Criptografia RSA

Aplicação do Método RSA

Raquel Nunes da Silva

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, 2020

Raquel Nunes da Silva

Criptografia RSA

Aplicação do Método RSA

Monografia apresentada ao curso de Bacharelado em Matemática Computacional do Centro de Informática, da Universidade Federal da Paraíba, como requisito para a obtenção do grau de Bacharel em Matemática Computacional

Orientador: Prof.Roberto Quirino do Nascimento

Agosto de 2020

Ficha catalográfica: elaborada pela biblioteca do CI.

Será impressa no verso da folha de rosto e não deverá ser contada.
Se não houver biblioteca, deixar em branco.

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

Trabalho de Conclusão do Curso de Bacharelado em Matemática Computacional intitulado Criptografia RSA de autoria de Raquel Nunes da Silva, aprovada pela banca examinadora constituída pelos seguintes professores:

Prof. DSc. Roberto Quirino do Nascimento
Depto. Comp. Científica-CI

Prof. DSc. Lucídio dos Anjos Formiga Cabral
Depto. Comp. Científica-CI

Prof. DSc. Gilberto Farias de Souza Filho
Depto. Comp. Científica-CI

Prof. DSc. Roberto Quirino do Nascimento
Coordenador de Curso
CI/UFPB

João Pessoa, 12 de agosto de 2020

Centro de Informática, Universidade Federal da Paraíba
Rua dos Escoteiros, Mangabeira VII, João Pessoa, Paraíba, Brasil CEP: 58058-600
Fone: +55 (83) 3216 7093 / Fax: +55 (83) 3216 7117

"A Matemática é o alfabeto que Deus usou para escrever o Universo."
Galileu Galilei

DEDICATÓRIA

A dedicatória é opcional

AGRADECIMENTOS

Quero agradecer, primeiramente a Deus por Ele ter me dado força para continuar estudando e chegar até onde eu cheguei, visto que o curso não é fácil. Segundo aos meus pais e irmãos que sempre me incentivaram e me ajudaram em tudo. Terceiro, quero agradecer ao meu orientador, que me ajudou muito ao longo da minha jornada pelo curso, estou muito grata por tudo. Quero agradecer aos meus amigos que em ajudaram de alguma forma. Obrigada a todos!

RESUMO

O objetivo desse trabalho foi implementar o método de criptografia RSA em um software algébrico, que escolhemos o Matlab. Foi demonstrados os teoremas e lemas necessários para entender o funcionamento do método, como é o caso do algoritmo de Euclides e o algoritmo de divisão, que é visto nas escolas, porém não com tanto ênfase. As rotinas computacionais utilizadas foram descritas, detalhadas o seu passo a passo. Foram detalhados alguns exemplos para que os teoremas e os métodos computacionais possam ficar mais fácil de compreender. Também ficou explicito a segurança que a criptografia tem e dar ao desenvolvedores/programadores e alguns exemplos de aplicação onde utilizam a criptografia foram destacados no texto.

Palavras-chave: <RSA>, <Números primos>, <Algoritmo euclidiano>.

ABSTRACT

The objective of this work was to implement the RSA encryption method in an algebraic software, which we chose from Matlab. The theorems and mottos necessary to understand the functioning of the method have been demonstrated, as is the case with euclid's algorithm and the division algorithm, which is seen in schools, but not with so much emphasis. The computational routines used were described, detailed their step by step. Some examples have been detailed so that the theorems and computational methods can be easier to understand. It was also explained the security that encryption has and give to developers/programmers and some application examples where encryption use have been highlighted in the text.

Key-words: RSA, Numbers Primes, Euclidean algorithm.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

SIGLA – NOME COMPLETO

RSA - Rivest, Shamir e Adleman

MDC - Máximo Divisor Comum

Sumário

1 INTRODUÇÃO

Hoje em dia valorizamos a nossa segurança, seja ela nas nossas redes sociais, em nossa agência bancária e etc. Como hoje em dia , nossa segurança é o mais importante, antigamente também existia essa pensamento. O primeiro povo a usar a Criptografia foi o povo Egípcio, o primeiro documento surgiu há cerca de 1990 anos antes de Cristo, onde foi utilizados hieróglifos fora do padrão.

Um dos maiores desafios do mundo digital é a segurança de dados, pessoas empresas e governos querem ter a certeza de que seus dados, informações sigilosas, mensagens confidenciais etc, não sejam violados nem cheguem em mãos não recomendadas, uma das formas de proteção de dados e informações é a codificação dos mesmos, tal fato consiste em escrever ou enviar mensagens cujo conteúdo seja irreconhecível ou indecifrável aos olhos de um agente que não saiba que a mensagem está codificada, esta sistemática chama-se criptografia, prática existentes há milhares de anos, há relatos que os egípcios já utilizavam tal prática a mais de 1900 anos A.C(ROMAGNOLO, 2007) .

Uma das mais conhecidas utilização da criptografia ocorreu durante a $2^{a.}$ Guerra Mundial, quando os alemães desenvolveram uma máquina para codificar e decodificar mensagens, chamada *Enigma*, inicialmente, alguns oficiais poloneses, entre eles o matemático Marian Jerewski, Jerewski exilou-se na França e trabalhando com a resistência francesa conseguiu iniciar um projeto, no entanto, não tendo obtido muito sucesso juntou-se ao grupo inglês liderado pelo matemático Alan Turing e finalmente conseguiram decodificar as mensagens nazistas

O filme "Jogo da imitação", onde foi documentada a máquina de Alan Turing, responsável por decifrar códigos alemães interceptados pela Grã-Bretanha, durante a segunda guerra mundial, é um dos exemplos clássicos do uso de criptografia durante conflitos. As agências especiais de inteligência criam suas próprias criptografias para se comunicar, evitando assim que mensagens possam ser interceptadas e lidas (ROMAGNOLO, 2007). Hoje em dia, vários aplicativos utilizam a criptografia para dar mais segurança aos seus usuários e dessa forma, evitam que os usuários tenham suas mensagens interceptadas por terceiros, como é o caso dosaplicativos citados abaixo:

- WhatsApp: um dos aplicativos mais utilizados do mundo, utiliza um protocolo de criptografia super seguro desenvolvido pela Open Whisper Systems (a empresa por trás do aplicativo de mensagens seguras Signal) em todas as plataformas móveis. Graças a esse protocolo, somente o remetente e o destinatário têm as chaves para descriptografar as mensagens enviadas via WhatsApp, o que significa que elas não podem ser acessadas e lidas por nenhuma outra pessoa. As chamadas de voz e vídeo também são criptografadas (CORRIGAN, 2018);

- Facebook Messenger: esse aplicativo adicionou seu recurso de Conversas secretas para proteger as mensagens com o protocolo Signal de criptografia completa (também usado pelo WhatsApp). No entanto, o Signal e o WhatsApp têm criptografia completa por padrão, enquanto as Conversas secretas devem ser ativadas (CORRIGAN, 2018).

O termo criptografia surgiu da fusão das palavras gregas "Kryptós"e "gráphein", que significam "oculto"e "escrever", respectivamente. A criptografia é um conjunto de técnicas e conceitos utilizados para codificar e decodificar uma mensagem/informação. As técnicas mais utilizadas, atualmente, utilizam o conceito de "chaves". Essas chaves são dados que são capazes de codificar/decodificar as informações a serem enviadas.

Uma chave pode ser de dois tipo: chave pública e chave privada.

* A chave pública é utilizada na codificação das informações, essa chave fica com o emissor (quem envia a mensagem);

* A chave privada é utilizada para decodificar a mensagem recebida (essa chave fica com o receptor).

Existem dois tipos de chaves criptográficas: as chaves simétricas e as chaves assimétricas.

* Chaves simétricas: As chave de codificação e decodificação são as mesmas, exemplos de algoritmos que utilizam esse tipo de chave: DES (Data Encryption Standard), RC (Ron's Code ou Rivest Cipher), EAS (Advanced Encryption Standard), IDEA (International Data Encryption Algorithm), entre outros.

* Chaves assimétricas: A chave de codificação e decodificação são diferentes, ou seja, utilizam a chave pública e privada. Exemplos de algoritmos que utilizam esse tipo de chave: RSA (Rivest, Shamir and Adleman), ElGamal, entre outros.

1.1 Definição do Problema

O método escolhido para ser analisado e implementado foi o método RSA. Esse método foi inventado em 1978 por R. L. Rivest, A. Shamir e L. Adleman. Como pode ser observado, RSA corresponde às iniciais dos nomes dos criadores.

O RSA é um método de Chave assimétrica, ou seja, ele utiliza chave pública e chave privada. Para a criação dessas chaves são necessário dois números primos. Neste trabalho iremos implementar esse método e provar sua eficiência e sua segurança. Será utilizado um software matemático e provas matemáticas para tal feito.

1.2 Premissas e Hipóteses

Diante do que já foi apresentado, algumas perguntas devem estar sendo feitas, por exemplo: "Como um método que tem apenas dois números primos, como protagonistas, pode funcionar?"ou "Porque esse método foi escolhido?"

