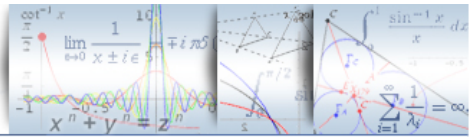
Métodos de Factorização em Números Primos

Vanessa Filipa Vicente Lourenço

DEPARTAMENTO DE MATEMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA



Métodos de Factorização em Números Primos

Vanessa Filipa Vicente Lourenço

Dissertação para a obtenção do Grau de **Mestre em Matemática**

Área de Especialização em **Computação**

Júri

Presidente:	Doutora Cristina Caldeira
Orientador:	Doutor Pedro Quaresma
Vogal:	Doutora Marta Pascoal

Data: 17 de Junho de 2011

Resumo

A factorização de números inteiros é um dos grandes temas de estudo da Teoria dos Números. Actualmente uma das razões do interesse por este tema deve-se à possível existência de métodos eficientes de factorização, que comprometam a segurança dos sistemas criptográficos de chave pública, como, por exemplo, o RSA [1]. O Crivo Quadrático é considerado um dos métodos mais importantes da actualidade a factorizar números inteiros, [4, pag. 169] pois é um dos mais poderosos. Este trabalho tem como objectivo o estudo do método RSA, onde veremos as técnicas que permitem encontrar os factores primos de n, e, para terminar, perceber se o método do Crivo Quadrático é ou não eficiente na quebra da cifra RSA. Para tal, necessitaremos também de estudar diversos conceitos matemáticos envolvidos no tema.

Palavras Chave: Criptografia, RSA, Crivo Quadrático

Abstract

The integers factorization is one of the most important study subject in Number's Theory. Nowadays the concern for this issue exists due to the possible existence of efficient factorization methods, which compromises the security of public key cryptosystems, e.g., the RSA method [1]. The Quadratic Sieve method is considered, in our days, one of the most important methods to factor integers [4, pag. 169], because it is one of the most powerful. This document aims to study the RSA method, where we look at techniques for finding the prime factors of n, and finally, realize that the method Quadratic Sieve is or is not effective in breaking the RSA cipher. To such, we need also to study various mathematical concepts involved in the issue.

Keywords: Criptography, RSA, Quadratic Sieve

Agradecimentos

Ao finalizar mais uma etapa bastante importante da minha vida, não poderia deixar de expressar o mais profundo agradecimento a todos aqueles que me apoiaram nesta longa caminhada e contribuíram para a realização deste trabalho.

Ao Professor Doutor Pedro Quaresma, meu orientador, agradeço toda a disponibilidade e orientação prestada, o apoio, compreensão e paciência que sempre manifestou.

Aos pais, avós, irmão, amigos e toda a família que sempre me apoio e me encorajou a não desistir.

Ao Nuno, por todo o apoio e companheirismo demonstrado.

Aos meus superiores e colegas da ***everis****, que sempre me apoiaram e me ajudaram a conseguir chegar até aqui.*

Um muito obrigado a todos.

Conteúdo

Capítulo 1

Introdução

Qualquer sociedade humana tem por base a capacidade de comunicar. Sem comunicação o entendimento entre pessoas seria impossível, e a evolução teria dado lugar ao caos. No entanto, nem toda a informação que um ser possui pode ser conhecida por todos. Já desde os primórdios que, além de se sentir necessidade de proteger os canais de comunicação entre pessoas da mesma comunidade, também se sente necessidade de esconder o conteúdo da mensagem. Assim, nasceu o conceito de cifrar uma mensagem.

Um dos grandes exemplos da História é o de Júlio César (Imperador Romano), que desenvolveu uma cifra bastante simples que lhe permitia comunicar com os seus Generais sem que mais ninguém percebesse a mensagem: com a "deslocação" de cada letra da mensagem original três posições para a direita. Note-se que esta técnica considera que o alfabeto se fecha sobre si próprio, isto é, que após a última letra vem a primeira. Para decifrar a mensagem o receptor só teria de "deslocar" cada letra três posições para a esquerda para obter a mensagem original. Com o passar do tempo as técnicas de cifragem de mensagens tem sido aperfeiçoadas e cada vez mais usadas, sendo que hoje em dia até muitas das transacções electrónicas na *Internet* têm a cifragem como base.

Ao conjunto de técnicas utilizadas para garantir que uma mensagem só é decifrada pelo seu verdadeiro destinatário chamamos *sistema criptográfico*.

A par da cifragem de mensagens surgiu também a necessidade de as tentar quebrar. A *criptoanálise* tem como objectivo quebrar essa confidencialidade, isto é, desenvolver técnicas que permitam recuperar o texto original sem que se tenha o conhecimento dos segredos por detrás da encriptação.

Um dos sistemas mais conhecidos para o efeito é o sistema RSA, o qual baseia a sua segurança, a sua capacidade de resistir aos ataques criptoanalíticos, na dificuldade da obtenção de uma factorização de uma dado $n = pq$ nos seus factores primos p e q.

Neste estudo, apresentarei o sistema RSA, e algumas técnicas que permitem encontrar os factores primos de n.

Entre os métodos mais conhecidos surgem:

1. Método das Divisões
2. Método da Fórmula Geradora $6k \pm 1$
3. Método de Euclides
4. Método de Fermat

Após o estudo completo de cada um dos métodos descritos, irei compará-los e perceber se a cifra RSA é ou não uma cifra segura. Perceberemos que apesar de ser bastante difícil, é possível quebrar a cifra RSA.

É neste contexto que surge o método do crivo quadrático. O objectivo final deste estudo é então estudar este método e perceber se realmente este método é eficiente na quebra da cifra RSA.

Capítulo 2

Criptoanálise de Sistemas Criptográficos

Como explicado na introdução deste trabalho, um sistema criptográfico é um conjunto de técnicas que permitem que somente o destinatário da mensagem a consiga decifrar, tornando-a legível. Isto é, dada uma mensagem e uma chave, o emissor terá de conseguir gerar uma nova mensagem ilegível que possa ser transmitida por canais desprotegidos, sem que terceiros a consigam compreender sem conhecimento da chave. O sistema só será completo se a mensagem cifrada puder ser recuperada, através de, geralmente, essa mesma ou de outra chave. Existem dois tipos de sistemas criptográficos: sistemas de chaves simétricas e sistemas de chave pública [6].

2.1. Sistema Criptográfico Simétrico

Este tipo de sistema criptográfico também é chamado sistema de chave secreta. A criptografia simétrica, utiliza, em geral, uma única chave que serve tanto para cifrar como para decifrar, isto é, os processos de cifração e decifração são simétricos. Um exemplo deste tipo de criptografia é a cifra de substituição mono-alfabética.

Exemplo 1 (Sistema de Substituição Mono-Alfabético) *Uma cifra de substituição mono-alfabética é uma cifra de substituição onde cada letra do texto é substituída por uma outra letra no texto cifrado, de forma constante.*

Existem assim, tabelas de correspondência, para se codificar e descodificar a mensagem. No entanto, cada letra tem uma frequência própria, e assim a descodificação torna-se mais simples. Logo, este tipo de cifra torna-se extremamente vulnerável para aplicações práticas, porém tem valor teórico, educacional e recreativo.

