Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

**Лабораторная работа №6**

студента 5 курса 531 группы

специальности 10.05.01 «Компьютерная безопасность»

факультета компьютерных наук и информационных технологий

Енца Михаила Владимировича

|  |  |  |
| --- | --- | --- |
| Преподаватель  профессор | \_\_\_\_\_\_\_\_\_\_ | В. А.  Молчанов |
|  | подпись, дата |  |
| Заведующий кафедрой  д.ф.-м.н., доцент | \_\_\_\_\_\_\_\_\_\_ | М. Б. Абросимов |
|  | подпись, дата |  |

Саратов 2018

**1. Постановка задачи.**

Изучение основных методов дискретного логарифмирования в конечном поле и их программная реализация.

**2. Теоретические сведения по рассмотренным темам с их обоснованием.**

Определение. Дискретным логарифмом (показателем) элемента h группы G по основанию g называется число , являющееся решением уравнения

. (1)

Алгоритм Гельфонда-Шенкса

*Вход.* Конечная циклическая группа , верхняя оценка для порядка группы , элемент .

*Выход*. Число

Шаг 1. Вычислить .

Вычислить , упорядочить массив пар по второй координате.

Шаг 2. Вычислить . Для каждого проверить, является ли элемент второй координатой какой-либо пары из упорядоченного на шаге 1 массива пар. Если , то запомнить число .

Шаг 3. Среди всех чисел, найденный на втором этапе, выбрать наименьшее. Оно и будет искомым значением .

Достоинством приведенного алгоритма является его детерминированный характер, а также отсутствие необходимости знать точное значение порядка группы .

*p-метод Полларда.*

Дана конечная циклическая группа , известен ее порядок и .

Метод Полларда применим к любой циклической группе , чьи элементы представлены таким образом, что их можно разбить на три примерно равные, попарно не пересекающиеся части При этом должен существовать эффективный способ проверки, к какому из этих подмножеств принадлежит данный элемент группы.

Будем формировать группы по следующему правилу:

Интуитивно ясно, что эти множества примерно равны по величине.

Определим функцию на таким образом, что

Идея p-метода логарифмирования в некотором смысле повторяет идею p-метода Полларда факторизации. Будет построена рекуррентная последовательность . Из определения функции нетрудно заметить, что при любом для некоторых . Также последовательности задаются следующим рекуррентными соотношениями:

При вычислении очередного члена последовательности числа , вычисляются по известным , очень легко. При этом для любого выполняется равенство

*Вход*: конечная циклическая группа порядка , элемент , функция , заданная соотношением выше, – точность алгоритма.

*Выход*: .

Шаг 1. Вычислить .

Шаг 2. Положить , выбрать случайное , вычислить , . Запомнить две тройки , и перейти к шагу 4.

Шаг 3. Положить , найти , . Запомнить две тройки , и перейти к шагу 4.

Шаг 4. Если и – перейти к шагу 3. Если , то остановить алгоритм и сообщить, что вычислить не удалось. Если , то перейти к шагу 5.

Шаг 5. Вычислить . Если , то сравнение

имеет различных решений по модулю . Для каждого из этих решений проверить выполнимость равенства и найти истинное решение . Если – перейти на шаг 2 и выбрать новое значение .

**Коды программ, реализующей рассмотренные алгоритмы**

Программа реализована на языке Python (версия интерпретатора 3.6).

def shanks\_gelfond(m, g, h):

def square\_root(n):

x1 = n

x2 = int((x1 + (n / x1)) / 2)

while x2 < x1:

x1, x2 = x2, int((x2 + (n / x2)) / 2)

return x1

# Step 1

r = square\_root(m) + 1

pairs = {pow(g, a, m): a for a in range(r)}

# Step 2

g1 = pow(utils.inverse(g, m), r, m)

for b in range(r): # r-1 ?

value = (pow(g1, b, m) \* h) % m

if value in pairs:

return pairs[value] + r \* b

def ppollard(m, g, h, e=0.05):

def equation(a, b, m):

a, b = a % m, b % m

d = math.gcd(a, m)

if b % d != 0:

return

a\_new, b\_new, m\_new = a // d, b // d, m // d

d\_new, q, r = euclid.euclid\_extended(a\_new, m\_new)

q, r = q % m, r % m

x0 = (b\_new \* q) % m\_new

for j in range(d):

yield x0 + m\_new \* j

def calculate\_yab(y, a, b, params):

g, h, m = params

if y <= m // 3:

y = (y \* h) % m

a = (a + 1) % (m - 1)

elif m // 3 < y <= 2 \* m // 3:

y = (y \* y) % m

a = (a \* 2) % (m - 1)

b = (b \* 2) % (m - 1)

elif 2 \* m // 3 < m:

y = (y \* g) % m

b = (b + 1) % (m - 1)

return y, a, b

# Step 1

t = square\_root(2 \* m \* math.log(1 / e)) + 1

while True:

# Step 2

i = 1

s = random.randint(0, m - 2) # m-1 ?

yi, ai, bi = calculate\_yab(pow(g, s, m), 0, s, (g, h, m))

y2i, a2i, b2i = calculate\_yab(yi, ai, bi, (g, h, m))

# Step 4

while i < t and yi != y2i:

# Step 3

i += 1

yi, ai, bi = calculate\_yab(yi, ai, bi, (g, h, m))

y2i, a2i, b2i = calculate\_yab(\*calculate\_yab(y2i, a2i, b2i, (g, h, m)), (g, h, m))

if yi == y2i:

# Step 5

aa, bb = (a2i - ai) % (m - 1), (bi - b2i) % (m - 1)

d = sympy.gcd(aa, m - 1)

if d < square\_root(m - 1):

for x in equation(aa, bb, m - 1):

if pow(g, x, m) == h:

return x

**Оценки сложности рассмотренных алгоритмов**

Сложность алгоритма Гельфонда-Шенкса составляет операций в группе Объем использованной памяти составляет ячеек.

Сложность алгоритма p-метод Полларда составляет операций в группе .

**Результаты тестирования программ**

Оценим корректность работы алгоритмов. Для этого возьмем достаточно большую циклическую группу и на нескольких образующих и нескольких значениях степеней проверим результаты работы алгоритмов.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | Алгоритм Гельфонда-Шенкса | -метод Полларда |
| 2163547 | 11 | 10012 | 682681 | 682681 |
| 2163547 | 11 | 100500 | 2069622 | 2069622 |
| 2163547 | 29 | 6789 | 1137605 | 1137605 |
| 2163547 | 28 | 184 | 1798464 | 1798464 |
| 89765387 | 5 | 79823 | 80999464 | 80999464 |
| 89765387 | 14 | 89765000 | 8104158 | 8104158 |
| 89765387 | 18 | 56473 | 28285821 | 28285821 |
| 89765387 | 18 | 2 | 19451523 | 19451523 |

4. Тестирование скорости работы алгоритмов

Протестируем скорость работы алгоритмов на 1000 запусков с различными параметрами, в ячейки таблицы запишем суммарное время работы (в секундах) на все запуски для каждого алгоритма.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | Алгоритм Гельфонда-Шенкса | -метод Полларда |
| 5471 | 7 | 101 | 0.293 c | 1.254 c |
| 5471 | 7 | 1012 | 0.437 c | 1.261 c |
| 5471 | 7 | 5470 | 0.289 c | 1.606 c |
| 5471 | 7 | 1 | 0.189 c | 0.466 c |

Скорость работы алгоритма Гельфонда-Шенкса варьируется для различных чисел до 2 раз, случай не многим быстрее других значений . -метод Полларда работает дольше в среднем в 3 раза, для любых .