Universidade do Minho
Escola de Engenharia

Congruências modulares e RSA

Tecnologia Criptográfica
Trabalho Prático V

Trabalho realizado por:
Filipe Freitas (PG42828)
Maria Barbosa (PG42844)

Dezembro de 2020

Índice

Listagens ii

1 Introdução 1

1.1 Contextualização 1

1.2 Estrutura do relatório 1

2 Congruências modulares 2

2.1 Alínea a) 2

2.2 Alínea b) 3

3 Decifragem do RSA com parâmetros inseguros 5

3.1 Inversão da codificação dos blocos de três caracteres 5

3.2 Cálculo do fator d 6

Listagens

Ficheiro *tp5.py*, linhas 4-21 6
Ficheiro *tp5.py*, linhas 74-82 6
Ficheiro *tp5.py*, linhas 93-95 6
Ficheiro *tp5.py*, linhas 97-101 7
Resultado da decifragem 7

Capítulo 1

Introdução

1.1 Contextualização

Na sequência da UC de Tecnologia Criptográfica foi proposto o presente trabalho cujo enunciado se encontra dividido em duas partes: na primeira pretende-se consolidar os conceitos sobre congruências modulares lecionados nas aulas, através da resolução de dois sistemas de congruências modulares; na segunda parte pretende-se a realização da análise e decifragem de um criptograma cifrado com uma cifra RSA.

1.2 Estrutura do relatório

Este relatório divide-se em três partes principais.

A primeira, correspondente a esta introdução, pretende contextualizar o trabalho e explicar a sua estrutura.

Na segunda parte apresentamos os cálculos realizados para resolver os dois sistemas de congruências modulares.

Na terceira parte explicamos o processo utilizado para quebrar a cifra RSA apresentada.

Capítulo 2

Congruências modulares

O Teorema Chinês dos Restos (TCR) diz-nos que um sistema de congruências lineares cujos módulos $(m_1, m_2, ..., m_r)$ são primos entre si possui uma solução única modulo $m_1 \cdot m_2 \cdot ... \cdot m_r$.

2.1 Alínea a)

Para a alínea a), sabemos que $gcd(13, 23) = gcd(23, 27) = 1$ e que, pelo TCR, existe uma única solução x módulo $13 \cdot 23 \cdot 27 = 8073$.

$$\begin{cases} x \equiv 48 \ (mod\ 13) \\ x \equiv 57 \ (mod\ 23) \\ x \equiv 39 \ (mod\ 27) \end{cases} \Leftrightarrow \begin{cases} x \equiv 9 \ (mod\ 13) \\ x \equiv 11 \ (mod\ 23) \\ x \equiv 12 \ (mod\ 27) \end{cases}$$

Se começarmos pela primeira equação, temos que qualquer solução x do sistema tem que satisfazer:

$$x = 13 \cdot k + 9 \qquad (1)$$

para algum $k \in \mathbb{Z}$; substituindo na segunda equação ficamos com:

$$13 \cdot k + 9 \equiv 11 \ (mod\ 23) \Leftrightarrow 13 \cdot k \equiv 2 \ (mod\ 23) \Leftrightarrow k \equiv 9 \ (mod\ 23)$$

Este último passo é justificado por: $gcd(13, 23) = 1$ e $13 \times 9 \equiv 2 \pmod{23}$.

Então, podemos escrever que:

$$k = 23 \cdot q + 9 \qquad (2)$$

onde mais uma vez, q representa uma nova incógnita, tal que $q \in \mathbb{Z}$.

Substituindo a equação (2) em (1), temos:

$$x = 13 \cdot (23 \cdot q + 9) + 9 = 299 \cdot q + 126 \qquad (3)$$

.

Substituindo na terceira equação:

$$\begin{aligned} & 299 \cdot q + 126 \equiv 12 \pmod{27} \\ \Leftrightarrow\ & 299 \cdot q \equiv -114 \pmod{27} \\ \Leftrightarrow\ & 2 \cdot q \equiv -6 \pmod{27} \\ \Leftrightarrow\ & q \equiv -3 \pmod{27} \end{aligned}$$

Este último passo é justificado por: $gcd(2, 27) = 1$ e $2 \cdot (-3) \equiv -6 \pmod{27}$.

