MainPage/Info Security/Encript Lab 2.1

Университет ИТМО
Факультет ФПИ и КТ

Отчёт по лабораторной работе 2.1

«Атака на алгоритм шифрования RSA посредством метода Ферма»

по дисциплину

«Информационная безопасность»

Студент: Чжоу Хунсян
Группа: P34131
Преподаватель:

Санкт-Петербург 2024

Цель работы

изучить атаку на алгоритм шифрования RSA посредством метода Ферма

Варианты заданий

> Вариант: 23

Вариант	Модуль, N	Экспонента, e	Блок зашифрованного текста, C
23	48992988576733	4545733	12530303611339 47274247086952 20068556933394 41300245344157 27564916776233 45997492729411 11416336760074 17516700753417 10586755223028 5642378694993 17949047899806 13276902592875

Исходный код

```python
import math


# Функция для нахождения простых чисел p и q методом квадратичных корней
def find_prime_factors(N):
    # Начальное приближение корня
    n = int(math.sqrt(N)) + 1

    i = 0
    while True:
        i += 1
        t = n + i
        w = t ** 2 - N
        sqrt_w = math.sqrt(w)
        if sqrt_w % 1 == 0:
            sqrt_w = int(sqrt_w)
            p = t + sqrt_w
            q = t - sqrt_w
            print(f"p={p}, q={q}")
            return p, q
        else:
            print("error")


# Функция для расшифровки сообщения
def decrypt_message(C, d, N):
    result = ""
    for i in C:
        # Расчет m = C^d mod N
        m = pow(int(i), d, N)
        # Преобразование числа в текст (кодировка cp1251)
        part = m.to_bytes(4, byteorder='big').decode('cp1251')
        print(f'{i}^{d} mod {N} = {m} => text({m}) = {part}')
        result += part
    return result


if __name__ == '__main__':
    # Исходные данные
```

```
N = 48992988576733  # Модуль N
e = 4545733  # Открытая экспонента e
C = [  # Зашифрованные данные
    12530303611339,
    47274247086952,
    20068556933394,
    41300245344157,
    27564916776233,
    45997492729411,
    11416336760074,
    17516700753417,
    10586755223028,
    5642378694993,
    17949047899806,
    13276902592875
]

# Нахождение простых чисел p и q
p, q = find_prime_factors(N)

# Вычисление функции Эйлера
phi = round((p - 1) * (q - 1))

# Вычисление закрытой экспоненты d
d = pow(e, -1, phi)

# Расшифровка сообщения
result = decrypt_message(C, d, N)

# Печать результата
print(f"result = {result}")
```

Результаты работы программы

Пример Usage

```
error
error
p=7006829, q=6992177
125303036113339^25037979834125 mod 48992988576733 = 4008702696 => text(4008702696) = опти
47274247086952^25037979834125 mod 48992988576733 = 3974163452 => text(3974163452) = маль
20068556933394^25037979834125 mod 48992988576733 = 3992710176 => text(3992710176) = ным
41300245344157^25037979834125 mod 48992988576733 = 1297372448 => text(1297372448) = MTU
27564916776233^25037979834125 mod 48992988576733 = 3990888691 => text(3990888691) = на у
45997492729411^25037979834125 mod 48992988576733 = 4042187501 => text(4042187501) = ровн
11416336760074^25037979834125 mod 48992988576733 = 3844097100 => text(3844097100) = е DL
17516700753417^25037979834125 mod 48992988576733 = 1126965484 => text(1126965484) = C, м
10586755223028^25037979834125 mod 48992988576733 = 3773556205 => text(3773556205) = ален
5642378694993^25037979834125 mod 48992988576733 = 4243253481 => text(4243253481) = ький
17949047899806^25037979834125 mod 48992988576733 = 539551979 => text(539551979) =  (ил
13276902592875^25037979834125 mod 48992988576733 = 3894435679 => text(3894435679) = и __
result = оптимальным MTU на уровне DLC, маленький (или __

Process finished with exit code 0
```

Вывод

В ходе выполнения работы мы реализовали метод Ферма для атаки на алгоритм шифрования RSA на языке python.