

Números Primos e a Criptografia RSA: um Possível Ataque a sua Segurança Prime Numbers and RSA Encryption: A Possible Attack on your Security

José Robyson Aggio Molinari ¹ e Franciéle Maria de Souza Retslaff ²

¹ Departamento de Matemática – Universidade Estadual do Centro-Oeste (UNICENTRO/IRATI).

² Doutoranda do Programa de Pós-graduação em Ciências Florestais – Universidade Estadual do Centro-Oeste (UNICENTRO/IRATI).

e-mail: ¹ aggio13@hotmail.com e ² francieleretslaff@yahoo.com.br

Resumo

O objetivo deste trabalho foi o estudo da Criptografia RSA em que foi desenvolvido um algoritmo para analisar um possível ataque a sua segurança por meio da função de Euler, para tentar encontrar os fatores primos de um número composto com menos iterações que o algoritmo de Fermat. A importância desta fatoração está presente na codificação e decodificação das mensagens no método de Criptografia RSA. Para o entendimento do método RSA, foi realizado a descrição matemática, os processos de codificação e decodificação, um caso particular para o método e uma análise do porquê o método ainda ser seguro. A busca da fatoração pela função de Euler foi quatro vezes mais rápida em relação a fatoração pelo método de Fermat, porém são necessários alguns estudos para tentar delimitar melhor o intervalo de busca na função proposta.

Palavras-chave: Números Primos; Criptografia; Fatoração.

Abstract

The objective of this work was the study of RSA Encryption in which an algorithm was developed to analyze a possible security attack through the Euler function to try to find the prime factors of a compound number with fewer iterations than the Fermat algorithm. The importance of this factorization is present in the encoding and decoding of messages in the RSA Encryption method. For the understanding of the RSA method, the mathematical description, coding and decoding processes were performed, a particular case for the method and an analysis of why the method is still safe. The search for factorization by the Euler function was four times faster in relation to factorization by the Fermat method, but some studies are needed to try to delimit the search interval in the proposed function better.

Keywords: Prime Numbers; Cryptography; Factorization.

Introdução

O nome Criptografia, em grego, cryptos significa secreto, oculto e grafia significa escrita. A criptografia estuda os métodos para codificar uma mensagem de modo que só seu destinatário legítimo consiga interpretá-la, ou seja, através dela é possível o envio de mensagens de uma forma segura, mesmo uma terceira pessoa tendo interceptado, não conseguirá ler a informação presente na mensagem.

A Criptografia é um assunto que sofreu muitas mudanças durante a civilização humana, os homens inventavam códigos secretos na tentativa de transmitir mensagens inteligíveis por um interceptador

Um dos primeiros fatos presentes no que concerne a Criptografia, foi utilizado por Júlio César para comunicar-se com as legiões em combate pela Europa. A criptografia era feita pela substituição de uma letra pela seguinte, isto é, trasladava uma casa para diante.

Todo código necessariamente utiliza-se de duas propriedades, uma para codificar e outra para decodificar. Decodificar é o que o destinatário faz quando recebe a mensagem e possui a chave de decodificação. Decifrar é tentar ler a mensagem sem saber a chave de decodificação.

A criptografia utilizada por Júlio César foi muito fácil de ser decifrada, ou qualquer outra forma de criptografia utilizando apenas substituição de letras por outras, isto se deve ao fato de que a frequência média com que cada letra é usada em uma língua é mais ou menos constante.

Uma maneira para contornar esse problema foi dividir a mensagem em grupos de letras e criptografar a mensagem em grupo, criando assim um sistema poligráfico, onde está presente as Cifras de Hill [1].

O procedimento com as Cifras de Hill para criptografar uma mensagem era simples, bastava apenas saber as operações de Matrizes (Multiplicação e o cálculo da Inversa). O remetente e o destinatário sabiam a chave (matriz), que codificava a mensagem. Com isso, ao receber a mensagem conseguiam lê-la, caso uma terceira pessoa tivesse acesso a essa mensagem era preciso saber a chave que a codificou. Mas ainda essa forma de criptografar não era totalmente segura, pois se fosse descoberto o significado de uma coluna da matriz as demais poderiam ser descobertas.

Segundo o autor [2], as matrizes surgiram na escola inglesa Trinity College, em um artigo do Matemático Arthur Cayley (1821-1895).

Com o avanço dos meios de comunicação e tecnológicos, tornou-se necessário o desenvolvimento de métodos seguros de transmissão de informações, ou seja, métodos de codificação de mensagens. Com isso surgiu a criptografia de chave pública, também conhecida por criptografia assimétrica. Esse método possui duas chaves distintas que são utilizadas. Uma delas a chave pública, que está disponível para qualquer pessoa, ou seja, é de conhecimento de todos e é utilizada para codificar as mensagens, que só poderão ser decodificadas por quem possuir a chave privada correspondente.

A criptografia foi se aperfeiçoando durante os anos e em 1977 foi inventado o método RSA (RIVEST, R.L., SHAMIR, A. E ADLEMAN, L.M., 1978) [3]. Há vários outros métodos de chave pública, mas o RSA é atualmente, o mais usado em aplicações comerciais [4].

