

Fundação CECIERJ - Vice Presidência de Educação Superior a Distância
Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação I
AD1 1º semestre de 2013.

GABARITO

1. Enunciado

Escreva um programa para criptografar números inteiros pelo algoritmo RSA (<http://pt.wikipedia.org/wiki/RSA>). Numa linguagem como Pascal ou C, manipular primos com 100 dígitos decimais só é possível com auxílio de bibliotecas adicionais, como o GMP (<http://gmplib.org/>). Portanto, a sua implementação vai usar dois primos pequenos, $p = 1217$ e $q = 1223$, que permitem implementar o algoritmo usando apenas inteiros de 64 bits (BigInteger). Da mesma forma, considere $e = 17$. Algumas sugestões e requerimentos da implementação:

- Neste local há uma boa discussão do algoritmo de Euclides estendido: [http://pt.wikipedia.org/wiki/Algoritmo de Euclides estendido](http://pt.wikipedia.org/wiki/Algoritmo_de_Euclides_estendido)
- A sua interface deve permitir-lhe testar a sua implementação, bem como criptografar e decriptografar inteiros no seguinte intervalo: $-1488389 \leq n \leq 1488390$.

Basicamente, são necessários apenas quatro procedimentos, a saber:

- // retorna o máximo divisor comum estendido de "a" e "b"
// $\text{gcd}(a, b)$ e (x, y) , onde x e y satisfazem a equação: $ax + by = \text{gcd}(a, b)$
function eGCD(a,b: BigInteger; var x, y: BigInteger): BigInteger;
- // retorna o inverso modular de "a": $d * a = 1 \bmod m$
function modinv(a, m: BigInteger): BigInteger;
- // retorna $c^d \bmod n$
function power_mod_n(c, d, n: BigInteger): BigInteger;
- // retorna o resto $(a \bmod b)$ correto, se este tiver o sinal oposto de "b"
// em Pascal, $17 \bmod -4 = 1$, $18 \bmod -4 = 2$, $19 \bmod -4 = 3$, $20 \bmod -4 = 0$
// mas deveria ser -3 , -2 , -1 e 0 : ex, $17 / -4 = -4 * -4 + 1$, mas $17 / -4 = -5 * -4 - 3$, também
function fixRemainder(a, b: BigInteger): BigInteger;

Solução

1 Descrição do algoritmo proposto

O RSA é um algoritmo de chave pública idealizado por Ron Rivest, Adi Shamir e Leonard Adleman. O nome do algoritmo deriva das iniciais dos sobrenomes dos autores. O RSA é um algoritmo assimétrico que envolve um par de chaves, uma chave pública que pode ser conhecida por todos e uma chave privada que deve ser mantida em sigilo. Toda mensagem cifrada usando uma chave pública só pode ser decifrada usando a respectiva chave privada.

Para gerar as chaves são necessárias as seguintes etapas:

1. Encontrar $n = pq$, onde p e q são primos.
2. Calcular a função *totiente* em n : $\phi(n) = (p - 1)(q - 1)$
3. Escolher $e < n$ tal que $\gcd(e, \phi) = 1$
4. Computar d de forma que $de \equiv 1 \pmod{\phi(n)}$, ou seja, d é o inverso multiplicativo de e em $(\pmod{\phi(n)})$.
5. Finalmente, temos:
 - Chave pública: (n, e)
 - Chave privada: (n, d)

1.1 Cifração

Para transformar uma mensagem m (no nosso caso, apenas um inteiro $\in [-1488389, 1488390]$), numa mensagem c cifrada, usando a chave pública do destinatário (n, e) , basta fazer uma potenciação modular:

$$c = m^e \pmod{n} \quad (1)$$

A mensagem então pode ser transmitida em canal inseguro para o receptor.