Um exemplo é a *cifra de Júlio César*: cada letra é substituída por uma outra três posições mais à direita no alfabeto romano.

2.2. Sistema Criptográfico de Chave Pública

Este tipo de criptografia foi uma descoberta de enorme importância, pois assim é possível desenvolver algoritmos em que se usa uma dada chave para cifrar os dados. No entanto, não serve para decifrar, para tal é necessária uma segunda chave diferente da primeira.

Neste caso, a chave de cifragem é pública e a chave de decifragem é privada.

Vamos supor que a entidade A pretende enviar à entidade B uma mensagem. A entidade A cifra-a com a chave pública de B antes do envio. Ninguém, nem sequer a entidade A é capaz de decifrar, apenas a entidade B que possui a chave privada adequada.

Com este tipo de criptografia é facilitada a implementação de mecanismos de autenticação de mensagens e assinatura digital.

Embora tenham sido propostos outros algoritmos, actualmente o RSA é o mais sólido [7]. Outro exemplo é o algoritmo "Merkle-Hellman Knapsacks", que demorou quatro anos a ser quebrado por Adi Shamir. Entretanto foi criada uma segunda versão, supostamente mais sólida, mas que foi quebrada em dois anos.

Estudaremos, no próximo capítulo o algoritmo RSA. Vamos perceber quais os conceitos em que assenta e quais as características em que assenta a sua segurança.

2.3. Criptoanálise dos Sistemas Criptográficos

No âmbito deste trabalho não são apresentados os métodos criptoanalíticos para os sistemas simétricos. Iremos ver sim, de seguida, os métodos criptoanalíticos para o sistema RSA.

O objectivo do criptoanalista é, com base em técnicas de criptoanálise, descodificar o texto codificado. Estudaremos agora algumas técnicas que os criptoanalistas utilizam.

Os sistemas simétricos são caracterizados por possuirem duas técnicas de criptoanálise: *Criptoanálise Linear* e *Criptoanálise Diferencial*.

A técnica de *Criptoanálise Linear* estuda a alta probabilidade de ocorrência de expressões lineares envolvendo bits do texto claro, do texto cifrado e das sub-chaves.

Esta técnica é basicamente um ataque de mensagem conhecida, pois o atacante tem à disposição, além dos textos claros, os correspondentes textos cifrados. Além disso, ele não tem modo de seleccionar os textos claros nem os textos cifrados a que

tem acesso.

Desta forma, nesta técnica, pretende-se determinar expressões que tenham uma alta ou baixa probabilidade de ocorrência [5].

Outra técnica existente é a *Criptoanálise Diferencial*. Ao contrário da Criptoanálise Linear, aqui existe um ataque ao texto claro escolhido, pois além do atacante ter capacidade de seleccionar textos claros e os correspondentes textos cifrados, ainda selecciona pares de textos claros, X' e X'', satisfazendo um ΔX, onde, para esse ΔX, ocorre um ΔY com probabilidade alta [5].

Relativamente aos Sistemas de Chave Pública, as técnicas utilizadas pelos Criptoanalistas assentam na factorização de números primos. Estudaremos, no capítulo seguinte, alguns destes métodos.

Capítulo 3

Criptoanálise da Cifra RSA

O RSA foi o primeiro sistema criptográfico utilizando chave pública implementado e ainda hoje é um dos mais importantes.

Este sistema criptográfico deve o seu nome às iniciais dos nomes dos seus três autores, Ronald Rivest, Adi Shamir e Leonard Adleman.

A sua segurança advêm do facto de ser bastante difícil encontrar uma factorização de um inteiro positivo composto no produto de dois primos.

É um sistema assimétrico pois possui uma chave pública (e, n) e uma chave privada (d, n), onde:

1. $n = pq$, com p e q números primos;

2. $1 < e < \varphi(n)$, em que $\varphi(n)$ representa o número de primos até n;

3. $de = 1(mod\varphi(n))$.

Para obtermos a chave secreta basta-nos, sabendo a chave pública (e, n), factorizar n no produto de números primos p e q.

Assim, vamos obter d, ou seja a chave secreta (d, n).

Qual é então o problema? A factorização de um número nos seus factores primos é uma problema computacionalmente difícil.

Vamos ver alguns métodos de criptoanálise para o sistema RSA [7].

3.1. Método das divisões

Neste método vamos tentar a divisão sucessiva por todos os números primos até $\lfloor\sqrt{n}\rfloor$, ou até que a solução seja encontrada. É também chamado método da força bruta pelas inúmeras divisões que são efectuadas. Para podermos aplicar este método necessitamos de aplicar o famoso crivo de Eratóstenes, onde é encontrado o vector de números primos. Vejamos então o algoritmo do crivo de Eratóstenes.

Algoritmo do Crivo de Eratóstenes:

1. gerar um vector com todos os números inteiros de 2 até n

2. 2 é primo: aplica-se o "crivo 2" à lista, retirando assim o 2 e todos os seus múltiplos

3. 3 é o próximo inteiro e é primo, então aplicamos o "crivo 3"e procedemos de igual forma

4. repetimos o processo até ao fim da lista original. Os elementos que não forem eliminados pelas sucessivas aplicações dos crivos constituem a lista de primos de 2 até n.

Vejamos agora o algoritmo do método das divisões.

```
//Cálculo do limite :
limite = trunc (sqrt (n));
//Construção do vector com todos os primos
//até ao limite pelo crivo de Eratóstenes:
c=crivoEratostenes(limite);
// Número de primos obtidos :
nprimos=columns(c)
//Ciclo de teste desde o primeiro até ao penúltimo primo
while(i>=nprimos && mod(n, c(i))!=0)) {
   if (mod(n,c(i))==0)
      resp=c(i);
   i=i+1;
//Teste para o último primo
if ((i>=nprimos) && mod(n, c(nprimos))==0)
 resp=c(nprimos);
```

A necessidade de gerar todos os primos até um determinado n, utilizando, por exemplo, o crivo de Eratóstenes, e a necessidade de a percorrer $n-1$ vezes para aplicar os sucessivos crivos são alguns dos pontos fracos deste método, pois é acrescentado um peso muito excessivo tanto temporal como espacial.

O método da fórmula geradora é uma alternativa a este método. Vejamos na secção seguinte o porquê.

