31. Introduction to Modern Cryptography" від Jonathan Katz та Yehuda Lindell. Режим доступу: https://eclass.uniwa.gr/modules/document/file.php/CSCYB105/Reading%20Material/%5BJonathan_Katz%2C_Yehuda_Lindell%5D_Introduction_to_Mo%282nd%29.pdf (дата звернення 15.05.2024).

32. Denise Demirel, Johannes A. Buchmann. Performing Computations on Hierarchically Shared Secrets

33. William Stallings. Cryptography and network security principles and practice Режим доступу: https://www.cs.vsb.cz/ochodkova/courses/kpb/cryptography-and-network-security_-principles-and-practice-7th-global-edition.pdf (дата звернення 16.05.2024).

34. Global Cybersecurity Outlook 2024: Режим доступу: https://www.weforum.org/publications/global-cybersecurity-outlook-2024/?utm_source=google&utm_medium=ppc&utm_campaign=cybersecurity&gad_source=1&gclid=CjwKCAjw34qzBhBmEiwAOUQcF499qXrk8dDYxZr-M6DcXPj_ub0mRPkJ8iN7vCGY6QL_gok84kBRIhoCvRQQAvD_BwE (дата звернення 16.05.2024).

35. Ronald Cramer, Ivan Bjerre Damgard, Jesper Buus Nielsen. Secure Multiparty Computation and Secret Sharing 2015. Режим доступу: https://assets.cambridge.org/97811070/43053/frontmatter/9781107043053_frontmatter.pdf (дата звернення 16.05.2024).

36. MetaNit. Vue.js Режим доступу: https://metanit.com/web/vuejs/(дата звернення 26.04.2024).

ДОДАТКИ

Додаток А. Технічне завдання

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ
Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор
_____Юрій ЯРЕМЧУК
" 22 " березня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до бакалаврської дипломної роботи на тему:

Розробка програми розподілу секрету між групою користувачів

08-72.БДР.029.00.55. ТЗ

Керівник бакалаврської дипломної роботи
д.т.н., професор кафедри МБІС
_____ Яремчук Ю. Є
" _____ "

Вінниця – 2024 р.

Вінниця – 2024 р.

1. Найменування та область застосування

Програмний засіб методу захисту секрету від зловмисників. Область застосування: захист інформаційних ресурсів від несанкціонованого доступу у системах безпеки.

2. Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ № 80 від 11.03.2024 р.

3. Мета та призначення розробки

3.1 Мета розробки: розробка ефективного методу розподілу секрету між групою користувачів

3.2 Призначення: розроблений програмний засіб виконує захист секрету від несанкціонованого доступу

4. Джерела розробки

4.1 "Secret Sharing" від Menezes, A. J. et al., "Handbook of Applied Cryptography":Режимдоступу:https://crypto.stanford.edu/~dabo/cryptobook/BonehShoup_0_4.pdf (дата звернення 15.05.2024).

4.2 Yvo Desmedt. "Secret Sharing Schemes" Conference: Coding and Cryptology - Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings.

4.3 A. Beimel and Y. Ishai. On the power of nonlinear secret-sharing. SIAM J. onDiscrete Mathematics, 19(1):258–280, 2005

4.4 J. R. Metcalf-Burton. Improved upper bounds for the information rates ofthe secret sharing schemes induced by the Vamos matroid. 2008

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний засіб повинен мати зручний, легкий у використанні інтерфейс користувача;

5.1.2 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

5.1.3 Програмний засіб повинен виконувати процес автентифікації користувачів у системі.

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

5.2.2 Бази даних повинні бути налаштовані на автоматичне створення резервних копій;

5.2.3 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Pentium 1500 МГц і подібні до них;

– оперативна пам’ять – не менше 512 Mb;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

6. Вимоги до програмної документації

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.3

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Неможливість отримання доступу незареєстрованих користувачів до інформаційних ресурсів.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Початок	Закінчення
1.	Визначення напрямку бакалаврської роботи, формулювання теми	23.03.2024	05.04.2024
2.	Аналіз предметної області обраної теми	05.04.2024	16.04.2024
3.	Розробка алгоритму роботи	16.04.2024	28.04.2024
4.	Написання бакалаврської роботи на основі розробленої теми	28.04.2024	20.05.2024
5.	Передзахист бакалаврської дипломної роботи	24.05.2024	02.06.2024
6.	Виправлення, уточнення, корегування бакалаврської дипломної роботи	03.06.2024	11.06.2024
7.	Захист бакалаврської дипломної роботи	13.06.2024	17.06.2024

10. Порядок контролю та прийому

10.1 До приймання бакалаврської дипломної роботи надається:

- ПЗ до бакалаврської дипломної роботи;
- демонстрація результату бакалаврської дипломної роботи;
- презентація;
- відзив керівника роботи;
- відзив рецензента

Технічне завдання до виконання прийняв_____Шкробот Є. С.

