федеральное государственное автономное образовательное учреждение
высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ОТЧЕТ

по лабораторной работе №1

«Атака на алгоритм шифрования RSA посредством метода Ферма»

по дисциплине «**Информационная безопасность**»

Вариант 12

Автор: Кулаков Н. В.

Факультет: ПИиКТ

Группа: P34312

Преподаватель: Маркина Т. А.

Санкт-Петербург 2023

1. Цель работы

Изучить атаку на алгоритм шифрования RSA посредством метода Ферма.

2. Описание

Используя разложение модуля на простые числа методом Ферма и полученные исходные данные определить показатели множителей модуля, значение функции Эйлера, обратное значение экспоненты и дешифровать зашифрованный текст.

3. Выполнение

Повторяем алгоритм атаки посредством метода Ферма, описанный в методическом пособии, базирующийся на случае, когда p и q - близкие друг к другу числа. Выполнение производилось в Jupyter.

Исходные данные (Вариант 12):

```
N = 74701165267919
e = 3145553
blocks = [
    32035658541536,
    35242897170964,
    6268303368709,
    6877322610982,
    16329207109754,
    35007623593376,
    26715311593240,
    36220800128563,
    25019660581036,
    61639733671958,
    21186453949445,
    72477207535811
]
```

Находим n как квадратный корень N и проверяем, является ли N квадратом:

```
n = int(math.sqrt(N))
print(n)

def is_integer(D):
    return int(D) == D

if (is_integer(N ** 0.5)):
    print("N - квадрат")
else:
    print("N - не квадрат")
########## output: ##########
8642983
N - не квадрат
```

Возводим число t в квадрат и пару w, таким образом, чтобы w ** 0.5 было целое:

```
t : int
w : int
for i in range(1, 1000):
    t = n + i
    w = t ** 2 - N
    print(w)
    r = w ** 0.5
    if (is_integer(r)):
        print(f"w={w} квадрат числа {int(r)}")
        break
    else:
        print(f"w={w} не квадрат. продолжаем поиск")

t = int(t)
w = int(w)
########## output: ##########
w=7156337 не квадрат. продолжаем поиск
w=24442306 не квадрат. продолжаем поиск
w=41728277 не квадрат. продолжаем поиск
w=59014250 не квадрат. продолжаем поиск
w=76300225 квадрат числа 8735
```

Получаем соответствующие p и q из t и w, найденных по алгоритму Ферма:

```
p = t + int(w ** 0.5)
q = t - int(w ** 0.5)
print(p, q)
########## output: ##########
8651723 8634253
```

Находим число Эйлера:

```
phi_n = (p - 1) * (q - 1)
phi_n
########## output: ##########
74701147981944
```

Вычисляем число, обратное e по алгоритму Евклида:

```
def gcd_extended(a, b):
    if (a == 0):
        x = 0
        y = 1
        return (b, x, y)

    gcd, x, y = gcd_extended(b % a, a)
    x1 = x
    y1 = y

    x = y1 - (b // a) * x1
    y = x1

    return (gcd, x, y)



def mod_inverse(A, M):
    g, x, y = gcd_extended(A, M)
    if (g != 1):
        return None # inverse doesn't exist
    else:
        res = (x % M + M) % M
```

```
        return res

d = int(mod_inverse(e, phi_n))
d
########## output: ##########
23647864249265
```

Расшифовываем зашифрованные блоки сообщения и соединяем их вместе:

```
def to_string(i):
    length = math.ceil(i.bit_length() / 8)
    return i.to_bytes(length, byteorder='big').decode('windows-1251')

blocks_decrypted = []

for b in blocks:
    bd = pow(b, d, N)
    print(f"{b} → {bd}")
    blocks_decrypted.append(bd)

print("encoded: ", "".join([to_string(bd) for bd in blocks_decrypted]))
########## output: ##########
32035658541536 → 3991269360
35242897170964 → 3772967147
6268303368709 → 4243451625
6877322610982 → 552592880
16329207109754 → 3857841131
35007623593376 → 3941081327
26715311593240 → 3773490674
36220800128563 → 4007796781
25019660581036 → 552595170
61639733671958 → 4075745517
21186453949445 → 4226097391
72477207535811 → 3857769773
encoded: неправильной пересылки пакетов - повторные пере-
```

4. Выводы

RSA — асинхронный алгоритм блочного шифрования сообщений. Как правило такие используются для начальной передаче синхронного ключа.

В ходе выполнения ЛР познакомился с различными видами алгоритмов атаки на RSA, в частности по методу Ферма. При выборе плохих p и q на алгоритм может быть произведена атака. В частности, если они близки, то одной из возможных атак является атака по методу Ферма. Как раз это и было продемонстрировано в текущей ЛР.

Хочется добавить, что для обеспечения достаточной надежности из-за различных выявленных уязвимостей при использовании RSA ключи должны быть гораздо больше, чем, например, при алгоритме эллиптических кривых.