Partindo da hipótese que o método do RSA é muito seguro, iremos detalhar o que torna esse método tão seguro. A princípio, sabemos que todo número pode ser decomposto ou fatorado em números primos, por exemplo: 20 pode é fatorado da seguinte maneira 2*2*5=20. Porém, quando número é primo, essa fatoração fica restrita a dois números, o número 1 e ele próprio, por exemplo: o número 17 pode ser fatorado da seguinte maneira 1*17=17; o número 13 é fatorado da seguinte maneira 1*13=13.

Fatorar um número é algo "fácil"de fazer, mas se o número for primo, isso deixa a tarefa difícil, tipo, muito difícil se for feito manualmente. Por exemplo: como fatorar o número 997?

* Esse número não é par, logo não é divisível por 2, 4, 6 e todos os pares possíveis;

* A soma dos dígitos 9+9+7=25=2+5=7, 7 não é divisível por 3, logo 3 não é um divisor para esse número;

* O último dígito não é 0 e nem 5, logo 5 não divide esse número;

* Um número é divisível por 7 se multiplicarmos o último número por 2 e subtrairmos o resultado pelos números que restaram. Se o resultado for divisível por 7, então o número é divisível por 7. Ou seja, 99-(2*7)=99-14=85, 85 não é divisível por 7. Então, 997 não é divisível por 7;

A partir daqui, isso já fica um pouco difícil, pois teríamos que dividir 997 por todos os números que não foram descartados. Em uma rotina (computacionalmente falando) simples podemos dizer se o número é primo ou não (997 é um número primo). Vale ressaltar que quanto maior os números de dígitos o número tiver, mais tempo será preciso para fatorá-lo. Por exemplo: para fatorar um número com 512 dígitos, precisamos apenas de 4 dias, já para calcular um número com 1024 dígitos precisamos apenas de 100 mil anos, utilizando os métodos atuais (OLIVEIRA et al, 2003).

O que explicamos até agora é para fatorar um número primo, porém no método RSA, precisamos de um produto de primos. Imagine multiplicar dois número com 1000 e 3000 dígitos, quanto tempo para fatorar um número dessa magnitude? Quanto tempo para achar os respectivos valores de cada primo?

1.3 Objetivo geral

O objetivo desse trabalho é estudar criptografia RSA, mostrar como a matemática funciona dentro dessa área que é bastante utilizada e que não é tão discutida e nem divulgada, de maneira simples e compreensível. Compreender como a tecnologia não consegue, de maneira simples e efetiva, quebrar a segurança desse método, já que em meio ao avanço da tecnologia, a criptografia continua sendo muito segura.

Para facilitar a compreensão de quem está estudando sobre a criptografia RSA iremos provar alguns teoremas e explicar onde eles se encaixam na criptografia. A implementação será feita através de uma linguagem algébrica, pela facilidade na implementação e nas contas, já que iremos trabalhar com números grandes, na teoria.

1.4 Objetivos específicos

A implementação do método será do seguinte modo:

* Função Primo: como o próprio nome diz, essa função irá dizer se o número é primo ou não. Será uma função onde o usuário irá digitar os números que serão utilizados como protagonistas;

* Função Codificador: função que codifica a mensagem original em outra. Essa etapa é a que se muda as letras da mensagem por número;

* Função Decodificador: ao contrário da função anterior, elas decodifica a mensagem criptografada;

* Função Bloco: essa é a função que irá dividir a mensagem codificada em blocos menores. A mensagem original criptografada será muito grande, como iremos trabalhar com exponenciação, será necessário fazer essa "quebra"para facilitar o cálculo;

* Função Reduzida: essa é função que calcula a forma reduzida modular de um determinado número, por exemplo: 7^{256} mod 13 para calcular esse número de forma rápida e eficiente ($7^{256} \bmod 13 = 9$);

* Função Euclides Estendido: é a função que calculará o máximo divisor comum de um número. É uma função melhorada do método de Euclides.

Vale ressaltar que algumas dessas funções tem como base alguns algoritmos, como é o caso da Função Euclides. O termo função aqui mencionado refere-se a uma estrutura do matlab denominada functions.

1.5 Estrutura da monografia

A monografia foi dividida em seções e subseções. Na seção 2 encontra-se os Conceitos Gerais e Revisão da Literatura, nessa seção é encontrado os teoremas e proposições utilizados e alguns trabalhos da área. Na seção 3 encontra-se o corpo da monografia, a implementação do método e explicação de cada parte. Na seção 4 encontra-se os resultados obtidos. Na seção 5 encontra-se a conclusão do trabalho. E por fim, encontra-se o apêndice e referências.

2 CONCEITOS GERAIS E REVISÃO DA LITERATURA

2.1 Algoritmo de Euclides

Nesta seção apresentaremos o Teorema da Divisão, seguido do algoritmo da divisão e dois Algoritmos denominados algoritmo de Euclides e Algoritmo de Euclides estendido. O Algoritmo de Euclides Estendido é o Algoritmo de Euclides "melhorado". O objetivos principal do algoritmo euclidiano é calcular o Máximo divisor Comum(MDC) entre dos números inteiros, ele pode ser calculado recursivamente.

Definição: Dado um número inteiro a definimos os múltiplos de a como $0, \pm a, \pm 2a, \pm 3a, \ldots,$, isto é, os números da forma $\pm ka$ com $k \in \mathbb{Z}$

Definição: Dado um número inteiro a definimos os múltiplos de a como $0, \pm a, \pm 2a, \pm 3a, \ldots,$, isto é, os números da forma $\pm ka$ com $k \in \mathbb{Z}$.

Quando se tem $a, b, \mathbb{Z}$ e $c = ab$ dizemos que a é um divisor de c ou simplesmente que a *Divide* c Notação: $a|c$ A relação $|$ possuem as seguintes propriedades:

a) $a|a$

b) Se $a|b$ e $b|a$ então e $a, b \in \mathbb{Z}$, então $a = b$.

c) Se $a|b$ e $b|c$, então $a|c$.

d) Se $a|b$ e $a|c$, então $a|(bx + cy)$ para todo $x, y \in \mathbb{Z}$

A princípio iremos ao Teorema de Divisão.

Teorema de Divisão: *Sejam a e b inteiros positivos. Existem números inteiros q e r tais que*

$$\text{a} = \text{bq} + \text{r e } 0 \leq r < b \tag{1}$$

Temos que: a é o dividendo, b o divisor, r é o resto da divisão e q é o quociente.

Prova: Se a é múltiplo de b então $a = kb + 0$ com $k \in \mathbb{Z}$, nesse caso $q = k$ e $r = 0$. Suponha que a não seja múltiplo de b, nesse casso existe $k \in \mathbb{Z}$ tal que:

$$kb < a < (k+1)b \Rightarrow 0 < a - kb < b$$

Como a, b e k são inteiros $a - kb$ também é inteiro, fazendo $r = a - kb$. Vamos provar agora a unicidade, suponha que exista $r_1 \neq r$ e $q_1 \neq q$ tais que:$a = bq_1 + r_1$ e $0 \leq r_1 < b$

se $r > r_1$ então $b(q - q_1) = r_1 - r$ como $r > r_1$, $q > q_1$ e $r1 = r + b(q - q_1)$ logo $r_1 \geq b$ pois $q - q_1 > 1$o que é uma contradição.
Obs: O Teorema da divisão também é conhecido como Algoritmo de Euclides Máximo Divisor Comum

Definição: Dados dois inteiros a e b dizemos que r é um divisor comum de a e b Se

$$r|a \text{ e } r|b$$

Definição: Dados $a, b \in \mathbb{Z}$, não ambos nulos, dizemos que $d \in \mathbb{Z}_+^*$, é Máximo Divisor Comum de ae b, quando d cumpre duas condições.

(i) $d|a$ e $d|b$;

(ii) Se $e \in \mathbb{Z}$, tal que $e|a$ e $e|b$, então $e|d$, ou seja, d é o maior divisor comum de ae b.

Notação: Denotaremos, o Máximo Divisor Comum de dois números a e b, simplesmente por $mdc(a, b)$.

Proposição 1: Dados $a, b \in \mathbb{Z}$ e $d = mdc(a, b)$. As seguintes afirmações são verdadeiras:

(i) Se $a = b = 0$, então $d = 0$;

(ii) Se $a = 0$ e $b \neq 0$, então $d = |b|$, já que $d \in \mathbb{Z}_+^*$

(iii) Se $d = mdc(a, b)$, então $d = mdc(-a, b) = mdc(a, -b) = mdc(-a, -b)$

Prova: (i) e (ii) são triviais;
(iii) Dados $a, b \in \mathbb{Z}$, ambos não nulos. Temos que o maior divisor de a é $|a|$. Daí temos que o maior divisor de $-a$ é $|a|$. Dessa forma, $mdc(a, b) = mdc(-a, b)$ e analogamente $mdc(a, -b) = mdc(-a, -b) = mdc(a, b)$

Lema: Dados $a, b \in \mathbb{Z}$, Existe um inteiro positivo d, que é máximo divisor comum de a e b.
Prova: Considere $a, b \in \mathbb{Z}$. Notemos que caso, $a|b$ ou a = 1, temos que o mdc(a,b) = —a—, assim podemos supor que $1 < a < b$ e que a não divide b, logo pelo Algoritmo de Euclides existem q_1, r_1, tais que:

$$b = aq_1 + r_1 \text{ com } 0 < r_1 < a$$

Daí surge duas possibilidades:

$$r_1|a$$

Assim:

$$r_1 = mdc(a, r_1) = mdc(a, b - q_1 a) = mdc(a, b)$$

Ou então

$$r_1 \text{ não divide a}$$

Aplicando o algoritmo de Euclides em a e r_1, desta maneira existem inteiros q_2 e r_2, tais que:

$$\text{a} = q_2 r_1 + r_2 \text{ com } 0 < r_2 < r_1 < a$$

O que também nos dar duas possibilidades

$$r_2 | r_1$$

Assim:

$$r_2 = mdc(r_1, r_2) = mdc(r_1, a - q_2 r_2) = mdc(r_1, a) = mdc(b - q_1 a, a) = mdc(a, b)$$

Ou então:

$$r_2 \text{ não divide } r_1$$

Aplicando novamente o algoritmo de Euclides. Portanto existem inteiros q_3 e r_3, tais que:

$$r_1 = q_3 r_2 + r_3 \text{ com } 0 < r_3 < r_2 < r_1 < a$$

Mas este processo não é infinito, pois pelo principio da Boa Ordenação a sequência $a > r_1 > r_2 > ...$ possui menor um menor elemento. Portanto, para algum n teremos $r_n | r_{n-1}$, pois teremos o resto 0 em algum momento, implicando em

$$mdc(a, b) = r_n$$

Ou seja, mostramos a existência do $mdc(a, b)$.