Додаток В Лістинг коду програми

```
const { app, BrowserWindow, ipcMain } = require("electron");
const path = require("node:path");
const sss = require("shamirs-secret-sharing");
const createWindow = () => {
  const win = new BrowserWindow({
    width: 800,
    height: 600,
    webPreferences: {
      preload: path.join(__dirname, "preload.js"),
    },
  });

  win.loadFile("index.html");
};
app.whenReady().then(() => {
  createWindow();

  app.on("activate", () => {
    if (BrowserWindow.getAllWindows().length === 0) createWindow();
  });
});

app.on("window-all-closed", () => {
  if (process.platform !== "darwin") app.quit();
});

ipcMain.on(
  "encode-secret",
  (event, secretKey, { sharesCount, thresholdCount }) => {
    console.log(sharesCount, thresholdCount);
    const secret = Buffer.from(secretKey);
    const shares = sss.split(secret, {
      shares: +sharesCount,
      threshold: +thresholdCount,
    });
    event.reply(
      "encode-secret-result",
      shares.map((chunk) => {
        return chunk.toString("base64");
      })
    );
  }
);
ipcMain.on("decode-secret", (event, shares) => {
```

```
  const bufferedShares = shares.filter(chunk => chunk.length).map((chunk) =>
Buffer.from(chunk, "base64"));
  console.log(shares.filter(chunk => chunk.length));
  const recovered = sss.combine(bufferedShares);
  event.reply("decode-secret-result", recovered.toString());
});
<!DOCTYPE html>
<html>
  <head>
    <title>Separation of Secrets</title>
    <script src="https://cdn.jsdelivr.net/npm/vue@2"></script>
    <style>
      body {
        font-family: Arial, sans-serif;
        background-color: hsl(0, 8%, 12%);
      }
      #app {
        max-width: 600px;
        margin: 50px auto;
        padding: 20px;
        border: 1px solid #ffffff;
        background-color: #beb6b6;
        text-align: center;
      }
      .button {
        display: block;
        width: 200px;
        margin: 10px auto;
        padding: 10px;
        background-color: #2b00ff;
        border: none;
        color: white;
        cursor: pointer;
      }
      .button:hover {
        background-color: #9992ff;
      }
      .form-group {
        margin: 15px 0;
      }
      label {
        display: block;
        margin-bottom: 5px;
      }
      input {
        padding: 8px;
        width: calc(100% - 16px);
```

```
    }
    ul {
      list-style-type: none;
      padding: 0;
    }
    li {
      background-color: #483921;
      margin: 5px 0;
      padding: 10px;
      color: white;
    }
    .tabBar {
      display: flex;
    }
    .addFieldButtonContainer {
      display: flex;
      width: 100%;
      justify-content: flex-end;
    }
  </style>
</head>
<body>
  <div id="app">
    <div class="tabBar">
      <button class="button" @click="changePage(0)">Encode</button>
      <button class="button" @click="changePage(1)">Decode</button>
    </div>
    <h1>Shamir's secret sharing</h1>
    <div v-if="pageIndex === 0">
      <div class="form-group">
        <label for="secretKey"></label>
        <input v-model="secretKey" id="secretKey" />
      </div>
      <button class="button" @click="encode">Encode</button>
      <div class="form-group">
        <h2>secret parts</h2>
        <ul>
          <li v-for="(chunk, index) in secretChunks" :key="index">
            {{ chunk }}
          </li>
        </ul>
      </div>
      <div class="form-group">
        <label for="sharesCount">input amount of Keys to split secret</label>
        <input type="number" v-model="sharesCount" id="sharesCount" />
      </div>
      <div class="form-group">
```

```
      <label for="thresholdCount">
        Input amount of Keys to decode secret
      </label>
      <input type="number" v-model="thresholdCount" id="thresholdCount" />
    </div>
  </div>
  <div v-if="pageIndex === 1">
    <div class="form-group">
      <label for="secretKey">Input keys to decript</label>

      <input
        v-for="(chunk, index) in secretChunksToDecode"
        :key="index"
        v-model="secretChunksToDecode[index]"
      />
      <div class="addFieldButtonContainer">
        <button class="button" @click="addChuknFieldToDecode">
          Add field
        </button>
      </div>
      <input disabled v-model="decodedSecret" />
      <button class="button" @click="decode">Submit</button>
    </div>
  </div>
</div>

<script>
  new Vue({
    el: "#app",
    data: {
      secretKey: "",
      sharesCount: 10,
      thresholdCount: 4,
      keysForSecret: 5,
      selectedSecret: 0,
      secretChunks: [],
      pageIndex: 0,
      secretChunksToDecode: [""],
      decodedSecret: "",
    },
    mounted() {
      window.electronAPI.onEncodeSecretResult((event, result) => {
        console.log("Результат операции:", result);
        this.secretChunks = result;
      });
      window.electronAPI.onDecodeSecretResult((event, result) => {
        console.log("Результат операции:", result);
```

```
            this.decodedSecret = result;
          });
        },
        methods: {
          decode() {
            window.electronAPI.decodeSecret(this.secretChunksToDecode);
          },
          addChuknFieldToDecode() {
            this.secretChunksToDecode.push("");
          },
          changePage(index) {
            this.pageIndex = index;
          },
          encode() {
            if (!this.secretKey.length) {
              return;
            }
            window.electronAPI.encodeSecret(this.secretKey, {
              sharesCount: this.sharesCount,
              thresholdCount: this.thresholdCount,
            });
          },
        },
      });
    </script>
  </body>
</html>
const { contextBridge, ipcRenderer } = require("electron");

window.addEventListener("DOMContentLoaded", () => {
  const replaceText = (selector, text) => {
    const element = document.getElementById(selector);
    if (element) element.innerText = text;
  };

  for (const dependency of ["chrome", "node", "electron"]) {
    replaceText(`${dependency}-version`, process.versions[dependency]);
  }
});

contextBridge.exposeInMainWorld("electronAPI", {
  encodeSecret: (secretKey, options) => ipcRenderer.send("encode-secret", secretKey,
options),
  onEncodeSecretResult: (callback) =>
    ipcRenderer.on("encode-secret-result", callback),
  decodeSecret: (chunks) => ipcRenderer.send("decode-secret", chunks),
  onDecodeSecretResult: (callback) =>
```

```
    ipcRenderer.on("decode-secret-result", callback),
});
```

