УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Дисциплина «Информационная безопасность»

Лабораторная работа №2.1

Атака на алгоритм шифрования RSA посредством метода Ферма

Вариант 6

Студент

Ершова А. И.

P34302

Преподаватель

Рыбаков С. Д.

Санкт-Петербург, 2023 г.

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

Вариант 6:

Вариант	Модуль, N	Экспонента, e	Блок зашифрованного текста, C
6	85 609 460 573 249	2448539	523815866990 26788001211021 34569932939126 85581094055910 23256663175806 62527703621248 7622521689363 32655715523491 81242663069415 60438288306445 73937478628138 7793112362388

Ход работы

Шаг 1. Определить множители p и q

$n = \sqrt{N} + 1 = 9\ 252\ 539$

$t_1 = n + 1 = 9\ 252\ 540$

$w_1 = t_1^2 - N = 35878351$

w_1 – не квадрат целого числа

$t_2 = n + 2 = 9\ 252\ 541$

$w_2 = t_2^2 - N = 54383432$

w_2 – не квадрат целого числа

$t_3 = n + 3 = 9\ 252\ 542$

$w_3 = t_3^2 - N = 72888515$

w_3 – не квадрат целого числа

$t_4 = n + 4 = 9\ 252\ 543$

$$w_4 = t_4^2 - N = 91393600$$

$$w_4 = 9560^2$$

$$p = t_4 + \sqrt{w_4} = 9\,252\,543 + 9\,560 = 9\,262\,103$$

$$q = t_4 - \sqrt{w_4} = 9\,252\,543 - 9\,560 = 9\,242\,983$$

$$\varphi(N) = (p-1)(q-1) = 85\,609\,442\,068\,164$$

$$d = e^{-1}\ mod\ \varphi(N) = 349635$$

Шаг 2. Расчет значений ф и d

$$\varphi(N) = (p-1)(q-1) = 85\,609\,442\,068\,164$$

$$d = e^{-1}\ mod\ \varphi(N) = 62467989361523$$

Шаг 3. Дешифрация

Для каждого значения в C необходимо произвести следующие вычисления:

$$m_i = C_i^d\ mod\ N$$

В m содержится дешифрованная информация.

Результат:

один раз в каждом кольце, когда кольца связаны _

Листинг разработанной программы

```
import java.math.BigDecimal;
import java.math.BigInteger;
import java.nio.charset.Charset;
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        BigInteger N = new BigInteger("856094605732 49");
        BigInteger e = new BigInteger("2448539");

        String[] C = {
```

```
                "523815866990",
                "26788001211021",
                "34569932939126",
                "85581094055910",
                "23256663175806",
                "62527703621248",
                "7622521689363",
                "32655715523491",
                "81242663069415",
                "60438288306445",
                "73937478628138",
                "7793112362388"

        };

        String decryptedStr = decrypt(N, e, C);
        System.out.println("decrypted string:");
        System.out.println(decryptedStr);

    }

    static String decrypt(BigInteger N, BigInteger e, String[]
C) {
        System.out.println("N = " + N);
        System.out.println("e = " + e);
        System.out.println("C = ");
        for (String c : C) {
            System.out.println(c);
        }
        BigInteger n = N.sqrt().add(BigInteger.ONE);
        System.out.println("n = [sqrt(N)] + 1 = " + n);
        int i = 0;
        BigInteger t;
        BigInteger sqrt_w;
        while (true) {
            i++;
            t = n.add(BigInteger.valueOf(i));
            System.out.println("t" + i + " = n + " + i + " = " +
t);
            BigInteger w = t.pow(2).subtract(N);
            System.out.println("w" + i + " = " + " = t" + i +
"^2 - N = " + w);
            float sqrt_w_double = (float)
Math.sqrt(w.longValue());
            if (sqrt_w_double % 1 == 0) {
                sqrt_w =
BigDecimal.valueOf(sqrt_w_double).toBigInteger();
                //sqrt_w = (long) sqrt_d1_double;
                System.out.println("sqrt(w" + i + ") = " +
sqrt_w);
                break;
            } else System.out.println("sqrt(w" + i + ") = " +
sqrt_w_double + " - error");
```

```
        }

        //p = t + sqrt_w
        BigInteger p = t.add(sqrt_w);
        //q = t - sqrt_w
        BigInteger q = t.subtract(sqrt_w);
        //phi = (p-1) * (q-1)
        BigInteger phi =
p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
        //long d =
BigInteger.valueOf(e).modInverse(BigInteger.valueOf(phi)).longVa
lue();
        BigInteger d = e.modInverse(phi);

        System.out.println("p = t + sqrt(d" + i + ") = " + p);
        System.out.println("q = t - sqrt(d)" + i + " = " + q);
        System.out.println("phi = (p - 1) * (q - 1) = " + phi);
        System.out.println("d = e^(-1) mod phi = " + d);

        String decryptStr = "";
        i = 0;
        for (String c : C) {
            i++;
            //c^d
            BigInteger m = (new BigInteger(c)).modPow(d, N);
            System.out.println("m" + i + " = " + m);
            String part = decodeWindows1251(m);
            System.out.println("m" + i + " = C[" + i + "]^d mod
N = " + c + "^" + d + " mod " + N + " = " + m + " => text(" + m
+ ") = " + part);
            decryptStr += part;



        }

        return decryptStr;
    }


    public static String decodeWindows1251(BigInteger
encodedValue) {
        byte[] bytes = encodedValue.toByteArray();

        List<Byte> byteList = new ArrayList<>();
        for (byte b : bytes) {
            if (b != 0) {
                byteList.add(b);
            }
        }
        bytes = new byte[byteList.size()];
        for (int i = 0; i < byteList.size(); i++) {
```

```
            bytes[i] = byteList.get(i);
        }
        try {
            return new String(bytes, Charset.forName("Windows-
1251"));
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

}
```

