Multiplicando Números Grandes

Mário Leite

...

O princípio básico do Método RSA de criptografia de chave dupla é escolher dois números primos (**p** e **q**) grandes o bastante para criar o produto **n** deles (**n = p*q**) que formará a chave pública, e que também vai compor a chave privada. Então, surge o seguinte problema: "*como obter o produto de dois números primos grandes, de modo que ele seja de difícil fatoração*?! Nas literaturas sobre criptografia, o Método RSA exige dois números primos de, no mínimo, cem dígitos cada um! Entretanto, mesmo se for possível fazer essa multiplicação em alguma calculadora "normal", o resultado poderá ser questionado. No caso dos códigos de programas para fazer tal multiplicação, o resultado pode não ser preciso, pois, existem limitações na própria linguagem para determinado tipo de dado. Por exemplo, o tipo *decimal* em C# abrange a faixa de 1.0×E-28 a 7.9×E28, com precisão de APENAS 29 dígitos! Então, o produto de dois números do tipo decimal com cem dígitos cada um não pode ser conseguido por meios "normais", pois, se a precisão dos fatores já é duvidosa, imagine, então, o produto deles! A calculadora do Windows©, primeira ferramenta a ser lembrada para cálculos numéricos, não suporta essa operação; e até o produto de dois números de "apenas" trinta dígitos cada um, pode ser questionado quanto à sua precisão, como mostra a **figura 1**.

O *link* https://en.wikipedia.org/wiki/RSA_numbers (acesso em 30/09/2020 - 17:21) informa que o maior número **n** de uma chave pública quebrada, na criptografia RSA-250 (até Fevereiro/2020), foi este número de 250 dígitos, de 895 *bits*, abaixo:.

2140324650240744961264423072839333563008614715144755017797754920881418023447140136643345519095804679610992851872470914587687396261921557363047454770520805119056493106687691590019759405693457452230589325976697471681738069364894699871578494975937497937

Os dois fatores primos que produzem esse número foram descobertos em fevereiro/2020, e são os seguintes:

6413528947707158027879019017057738908482501474294344720811685963202453234463023862359875266834770873766192558569463979885336 7 e

3337202759497815655622601060535511422794076034476755466678452098702384172921003708025744867329688187756571898625803693206271 1

E, embora a publicação não fale sobre o tempo exato que levou a fatoração, deve ter sido muito longo se foi utilizado algum algoritmo convencional e executado em computador não quântico. Deste modo, a solução para garantir uma chave privada bem segura, é obter um valor de **n** como produto de dois primos com centenas; talvez, milhares de dígitos. A meta atual (ainda não atingida) é fatorar o número de 2048 *bits* com 617 dígitos para obter os dois números primos que lhe dão origem.

Assim, resolvi criar o programa **"MultiplicaNumeros"**, que é uma simples, e boa solução para resolver o problema da multiplicação de dois números primos gigantes (não a fatoração, que é bem mais complexa: para o RSA-250, por exemplo, levaria mais de 900 anos com uma CPU de apenas um núcleo, executado em computador não quântico).

O programa que desenvolvi lê dois números, de quaisquer tamanhos, e faz o produto deles, dando o resultado correto e muito eficientemente. A função **FunLtrim()** é utilizada para retirar os zeros à esquerda do número resultante da multiplicação. A codificação do programa foi feita em SciLab; um gerenciador de cálculos numéricos (da família do MatLab) que, além de ser uma poderosa ferramenta para resolução de problemas numéricos com uma robusta biblioteca matemática, possui, também, uma linguagem de programação muito flexível e com tipagem dinâmica, que parece uma mistura de C com Pascal, o que facilita a migração do código para outras linguagens. A **figura 2** mostra o programa em ação, multiplicando dos dois números de cento e trinta e cinco dígitos cada um, do RSA-250, confirmando a corretude do programa.