O método RSA é muito seguro, utilizado no sistema bancário para garantir a segurança em transações financeiras pela internet. O método de codificação de uma mensagem é muito simples, porém o processo inverso, o de decodificação é impossível de ser resolvido se a mensagem for

interceptada por um hacker, mesmo utilizando-se da computação algébrica ou da programação computacional. Este fato de decifração da mensagem é um problema em aberto na Matemática até o presente momento, pois a única pessoa que consegue decodificar é o destinatário da mensagem.

Segundo o autor [4]:

“para implementar o RSA precisamos de dois parâmetros básicos: dois números primos que vamos chamar de p e q . Para codificar uma mensagem usando o RSA é suficiente conhecer o produto dos dois primos, que vamos chamar de n . A chave de codificação do RSA é portanto constituída essencialmente pelo número $n=pq$. Cada usuário do método tem sua própria chave de codificação. Esta chave é tornada pública: todos ficam sabendo que, para mandar uma mensagem para o banco Acme, deve ser usada a chave n . Por isso n também é conhecido como “chave pública”. Já a chave de decodificação é constituída pelos primos p e q . Cada usuário tem que manter sua chave de decodificação secreta ou a segurança do método estará comprometida”.

Aparentemente fatorar o número n parece ser um processo teoricamente fácil, mas usando como chaves de codificação RSA números muito grandes (de 200 algarismos ou mais), levaria milhares de anos. De acordo com [4], é disto que depende a segurança do RSA, da ineficiência dos métodos de fatoração atualmente conhecidos.

Na literatura pode-se encontrar vários trabalhos sobre o estudo de criptografia, em relação a criptografia RSA, pelo fato da fatoração de $n = pq$ ser um problema em aberto na Matemática, há diversos pesquisadores que comentam sobre o assunto, entre eles [4] e [5].

Diante de tal importância na sociedade, esta pesquisa justificou o seu estudo no entendimento do funcionamento do método RSA, abordando alguns tópicos de teoria dos números: o Pequeno Teorema de Fermat; os algoritmos de fatoração; ineficiência dos algoritmos; a fatoração por Fermat; a descrição matemática do Método RSA; o porquê do Método ser seguro; além de propor uma alternativa para fatorar o valor de n de uma maneira otimizada por meio da função de Euler, fazendo um comparativo com o método de Fermat.

Fundamentação Teórica

O Pequeno Teorema de Fermat afirma que se p é um número primo e a um inteiro qualquer então p divide $a^p - a$. Casos particulares desse teorema já eram conhecidos desde a antiguidade. Segundo [4], os chineses sabiam que se p é primo então p divide $2^p - 2$, mas foi Fermat quem obteve o resultado geral e o introduziu na Matemática Européia do século XVII utilizando a linguagem de congruências.

Teorema de Fermat I. Seja p um número primo e a um inteiro, então $a^p \equiv a \pmod{p}$.

Demonstração: A prova deste teorema será feita por indução finita. Para isto precisa-se encontrar uma proposição $p(n)$ para aplicar a indução. $p(n): n^p \equiv n \pmod{p}$.

É evidente que $p(1)$ é válido, pois $1^p = 1$. Suponha então que $n^p \equiv n \pmod{p}$. A passagem de $p(n)$ para $p(n+1)$ é pelo Binômio de Newton. Para essa passagem utiliza-se o seguinte Lema 1.

Lema 1. Seja p um número primo e a e b inteiros. Então, $(a+b)^p \equiv a^p + b^p \pmod{p}$.

Demonstração do Lema 1: Utilizando a expressão usual do Binômio de Newton, tem-se:

$$(a+b)^p = a^p + b^p + \sum_{i=1}^{p-1} \binom{p}{i} a^{p-i} b^i. \text{ Para obter o lema é suficiente mostrar que o termo } \sum_{i=1}^{p-1} \binom{p}{i} a^{p-i} b^i$$

é congruente a zero módulo p . Considere o número binomial $\binom{p}{i} = \frac{p(p-1)\dots(p-i+1)}{i!}$. Para que a

fração corresponda a um número inteiro é preciso que o denominador seja completamente simplificado por termos no numerador. Suponha que $1 \leq i \leq p-1$, então o denominador $i!$ não tem p como um de seus fatores primos. Assim o fator p que aparece no numerador não é cancelado por nenhum fator do

denominador. Portanto o número inteiro $\binom{p}{i}$ é múltiplo de p quando $1 \leq i \leq p-1$,

consequentemente $\sum_{i=1}^{p-1} \binom{p}{i} a^{p-i} b^i \equiv 0 \pmod{p}$.

Voltando à demonstração do Teorema de Fermat, supondo que $n^p \equiv n \pmod{p}$ e, deseja-se mostrar que $(n+1)^p \equiv n+1 \pmod{p}$. Utilizando o Lema 1, tem-se que:

$(n+1)^p \equiv n^p + 1^p \equiv n^p + 1 \pmod{p}$. Como a hipótese de indução é $n^p \equiv n \pmod{p}$, então, $(n+1)^p \equiv n^p + 1 \equiv n+1 \pmod{p}$. Com isso, prova-se o enunciado do teorema para os números naturais, mas este foi enunciado para qualquer inteiro. Então, chamando de a um inteiro negativo, $-a$ é positivo, logo aplicando o teorema já provado, para $-a$. Tem-se:

$$(-a)^p \equiv -a \pmod{p}. \quad (1)$$

Supondo que p é ímpar, $(-a)^p = -a^p$. Substituindo em (1), $-a^p \equiv -a \pmod{p}$, e multiplicando por -1 , conclui-se que $a^p \equiv a \pmod{p}$, restando apenas o caso em que $p=2$. Se $p=2$ então $(-a)^p = a^p$. O que ocorre no teorema $a^p \equiv a \pmod{p}$.

