Criptografia RSA gaussiana

Luis Antonio Coêlho

Trabalho de Conclusão de Curso - apresentado à
Faculdade de Tecnologia da
Universidade Estadual de Campinas

Orientadora: **Profa. Dra. Juliana Bueno**

3 de junho de 2017

Agradecimentos

Agradeço à todos que me apoiaram no decorrer deste projeto, desde de minha orientadora Profa. Dra. Juliana Bueno até você, meu caro leitor.

Prefácio

O presente artigo expõe o resultado da pesquisa para TCC em razão da viabilidade algoritmo de criptografia RSA gaussiano. Para isso veremos ao longo dos capítulos desta obra o algoritmo de criptografia RSA e a teoria de números envolvida por ele, passando pelo Teorema de Fermat e pelo Teorema Chinês do Resto. Veremos também o que já conhecido no campo dos números inteiros e primos gaussianos e como isso irá influenciar em nosso resultado final.

Sumário

Capítulo 1

Introdução

O sigilo sempre foi uma arma explorada pelos seres humanos para vencer certas batalhas, e até mesmo que na cotidiana missão de se comunicar. Foi a partir dessa necessidade que se criou a *criptografia*, nome dado ao conjunto de técnicas usadas para se comunicar em códigos. Seu objetivo é garantir que apenas os envolvidos na comunicação possam compreender a mensagem codificada (ou criptogtafada), garantindo que terceiros não saibam o que foi conversado.

Para compreender como funciona o processo de codificação e decodificação faz-se necessário o uso de uma série de termos técnicos, e para fins pedagógicos iremos introduzir tais conceitos apresentando um dos primeiros algoritmos criptográficos que se tem conhecimento, a criptografia de César.Para mais detalhes sobre o tema, veja [Cou07].

A chamada *criptografia de César*, criada pelo imperador romano César Augusto, consistia em substituir cada letra da mensagem por outra que estivesse a três posições a frente, como, por exemplo, a letra **A** ser substituída pela letra **D**.

Uma forma muito natural de generalizar o algoritmo de César é fazer a troca de cada letra da mensagem por outra que venha em uma posição qualquer fixada. A chamada *criptografia de substituição monoalfabética* consite em substituir cada letra por outra que ocupe n posições a sua frente, sendo que o número n é conhecido apenas pelo emissor e pelo receptor da mensagem. O número n é a *chave criptográfica*. Para decifrar a mensagem, precisamos substituir as letras que formam a mensagem criptografada pelas letras que estão n posições antes.

O algoritmo monoalfabético tem a característica indesejada de ser de fácil decodificação, pois possui apenas 26 chaves possíveis, e isso faz com

que no máximo em 26 tentativas o código seja decifrado. Com o intuito de dificultar a quebra do código monoalfabético foram propostas as *cifras de substituição polialfabéticas* em que a chave criptográfica passa a ser uma *palavra* ao invés de um número. A ideia é usar as posições ocupadas pelas letras da chave para determinar o número de posições que devemos avançar para obter a posição da letra encriptada. Vejamos, por meio de um exemplo, como funciona esse sistema criptográfico.

Sejam "SENHA" a nossa chave criptográfica e "ABOBORA" a mensagem a ser encriptada. Abaixo colocamos as letras do alfabeto com suas respectivas posições. Observe que repetimos a primeira linha de letras para facilitar a localização da posição da letra encriptada e usamos a barra para indicar que estamos no segundo ciclo.

