Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО—МОГИЛЯНСЬКА АКАДЕМІЯ»
Факультет інформатики
Кафедра математики

«Схеми таємного поширення інформації»

Курсова робота за спеціальністю
124 «Системний аналіз»

Керівник курсової роботи
д. фіз.—мат. наук, проф.
Олійник Б.В.
(прізвище та ініціали)

_____
(підпис)
“____”_____2020р.

виконав студент 1 курсу

Степанюк С.В.
(прізвище та ініціали)

Київ 2020

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО—МОГИЛЯНСЬКА АКАДЕМІЯ»

Факультет інформатики

Кафедра математики

ЗАТВЕРДЖУЮ
Зав. кафедри математики,
проф. Олійник Б. В.

_____

(підпис)

“____”_____2020р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студенту *Степанюку С.В.*. факультету *інформатики* *1* курсу МП
ТЕМА «Схеми таємного поширення інформації»
Вихідні дані:

1. Shamir, Adi (1979), "How to share a secret", Communications of the ACM
2. Akshayaram Srinivasan, Miao, Peihan, and Prashant Nalini Vasudevan. “Efficient Leakage Resilient Secret Sharing.” March 4, 2019.
3. Kumar, Ashutosh, et al. “Leakage-Resilient Secret Sharing”, 2018.
4. Srinivasan, Akshayaram, and Prashant Nalini Vasudevan. “Leakage Resilient Secret Sharing and Applications” August 18, 2019.

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

1. Розділення секрету
2. Означення
3. Приклади схем
4. Пропозиція реалізації схеми

Висновок

Список літератури

Дата видачі „___” _____ 2020 р . Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Тема: «Зберігання секрету при документообігу»

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	07.10.2019	
2.	Огляд наукової літератури за темою роботи.	08.10.2019	
3.	Опрацювання матеріалів	10.02.2020	
3.	Проведення аналізу	01.04.2020	
4.	Наведення прикладів та ілюстрацій	30.04.2020	
5.	Перевірка та погодження роботи	10.05.2020	
6.	Створення слайдів для доповіді та написання доповіді	10.05.2020	
7.	Остаточне оформлення роботи та слайдів.	11.05.2020	
8.	Захист курсової роботи	18.05.2020	

Студент _____***Степанюк С.В.***_____

Керівник _____***Олійник Б. В.***_____

“_____”_____2020 р.

Зміст

Анотація

В даній курсовій роботі проводилось дослідження та аналіз різноманітних систем розподілення секрету. Вона складається зі вступу, п’яти розділів, висновків та переліку використаної літератури. У першому розділі розглядається загальна схема розподілу секрету. У другому розділі даються необхідні означення. Цей розділ містить також важливу інформацію про основи та принципи роботи схем, а також їх основні складові. У третьому розділі розглядаються безпосередньо приклади схем. В останньому четвертому розділі наводиться пропозиція реалізації ускладненої схеми Шаміра для підвищення безпеки.

У висновках підбивається підсумок проведених досліджень та отриманих результатів. У переліку використаної літератури містяться джерела, які використовувалися при дослідженнях.

Ключові слова: Схема, секрет, розподілення секрету, порогова схема, дозволена коаліція, досконала схема, схема Шаміра.

Вступ

Таємне розподілення секрету – це один із способів розповсюдження секрету серед групи учасників, кожному з яких виділяється частина секрету. Секрет можна відновити лише тоді, коли достатня кількість, можливо, різних частин секрету об'єднана разом; окремі частини самостійно не приносять користі, тобто не дають можливості відновити інформацію.

Даний обмін був вперше запропонований Аді Шаміром у 1979 році. Такі схеми обміну ідеально підходять для зберігання дуже чутливої та дуже важливої інформації. Наприклад: ключі шифрування, таємні коди, коди банківських рахунків. Всі ці дані повинні зберігатись в таємниці, оскільки їх витік може бути небезепечним, однак, також не менш важливо, щоб ці дані не були втрачені. Використання лише традиційних методик шифрування часто не підходять для одночасного досягнення високого рівня конфіденційності та надійності. Адже при зберіганні ключа шифрування доводиться вибрати між збереженням оригіналу ключа в одному місці для максимальної секретності та збереженням декількох копій ключа в різних місцях для більшої надійності. Підвищення надійності ключа шляхом зберігання кількох копій знижує конфіденційність, створюючи додаткові можливості для атак; виникає більше можливостей для того, щоб копія потрапила в чужі руки. Схеми розподілення секрету вирішують цю проблему і дозволяють досягати досить високого рівня надійності та конфіденційності.