Será utilizado a ideia do Pequeno Teorema de Fermat, no método RSA para justificar uma passagem Matemática na fórmula, mas sendo este o enunciado a seguir.

Teorema de Fermat II. Seja p um número primo e a um inteiro que não é divisível por p . Então $a^{p-1} \equiv 1 \pmod{p}$.

Demonstração: Segundo o Teorema de Fermat I, se p é primo e a é um número inteiro qualquer, então $a^p \equiv a \pmod{p}$. Suponha que p não divide a . Neste caso a é invertível módulo p , de acordo com o teorema de inversão. Seja a' um inteiro positivo tal que $aa' \equiv 1 \pmod{p}$. Multiplicando ambos os membros de $a^p \equiv a \pmod{p}$ por a' , obtém-se: $a' a^p \equiv a' a \pmod{p}$. Com a substituição de $aa' \equiv 1 \pmod{p}$ nesta equação, tem-se $a^{p-1} \equiv 1 \pmod{p}$, o que conclui a demonstração.

Algoritmos de fatoração

O processo de encontrar os fatores primos de um número composto denomina-se fatoração. Existem diversos algoritmos de fatoração, mas não existe um algoritmo que funcione perfeitamente, em que o computador possa executar em tempo polinomial para todos os números inteiros. Nesta pesquisa foram abordados alguns algoritmos de fatoração.

Considere o seguinte Problema: tendo por entrada o valor $n \in \mathbb{N}$, determine seus fatores primos e respectivos expoentes.

Com foco apenas no primeiro fator de um inteiro dado. Tendo n como entrada, tente dividir n por cada um dos inteiros de 2 a $n - 1$, caso algum desses inteiros dividir n , então encontra-se um fator de n , onde f é o menor fator e este fator é um número primo.

Sabe-se que um número inteiro não pode ter um fator maior que ele próprio e também pode-se restringir a busca em um intervalo menor que 2 a $n - 1$, sendo este intervalo de 2 a $\sqrt{n}$. Porém se n é primo o único fator será o próprio n . É necessário verificar, entretanto, que se n é composto, seu menor fator é no máximo $\sqrt{n}$.

Assim, se n é um número composto e se $f > 1$ é seu menor fator, existe um inteiro positivo a tal que $n = f.a$. Como f é o menor fator, certamente $f \leq a$. Mas $a = \frac{n}{f}$, logo $f \leq \frac{n}{f}$. Disso decorre que $f^2 \leq n$, que é equivalente a $f \leq \sqrt{n}$.

O procedimento descrito é representado pelo Algoritmo a seguir.

Algoritmo da fatoração (menor fator)

Entrada: Digite um inteiro positivo n .

Saída: Inteiro positivo f que é o menor fator primo de n ou a indicação que n é primo.

Etapas: Comece fazendo $f = 2$.

Etapa 2: Se $\frac{n}{f}$ é inteiro escreva 'f é fator de n' e pare; senão vá para a Etapa 3.

Etapa 3: incremente a f uma unidade e vá para a Etapa 4.

Etapa 4: Se $f > \sqrt{n}$, escreva n é primo e pare. Caso contrário, retorne à Etapa 2.

Logo, dado um inteiro $n > 0$, pode-se determinar se n é primo ou composto. Se n é primo encontra-se a sua fatoração, mas se n for composto, pode-se encontrar todos os seus fatores primos e suas respectivas multiplicidades aplicando o algoritmo da fatoração várias vezes, ou seja, aplicando o algoritmo a n encontra-se o fator q_1 . Então q_1 é o menor fator primo de n. Aplicando o algoritmo da fatoração ao co-fator $\frac{n}{q_1}$, determina-se um segundo fator q_2 . Pode-se repetir esse procedimento aplicando ao co-fator $\frac{n}{q_1 q_2}$, e assim por diante. Dessa forma, determina-se uma sequência crescente de números primos $q_1 \leq q_2 \leq \dots \leq q_s$, em que cada um é um fator de n.

O procedimento descrito é representado pelo Algoritmo a seguir.

Algoritmo dos fatores (todos os fatores)

Entrada: Digite um inteiro positivo n.

Saída: $q_1, q_2, \dots, q_s$, são os fatores primos de n, ou indicativo de que n é primo.

Etapa1: Comece fazendo $f = 2$.

Etapa 2: Se $\frac{n}{f}$ é inteiro armazenar $q_i = f$ e $n = \frac{n}{q_i}$. Vá para a Etapa 3.

Etapa 3: Se for verdade a etapa 2, então efetuar o cálculo para o novo valor em n e para o mesmo q_i para $i = 1, 2, \dots, k$. Enquanto houver o mesmo fator repetir a Etapa 2. Senão vá para a Etapa 4.

Etapa 4: incremente a f uma unidade e vá para a Etapa 5.

Etapa 5: Se $f > \sqrt{n}$, então escreva os fatores $q_1, \dots, q_s$ ou n é primo e pare. Senão volte a Etapa 2.