3.2. Fórmula Geradora $p = 6k \pm 1$

Como foi referido acima, uma alternativa ao método das divisões é então a utilização de uma fórmula que gere todos os números primos sucessivos e outros não primos. Apesar de tornar o ciclo de tentativas de divisão menos eficiente, os ganhos em termos espaciais como temporais, em relação à utilização do crivo de Eratóstenes, compensam as perdas.

```
int formula(unsigned long long int n) {
unsigned long long int limite,p;
limite = trunc(sqrt(n));
if (n%2 == 0 ) return 2;
if (n%3 == 0 ) return 3;
p=5;
while (p <= limite) {
if (n%p == 0 )
return p;
if (n%(p+2) == 0)
return (p+2);
p = p+6;
}
return(0);
}
```

3.3. Método de Euclides

Este método utiliza o algoritmo de Euclides para o cálculo do máximo divisor comum (mdc) entre 2 inteiros. Com o auxilio deste método podemos calcular um dos factores primos de n. Basta para isso multiplicar todos os números primos entre 2 e $\lfloor\sqrt{(n)}\rfloor$ e, de seguida calcular o mdc entre esse produto e n. Assim conseguimos encontrar o factor primo que procurávamos.

Apesar de ser um algoritmo bastante eficiente, apresenta dois pontos fracos:

1. A geração da lista dos números primos até um determinado limite, pois é uma tarefa pesada tanto temporalmente como espacialmente

2. A representação computacional de produtos de números primos, pois rapida-

mente o resultado obtido ultrapassa a capacidade de representação da maioria das linguagens de programação.

Este último ponto fraco apontado ao método foi resolvido dividindo a multiplicação em várias multiplicações parcelares. Este é o primeiro passo do algoritmo em questão.

Algoritmo:

1. Definir os seguintes conjuntos auxiliares:

   $R = r_1, ..., r_m$, onde r_i é um limite inferior $(r_1 = 1, r_i < r_{i+1})$

   $S = s_1, ..., s_m$, onde s_i é um limite superior $(s_i < s_{i+1}, s_{m-1} < \lfloor\sqrt{n}\rfloor \leq s_m)$

2. Para cada r_i e s_i, multiplicar todos os números primos entre estes 2 limites, $P_i = \prod_{r_i \leq p_j \leq s_i} p_j$

3. Para cada um dos P_i calcular $mdc(P_i, n) = a_i$.

4. Se $a_i \neq 1$ então a_i é o factor primo de n que se pretende obter.

Vejamos um exemplo da aplicação do algoritmo.
Seja $n = 1457$, $\lfloor\sqrt{n}\rfloor = 38$. Consideremos as somas parcelares de 10 em 10.

$$R = \{1, 11, 21, 31\} \; e \; S = \{10, 20, 30, 40\}$$

$$\begin{aligned}
P_1 &= \prod_{1<=p_j<=10} p_j = 2 \times 3 \times 5 \times 7 = 210 && mdc(210, 1457) = 1 \\
P_2 &= \prod_{11<=p_j<=20} p_j = 11 \times 13 \times 17 \times 19 = 46189 && mdc(46189, 1457) = 1 \\
P_3 &= \prod_{21<=p_j<=30} p_j = 23 \times 29 = 667 && mdc(667, 1457) = 1 \\
P_4 &= \prod_{31<=p_j<=40} p_j = 31 \times 37 = 1147 && mdc(1147, 1457) = 31
\end{aligned}$$

Assim, 31 é um dos factores primos de 1457.

3.4. Método de Fermat

Consiste em encontrar 2 inteiros a e b que permitam representar o número natural n como a diferença de 2 quadrados.

$$n = a^2 - b^2 \Leftrightarrow n = (a - b)(a + b)$$

Para isso precisamos de uns resultados auxiliares.

Teorema: Qualquer inteiro n ímpar maior do que 1 pode ser escrito como a diferença de 2 quadrados.

Demonstração: Seja $n = pq$, com $p \geq q$. Como, por hipótese, n é ímpar, p e q também o são. Assim,

$$\frac{p+q}{2} \quad e \quad \frac{p-q}{2}$$

são inteiros,

Assim, temos que

$$\begin{aligned}\left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2 &= \\ &= \frac{p^2+2pq+q^2}{4} - \frac{p^2-2pq+q^2}{4} \\ &= \frac{p^2+2pq+q^2-p^2+2pq-q^2}{4} \\ &= \frac{4pq}{4} = pq = n\end{aligned}$$

isto é,

$$n = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$$

1. Dado um inteiro n ímpar fazemos

$$a = \lfloor \sqrt{n} \rfloor + 1$$

2. Se $b = \sqrt{a^2 - n}$ for um número inteiro obtemos o pretendido

3. Caso contrário, incrementa-se uma unidade a a até que b seja um inteiro

Consideremos $n = 25$.

Assim, $a = \lfloor \sqrt{25} \rfloor + 1 = 6$.

$$
\begin{aligned}
b &= \sqrt{6^2 - 25} = 3.32 \\
b &= \sqrt{7^2 - 25} = 4.90 \\
b &= \sqrt{8^2 - 25} = 6.24 \\
b &= \sqrt{9^2 - 25} = 7.48 \\
b &= \sqrt{10^2 - 25} = 8.66 \\
b &= \sqrt{11^2 - 25} = 9.80 \\
b &= \sqrt{12^2 - 25} = 10.91 \\
b &= \sqrt{13^2 - 25} = 12
\end{aligned}
$$

Assim,

$$n = 13^2 - 12^2.$$

3.5. Comparação entre os vários Métodos

Depois de implementar os algoritmos referidos nos pontos anteriores, vamos agora comparar o comportamento dos métodos.

Foram gerados 3 ficheiros distintos de números primos para realizar os testes, com as seguintes características:

- Ficheiro 1 - Onde n é o produto de números primos afastados, isto é, $n = p1 * p2$, com $|p1 - p2| > 100000$

- Ficheiro 2 - Onde n é o produto de números primos próximos, isto é, $n = p1 * p2$, com $|p1 - p2| < 1000$

- Ficheiro 3 - Onde n é o produto de números primos aleatórios, isto é, gerados sem restrições

Note-se que para se garantir que eram utilizados números grandes, foi definido que a lista só iria conter números de 29 bits, isto é, números maiores ou iguais a 268435455.

Para realizar os testes foi ulilizado uma máquina com as seguintes características:

- Sistema Operativo: Linux 2.6.31.14-0.6-desktop #1 SMP PREEMPT 2010-12-10 11:18:32 +0100 x86_64 x86_64 GNU/Linux

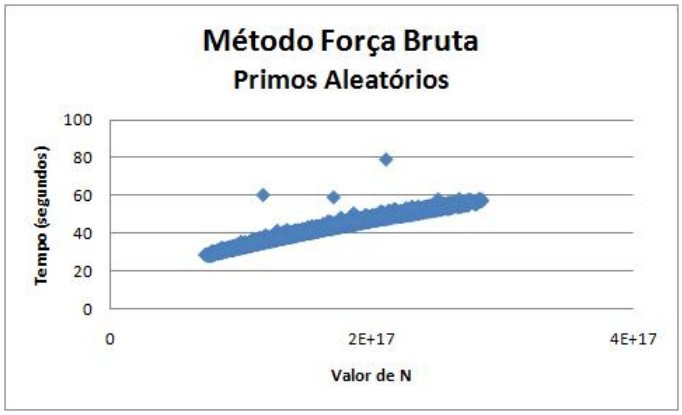
- CPU: bi-processador - Intel(R) Xeon(R) CPU, E5520, @ 2.27GHz (2 x processadores x 4 núcleos ("cores") x 2 linhas de dados ("threads") = 16 "processadores")

