МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КУБГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**Отчет**

**по практическому заданию №5**

**по курсу**

**«КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ»**

Работу выполнил

Студенты 46 группы

Нагалевский А.М.

Преподаватель:

Крамаренко А.А.

Краснодар 2024

**Постановка задачи.**

**Задание 1**

Числа Мерсена — это числа вида (M\_n = 2^n - 1), где n — натуральное число. Чтобы число Мерсена было простым, n также должно быть простым числом, однако не каждое число вида (M\_n = 2^n - 1), где n простое, является простым числом.

Примеры простых чисел Мерсена:

1. (n = 2), (M\_2 = 2^2 - 1 = 3)

2. (n = 3), (M\_3 = 2^3 - 1 = 7)

3. (n = 5), (M\_5 = 2^5 - 1 = 31)

4. (n = 7), (M\_7 = 2^7 - 1 = 127)

5. (n = 13), (M\_13 = 2^13 - 1 = 8191)

**Задание 2**

Числа Ферма — это числа вида (F\_n = 2^{2^n} + 1), где n — целое неотрицательное число. Пьер Ферма выдвинул гипотезу, что все такие числа являются простыми, однако позже было доказано, что это не так для всех n.

Примеры простых чисел Ферма:

1. (F\_0 = 2^{2^0} + 1 = 2^1 + 1 = 3)

2. (F\_1 = 2^{2^1} + 1 = 2^2 + 1 = 5)

3. (F\_2 = 2^{2^2} + 1 = 2^4 + 1 = 17)

4. (F\_3 = 2^{2^3} + 1 = 2^8 + 1 = 257)

**Задание 3**

Тест Ферма — это простой способ проверки числа на простоту, который опирается на малую теорему Ферма. Согласно малой теореме Ферма, если p — простое число и a — целое число, не делящееся на p, то (a^{p-1} ≡ 1 mod p).

Для проведения теста Ферма для заданных чисел, выберем a так, чтобы 1 < a < p (чаще всего используют a = 2), и вычислим (a^{p-1} mod p). Если результат не равен 1, число не является простым. Если результат равен 1, число может быть простым, но для полной уверенности требуются дополнительные проверки, так как существуют составные числа, которые также удовлетворяют этому условию (псевдопростые числа).

Проведем тест для чисел 23, 41, 15, 35 и 561 с a = 2.

Расчеты:

1. Для p = 23, проверим (2^{22} mod 23)

2. Для p = 41, проверим (2^{40} mod 41)

3. Для p = 15, проверим (2^{14} mod 15)

4. Для p = 35, проверим (2^{34} mod 35)

5. Для p = 561, проверим (2^{560} mod 561)

Результаты теста Ферма для данных чисел следующие:

1. Для p = 23, (2^{22} mod 23 = 1)

2. Для p = 41, (2^{40} mod 41 = 1)

3. Для p = 15, (2^{14} mod 15 = 4)

4. Для p = 35, (2^{34} mod 35 = 9)

5. Для p = 561, (2^{560} mod 561 = 1)

Из результатов видно, что числа 23 и 41 проходят тест Ферма, что соответствует их простоте. Числа 15 и 35 не проходят тест Ферма,

что подтверждает их составной характер. Однако число 561 также проходит тест Ферма, возвращая результат 1 несмотря на то, что оно является составным.

Это делает 561 псевдопростым числом по основанию 2.

Последнее число, 561, известно как число Кармайкла. Числа Кармайкла — это особый класс составных чисел, которые удовлетворяют условию малой теоремы Ферма

(a^{p-1} ≡ 1 mod p) для всех a, взаимно простых с p, и поэтому они могут вводить в заблуждение тесты простоты, основанные на малой теореме Ферма.

561 = 3 \* 11 \* 17, соотвественно.

**Задание 4**

Испытание квадратным корнем — это метод проверки числа на простоту, который заключается в поиске делителей числа в пределах, не превышающих квадратный корень из этого числа.

Если в этом диапазоне не найдено никаких делителей, число считается простым. В противном случае оно составное.

Произведем испытание квадратным корнем для чисел 23, 41, 15, 35 и 561.

Алгоритм проверки:

1. Вычислить квадратный корень из числа.

2. Проверить все целые числа от 2 до округленного вверх значения квадратного корня на деление проверяемого числа без остатка.

3. Если делитель найден, число составное; если нет — простое.

Результаты испытания квадратным корнем для данных чисел следующие:

- 23: Простое число

- 41: Простое число

- 15: Составное число

- 35: Составное число

- 561: Составное число

Эти результаты подтверждают, что 23 и 41 являются простыми числами, так как для них не было найдено никаких делителей в пределах до квадратного корня из числа.

Числа 15, 35 и 561 являются составными, поскольку для них были найдены делители, не равные 1 и самому числу.

**Задание 5**

Тест Миллера-Рабина — это вероятностный тест простоты, который используется для определения, является ли данное число простым.

Он основан на свойствах простых чисел и предполагает выполнение определенных условий для числа n, чтобы считаться простым.