Fatoração por Fermat

A fatoração por Fermat é muito eficiente quando n tem um fator primo próximo de $\sqrt{n}$. Supõe-se n ímpar, pois se n for par então 2 é um de seus fatores. Fermat teve a brilhante ideia de tentar encontrar inteiros positivos x e y tais que $n = x^2 - y^2$. Se encontrados esses números $n = x^2 - y^2 = (x - y)(x + y)$ e por consequência, $x - y$ e $x + y$ são fatores de n .

Para implementar o algoritmo de Fermat primeiro é preciso determinar a parte inteira de $\sqrt{n}$. Se n é um quadrado perfeito então $f = \sqrt{n}$, será o próprio fator. Pela notação acima tem-se: $x = f$ e $y = 0$. Para $y > 0$, então $x = \sqrt{n + y^2} > \sqrt{n}$. Logo pode-se elaborar o seguinte algoritmo.

Algoritmo de Fermat

Entrada: Inteiro positivo ímpar n .

Saída: Um fator de n ou uma mensagem que n é primo.

Etapa 1: Inicie $x = [\sqrt{n}]$; se $n = x^2$, então x é fator de n e pode parar. Senão vá para a Etapa 2.

Etapa 2: Incremente x de uma unidade e calcule $y = \sqrt{x^2 - n}$.

Etapa 3: Repita a Etapa 2 até encontrar um valor inteiro para y (1º caso), ou até que x seja igual a $\frac{n+1}{2}$ (2º caso): No 1º caso n tem fatores $x - y$ e $x + y$, no 2º caso n é primo.

Demonstração do Algoritmo de Fermat

É necessário considerar separadamente o que ocorre quando n é composto e quando n é primo. No caso de n ser composto, é necessário mostrar que existe um inteiro $x > [\sqrt{n}]$, em que os colchetes representa a parte inteira da raiz quadrada, tal que $\sqrt{x^2 - n}$ é um inteiro menor que $\frac{n+1}{2}$. Isto significa que se n é composto então o algoritmo irá parar antes de chegar em $\frac{n+1}{2}$. Se n é primo, então é necessário mostrar que o único valor possível para x é $\frac{n+1}{2}$.

Suponha que n pode ser fatorado na forma $n = pq$, em que $p \leq q$. Deseja-se obter inteiros positivos x e y tais que $n = x^2 - y^2$, ou seja, $n = pq = (x - y)(x + y) = x^2 - y^2$.

Como $x - y \leq x + y$, isto sugere que $p = x - y$ e $q = x + y$. Desse sistema, obtém-se:

$x = \frac{p+q}{2}$ e $y = \frac{q-p}{2}$, e portanto:

$$\left(\frac{p+q}{2}\right)^2 - \left(\frac{q-p}{2}\right)^2 = \frac{p^2 + 2pq + q^2 - q^2 + 2pq - p^2}{4} = pq = n. \quad (2)$$

Entretanto, x e y devem ser números inteiros e por hipótese n é ímpar então $x = \frac{p+q}{2}$ e $y = \frac{q-p}{2}$, logo p e q , que são fatores de n , têm que ser ímpares. Com isso, $p+q$ e $q-p$ são pares e consequentemente, $\frac{p+q}{2}$ e $\frac{q-p}{2}$ são inteiros. Agora se n é primo então $p=1$ e $q=n$. E, $x = \frac{n+1}{2}$ é o único valor possível para x se n é primo. Resta agora considerar o caso em que n é composto. Se $p=q$, o algoritmo obtém a resposta na Etapa 1. Supondo que n é composto e não é um quadrado perfeito, isto é, $1 < p < q < n$. Neste caso, o algoritmo vai parar se forem satisfeitas as desigualdades: $[\sqrt{n}] \leq \frac{p+q}{2} < \frac{n+1}{2}$.

A desigualdade da direita nos diz que $p+q < n+1$. Para $n = pq$, nesta última desigualdade, e subtraindo $q+1$ de ambos os membros, obtém-se $p-1 < q(p-1)$. Como $p > 1$, então $1 < p < q$. Logo $\frac{p+q}{2} < \frac{n+1}{2}$.

Para a desigualdade da esquerda, sabe-se que $[\sqrt{n}] \leq \sqrt{n}$, e basta verificar que $\sqrt{n} \leq \frac{p+q}{2}$.

Logo esta desigualdade é válida se, e somente se, $n \leq \frac{(p+q)^2}{4}$.

Pela Equação (2), tem-se: $\left(\frac{p+q}{2}\right)^2 - \left(\frac{q-p}{2}\right)^2 = n$. Então $\frac{(p+q)^2}{4} - n = \frac{(q-p)^2}{4}$. Como

$$\frac{(q-p)^2}{4} \geq 0, \text{ logo } \frac{(p+q)^2}{4} - n \geq 0.$$

Este algoritmo de Fermat tem uma relação muito importante com a criptografia RSA, lembrando que a segurança do método RSA está na dificuldade em se fatorar a chave pública n , que é o produto de dois números primos. Pensar que escolher dois primos grandes basta para a segurança do método RSA é errôneo, pois se estes dois primos forem muito próximos, o seu produto irá gerar um

número n , onde a sua raiz quadrada será próxima dos dois fatores primos, logo n é facilmente fatorável pelo algoritmo de Fermat [5].

