

Реализация Схемы разделения секрета Шамира

Роман Астраханцев, СКБ-171

1 февраля 2022 г.

1 Написание программы

Была написана программа на языке программирования Python, которая реализует схему разделения секрета Шамира для n участников, при которой обеспечивается восстановление секрета любыми t участниками, $1 \leq t \leq n$. Величины p, n, t задаются пользователем с консоли, $r_1 = 1, \dots, r_n = n$. Коэффициенты a_i исходного многочлена $f(x) = \sum_{i=0}^{t-1} a_i x^i$ инициализируются случайно (как элементы \mathbb{Z}_p) при запуске программы. Текст программы может быть найден в приложении А.

2 Расчёт ключевых значений

Студент выполнял вариант № k=4.

2.1 p=73, t=7, n=15

$$f(x) = 17 + 54x + 52x^2 + 14x^3 + 13x^4 + 70x^5 + 55x^6$$

$$\begin{array}{lll} s_1 = 56 & s_6 = 55 & s_{11} = 24 \\ s_2 = 62 & s_7 = 46 & s_{12} = 58 \\ s_3 = 53 & s_8 = 35 & s_{13} = 6 \\ s_4 = 29 & s_9 = 64 & s_{14} = 28 \\ s_5 = 62 & s_{10} = 39 & s_{15} = 60 \end{array}$$

$$\omega_1 = 7 \quad \omega_2 = 52 \quad \omega_3 = 35 \quad \omega_4 = 38 \quad \omega_5 = 21 \quad \omega_6 = 66 \quad \omega_7 = 1$$

$$s = \sum_{i=1}^t s_i \omega_i = 56 \cdot 7 + 62 \cdot 52 + 53 \cdot 35 + 29 \cdot 38 + 62 \cdot 21 + 55 \cdot 66 + 46 \cdot 1 \pmod{p = 17}$$

2.2 p=139, t=5, n=10

$$f(x) = 131 + 6x + 77x^2 + 127x^3 + 63x^4$$

$$s_1 = 126 \quad s_6 = 124$$

$$s_2 = 112 \quad s_7 = 0$$

$$s_3 = 61 \quad s_8 = 0$$

$$s_4 = 67 \quad s_9 = 133$$

$$s_5 = 68 \quad s_{10} = 113$$

$$\omega_1 = 5 \quad \omega_2 = 129 \quad \omega_3 = 10 \quad \omega_4 = 134 \quad \omega_5 = 1$$

$$s = \sum_{i=1}^t s_i \omega_i = 126 \cdot 5 + 112 \cdot 129 + 61 \cdot 10 + 67 \cdot 134 + 68 \cdot 1 \pmod{p = 131}$$

ПРИЛОЖЕНИЯ

А Текст программы

```
1 import numpy as np
2 import argparse
3
4 def extendedEuclideanAlgorithm(a, b):
5     if abs(b) > abs(a):
6         (x,y,d) = extendedEuclideanAlgorithm(b, a)
7         return (y,x,d)
8
9     if abs(b) == 0:
10        return (1, 0, a)
11
12    x1, x2, y1, y2 = 0, 1, 1, 0
13    while abs(b) > 0:
14        q, r = divmod(a,b)
15        x = x2 - q*x1
16        y = y2 - q*y1
17        a, b, x2, x1, y2, y1 = b, r, x1, x, y1, y
18
19    return (x2, y2, a)
20
21 def inverse(n, p):
22     x,y,d = extendedEuclideanAlgorithm(n, p)
23     return x % p
24
25 def generate_polynomial(t, p):
26     secret = np.random.randint(p)
27     coefs = np.random.randint(p, size=t-2)
28     last_coef = np.random.randint(1, p)
29     polynomial = np.concatenate([[secret], coefs, [last_coef]])
30     return polynomial
31
32
33 def print_polynomial(polynomial):
34     monoms = []
35     for i,g_i in enumerate(polynomial):
36         if g_i == 0:
37             continue
38
39         mon = (str(g_i))
40
41         if i > 1:
42             mon += (f" x^{i}")
43         elif i == 1:
```

```

44         mon += (" x")
45
46         monoms += [mon]
47     return " + ".join(monoms)
48
49 def get_secret(polynomial, r_i, p):
50     x = [(r_i**j % p) for j in range(len(polynomial))]
51     secret = np.dot(polynomial, x) % p
52     return secret
53
54 def get_omega(i, r, t, p):
55     prod = 1
56     for j in range(t):
57         if i != j:
58             value = (r[j] * inverse( (r[i] - r[j]) % p , p))
59             % p
60             prod *= value % p
61     return prod % p
62
63 parser = argparse.ArgumentParser(
64     description="Shamir's secret sharing scheme.")
65 parser.add_argument('p', type=int)
66 parser.add_argument('t', type=int)
67 parser.add_argument('n', type=int)
68
69 args = parser.parse_args()
70
71 p = args.p
72 t = args.t
73 n = args.n
74 r = [i for i in range(1,n+1)]
75
76 polynomial = generate_polynomial(t, p)
77 print(print_polynomial(polynomial))
78
79 s = []
80 for i,r_i in enumerate(r):
81     secret = get_secret(polynomial, r_i, p)
82     to_print = f"s_{i+1} = {secret}"
83     print(to_print)
84     s+= [secret]
85
86 s = s[:t]
87 omegas = []
88 for i in range(t):
89     omega_i = get_omega(i, r[:t], t, p)
90     omegas += [omega_i]
91     print (f"\omega_{i+1} = {omega_i}")

```

```
92  
93  
94 print(np.dot(s, omegas) % p)
```