Таємні схеми обміну є важливими в хмарних обчислювальних середовищах. Таким чином, ключ може бути розподілений на багатьох серверах за допомогою порогового секретного механізму.

Безпека схем таємного розподілення секрету може також бути підвищена за рахунок постійної зміни способу розбиття секрету на частини.

Розділ 1. Розділення секрету

Розділення секрету є новою альтернативою для захищеного аутсорсингу даних. Це дозволяє уникати необхідності трудомісткого процесу пов'язаного з управлінням ключами. Дані також повинні бути захищені від ненадійних постачальників хмарного сховища. Безпечне розділення секрету досягається шляхом спільного використання даних та розподілення їх між великою кількістю серверів. Дані отримані з порогової кількості серверів використовуються для реконструкції початкових даних. Зазвичай, недоцільно розповсюджувати дані між великою кількістю серверів. Необхідно домогтися оптимального вибору для використання тієї чи іншої порогової схеми розділення секрету (наприклад 2,3 або 2,4), де дані розподіляються у вигляді частин секрету між трьома-чотирма серверами, а частини секрету з будь-яких двох використовуються для відбудови початкових даних. Це забезпечує надійність, ефективність та безпеку.

Така схема обміну дозволяє розділяти секрет між набором сторін таким чином, щоб деяка уповноважена підмножина сторін могла відновити секрет, а будь-яка несанкціонована підмножина сторін не могла дізнатися жодної інформації про секрет. Схема спільного використання розділяти секрету також вимагає збереження секретності проти кожного несанкціонованого набору сторін, навіть при отриманні деякого обмеженого витоку ці сторони не можуть відновити власне секрету. Витоки можуть бути
локальним, якщо кожен з них обчислюється незалежно від інших частин секрету.

Стійкість до витоків широко вивчається і розглядається, як бажана властивість при різних налаштуваннях та криптографічних примітивах. В більшості розглядають розділення секрету, яке є стійким до витоків, щоб секрет залишався прихованим від несанкціонованого доступу деякої підмножини сторін, навіть якщо вони мають доступ до певної невеликої кількості інформації про частини секрету решти сторін.

2. Означення

Розділення секрету – криптографічний термін, який описує будь-який зі способів розподілу секрету серед певної кількості сторін, при якому кожна зі сторін отримує свою частину. Секрет можна відтворити тільки за наявності коаліції сторін основної групи, причому до коаліції має входити не менше за деяку спочатку відому кількість сторін. Дані схеми застосовують, коли існує можливість компроментації однієї чи кількох сторін, в той час як імовірність змови недобросовісних сторін мізерно мала.

2.1 Основні складові

Дані схеми мають дві основні складові:

- Розподіл секрету
- Відновлення секрету

Розподіл – це формування частин секрету та поширення їх між усіма сторонами, таким чином розподіляється відповідальність за секрет. Функції для реалізації повинні бути рандомізовані для забезпечення максимальної надійності.

Відновлення – має забезпечувати відтворення секрету за наявності наперед відомої кількості сторін. Функції для відновлення можуть бути детерміністичними, адже безпека вже гарантується коаліцією сторін.

3. Приклади схем

3.1 Найпростіша схема (n, n).

Нехай маємо групу з **t** людей і деяке повідомлення **s** довжини **n**. Якщо випадковим чином підібрати такі повідомлення $s_1,\ldots, s_t$, які в сумі даватимуть **s**, а потім розподілити ці повідомлення між усіма сторонами – отримаємо ситуацію, коли прочитати повідомлення можна буде лише за умови присутності всіх сторін.

Слабкість схеми: якщо буде встрачений хоча б один із членів групи – відновити секрет буде неможливо.

