دانشگاه اصفهان - دانشکده مهندسی کامپیوتر

رمزنگاری و امنیت در شبکه

استاد درس: دکتر ملا

پیاده‌سازی تسهیم راز شمیر با استفاده از پایتون

(Shamir Secret Sharing Scheme)

شیدا عابدپور

۴۰۰۳۶۲۳۰۲۵

تیر ۱۴۰۳

روش "تقسیم راز شمیر" یک تکنیک رمزنگاری است که توسط آدی شمیر در سال ۱۹۷۹ توسعه یافته است و امکان تقسیم یک راز یا اطلاعات حساس به صورت امن بین چند نفر یا دستگاه را فراهم می‌کند.

یک راز بین n نفر تقسیم می‌شود به طوریکه امکان بازسازی راز با حداقل T نفر امکان‌پذیر باشد(نه کمتر). بنابراین نیاز است تا یک منحنی از درجه T-1 انتخاب شود. راز همان عرض از مبدا منحنی خواهد بود.

با توجه به توضیحات، ابتدا باید یک منحنی از درجه T-1 با ضرایب رندم ساخته شود:

$$f(x) = a_{t-1}x^{t-1} + \cdots + a_1x^1 + s$$

```python
def __init__(self, s, n, t, p=None):
     self.s = s
     self.n = n
     self.t = t
     self.p = p if p else nextprime(s)

     if t > n:
        raise ValueError("Threshold can't be greater than Shares")
     if not isprime(p):
         raise ValueError("p must be prime")
     self.coeffs = [self.s] + [random.randint(1, self.p-1) for _ in range(t-1)]
     self.shares = self._generate_shares()
```

پس از ایجاد منحنی، باید راز را بین n نفر تقسیم کرد.(n نقطه از منحنی انتخاب می‌شود و به سهام‌داران داده می‌شود.)

```python
def _generate_shares(self):
    shares = []
    for i in range(1, self.n+1):
        poly_i = sum([coff * (i ** power) for power, coff in enumerate(self.coeffs)]) % self.p
        shares.append((i, poly_i))
    return shares

def get_shares(self):
    return self.shares, self.coeffs, self.p
```

اکنون حداقل به t-1 نفر از آن n نفر نیاز است تا راز بازسازی شود. این افراد باید با مشارکت یکدیگر t معادله و t مجهول را حل کنند تا به عرض از مبدا منحنی که همان راز است دست یابند.

برای پیدا کردن عرض از مبدا از فرمول درون‌یابی لانگراژ استفاده می‌شود:

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

راز مدنظر همان عرض از مبدا یعنی $f(0)$ است.

```python
def reconstruct_secret(self, shares):
    if len(shares) < self.t:
        raise ValueError("Insufficient number of shares to reconstruct the secret.")
    secret = self._lagrange_interpolation(0, shares)
    return secret

def _lagrange_interpolation(self, x, shares):
    def L(x, i):
        numerator = 1
        denominator = 1
        for j in range(self.t):
            if j != i:
                numerator = (numerator * (x - shares[j][0])) % self.p
                denominator = (denominator * (shares[i][0] - shares[j][0])) % self.p
        return (numerator * mod_inverse(denominator, self.p)) % self.p

    return sum([shares[i][1] * L(x, i) % self.p for i in range(self.t)]) % self.p
```

اکنون می‌توان با تعیین راز مدنظر، افراد سهیم در راز، حداقل افراد سهیم جهت بازسازی راز و پیمانه محاسبات، یک شی از کلاس SSS ساخت و با استفاده از تابع reconstruct_secret و نقاط t فرد مدنظر، راز را بازسازی کرد.

باید توجه کرد t نمیتواند بیشتر از n باشد، همچنین p باید عددی اول و بزرگتر از n و s باشد.

```python
if __name__ == "__main__":

    secret = 6
    num_shares = 5
    threshold = 3
    p_mode = 13

    sss = SSS(secret, num_shares, threshold, p_mode)
    shares, coeffs, p_mode = sss.get_shares()
    print(f'Generated shares: {shares} \n')

    # Check if reconstructing with fewer shares than 't' raises an error
    try:
        invalid_shares = shares[:threshold-1]
        sss.reconstruct_secret(invalid_shares)
    except ValueError as e:
        print(f'Error: {e} \n')

    # Check if reconstructing with correct number of shares works
    try:
        valid_shares = shares[:threshold]
        reconstructed_secret = sss.reconstruct_secret(valid_shares)
        print("Reconstructed secret with valid shares:", reconstructed_secret)
    except ValueError as e:
        print(f'Error: {e} \n')

    # Check if reconstructing with more shares than 't' still works
    try:
        more_shares = shares[:num_shares]
        reconstructed_secret = sss.reconstruct_secret(more_shares)
        print("Reconstructed secret with more shares:", reconstructed_secret)
    except ValueError as e:
        print(f'Error: {e} \n')

    sss.plot_polynomial(shares[:threshold])
```

```
Generated shares: [(1, 10), (2, 8), (3, 0), (4, 12), (5, 5)]

Error: Insufficient number of shares to reconstruct the secret.

Reconstructed secret with valid shares: 6
Reconstructed secret with more shares: 6
```