- RAM: 42 GB

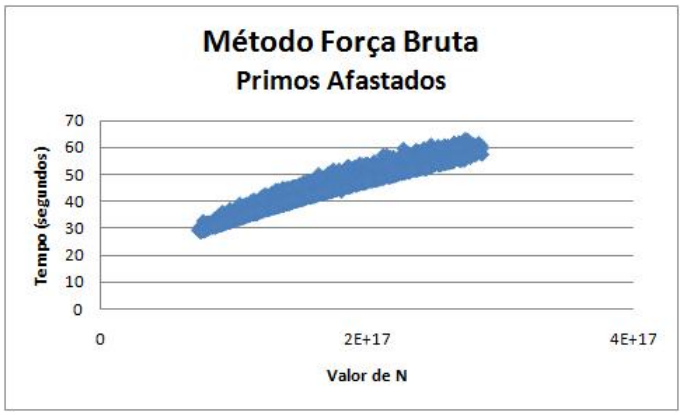
Vamos então analisar os resultados obtidos nos testes.

Analisando os resultados para o Método das Divisões, (também chamado Método da Força Bruta), notamos que, tendencialmente quanto maior é o valor de n maior é o tempo necessário para que o CPU consiga processar os dados.

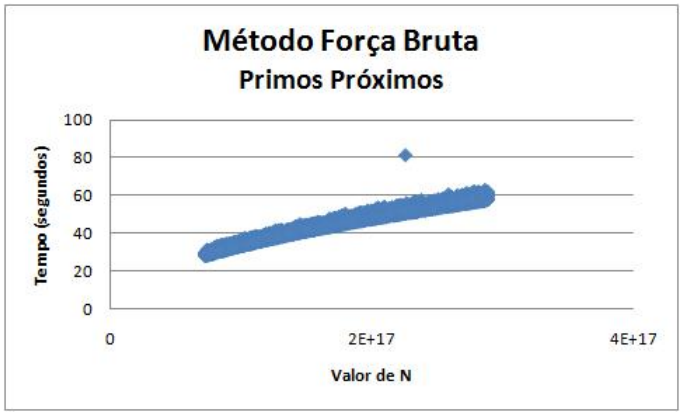
Notemos, no gráfico abaixo apresentado, que utilizando o ficheiro constítuido por números primos aleatórios, o tempo de processamento dos dados está, excluindo valores absurdos, entre 20 e 65 segundos.



Se em vez de números primos aleatórios, escolhermos números primos afastados, a velocidade de processamento aumenta significativamente, estando agora os valores entre os 25 e o 70 segundos.

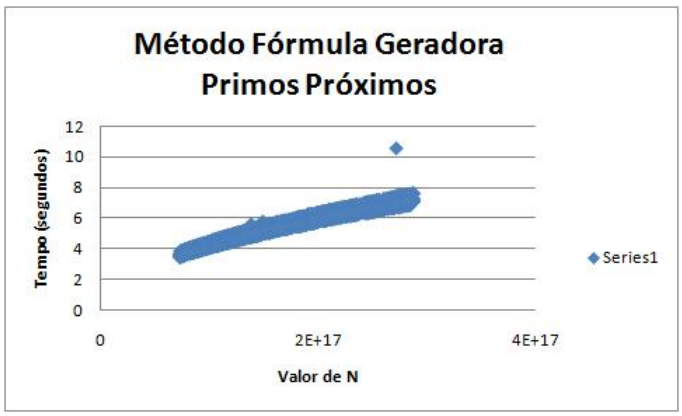


Ao utilizar o ficheiro de números primos próximos, não nos apercebemos de uma grande diferença nos resultados temporais relativamente ao ficheiro de números primos aleatórios, o que nos leva a crer que, desde que os primos não sejam muito afastados o tempo de execução não é muito alterado.



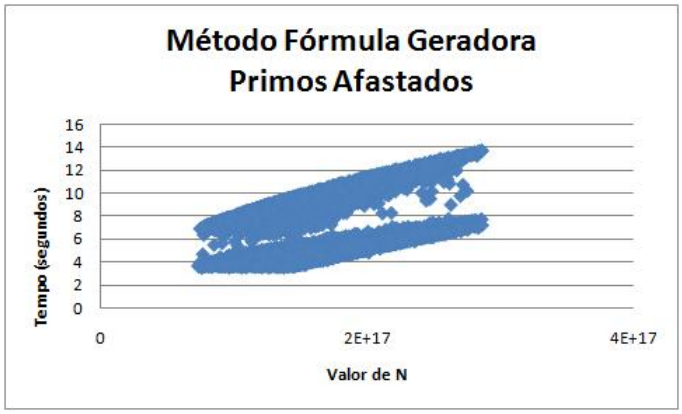
Tal como visto aquando do estudo teórico destes métodos, o método da fórmula geradora apresenta ganhos muito significativos relativamente ao método das divisões. Assim, observemos que, para primos aleatórios, os resultados temporais estão compreendidos entre os 3 e os 16 segundos, o que, comparando com o método das divisões é uma grande melhoria.

Se, em vez de factores primos aleatórios, utilizarmos factores primos próximos, os valores seguem uma tendência linear, com intervalo de resultados ainda inferior, na ordem dos 5 segundos.

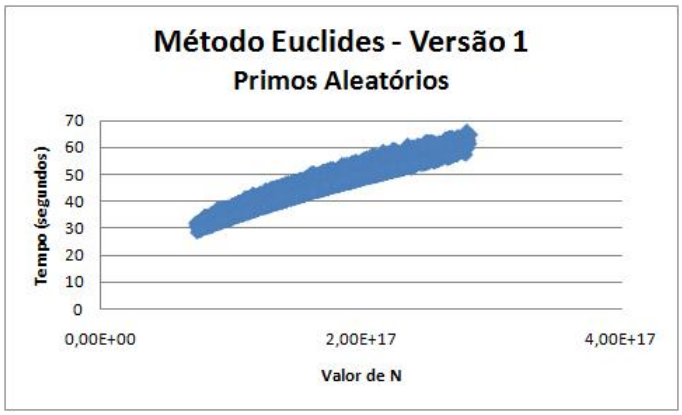


O comportamento, para o caso de factores primos afastados, é semelhante ao caso em que os primos são aleatórios, estando os valores, em média, na ordem dos

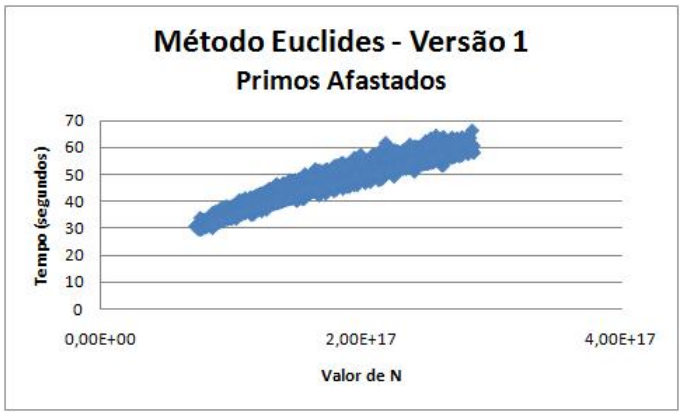
7,02 segundos.