Vamos mostrar a unicidade. Para tal, suponhamos que $mdc(a, b) = d$ e $mdc(a, b) = d'$. Notemos que tanto d, quanto d' são divisores comuns de a e b, assim $d | d'$ e $d' | d$, e como d e d' são ambos positivos segue que $d = d'$. Provando assim a unicidade de $mdc(a, b)$.

Proposição 2: (Identidade de Bezout). O máximo divisor comum de dois inteiros a e b, não nulos simultaneamente, se escreve como combinação linear de a e b, ou seja, existem inteiros x e y tais que $mdc(a, b) = ax + by$.

Prova: Aplicando a proposição 1.(iii), sem perda de generalidade, podemos supor que $a > 0$ e $b > 0$. Tomemos o conjunto

$$\mathrm{A} = ax + by; x, y \in \mathbb{Z}$$

notamos facilmente que existem elementos estritamente positivos em A, já que $a \in A$, basta tomar $x = 1ey = 0$ e $b \in A$ tomemos $x = 0ey = 1$. Seja d o menor dos elementos positivos de A. Mostraremos que d é o máximo divisor comum entre a e b. É fato que $d > 0$. Como $d \in A$, então existem $x_0, y_0 \in \mathbb{Z}$, de maneira que $d = ax_0 + by_0$. Aplicando o algoritmo da divisão aos números a e d segue que:

$$a = dk + r \ 0 \leq r < a$$

Das duas últimas igualdades tiramos $a = (ax_0 + by_0)k + r$. Ou ainda podemos concluir que $r = a(1kx_0) + b(y_0)k$. Portanto $r \in A$. Sendo r positivo e levando em conta a escolha do d a conclusão é que $r = 0$. Daí ficamos com $a = dk$, o que mostra que $d|a$. A prova que $d|b$ é análoga. Para finalizar temos que se $d^{'}|aed^{'}|$b, então, pelo fato de $d = ax_0 + by_0$, $d|^{'}d$.

Algoritmo de Euclides.

O objetivo do algoritmo euclidiano calcula o mdc entre dois números. Tal algoritmo é definido da seguintes Forma:

Lema: *Sejam a e b inteiros positivos. Suponhamos que existam unteiros g e s tais que a = bg + s. Então mdc(a,b) = mdc(b,s).*

Para calcular o mdc pelo algoritmo euclidiano teremos que calcular uma sequência de divisões, que são do tipo:

$$\begin{gathered} a = bq_1 + r_1 e 0 \leq r_1 < b \\ b = r_1 q_2 + r_2 \text{ e } 0 \leq r_2 < b \\ r_1 = r_2 q_3 + r_3 \text{ e } 0 \leq r_3 < b \\ r_2 = r_3 q_4 + r_4 \text{ e } 0 \leq r_4 < b \\ \ldots \qquad \ldots \end{gathered} \tag{2}$$

É fácil perceber que b é menor que r_1 e que r_1 é menor que r_2 e assim sucessivamente, logo teremos a seguinte desigualdade:

$$b > r_1 > r_2 > r_3 > r_4 ... \geq 0$$

Utilizando a equação (2) com um pouco mais de detalhe, teremos:

$$a = bq_1 + r_1 \text{ e } 0 \leq r_1 < b$$

$$
\begin{gathered}
b = r_1 q_2 + r_2 \text{ e } 0 \leq r_2 < b \\
r_1 = r_2 q_3 + r_3 \text{ e } 0 \leq r_3 < b \\
r_2 = r_3 q_4 + r_4 \text{ e } 0 \leq r_4 < b \\
\ldots \qquad \ldots \\
r_{n-3} = r_{n-2} q_{n-1} + r_{n-1} \text{ e } 0 \leq r_n < r_{n-2} \\
r_{n-2} = r_{n-1} q_n \text{ e } 0 \; r_n = 0
\end{gathered}
\tag{3}
$$

No lado esquerda da equação, na última linha, temos que r_{n-1} divide r_{n-2}. Dessa maneira temos que o mdc entre os dois é o próprio r_{n-1}. Então, mdc(r_{n-2},r_{n-1})=r_{n-1}. Vamos a um exemplo prático:

r_{n-2} = 10, r_{n-1} = 5 e q_n = 2
*isso significa que 10 = 5*2,*
logo 5 divide 10 e o mdc(10,5)=5

Aqui é onde entra o lema. Aplicando o lema na penúltima linha, concluímos que mdc(r_{n-3},r_{n-2}) = mdc(r_{n-2},r_{n-1}). Para deixar mais claro, iremos ao um exemplo com números.

r_{n-3} = 1234, r_{n-2} = 54
r_{n-3} é da forma $r_{n-3} = r_{n-2} g + s \Rightarrow 1234 = 54\text{*}22 + 48$
Agora vamos para a segunda parte: r_{n-2} = 54, $r_{n-1} = 48$ Obs.: lembre que o s passa a ser o r_{n-1}, ou melhor, ele passa a ser o segundo elemento do mdc.
$r_{n-2} = r_{n-1} g + s \Rightarrow 54 = 48\text{*}1 + 6$
o mdc(1234,54) = 2 e o mdc(54,48) = 2, logo mdc(1234,54) = mdc(54,48)

Algoritmo de Euclides Estendido.
Agora vamos de fato ao algoritmo euclidiano estendido. Sejam mais uma vez a e b inteiros positivos, então dizemos que d é o mdc entre ele. Dessa maneira é possível achar inteiro β e α tais que

$$
a\alpha + b\beta = d \tag{4}
$$

Em alguns caso é possível encontrar β e α inteiros negativos.

É possível modificar o algoritmo de Euclides de modo a calcular α, β e d simultaneamente, como na equação acima. Este algoritmo é chamado de Algoritmo de Euclides Estendido.

Os valores de β e α são os valores do resto e quociente, respectivamente. Para encontrarmos esses valores e que d vamos usar a fórmula (2), porém um pouco modificada. Utilizar as fórmulas (2) ao pé da letra não é muito eficiente pelo fato do computador ter que armazenar os valores dos quociente e restos de cada divisão.Por isso, vamos utilizar a implementação do Knuth. A ideia dele consiste em expressar cada um dos restos obtidos na sequência de divisões em termos de a e b, semelhante a fórmula (4). Ou seja, para uma certa divisão é necessário guardar apenas as duas interações anteriores.

Agora vamos aos cálculos. Para isso iremos utilizar a fórmula(2) e fórmula(4). Teremos

$$\begin{gathered}
a = bq_1 + r_1 \text{ e } r_1 = ax_1 + by_1 \\
b = r_1q_2 + r_2 \text{ e } r_2 = ax_2 + by_2 \\
r_1 = r_2q_3 + r_3 \text{ e } r_3 = ax_3 + by_3 \\
r_2 = r_3q_4 + r_4 \text{ e } r_4 = ax_4 + by_4 \\
\ldots \qquad\qquad \ldots \\
r_{n-3} = r_{n-2}q_{n-1} + r_{n-1} \text{ e } r_1 = ax_{n-1} + by_{n-1} \\
r_{n-2} = r_{n-1}q_n \text{ e } 0 \; r_n = 0
\end{gathered} \tag{5}$$

Vamos colocar as informações das fórmula anterior em uma tabela. Lembrando que ps valores dos x's e y's são números a determinar.

restos	quociente	x	y
a	*	x_{-1}	y_{-1}
b	*	x_0	y_0
r_1	q_1	x_1	y_1
r_2	q_2	x_2	y_2
r_3	q_3	x_3	y_3
	...		
r_{n-2}	q_{n-2}	x_{n-2}	y_{n-2}
r_{n-1}	q_{n-1}	x_{n-1}	y_{n-1}

(6)

Vamos aos detalhes de como preencher essa tabela.

* Na coluna *resto* divide-se o a por b e o resto da divisão é atribuído no lugar do r_1. Depois calcula-se a divisão de b por r_1 e o resto da divisão é atribuído a r_2 e assim sucessivamente até o *resto* ser 0. Observação, as duas primeiras linhas adicionada na tabela serão chamadas de -1 e 0, foram adicionada para facilitar na hora da implementação.

