Министерство образования и науки Российской Федерации

Московский государственный институт электроники и математики

(технический университет) Кафедра «Компьютерная безопасность»

ОТЧЕТ К РАБОТЕ

«**ρο-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 а и по модулю m на основе ρο-метода Полларда. Результат оформить в виде отчета.

**Использованные инструменты:**

В работе были использованы SageMath, JupyterNotebook

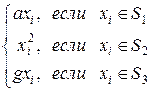
**Теоретическая база:**

Пусть требуется вычислить *logga*в конечной группе G порядка *n*.

Группа G разбивается на три непересекающихся подмножества примерно равной мощности: G=S1US2US3, так чтобы 1  S2. Причем разбиение должно быть построено таким образом, чтобы проверка, к какому подмножеству принадлежит данный элемент x, была простой.

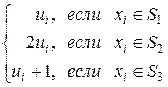
Например, если G=Z*p*, где *p* – простое число, то можно задать разбиение S1={1,…,  }, S2={  ,…,  }, S3={  , … , *p*—1}, или разбиение может быть таким: если *x* mod 3=1, то x  S1, если *x*mod 3=2, то x  S2, если *x*mod 3=0, то *x*  S3.

Далее на G задается последовательность *x*0, *x*1, *x*2, … , где *x*0=1, *xi*+1 вычисляется по *xi* посредством функции *f*для *i*≥0:

*xi*+1=*f*(*xi*)= 

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

Такая последовательность групповых элементов может быть представлена двумя последовательностями *u*0, *u*1, *u*2,… и *v*0, *v*1, *v*2,… такими, что *xi=*  , *u*0=*v*0=0,

*ui*+1=  , *vi*+1= Изображение выглядит как ночное небо

Автоматически созданное описание

Вычисления в последовательностях *u* и *v* производятся по модулю *n*.

В силу того, что группа G – конечная, при помощи метода Флойда можно найти такие *xi* и *x*2*i*, что *xi* = *x*2*i*. Тогда  =    . Логарифмируя по основанию *g* обе части данного уравнения, получим

(*vi—v*2*i*)log*ga*≡(*u*2*i—ui*) (mod *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)

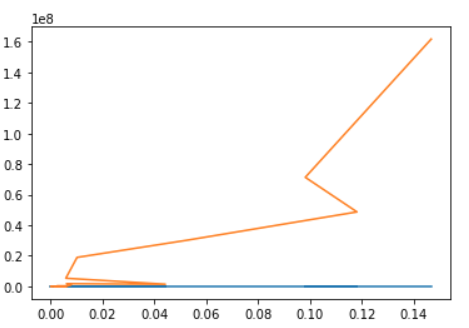


Рисунок 1Сравнение Экспериментального и Теоретического Логарифма.

:

# 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.