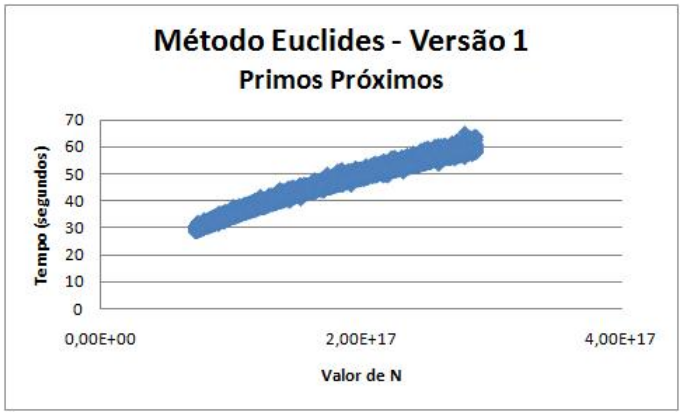
O Método de Euclides foi testado de duas formas diferentes. Nesta primeira versão, a lista de números primos é gerada a cada execução do algoritmo. Desta forma, o esforço exigido é elevado, o que resulta em intervalos de tempo na ordem dos gerados pelo Método das Divisões, isto é, entre 25 e 70 segundos.



Da mesma forma que para o Método das Divisões também neste método se verifica que o tempo de execução é maior quanto maior for o valor de n.

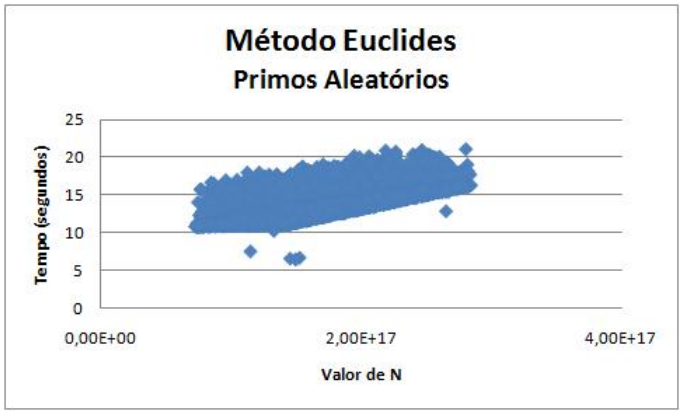


Não há diferenças signficativas nos resultados aquando da escolha de primos afastados, próximos ou aleatórios, sendo que o comportamento do algoritmo é semelhante para as três situações.



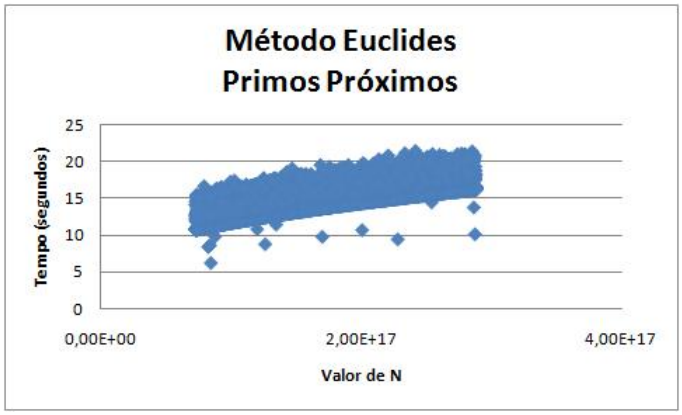
Vejamos agora os resultados para o mesmo algoritmo mas, em vez da lista de primos ser gerada a cada execução, esta será gerada exteriormente e só depois é realizada sua leitura.

Para números primos aleatórios confirma-se desde já o esperado: o esforço exigido é muito menor do que na versão anterior, isto é, o intervalo temporal decresce de forma significativa, passando de 25 a 70 segundos de 5 a 22 segundos. Desta forma o algoritmo torna-se bastante mais eficiente do que o anterior.



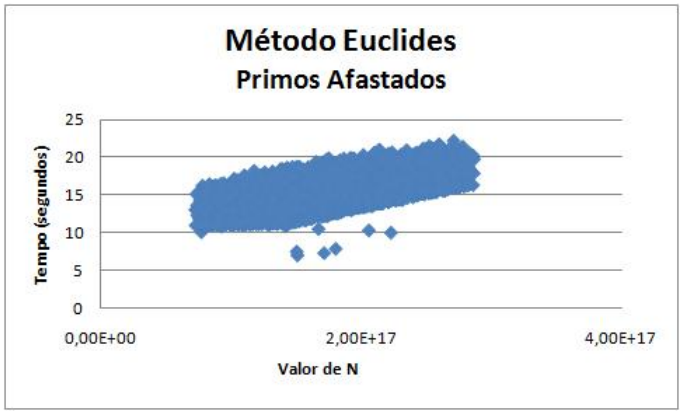
Neste caso a média de tempo que o CPU necessita para factorizar n é $13,52$ segundos.

Utilizando factores de n próximos, a média de tempo utilizado é $14,29$, como se pode comprovar no gráfico abaixo representado.

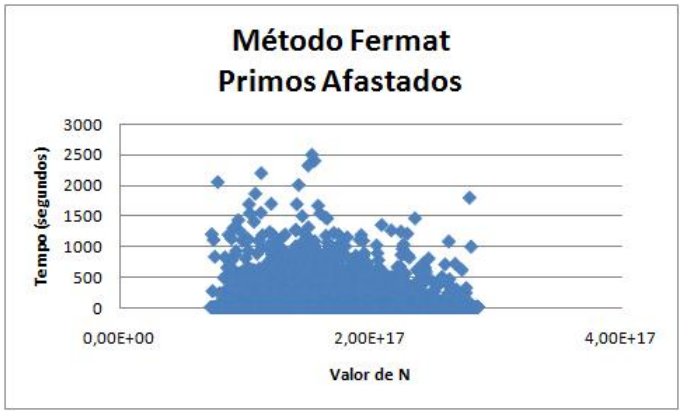


Utilizando factores primos de n afastados, a média dos tempos de execução aumenta para $14,73$ segundos.