Nota: O algoritmo da solução do programa **"MultiplicaNumeros"** foi obtido MANUALMENTE, com caneta/lápis/papel, sem computador; não foi obtido diretamente no IDE do SciLab! Primeiramente, foi planejado e testado, baseado no que aprendi na 3ª Série Primária - teoricamente equivalente, hoje, à 4ª. Série do Ensino Fundamental. Inicialmente, com dois números de até dez dígitos; e somente depois disso, que a solução foi estendida para números maiores na codificação. É como eu sempre afirmei: "***a solução de um programa deve ser rabiscada, planejada e testada, ANTES de sua codificação final***". Nenhum IDE, por mais "moderno" que seja, não vai resolver o problema; quem resolve o problema é a cabeça do programador. A **figuras 3** mostra uma foto do planejamento do algoritmo do programa, ANTES de sua codificação.

Para adquirir o *pdf/book* de meus livros sobre programação, entre em contado:
marleite@gmail com

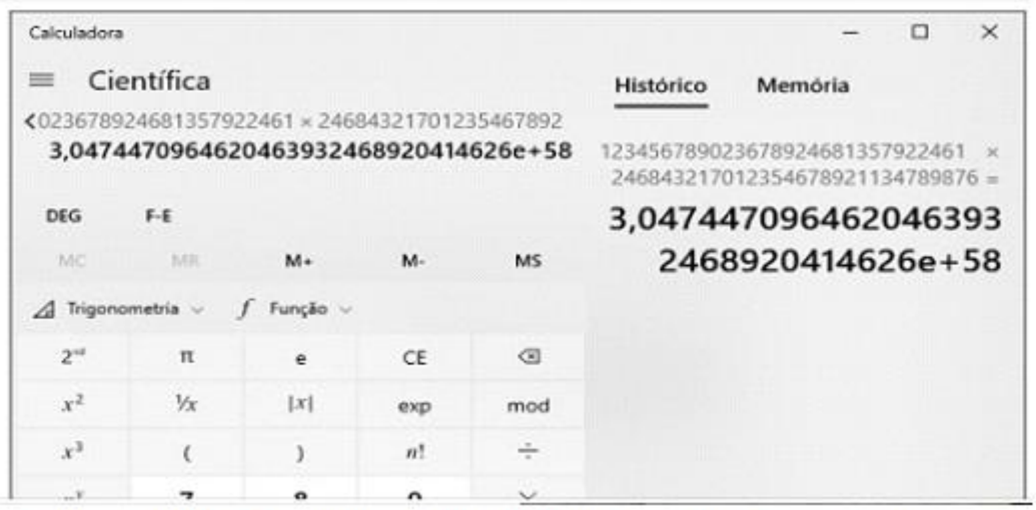

Figura 1 - Usando a calculadora do Windows para multiplicar dois números

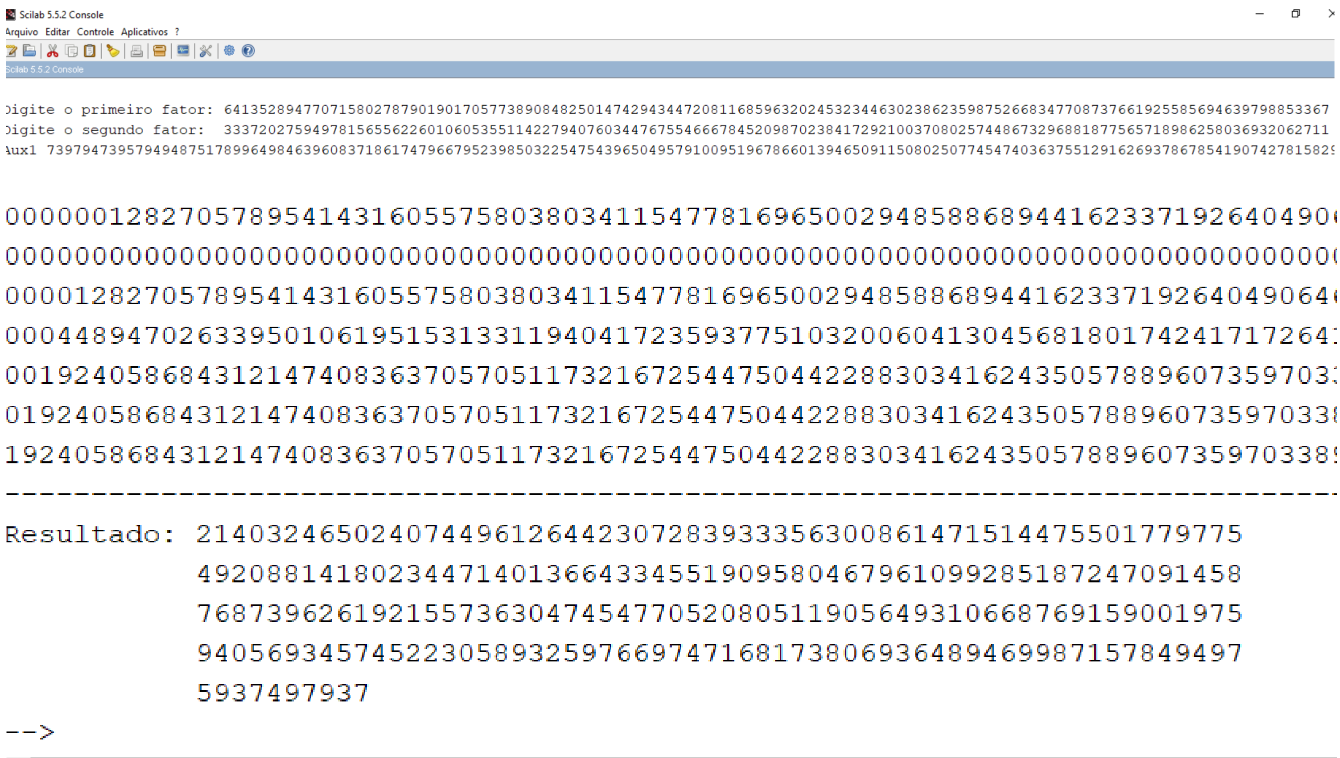

Figura 2 - O programa “MultiplicaNumeros” em ação

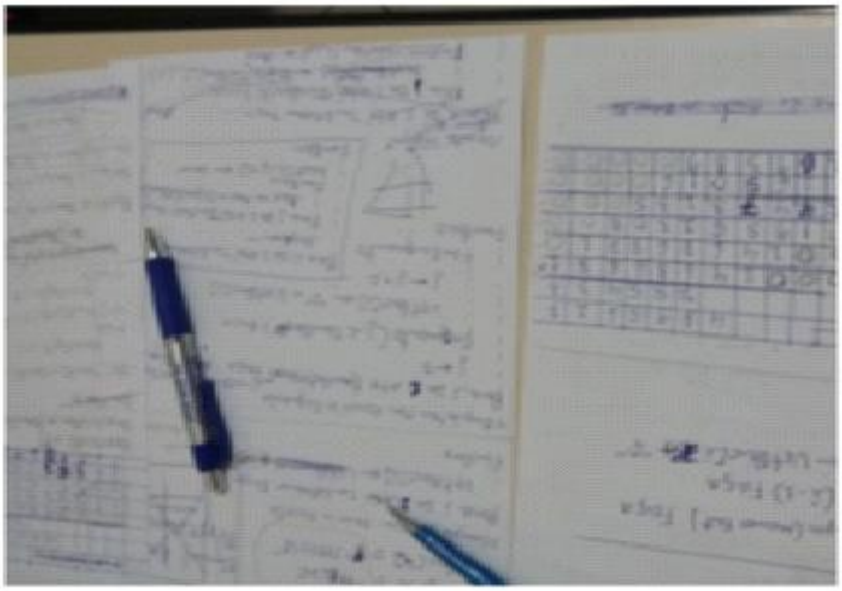

Figura 3 - Programação da solução do problema

```
//MultiplicaNumeros.sce
//Faz multiplicação de dois números
//Implementado em SciLab
//Autor: Mário Leite
//=============================================================================

function retorno=FunLtrim(xString);
//Retorna uma string retirando os espaços da esquerda para a direita.
//-----------------------------------------------------------------------------
   xLTR = length(xString);
   sLTR = "";
   iLTR = 1;
   eLTR = char(48)  //dígito 0
   cLTR = part(xString,iLTR:iLTR) //pega dígitos individuais do número
   while (cLTR == eLTR) do
      iLTR = iLTR + 1;
      cLTR = part(xString,iLTR:iLTR);
      if(iLTR > xLTR) then
         cLTR = "";