1	2	3	4	5	6	7	8	9	10	11	12	13
A	B	C	D	**E**	F	G	**H**	I	J	K	L	M

14	15	16	17	18	19	20	21	22	23	24	25	26
N	O	P	Q	R	**S**	T	U	V	X	Y	W	Z

27	28	29	30	31	32	33	34	35	36	37	38	39
$\overline{A}$	$\overline{B}$	$\overline{C}$	$\overline{D}$	$\overline{E}$	$\overline{F}$	$\overline{G}$	$\overline{H}$	$\overline{I}$	$\overline{J}$	$\overline{K}$	$\overline{L}$	$\overline{M}$

Vejamos como encriptar a palavra "ABOBORA". Iniciamos o processo escrevendo a mensagem. Ao lado de cada letra da mensagem aparece entre parênteses o número que indica a sua posição. Abaixo da mensagem escrevemos as letras da chave criptográfica, repetindo-as de forma cíclica quando necessário. Analogamente, ao lado de cada letra da chave aparece entre parênteses o número da posição ocupada de cada letra, e o sinal de soma indica que devemos avançar aquele número de posições. Ao final do processo aparecem as letras encriptadas. Entre parênteses está a posição resultante da combinação das posições da mensagem e da chave.

$A(1)$	$B(2)$	$O(15)$	$B(2)$	$O(15)$	$R(18)$	$A(1)$	Mensagem
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	
$S(+19)$	$E(+5)$	$N(+14)$	$H(+8)$	$A(+1)$	$S(+19)$	$E(+5)$	Chave
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	
$T(20)$	$G(7)$	$C(29)$	$J(10)$	$P(16)$	$K(37)$	$F(6)$	Mensagem encriptada

Observe que a encriptação polialfabética é mais difícil de ser quebrada que a monoalfabética uma vez que letras iguais não têm, necessariamente, a

mesma encriptação. Observe que neste tipo de criptografia o emissor precisa passar a chave para o receptor da mensagem de forma segura para que o receptor possa decifrar a mensagem, isto é, a chave usada para encriptar a mensagem é a mesma que deve ser usada para decifrar a mensagem. Veremos que esse é justamente o ponto fraco nesse tipo de encriptação pois usa a chamada *chave simétrica*, ou seja, a chave usada pelo emissor para codificar a mensagem é a mesma usada pelo receptor para decodificar a mensagem. Nesse processo, a chave deve ser mantida em segredo e bem guardada para garantir que o código não seja quebrado e isso requer algum tipo de contato físico entre emissor e receptor da mensagem.

Durante a Primeira Guerra Mundial o contato físico para a troca de chaves era complicado, e isso estimulou a criação de máquinas automáticas de criptografia. O *Enigma* foi uma dessas máquinas e era utilizada pelos alemães tanto para criptografar como para descriptografar códigos de guerra. Semelhante a uma máquina de escrever, os primeiros modelos foram patenteados por Arthur Scherbius em 1918. Essas máquinas ganharam popularidade entre as forças militares alemães devido a facilidade de uso e sua suposta indecifrabilidade do código.

O matemático Alan Turing foi o responsável por quebrar o código dos alemães durante a Segunda Guerra Mundial. A descoberta de Turing mostrou a fragilidade da criptografia baseada em chave simétrica e colocou novos desafios à criptografia. O grande problema passou a ser a questão dos protocolos, isto é, como transmitir a chave para o receptor de forma segura sem que seja necessário o contato físico entre as partes?

Em 1949, com a publicação do artigo *Communication Theory of Secrecy Systems* [Sha49] de Shannon, temos a inauguração da criptografia moderna. Neste artigo ele escreve matematicamente que cifras teoricamente inquebráveis são semelhantes as cifras polialfabéticas. Com isso ele transformou a criptografia que até então era uma arte em uma ciência.

Em 1976 Diffie e Hellman publicaram *New Directions in Cryptography* [DH76]. Neste artigo há a introdução ao conceito de *chave assimétrica*, onde há chaves diferentes entre o emissor da mensagem e seu receptor. Com a assimetria de chaves não era mais necessário um contato tão próximo entre emissor e receptor. Neste mesmo artigo é apresentado o primeiro algoritmo de criptografia de chave assimétrica ou como é mais conhecido nos dias atuais *Algoritmo de Criptografia de Chave Pública*, o protocolo de Diffie-Hellman.