* Na segunda coluna *quociente* é o quociente das divisões citadas acima. Por exemplo, $q_1 = a/b$ e $q_2 = b/r_1$.

* Na terceira coluna x é calculo é da seguinte maneira: $x_j = x_{j-2} - q_j * x_{j-1}$.

* Na quarta coluna y o calculo é da semelhante ao anterior, a diferença é que se utiliza os valores de y: $y_j = y_{j-2} - q_j * y_{j-1}$. Vamos a um exemplo prático, para podermos pegar a lógica. Utilizando os valores $a = 1234$ e $b = 54$: (estamos utilizando "%"para representar o resto da divisão)

resto	$\Rightarrow$	resto
1234	$\Rightarrow$	1254
54	$\Rightarrow$	54
$1234\%54 = 46$	$\Rightarrow$	46
$54\%46 = 8$	$\Rightarrow$	8
$46\%8 = 6$	$\Rightarrow$	6
$8\%6 = 2$	$\Rightarrow$	2
$6\%2 = 0$	$\Rightarrow$	0

Isso foi a parte do *resto*. Agora vamos para a segunda parte da tabela *quociente*.

quociente	$\Rightarrow$	quociente
*	$\Rightarrow$	*
*	$\Rightarrow$	*
$1234/54 = 22$	$\Rightarrow$	22
$54/46 = 1$	$\Rightarrow$	1
$46/8 = 5$	$\Rightarrow$	5
$86 = 1$	$\Rightarrow$	1
$6/2 = 3$	$\Rightarrow$	3

Agora vamos para a terceira coluna x. Para isso iremos utilizar a fórmula
$x_j = x_{j-2}z - q_j * x_{j-1}$. Vamos fazer linha por linha.
1º Por convenção o $x_1 = 1$;
2º Da mesma forma $x_2 = 0$;
3º j = 3 $\Rightarrow x_3 = x_1 - q_3 * x_2 \Rightarrow x_3 = 1$ - 22*0 = 1
4ºj = 4 $\Rightarrow x_4 = x_2 - q_4 * x_3 \Rightarrow x_4 = 0$ - 1*1 = -1
5ºj = 5 $\Rightarrow x_5 = x_3 - q_5 * x_4 \Rightarrow x_5 = 1$ - 5*(-1) = 6
6º j = 6 $\Rightarrow x_6 = x_4 - q_6 * x_5 \Rightarrow x_6 =$ -1 - 1*6 = -7

Agora, por fim, vamos para a última coluna y. Para isso iremos utilizar a fórmula $y_j = y_{j-2}z - q_j * y_{j-1}$. Vamos fazer linha por linha.

1º Por convenção o $y_1 = 0$;
2º Da mesma forma $y_2 = 1$;
3º j = 3 $\Rightarrow y_3 = y_1 - q_3 * y_2 \Rightarrow y_3 = 0$ - 22*1 = -22
4ºj = 4 $\Rightarrow y_4 = y_2 - q_4 * y_3 \Rightarrow y_4 = 1$ - 1*(-22) = 23
5ºj = 5 $\Rightarrow y_5 = y_3 - q_5 * y_4 \Rightarrow y_5 =$ -22 - 5*23 = -137
6º j = 6 $\Rightarrow y_6 = y_4 - q_6 * y_5 \Rightarrow y_6 = 23$ - 1*(-132) = 160

Vamos juntar tudo em uma tabela, e ela ficara da forma:

restos	quociente	x	y
1234	*	1	0
54	*	0	1
46	22	1	-22
8	1	-1	23
6	5	6	-137
2	1	-7	160
0	3	*	*

Agora para encontrar o mdc entre 1234 e 54, só teremos que utilizar a fórmula (4), lembrando que os valores de α e β são os últimos valores de x e y, respectivamente. Assim, teremos α = -7 e β = 160

$$a\alpha + b\beta = d \Rightarrow 1234*(-7) + 54*(160) = 2$$
$$\text{mdc}(1234,54) = 2$$

2.2 Decodificação: d

Essa seção é exclusiva para explicar como encontrar o d da chave de decodificação, seção (3). Aqui utiliza-se o teorema de Bezout e outro teorema.

Teorema:: *se a e b são inteiros primos entre si e b > 1, então um inverso a módulo b existe.* Demonstração: *Como eles são primo entre si, então mdc(a,b) = 1*
*Pelo teorema 1, temos que mdc(a,b) = a*s + b*t*
*Então, a*s + b*t ≡ 1 (mod b)*
*Como b ≡ 0 ⇒ b*t ≡ 0*t logo b*t ≡ 0 (mod b)*
*Daí, a*s + b*t ≡ 1 (mod b) ⇒ a*s + 0 ≡ 1 (mod b) ⇒ a*s ≡ 1 (mod b)*
Então s é o inverso de a módulo b

Vamos a um exemplo prático: sabendo que o mdc(3,7) = 1, então o inverso $\bar{a}$ existe. *então 3*$\bar{a}$ ≡ 1 (mod 7)*

Pelo teorema de Euclides, temos a = bg + s
*então teremos o seguinte: 7 = 3*2 + 1 → 7 - 3*2 = 1*
isso mostra que o inverso -2 é um inverso de 3 módulo 7.

Agora vamos ao d da decodificação. Sabemos que no algoritmo de Euclides estendido o aα + bβ = mdc(a,b) = d. Os valores de α e β são os valores de x e y da última interação. No RSA, o d é o inverso de b em a. Ou seja, mdc(a,b) = 1. Então, aα + bβ = 1, nesse caso especifico, queremos encontrar aα = 1 - bβ, como β pode ser negativo ou positivo, vamos ocultar o sinal, então estamos procurando um d da forma aα = 1 + bβ, como d não será maior que a, teremos a seguinte equação:

aα = 1 + bβ, sendo α = 1 e β = f, logo
*a = 1 + b*d é a equação do lema de Euclides*
Então o d *é y do algoritmo de Euclides estendido.*
Lembrando que é o y da última interação e que ele pode
ser negativo, caso isso ocorra, deve ser tratado de outra forma
(explicaremos nas próximas seções).

2.3 Revisão da Literatura

Na construção desse trabalho foram encontrados alguns trabalhos com o tema abordado. Iremos citar os trabalhos mais relevantes e que de alguma forma tenha contribuído na produção do presente trabalho.

Vale ressaltar que a base principal desse trabalho foi o livro *Números Inteiros e Criptografia RSA* seu autor é *S.C. Coutinho*. "A palavra criptografia evoca imagens de agentes secretos sorrateiramente transferindo informação sigilosa as nações rivais", essa frase é do livro citado acima. Ela é bem interessante pois é bem verdadeira. Quase não se sabe sobre a criptografia, onde é utilizada, como é utilizada ou até mesmo implementada, dos vários tipos de criptografias que se tem conhecimento. Muitos já ouviram falar sobre inteligencia artificial, mas poucos ouviram falar da criptografia e poucos sabem que ela está presente em nosso. Isso se deve ao pouco conteúdo ou a pouca divulgação.

A internet é o meio de difusão da informação mais utilizado atualmente, e seu acesso por parte de usuários, autorizados ou não, independente do tipo de aplicação, gera diversas transmissões de dados, que além de públicos podem precisar ser confidenciais como nome de usuário e senha. O problema é que o interesse nas informações de terceiros por parte de pessoas mal-intencionadas ou não autorizadas cresceu muito, e os ataques às aplicações disponibilizadas na Internet se tornaram frequentes (Anjos, 2016). Um dos pontos abordado nesse trabalho é justamente a segurança que os usuários presam em ter e manter. Hoje em dia com tanta modernidade podemos abrir uma conta sem sair de casa, podemos olhar nosso saldo bancário pelo nosso celular ou computador, entre outras

coisas. Mas isso tudo é possível graças a criptografia, que até hoje é e continuará sendo segura.

Primos Gêmeos são todos os pares de números primos cuja distância entre eles é de duas unidades, como por exemplo os pares: (3,5), (5,7) e (11,13), (41,43) e (71,73)(Kilhian, 2018). Uma informação irrelevante para esse trabalho, mas que é bem curioso, pois como não sabemos muito sobre os números primos, essa pequena informação veio para mostrar mais da beleza desses números que são tão importantes para esse trabalho.

3 METODOLOGIA

3.1 Pré-Codificação

O pré-codificação nada mais é do que mascarar a mensagem original, ou melhor, substituir a mensagem original por números. Nesse caso, só iremos trabalhar com mensagens sem acentos e sem números, para não causar erro na hora da pré-codificação. Cada programador cria sua própria tabela de pré-codificação, alguns utilizam a tabela ASCII, não é o caso desse trabalho.
Na pré-codificação utilizamos a seguinte tabela de convenção:

A	B	C	D	E	F	G	H	I	J	K	L	M
11	12	13	14	15	16	17	18	19	21	22	23	24
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
25	26	27	28	29	31	32	33	34	35	36	37	38

Tabela 1: Tabela de pré-codificação

Para o espaço entre as palavras utilizamos *99*.

