Министерство образования и науки Российской Федерации

Московский государственный институт электроники и математики
(технический университет) Кафедра «Компьютерная безопасность»

ОТЧЕТ К РАБОТЕ
«**ро-методом Полларда**» по дисциплине
«Теоретико-числовые методы в криптографии»

Работу выполнили
Студенты группы СКБ 181
Файнберг Т.
Кекеев А.

Работу проверил
Нестеренко А. Ю.

Москва, 2022

Постановка задачи:

Разработать алгоритм, выполняющий получение дискретного алгоритма числа b по основанию a и по модулю m на основе ρo-метода Полларда. Результат оформить в виде отчета.

Использованные инструменты:
В работе были использованы SageMath, JupyterNotebook

Теоретическая база:

Пусть требуется вычислить $log_g a$ в конечной группе G порядка n.

$\in$

$\left\lfloor \frac{p}{3} \right\rfloor$ $\left\lceil \frac{p}{3} \right\rceil$ $\left\lfloor \frac{2p}{3} \right\rfloor$ $\left\lceil \frac{2p}{3} \right\rceil$

$\in$

$\in$ $\in$

Далее на G задается последовательность $x_0, x_1, x_2, \ldots$, где $x_0=1$, x_{i+1} вычисляется по x_i посредством функции f для $i \geq 0$:

$$\begin{cases} ax_i, & если \quad x_i \in S_1 \\ x_i^2, & если \quad x_i \in S_2 \\ gx_i, & если \quad x_i \in S_3 \end{cases}$$

Вычисления проводятся в группе G, то есть если $G=Z_m$, то вычисления следует производить по модулю m.

$g^{x_i} a^{y_i}$

$$\begin{cases} u_i, & если \quad x_i \in S_1 \\ 2u_i, & если \quad x_i \in S_2 \\ u_i + 1, & если \quad x_i \in S_3 \end{cases} \qquad \begin{cases} v_i + 1, & если \quad x_i \in S_1 \\ 2v_i, & если \quad x_i \in S_2 \\ v_i, & если \quad x_i \in S_3 \end{cases}$$

Вычисления в последовательностях u и v производятся по модулю n.

$$g^{u_i} a^{v_i} \quad g^{u_{2i}} a^{v_{2i}} \Rightarrow a^{(v_i - v_{2i})} = g^{(u_{2i} - u_i)}$$

$(v_i — v_{2i}) \log_g a \equiv (u_{2i} — u_i) \pmod{n}$

Решая это сравнение, получим искомый логарифм.

Результаты выполнения работы:

```
def filledMassive(count, modulo,):
    arr = []
    l=0
    while(l<count):
        arr.append(randint(0, modulo))
        l=l+1
    return arr

def filledMassive(count, modulo,):
    arr = []
    l=0
    while(l<count):
        arr.append(randint(0, modulo))
        l=l+1
    return arr

def Poland(a: Mod, b: Mod, p: Integer) -> Integer:
    s = 500

    if p < 500:
        s = p // 2
    m = p - 1
    k0 = Mod(ZZ.random_element(m, distribution='uniform'), m)
    y = z = (a ** k0) % p
    Ay = Az = Mod(k0, m)
    By = Bz = Mod(0, m)
    x = Mod(0, m)
    i = j = 0
    alphas = filledMassive(s, m)
    betas = filledMassive(s, m)

    isStart = True
    while isStart or z != y:
        isStart = False
```

```
        z = f(z, a, b, s, alphas, betas)
        i = lift(z) % s
        Az += alphas[i]
        Bz += betas[i]

        y = f(y, a, b, s, alphas, betas)
        i = lift(y) % s
        y = f(y, a, b, s, alphas, betas)
        j = lift(y) % s
        Ay += alphas[i] + alphas[j]
        By += betas[i] + betas[j]

    Adif = lift(Az - Ay)
    Bdif = lift(By - Bz)

    GCD = gcd(Bdif, m)
    if GCD > 1:
        if Adif % GCD != 0:
            x = Poland(a, b, p)
            return x
        else:
            Adif = Adif // GCD
            Bdif = Bdif // GCD
            m = m // GCD

    Bdif = inverse_mod(Bdif, m)
    x = Mod(Adif * Bdif, m * GCD)
    x -= m
    for i in range(GCD):
        x += m
        if a ** x == b:
            return x, opCount
    x = Poland(a, b, p)
    return x

sptime = []
import time
xArr = []
lArr = []
for jj in range(30):
    print(jj)
    p = next_prime(2 ** jj)
    a = Mod(ZZ.random_element(p), p)
    b = a ** ZZ.random_element(p - 1)

    t1 = time.time()
    x = Poland(a, b, p)

    t2 = time.time()
    sptime.append((t2 - t1))
    xArr.append(x)
    if b==1:
        lArr.append(-1)
    else:
        lArr.append(log(lift(b), lift(a)))
```

Ниже приведен график, сравнения теоретического и экспериментального логарифма. Синий – Теоретический, Оранжевый экспериментальный. Для проверки использовалась функция math.log(b,a)

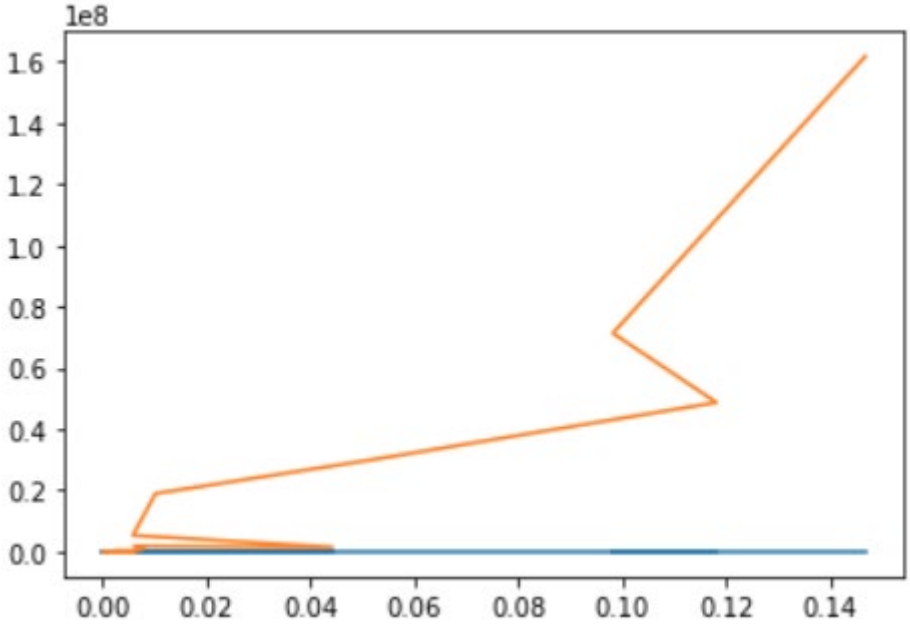

Рисунок 1Сравнение Экспериментального и Теоретического Логарифма.

:

Список литературы

Shi Bai, R. P. On the Efficiency of Pollard’s Rho Method for Discrete Logarithms.
Teske, E. SPEEDING UP POLLARD'S RHO METHOD FOR COMPUTING.
А.Ю.Нестеренко. Теоретико-числовые методы в крипографии.