3.2 Порогова схема (t, n)

Під час процедури розподілу кількість сторін які необхідні для відновлення секрету **t** може не дорівнювати кількості всіх учасників **n.** Такі схеми називають пороговими **(t, n)**.

Ефективні порогові схеми можуть бути дуже корисними для управління криптографічними ключами. Задля захисту даних, їх можна зашифрувати, але для захисту ключа шифрування потрібен інший метод (подальше шифрування може лише змінити проблему, але ніяк не вирішить її). Найбезпечніша схема управління ключами пропонує зберігати ключ в єдиному, добре захищеному місці (в комп’ютері, людському мозку чи сейфі). Ця схема є дуже ненадійною, оскільки будь-яка проблема (поломка комп'ютера, раптова смерть або саботаж) може зробити ключ недоступним. Очевидною альтернативою є зберігання кількох копій ключа в різних місцях, але з іншого боку це підвищує небезпеку (втручання в комп’ютер або ж зрада чи помилка людини). Використовуючи **(t, n)** порогову схему з **n = 2t- 1** отримаємо дуже надійну схему управління ключами адже початковий ключ можна відновити, навіть коли **n / 2 = t- 1** з **n** частин знищені, але наші противник не може реконструювати ключ навіть тоді, коли **n / 2 = t- 1** з решти **t** штук порушать безпеку.

Дозволена коаліція – це така коаліція, яка має достатню кількість учасників достатніх для відновлення секрету

Досконала схема – це така схема поділу секрету, в якій дозволена коаліція може відновити секрет однозначно, при тому як недостатня кількість сторін не отримує жодної апостеріорної інформації про можливе значення секрету.

3.3 Схема Шаміра (класична схема)

Дана схема базується на інтерполяційних поліномах. З курсу алгебри відомо, що щоб задати многочлен степеня **t** знадобиться **t+1** точок. Можна побудувати інтерполяційний многочлен у формі Ньютона або Лагранжа. Для відновлення секрету **t** учасниками при розподілі його зашифровують у формулу многочлена степеня **(t+1)** над деяким скінченним полем **G.** Для того, щоб відновити такий многочлен необхідно знати його значення в усіх **t** точках, причому, знаючи значення в менш ніж **t** точках однозначно відновити початковий многочлен – неможливо. Для схеми Шаміра використовується інтерполяційний многочлен у формі Лагранжа.

Опис алгоритму: Нехай маємо скінченне поле **G**. Фіксуємо в ньому **n** різних ненульових несекретних елементів. Кожен з цих елементів приписується до певного члена групи. Далі обираємо довільний набір з **t** елементів цього поля, з яких і складатиметься многочлен **f(x)** степеня **t-1**, $\mathbf{1} < \boldsymbol{t} \leq \boldsymbol{n}$ над цим полем. Після отримання многочлена необхідно обчислити його значення в несекретних точках і розподілити отримані результати між відповідними членами групи.

Для відновлення секрету можна використати інтерполяційну формулу, наприклад формулу Лагранжа.

Важлива перевага класичної схеми полягає в масштабованості. Для збільшення кількості членів у групі достатньо просто додати певну кількість несекретних

елементів до вже існуючого набору. Причому компрометація однієї сторони володіння секрету просто переводить цю порогову схему з **(t, n) в (t-1, n-1).**

3.4 Деякі корисні властивості порогової схеми Шаміра (t, n) :

1. **Безпека:** інформаційно-теоретична безпека.
2. **Мінімалізм:** розмір кожної частини секрету не перевищує розміру початкового секрету.
3. **Розширюваність:** коли **t** залишається фіксованим, нові частини секрету можна додавати та видаляти без жодного впливу на інші.
4. **Динамічність:** Безпека може легко бути покращена, не змінюючи при цьому секрет. Це можливо досягти варіацією поліномів час від часу (зберігаючи той самий вільний член) та генеруючи нові частини секрету для учасників.
5. **Гнучкість:** У випадках, де важлива ієрархія, можна надати кожному учаснику різну кількість частин секрету у відповідності до важливості того чи іншого члена організації. Наприклад, президент може відкрити сейф самостійно (має коаліцію частин секрету), віце-президенти можуть зробити чи лише удвох, тоді як для його розблокування сейфу менеджерами потрібна присутність мінімум трьох людей.