Vamos pré-codificar a seguinte frase:

A Matemática é o Alfabeto = 11992411321524113219131199159926991123161112153226

(7)

Observe que na tabela de convenção não existe número com "0"no final. O motivo é que precisaremos dividir a mensagem pré-codificada em blocos menores, para não precisarmos tratar desse caso em especial optamos em não colocar valores como "10","20", "30"e etc na tabela de convenção. É importante ressaltar que seguimos o padrão "cada letra equivale a um número com dois dígitos", isso para não confundir, por exemplo, se a letra A=2, B=3 e L=23, quando ocorrer um bloco 23, não saberíamos se seria AB ou apenas um L.

Como já foi citado, o método RSA necessita de dos primos, vamos utilizar os primos 13 e 17. O n é o produto dos primos escolhidos, logo

$$\text{n} = \text{pq} \Rightarrow \text{n} = 13*17 \Rightarrow \text{n} = 221.$$

(8)

Depois da mensagem pré-codificada e os primos escolhidos, a outra fase da pré-codificação é quebrar a mensagem pré-codificada em blocos menores que o n. Como nosso $n = 221$, os blocos não podem ser maior ou igual a esse número. A quebra será feita da seguinte maneira:

$$b = 119 - 92 - 41 - 132 - 152 - 41 - 132 - 191 - 31 - 199 - 159 - 92 - 69 - 91 - 123 - 161 - 112 - 153 - 22 - 6 \tag{9}$$

o b representa o bloco completo e o b_i representa seus elementos, por exemplo: $b_5 = 152$.

3.2 Codificando

Agora vamos para a parte da codificação. Nessa parte, iremos "encontrar"a chave de codificação. Essa chave é composta por dois elementos.

O primeiro elemento é o n, que como já foi citado é o produto dos primos p=13 e q=17. Para encontrarmos o segundo elemento da chave, precisamos calcular $\phi(n)$, que nada mais é do que calcular a fórmula:

$$\phi(n) = (\text{p - 1})(\text{q - 1}) \tag{10}$$

Como já conhecemos o p=13 e q=17, podemos calcular o $\phi(n)$:

$$\phi(n) = (\text{p - 1})(\text{q - 1}) \Rightarrow (13 - 1)(17 - 1) = 192$$

Depois de calculado o $\phi(n)$, precisamos encontrar um número inteiro positivo que seja inversível módulo $\phi(n)$, ou seja, que

$$\text{mdc}(e, \phi(n)) = 1 \tag{11}$$

Como já conhecemos o $\phi(n)$, vamos calcular o e:

$$\text{mdc}(e, \phi(n)) \Rightarrow \text{mdc}(5, 192) = 1$$

ou seja, 5 e 192 são primos entre sim, pois o seu máximo divisor comum é 1.

Depois de encontrado a chave de codificação (221, 5), teremos que codificar o bloco b (9). Os blocos já codificados serão chamados de C(b). Para calcular C(b) teremos que usar o seguinte:

$$\text{C}(b) = \text{resto da divisão de } b^e \text{ por } n \tag{12}$$

O C(b) é a forma reduzida de b^e módulo n.

Vamos relembrar os dados que já temos até agora: $p = 13$, $q = 17$, $n = 221$, $e = 5$ e o $\phi(n) = 192$. Para codificar os bloquinhos temos que utilizar a fórmula (12). Então, vamos criptografar o primeiro elemento do bloco que é $b_1 = 119$ (para calcular o C(b) foi criado uma rotina computacional, que será explicado nas próximas subseções):

$$\text{C}(1) = 119^5 \bmod 221 = 136$$

Fazendo para todos os bloquinhos, temos a seguinte sequência:

C(b) = 136 - 79 - 45 - 149 - 16 - 45 - 149 - 55 - 148 - 88 - 126 - 79 - 205 - 143 - 106 - 213 - 125 - 17 - 133 - 41

3.3 Decodificação

Na decodificação precisamos da chave de decodificação. Essa chave é composta por dois números, o nosso conhecido n e o d. Esse d é o inverso do e em $\phi(n)$. O resultado dessa decodificação será chamado de D(a). Vale salientar que o a_i é os valores de C(b), por exemplo: C(b_1) = 136 = a_1 ou seja, D(a_1) = D(136). Voltando para a decodificação, para decodificar D(a) usaremos o mesmo método que utilizamos na codificação, que é calcular forma reduzida. Então,

$$\text{D}(a) = \text{resto da divisão de } a^d \text{ por } n \tag{13}$$

A mesma rotina para calcular fórmula reduzida que utilizamos na codificação, também utilizaremos aqui. A diferença é que vamos utilizar o D(a).

Então, para encontrarmos o nosso d, utilizaremos o principio que d é o inverso de e em $\phi(n)$, então

$$\begin{gathered} mdc(\phi(n),e) = 1, \text{ então} \\ \phi(n) = ed + 1 \\ \phi(n) - ed = 1 \end{gathered}$$

(14)

Obs.: lembrando que o d pode ser negativo, mas como estamos trabalhando com números positivos, devemos utilizar a seguinte fórmula para esse caso especifico:

$$d_{novo} = \phi(n) + d_{antigo}$$

(15)

Utilizando nossos dados, temos que $d = 77$. Agora utilizando nossa rotina computacional, vamos encontrar o D(a):

D(a) = 119 - 92 - 41 - 132 - 191 - 31 - 199 - 159 - 92 - 69 - 91 - 123 - 161 - 112 - 153 - 22 - 6

Agora comparando nosso D(a) com o b_i, vemos que ambos são iguais, ou seja, a decodificação foi um sucesso. Foi criada uma rotina computacional, na qual iremos "revelar"a mensagem original. Vale ressaltar que não fizemos isso para a mensagem codificada, pelo fato dos valores encontrados não serem compatíveis com a tabela de convenção. Por exemplo, bloquinhos 1 e 2:

C(1) = 136
C(2) = 79
Como cada letra da nossa tabela têm dois dígitos,
então teremos os seguintes números: 13 - 67 - 94
Como podemos observar, na tabele não consta valores acima de 38,
e por esse motivo não iremos usar a rotina para esses bloquinhos,
pois a mensagem seria "errada"de acordo com a tabela.

D(a) = 1199241132191311991599269911231611121532226 ⇒
A MATEMATICA E O ALFABETO

3.4 Algoritmos

Essa subseção é exclusiva para as rotinas computacionais que foram implementadas para o método RSA. Antes de entrar nas rotinas, vamos explicar alguns comando que utilizamos com muita frequência.

While: é um laço de repetição que executa uma instrução até que a condição seja falsa;

For: é um laço de repetição, porém ao contrário do While, ele executa uma instrução x vezes, ou seja, a quantidade de execuções é determinada;

IF - ELSEIF - ELSE: é uma estrutura de condição, eles verificam se determinado dado é verdadeiro ou falso;

Eval: avalia uma expressão;

Str2num: transforma caracteres ou strings em números;

Int2str: ao contrário do str2num, ele transforma números em caracteres;

Break: encerra algum laço de repetição, geralmente é utilizado dentro de alguma estrutura de condição;

Floor: fornece o inteiro de um número;

Min: retorna os elementos mínimos de uma matriz;

Sum: retorna a soma das colunas de uma matriz;

Abs: número absoluto;

Length: essa função retorna o comprimento da maior dimensão da matriz;

Linspace: gera n pontos com espaçamento ou um vetor com 100 pontos;

Unidrnd: gera números aleatórios.

As rotinas então divididas em duas: a rotina de codificação e a de decodificação. Iniciaremos com as rotinas de codificação:

Codificação

Pré-codificação: foi criada a rotina de pré-codificação, nessa rotina o usuário entra com a mensagem a ser criptografada, seguindo as regras da tabela, ou seja, a mensagem não pode ter acentos gráficos e nem pontuação, e a mensagem tem que ser escrita toda em letras maiúsculas. Na figura (16), é a etapa da pré-codificação. Para criar a tabela de pré-codificação foi utilizado a tabela (1), encontra-se no apêndice. Foi criada uma variável l que recebe o comprimento da maior dimensão da matriz, utilizamos a função "length"para isso. Depois criamos o vetor *codigo* para receber as pré-codificações da mensagem original. Essa pré-codificação é feita dentro de um laço de repetição, esse laço vai até o comprimento da mensagem. Utilizamos a função "eval"para avaliar/trocar as

letras por números.

```
frase = mensagem;
l = length(frase);
codigo =[];

    for i=1:l
        codigo = [codigo eval(frase(i))];
    end
Codigo = codigo;
```
(16)

Escolha dos primos: a escolha dos números primos é aleatória, a única coisa que o usuário digita é a quantidade de dígitos que ele deseja que os primos tenham, a variável *NumCasas* é quem recebe esse valor. A primeira coisa feita na função, figura (17), foi criar um vetor *primos* para que guarde os números primos encontrados. A variável *vaux* recebe uma sequência de números uniformemente espaçados. No primeiro comando de repetição *while*, esse bloco é executado até que o vetor *primos* contenha os dois primos. A variável *cpr* recebe um vetor com NumCasas posições de números aleatórios de 1 a 9. O x recebe o produto entre *vaux* e *cpr*. O segundo laço de repetição *for* é executado até o teto da raiz de $x + 1$. A primeira condição *if* e a segunda *elseif* é uma exceção para considerar o 3 como primo e descartar o 1 da lista dos primos. A terceira condição dentro do *for* verifica se o resto da divisão de x por i é igual a 0, quando essa condição for satisfeita v recebe +1. Na ultima condição *if* é verificado se o valor de v é igual a 0, caso isso seja satisfeito *primos* recebe esse valor.

```
primos = [];
vaux = 10.^linspace(NumCasas-1,0,NumCasas);

    while length(primos)<2
        cpr= unidrnd(9,NumCasas,1);
        x = vaux*cpr;
        for i=2:ceil(x^.5)+1
            if x == 3;
                v = 0;
            elseif x == 1;
                v = 1;
            elseif mod(x,i) == 0;
                v = v+1;
            end
         end
            if v == 0
                primos = [primos x];
            end
            v = 0;
```
(17)