Додаток В. Ілюстративний матеріал

РОЗРОБКА ПРОГРАМИ РОЗПОДІЛУ СЕКРЕТУ
МІЖ ГРУПОЮ КОРИСТУВАЧІВ

Виконав ст. групи 1КІТС-20б Шкробот Є. С.
Керівник: к.т.н., професор кафедри МБІС Яремчук Ю. Є.

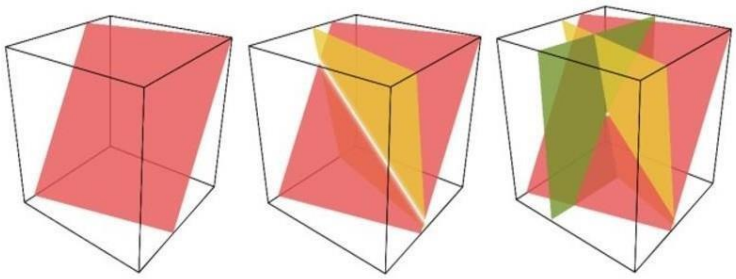
Мета

+ Дана бакалаврська дипломна робота присвячена розробці програми для розподілу секрету між групою користувачів на основі схеми Шаміра. Метою дослідження є створення ефективного механізму захисту конфіденційної інформації, який забезпечить високий рівень безпеки та гнучкість у використанні.

Актуальність задачі.

+ Розподіл секрету між групою користувачів є важливим інструментом у сфері інформаційної безпеки, який дозволяє захистити критичну інформацію від несанкціонованого доступу, зберігаючи її у розподіленій системі.

Розподілу секрету



Графічний приклад реалізації схеми Блекли в трьох вимірах

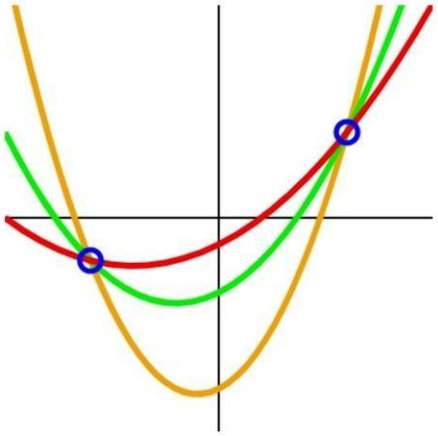
Принцип 1: Порогове значення

Порогове значення t визначає мінімальну кількість частин, необхідних для відновлення оригінального секрету. Це значення вибирається заздалегідь і забезпечує баланс між безпекою та доступністю

Принцип 2: Відновлення секрету

Відновлення секрету здійснюється за допомогою об'єднання t або більше частин. Цей процес зазвичай базується на математичних методах, таких як інтерполяція поліномів, у разі схеми Шаміра, або геометричних методах, як у схемі Блеклі.

