**ЛАБОРАТОРНАЯ РАБОТА №1**

“Атака на алгоритм шифрования RSA посредством метода Ферма”

по дисциплине

‘Информационная безопасность’

Вариант 15

***Выполнил:***

Соболев Иван Александрович

**Группа:** P34312

***Преподаватель:***

Маркина Татьяна Анатольевна

Санкт-Петербург, 2024

# Цель работы

# Изучить атаку на алгоритм шифрования RSA посредством метода Ферма.

# Программные и аппаратные средства

Для выполнения лабораторной работы был использован компьютер со следующими характеристиками:

* Процессор: Apple M2
* Видеокарта: Apple M2
* Объем оперативной памяти: 8GB
* Использована операционная система: macOS 14.4.1
* Версия Python: 3.12

# Задание

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | Модуль, N | Экспонента, e | Блок зашифрованного текста, C |
| 15 | 67510894259489 | 3543923 | 1834956116931  7762509478845  22384877417897  36443182878894  61287041306052  17680469174617  14632055288035  23212409940234  45782556562975  7533626343287  14537172455552  60777304839141 |

# Листинг разработанной программы

main.py

from RSA import decrypt\_block, calculate\_phi, find\_d\_component  
from ferma import ferma\_attack  
from io\_utils import int\_to\_bytes, print\_green, print\_red, read\_config  
  
  
def decrypt\_ciphertexts(N, e, ciphertexts, p, q):  
 *"""Дешифрует список шифротекстов и возвращает расшифрованные байты."""* bytes = []  
 phi = calculate\_phi(p, q)  
 print(f'Результат вычисления функции Эйлера: {phi}')  
 d = find\_d\_component(e, phi)  
 print(f'Результат вычисления параметра d: {d}\n')  
 for c in ciphertexts:  
 decrypted\_block = decrypt\_block(d, c, N)  
 bytes.append(int\_to\_bytes(decrypted\_block))  
 decrypted\_bytes = b''.join(bytes)  
 return decrypted\_bytes  
  
  
def main():  
 N, e, ciphertexts = read\_config('config.json')  
 p, q = ferma\_attack(N)  
 print\_green(f"Результат факторизации Ферма:")  
 print(f"p = {p}, q = {q}\n")  
  
 decrypted\_bytes = decrypt\_ciphertexts(N, e, ciphertexts, p, q)  
 try:  
 plaintext = decrypted\_bytes.decode('cp1251')  
 print\_green("Расшифрованный текст:")  
 print(plaintext)  
 except UnicodeDecodeError:  
 print\_red("Ошибка декодирования сообщения:")  
 print(decrypted\_bytes)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

print(f'Результат вычисления функции Эйлера: {phi}')  
 d = find\_d\_component(e, phi)  
 print(f'Результат вычисления параметра d: {d}\n')  
 for c in ciphertexts:  
 decrypted\_block = decrypt\_block(d, c, N)  
 bytes.append(int\_to\_bytes(decrypted\_block))  
 decrypted\_bytes = b''.join(bytes)  
 return decrypted\_bytes  
  
  
def main():  
 N, e, ciphertexts = read\_config('config.json')  
 p, q = ferma\_attack(N)  
 print\_green(f"Результат факторизации Ферма:")  
 print(f"p = {p}, q = {q}\n")  
  
 decrypted\_bytes = decrypt\_ciphertexts(N, e, ciphertexts, p, q)  
 try:  
 plaintext = decrypted\_bytes.decode('cp1251')  
 print\_green("Расшифрованный текст:")  
 print(plaintext)  
 except UnicodeDecodeError:  
 print\_red("Ошибка декодирования сообщения:")  
 print(decrypted\_bytes)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

Io\_utils.py

import json  
  
def int\_to\_bytes(m):  
 *"""Метод конфертации чисел в байты"""* hex\_str = hex(m)[2:]  
 if len(hex\_str) % 2:  
 hex\_str = '0' + hex\_str  
 return bytes.fromhex(hex\_str)  
  
  
def read\_config(file\_path):  
 *"""Метод чтения параметров из конфигурационного файла."""* with open(file\_path, 'r') as f:  
 config = json.load(f)  
 return config['N'], config['e'], config['ciphertexts']  
  
  
def print\_green(message: str) -> None:  
 *"""Метод для вывода ключа"""* print(f"\033[92m{message}\033[0m")  
  
  
def print\_red(message: str) -> None:  
 *"""Метод для вывода расшифрованного текста"""* print(f"\033[91m{message}\033[0m")

def print\_red(message: str) -> None:  
 *"""Метод для вывода расшифрованного текста"""* print(f"\033[91m{message}\033[0m")

ferma.py

import math  
  
def ferma\_attack(n):  
 *"""Метод для проведения факторизации Ферма"""* a = math.isqrt(n)  
 b2 = a \* a - n  
  
 while b2 < 0 or not is\_square(b2):  
 a += 1  
 b2 = a \* a - n  
  
 b = math.isqrt(b2)  
  
 p = a - b  
 q = a + b  
  
 return p, q  
  
  
def is\_square(x):  
 *"""Метод проверки числа на квадрат"""* s = int(math.isqrt(x))  
 return s \* s == x

RSA.py

def decrypt\_block(d, c, N):  
 *"""Метод дешифрации блока"""* return pow(c, d, N)  
  
  
def find\_d\_component(e, phi):  
 *"""Метод вычисления параметра закрытого ключа"""* return pow(e, -1, phi)  
  
  
def calculate\_phi(p, q):  
 *"""Метод вычисления функции Эйлера"""* return (p - 1) \* (q - 1)

config.json

{  
 "N": 67510894259489,  
 "e": 3543923,  
 "ciphertexts": [  
 1834956116931,  
 7762509478845,  
 22384877417897,  
 36443182878894,  
 61287041306052,  
 17680469174617,  
 14632055288035,  
 23212409940234,  
 45782556562975,  
 7533626343287,  
 14537172455552,  
 60777304839141  
 ]  
}

7762509478845,  
 22384877417897,  
 36443182878894,  
 61287041306052,  
 17680469174617,  
 14632055288035,  
 23212409940234,  
 45782556562975,  
 7533626343287,  
 14537172455552,  
 60777304839141  
 ]  
}

# Промежуточные вычисления

Пусть a = , b =

[( )] = (8216502, 10856515), (8216503, 27289520), (8216504, 43722527), (8216505, 60155536)

60155536 - полный квадрат => a = 8216505, b = 7756

p = a + b = 8216505 + 7756= 8224261  
q = a - b = 8216505 - 7756= 8208749

φ(n) = (p - 1)(q - 1) = 67510877826480

𝑑 = 𝑚𝑜𝑑 φ(𝑛) = 13087298491547

# Результаты работы программы

A screenshot of a computer

Description automatically generated