دانشگاه اصفهان
دانشکده مهندسی کامپیوتر

مستند پروژه طرح تسهیم راز شمیر

درس رمزنگاری و امنیت در شبکه

استاد:

دکتر ملا

زهرا معصومی **(۴۰۰۳۶۲۳۰۳۴)**

بهار ۱۴۰۳

بخش اول

در بخش اول پروژه، می‌خواهیم برنامه‌ای بنویسیم که پارامترهای n و t و s و p را دریافت کرده و راز s را بین n نفر توزیع کند، به صورتی که حداقل هر t نفر بتوانند با مشارکت یکدیگر آن راز را بازسازی کنند. عدد اول p پیمانه محاسبات خواهد بود.

این کار را با استفاده از تشکیل یک چندجمله‌ای از درجه t-1 و انتخاب n نقطه روی منحنی آن انجام می‌دهیم.

در تمام بخش‌های این پروژه، از کلاس BigInteger استفاده شده است، که هم قابلیت ذخیره اعداد بزرگ را دارد و هم متدهای آماده برای محاسبات پیمانه‌ای.

در ابتدا ضرایب چندجمله‌ای توسط تابع generateCoefficients به صورت تصادفی در پیمانه p تولید می‌شوند (s = f(0)).

```
private static List<BigInteger> generateCoefficients(int k, BigInteger secret, BigInteger prime) {
    List<BigInteger> coefficients = new ArrayList<>();
    coefficients.add(secret); // a0

    SecureRandom random = new SecureRandom();

    for (int i = 1; i < k; i++)
        coefficients.add(new BigInteger(prime.bitLength(), random).mod(prime));

    return coefficients;
}
```

با استفاده از تابع کمکی evaluatePolynomial، مقدار یک x مشخص در چندجمله‌ای که پیش‌تر ضرایب آن را ساختیم، محاسبه می‌شود.

```
private static BigInteger evaluatePolynomial(List<BigInteger> coefficients, BigInteger x, BigInteger prime) {
    BigInteger result = BigInteger.ZERO;

    for (int i = 0; i < coefficients.size(); i++) {
        BigInteger term = coefficients.get(i).multiply(x.pow(i)).mod(prime);
        result = result.add(term).mod(prime);
    }

    return result;
}
```

سپس در تابع generateShares مقادیر 1 تا n به چندجمله‌ای داده می‌شود تا زوج‌های (x,y) را به دست آوریم. هر کدام از این x و y ها، یک سهم هستند که می‌توانند به هر کدام از n نفر تعلق بگیرند.

```
public static List<Share> generateShares(int n, int k, BigInteger secret, BigInteger prime) {
    List<BigInteger> coefficients = generateCoefficients(k, secret, prime);
    List<Share> shares = new ArrayList<>();
    for (int i = 1; i <= n; i++) {
        BigInteger x = BigInteger.valueOf(i);
        BigInteger y = evaluatePolynomial(coefficients, x, prime);
        shares.add(new Share(x, y));
    }
    return shares;
}
```

از آنجایی که منحنی از درجه t-1 است، حداقل با t نقطه روی آن می‌توان معادله را حل کرد و مقدار f(0) که همان راز است را بدست آورد. این کار با استفاده از فرمول درون‌یابی لاگرانژ انجام می‌شود. در بخش دوم این پروژه، نحوه بازیابی راز توسط حداقل t نفر را بررسی خواهیم کرد.

بخش دوم

در این بخش می‌خواهیم نحوه بازیابی رمز را توسط t نفر در پیمانه p بررسی کنیم. فرض می‌کنیم که راز بین n نفر تقسیم شده و اکنون حداقل t نفر (مقدار t در قسمت قبل مشخص شد) از آنها می‌خواهند راز را به دست آورند. این کار توسط فرمول درون‌یابی لاگرانژ در پیمانه p قابل انجام است.

در تابع reconstructSecret، طبق فرمول زیر مقدار f(0) که همان s است را بدست می‌آوریم.

