Министерство образования Республики Беларусь
Учреждение образования
“Брестский государственный технический университет”
Кафедра интеллектуально-информационных технологий

Лабораторная работа №3

“Атака на алгоритм шифрования RSA посредством метода бесключевого чтения”

Выполнил:
студент 4 курса
группы ИИ-22
Нестерчук Д. Н.
Проверила:
Хацкевич А. С.

Брест 2024

Цель работы: изучить атаку на алгоритм шифрования RSA посредством метода бесключевого чтения.

Ход работы:

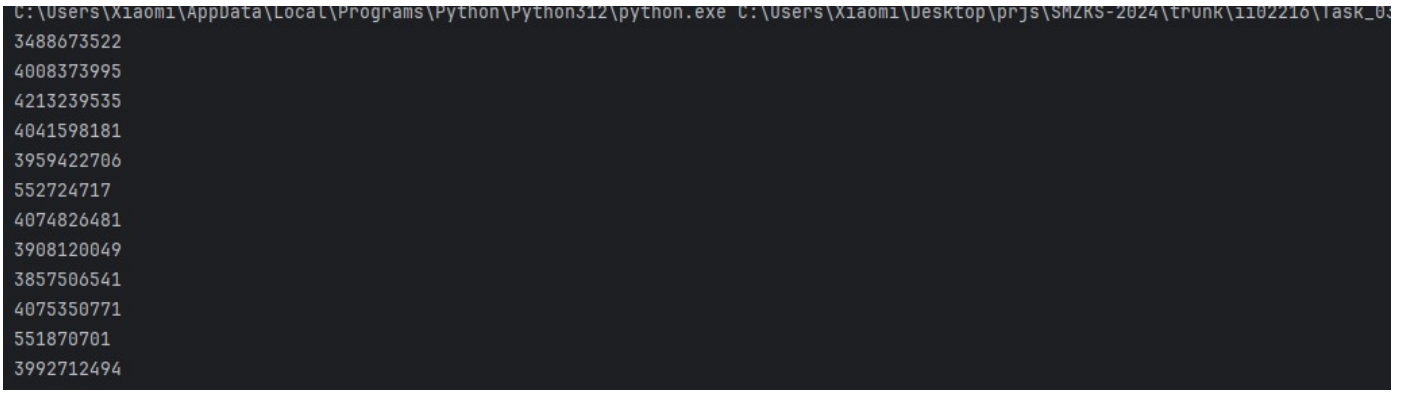
- ознакомиться с теорией;
- по исходным данным определить значения r и s пр условии, что e1*r - e2*s = 1. Для этого необходимо использовать расширенный алгоритм Евклида;
- используя значения r и s, получить исходный текст;
- результаты и промежуточные вычисления значений для любых трех блоков шифрованного текста оформить в виде отчета.

Код программы:

```
from sympy import mod_inverse

class MethodFermat:
    def __init__(self, N, e):
        self.N = N
        self.e = e

    def sqrt(self, x):
        if x < 0:
            raise ArithmeticError("Cannot compute square root of a
negative number")
        if x == 0 or x == 1:
            return x
        a = x
        b = x // 2

        while b < a:
            a = b
            b = (x // b + b) // 2  # (x / b + b) // 2

        return a

    def check_sqrt(self, C):
        sqrt_result = self.sqrt(self.N)
        square = sqrt_result * sqrt_result
```

```
        print(f"Корень: {sqrt_result} Квадрат корня: {square} N:
{self.N}")

        if square == self.N:
            print(f"{sqrt_result} является точным квадратным корнем
числа {self.N}")
        else:
            print(f"{sqrt_result} НЕ является точным квадратным корнем
числа {self.N}")
            w1 = abs(square - self.N)

            while not (w1 == self.sqrt(w1) ** 2):
                sqrt_result += 1
                square = sqrt_result * sqrt_result
                w1 = abs(square - self.N)
                print(f"{sqrt_result} {square} - {self.N} Разность: {w1}
квадрат: {self.sqrt(w1) ** 2} Корень: {self.sqrt(w1)}")

            p = sqrt_result + self.sqrt(w1)
            q = sqrt_result - self.sqrt(w1)
            Composition = (q - 1) * (p - 1)
            print(f"{sqrt_result} p: {p} q: {q} q*p = {Composition}")

            inverse = mod_inverse(self.e, Composition)
            print(f"Обратное к e: {inverse}")

            decrypted_message = pow(C, inverse, self.N)
            print(f"Исходное сообщение: {decrypted_message}")


# Пример использования
N = 3233  # Например, значение N (модуль)
e = 17    # Примерное значение e (открытая экспонента)
C = 855   # Шифрованное сообщение

method_fermat = MethodFermat(N, e)
method_fermat.check_sqrt(C)
```

Вывод программы:



Вывод: освоил на практике основные принципы атаки на RSA