Слабкістю схеми Шаміра є проблема, що виникає при перевірці правильності частин секрету під час процесу відновлення, яка також відома як перевірюване поширення секрету, даний процес спрямований на те, щоб перевірити, що учасники чесні та не поширюють підроблені частини секрету**.**

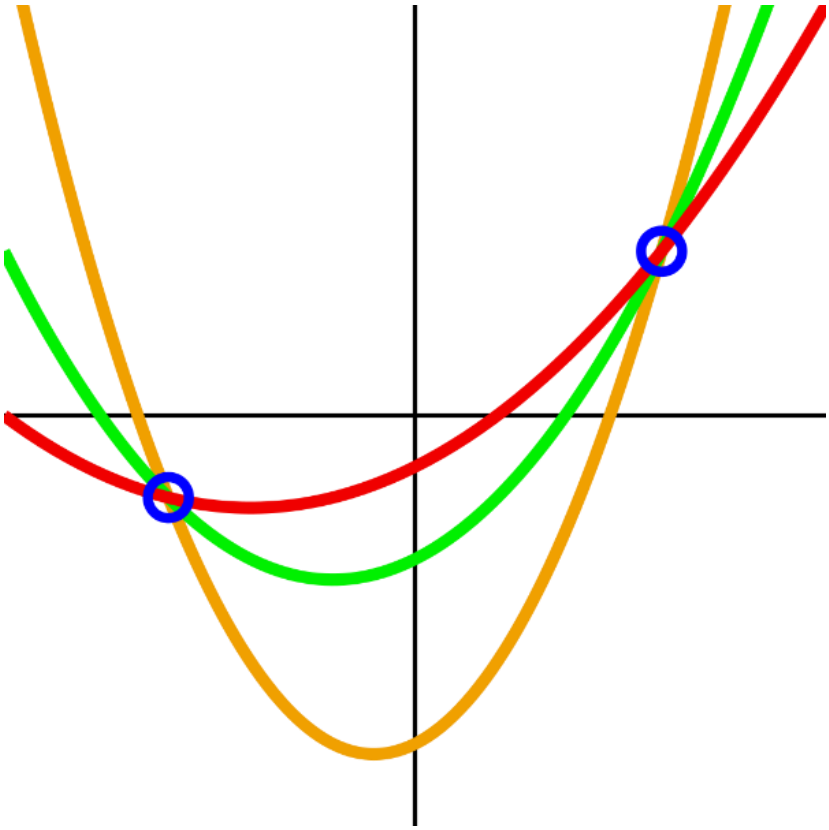

Рис. 1 Ілюстрація підходу

Через дві точки можна провести нескінченну кількість поліномів, через 3 точки – лише один. Даний рисунок ілюструє підхід вцілому, адже схема Шаміра використовує поліноми над скінченним полем, і це недоцільно відображати у 2-вимірному просторі.

Означення 3.1

Нехай **Share (функція розподілу):** $\{\mathbf{0},\mathbf{1}\}^{\boldsymbol{t}} \rightarrow (\{\mathbf{0},\mathbf{1}\}^{\boldsymbol{l}})^{\boldsymbol{n}}$ - будь-яка **рандомізована** функція відображення секрету довжиною в **k-біт** в **n** частин секрету довжини **l** бітів. Нехай **Rec (функція відновлення) :** $(\{\mathbf{0},\mathbf{1}\}^{\boldsymbol{l}})^{\boldsymbol{n}}$ – деяка **детерміністична** функція, що відображає колекцію частин секрету назад в секрет.