Результаты работы программы

```
N = 85609460573249

e = 2448539

C =

523815866990

26788001211021

34569932939126

85581094055910

23256663175806

62527703621248

7622521689363

32655715523491

81242663069415

60438288306445

73937478628138

7793112362388

n = [sqrt(N)] + 1 = 9252539

t1 = n + 1 = 9252540

w1 = t1^2 - N = 35878351

sqrt(w1) = 5989.854 - error

t2 = n + 2 = 9252541

w2 = t2^2 - N = 54383432

sqrt(w2) = 7374.512 - error

t3 = n + 3 = 9252542

w3 = t3^2 - N = 72888515

sqrt(w3) = 8537.478 - error
```

t4 = n + 4 = 9252543

w4 = t4^2 - N = 91393600

sqrt(w4) = 9560

p = t + sqrt(d4) = 9262103

q = t - sqrt(d)4 = 9242983

phi = (p - 1) * (q - 1) = 85609442068164

d = e^(-1) mod phi = 62467989361523

m1 = 4007979245

m1 = C[1]^d mod N = 523815866990^62467989361523 mod 85609460573249 = 4007979245 => text(4007979245) = один

m2 = 552657127

m2 = C[2]^d mod N = 26788001211021^62467989361523 mod 85609460573249 = 552657127 => text(552657127) = раз

m3 = 551690474

m3 = C[3]^d mod N = 34569932939126^62467989361523 mod 85609460573249 = 551690474 => text(551690474) = в к

m4 = 3773228270

m4 = C[4]^d mod N = 85581094055910^62467989361523 mod 85609460573249 = 3773228270 => text(3773228270) = аждо

m5 = 3961580270

m5 = C[5]^d mod N = 23256663175806^62467989361523 mod 85609460573249 = 3961580270 => text(3961580270) = м ко

m6 = 3959224037

m6 = C[6]^d mod N = 62527703621248^62467989361523 mod 85609460573249 = 3959224037 => text(3959224037) = льце

m7 = 740354798

m7 = C[7]^d mod N = 7622521689363^62467989361523 mod 85609460573249 = 740354798 => text(740354798) = , ко

m8 = 3823427616

m8 = C[8]^d mod N = 32655715523491^62467989361523 mod 85609460573249 = 3823427616 => text(3823427616) = гда

m9 = 3941526524

m9 = C[9]^d mod N = 81242663069415^62467989361523 mod 85609460573249 = 3941526524 => text(3941526524) = коль

m10 = 4141883633

m10 = C[10]^d mod N = 60438288306445^62467989361523 mod 85609460573249 = 4141883633 => text(4141883633) = ца с

m11 = 3808421856

m11 = C[11]^d mod N = 73937478628138^62467989361523 mod 85609460573249 = 3808421856 => text(3808421856) = вяза

m12 = 3992658015

```
      m12 = C[12]^d mod N = 7793112362388^62467989361523 mod 85609460573249 =
3992658015 => text(3992658015) = ны _

      decrypted string:

      один раз в каждом кольце, когда кольца связаны _
```

Вывод

В ходе выполнения лабораторной работы я изучила метод Ферма для атаки на алгоритм шифрования RSA и реализовала его работу на Java.