

Criptografia: Parte 12 (Método RSA-5)

Mário Leite

...

As partes **10** e **11** mostraram o algoritmo do método de criptografia RSA, baseado em chaves assimétricas. Este é um sistema de proteção de mensagens muito eficiente e empregado na maioria dos casos em que há necessidade de proteger dados e informações sigilosos. E conforme foi explicado, a segurança do Método RSA está na dificuldade do pretenso criptoanalista em fatorar um número **n** muito grande para descobrir **p** e **q** e depois o parâmetro **d** da chave privada (**d,n**) para tentar a decodificação da mensagem; mas, isto poderia levar até milhões de anos para ser conseguido, conforme já explicado. Este é o principal argumento a favor da segurança deste método de criptografia. Além do mais, a matemática requerida para calcular **d** é não trivial pois, envolve álgebra modular e congruências do tipo $a \equiv b \pmod{n}$ e o cálculo de **d** em expressões do tipo *inverso*[(**a mod b**)]. Entretanto, este rigor matemático pode ser driblado pelo programador com um pouco de raciocínio lógico através de simples *loop* que resolve satisfatoriamente o problema (como foi mostrado no programa “PotMod”, apresentado na **Parte 11**). A função **FunChavePri()** do programa “ProgCriptoRSA” (referenciado mais abaixo) usa este código para resolver o problema de maneira bem simples e sem sofisticções. Neste programa acontece algo semelhante nas expressões requeridas do tipo $a^x \pmod{n}$ (para **x** muito grande).

O programa “ProgCriptoRSA” é uma solução simples, porém, bem didática e correta de como *encriptar/decriptar* um texto através do método RSA. E por questões de praticidade, em vez de usar o sistema ASCII para definir os códigos dos caracteres da mensagem a ser encriptada, optou-se por numerar os caracteres de acordo com a **tabela RSAI.1** (mostrada no início deste método, na **Parte 08**), considerando só letras maiúsculas além do espaço cujo valor convencional foi **55**. Isto torna as operações de potenciação mais “leves” nos cálculos e no processo computacional de um modo geral. O programa foi escrito modularmente com sete rotinas, além do programa principal:

- Função **FunCalcMDC(N1,N2)**: retorna o máximo divisor comum de dois números.
- Função **FunVerifPrimo(N)**: recebe um número e verifica se é primo, retornando *Verdadeiro* se for primo ou *Falso* se não for.
- Procedimento **ProPreCodifTexto(texto)**: faz a pré-codificação da mensagem de acordo com a tabela de caracteres.
- Procedimento **ProCodifTexto(c,n)**: recebe dois números inteiros (**c** e **n**) e gera o texto codificado no vetor VetCodif[].
- Função **FunChavePub(n)**: recebe um valor ($n=p*q$) e retorna o elemento **c** da chave pública (**c,n**).
- Função **FunChavePri(c,y)**: recebe dois números inteiros [**c** e $y=(p-1)*(q-1)$] e retorna o elemento **d** da chave privada (**d,n**).
- Procedimento **ProDeCodifTexto(d,n)**: recebe dois números inteiros (**d** e **n**) e gera o texto decodificado no vetor VetTexto[].

As **figuras RSAV.1, RSAV.2 e RSAV.3** mostram um exemplo de encriptação/decriptação de uma mensagem,

```
Mensagem original: O BEBE BABA

Elementos básicos para o Método RSA
p: 59
q: 113
n: (p*q) = 6667
y: (p-1)*(q-1) = 6496
e: 3
d: 4331
Chave Pública: (3,6667)
Chave Privada: (4331,6667)

Mensagem pré-codificada com caracteres segundo a tabela dada
2455111411145511101110

Mensagem codificada com a chave pública
4906367133127441331274463671331100013311000

Mensagem decodificada com a chave privada
2455111411145511101110

Mensagem original decodificada
O BEBE BABA

*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

Figura RSAV.1 - Exemplo de criptografia com o programa “ProgCriptoRSA”