Така пара **(Share, Rec)** називається таємною схемою обміну **(t, n)** для відображення секрету довжиною в **k-біт** в частини секрету довжини **l-**біт, якщо виконуються:

- **Ідеальна коректність :** будь-які **(t, n)** частин секрету можуть бути використані для відновлення секрету правильно.
  Для будь-якого секрету $a \in \{0,1\}^k$, для будь-якого набору $\boldsymbol{T} \subseteq [\boldsymbol{n}]$**,** за умови, що $|\mathbf{T}| \geq \mathbf{t}$**,**
  $$\boldsymbol{Pr}[\boldsymbol{Rec}(\boldsymbol{Share}(\boldsymbol{a})_{\boldsymbol{T}}) = \boldsymbol{a}] = \mathbf{1}$$
  де ймовірність переважає рандомізованість функції розподілу,
  та $\boldsymbol{Share}(\boldsymbol{a})_{\boldsymbol{T}}$ позначає обмеження **n** частин секрету згенерованих за допомогою $\boldsymbol{Share}(\boldsymbol{a})$ до тих, що визначаються набором **T.**
- **Ідеальна секретність :** Група з менш ніж **t** учасників не може дізнатися жодної інформації про секрет. Більш формально, для будь-яких двох секретів **a** та $\mathbf{b} \in \{0,1\}^k$, будь-який набір $\boldsymbol{U} \subseteq [\boldsymbol{n}]$**,** за умови, що $|\mathbf{U}| \geq \mathbf{t}$**,** функції $\boldsymbol{Share}(\boldsymbol{a})_{\boldsymbol{U}}$ та $\boldsymbol{Share}(\boldsymbol{b})_{\boldsymbol{U}}$ матимуть однаковий розподіл.

3.5 Таємне поширення секрету на основі операції ⊕ (XOR)

Пригадаємо звичайну **(n, n)** схему, але побудуємо її на основі **⊕ (XOR).**

Зауважимо, що секрет має довжину в **a**-біт, **a ≥ 1.**

- **(Функція розподілу $XOR\text{Share}_n$) :** Нехай $XOR\text{Share}_n$ **:**

$$\{0,1\}^a \rightarrow \bigoplus_{i\in[n]}\{0,1\}^a$$

деяка рандомізована функція розподілу.

На вхід подається секрет $s \in \{0,1\}^a$, і рівномірно розподіляється на перші **n-1** частин секрету, назвемо їх $s_1, \dots, s_{n-1}$, такі, що кожен $s_i \in \{0,1\}^a$**.**

Останню частину s_nобраховуємо, використовуючи секрет **s** та уже обраховані попередньо частини секрету наступним чином: $s_n \leftarrow s \oplus s_1 \oplus \dots \oplus s_{n-1}$**.**

Врешті на виході маємо **n** частин секрету $s_1, \dots, s_n$.

- **(Функція відновлення $XOR\text{Rec}_n$) :** Нехай $XOR\text{Rec}_n$ **:**

$$\bigoplus_{i\in[n]}\{0,1\}^a \rightarrow \{0,1\}^a$$

деяка детерміністична функція відновлення.

На вхід подається **n** частин секрету $s_1, \dots, s_n$, виконуються обрахунки секрету $s \leftarrow s_1 \oplus \dots \oplus s_n$ **,** і на виході маємо власне секрет **s**.

Лема 3.5.1

Для секрету довжини **a ≥ 1** біт ($XOR\text{Share}_n$; $XOR\text{Rec}_n$) – це ні що інше, як **(n, n, 0)** – схема поширення секрету.

Крім того, дана схема має корисну властивість, а саме, враховує секрет та всі частини, крім однієї. Залишкові частини секрету можуть бути ефективно обраховані.

Лема 3.5.2

Нехай ($XOR\text{Share}_2$; $XOR\text{Rec}_2$) є **(2, 2, 0)** – схемою поширення для однобітових секретів. Тоді для будь-яких **m**, $sh_1, sh_2 \in \{0,1\}$**,** якщо **m** можна знайти з функції $XOR\text{Rec}_2(sh_1, sh_2)$**, то** sh_1можна знайти з $XOR\text{Rec}_2(m, sh_2)$**.**