Na figura (18) encontramos os valores de n usando a equação (8) e o valor de *phi* usando a equação (10). E v recebe o valor de *phi*. No laço de repetição é encontrado o valor de e, que deve ser um número inteiro positivo e que seja primo com *phi*, equação (11). Utilizamos a função do Matlab *gcd* do Matlab que calcula o mdc entre dois números.

```
p = primos(1);
q = primos(2);
n = p*q;
phi = (p-1)*(q-1);
v = phi;
    for i=2:v
        if gcd(v,i) == 1
            e = i;
            break
        end
    end
```

(18)

Blocos: essa é a etapa da quebra da mensagem pré-codificado em bloquinhos. A mensagem pré-codificada é um conjunto de caracteres e não um número gigante. Então, a primeira coisa feita foi criar 5 variáveis: a primeira foi um auxiliar (*aux2*) que recebe a mensagem pré-codificada; a segunda (m) recebe o comprimento da mensagem (*msg*), a terceira (p) converte o n em carácter e recebe o seu comprimento; a quarta e quinta são variáveis que serão utilizadas como contagem, figura(19).

```
aux2 = msg
m = length(msg);
p = length(int2str(n));
v = 1;
s = 1;
```

(19)

Na figura (20), foi criada uma variável d para converter *aux2* em números e poder dividir em bloquinhos menores. Ou seja, d recebe um bloquinho com p posições, isso em cada execução do laço de repetição. Na primeira condição é verificada se o bloquinho é menor que o n, caso isso seja verdade, a variável w recebe esse valor e s recebe o seu valor mais o valor d p; caso essa condição seja falsa, ele vai para o *else*, onde o f recebe *p-1*, subtrai 1 de p, e o w recebe um bloquinho com f posições e s recebe seu valor mais o valor de f.

```
d = str2num(aux2(1:p));

if d < n
   w = d;
   s = s+p;

else
    f = p-1;
    w = str2num(aux2(1:f));
    s = s+f;
end
```

(20)

Na figura (21), o *aux2* é alterado, ele deixa de ser a mensagem original completa de 1 até m e passa a ser de s até m (retiram-se os bloquinhos em cada interação), em

cada interação, o valor de *aux2* é substituído.Foi criado um vetor *h(v)* para agrupar os bloquinhos w; v é uma variável de controle, ela indica quantas linhas tem o vetor *h(v)*; k é a variável que verifica o tamanho do *aux2*; a variável *(*aux) recebe o valor de *aux2* convertido em inteiro; na última condição verifica se o k é menor ou igual que o p e se ao mesmo tempo se o valor de *aux* é menor que o n, caso essas condições sejam satisfeitas, o laço de repetição é encerrado (*break*).

```
aux2 = msg(s:m);
h(v) = w;
v = v+1;
k = length(aux2);
aux = str2num(aux2);
    if k <= p & aux < n
        w = str2num(aux2);
        h(v) = w;
        break
    end
```
(21)

Algoritmo Euclidiano Estendido: na figura (22) foi a implementação do Algoritmo euclidiano estendido. Os dados a ser utilizado são os valores de $a = \phi(n)$ e do $b = e$. Nesse método precisa apenas dos números inteiros, por isso foi criado a variável *Q(i)* para receber apenas a parte inteira da divisão de *a/b*, para isso foi utilizado a função do Matlab *floor*; a variável c recebe o valor de b e o novo valor de b é o resto da divisão de a pelo b anterior. Feito isso, a variável de incremento i soma mais 1 ao seu valor antigo.

```
while b~=0

    Q(i) = floor(a/b);
    c = b;
    b = mod(a,b);
    a = c;
    i=i+1;
```
(22)

Nessa parte, figura (23), foi criado dois vetores de zeros, x e y, e que na posição 1 e 2 dos vetores são atribuídos os valores "1 e 0"e "0 e 1", respectivamente.

```
x(i)= zeros;
x(1)=1;
x(2)=0;

y(i)= zeros;
y(1)=0;
y(2)=1;
```
(23)

Nessa parte do código, figura (24), é o preenchimento dos vetores criados, eles só são preenchidos da posição 3 em diante, por isso foi criado a condição. O preenchimento desses vetores é da forma da equação (6).

```
if i>2
    x(i)=x(i-2)-x(i-1)*Q(i-2);
    y(i)=y(i-2)-y(i-1)*Q(i-2);
end
```
(24)

Na figura (25) mostra como encontrar o d da chave de decodificação. Sabe-se que $\phi(n) - ed = 1$, como os valores de $\phi(n)$ e e são conhecidos dentro da rotina, temos que encontrar um valor d que multiplicado com e e somado com 1 seja igual ao valor de $\phi(n)$. Esse valor já é conhecido no algoritmo euclidiano, ele nada mais é do que o β. Por isso foi calculado as dimensões do vetor x (linha e coluna) e como resultado ele mostra o último valor do vetor y. Esse detalhe foi explicado na seção (2) em uma parte exclusiva para o d.

```
[l,c] = size(x);
Euclides = y(c)
```
(25)

Caso o d seja negativo, figura (26), deve-se fazer o a substituição do d, o novo s recebe a subtração do $\phi(n)$ pelo d antigo.

```
d = EuEstendido(phi,e);
    if d < 0
        d = phi + d;
    end
```
(26)

Agora é a implementação do algoritmo de exponenciação, ou seja, da forma reduzida do tipo a^e *módulo* n. Para essa implementação foi utilizado o pseudocódigo de S.C. Coutinho (2000). Nessa figura (27) é simplesmente a primeira condição do método, que quando o expoente for zero, o método deve parar e informar que P=1 e P é a forma reduzida.

Nessa segunda parte, figura (28), é a segunda condição do método, que verificado se o resto da divisão do expoente por 2 é diferente de zero, caso seja verdade, ele é executado. O primeiro passo é fazer é calcular o resto da divisão de $P*A$, onde A são os bloquinhos que foram encontrado na rotina dos blocos, os resultados desses cálculos é atribuído para

```
if E == 0;
    P;
    break;
```
(27)

o novo P. O segundo passo é calcular o novo expoente, então, E recebe o valor (E-1)/2. O terceiro passo é calcular o novo valor de A que é o resto da divisão de A^2 por n.

```
elseif mod(E,2)~=0
    P = mod(P.*A,n);
    E = (E-1)/2;
    A = mod(A.*A,n);
```
(28)

Essa é a última etapa do método, figura (29), caso nenhuma das condições anteriores sejam atendidas, essa condição é executada. Simplesmente os valores de E e A são alterados. E recebe a metade do E antigo e A recebe o resto da divisão de A^2 por n.

```
else
    E = E/2;
    A = mod(A.^2,n);
end
```
(29)

Decodificação

Pré-codificação: para decodificar o texto é necessário ter a chave de decodificação (n,d). Como o receptor não sabe quais são os primos e o valor do e, foi criado um arquivo que recebe a mensagem codificada e os valores de n e d, figura (30).

```
arq = fopen('Enviar.txt','w');
fprintf(arq,'%d\n%d\n %d\n',a,n,d);
fclose(arq);
```
(30)

Para a decodificação de fato foi criado um vetor a que recebe a codificação (a mensagem codificada). Onde p a recebe os valores da codificação de 1 até k-2 (k é o tamanho do vetor, n é o elemento do vetor k-1 e d é o elemento do vetor k. Chamamos a função *Ex* que utilizamos na codificação, depois chamamos a função *Decifra*, figura (31).

```
a = [];
arquivo = fopen('Enviar.txt');
codigo = fscanf(arquivo,'%d');
fclose(arquivo);
k = length(codigo);
a = codigo(1:k-2);
n = codigo(k-1);
d = codigo(k);
Dd = Ex(a,d,n);
mensagem = Decifra(Dd);
Frase = C_mensagem(mensagem,1);
```
(31)

Essa é a etapa da pré-codificação ao contrário, figura (32), onde os números são substituídos por letras. A primeira coisa a fazer verificar o tamanho da mensagem codificada, depois criar um laço de repetição e dentro de laço juntar todos os bloquinhos em um único vetor. A variável *cd1* está recebendo esses bloquinhos e os transformando em um vetor de carácter.

```
for i=1:length(cod)
    cd1 = [cd1 int2str(cod(i))];
end
```
(32)