Um dos algoritmos mais famosos da criptografia de chave pública é o *RSA*(RIVEST et al, 1983) [RSA78], algoritmo desenvolvido por Rivest, Shamir e Adleman. Este algoritmo se tornou poular por estar presente em muitas aplicações de alta segurança, como bancos, sistemas militares e servidores

de internet.

Para que se possa compreender por completo o algoritmo faz-se necessário possuir alguns conhecimentos em Teoria de números como fatoração e aritmética modular. Estes conhecimentos serão apresentados mais adiante neste trabalho.

No algoritmo RSA existe uma chave pública n, que é a multiplicação dos primos p e q. O emissor E codifica a mensagem usando um número primo p. Em seguida E envia publicamente a mensagem codificada junto com a chave n para o receptor R. R possui o número q, que juntamente ao número n servem para decodificar a mensagem.

Embora a quebra do RSA seja aparentemente simples, bastando fatorar n para descobrir seus fatores, o grande problema é na realidade computacional, pois usa-se como p e q números primos muito altos, próximos a 2^{512}. Com um número tão alto um computador comum levaria bem mais que uma vida humana para decifrar a mensagem.

Com base nestes conhecimentos sobre criptografia, temos que o objetivo deste trabalho é analisar a viabilidade de uma criptografia inspirada pelo algoritmo RSA clássico, a qual substitui os números primos pelo conjunto denominado de *primos de Gauss* [PR13], resultando, assim, no que chamamos por *criptografia RSA gaussiana*. Para que tal algoritmo seja viável é necessário adaptar uma série de resultados relativas aos números primos aos número primos de Gauss. Dessa forma, nossa tarefa será adaptar tanto quanto o possível os primos de Gauss às demosntações desses teoremas.

Como se trata de uma proposta inovadora, deixamos para trabalhos futuros uma análise comparativa entre as criptografias RSA clássica e a RSA gaussiana.

Capítulo 2

Um passeio pela Teoria de Números

A teoria de números é umas das mais antigas áreas da matemática e dedica-se ao estudo relacionado às propriedades relativas aos números inteiros tais como a questão da fatoração, máximo divisor comum, entre outras. Ao longo deste capítulo mostraremos os principais resultados de teoria de números essenciais para a compreensão do método de criptografia de chave pública conhecido como RSA.

2.1 Números Primos e Fatoração Única

Os números primos ocupam lugar importante tanto na teoria de números quanto na criptografia RSA: na primeira por serem capazes de gerar todos os elementos do conjunto dos números inteiros e suas consequentes propriedades; na segunda pelo fato de formarem um conjunto infinito e isso permite que se tome um primo de dimensão extrondosa para codificar uma mensagem, consequentemente dificultando que o código seja decifrado por terceiros em tempo razoável,como mostraremos ao longo deste trabalho.

Os primos atuam como átomos dentro do conjunto dos números inteiros no sentido em que todo número pode ser escrito como um produto de primos. Esse fato é consequencia do chamado *Teorema da Fatoração Única* também conhecido por *Teorema Fundamental da Aritmética*. Além de ser um resultado fundamental para a teoria de números, ele também é um dos pilares da criptografia RSA, pois a decodificação de uma mensagem vai passar pela fatoração de um número, e para demonstrar esse teorema é preciso ter a disposição o chamado *Teorema de Divisão*.

Teorema 2.1.1 (Teorema de divisão)**.** *Sejam a e b inteiros positivos. Existem números inteiros q (quociente) e r (resto) tais que:*

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

Além disso, os valores de q e r satisfazendo as relações acima são únicos.

Demonstração
Confira em Coutinho [Cou07], seção 3 do capítulo 1, p. 22.

