Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра математики

Магістерська робота

освітній супінь – магістр

на тему: **«АНАЛІЗ СХЕМ РОЗПОДІЛУ СЕКРЕТУ»**

Виконав: студент 2-го року навчання,
освітньо-наукової програми
«Системний аналіз»,
спеціальності 124 Системний аналіз
Степанюк Станіслав Вікторович
Керівник: Олійник Б.В.
д. фіз.-мат. наук, проф.
Рецензент _____
(прізвище та ініціали)

Магістерська робота захищена
з оцінкою _____

Секретар ЕК _____
“____”_____20___р.

Київ 2021

Зміст

Анотація

В даній магістерській 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 четвертому розділі наводиться пропозиція реалізації ускладненої схеми Шаміра для підвищення безпеки.
В останньому п'ятому розділі надається пропозиція нового алгоритму розподілу секрету.
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. [5,с.3]

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 мають бути детерміністичними, а безпека гарантується коаліцією сторін.

Досконалість – забезпечується, коли шифротекст не передає жодної інформації про зміст відкритого тексту. Фактично це означає, що незалежно від того, скільки б частин шифротексту не мав зловмисник, він жодним чином не зможе дізнаєтися з них про те, якими були відкритий текст та ключ. Можна довести, що будь-яка така схема повинна використовувати принаймні стільки матеріалу для ключа, скільки є відкритого тексту для шифрування. Що стосується ймовірностей, це означає, що розподіл ймовірностей можливих відкритих текстів не залежать від шифротексту. [2,с.1-2]

CPA атака – атака на основі підібраного відкритого тексту – одна з основних криптоаналітичних атак при якій припускається, що нападник може отримати шифротекст для довільного відкритого тексту. Метою цієї атаки є отримання інформації про секрет.

Під час **CPA атаки** криптоаналітик може вибрати довільний відкритий текст, який потрібно зашифрувати, і отримати відповідний шифротекст. Він намагається або отримати секретний ключ або, як альтернативу, створити алгоритм, який дозволив би розшифрувати певні шифротексти, зашифровані цим ключем (при цьому навіть не знаючи власне секретного ключа). Зловмисник, в свою чергу може отримати більше інформації про секретний ключ, а також власне й про всю систему на яку здійснюється атака, оскільки він може вибрати будь-який текст, який буде оброблений шифром. Він може аналізувати поведінку системи та виводити зашифрований текст на основі вхідних даних будь-якого вигляду.

Під час розбиття детермінованих шифрів за допомогою відкритого ключа зловмисник може легко створити базу даних із популярними шифротекстами, наприклад з популярними запитами до сервера. Після цього він зможе знайти значення багатьох перехоплених шифротекстів, просто порівнявши їх із власними записами в базі даних.

Найвідоміші **CPA атаки** були здійснені криптоаналітиками під час Другої світової війни для зламу німецьких шифрів системи Enigma.

CPA атаки бувають двох видів:

• **Пакетна CPA** – атака, при якій зловмисник обирає всі відкриті тексти перш ніж він може проглянути будь-який із відповідних шифротекстів.

• **Адаптивна CPA** – атака, під час якої зловмисник має можливість багато разів вибирати відкритий текст для шифрування. Замість того, щоб використовувати один великий блок тексту, він може вибрати менший, отримати його шифротекст, а потім на основі цієї відповіді вибрати інший і т.д. Це дозволяє йому набагато детальніше дослідити систему, на яку здійснюється атака. [7]

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 наш супротивник 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. [1,с.2]

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**, $1 < \boldsymbol{t} \leq \boldsymbol{n}$ над цим полем. Після отримання многочлена необхідно обчислити його значення в несекретних точках і розподілити отримані результати між відповідними членами групи. [1,с.1-2]

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. [1,с.1-2]