Nesse segundo laço, figura (33), a variável *letra* recebe números com dois dígitos, o algoritmo verifica os índices dos elementos e faz a distribuição (um número com dois dígitos tem um índice impar e outro par, essa foi a lógica), pois cada letra é representada por números com dois dígitos. Na terceira linha contém muita informação, então iremos aos pouco. Nessa parte do código *(ones(numletras,1)*letra)* cria-se uma matriz com *"x = numletras"* linhas e 2 colunas, depois multiplica essa matriz de 1 pelo *letra*. Depois disso subtrai a matriz (mesma tabela utilizada na pré-codificação, só trocando a ordem)

Mat pelo resultado anterior *(Mat-ones(numletras,1)*letra)*. Feito isso, soma-se as linhas *sum(abs(Mat-ones(numletras,1)*letra),2)*. Depois verifica o menor elemento dessa coluna, utilizando a função *sum*. A variável *v* recebe o valor mínimo (nesse caso é o zero) e *coord* recebe a posição do elemento (em que linha ele está). Foi criado um vetor *msg* que recebe as letras (*Alfabeto* é vetor que contém as letras e seus respectivos números está na matriz *Mat*).

```
for i=1:length(cd1)/2
    letra = [eval(cd1(2*i-1)) eval(cd1(2*i))];
    [v,coord] = min(sum(abs(Mat-ones(numletras,1)*letra),2));
    msg = [msg Alfabeto(coord)];
end
```
(33)

Para mostrar a mensagem na tela de forma organizada e legível foi criada uma rotina que ler o texto do arquivo e exibe seu conteúdo de forma legível, figura (34). No laço de repetição é executado até *bt* que é o tamanho da mensagem, o *tipo* é se a mensagem vem de um arquivo. Depois é só substituir os *e* por espaços e o *f* por linha.

```
for i =1:nt
    if tipo == 0
       if texto(i)== ' ';
            texto(i) ='e';
       end
    else
        if texto(i)== 'e';
            texto(i) =' ';
        end
        if texto(i)== 'f';
            texto(i) =' ';
            nl=nl+1;
            fl(nl) = i;
        end
    end
end
```
(34)

Caso a mensagem esteja no prompt de comando, a variável *tipo* recebe 1. A variável *S* recebe uma matriz de *string*. Caso o *tipo* seja 0, a variável *frase* recebe *texto* a ser exibido na tela e depois cria-se um vetor *textos*. caso a mensagem esteja no prompt de comando, *fl* = [*fl* 1], declara-se que *frase* como vetor. O laço de repetição *for* é executado até o tamanho da mensagem original mais um e *S* = S(k-1) = texto(fl(k-1):fl(k)), cada elemento da mensagem original é adicionada no *S*, figura(35).

```
if tipo == 1
    S = string({});
end
if tipo == 0
    frase = texto;
    textos =[];
else
    fl = [1 fl];
    frase =[];
        for k=2:nl+1
            S(k-1) = texto(fl(k-1):fl(k));
        end
   texto = S'
end
```

(35)

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

O objetivo do trabalho foi alcançado, que era implementar o método RSA em um ambiente algébrico. Agora, vamos apresentar os resultados que foram obtidos nesse trabalho. Vale ressaltar que os materiais de apoio como os códigos implementados estão no Apêndice.

Decodificando um texto

O texto escolhido para ser codificado é a letra de uma música, *Sabes* banda *Reik*.

Sabes
No pido nada mas
Que estar entre tus brazos
Y huir de todo el mal
Que a todo he renunciado
Por estar junto a ti
Sabes
No dejo de pensar
Que estoy enamorado
Te quiero confesar
Que soy solo un esclavo
Que no sabe vivir sin ti
Cuando llegaste tu te metiste en mi ser
Encendiste la luz
Me llenaste de fe
Tanto tiempo busque
Pero al fin te encontré
Tan perfecta como te imagine
Como aguja en un pajar
Te busque sin cesar
Como huella en el mar tan difícil de hallar
Tanto tiempo busque pero al fin te encontré
Tan perfecta
Como te imagine
Reik

Como já citado no capítulo anterior, a mensagem devem estar toda em maiúsculas, onde tiver espaço seja substituído pela letra e e onde tiver que pular uma linha sera utilizado a letra d, o texto ficaria dessa forma:

eSABESefNOePIDOeNADAeMASefQUEeESTAReENTREeTUSeBRAZOSefYeHUIRe

DEeTODOeELeMALefQUEeAeTODOeHEeRENUNCIADOefPOReESTAReJUNTOeAe TIefSABES efNOeDEJOeDEePENSARefQUEeESTOYeENAMORADOefTEeQUIEROe CONFESARefQUE eSOYeSOLOeUNeESCLAVOefQUEeNOeSABEeVIVIReSINeTIe fCUANDOeLLEGASTEe TUefTEeMETISTEeENeMIeSERefENCENDISTEeLAeLUZef-MEeLLENASTEeDEeFEefTANTOe TIEMPOeBUSQUEefPEROeALeFINeTEeENCON-TREefTANePERFECTAefCOMOeTEeIMAGINEef COMOeAGUJAeENeUNePAJARef-TEeBUSQUEeSINeCESARefCOMOeHUELLAeENeELeMARef TANeDIFICILeDEeHAL-LARefTANTOeTIEMPOeBUSQUEefPEROeALeFINeTEeENCONTREef TANePERFEC-TAefCOMOeTEeIMAGINEefREIKef

A mensagem depois de codificada é:
Codigo = ’993111121531999825269927191426992511141199241131999828331599
153132112999152532291599323331991229113826319998379918331929991415993 2
261426991523992411239998283315991199322614269918159929152533251319111 4
269998272629991531321129992133253226991199321999983111121531999825269 9
141521269914159927152531112999982833159915313226379915251124262911142 6
999832159928331915292699132625161531112999982833159931263799312623269 9
332599153113231134269998283315992526993111121599341934192999311925993 2
199998133311251426992323151711313215993233999832159924153219313215991 5
259924199931152999981525131525141931321599231199233338999824159923231 5
251131321599141599161599983211253226993219152427269912333128331599982 7
152926991123991619259932159915251326253229159998321125992715291615133 2
119998132624269932159919241117192515999813262426991117332111991525993 3
259927112111299998321599123331283315993119259913153111299998132624269 9
183315232311991525991523992411299998321125991419161913192399141599181 1
232311299998321125322699321915242726991233312833159999827152926991123 9 9
161925993215991525132625322915999832112599271529161513321199981326242 6
993215991924111719251599982915192299 98’

A mensagem codificada para a chave (n,e) = (720553,5), com primos de 3 dígitos é:
179080 264834 96576 531092 233053 195099 581720 507588 344032 165482 670905 677137 409856 529939 185138 48360 240455 26749 353606 577631 114322 51250 692902 109037 77345 18416 46017 366674 131300 41228 109037 277465 54293 590057 552300 151764 498699 464477 149048 48118 68170 258285 682986 211363 191637 599475 606536 388622 538312 631367 317343 253591 697198 366674 344361 155335 682484 569669 352668 100778 467771 51416 510130 77042 90477 597712 48118 208721 558367 614359 557038 715546 719924 712310 262268 95918 165482 547856 658928 393663 372112 483864 179080 184943 567559 552461 280025 716961 583882 566308 605221 182705 81078 18416 581787 157762 309431 18416 52556 477663 107834 584610 365886 157762 474523 194443 224723 42266

613712 331703 491350 509541 240600 501721 394584 456853 520747 69095 492475 78044 540775 117477 691630 548621 78660 104734 257974 446322 304579 467771 69533 276542 39722 13009 249136 53482 104734 514455 440471 540775 403499 691630 704626 457538 122601 430689 299577 720315 104734 484195 165482 179080 184943 12190 48118 303451 487916 569637 441931 457538 586089 31274 48118 467771 534463 49338 258960 678734 289316 384517 48118 467771 114487 136417 705808 665732 484195 165482 248063 591243 124841 492861 184943 104734 257974 446322 304579 467771 69533 276542 39722 13009 249136 53482 104734 514455 440471 540775 79086 561636 720553 431309

A mensagem é decodificada com a chave (n,d) = (720553,431309) é:

"SABES "
"NO PIDO NADA MAS "
"QUE ESTAR ENTRE TUS BRAZOS "
"Y HUIR DE TODO EL MAL "
"QUE A TODO HE RENUNCIADO "
"POR ESTAR JUNTO A TI "
"SABES "
"NO DEJO DE PENSAR "
"QUE ESTOY ENAMORADO "
"TE QUIERO CONFESAR "
"QUE SOY SOLO UN ESCLAVO "
"QUE NO SABE VIVIR SIN TI "
"CUANDO LLEGASTE TU "
"TE METISTE EN MI SER "
"ENCENDISTE LA LUZ "
"ME LLENASTE DE FE "
"TANTO TIEMPO BUSQUE "
"PERO AL FIN TE ENCONTRE "
"TAN PERFECTA "
"COMO TE IMAGINE "
"COMO AGUJA EN UN PAJAR "
"TE BUSQUE SIN CESAR "
"COMO HUELLA EN EL MAR "
"TAN DIFICIL DE HALLAR "
"TANTO TIEMPO BUSQUE "
"PERO AL FIN TE ENCONTRE "
"TAN PERFECTA "
"COMO TE IMAGINE "
"REIK

5 CONCLUSÕES E TRABALHOS FUTUROS

O objetivo desse trabalho foi concluído e posso dizer que foi com sucesso. O método RSA foi implementado, testado e explicado detalhe por detalhe o que foi feito em cada etapa. Quando escutei sobre a criptografia e sua matemática por trás, confesso que foi um dos fatores que motivaram a escrever esse trabalho. Depois de ler e analisar tudo que foi descrito nesse trabalho, posso dizer que esse TCC foi muito produtivo e que me deu uma visão mais ampla sobre a criptografia e sua segurança, sobre os números primos, que pouco se sabe sobre eles, mas que nos fornece uma segurança inigualável. Sem esquecer das ferramentas computacionais, que foram nossas aliadas, sem elas, creio que não seria possível chegar até aqui e mostrar com detalhes tudo o que foi mostrado no trabalho.