O teorema acima faz duas afirmações: a primeira que o quociente e o resto da divisão sempre existem; a segunda, que o quociente e o resto são únicos. A garantia da unicidade é o ponto crucial na aplicação à criptografia RSA, pois assim temos a garantia de que uma mensagem possa ser decodificada de maneira única. Um outro resultado igualmente importante é o *Algoritmo de Euclides* mais conhecido como método para se calcular o máximo divisor comum entre dois números. Esse resultado é importante para definir o que entendemos por números primos e consequentemente para o teorema da fatoração única que mostra como expressar um número em fatores primos de forma única. Para este trabalho vamos precisar da versão estendida desse método.

Teorema 2.1.2 (Teorema do Máximo Divisor Comum)**.** *Sejam a e b inteiros positivos e seja d o máximo divisor comum entre a e b. Existem inteiros α e β tais que:*

$$\alpha \cdot a + \beta \cdot b = d$$

Diferente do teorema da divisão, o teorema do máximo divisor comum não garante a unicidade com relação aos inteiros α e β. Isso acaba sendo um complicador para a criptografia RSA, mas veremos que esse problema acaba sendo contornado por temos a disposição um método eficiente para calcular esses números.

Tendo os resultados acima a nossa disposição podemos, agora, definir o que entendemos por números primos para então atingir nossa meta com este capítulo: o Teorema da Fatoração Única.

Definição 2.1.3. Um número inteiro p é *primo* se $p \neq \pm 1$ e os únicos divisores de p são ± 1 e $\pm p$.

São exemplos de números primos: ± 2, ± 3, ± 5, ± 7, ± 11, ± 13, etc.

Um número inteiro, diferente de ± 1, que não é primo é chamado *composto*. Observe que os números 1 e -1 não são nem primos e nem compostos.

A exclusão desses números do conjunto dos primos é para garantir a unicidade da fatoração no teorema a seguir. Um outro aspecto a se destacar acerca desse par de números é que eles são os únicos que admitem inverso multiplicativo, que virá a ser explicado mais adiante.

Teorema 2.1.4 (Teorema da Fatoração Única)**.** *Dado um inteiro positivo $n \geq 2$ podemos sempre escrevê-lo, de modo único, na forma*

$$n = p_1^{e_1} \dots p_k^{e_k}$$

onde $1 < p_1 < p_2 < p_3 < \cdots < p_k$ são números primos e $e_1, \cdots, e_k$ são inteiros positivos.

No teorema acima, os expoentes e_i, para $1 \leq i \leq k$ são chamados de *multiplicidades*, pois indicam a quantidade de vezes que um mesmo número primo ocorre na fatoração. A prova de que é sempre possível encontrar os fatores de usados decompor o número em fatores primos consiste no procedimento para fatorar um número, esse procedimento é chamado *Algoritmo de Euclides*: trata-se do método que se aprende na escola para fatorar um número e que não iremos detalhar aqui. Como bem sabemos, esse método é bastante ineficaz quando pensamos em números muito grande, pois depende de realizar uma sequência bem grande de divisões, dependendo do número. A prova garante que o procedimento termina, mas o que se nota é que tal procedimento é muito ineficiente no sentido em que demanda muito tempo para se chegar a uma resposta dependendo do número de desejamos fatorar. Na literatura existem vários algoritmos de fatoração que tornam o método mais eficiente, no entanto nenhum desses métodos funciona bem para todos os números inteiros. A criptografia RSA aproveita a ineficiência do método para fatorar um número para garantir a segurança do seu sistema. É um problema em aberto saber se existe ou não um método rápido para fatorar números inteiros.

A demonstração do teorema acima requer uma série de resultados acerca de números primos os quais detalharemos abaixo. O interesse em apresentar as demonstrações de tais resultados é devido ao fato de estarmos interessados em adaptar tais provas para o primos de Gauss se quisermos implementar um método de criptografia RSA baseado nesses primos.

