

Реализация Схемы Блома распределения ключей

Роман Астраханцев, СКБ-171

30 января 2022 г.

1 Написание программы

Была написана программа на языке программирования Python, которая вычисляет ключевые элементы по схеме Блома для n участников, при которой обеспечивается стойкость к m компрометациям, $1 \leq m \leq n - 2$. Величины p, n, m задаются пользователем с консоли, $r_1 = 1, \dots, r_n = n$. Элементы симметричной матрицы коэффициентов A исходного многочлена $f(x, y) = \sum_{i=0}^m \sum_{j=0}^m a_{ij} x^i y^j$ 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, m=1, n=15

$$A = \begin{pmatrix} 62 & 51 \\ 51 & 27 \end{pmatrix}$$

$$g_1(x) = 40 + 5x$$

$$g_2(x) = 18 + 32x$$

$$g_3(x) = 69 + 59x$$

$$g_4(x) = 47 + 13x$$

$$\begin{aligned}
g_5(x) &= 25 + 40x \\
g_6(x) &= 3 + 67x \\
g_7(x) &= 54 + 21x \\
g_8(x) &= 32 + 48x \\
g_9(x) &= 10 + 2x \\
g_{10}(x) &= 61 + 29x \\
g_{11}(x) &= 39 + 56x \\
g_{12}(x) &= 17 + 10x \\
g_{13}(x) &= 68 + 37x \\
g_{14}(x) &= 46 + 64x \\
g_{15}(x) &= 24 + 18x
\end{aligned}$$

2.2 p=139, m=2, n=10

$$A = \begin{pmatrix} 116 & 131 & 35 \\ 131 & 84 & 83 \\ 35 & 83 & 26 \end{pmatrix}$$

$$\begin{aligned}
g_1(x) &= 4 + 20x + 5x^2 \\
g_2(x) &= 101 + 75x + 27x^2 \\
g_3(x) &= 129 + 18x + 101x^2 \\
g_4(x) &= 88 + 127x + 88x^2 \\
g_5(x) &= 117 + 124x + 127x^2 \\
g_6(x) &= 77 + 9x + 79x^2 \\
g_7(x) &= 107 + 60x + 83x^2 \\
g_8(x) &= 68 + 138x \\
g_9(x) &= 99 + 104x + 108x^2 \\
g_{10}(x) &= 61 + 97x + 129x^2
\end{aligned}$$

ПРИЛОЖЕНИЯ

А Текст программы

```
1 import numpy as np
2 import argparse
3
4 def generate_matrix(m, p):
5     matrix = []
6     for i in range(m+1):
7         zeros = np.full(i,0)
8         values = np.random.randint(p, size=m+1-i)
9         string = np.concatenate([zeros, values])
10        matrix += [string]
11    matrix = np.array(matrix)
12    return matrix + np.tril(matrix.T, k=-1)
13
14 def get_secret(matrix, r_i, p):
15     secret = np.matmul(matrix, [(r_i**j % p) for j in range(
16     len(matrix))])
17     return secret % p
18
19 def secret_to_string(secret):
20     monoms = []
21     for i,g_i in enumerate(secret):
22         if g_i == 0:
23             continue
24
25         mon = (str(g_i))
26
27         if i > 1:
28             mon += (f" x^{i}")
29         elif i == 1:
30             mon += (" x")
31
32         monoms += [mon]
33     return " + ".join(monoms)
34
35 parser = argparse.ArgumentParser(
36     description="Blom's key exchange scheme.")
37 parser.add_argument('p', type=int)
38 parser.add_argument('m', type=int)
39 parser.add_argument('n', type=int)
40
41 args = parser.parse_args()
42
43 p = args.p
44 m = args.m
```

```
44 n = args.n
45 r = [i for i in range(1,n+1)]
46
47 matrix = generate_matrix(m, p)
48 print(matrix)
49
50 for i,r_i in enumerate(r):
51     secret = get_secret(matrix, r_i, p)
52     to_print = f"g_{i+1}(x) = " + secret_to_string(secret)
53     print(to_print)
```