Concluímos assim que:

$$\begin{aligned} & q \equiv -3 \pmod{27} \\ \Leftrightarrow\ & q \equiv 24 \pmod{27} \\ \Leftrightarrow\ & q = 27 \cdot w + 24, w \in \mathbb{Z} \end{aligned}$$

Substituindo em (3) obtemos:

$$q = 299\ (27 \cdot w + 24) + 126 = 8073 \cdot w + 7302$$

Deste modo podemos concluir que 7302 é uma solução do sistema de equações. Podemos também concluir que qualquer inteiro da forma $8073 \cdot w + 7302$ é também solução do sistema de equações.

2.2 Alínea b)

Para a alínea b):

$$\begin{cases} 19x \equiv 21\ (mod\ 16) \\ 37x \equiv 100\ (mod\ 15) \end{cases} \Leftrightarrow \begin{cases} 3x \equiv 5\ (mod\ 16) \\ 7x \equiv 10\ (mod\ 15) \end{cases}$$

Comecemos por simplificar as congruências. Como $gcd(3, 16) = gcd(7, 15) = 1$ e $3 \times 7 \equiv 5 \pmod{16}$ e $3 \times 10 \equiv 10 \pmod{15}$

Obtemos o seguinte sistema de congruências modulares:

$$\begin{cases} x \equiv 7\ (mod\ 16) \\ x \equiv 10\ (mod\ 15) \end{cases}$$

Estamos agora em condições de aplicar o TCR, pois $gcd(16, 15) = 1$, e então concluímos que existe seguramente uma solução x, que é única modulo $16 \times 15 = 240$.

Se começarmos pela primeira equação, temos que qualquer solução x do sistema tem que satisfazer:

$$x = 16 \cdot k + 7 \qquad (1)$$

para algum $k \in \mathbb{Z}$; substituindo na segunda equação ficamos com:

$$16 \cdot k + 7 \equiv 10 \pmod{15} \Leftrightarrow 16 \cdot k \equiv 3 \pmod{15} \Leftrightarrow k \equiv 3 \pmod{15}$$

Este último passo, é justificado por: $gcd(16, 15) = 1$ e $16 \times 3 \equiv 3 \pmod{15}$.

Então, podemos escrever que:

$$k = 15 \cdot w + 3 \qquad (2)$$

onde mais uma vez, w representa uma nova incógnita, tal que $w \in \mathbb{Z}$. Substituindo a equação (2) em (1), temos:

$$x = 16\ (15 \cdot w + 3) + 7 = 240 \cdot w + 55$$

.

Deste modo podemos concluir que 55 é uma solução da equação, assim como qualquer inteiro da forma $240 \cdot w + 55$. s

Capítulo 3

Decifragem do RSA com parâmetros inseguros

Na segunda parte deste trabalho foi pedido para decifrar um bloco de texto cifrado com recurso ao RSA, mas com a particularidade de que o módulo utilizado é demasiado pequeno para ser seguro. Assim sendo, é fácil quebrar a segurança: basta, mesmo que por força bruta, calcular os fatores p e q de n, e portanto, torna-se fácil calcular o parâmetro privado d que permite fazer a decifragem do RSA.

No entanto, ainda existe o problema de inverter a codificação dos blocos de três caracteres. Uma solução simples passa por força bruta: calcular, antecipadamente, todos as as combinações de três caracteres e a sua correspondente codificação, de acordo com o método fornecido. No entanto, isto é uma solução ineficiente.