Como esse assunto é muito interessante, pretendo em continuar minha pesquisa nessa área. Conhecer mais sobre os outros tipos de criptografia e que meu trabalho possa ajudar mais pessoas a entender esse método.

REFERÊNCIAS

[1] COUTINHO, S.C. **Números Inteiros e Criptografia RSA**. Rio de Janeiro: IMPA, 2014.

[2] ANJO, Leandro T.V. **Segurança da Informação na Programação com uso de Criptografia e Certificação Digital**. 2016. Universidade Federal de Fluminense, Niterói, 2016.

[3] SILVA, Alecio S. **Um Estudo sobre o Algoritmo de Euclides**. 2014. Universidade Federal de Campina Grande, Campina Grande, 2014.

[4] OLIVEIRA, Ivan S. et al Nome. **Ação Quântica: Manipulando a Informação oculta do mundo Quântico**, 33, 193, 6, maio, 2013.

[5] CAVALCANTI, George D.C. Introdução à Aritmética Modular. Pernambuco: CIn UFPE. Disponível em: $< https : //www.cin.ufpe.br/\ gdcc/matdis/aulas/aritmeticaModular_parte2.pdf >$. Acesso em: 7 de Abril de 2020.

[6] CORIGAN, Caroline. Os melhores aplicativos de mensagens privadas. AVG Secure VPN, 2018. Disponível em: <https://www.avg.com/pt/signal/secure-message-apps>. Acesso em: 5 de Fev 2020.

[7] KILHIAN, Kleber. Distância entre primos gêmeos que possuem algorismos da unidade iguais. 2018. Disponível em: <https://www.obaricentrodamente.com/2018/02/distancia-entre-primos-gemeos-com-algarismos-das-unidades-iguais.html>. Acesso em: 16 de Abril de 2020.

ANEXO A – ANEXOS E APÊNDICES 1

```
function valores = GeraPrimos(NumCasas)
v = 0;
b = 0;
primos = [];
vaux = 10.^linspace(NumCasas-1,0,NumCasas);
while length(primos)¡2
cpr= unidrnd(9,NumCasas,1);
x = vaux*cpr;
for i=2:ceil(x.^5)+1
if x == 3;
v = 0;
elseif x == 1;
v = 1;
elseif mod(x,i) == 0;
v = v+1;
end
end
if v == 0
primos = [primos x];
end
v = 0;
end
p = primos(1);
q = primos(2);
n = p*q;
phi = (p-1)*(q-1);
v = phi;
for i=2:v
if gcd(v,i) == 1
e = i;
break
end
end
valores = [e,n,phi];
end
```

```
function AlgEx = Ex(a,e,n)
A = a;
P = 1;
E = e;
while(1)
if E == 0;
P;
break;
elseif mod(E,2) =0
P = mod(P.*A,n);
E = (E-1)/2;
A = mod(A.*A,n);
else
E = E/2;
A = mod(A.*A,n);
end
end
AlgEx = P;
end
```

```
function Teste = blocos(msg,n)
aux2 = msg;
m = length(msg);
p = length(int2str(n));
v = 1;
s = 1;
while(true)
d = str2num(aux2(1:p));
if d ¡ n
w = d;
s = s+p;
else
f = p-1;
w = str2num(aux2(1:f));
s = s+f;
end
aux2 = msg(s:m);
h(v) = w;
v = v+1;
k = length(aux2);
aux = str2num(aux2);
if k ¡= p e aux ¡ n
w = str2num(aux2);
h(v) = w;
break
end
end
Teste = h();
end
```

```
function Euclides = EuEstendido(a,b)
A = a;
B = b;
i = 1;
while b =0
Q(i) = floor(a/b);
c = b;
b = mod(a,b);
a = c;
i=i+1;
x(i)= zeros;
x(1)=1;
x(2)=0;
y(i)= zeros;
y(1)=0;
y(2)=1;
if i¿2
x(i)=x(i-2)-x(i-1)*Q(i-2);
y(i)=y(i-2)-y(i-1)*Q(i-2);
end
end
(l,c) = size(x);
Euclides = y(c);
end
```

```
function [b, Dd] = Codificador(mensagem)
frase = mensagem;
l = length(frase);
**Aqui fica a próxima tabela, tabela de codificação
ccodigo =[];
for i=1:l
codigo = [codigo eval(frase(i))];
end
Codigo = codigo;
valores = GeraPrimos(numcasas);
e = valores(1);
n = valores(2);
phi = valores(3);
h = blocos(codigo,n);
v = [];
v = h();
a = Ex(v,e,n);
d = EuEstendido(phi,e);
if d ¡ 0
d = phi + d;
end
arq = fopen('Enviar.txt','w');
fprintf(arq,'%d%d%d',a,n,d);
fclose(arq);
CodCrip = a
end
```

```
**
ch = ['1' '2' '3' '4' '5' '6' '7' '8' '9' '0'];
A = [ch(1) ch(1)];
B = [ch(1) ch(2)];
C = [ch(1) ch(3)];
D = [ch(1) ch(4)];
E = [ch(1) ch(5)];
F = [ch(1) ch(6)];
G = [ch(1) ch(7)];
H = [ch(1) ch(8)];
I = [ch(1) ch(9)];
J = [ch(2) ch(1)];
K = [ch(2) ch(2)];
L = [ch(2) ch(3)];
M = [ch(2) ch(4)];
N = [ch(2) ch(5)];
O = [ch(2) ch(6)];
P = [ch(2) ch(7)];
Q = [ch(2) ch(8)];
R = [ch(2) ch(9)];
S = [ch(3) ch(1)];
T = [ch(3) ch(2)];
U = [ch(3) ch(3)];
V = [ch(3) ch(4)];
W = [ch(3) ch(5)];
X = [ch(3) ch(6)];
Y = [ch(3) ch(7)];
Z = [ch(3) ch(8)];
e = [ch(9) ch(9)];
```

```
function msg = decifra(codigo)
***Aqui fica a próxima tabela
(numletras, q) = size(Mat);
cd1 = [ ];
for i=1:length(codigo)
cd1 = [cd1 int2str(codigo(i))];
end
msg = [ ];
for i=1:length(cd1)/2
letra = [eval(cd1(2*i-1)) eval(cd1(2*i))];
(v,coord) = min(sum(abs(Mat-ones(numletras,1)*letra),2));
msg = (msg Alfabeto(coord));
end
msg
end
```

```
***
Alfabeto = ['A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J' 'K' 'L' 'M' 'N' 'O'
'P' 'Q' 'R' 'S' 'T' 'U' 'V' 'W' 'X' 'Y' 'Z' 'e'];
Mat =[1 1 ;
1 2 ;
1 3 ;
1 4 ;
1 5 ;
1 6 ;
1 7 ;
1 8 ;
1 9 ;
2 1 ;
2 2 ;
2 3 ;
2 4 ;
2 5 ;
2 6 ;
2 7 ;
2 8 ;
2 9 ;
3 1 ;
3 2 ;
3 3 ;
3 4 ;
3 5 ;
3 6 ;
3 7 ;
3 8 ;
9 9 );
```

```
function Frase = Decodifica(a,d,n )
a = [];
arquivo = fopen('Enviar.txt');
codigo = fscanf(arquivo,'%d');
fclose(arquivo);
k = length(codigo);
a = codigo(1:k-2);
n = codigo(k-1);
d = codigo(k);
Dd = Ex(a,d,n);
mensagem = Decifra(Dd);
Frase = Cmensagem(mensagem,1);
end
```

```
function texto = Cmensagem(texto,tipo)
nt=length(texto);
fl = 0;
nl = 0;
for i =1:nt
if tipo == 0
if texto(i)== ' ';
texto(i) ='e';
end
else
if texto(i)== 'e';
texto(i) =' ';
end
if texto(i)== 'f';
texto(i) =' ';
nl=nl+1;
fl(nl) = i;
end
end
end
if tipo == 1
S = string();
end
if tipo == 0
frase = texto;
textos =[];
else
fl = [1 fl];
frase =[];
for k=2:nl+1
S(k-1) = texto(fl(k-1):fl(k));
end
texto = S';
end
end
```

Listas dos algorítimos

1º Gera os números primos

2º Algoritmo exponencial ou forma reduzida

3º Algoritmo que quebra em bloquinhos
4º Algoritmo de Euclides estendido
5º Algoritmo que codificador
6º Continuação do algoritmo 5, tabela de pré-codificação
7º Algoritmo que decifra a mensagem
8º Continuação do algoritmo 7
9º Algoritmo da decodificação
10º Algoritmo que imprime a mensagem na tela