      end;
   end;
   sLTR = part(xString,iLTR:xLTR);
   retorno = sLTR;
 endfunction  //fim-Função
//=============================================================================
//Programa principal
   //Entradas dos fatores
   clc;  //limpa a tela do monitor de vídeo
   num1S = input("Digite o primeiro fator: ", "s")
   num2S = input("Digite o segundo fator:  ", "s")

   //Verifica os tamanhos dos fatores como strings
   if(length(num2S) > length(num1S)) then  //troca as posições dos fatores
      troca = num2S;
      num2S = num1S;
      num1S = troca;
   end;

   //Converte os fatores lidos (num1S e num2S) em vetores
   for i=1:length(num1S) do
      digS = part(num1S,i:i)
      VetNum1(i) = eval(digS);
   end;
   for i=1:length(num2S) do
      digS = part(num2S,i:i)
      VetNum2(i) = eval(digS);
   end;

   //Faz multiplicações para todos os dígitos do primeiro fator
   vai = 0;
   Aux1 = "";
   parc = 0;
   for j=length(num2S):-1:1 do
      //Faz multiplicações para um dígito do segundo fator
      for i=length(num1S):-1:1 do
         multi = (VetNum1(i))*(VetNum2(j)) + vai;
         vai   = int(multi/10);  //pega a parte inteira da divisão
         fica  = modulo(multi,10);
         Aux1  = Aux1 + string(fica);
      end;
      if(multi>=10) then  //alcançou ou ultrapassou a base 10
         Aux1 = Aux1 + string(vai);
      end;
      vai = 0;
```

```
   //Inverte Aux1 para obter o resultado correto da multiplicação
    Aux2 = "";
    for k=length(Aux1):-1:1 do
       Aux2 = Aux2 + part(Aux1,k:k);
    end;

  //Monta o vetor da parcela obtida
    parc = parc + 1;
    VetParc(parc) = Aux2;
    Aux1 = "";  //reinicia para obter uma nova multiplicação
end;

//Verifica qual dos dois números tem menor comprimento}
if(length(num1S) < length(num2S)) then
   menorComp = length(num1S);
else
   menorComp = length(num2S);
end;