3.1 Inversão da codificação dos blocos de três caracteres

Um trio de caracteres (a, b, c), após ser convertido em números de 0 a 27, é codificado da seguinte forma:

$$d = 27^2 \cdot a + 27 \cdot b + c$$

Para inverter isto, basta entender que $27 * b + c < 27^2$ e $c < 27$. Assim sendo:

$$a = \left\lfloor \frac{d}{27^2} \right\rfloor$$
$$b = \left\lfloor \frac{d \bmod 27^2}{27} \right\rfloor$$
$$c = (d \bmod 27^2) \bmod 27$$

Na implementação existe a particularidade de que é necessário passar do intervalo de 0-27 para o intervalo de 0x41 até 0x5A, e é necessário também converter o carácter [para o carácter espaço.

Assim sendo, temos a seguinte implementação:

Ficheiro *tp5.py*

```
4  def convert_to_plaintext(numbers):
5      # Dados blocos de texto (na forma de um inteiro 27 * 27 * L1 + 27 * L2 + L3), conveter
       ↪  esses blocos para texto limpo
6      plaintext = ''
7
8      for d in numbers:
9          a = chr(ord('A') + d // (27 ** 2))
10         b = chr(ord('A') + (d % (27 ** 2)) // 27)
11         c = chr(ord('A') + (d % (27 ** 2)) % 27)
12
13         block = a + b + c
14         # O caracter espaço é representado por 0x20, enquanto que a primeira letra do
           ↪  alfabeto é representada por 0x41.
15         # Isto significa que o número 26 vai ser descodificado no caracter 0x5B, que é o
           ↪  caracter [, e portanto, vamos fazer uma simples
16         # substituição para substituir o [ por um espaço, de modo a que o resultado final
           ↪  fique correto
17         block = block.replace('[', ' ')
18
19         plaintext += block
20
21     return plaintext
```

3.2 Cálculo do fator d

Para calcular o fator d, é primeiro preciso calcular os fatores p e q. Visto que o módulo n é relativamente pequeno, podemos utilizar um algoritmo de força bruta para fazer este cálculo:

Ficheiro *tp5.py*

```
74 # Calcular o fator p tal que n % p == 0 por força bruta
75 # Como n é relativamente pequeno, este processo é rápido o suficiente
76 for x in range(2, n):
77     if n % x == 0:
78         p = x
79         break
80
81 # O outro fator, q, é simplesmente n // p
82 q = n // p
```

Agora temos o cálculo do valor de $\phi(n) = (p-1) \cdot (q-1)$, pois p, q são primos e $n = p \cdot q$.

Podemos agora, recorrendo ao Algoritmo de Euclides, calcular o valor de $d \equiv e^{-1} \pmod{\phi(n)}$:

Ficheiro *tp5.py*

```
93 # Calcular o parâmetro d privado tal que
94 # d = e^-1 (mod n)
95 d = modinv(e, totient_n)
```

Munidos do parâmetro d secreto, podemos agora decifrar a mensagem fornecida e converter o resultado para texto limpo usando o método descrito em 3.1:

Ficheiro *tp5.py*

```
97  # Decifrar o RSA usando o parâmetro d privado
98  decrypted_numbers = rsa(numbers, n, d)
99
100 # Converter os blocos de três letras nas suas letras componentes
101 print(convert_to_plaintext(decrypted_numbers))
```

O resultado da decifragem é, portanto:

Resultado da decifragem

```
LET US THEREFORE PERMIT THESE NEW HYPOTHESES TO BECOME KNOWN TOGETHER WITH THE ANCIENT HYPOTHESES
WHICH ARE NO MORE PROBABLE LET US DO SO ESPECIALLY BECAUSE THE NEW HYPOTHESES ARE ADMIRABLE AND
ALSO SIMPLE AND BRING WITH THEM A HUGE TREASURY OF VERY SKILLFUL OBSERVATIONS SO FAR AS
HYPOTHESES ARE CONCERNED LET NO ONE EXPECT ANYTHING CERTAIN FROM ASTRONOMY WHICH CANNOT FURNISH
IT LEST HE ACCEPT AS THE TRUTH IDEAS CONCEIVED FOR ANOTHER PURPOSE AND DEPART FROM THIS STUDY A
GREATER FOOL THAN WHEN HE ENTERED IT FAREWELL
```