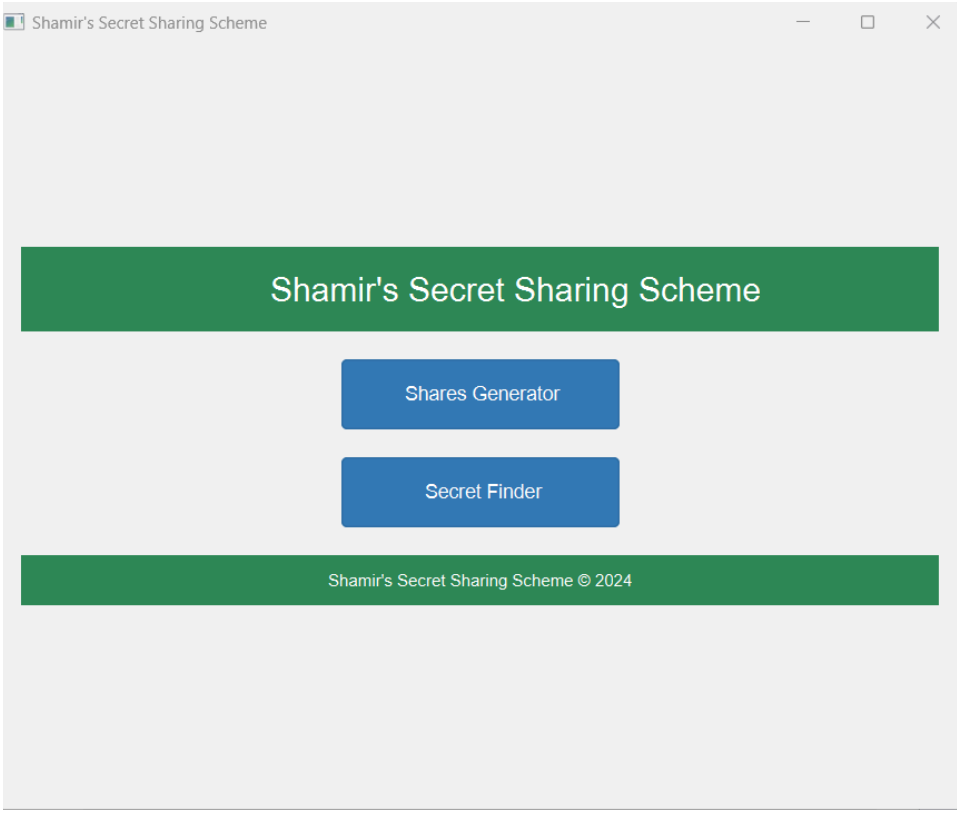
$$f(0) = \sum_{i=1}^{t} (y_i \prod_{j \neq i}^{t} \frac{x_j - x_i}{x_j})$$

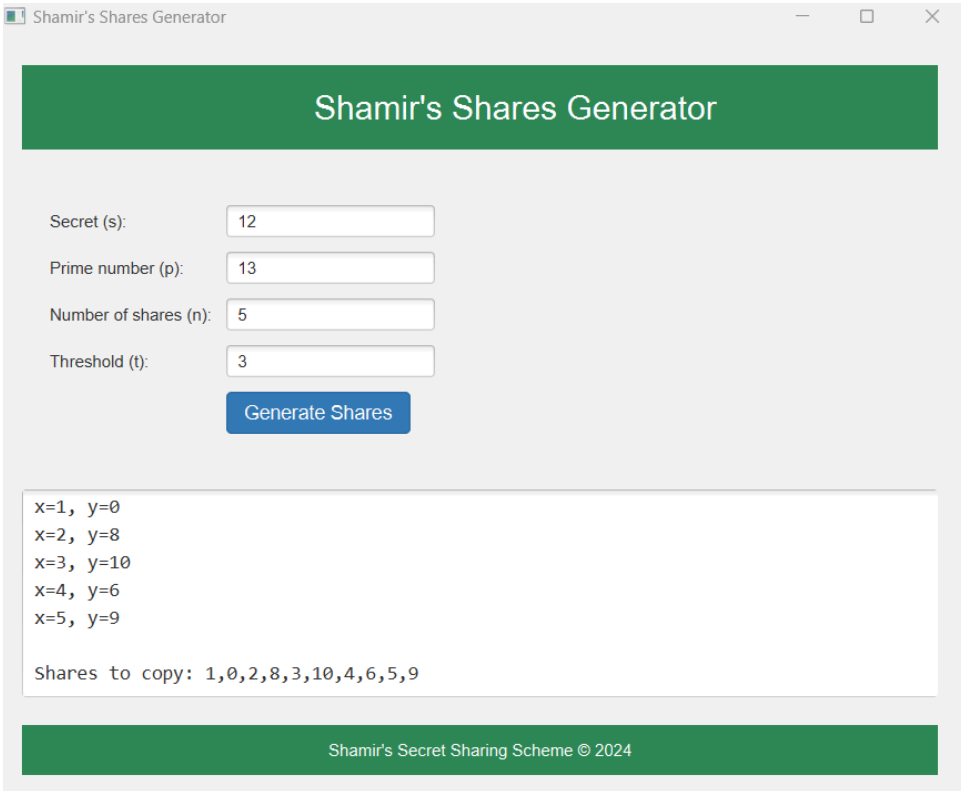
```
public static BigInteger reconstructSecret(List<Share> shares, BigInteger p) {
    BigInteger secret = BigInteger.ZERO;

    for (int i = 0; i < shares.size(); i++) {

        BigInteger x_i = shares.get(i).x;
        BigInteger pi = shares.get(i).y;

        for (int j = 0; j < shares.size(); j++) {
            if (i != j) {
                BigInteger x_j = shares.get(j).x;
                pi = pi.multiply(x_j.multiply((x_j.subtract(x_i)).modInverse(p)).mod(p));
            }
        }

        secret = secret.add(pi).mod(p);
    }
```