//Preenche as parcelas com zeros à direita
for i=2:menorComp do
  for j=1:(i-1) do
       VetParc(i) = VetParc(i) + "0";
  end;
end;

//Preenche as parcelas com zeros à esquerda
tamParc = length(num1S) + length(num2S);
for i=1:menorComp do
   j = 1;
   while (length(VetParc(i)) < tamParc) do
      VetParc(i) = "0" + VetParc(i);
      j = j + 1;
   end;
end;

//Obtém o resultado final da multiplicação
Soma = 0;
vai = 0;
Aux1 = "";
tamParc = length(VetParc(parc));
for k=tamParc:-1:1 do
   Soma = 0;
   for j=1:menorComp do
      VetParcN(j) = eval(part(VetParc(j),k:k));
   end;
   for i=1:menorComp do
      Soma = Soma + VetParcN(i);
   end;
   Soma = Soma + vai;
   vai  = int(Soma/10);
   fica = modulo(Soma,10);
   Aux1 = Aux1 + string(fica);
end;

//
printf("%s %s \n", "Aux1", Aux1)
//

//Inverte Aux1 para obter resultado final da multiplicação
Aux2 = "";
for k=length(Aux1):-1:1 do
   Aux2 = Aux2 + part(Aux1,k:k);
end;
Aux2 = FunLtrim(Aux2)  //chama função para retirar os zeros à esquerda
```

```
    //Monta o dispositivo das parcelas das multiplicações
    printf("\n");
    for i=1:parc do
       sleep(400)  //aguarda 400 milissegundos
       printf("%s\n",VetParc(i));
    end;
    for i=1:length(VetParc(1)) do
      sleep(10)
      printf("-");
    end;

    //Exibe o resultado da multiplicação
    printf("\n")
    printf("%s","Resultado: ")
    col = 0
    for j=1:length(Aux2) do
       col = col + 1;
       if(col>60) then  //limita em 60 dígitos por linha
          col = 1; //recomeça na coluna inicial
          printf("\n")  //recomeça na próxima linha
          printf("            ")  //recomeça na próxima linha com espaços
       end;
       DigS = part(Aux2,j:j);
       sleep(20);  //espera 20 milissegundos para imprimir um dígito
       printf("%s",DigS)
    end;
//Fim-Programa ==========================================================================
```