```
Mensagem original: YROWH D URPD SDUD YLJLDU EUXWXV

Elementos básicos para o Método RSA
p: 59
q: 113
n: (p*q) = 6667
y: (p-1)*(q-1) = 6496
e: 3
d: 4331
Chave Pública: (3,6667)
Chave Privada: (4331,6667)

Mensagem pré-codificada com caracteres segundo a tabela dada
34272432175513553027251355281330135534211921133055143033323331

Mensagem codificada com a chave pública
5969634949061004913636721976367332634922912197636719512197332219763675969259419225942197332636727443322602610026023123

Mensagem decodificada com a chave privada
34272432175513553027251355281330135534211921133055143033323331

Mensagem original decodificada
YROWH D URPD SDUD YLJLDU EUXWXV

*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

Figura RSAV.2 - Exemplo de criptografia com o programa “ProgCriptoRSA”

```
Mensagem original2: A CASA CAIU PESSOAL. A LIBÉLULA ESVOAÇANTE ENTREGOU A GENTE,
E A POLÍCIA ESTÁ CHEGANDO MUITO PERTO; NÃO PODEMOS MAIS NOS ARRISCAR. REPASSE LOGO
O DINHEIRO AO AZUL CONFORME COMBINAMOS.

Elementos básicos para o Método RSA
p: 191
q: 211
n: (p*q) = 40301
y: (p-1)*(q-1) = 39900
e: 3
d: 39901
Chave Pública: (3,40301)
Chave Privada: (39901,40301)

Mensagem pré-codificada de acordo com uma tabela de caracteres dada
10671210281067121018306725142828241021486710672118114021302110671428
31241047102329146714232927141624306710671614232914496714671067252421
42121810671428293767121714161023132467223018292467251427292450672336
2467252413142224286722101828672324286710272182812102748672714251028
28146721241624672467131823171418272467102467103530216712242315242722
14671224221118231022242848

Mensagem codificada com a chave pública
10001865617281000219521000186561728100058322700018656156252744219522
19521382410009261299901865610001865692615832133123699926127000926110
00186562744219522979113824100023221100012167243892744186562744121672
43891968327444096138242700018656100018656409627441216724389274437047
18656274418656100018656156251382492613378717285832100018656274421952
24389103521865617284913274440961000121672197138241865610648270005832
24389138241865615625274419683243891382440971865612167635513824186561
56251382421972744106481382421952186561064810005832219521865612167138
24219521865610001968319683583221952172810000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000

Mensagem a ser decodificada com a chave privada
100018656172810002195210001865617281000583227000186561562527442195221
952138241000926129990186561000186569261583213312369992612700092611000
186562744219522979113824100023221100012167243892744186562744121672438
919683274440961382427000186561000186564096274412167243892744370471865
627441865610001865615625138249261337871728583210001865627442195224389
103521865617284913274440961000121672197138241865610648270005832243891
382418656156252744196832438913824409718656121676355138241865615625138
242197274410648138242195218656106481000583221952186561216713824219521
86561000196831968358322195217281000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000

Mensagem original decodificada
A CASA CAIU PESSOAL. A LIBÉLULA ESVOAÇANTE ENTREGOU A GENTE, E A POLÍCIA ESTÁ CHEGANDO
MUITO PERTO; NÃO PODEMOS MAIS NOS ARRISCAR. REPASSE LOGO O DINHEIRO AO AZUL CONFORME
COMBINAMOS.

*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

Figura RSAV.3 - Exemplo de criptografia com o programa “ProgCriptoRSA”

Nota: O código-fonte completo (em pseudocódigo) do programa “ProgCriptoRSA” e de mais outros métodos de criptografia, estão no meu livro **“1001 Programas Prontos Para Você Codificar Na Sua Linguagem Preferida”**, que pode ser adquirido nas Livrarias Amazon e Clube de Autores (em e-book) ou em pdf, diretamente com o autor pelo e-mail: marleite@gmail.com; neste caso, o código-fonte (em Visualg) também será disponibilizado.

Mário Leite

1001
PROGRAMAS
PRONTOS
para Você Codificar
na Sua
Linguagem
Preferida

Mário Leite

1001 PROGRAMAS PRONTOS

30

