МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА №2.1

**“Атака на алгоритм шифрования RSA,
посредством метода Ферма”**

по дисциплине

“Информационная безопасность”

Студент:

Алексеев Даниил Иннокентьевич

Группа P34302

Преподаватель:

Рыбаков Степан Дмитриевич

Санкт-Петербург

2023

Цели работы:

- Изучить атаку на алгоритм шифрования RSA посредством Китайской теоремы об остатках
- Найти открытый текст шифротекста

Задание (вариант 26)

Для имеющихся значений **N**, **e** и шифрованного сообщения выполнить атаку на алгоритм шифрования, используя уязвимость плохо подобранных чисел **p** и **q**.

N	e	Шифротекст
40874866482797	4890013	30098470920348 10084491526640 23441958595352 33281521148728 37973385618526 9343475069587 2406343345685 7678583166238 37712932671543 31339429556436 26029018118292 35429221689605

Алгоритм RSA

При шифровании алгоритмом RSA исходное сообщение представляется в виде набора блоков-чисел, каждое из которых лежит в [0; N-1]. При отправке сообщение шифруется при помощи открытого ключа – пары чисел (N, e). Получатель должен воспользоваться секретным ключом (N, d). Числа N, e, d удовлетворяют условиям:

1. $N = p \cdot q$, где p и q – простые числа
2. $gcd(e, \varphi(N)) = 1$, т.е. e взаимопростое с значением функции Эйлера для N
3. $e \cdot d \equiv 1 mod\ \varphi(N)$

Числа p, q, и, соответственно, $\varphi(N)$ в дальнейшем не используется и должны быть уничтожены.

Например:

A отправляет B сообщение x

A (N, e)	B (N, d)
$y = x^e\ mod\ N$ y – зашифрованный блок	Сообщение x получается в результате операции: $x = y^d\ mod\ N$

Доказательство второго равенства такое:

Из $e \cdot d \equiv 1 mod\ \varphi(N)$ следует что $e \cdot d = \varphi(N) \cdot k + 1, k \in \mathbb{Z}$. Поэтому можно выполнить преобразование:

$$y^d \equiv (x^e)^d \equiv x^{ed} \equiv x^{\varphi(N) \cdot k + 1} \equiv (x^{\varphi(N)})^k \cdot x\ (mod\ N) \quad (1)$$

Также можно показать, что:

$x \cdot x^{\varphi(N) \cdot k} \equiv x \cdot x^{(p-1) \cdot (q-1) \cdot k} \equiv x \cdot (x^{p-1})^{(q-1) \cdot k} \equiv x(1)^{(q-1) \cdot k} \equiv x\ mod(p)$, согласно малой теореме Ферма

Аналогично:

$x \cdot x^{\varphi(N) \cdot k} \equiv x \cdot x^{(p-1) \cdot (q-1) \cdot k} \equiv x \cdot (x^{q-1})^{(p-1) \cdot k} \equiv x(1)^{(p-1) \cdot k} \equiv x\ mod(q)$

Т.к **p**, **q**, **N** попарно взаимопростые, то по китайской теореме об остатках получим:

$x \cdot x^{\varphi(N) \cdot k} \equiv x\ mod(N)$

тогда (1) превращается в:

$$y^d \equiv x\ (mod\ N)$$

Неудачный выбор параметров

Атака на алгоритм RSA подразумевает выполнение крайне трудоёмкой операции разложения большого числа **N** на простые множители **p** и **q**. Если злоумышленнику удастся получить такое разложение, то он сможет получить секретный ключ и расшифровать сообщение. Стойкость алгоритма RSA напрямую зависит от выбора параметров.

В данный работе рассматривается случай, когда **p** и **q** довольно близки друг другу. Что позволяет выразить N так:

$$N = p \cdot q \approx p^2 \approx q^2 \approx \frac{p^2+q^2}{2} = \frac{(p+q)^2}{2} ¿ + \frac{(p-q)^2}{2}$$, если $p \approx q$

Такой выбор параметров сильно упрощает задачу факторизации числа N.

Пусть $t = \frac{(p+q)^2}{2}; S = \frac{(p-q)^2}{2}$

Можно начать искать число *t* прямо с $t = \sqrt{N}$, и увеличвать его на 1 пока не выполнится равенство: $t^2 - N = S^2$.

Именно так работает метод Ферма по факторизации числа.