4. Пропозиція реалізації схеми Шаміра з додатковим захистом частин секрету

З метою додаткового захисту при розподіленні секрету з використанням класичної схеми Шаміра пропонуємо скористатися асиметричною технікою шифрування **RSA,** шифруючи кожну частин окрему. Таким чином, використовуючи **RSA** згенеруємо **public** та **private** ключі, за допомогою яких і шифруватимемо частини секрету. Відповідно, під час процесу розподілу секрету кожен учасних схеми отримуватиме не власне частину, а зашифровану, а отже, відкидається можливість відновлення секрету при викраденні зловмисником необхідної кількості частин секрету, або ж зради учасників. Адже навіть за умови створення коаліції для відновлення секрету зробити це буде неможливо без поперенього дешифрування кожної частини секрету. Також, за рахунок застосування додаткового шифрування власне процес розповсюдження частин секрету може здійснюватись навіть з використанням відкритих каналів, оскільки зашифровані частини, навіть за умови ймовірного витоку інформації при передачі даних ніяк не полегшують процес відновлення секрету та не несуть жодної загрози цілісності самого секрету.

Спершу створимо сам секрет та розіб’ємо його на частини

```
[ ] import random
    import functools
    import rsa
    import Crypto
    from Crypto.PublicKey import RSA
    from Crypto import Random
    import ast


    MersPrime = 2 ** 127 - 1
    RandInt = functools.partial(random.SystemRandom().randint,0)
```

```
def createSecret(min):
  secret = [RandInt(MersPrime - 1) for i in range(min)]
  return secret

def evaluation(polynom, x):
  accval=0
  revpolynom = reversed(polynom)
  for i in revpolynom:
    accval = accval * x
    accval = accval + i
    accval = accval % MersPrime
  return accval

def createShares(secret,nshares):
  shares = [(i, evaluation(secret, i))
              for i in range(1, nshares + 1)]
  return shares
```

```
[ ]  def main():
         secret = createSecret(min=3)
         shares = createShares(secret, nshares=5)


         print('Secret: ',
               secret[0])
         print('Shares:')
         for share in shares:
               print('   ', share)
```

Отримаємо наступне:

```
Secret:  484687229586353813404802762306272094396498242733537586450461632011654568213 45
Shares:
   (1, 366984513475066187910800389457664267285695563640692338406558785812072182314 3)
   (2, 432199604141304984800048676153786506542351366457863989355760841590491709606 19)
   (3, 513269795594586957195998823843453255305143826358516212601200680960376745938 39)
   (4, 279909025707352535978930482014766673016946936066025903576975396690862327228 03)
   (5, 311077740666182698266698575711166298944110618908595882480372908821514101674 78)
```

Тепер використовуючи **RSA** зашифруємо отримані частини секрету:

Спершу згенеруємо **public** та **private** ключі та зашифруємо отримані частини секрету**:**

```
        random_generator = Random.new().read
        key = RSA.generate(1024, random_generator)
        publickey = key.publickey()

        for share in shares:
              share = publickey.encrypt(share, 32)
```

Для прикладу перша частина секрету **до шифрування:**

(1,366984513475066187910800389457664267285695563640692338406558785812
0721823143)

Та **після:**

bhXBGeLy+vkJdVROD1Due+MDz4AUGIe897MWQLtjYNOL+iCdAcyviMCvXt/
0//aLsM2juVM8tQHLNOyvjteblSnj8WgmWd0uFyJLt7yQqb2M0qPmlvRVaATrKQ
WE1J3EnezIUwYRAumEHtgzbqZwUtqM/PXmyiWVdoDN+y2caVQ=

Тепер ці частин секрету можна розповсюджувати між учасниками схеми, а ключі залишати в надійному місці, при цьому копії ключів можна роздати деяким окремим учасникам.

Тепер перед тим як відновлювати секрет потрібно буде знову ж, як і в звичайній схемі Шаміра зібрати коаліцію учасників, після чого розшифрувати їхні частини секрету і тільки після цього відновлювати секрет.

```
    for share in shares:
          share = key.decrypt(ast.literal_eval(str(encrypted)))
```

```
def recoverSecret(shares):
  x = 0
  xs, ys = zip(*shares)
  k = len(xs)
  def inProd(points):
    count = 1
    for p in points:
      count = count * p
    return count;
  numerators = []
  denominators = []
  for i in range(k):
    extra = list(xs)
    current = extra.pop(i)
    numerators.append(inProd(x - e for e in extra))
    denominators.append(inProd(current - e for e in extra))
  denominator = inProd(denominators)
  numerator = sum([numDenPrime(numerators[i] * denominator * ys[i] % MersPrime, denominators[i])
              for i in range(k)])
  return (numDenPrime(numerator, denominator) + MersPrime) % MersPrime

def numDenPrime(numerator,denominator):
    invariant, _ = euclidean(denominator,MersPrime)
    return numerator * invariant

def euclidean (a,b):
  x = 0
  y = 1
  fx = 1
  fy = 0
  while b!=0:
    q = a // b
    a, b = b, a % b
    x, fx = fx - q * x, x
    y, fy = fy - q * y, y
  return fx, fy
```