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**.**

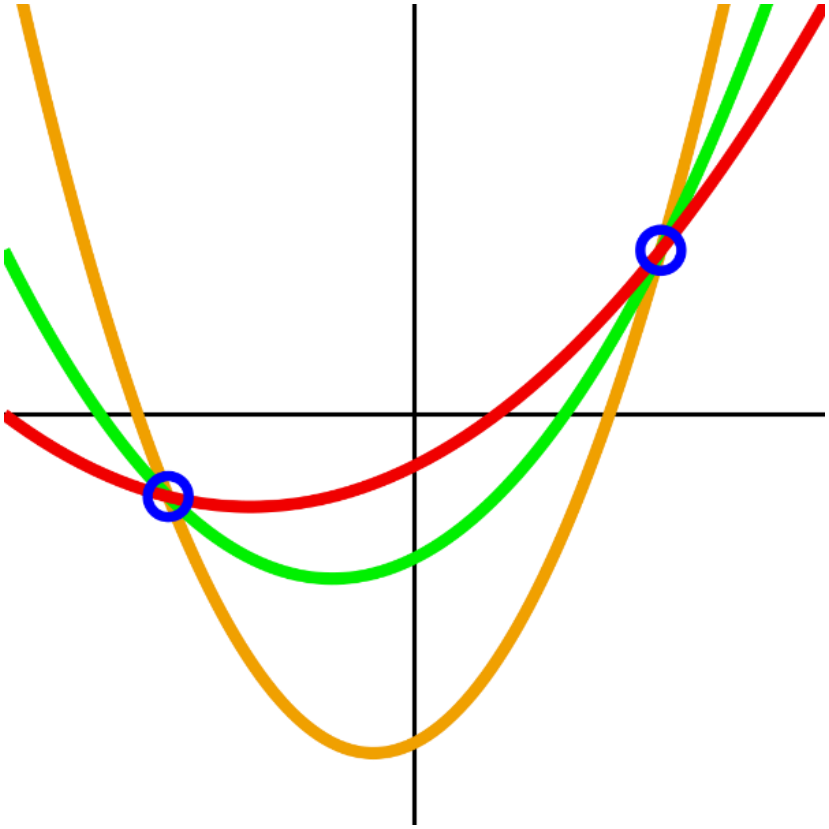

Рис. 1 Ілюстрація підходу

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. [6]

Означення 3.1

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

Така пара **(Share, Rec)** називається секретною 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 $|\boldsymbol{T}| \geq \boldsymbol{t}$,

  $$\boldsymbol{Pr}[\boldsymbol{Rec}(\boldsymbol{Share}(\boldsymbol{a})_{\boldsymbol{T}}) = \boldsymbol{a}] = 1$$

  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 **b** $\in \{0,1\}^k$, будь-який набір $\boldsymbol{U} \subseteq [\boldsymbol{n}]$, за умови, що $|\boldsymbol{U}| \geq \boldsymbol{t}$, функції $\boldsymbol{Share}(\boldsymbol{a})_{\boldsymbol{U}}$ та $\boldsymbol{Share}(\boldsymbol{b})_{\boldsymbol{U}}$ матимуть однаковий розподіл. [4,с.1]

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** $\geq$ **1.**

- **(Функція розподілу $XORShare_n$) :** Нехай $XORShare_n$: $\{0,1\}^a \rightarrow \bigoplus_{i\in[n]} \{0,1\}^a$ деяка рандомізована функція розподілу.

  На вхід подається секрет $s \in \{0,1\}^a$, і рівномірно розподіляється на перші **n-1** частин секрету, назвемо їх $s_1, \ldots, s_{n-1}$, такі, що кожен $s_i \in \{0,1\}^a$**.** Останню частину s_nобраховуємо, використовуючи секрет **s** та уже обраховані попередньо частини секрету наступним чином: $s_n \leftarrow s \oplus s_1 \oplus \ldots \oplus s_{n-1}$.

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

- **(Функція відновлення $XORRec_n$) :** Нехай $XORRec_n$:

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

деяка детерміністична функція відновлення.