Ineficiência dos algoritmos

Apesar da facilidade em entender e programar os algoritmos da fatoração e dos fatores, estes algoritmos são muitos ineficientes, mesmo com a tecnologia atual. O pior caso para executar o algoritmo é aquele em que o algoritmo executa o maior número de laços, ou seja, $\sqrt{n}$ laços. Para uma estimativa do tempo de execução, considere um número n primo, de 100 ou mais algarismos, ou seja, $n \geq 10^{100}$ e portanto o número de laços será igual a $\sqrt{n} \geq 10^{50}$. Assim, são necessárias pelo menos 10^{50} divisões para garantir que n é primo. Segundo [4], Digamos que nosso computador executa 10^{10} divisões por segundo. Este é um número muito alto, que não é atingido no estado atual da tecnologia. Para estimar o tempo basta calcular $\frac{10^{50}}{10^{10}} = 10^{40}$ segundos para determinar que n é primo.

Um ano tem $60 \text{ (segundos)} \cdot 60 \text{ (minutos)} \cdot 24 \text{ (horas)} \cdot 365 \text{ (dias)} = 31536000$ segundos, resulta em $\frac{10^{40}}{31536000} = 3,1709791983764585 \cdot 10^{32}$ anos.

Portanto, percebe-se que é inviável confirmar que um número de 100 ou mais algarismos é primo usando esse algoritmo. Porém, isso também não significa que o algoritmo é inútil. Se vamos fatorar um inteiro sobre o qual nada sabemos, há sempre a possibilidade que tenha um fator primo pequeno, digamos menor que 10^6 , neste caso o algoritmo da fatoração pode ser utilizado [4].

É muito importante entender que não existe um algoritmo de fatoração que funcione bem para todos os inteiros: disso depende a segurança do método RSA [4]. Ninguém sabe se a inexistência deste algoritmo geral é um problema intrínseco ou tecnológico, ou seja, se um tal algoritmo pode existir ou se ainda ninguém foi esperto o suficiente para inventá-lo [5].

Por que o RSA é seguro?

Cabe ressaltar que o RSA é um método de chave pública, então sejam p e q os dois números primos do método, e $n = pq$. A chave de codificação corresponde à chave pública. Portanto o par (n, λ) é acessível para qualquer usuário. O método RSA só será seguro se for difícil calcular d , quando apenas se conhece os valores de (n, λ) .

Para calcular o valor de d , aplica-se o algoritmo Euclidiano estendido a $\varphi(n)$ e λ , pois $\lambda d \equiv 1 \pmod{\varphi(n)}$. Mas para se calcular $\varphi(n)$, é necessário saber quais são os primos p e q , pois $\varphi(n) = (p-1) \cdot (q-1)$. Portanto para decifrar o código precisa-se fatorar n . Porém se n é um número

muito grande, sabe-se que fatorar n é um problema extremamente difícil, pois não se conhece um algoritmo rápido de fatoração [5].

Um caso particular do RSA

Escolhendo números primos p e q da seguinte maneira:

$p \equiv 5 \pmod{6}$ e $q \equiv 5 \pmod{6}$, logo tem-se que $p-1 \equiv 4 \pmod{6}$ e $q-1 \equiv 4 \pmod{6}$, então $(p-1)(q-1) \equiv 16 \equiv 4 \pmod{6}$, pode-se escrever $(p-1)(q-1) = 6k + 4 = 6k + 3 + 1$, ou seja, $(p-1)(q-1) = 3(2k+1) + 1$, logo:

$3(2k+1) \equiv -1 \pmod{6k+4}$, multiplicando por -1 , para que o resto seja 1.

$3(-2k-1) \equiv 1 \pmod{6k+4}$, somando $6k+4$, obtém-se:

$3(4k+3) \equiv 1 \pmod{6k+4}$.

Por exemplo, considere $p=5$ e $q=11$, logo $n=5 \cdot 11=55$ e $\varphi(n) = (p-1)(q-1) = (5-1)(11-1) = 40$, lembrando que, para codificar, utiliza-se o par (n, λ) , em que $MDC(\lambda, \varphi(n)) = 1$. Neste caso, como p e q estão no caso particular, pois ambos deixam resto 5 na divisão por 6, então pode ser utilizado $\lambda = 3$, mas por que esse valor para λ ? Na sequência será respondida essa pergunta, mas antes, para decodificar, precisa-se conhecer o par (n, d) .

O valor de d é calculado por $\lambda d \equiv 1 \pmod{\varphi(n)}$. Como $\lambda = 3$, fazendo uma relação com o caso particular desenvolvido, em que $3(4k+3) \equiv 1 \pmod{6k+4}$, tem-se que $d = 4k+3$ e $\varphi(n) = 6k+4$. Esta relação justifica o porquê de $\lambda = 3$. Nesse exemplo, pode ser calculado o valor de k em $\varphi(n) = 6k+4$, pois sabe-se o valor de $\varphi(n) = 40$, assim $40 = 6k+4 \Rightarrow k=6$. De posse do valor de $k=6$, calcula-se o valor $d = 4k+3$, então $d = 4 \cdot 6 + 3 = 27$. Portanto é fácil calcular o valor de d , conhecendo-se o $\varphi(n)$, sem precisar aplicar o algoritmo de Euclides [6].

Descrição matemática do método RSA

Para codificar uma mensagem, precisa-se de n , que é o produto de dois números primos p e q , logo $n = pq$ e de um inteiro positivo λ , que seja invertível módulo $\varphi(n)$, ou seja, o $MDC(\lambda, \varphi(n)) = 1$, em que $\varphi(n) = (p-1)(q-1)$. Denomina-se o par (n, λ) chave de codificação e o par (n, d) chave de decodificação do método RSA.