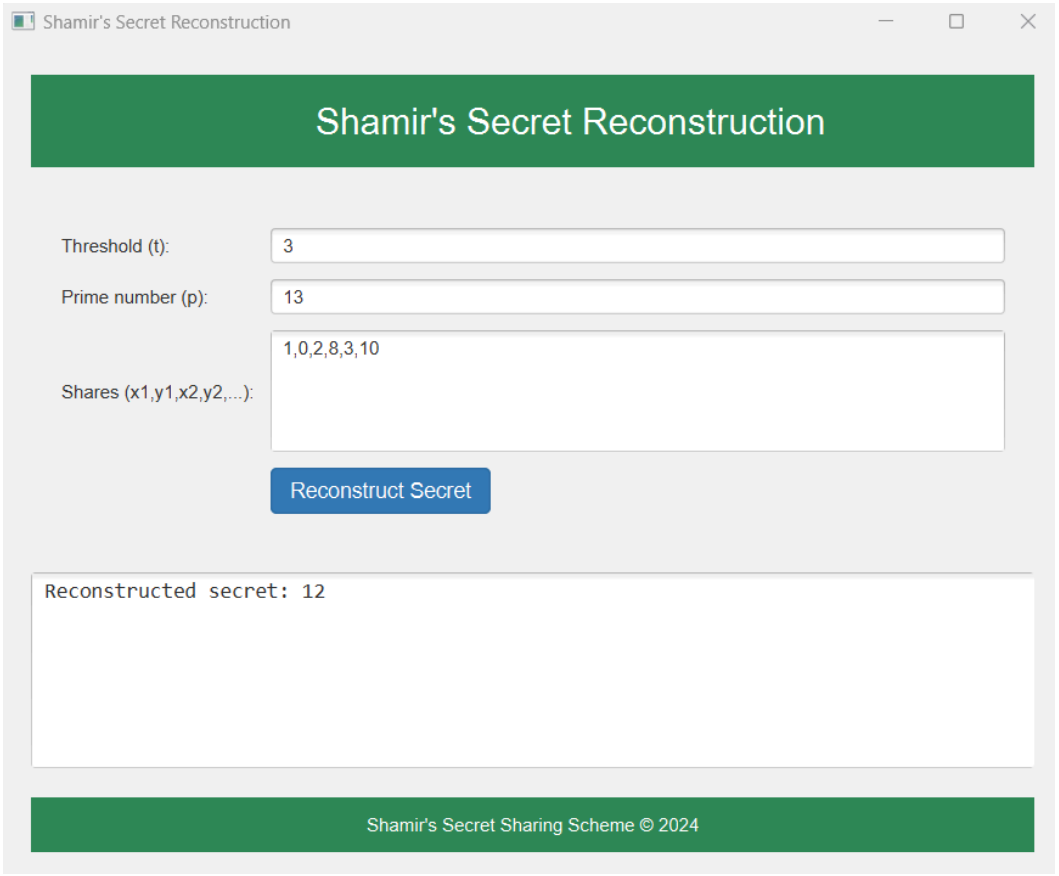
بخش امتیازی

این پروژه هم به صورت کنسولی و هم به صورت گرافیکی قابل اجرا است.

- تصاویر اجرای گرافیکی برنامه:






Shamir's Secret Reconstruction
Shamir's Secret Reconstruction
Threshold (t):
3
Prime number (p):
13
Shares (x1,y1,x2,y2,...):
1,0,2,8,3,10
Reconstruct Secret
Reconstructed secret: 12
Shamir's Secret Sharing Scheme © 2024