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

Лема 3.5.1

Для секрету довжини **a** $\geq$ **1** біт ($XORShare_n; XORRec_n$) – 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 ($XORShare_2; XORRec_2$) 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 $XORRec_2(sh_1, sh_2)$**, то** sh_1можна знайти з $XORRec_2(m, sh_2)$**. [4,с.15-17]**

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 попереднього 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  48468722958635381340480276230627209439649824273353758645046163201165456821345

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:  201671561230369185549759027610943321592012578595374943010058334122146310212 88
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.

5. Пропозиція нового алгоритму

Використання схем обміну секретом є альтернативним способом безпечної передачі даних. Це дозволяє розповсюдити розділений секрет серед групи користувачів схеми. Відновленням секрету займається в свою чергу порогова кількість користувачів, які беруть участь у процесі, тоді як кожна частина секрету або ж менша група користувачів схеми не має можливості отримати жодної інформації про сам секрет. Існує багато схем та алгоритмів розподілення секрету.

Оскільки ми вже коротко розглянули, дослідили та проаналізували принципи роботи та основні компоненти базових схем та одного з наших “ускладнених” алгоритмів, перейдемо до представлення нашого нового методу. Ми назвали цей алгоритм **"Перехідною схемою розподілу"**.

Для початку дамо необхідні нам теоретичні поняття.

Секретом, з яким ми працюємо в цьому методі є деяке число $\boldsymbol{s}$. Це число є сумою всіх координат деякого вектора $\boldsymbol{v} = (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ тривимірної декартової системи координат.

Очевидно, що існує деяка нескінченна множина векторів, сума координат яких дорівнює одному й тому ж числу.

В свою чергу координати таких векторів обираються натупним чином:

- Координати $\boldsymbol{x_i}, \boldsymbol{y_i}$ – деякі випадкові числа;
- Координата $\boldsymbol{z_i}$ **обчислюється як** $\boldsymbol{z_i = s - x_i - y_i}$

Визначимо також функцію $\boldsymbol{SumCoord(v)}$**.** Це деяка функція, що обраховує суму всіх координат заданого вектора.

Тобто $\boldsymbol{SumCoord(v)}$ = $\boldsymbol{x_v + y_v + z_v}$ = **s**

Приклад:

Нехай $\boldsymbol{s} = 3$
$\boldsymbol{v}_1 = (9, -6, 0)$
$\boldsymbol{v}_2 = (2, 1, 0)$
$\boldsymbol{v}_3 = (5, -1, -1)$

В усіх цих випадках бачимо, що сума координат кожного з векторів $\boldsymbol{SumCoord}(v_1) = \boldsymbol{SumCoord}(v_2) = \boldsymbol{SumCoord}(v_3) = 3.$

Алгоритм "Перехідної схеми розподілу":

1. На початку ми маємо вектор, координати якого записані в нестандартному базисі $(\boldsymbol{e_1}..\boldsymbol{e_n})$ простору $\boldsymbol{R^n}$.
2. Оберемо інший нестандартний базис $(\boldsymbol{b_1}..\boldsymbol{b_n})$ в просторі $\boldsymbol{R^n}$. Зрозуміло, що кожен з векторів нового базису має власні координати у базисі $(\boldsymbol{e_1}..\boldsymbol{e_n})$:

$$\begin{gathered}\boldsymbol{b_1 = t_{11}\cdot e_1 + t_{21}\cdot e_2 + .. + t_{n1}\cdot e_n}\\ \boldsymbol{b_2 = t_{12}\cdot e_1 + t_{22}\cdot e_2 + .. + t_{n2}\cdot e_n}\\ \cdots\cdots\cdots\cdots\cdots\cdots\\ \boldsymbol{b_n = t_{1n}\cdot e_1 + t_{2n}\cdot e_2 + .. + t_{nn}\cdot e_n}\end{gathered}$$