Решение задачи

Метод можно перевести в код на Kotlin и выполнить дешифрацию полученного шифрованного сообщения.

```
import java.io.File
import java.math.BigInteger

fun main(args: Array<String>) {
    val text = File(args[0]).readText().split("\n")
    val blocks_n = text[0].toInt()

    val N: BigInteger = BigInteger.valueOf(text[1].toLong())
    val e: BigInteger = BigInteger.valueOf(text[2].toLong())

    val verbose = false

    val secretD = hack(N,e, verbose)

    for(i in 0..<blocks_n){
        val C: BigInteger = BigInteger.valueOf(text[3 + i].toLong())

        if(verbose){
            println("--------------------------------")
            println(String.format("block [%2d] %d", i + 1, C.toLong()))
        }
        val msg = C.modPow(secretD, N)
        if(verbose)
            println(String.format("msg = %d\n--------------------------------", msg.toLong()))
        else
            println(msg.toLong())
    }

}

fun hack(N: BigInteger, e: BigInteger, verbose: Boolean): BigInteger {
    val p = fermaFactorize(N).second
    val q = N.divide(p)

    if(verbose)
    println(String.format("N Factorization: N = %d * %d", p.toInt(), q.toInt()))

    val phi = (p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE))
    if(verbose)
    println(String.format("φ(N)=φ(p)*φ(q)=(p-1)(q-1): φ(N) = %d", phi.toLong()))

    val secret_d = e.modInverse(phi)
    if(verbose)
    println(String.format("d=invmod_phi(e): d = %d", secret_d.toLong()))

    return secret_d
}
```

```
fun fermaFactorize(N: BigInteger): Pair<Boolean, BigInteger> {
    var x = N.sqrt() + BigInteger.ONE

    var y = BigInteger.ZERO
    var r = ((x.multiply(x)).subtract(y.multiply(y))).subtract(N)

    while (true) {
        val cmp = r.compareTo(BigInteger.ZERO)
        if (cmp == 0){
            return if (x.compareTo(y) != 0) Pair(true, x.subtract(y)) else Pair(true, x.add(y))
        } else if(cmp > 0) {
            r = r.subtract(y).subtract(y).subtract(BigInteger.ONE)
            y = y.add(BigInteger.ONE)
        } else{
            r = r.add(x).add(x).add(BigInteger.ONE)
            x = x.add(BigInteger.ONE)
        }
    }
}
```

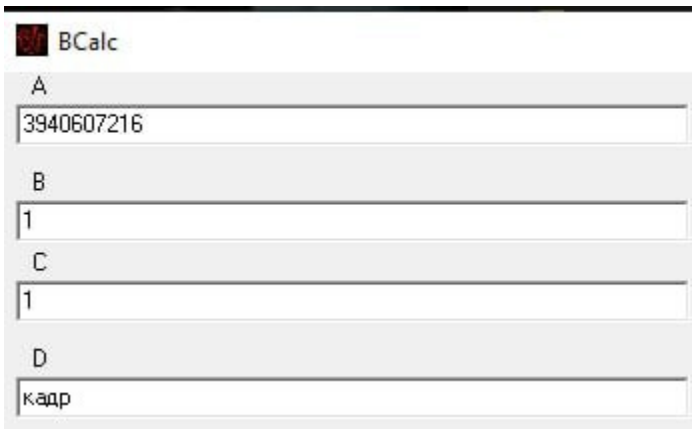
Значение полученных **p**,**q**, и **d**:

```
N Factorization: N = p*q = 6384953 * 6401749
φ(N)=φ(p)*φ(q)=(p-1)(q-1): φ(N) = 40874853696096
d=invmod_phi(e): d = 27434000421013
```

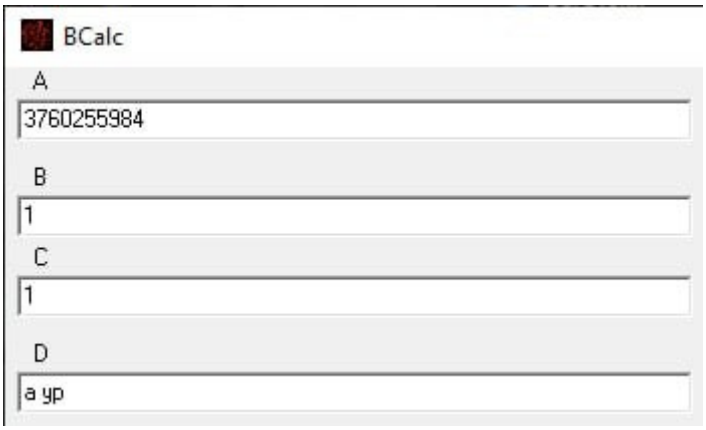
После исполнения програмым получены расшифрованные блоки:

```
3940607216
4213188845
3760255984
4007849445
541346883
690022432
4024824037
4059818976
4277334527
552594976
3991859688
4075154430
```

Воспользуемся программой BCalc для преобразования чисел в текст:


BCalc
A
3940607216
B
1
C
1
D
кадр


BCalc
A
4213188845
B
1
C
1
D
ы (н


BCalc
A
3760255984
B
1
C
1
D
а ур

и т.д.

Полученный текст

кадры (на уровне DLC) и пересылаются по носителю

Вывод

В ходе выполнения лабораторной работы я ознакомился с основами шифрования RSA и способом атаки на алгоритм посредством метода Ферма.