Врешті, отримуємо відновлений секрет:

```
Recovered Secret  484687229586353813404802762306272094396498242733537586450461632011654568213 45

Process finished with exit code 0
```

В прикладі була продемонстрована **(t, n)** – схема розподілення секрету **при t = 3, n=5.** І секрет відновлювався з **3-х** частин секрету (мінімально достатніх для коаліції) аналогічно працюватиме і для **4-х** та **5-ти**.

Якщо спробувати відновити секрет маючи лише **2** частини секрету

```
79          print('Recovered Secret ',
80                recoverSecret(shares[:2]))
81

Shamir
/Users/stasstepaniuk/venv/bin/python /Users/stasstepaniuk/Documents/Shamir.py
Secret:  20167156123036918554975902761094332159201257859537494301005833412214631021288
Shares:
   (1, 31565749963126225570571223897784813152130291111137836237066402510499321741586)
   (2, 18832755854384147582532748073497795227674705525162208958165641332914038083746)
   (3, 39864218415468782302645967792577232312469493434430894484032341883415344867735)
   (4, 36764093027722032019125390550679170479879662506123610794937712158046677273586)
   (5, 95323796911438967319710163478036097299052127402403578908817521568080353012 99)
Recovered Secret  44298744071868303558609699722071831076585876697113463515967163688084605399426

Process finished with exit code 0
```

Бачимо, що відновлений секрет та початковий не співпадають.

Висновок

В цій роботі було розглянуто задачу розподілення секрету. В роботі запропоновано порогову **(3, 5)** - схему для розподілення секрету. Дана схема базується на стандартній пороговій схемі Шаміра, проте з додатковим шифруванням частин секрету перед розподілом їх між учасниками. Для цього використовується асиметрична система шифрування **RSA**.

Відповідно, під час процесу розподілу секрету кожен учасних схеми отримуватиме не власне частину секрету, а зашифровану, а отже, відкидається можливість відновлення секрету при викраденні зловмисником необхідної кількості частин секрету, або ж зради учасників. Адже навіть за умови створення коаліції для відновлення секрету зробити це буде неможливо без поперенього дешифрування кожної частини секрету. Процес відновлення секрету з нерозшифрованих частин є неможливим навіть за умови збору всіх частин, а не лише мінімальної необхідної коаліції. Також, за рахунок застосування додаткового шифрування власне процес розповсюдження частин секрету може здійснюватись навіть з використанням відкритих каналів, оскільки зашифровані частини, навіть за умови ймовірного витоку інформації при передачі даних ніяк не полегшують процес відновлення секрету та не несуть жодної загрози цілісності самого секрету.

Список літератури

5. Shamir, Adi (1979), "How to share a secret", Communications of the ACM
6. Akshayaram Srinivasan, Miao, Peihan, and Prashant Nalini Vasudevan. “Efficient Leakage Resilient Secret Sharing∗.” March 4, 2019.
7. V.P, Binu, and Sreekumar A. “Simple and Efficient Secret Sharing Schemes for Sharing Data and Image.”, Feb 26, 2015.
8. Kumar, Ashutosh, et al. “Leakage-Resilient Secret Sharing” , 2018.
9. Srinivasan, Akshayaram, and Prashant Nalini Vasudevan. “Leakage Resilient Secret Sharing and Applications∗.” Leakage Resilient Secret Sharing and Applications∗, August 18, 2019.
10. “Secret Sharing.” Wikipedia, Wikimedia Foundation, 26 Mar. 2020,