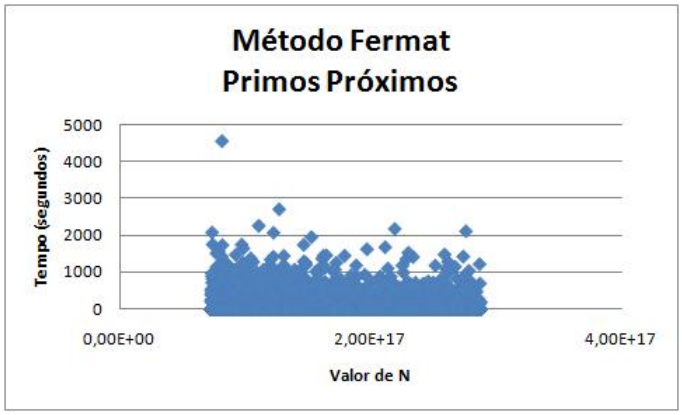
Podemos concluir desde já, que a utilização da segunda versão do método de Euclides (geração da lista de primos exteriormente) torna-se muito mais eficiente (em termos temporais) do que utilizando a primeira versão (geração da lista de primos a cada execução).



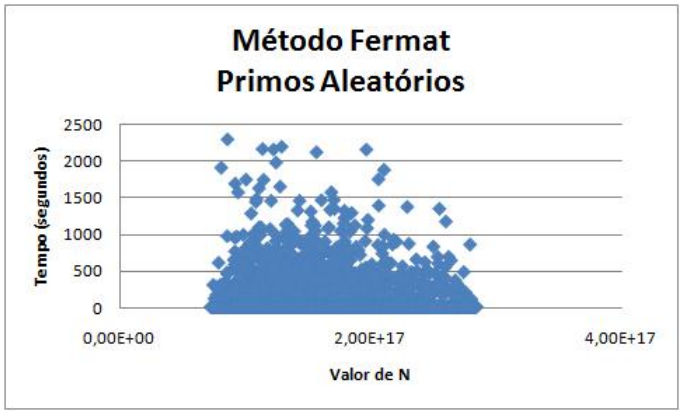
Analisando agora o Método de Fermat, vejamos que, utilizando, tal como no exemplo anterior, factores primos de n afastados o tempo de execução é bastante mais elevado que no Método de Euclides, sendo, neste caso, a média igual a 42,08 segundos.



Se considerarmos, para o mesmo método, factores primos próximos, a média dos tempos de execução ainda aumenta, sendo 46,47 segundos.



Já para o caso em que os valores utilizados são aleatórios, a média dos tempos de execução decresce significativamente, sendo 39,94 segundos.



Para terminar a comparação no desempenho destes métodos, é apresentada uma tabela (Tabela 3.1) onde estão compiladas as médias de cada um dos métodos estudados.

	Primos Afastados	Primos Aleatórios	Primos Próximos
Força Bruta	44,15	42,21	43,4
Fórmula Geradora	7,02	7,03	5,35
Euclides V1	45,19	44,84	44,24
Euclides V2	14,73	13,51	14,3
Fermat	42,08	39,94	46,48

Tabela 3.1: Tabela com Valores Médios (Tempos CPU)

Concluíndo:

1. Todos estes métodos apresentam tempos de execução cada vez mais elevados consoante o aumento dos valores dos números primos.

2. O método que apresenta menores tempos de execução é o método da fórmula geradora

3. No entanto, o método de Euclides (versão 1) é o que apresenta maiores tempos de execução, que se deve à geração a cada execução da lista de primos.

Para assegurar a segurança do RSA os factores primos têm de ser distantes um do outro e n tem de ter dimensão superior aos 20 dígitos.

Neste momento o RSA apresenta valores de n com 1024 ou mais dígitos.

Apesar de ser bastante difícil, é possível quebrar a cifra RSA. O método do Crivo Quadrático foi inventado com esse objectivo.

Queremos encontrar inteiros a e b tais que

$$a^2 \equiv b^2 (mod\ n)\ e\ a \neq \pm b (mod\ n).$$

Assim,

$$n \mid (a^2 - b^2) \Leftrightarrow n \mid (a+b)(a-b)$$

e os valores

$$d = mdc(a+b, n) \quad e \quad f = mdc(a-b, n)$$

poderão ser os factores não triviais de n.

Teremos então de estudar como determinar a e b.

O próximo capítulo consiste no estudo deste método, sua descrição matemática com a apresentação de um exemplo de utilização.

Capítulo 4

Crivo Quadrático

Tal como explicado anteriormente, o método do crivo quadrático tem como objectivo a factorização de números inteiros grandes.

Este método foi apresentado por Pomerance em 1982 [2], a partir dos estudos anteriores de Kraitchik e Dixon.

Até 1993, altura em que se desenvolveu o *Number Field Sieve*, este era o algoritmo de factorização mais rápido. Neste momento, o crivo quadrático é mais rápido só para números constituídos por 110 ou mais dígitos.

Sendo n o número a ser factorizado, o algoritmo do crivo quadrático pretende encontrar dois números x e y tais que [3]:

$$\begin{cases} x^2 \equiv y^2 (\text{mod } n) \\ x \not\equiv \pm y (\text{mod } n) \end{cases}$$

Assim,

$$\begin{cases} n \mid (x^2 - y^2) \\ n \not\mid (x - y) \wedge n \not\mid (x + y)) \end{cases}$$

O que implica que,

$$\begin{cases} n \mid (x + y)(x - y) \\ n \not\mid (x - y) \wedge n \not\mid (x + y)) \end{cases}$$

Logo,

$$\begin{cases} (x - y)(x + y) \equiv 0 (\text{mod } n) \\ n \not\mid (x - y) \wedge n \not\mid (x + y)) \end{cases}$$

Desta forma, $d = \text{mdc}(x + y, n)$ e $f = \text{mdc}(x - y, n)$ poderão ser os factores não triviais de n.

Assim, este método consiste em conseguir encontrar congruências da forma

$$x_i^2 \equiv y_i (mod\ n),$$

onde $\prod y_i = y^2$ é um quadrado perfeito.

4.1. Exemplo da Aplicação do Crivo Quadrático

Consideremos $n = 8051$. Com o auxilio do método do crivo quadrático, vamos factorizar n.

Seja $B = 105$, onde B é o limite para a construção da base de factores de n. Terá de ser constítuida por valores primos menores do que B. Assim, $2 < p < 105$.

Sabendo que p não divide n, consideremos os primos menores do que B tais que, pelo símbolo de Legendre:

$$\left(\frac{n}{p}\right) = 1,$$

isto é, n é resíduo quadrático módulo n.