- Represente a mensagem com números inteiros, quebrando em bloco de maneira que esses blocos não ultrapassem o valor de n e não iniciem em zero.

- Para codificar a mensagem B , eleva-se cada bloco B_i à " λ -ésima" potência módulo n , ou seja, $B_i^\lambda \equiv A_i \pmod{n}$. Então cada resultado criptografado é o valor em A_i .

- Para decodificar a mensagem A criptografada, eleve-a a uma outra potência d e calcule o resto da divisão por n , ou seja, $A_i^d \equiv B_i \pmod{n}$. Então o resultado descryptografado é o valor em B .

O valor do d é o inverso de $\lambda[\text{mod}(p-1)(q-1)] = \lambda[\text{mod } \varphi(n)]$, ou seja, $\lambda d \equiv 1 \pmod{\varphi(n)}$.

Pré-codificação

Inicialmente, para utilizar o método RSA é preciso converter a mensagem em uma sequência de números, relacionando em uma tabela de letras com seus respectivos números. Um cuidado na hora de relacionar as letras com os números é importante: por exemplo, se escolher a letra $A = 1$, $B = 2$ e assim por diante, quando escrever o número 12, qual será a interpretação? Nesse sistema de conversão há uma ambiguidade, se $12 = AB$ ou se $12 = L$. Como alternativa, foi utilizado a seguinte tabela de conversão de letras para números (Tabela 1).

Tabela 1: Tabela de Conversão.

A	B	C	D	E	F	G	H	I	J	K	L	M
10	11	12	13	14	15	16	17	18	19	20	21	22
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
23	24	25	26	27	28	29	30	31	32	33	34	35

Fonte: Elaborada pelo autor

Para aplicar o método RSA, foi pré-codificado a palavra *RSA* pela Tabela 1. Com a conversão da mensagem letra a letra, obtém-se: $R = 27, S = 28, A = 10$, assim a mensagem codificada é dada pelo número: 27.28.10.

Codificando e decodificando

Foram utilizados números primos "pequenos" para fins pedagógicos, afim de que os cálculos possam ser verificados facilmente. Porém para uma maior segurança recomenda-se fatores grandes diante da dificuldade da fatoração.

Considerando $p = 5$ e $q = 7$, tem-se que $n = pq = 5 \cdot 7 = 35$ e $\varphi(n) = (p-1)(q-1)$, logo $\varphi(n) = 24$. Para iniciar o processo foi quebrado o código 27.28.10 em blocos menores que $n = 35$, conforme Tabela 2.

Tabela 2: Código em blocos.

B1	B2	B3
27	28	10

Fonte: Elaborada pelo autor

Cada um dos blocos foi codificado por $B_i^\lambda \equiv A_i \pmod{n}$, em que o $MDC(\lambda, \varphi(n)) = 1$, então $MDC(\lambda, 24) = 1$, escolhendo assim $\lambda = 7$. Logo, codificou-se da seguinte maneira:

$$B_i^\lambda \equiv A_i \pmod{n}$$

$$27^7 \equiv A_1 \pmod{35}$$

$$27^7 \equiv (-8)^7 \equiv [(-8)^2]^3 \cdot (-8) \equiv 64^3 \cdot (-8) \equiv (-6)^3 \cdot (-8) \equiv 48 \equiv 13 \pmod{35}$$

$$28^7 \equiv A_2 \pmod{35}$$

$$28^7 \equiv (-7)^7 \equiv [(-7)^2]^3 \cdot (-7) \equiv 49^3 \cdot (-7) \equiv 14^3 \cdot (-7) \equiv 21 \cdot (-3) \equiv 7 \pmod{35}$$

$$10^7 \equiv A_3 \pmod{35}$$

$$10^7 \equiv (10^2)^3 \cdot 10 \equiv (-5)^3 \cdot 10 \equiv (-125) \cdot 10 \equiv 15 \cdot 10 \equiv 10 \pmod{35}$$

Os valores são: $A_1 = 13$, $A_2 = 7$ e $A_3 = 10$, logo a mensagem codificada é: 13.7.10

Tabela 3: Código de conversão.

Mensagem Original	27.28.10
Mensagem Codificada	13.7.10

Fonte: Elaborada pelo autor

A informação necessária para a decodificação consiste no par (n, d) , lembrando que $\varphi(n) = (p-1)(q-1)$ e o valor d é o inverso de λ em $\varphi(n)$, ou seja, $\lambda d \equiv 1 \pmod{\varphi(n)}$. Neste caso, $\lambda = 7$, $\varphi(n) = 24$, donde $7d \equiv 1 \pmod{24}$, logo $d = 7$. Assim, decodificou-se 13.7.10, da seguinte maneira:

$$A_i^d \equiv B_i \pmod{n}$$

$$13^7 \equiv B_1 \pmod{35}$$

$$13^7 \equiv (13^2)^3 \cdot (13) \equiv (-6)^2 \cdot (-6) \cdot 13 \equiv (-6) \cdot 13 \equiv -78 \equiv 27 \pmod{35}$$

$$7^7 \equiv B_2 \pmod{35}$$

$$7^7 \equiv (7^2)^3 \cdot 7 \equiv 14^3 \cdot 7 \equiv 196 \cdot 14 \cdot 7 \equiv -147 \equiv 28 \pmod{35}$$

$$10^7 \equiv B_3 \pmod{35}$$

$$10^7 \equiv (10^2)^3 \cdot 10 \equiv (-5)^3 \cdot 10 \equiv (-125) \cdot 10 \equiv 15 \cdot 10 \equiv 10 \pmod{35}$$

Os valores são: $B_1 = 27$, $B_2 = 28$ e $B_3 = 10$, logo a mensagem original é: 27.28.10,

que corresponde as letras RSA.