Этот тест может дать два типа результатов: "составное" с абсолютной уверенностью и "вероятно простое", где вероятность ошибки в принятии составного числа за простое снижается с увеличением количества раундов теста.

Для проведения теста Миллера-Рабина для чисел 23, 41, 15, 35 и 561, я выполню вычисления и предоставлю результаты.

Поскольку это вероятностный тест, для чисел, которые он определит как "вероятно простые", будет использовано несколько раундов для увеличения достоверности результатов.

Результаты теста Миллера-Рабина

- 23: Вероятно простое (тест подтверждает простоту)

- 41: Вероятно простое (тест подтверждает простоту)

- 15: Составное

- 35: Составное

- 561: Составное

Шаги:

1. Проверить, не является ли число n четным или меньше 2, так как все четные числа, кроме 2, являются составными.

2. Найти d и r такие, что (n - 1 = d \* 2^r), где d нечетно. Это представление n - 1 как произведения степени двойки на нечетное число.

3. Провести заданное количество раундов тестирования. Для каждого раунда:

- Выбрать случайное число a в интервале [2, n - 2].

- Вычислить (x = a^d mod n). Если x = 1 или x = n - 1, перейти к следующему раунду.

- Повторять возведение x в квадрат ((x^2 mod n)) r - 1 раз. Если в какой-то момент получается n - 1, перейти к следующему раунду. Если доходит до конца цикла и n - 1 не встретилось, вернуть "составное".

**Задание 6**

Метод Ферма для разложения на множители основан на поиске таких целых чисел x и y, что (n = x^2 - y^2 = (x + y)(x - y)), где n — данное нечетное составное число,

которое необходимо разложить. Этот метод наиболее эффективен, когда n является произведением двух простых чисел, близких друг к другу по значению.

Алгоритм:

1. Начать с (x = |√{n}|), где (|\*|) обозначает округление до ближайшего большего целого числа.

2. Вычислить (y^2 = x^2 - n). Если (y^2) является полным квадратом, то n можно разложить на множители как (n = (x + y)(x - y)).

3. Увеличить (x) на 1 и повторить шаг 2 до тех пор, пока не будет найдено разложение.

Разложение на множители методом Ферма для данных чисел:

- 483 = 23 × 21

- 1207 = 71 × 17

- 561 = 33 × 17

- 1219 = 53 × 23

**Задание 7**

Метод Полларда (p-1) для разложения на множители основан на предположении, что для составного числа n, существует делитель p, такой что p-1 имеет только малые простые делители.

Идея метода заключается в том, что если p является делителем n, и a — случайно выбранное число, не делящееся на p, то с большой вероятностью (gcd(a^{k!} - 1, n)) будет ненулевым делителем n для некоторого k.

Алгоритм:

1. Выбрать начальное значение a > 1 (обычно a = 2).

2. Выбрать границу B для простых чисел p, которые будут использоваться.

3. Вычислить (a^{B!} mod n) посредством последовательного возведения в степень, используя малые простые числа до B.

4. Вычислить (gcd(a^{B!} - 1, n)). Если результат больше 1, это будет делитель n.

Применение метода (p-1) Полларда для разложения на множители к указанным числам с базовой границей B=10 дало следующие результаты:

- 483: найден делитель 21

- 1207: делитель не найден

- 561: делитель не найден

- 1219: делитель не найден

Для числа 483 метод успешно нашел нетривиальный делитель. Однако для остальных чисел делители не были найдены при выбранной границе B=10. Это может означать, что для успешного разложения этих чисел требуется использовать более высокое значение B, так как возможно, что p-1 для их делителей p содержит более крупные простые числа.

**Текст программы:**

**Файл LR5.py:**

def legendre\_symbol(a, p):

"""

Вычисляет символ Лежандра a/p

"""

ls = pow(a, (p - 1) // 2, p)

return ls if ls != p - 1 else -1

def inverse\_mod(a, m):

"""

Нахождение обратного элемента для 'a' по модулю 'm'

"""

m0, x0, x1 = m, 0, 1

while a > 1:

q = a // m

m = a % m

a = m

x0, x1 = x1 - q \* x0, x0

return x1 + m0 if x1 < 0 else x1

def quadratic\_residue\_modulo(d, m):

"""

Решение квадратичного сравнения x^2 ≡ d (mod m)

"""

solutions = []

if legendre\_symbol(d, m) == -1:

return solutions # нет решений

for x in range(m):

if (x\*x - d) % m == 0:

solutions.append(x)

return solutions

# Примеры использования:

d = 8

m = 14

print(f"Решения x^2 ≡ {d} (mod {m}): {quadratic\_residue\_modulo(d, m)}")

d = 4

m = 21

print(f"Решения x^2 ≡ {d} (mod {m}): {quadratic\_residue\_modulo(d, m)}")

d = 8

m = 23

print(f"Решения x^2 ≡ {d} (mod {m}): {quadratic\_residue\_modulo(d, m)}")