Como

$$\left(\frac{n}{p}\right) \equiv n^{\frac{p-1}{2}} \ mod\ p,$$

vamos testar todos os números primos, para assim construir a base.

$$p = 3 \Rightarrow \left(\frac{8051}{3}\right) \equiv 8051^1 \ mod\ 3 \equiv 2 \ mod\ 3$$

Logo 3 não pertence à base.

$$p = 5 \Rightarrow \left(\frac{8051}{5}\right) \equiv 8051^2 \ mod\ 5 \equiv 1 \ mod\ 5$$

Logo 5 pertence à base.

$$p = 7 \Rightarrow \left(\frac{8051}{7}\right) \equiv 8051^3 \ mod\ 7 \equiv 1 \ mod\ 7$$

Logo 7 pertence à base.

$$p = 11 \Rightarrow \left(\frac{8051}{11}\right) \equiv 8051^5 \ mod\ 11 \equiv 10 \ mod\ 11$$

Logo 11 não pertence à base.

$$p = 13 \Rightarrow \left(\frac{8051}{13}\right) \equiv 8051^6 \ mod\ 13 \equiv 1 \ mod\ 13$$

Logo 13 pertence à base.

Procedendo de igual forma para os restantes números primos até 105, encontramos a base desejada:

$$5 \quad 7 \quad 13 \quad 23 \quad 43 \quad 47 \quad 59 \quad 61 \quad 79 \quad 103$$

Para calcular os valores de $f(r)$, onde $f(r) = r^2 - n$ comecemos por calcular o valor de $k = \lfloor \sqrt{n} \rfloor$, onde $r = k + 1, k + 2, ...$

Suponhamos que p não divide n mas divide $f(r)$. Assim, $n \equiv r^2 \ mod \ p$.

Com $n = 8051$, temos $k = \lfloor \sqrt{8051} \rfloor = 89$. Considerando 200 valores de r, temos $r = 89 + i, \ i = 1, 2, 3, ..., 200$.

Calculemos então os valores de $f(r)$.

$$r = 90 \Rightarrow f(90) = 90^2 - 8051 = 49 = 7^2$$

Logo pode ser utilizado como valor de $f(r)$.

$$r = 91 \Rightarrow f(91) = 91^2 - 8051 = 230 = 2.5.23$$

Como 2 não pertence à base, não pode ser utilizado como valor de $f(r)$.

$$r = 92 \Rightarrow f(92) = 92^2 - 8051 = 413 = 7.59$$

Logo pode ser utilizado como valor de $f(r)$.

$$r = 93 \Rightarrow f(93) = 93^2 - 8051 = 598 = 2.13.23$$

Logo não pode ser utilizado como valor de $f(r)$.

Repetindo o processo para todos os valores de r, encontramos os seguintes valores de $f(r)$ válidos:

$$f(90) = 49 = 7^2$$

$$f(92) = 413 = 7.59 = 92^2 - 8051$$

$$f(104) = 2765 = 5.7.79 = 104^2 - 8051$$

$$f(106) = 3185 = 5.7^2.13 = 106^2 - 8051$$

$$f(114) = 4945 = 5.23.43 = 114^2 - 8051$$

$$f(116) = 5405 = 5.23.47 = 116^2 - 8051$$

$$f(132) = 9373 = 7.13.103 = 132^2 - 8051$$

$$f(144) = 12685 = 5.43.59 = 144^2 - 8051$$

$$f(206) = 34385 = 5.13.23^2 = 206^2 - 8051$$

$$f(210) = 36049 = 13.47.59 = 210^2 - 8051$$

$$f(262) = 60593 = 13.59.79 = 262^2 - 8051$$

$$f(286) = 73745 = 5.7^3.43 = 286^2 - 8051$$

Temos agora os dados necessários para podermos construir a matriz constítuida por:

1. número de linhas é o número de $f(r)'s$ completamente factorizados
2. número de colunas é o número de primos da base de factores acima construida

A matriz será construída tendo em conta o seguinte critério: se o primo correspondente têm potência par, então é representado pelo digíto 0, se a potência for ímpar, o dígito é 1. Assim, vemos já que a linha para $f(90)$ ficará toda a zeros, sendo assim desprezada. Desta forma, vamos construir uma matriz $11 * 10$, pois temos 11 valores de $f(r)$ e 10 valores primos na nossa base de factorização.

Esta matriz vai-nos permitir, através da Eliminação de Gauss, encontrar uma combinação de $f(r)'s$, que seja um quadrado perfeito.

$$B = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Para encontrarmos então o quadrado perfeito que procuramos, vamos então utilizar o método de Eliminação de Gauss.

Para tal, necessitamos do auxílio de uma matriz identidade 11×11, que, em conjunto com a matriz B nos dará o quadrado perfeito. Vamos então proceder à

eliminação de Gauss nas duas matrizes, até que seja encontrada uma linha com todos os elementos a zero.

A linha, na matriz I, associada à linha a zeros encontrada dar-nos-á a informação desejada, os seja, indicar-nos-á quais os $f(r)'s$ que teremos de multiplicar para que seja encontrado um quadrado perfeito.

Caso não se consiga encontrar um factor não trivial de n, deverá ser escolhida uma nova base e recomeçar-se todo o processo novamente.

Neste caso, notemos que a primeira linha a ficar a zeros, na matriz B, será a $8^{\underline{a}}$ linha (ao fazer $L8 \rightarrow L8 + L3$), e que, a linha correspondente na matriz identidade é:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Tal como acima explicado, as colunas com dígito 1 indicam quais os valores de $f(r)'s$ que devem ser multiplicados.

Assim, neste caso teremos de multiplicar o $3^{\underline{o}}$ e o $8^{\underline{o}}$:

$$f(106) = 3185 = 5.7^2.13 = 106^2 - 8051$$

$$f(206) = 34385 = 5.13.23^2 = 206^2 - 8051$$

Deste modo, x é dado pelo produto dos $r's$ correspondentes módulo n e y é dado pela raíz do produto dos factores dos $f(r)'s$.

Assim,

$$x = 106.206 \ mod \ 8051 \ \equiv 5734 \ mod \ 8051$$

$$y = \sqrt{(5^2.7^2.13^2.23^2)} \ mod \ 8051 \ \equiv 10465 \ mod \ 8051 \equiv 2414 \ mod \ 8051$$

Logo,

$$x - y = 5734 - 2414 = 3320$$

Desta forma,

$$mdc(3320, 8051) = 83$$

é um factor de n.

De outra forma,

$$x + y = 8148 = mdc(8148, 8051) = 97$$

também é factor de n.

Assim, conseguimos factorizar o número 8051 em números primos, com $8051 = 83 \times 97$.

4.2. Conclusões

Com este exemplo conseguimos demonstrar que o funcionamento do método do crivo quadrático não é trivial. Pensemos somente que se a base encontrada não servir para se aplicar o método de eliminação de Gauss para encontrar um quadrado perfeito, terá de ser iniciado novamente o processo para se encontrar uma nova base.

No entanto, este exemplo serviu também para provar que, com o método do crivo quadrático conseguimos factorizar números inteiros grandes, e, desta forma, o método RSA pode ser quebrado.

No entanto, o problema da factorização ainda está longe de ser resolvido de forma rápida, o que garante a segurança nas comunicações com a cifra RSA, desde que a dimensão dos primos usados no cálculo da chaves seja muito grande. Estes devem ter uma dimensão igual ou superior a 1024 bits.