3. Перейдемо до нового базису, побудувавши перехідну матрицю $\boldsymbol{T}$, стовпцями якої є координати векторів нового базису в старому базисі

$$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{t_{11}} & \boldsymbol{t_{12}} & \cdots & \boldsymbol{t_{1n}} \\ \boldsymbol{t_{21}} & \boldsymbol{t_{22}} & \cdots & \boldsymbol{t_{2n}} \\ \cdots & \cdots & \cdots & \cdots \\ \boldsymbol{t_{n1}} & \boldsymbol{t_{n2}} & \cdots & \boldsymbol{t_{nn}} \end{pmatrix}$$

Тобто $(\boldsymbol{b_1}, \boldsymbol{b_2}, \ldots, \boldsymbol{b_n}) = (\boldsymbol{e_1}, \boldsymbol{e_2}, \ldots, \boldsymbol{e_n}) \cdot \boldsymbol{T}$

4. Скориставшись отриманою матрицею подамо наш вектор $\boldsymbol{v}$ в базисі $(\boldsymbol{b_1}..\boldsymbol{b_n})$. Нехай це буде $(\boldsymbol{u_1}..\boldsymbol{u_n})$.

5. Після цього кожному користувачу схеми роздаємо по одному числу $(\boldsymbol{u_i})$, та по одному вектору $(\boldsymbol{b_i})$.

Відновлення "Перехідної схеми розподілу":

1. Для відновлення необхідно зібрати залучених учасників схеми та побудувати матрицю переходу $\boldsymbol{T}$ з їхніх векторів $\boldsymbol{b_i}$.

$$\boldsymbol{T}\begin{pmatrix}\boldsymbol{x}_1\\ \dots\\ \boldsymbol{x_n}\end{pmatrix} = \begin{pmatrix}\boldsymbol{u}_1\\ \dots\\ \boldsymbol{u_n}\end{pmatrix}$$

$$\boldsymbol{T}\cdot\overline{\boldsymbol{x}} = \overline{\boldsymbol{u}}$$

Домножимо праву і ліву частини на $\boldsymbol{T}^{-1}$

$$\boldsymbol{T}^{-1}\cdot\boldsymbol{T}\cdot\overline{\boldsymbol{x}} = \boldsymbol{T}^{-1}\cdot\overline{\boldsymbol{u}}$$

Звідси $\overline{\boldsymbol{x}} = \boldsymbol{T}^{-1}\cdot\overline{\boldsymbol{u}}.$

Також відомо, що $\boldsymbol{T}^{-1} = (\boldsymbol{T})^{-1}$

2. Процес реконструкції можливий за допомогою матриці переходу та порогової кількості залучених користувачів, це дозволяє знаходити координати вектора $\boldsymbol{v}$. Після цього ми можемо відновити наш секрет, обчисливши суму векторних координат.

$$\boldsymbol{x}_1 + \boldsymbol{x}_2 + \dots + \boldsymbol{x_n} = \boldsymbol{s}$$

Теорема 5.1

“Перехідна схема розподілу” є досконалою, тобто ідеально коректною та ідеально секретною.

Доведення: Для того, щоб довести, що “Перехідна схема розподілу” є секретною схемою обміну **(n, n),** тобто для віддтворення секрету довжиною в **k-біт** в частини секрету довжини **l-**біт нам потрібна інформація від усіх учасників, а також потрібно показати, що виконуються наступні умови:

- **Ідеальна коректність :** при зборі інформації секрет відтворюється правильно, тобто для будь-якого секрету $a \in \{0,1\}^k$

$$\boldsymbol{Pr[Rec(Share(a)) = 1] = 1,}$$

де ймовірність переважає рандомізованість функції розподілу**.**

Доведення цього факту безпосередньо випливає з означення алгоритму схеми розподілення секрету.

- **Ідеальна секретність :** Група з менш ніж **n** учасників не може дізнатися жодної інформації про секрет. Більш формально, для будь-яких двох секретів **a** та **b** $\in \{0,1\}^k$ множини $\boldsymbol{Share(a)_U}$ та $\boldsymbol{Share(b)_U}$ повинні бути розподілені рівномірно.