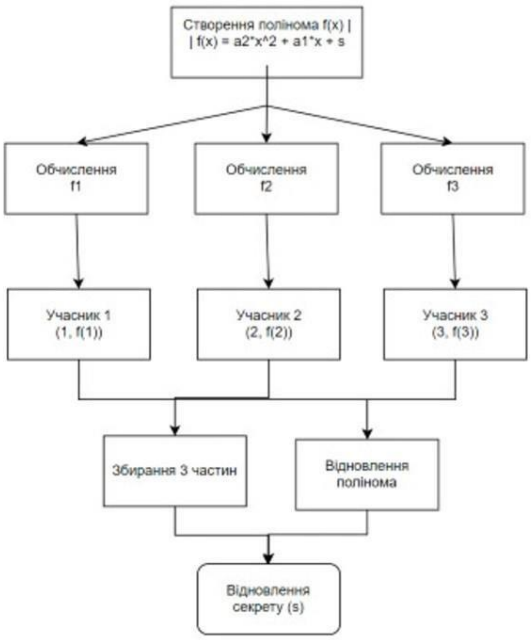
Схема Шамира



Ілюстрація ідеї Полімерної Інтерполяції порогової схеми Шаміра.

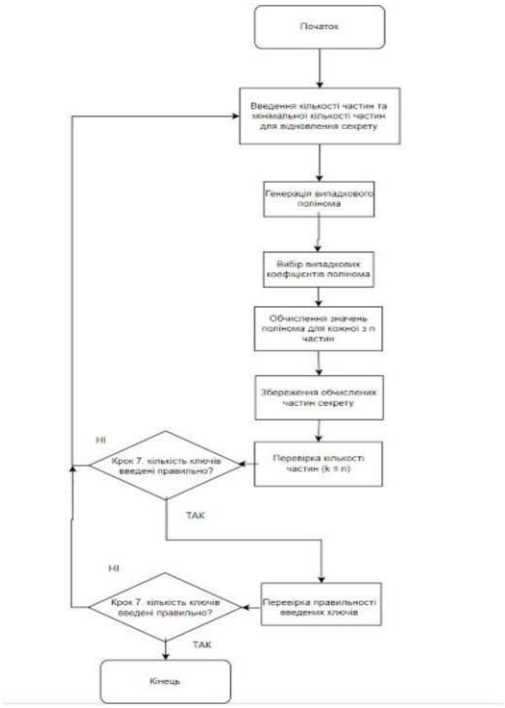
Якщо потрібно розділити секрет між n людьми так, щоб відновити його могли лише ті, у кого є k або більше частин секрету, то ми формуємо багаточлен ступеня k−1. Відновити цей багаточлен і вихідний секрет можна тільки за k точками. Кількість різних точок множини не обмежена (на практиці воно обмежується розміром числового поля, в якому ведуться розрахунки)