Teorema 2.1.5. *Sejam a e b inteiros positivos e suponhamos que a e b são primos entre si.*

1. *Se b divide o produto ac então b divide c.*
2. *Se a e b dividem c então o produto ab divide c.*

Demonstração

1. Se a e b são primos entre si, então o máximo divisor comum entre a e b é 1, isto é,$mdc(a, b) = 1$. Pelo algoritmo euclideano estendido, temos que existem inteiros α e β tais que $\alpha \cdot a + \beta \cdot b = 1$. Daí, multiplicando toda a expressão por c temos que: $\alpha \cdot ac + \beta \cdot bc = c$ (1). Dado que b divide ac e b divide cb, então b divide $\alpha \cdot ac + \beta \cdot bc$. Portanto, a partir da igualdade (1) temos que b divide c.

2. Se a divide c, então existe $t \in \mathbb{Z}$ tal que $c = at$. Como, por hipótese, b divide c e a e b são primos entre si, então b tem que dividir t. Logo, para algum t vale que $t = bk$. Portanto, $c = at = a(bk) = (ab)k$ é divisível por ab.

Teorema 2.1.6 (Propriedade Fundamental dos Primos)**.** *Seja p um número primo e a e b inteiros positivos. Se p divide o produto ab então p divide a ou p divide b.*

Demonstração
Se a e p não forem primos entre si então máximo divisor comum entre eles é p, logo p divide a. Suponhamos que a e p são primos entre si, isto é, $mdc(p, a) = 1$. Como, por hipótese, p divide $a\dot{b}$, então pelo Teorema 2.1.5 segue que p divide b.

Estamos interessados, agora, em mostrar que a lista de primos é infinita, para tal vejamos o seguinte resultado intermediário.

Teorema 2.1.7 (Existência de Divisor Primo)**.** *Se n é um número inteiro positivo composto, então n tem um divisor primo p tal que $p \leq \sqrt{p}$.*

Demonstração:
Se n é um número composto e positivo, podemos supor que $n = a \cdot b$, com $1 < a \leq b$.

1. De $1 < a$, temos que existe um primo p que divide a (Teorema 2.1.4), com $p \leq a$, daí $p^2 \leq a^2$.

2. De $a \leq b$ temos que $a^2 \leq a \cdot a = n$

De (1) e (2) segue que $p^2 \leq n$, logo $p \leq \sqrt{n}$.

Teorema 2.1.8 (Infinidade de Primos)**.** *Existe uma quantidade infinita de números primos.*

Demonstração:
Suponha, por redução ao absurdo, que exite apenas uma quantidade finita de números primos: $p_1, p_2, \cdots p_n$. Tome $a = 1 + p_1 \cdot p_2 \cdot \cdots \cdot p_n$ um número inteiro. Claramente $a > p_i$ para cada $1 \leq i \leq n$, então a deve ser um número composto, caso contrário a lista acima estaria incompleta. Dessa forma, pelo Teorema2.1.7 existe um primo p_i tal que divide a. Mas se p_i divide a, então p_1 divide 1 e p_i divide $p_1 \cdot p_2 \cdot \cdots \cdot p_n$. Absurdo, pois o único divisor de 1 é ele mesmo. Portanto, é falso supor que a lista de primos seja finita, logo ela deve ser infinita.

Existe, ainda, um outro debate acerca dos números primos: como gerar os números primos. Existem diversos métodos para gerar os primos, como por exemplo, o Crivo de Eratóstenes, o mais antigos deles, mas não envolve nenhuma fórmula específica. No entanto, todos esses métodos são ineficazes. Para mais detalhes sobre esse tema recomendamos a leitura de Criptografia, de Coutinho[Cou07].

Como mostramos, temos o Teorema 2.1.4 que garante que um número possa ser decomposto em fatores primos de forma única e o Terema 2.1.8 que garante uma infinidade de números primos, no entanto, como dissemos, os procedimentos atrelados a esses resultados são todos muito ineficientes em termos computacionais. Para implementar a criptografia RSA vamos precisar de procedimentos mais eficazes e por essa razão será conveniente trabalhar com o conjunto de números inteiros. Para isso vamos separar os números inteiros em classes de equivalências, pois dessa forma será possível operar com essas classes de forma semelhante como fazemos com os inteiros. Esse é justamente o tema da próxima seção.