1.2 Decifração

Para recuperar a mensagem original m , a partir da mensagem cifrada c , usando a respectiva chave privada do receptor (n, d) , basta fazer outra potenciação modular:

$$m = c^d \pmod{n} \quad (2)$$

2 Descrição da implementação

As quatro funções sugeridas serão apresentadas aqui resumidamente, e estão detalhadas na seção “Notes on practical applications” em http://www.di-mgt.com.au/rsa_alg.html

- **function eGCD(a,b: BigInteger; var x, y: BigInteger): BigInteger;**

A função deve retornar o máximo divisor comum estendido de a e b . Logo, tem-se que:

$$eGCD(a, b) = ax + by = GCD(a, b). \quad (3)$$

- **function modinv(a, m: BigInteger): BigInteger;**

Essa função retorna o inverso modular de a , tal que: $d * a = 1 \bmod m$. Para isso, usa-se a função eGCD:

- calcule $egcd(a, m, x, y)$.
- se o retorno for diferente de 1, então o inverso modular não existe.
- caso contrário, retorne $d = fixRemainder(x, m)$.

- **function power_mod_n(c, d, n: BigInteger): BigInteger;**

Calcular $c^d \bmod n$ pode gerar o problema de *overflow* (mesmo utilizando INT64). Por isso, essa função deve retornar o resto sem efetuar a potênciação diretamente.

Exemplo: calculando $p = c^d \bmod n^1$, para $c = 30$, $d = 13$ e $n = 35$.

Passo 0: $p = 30^{13} \bmod 35$

Passo 1: $p = (30^{12}) * 30 \bmod 35$

Passo 2: $30^{12} = (30^6)^2$

Passo 3: $30^6 = (30^3)^2$

Passo 4: $30^3 = (30^2) * 30$

Passo 5: $p = (30^2 * 30) \bmod 35$

Agora, resolvendo de baixo para cima:

Passo 5: $p = 30^2 \bmod 35 = 900 \bmod 35 = 25$ (já que $25 * 35 = 875 + 25 = 900$)

Passo 4: $p = 25 * 30 \bmod 35 = 750 \bmod 35 = 15$ (antes era $30^2 * 30 \bmod 35$)

Passo 3: $p = 15^2 \bmod 35 = 225 \bmod 35 = 15$ (antes era $30^6 = ((30^2) * 30^{(1)})^2 \bmod 35$)

Passo 2: $p = 15^2 \bmod 35 = 225 \bmod 35 = 15$ (antes era $30^{12} = (30^6)^2 \bmod 35$)

Passo 1: $p = 15 * 30 \bmod 35 = 450 \bmod 35 = 20$

Portanto: $p = 30^{13} \bmod 35 = 20$

¹Fonte: [http://www.braghetto.eti.br/files/Trabalho Oficial Final RSA.pdf](http://www.braghetto.eti.br/files/Trabalho%20Oficial%20Final%20RSA.pdf)

- **function fixRemainder (a, b: BigInteger): BigInteger;**
 // retorna o resto (a mod b) correto, se este tiver o sinal oposto de "b".

```

1 function fixRemainder (a, b: BigInteger): BigInteger;
2 var r: BigInteger;
3 begin
4     r := a mod b;
5     if ((r * b) < 0) then
6         fixRemainder := b + r
7     else
8         fixRemainder := r;
9 end;
```

3 Interface

A Figura 1 apresenta a interface do sistema proposto. Como os valores p , q e e foram fixados na descrição do enunciado, o único valor de entrada é a "Mensagem Original" que representa a mensagem que deverá ser criptografada.

RSA

Curso de Tecnologia em Sistemas de Computação
 Disciplina: Programação I - AD1 1 semestre de 2013.
 Algoritmo RSA

Valores de entrada

p: 1217 q: 1223 e: 17

Calculos intermediários

Mostrar valores calculados

n: 1488391
 d: 87409
 Phi: 1485952
 eGCD: x=-204, y=203

Chave Pública: (1488391, 17)
 Chave Privada: (1488391, 87409)

Mensagem Original: 2 **Criptografar ->** Mensagem Criptografada: 131072

Mensagem Criptografada: 131072 **Decriptografar ->** Mensagem Original: 2

Figura 1: Interface criptografando e deciptografando o valor 2.