4.5. Análise da função de Euler no método RSA

Suponha que $n = pq$ e $\varphi(n) = (p-1) \cdot (q-1)$ sejam ambos conhecidos. Pode-se determinar p e q a partir deles.

Para $\varphi(n) = (p-1) \cdot (q-1)$, tem-se que $\varphi(n) = pq - (p+q) + 1 = n - (p+q) + 1$, logo $p+q = n - \varphi(n) + 1$, portanto tem-se a soma dos dois números primos.

Sabe-se que $(p+q)^2 - 4n = (p^2 + 2pq + q^2) - 4n = (p^2 + 2pq + q^2) - 4pq = (p-q)^2$, logo:

$p-q = \sqrt{(p+q)^2 - 4n}$ ou $p-q = \sqrt{(n - \varphi(n) + 1)^2 - 4n}$, portanto tem-se a diferença dos dois números primos.

Sendo conhecidos $p+q$ e $p-q$, calcula-se facilmente o valor de p e q por meio da resolução de um sistema linear, ou seja, o procedimento fatora o valor em n .

A função $f(m) = (m-n-1)^2 - 4n$

Seja a função $f: R \rightarrow R$, tal que $f(m) = (m-n-1)^2 - 4n$. Assim, tem-se ainda que $f(m) = m^2 - 2mn - 2m + n^2 - 2n + 1$, cujas raízes são: $m_1 = n+1+2\sqrt{n}$ e $m_2 = n+1-2\sqrt{n}$. A raiz m_1 é positiva, basta analisar m_2 . Logo, $m_2 = n+1-2\sqrt{n} > 0$, tem-se $n+1 > 2\sqrt{n}$. Elevando ao quadrado ambos os membros obtém-se $(n+1)^2 > (2\sqrt{n})^2$. Logo $n^2 + 2n + 1 > 4n$ que é equivalente a $n^2 - 2n + 1 > 0$ e $n^2 - 2n + 1 = (n-1)^2 > 0$; portanto ambas as raízes são positivas.

Escolher dois números primos quaisquer $p < q$, efetuar o produto deles para gerar o valor de n e efetuar a subtração de q por p para gerar o valor de g , logo $n = pq$ e $g = q - p$. Considere a função $f: R \rightarrow R$, tal que $f(m) = (m-n-1)^2 - 4n$. Substituir o resultado de g^2 no lugar de $f(m)$ e resolver a equação em função de m . Sendo assim, $g^2 = (m-n-1)^2 - 4n$ ou $m^2 - 2mpq - 2m + p^2q^2 + 1 - q^2 - p^2 = 0$. As raízes dessa equação são $m_1 = 1 - (p+q) + pq$ e

$m_2 = 1 + p + q + pq$. O importante para o método RSA é a menor raiz, sendo esta o $m_1 = 1 - (p + q) + pq = (p - 1) \cdot (q - 1)$.

O que é notável, caso alguém invente um método rápido que encontre o valor de $g^2 = (q - p)^2$, fica fácil de encontrar os fatores primos p e q .

O conceito do vértice da parábola por meio das raízes é a média aritmética das raízes que resulta no x_v da função, ou seja, $x_v = \frac{1 - (p + q) + pq + 1 + p + q + pq}{2} = \frac{2pq + 2}{2} = pq + 1$ e também o conceito do y_v , calculado por meio de $f(x_v)$, ou seja, $f(x_v) = (pq + 1 - pq - 1)^2 - 4pq = -4pq$. A

figura 1, apresenta um esboço do gráfico da função $f(m) = (m - n - 1)^2 - 4n$.

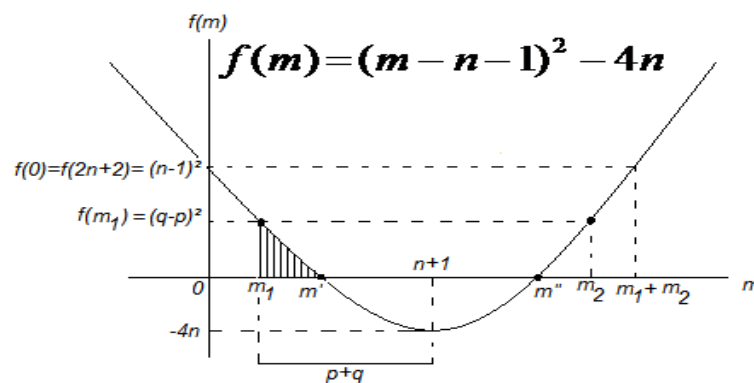


Figura 1: A função $f(m) = (m - n - 1)^2 - 4n$

Fonte: Elaborada pelo autor

Resultados e Discussão

Na área de estudos em algoritmos de fatoração não é conhecido um algoritmo determinístico, em tempo polinomial, que encontre um fator de um inteiro composto n com muitos algoritmos. Este fato está diretamente relacionado ao método RSA, pois caso este algoritmo existisse, seria possível encontrar facilmente os fatores primos de n , e consequentemente saber o valor de $\varphi(n)$ e também encontrar o expoente d , que é a chave de decodificação do par (n, d) .