5



Ілюстрація схеми порогового розподілу секрету Шаміра

6



Детальний алгоритм роботи схеми Шамира

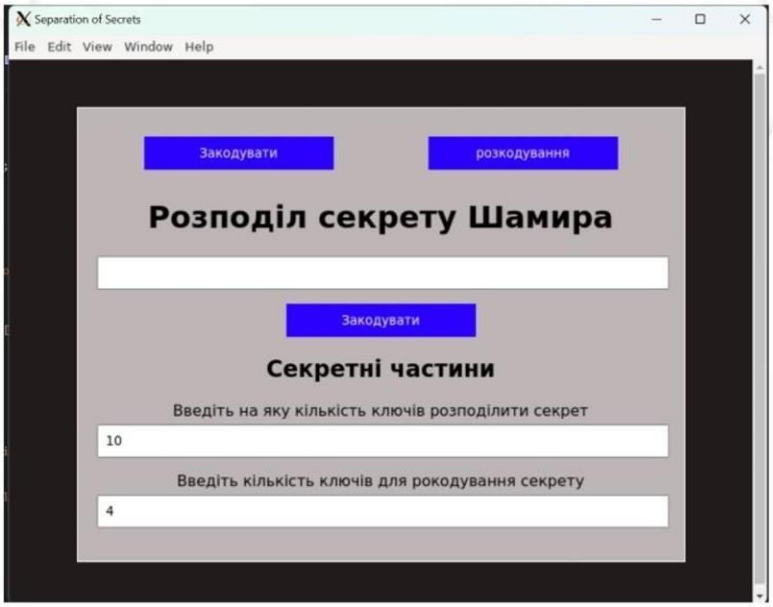
7

Технології використані для створення програми

Vs-code

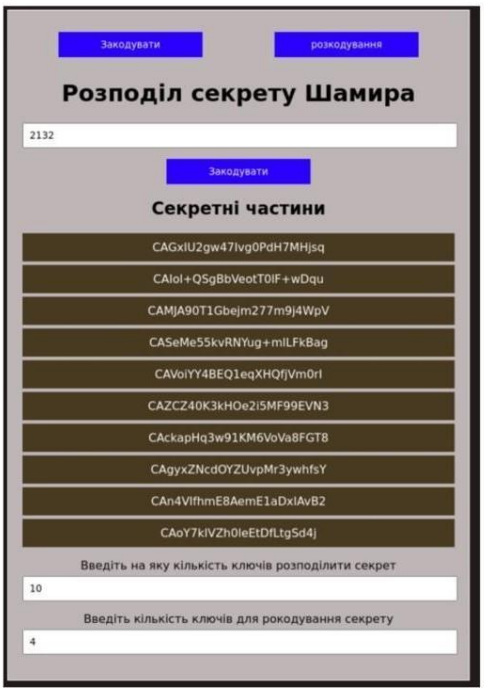
Vue.js

8



Меню кодування секрету

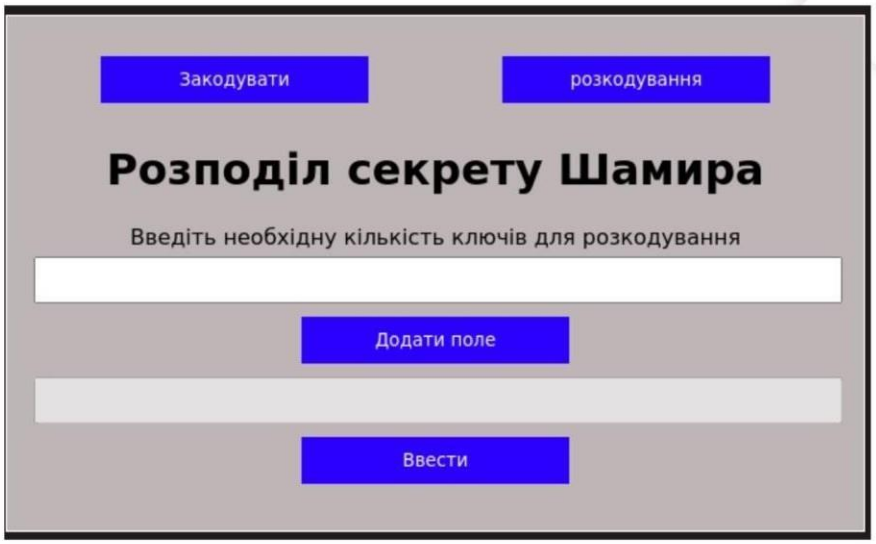
9



Закодоване повідомлення

10

Меню розкодування повідомлення



11

Висновки

У даній бакалаврській дипломній роботі:

+ проведено аналіз існуючих схем розподілу секрету та їх відновлення, включаючи криптосистеми з прогресивним розподілом секрету.
+ було розроблено алгоритм відновлення секрету, який передбачає зберігання засобу для відновлення на окремому пристрої з метою підвищення безпеки зберігання та ускладнення несанкціонованого доступуогресивним розподілом секрету.
+ Також було розроблено програмний додаток для розподілення секрету між групою користувачів з урахуванням принципів безпеки та ефективності.
+ Отже, результати дослідження та розробки показують, що використання схем розподілу секрету та відновлення може ефективно забезпечувати безпеку конфіденційної інформації та забезпечувати можливість відновлення даних в разі потреби.

12

Додаток Г.Протокол перевірки на антиплагіат

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ
ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програми розподілу секрету між групою користувачів

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ: Кафедра менеджменту та безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 92 % Схожість 8 %

Аналіз звіту подібності (відмітити потрібне):
1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Коваль Н.П.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Шкробот Є.С.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Яремчук Ю.Є.
(підпис) (прізвище, ініціали)