

Primeiro Projeto de Matematica Discreta

Henrique da Silva
hpsilva@proton.me

13 de agosto de 2022

Sumário

- 1 Introdução
- 2 O Codificador de texto
 - 2.1 TextoParaInteiro
 - 2.2 InteiroParaTexto
 - 2.3 Restricoes e limitacoes
- 3 A classe *BigNumber*
 - 3.1 Multiplicacao de *BigNumber*
 - 3.2 Soma de *BigNumber*
- 4 Aritmetica Modular
 - 4.1 AddMod
 - 4.2 MulMod
 - 4.3 ExpMod
 - 4.4 InvMod

1 Introdução

Neste relatório, vamos discutir e implementar o sistema RSA.

Todos arquivos utilizados para criar este relatorio, e o relatorio em si estão em: https://github.com/Shapis/ufpe_ee/tree/main/4thsemester/

2 O Codificador de texto

Este foi criado para transformar uma *string* de texto em um *int*, Atravez de dois metodos. `TextoParaInteiro(string)` e `InteiroParaTexto(int)`.

2.1 TextoParaInteiro

Este metodo recebe um texto e o torna em um *m* do tipo *int* da seguinte maneira:

$$m = \sum_{i=0}^{N-1} cod(a_i) * 27^i \quad (1)$$

Com *a* ate *z* sendo definidos como 1 ate 26, "espaco" sendo definido como 27.

2.2 InteiroParaTexto

Para retornar o texto, este metodo recebe um inteiro *m* e faz a seguinte operacao:

$$a_i = cod\left(\frac{m}{27^i} \pmod{27}\right) \quad (2)$$

Para todo *i* que nao faca *m* ser menor que

2.3 Restricoes e limitacoes

A principal restricao eh que isto foi implementado usando o tipo *int* do *C#* que tem 32 bits. Porem, ja que ele contem tanto numeros positivos quanto negativos o valor maximo dele eh de:

$$\frac{2^{32}}{2} - 1 = 2147483647 \quad (3)$$

Estamos codificando o texto de maneira que cada digito ocupa ate: $2^N = 27$, $N = \frac{\log 27}{\log 2}$ bits

Entao a quantidade maxima de bits ocupados eh simplesmente $N * L$

Para o nosso caso em especifico, que o tipo *int* tem $2^{31} - 1$ de tamanho, ou seja, seguramente ate 30 bits. Temos que:

$$L * \frac{\log 27}{\log 2} \leq 30 \quad (4)$$

Que nos da $L = 6$, ou seja, podemos seguramente converter ate 6 caracteres para tipo *int* e converte-los de volta.

Vale notar, que isto eh um limite inferior de seguranca. Na verdade temos 6.3 digitos disponiveis, que nos permitiria por exemplo, guardar e recuperar, uma frase de sete digitos do tipo *zzzzzd*, Mas para ter certeza. Tem de ser 6 ou menos digitos.

3 A classe *BigNumber*

Esta sera uma classe que armazenara os numeros que utilizarei para a criacao do RSA.

Utilizarei como base para meu *BigNumber* a classe *BigInteger* do *C#*, que tem limite de tamanho tao grande quanto couber na memoria do computador que o esta utilizando.

Para o nossos fins, queremos um *BigNumber* que tenha no maximo 2048 bits. Entao para todas operacoes de *BigNumber* incluindo a sua propria criacao, criarei um *SafetySizeCheck* que caso o *BigNumber* exceda 2048 bits, ele ira lancar

uma excecao e parar o programa com a mensagem de erro apropriada.

Importante lembrar que inclui o zero no *BigNumber*, entao na verdade o limite superior dele fica da seguinte maneira:

$$BigNumber \leq 2^{2048} - 1 \quad (5)$$

E tambem importante lembrar que todas operacoes de checagem de seguranca ocorreram *apos* a operacao ser realizada.

Ou seja, o programa permitira operacoes inseguras, desde que o *BigNumber* resultante desta operacao insegura nao exceda 2048 bits.

3.1 Multiplicacao de *BigNumber*

Aqui podemos observar o seguinte:

$$2^a * 2^b = 2^{a+b} \quad (6)$$

Entao a multiplicacao de dois *BigNumber* de tamanho a e b , pode no maximo nos dar um *BigNumber* de tamanho $a + b$

3.2 Soma de *BigNumber*

Neste caso temos o seguinte:

$$2^a + 2^a = 2 * (2^a) = 2^1 * 2^a = 2^{a+1} \quad (7)$$

Logo podemos concluir que no maximo a soma de dois numeros de tamanho N bits dara um numero de tamanho $N + 1$ bits.

4 Aritmetica Modular

4.1 AddMod

As limitacoes aqui sao as mesmas da soma de dois *BigNumber* como vimos acima em (7).

A funcao *AddMod* pode no maximo dar um *BigNumber* de tamanho $N + 1$ bits, N sendo o tamanho do maior dos dois *BigNumber*.

4.2 MulMod

Vimos acima em (6) as limitacoes de multiplicacao de dois *BigNumber*.

Entao no maximo a soma dos tamanhos em bits dos nossos *BigNumber* deve dar 2048 que eh o tamanho que escolhemos para o nosso *BigNumber*

4.3 ExpMod

No caso da exponenciacao precisamos que o produto dos tamanhos dos dois *BigNumber* seja menor que 2048

4.4 InvMod