**Молдавский государственный университет**

**Факультет математики и информатики**

**Департамент Информатики**

**УПРАЖНЕНИЕ**

**Shamir**

**Выполнил**: Гуцу Даниил.

**Группа:** IA2102

**Проверила:** др.,Чербу О.

Кишинев, 2023

**КИШИНЕВ – 2023**

Алгоритм Шамира (Shamir's Secret Sharing) - это метод секретного разделения, разработанный Ади Шамиром (Adi Shamir) в 1979 году. Этот метод позволяет разделить секретное сообщение на несколько частей (пороговое разделение секрета) таким образом, что для восстановления секрета требуется наличие определенного числа частей (порога).

Вот основные принципы работы алгоритма Шамира:

1. \*\*Загадывание секрета\*\*: Исходный секрет (например, пароль, ключ, или другая конфиденциальная информация) загадывается владельцем секрета.

2. \*\*Разделение секрета\*\*: Секрет разделяется на n частей с помощью математических операций, таких как полиномиальное разложение. При этом указывается пороговое число t, которое определяет, сколько частей из n необходимо для восстановления секрета.

3. \*\*Распределение частей\*\*: Разделенные части секрета распределяются среди n участников, каждый из которых получает свою часть, но не знает секрета.

4. \*\*Восстановление секрета\*\*: Для восстановления секрета требуется наличие как минимум t частей секрета. Если участники объединяют свои части, они могут восстановить исходный секрет.

Алгоритм Шамира обладает высокой стойкостью и безопасностью, так как злоумышленнику нужно иметь доступ к как минимум t частям секрета, чтобы восстановить его. Этот метод часто используется в различных областях, включая криптографию, информационную безопасность и управление ключами, где необходимо обеспечить надежное разделение секретных данных и восстановление их при необходимости.

Код программы:

package shamir;  
  
import java.math.BigInteger;  
import java.util.\*;  
  
  
public class Shamir {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter the secret key : ");  
 int secret = sc.nextInt();  
 Map<Integer, Integer> shares = *generateShares*(secret);  
 secret = *combineShare*(shares);  
 System.*out*.println("The recovery key : " + secret);  
 }  
  
 public static Map<Integer, Integer> generateShares(int secret) {  
 ArrayList<Integer> poly = new ArrayList<>();  
 poly.add(secret);  
 for (int i = 0; i < 4; i++) {  
 poly.add(*generateRandomPrime*(BigInteger.*valueOf*(secret), BigInteger.*valueOf*(999)).intValue());  
 }  
  
 Map<Integer, Integer> shares = new HashMap<>();  
 int share;  
 for (int i = 1; i < 8; i++) {  
 share = (int) (poly.get(0) + poly.get(1) \* i + poly.get(2) \* Math.*pow*(i, 2) + poly.get(3) \* Math.*pow*(i, 3));  
 shares.put(i, share);  
 }  
  
 System.*out*.println("Shares : ");  
 for (int i = 1; i < 8; i++) {  
 if (i != 7) {  
 System.*out*.print("[" + i + ", " + shares.get(i) + "], ");  
 } else {  
 System.*out*.print("[" + i + ", " + shares.get(i) + "]");  
 }  
 }  
 System.*out*.println();  
 return shares;  
 }  
  
 public static BigInteger generateRandomPrime(BigInteger min, BigInteger max) {  
 Random rand = new Random();  
 BigInteger randomPrime;  
  
 do {  
 randomPrime = new BigInteger(max.bitLength(), rand);  
 } while (randomPrime.compareTo(min) < 0 || !*isPrime*(randomPrime));  
  
 return randomPrime;  
 }  
  
 public static boolean isPrime(BigInteger num) {  
 if (num.compareTo(BigInteger.*ONE*) <= 0) {  
 return false;  
 }  
 BigInteger sqrtNum = num.sqrt();  
 for (BigInteger i = BigInteger.*valueOf*(2); i.compareTo(sqrtNum) <= 0; i = i.add(BigInteger.*ONE*)) {  
 if (num.mod(i).equals(BigInteger.*ZERO*)) {  
 return false;  
 }  
 }  
 return true;  
 }  
 public static int combineShare(Map<Integer, Integer> shares) {  
 double secret = 0;  
 int xi = 0;  
  
 int[] x = new int[4];  
 Set<Integer> set = new HashSet<>();  
 Random random = new Random();  
  
 for (int i = 0; i < 4; ) {  
 int num = random.nextInt(7) + 1;  
 if (set.add(num)) {  
 x[i++] = num;  
 }  
 }  
  
 int[] yi = new int[4];  
 for (int i = 0; i < 4; i++) {  
 yi[i] = shares.get(x[i]);  
 }  
 for (int i = 0; i < 4; i++) {  
 double term = yi[i];  
 for (int j = 0; j < 4; j++) {  
 if (j != i) {  
 term \*= (double) (xi - x[j]) / (x[i] - x[j]);  
 }  
 }  
 secret += term;  
 }  
 return (int) secret;  
 }  
  
}

Вывод программы:

Enter the secret key : 233

Shares :

[1, 2130], [2, 7463], [3, 18686], [4, 38253], [5, 68618], [6, 112235], [7, 171558]

The recovery key : 233