Apêndice A

Resultados de Teoria dos Números

Tal como definido em [8] um número Primo é um número inteiro $p > 1$ cujos únicos divisores inteiros são os números 1 e p. Por definição, o número 1 não é considerado primo nem composto. Se um número inteiro $a > 1$ não for primo diz-se *composto*. Euclides provou que existe uma infinidade de números primos.

Teorema 1 *: Qualquer número natural $a > 1$ é um produto de números primos.*

Demonstração: Seja $a \in \mathbb{N}, a > 1$.

Se a for primo, não há nada a provar pois temos o produto de um só factor.

Suponhamos que a é composto.

Então a tem divisores entre 1 e a. Se m for o menor desses divisores, m é de certeza primo (porque, se não, teria divisores menores do que m que seriam também divisores de a).

Designemos m por p_1. Então temos:

$$a = p_1 a_1,$$

com p_1 primo e $1 < a_1 < a$. Se a_1 for primo, já chegámos à conclusão desejada. Se a_1 for composto, repetindo o raciocínio anterior concluímos que a_1 tem um divisor primo p_2 satisfazendo $1 < p_2 < a_1$, donde

$$a = p_1 p_2 a_2,$$

com p_1 e p_2 primos e $1 < a_2 < a_1 < a$.

Prosseguindo desta forma, obtemos números naturais $a > a_1 > a_2 > ....$

Como uma sucessão de números naturais não pode decrescer indefinidamente, há-de haver um momento em que um destes números é primo, digamos p_s, pelo que

$$a = p_1 p_2 ... p_s$$

□

Lema 1 *: Se um número primo dividir um produto de números inteiros, tem de dividir pelo menos um dos factores.*

Demonstração:

Seja p um número primo. Vamos provar que, sendo n um natural qualquer ≥ 2, se p dividir um produto de n números inteiros, então tem que dividir pelo menos um dos factores.

Vamos provar por indução.

Para $n = 2$, sejam a_1, a_2 dois inteiros quaisquer e suponhamos que $p|a_1 a_2$. Se p dividir a_1, está provado.

Se p não dividir a_1, então p e a_1 são primos entre si, porque p não tem outros divisores positivos senão 1 e ele próprio.

Então, de certeza que $p|a_2$.

Suponhamos agora que a afirmação é verdadeira para produtos de k factores e sejam $a_1, a_2, ..., a_{k+1}$ inteiros quaisquer tais que $p|a_1 a_2 ... a_{k+1}$.

Se p dividir a_{k+1} então não há mais nada a provar.

Se p não dividir a_{k+1}, então, de forma análoga, p tem que dividir o produto $a_1 a_2 ... a_k$ e portanto, pela hipótese de indução, tem que dividir um dos inteiros $a_1, a_2, ..., a_k$. □

Teorema 2 (Teorema Fundamental da Aritmética) *Qualquer número natural $a > 1$ escreve-se de forma única como um produto de números primos.*

Demonstração: : Seja $a > 1$ qualquer. Já sabemos que a se escreve como um produto de números primos. Suponhamos que era possível escrever a como produto de números primos de duas maneiras diferentes:

$$a = p_1 p_2 ... p_s = q_1 q_2 ... q_t$$

factorizações em que podemos supor já retirados os factores comuns, de modo que não haja nenhum primo que figure em ambos os membros.

Como p_1 divide o primeiro membro, divide também o segundo, isto é, $p|q_1 q_2 ... q_t$.

Pelo lema anterior, segue-se que p_1 tem que dividir um dos factores do segundo membro, digamos $p|q_j$.

Como ambos os números são primos, isto só pode acontecer se $p_1 = q_j$, o que contradiz o facto de não haver primos comuns nas duas factorizações. □

Teorema 3 *Existe uma infinidade de números primos.*

Demonstração: Suponhamos que o número de primos é finito, digamos $p_1, p_2, ..., p_r$. Seja n o número natural $p_1 p_2 ... p_r + 1$.

Como n é maior do que todos os primos $p_1, p_2, ..., p_r$, tem de ser divisível pelo menos por um deles, pois n é um produto de primos.

Suponhamos que n é divisível por p_1. Então $n = qp_1$ para certo inteiro q. Assim,

$$p_1(q - p_2 ... q_r) = 1$$

e p_1 dividiria 1, o que é impossível. □

Definição A.1 (Resíduo Quadrático) *Considere-se o conjunto Z_p^*, onde p é um número primo maior do que 2 e $a \in Z_p^*$. Dizemos que a é um* resíduo quadrático *módulo p se*

$$b^2 \equiv a(mod\ p)\ para\ algum\ b \in Z_p^*$$

Lema 2 *Seja $p > 2$ um número primo. Exactamente metade dos elementos de Z_p^* são Resíduos Quadráticos.*

Definição A.2 (Símbolo de Legendre) *Seja p um número inteiro primo ímpar e a um número inteiro. Definimos o* símbolo de Legendre *por*

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & se\ p \mid a \\ 1 & se\ a\ é\ resíduo\ quadrático \bmod p \\ -1 & caso\ contrário \end{cases}$$

Apêndice B

Problemas Computacionais Difíceis

Chamamos *problemas computacionais difíceis* [8] aos problemas que só são resolvidos computacionalmente envolvendo enorme tempo computacional dispendido e/ou enorme volume de memória computacional para a sua resolução.

Exemplos:

1. RSA - dado $n \in \mathbb{N}, n = pq$, com p e q primos, factorizar n nos seus factores primos.

2. El Gamal - Dado um grupo cíclico finito G, um gerador $g \in G$ e um elemento $x \in G$, determinar o valor $i \in \mathbb{Z}_{\mathbb{G}}$ tal que $x = g^i$.

Para todos estes métodos é possível construir um sistema criptográfico em que a cifração seja efectuada de forma eficiente, no entanto a sua quebra é um problema computacional difícil.

Bibliografia

[1] S. Coutinho. *Números inteiros e Criptografia RSA*. IMPA-SBM, 2000.

[2] Richard Crandall and Carl Pomerance. *Prime Numbers: A computacional Perspective*. Springer, 2ed edition, 2005.

[3] Adriana Betânia de Paula Molgora and Aline de Paula Sanches. Um estudo do método de fatoração de inteiros crivo quadrático. *Universidade Estadual de Mato Grosso do Sul, Ciência da Computação*.

[4] Carl Pomerance. *The Quadratic Sieve Factoring Algorithm*. Springer-Verlag, 1985.

[5] Pedro Quaresma and Cristina Caldeira. *Apontamentos de Apoio da Disciplina de Códigos e Criptografia*. Departamento de Matemática, Universidade de Coimbra, 2010/2011.

[6] Pedro Quaresma and Elsa Lopes. Criptografia. *Gazeta de Matemática*, 154:7 – 11, 2008.

[7] Pedro Quaresma and Augusto Pinho. Criptoanálise. *Gazeta de Matemática*, 157:22–31, 2009.

[8] João Filipe Queiró. *Teoria dos Números*. Departamento de Matemática, Universidade de Coimbra, 2002/2003.