Оскільки кожен секрет – це сума координат деякого вектора, який в свою чергу обирається будь-яким зручним для нас способом - ми можемо подати другий секрет через вектор так, щоб перші $\boldsymbol{n-1}$ координат співпадали з координатами першого вектора. Відповідно, скільки б таких векторів ми не обрали для подання наших секретів, ми можемо обрати їх так, що вони матимуть ідентичні перші $\boldsymbol{n-1}$ координат.

А отже і при переході до нового базису у відповідних векторів, що задають перший та другий секрет, перші $\boldsymbol{n-1}$ координат також можуть бути однакові.

Звідси, відповідні множини $\boldsymbol{Share(a)_U}$ та $\boldsymbol{Share(b)_U}$, які є результатами розподілу секретів **a** та **b** для меншої ніж $\boldsymbol{n}$ кількості учасників будуть розподілені рівномірно.

Отже, наша нова запропонована схема є **досконалою.**

Теорема 5.2

“Перехідна схема розподілу” є стійкою до **CPA атак**.

Доведення: Щодо перевірки стійкості “Перехідної схеми розподілу” до **CPA атак** можемо зробити натупне:

Для початку оберемо рандомізовано два різні секрети $\boldsymbol{S_0}$ та $\boldsymbol{S_1}$і покажемо один з них нападнику.
Після чого розподіляємо дані секрети $\boldsymbol{S_0}$ та $\boldsymbol{S_1}$, відповідно до запропонованого вище алгоритму **"Перехідної схеми розподілу"** та розповсюджуємо отримані пари векторів та чисел між усіма учасниками нашої схеми**.**

У випадку, якщо зловмисник заволодіє всіма, окрім однієї парами векторів та чисел учасників схеми, він все одно не зможе відновити початковий секрет. Оскільки для такого відтворення необхідно спершу зібрати матрицю, для якої не вистачатиме одного з векторів, а також вектор, в якому не вистачатиме одного числа.
Відповідно, зловмисник зможе отримати лише неповний набір розподіленої інформації. Отже, він може спробувати вгадати який саме з двох секретів був розподілений.

Оскільки дані розподіли є розподілами множин $\boldsymbol{Share(a)_U}$ та $\boldsymbol{Share(b)_U}$, а з доведеного вище випливає, що ці множини розподілені рівномірно, то зловмисник з однаковою ймовірністю близькою до $\boldsymbol{\frac{1}{2}}$ зможе вгадати який саме із двох початкових секретів був розподілений.

Отже, наша схема стійка до **CPA атак.**

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.

Врешті, було запропоновано новий досконалий, стійкий до CPA-атак, альтернативний спосіб безпечної передачі даних, а саме “Перехідну схему розподілу”. Ця схема розподілу секрету використовує єдиність зображення вектора координатами при переході до іншого базису. Доведено, що нова запропонована схема є досконалою, тобто ідеально коректною та ідеально секретною і стійкою до CPA атак.

Список літератури

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. V.P, Binu, and Sreekumar A. “Simple and Efficient Secret Sharing Schemes for Sharing Data and Image.”, Feb 26, 2015.
4. Kumar, Ashutosh, et al. “Leakage-Resilient Secret Sharing” , 2018.
5. Srinivasan, Akshayaram, and Prashant Nalini Vasudevan. “Leakage Resilient Secret Sharing and Applications∗.” Leakage Resilient Secret Sharing and Applications∗, August 18, 2019.
6. “Secret Sharing.” Wikipedia, Wikimedia Foundation, 26 Mar. 2020,
7. http://www.crypto-it.net/eng/attacks/chosen-plaintext.html
8. Jackson, WA., Martin, K.M. Perfect secret sharing schemes on five participants. Des Codes Crypt 9, 267–286 (1996). https://doi.org/10.1007/BF00129769
9. Smart N. P. Cryptography Made Simple. Cham : Springer International Publishing, 2016. URL: https://doi.org/10.1007/978-3-319-21936-3