Foram analisados três casos que podem ser estudados na tentativa de quebrar o método RSA (fatorar n , encontrar $\varphi(n)$ ou encontrar d sem fatorar n ou encontrar $\varphi(n)$) e também proposto um método para tentar encontrar $\varphi(n)$. No apêndice encontra-se o algoritmo implementado no Maple12, que utiliza a função $f(m) = (m - n - 1)^2 - 4n$, até gerar o quadrado perfeito, encontrando assim $\varphi(n) = (p - 1) \cdot (q - 1)$ e por meio da resolução de uma equação do segundo grau, encontram-se os fatores primos p e q .

Para o caso de encontrar $\varphi(n)$, conhecendo apenas n , utilizando a função $f(m)=(m-n-1)^2-4n$. Observa-se que a busca pelo valor de $\varphi(n)$, pode ser feita por meio da menor raiz na função, sendo verificado se a parte inteira da raiz é múltiplo de quatro; caso negativo, deve-se subtrair uma unidade até encontrar o primeiro valor múltiplo de quatro. Substitui-se esse valor próximo da raiz, se o resultado for um quadrado perfeito, então o valor é o $\varphi(n)$; senão subtrai-se quatro unidades e substitui-se novamente na função até gerar um quadrado perfeito.

O fato de verificar se o número é um múltiplo de quatro, e também de subtrair quatro até encontrar um quadrado perfeito, deve-se a característica da função $\varphi(n)=(p-1)\cdot(q-1)$, pois p e q são primos, logo $\varphi(n)$ será o produto de dois números pares, resultando num múltiplo de quatro.

Portanto, se existisse um método aplicado na função $f(m)=(m-n-1)^2-4n$, que encontrasse rapidamente um quadrado perfeito no intervalo de 0 a $n+1-2\sqrt{n}$, então o valor aplicado na função é o valor de $\varphi(n)$.

Por meio da função $f(m)=(m-n-1)^2-4n$, foi possível encontrar os fatores primos com um menor número de repetições em relação ao algoritmo de Fermat, pois pelo método de Fermat, o incremento de x é de apenas uma unidade e calcula-se $y=\sqrt{x^2-n}$, até y gerar um número inteiro, enquanto na função $f(m)=(m-n-1)^2-4n$, subtrai-se quatro unidades e aplica-se o novo ponto até gerar um quadrado perfeito.

Considerações finais

Para a continuidade deste estudo sobre a criptografia RSA, recomenda-se a análise da função $f(m)=(m-n-1)^2-4n$, tentando delimitar um intervalo mais eficiente de busca do quadrado perfeito, pois no algoritmo que se encontra no apêndice, o critério utilizado para iniciar, foi pela parte inteira da menor raiz da função, avaliando se é múltiplo de quatro, sendo assim, a busca inicia na proximidade do quadrado perfeito 1, ou seja, para números com muitos algarismos, o quadrado perfeito na função será um número muito alto, visto que esse quadrado perfeito é a diferença dos números primos elevado ao quadrado, os quais não são próximos. Assim, sugere-se um estudo para delimitar melhor esse intervalo de busca ou algum método que identifique rapidamente os quadrados perfeitos na função.

Referências

- [1] RIBENBOIM, P. **Números Primos: Velhos Mistérios e Novos Recordes**. 1 ed. Rio de Janeiro: IMPA, 2012. 328 p.
- [2] IEZZI, G. [et al.]. **Matemática: ciência e aplicações 2: ensino médio**. 6. ed. São Paulo: Saraiva, 2010.
- [3] RIVEST, R.L., SHAMIR, A. E ADLEMAN, L.M. **A Method for obtaining digital signatures and public-key cryptosystems**. Commun ACM 21, (1978), 120-126.
- [4] COUTINHO, S.C. **Números Inteiros e Criptografia RSA**. Rio de Janeiro, IMPA, 2011. 226 páginas (Coleção Matemática e Aplicações).
- [5] MOLINARI, J. R. A.. **Números Primos e a Criptografia RSA**. PROFMAT - UEPG: Ponta Grossa, 2016.
- [6] EUCLIDES. **Os elementos**. UNESP, 2009. Tradução brasileira por Irineu Bicudo.

Apêndice

Algoritmo programado no Maple 12.

```
> #N:= p*q; # M:= (p-1)*(q-1);
> p:= ; q:= ; N:=p*q;
> Digits:= 600;
> f:= x -> x^2 + (M - N - 1)*x + N;
> Delta:= (M - N - 1)^2 - 4*N;
> fDelta:= unapply(Delta,M);
> expand(fDelta(M));
> Raizes_de_M:= solve(Delta,M);
> M1:=floor(min(evalf(Raizes_de_M)))+1; M2:=floor(max(evalf(Raizes_de_M)));
> resid:=5;

passo4ou1:=1;
while passo4ou1 = 1 do
    M1:= M1-passo4ou1;
    if irem(M1,4) = 0 then
        passo4ou1:=4;
    end if;
end do;
> M1;fDelta(M1);sqrt(fDelta(M1));
> TempoIni:=time();
resid:=5;
while resid > 0 do
    M1:=M1-4;
    resid:= frac(evalf(sqrt(fDelta(M1))))
end do;
TempoFinal:=time()-TempoIni;
> M1;
```