2.2 Aritmética Modular

Para compreender a intuição por trás da aritmética modular é interessante pensar na ideia de *ciclicidade*, isto é, fatos que ocorrem após um determinado período constante. Por exemplo, o nascer do sol é um evento que ocorre sempre após um ciclo de 24 horas; a data de seu aniversário ocorre a cada ciclo de um ano. Trabalhar com ciclos requer que tenhamos

uma nova forma de operar com números, pois quando somamos 13 com 15 o resultado pode ser 4 se estivermos pensando em termos de horas, pois após 24 horas retornamos ao marco zero e reiniciamos a contagem para facilitar o reconhecimento da hora em questão.

Quando mostramos o processo de codificação e de decodificação de um código em *cifras de substituição monoalfabéticas* você deve ter notado que precisamos repetir o alfabeto a fim de podermos operar com as posições ocupadas por uma determinada letra do alfabeto. A repetição do alfabeto foi usada para mostrar as diferentes representações das letras dentro do ciclo estipulado.

Podemos observar que os ciclos geram classes de números, isto é, os números 0, 24, 48, 72, etc. indicam todos o marco zero do relógio se estivermos iniciando a contagem das horas no marco zero. Esses números formam a *classe da zero hora*. Da mesma forma podemos compor a *classe da uma hora*, *classe das duas horas* e assim por diante. No exemplo das posições ocupadas pelas letras do alfabeto temos que a *classe da letra A* é composta pelos números 1, 27, 53, etc.

Nosso interesse está voltado para a uniformização do modo de separar tais classes para podemos operar com essas classes e daí fazer uso dessa intuição de forma operacionalizada.

Definição 2.2.1. Uma classe é chamada de *equivalência* se satisfaz as seguintes propriedades:

- Reflexividade: $\forall x, x = x$;
- Simetria: se $x = y$, então $y = x$;
- Transitividade: se $x = y$ e $y = z$, então $x = z$.

Para exemplificar a definição acima, cosidere o exemplo das horas: a propriedade reflexiva afirma que toda hora é igual a ela própria; a simetria afirma que se $0h$ é igual a $24h$, então temos também que $24h$ ígual a $0h$; a transitividade afirma que $0h$ é igual a $24h$ e $24h$ é igual a $72h$, então $0h$ é igual a $72h$.

Seja X um conjunto e $\sim$ uma relação de equivalência definida em X. Denotamos por $\overline{X}$ a classe de equivalência de x e escrevemos, em símbolos, da seguinte forma:

$$\overline{x} = \{y \in X : y \sim x\}$$

Nosso interesse será separar em classe de equivalência os números inteiros, dessa forma o X representa o conjunto $\mathbb{Z}$ enquanto que x representa

um número inteiro, enquanto que $\sim$ representa alguma relação estabelecida entre os números, por exemplo o resto da divisão pelo número 5. Dessa forma, podemos formar cinco classes distintas:

- Classe dos restos 0: $\overline{0} = \{0, 5, 10, 15, 20, \cdots\}$
- Classe dos restos 1: $\overline{1} = \{1, 6, 11, 16, 21, \cdots\}$
- Classe dos restos 2: $\overline{2} = \{2, 7, 12, 17, 22, \cdots\}$
- Classe dos restos 3: $\overline{3} = \{3, 8, 13, 18, 23, \cdots\}$
- Classe dos restos 4: $\overline{4} = \{4, 9, 14, 19, 24, \cdots\}$

O conjunto das classes de equivalência em X é chamado *conjunto quociente de X por $\sim$*. No nosso exemplo $\{\overline{0}, \overline{1}, \overline{2}, \overline{3}, \overline{4}\}$ representa conjunto quociente de $\mathbb{Z}$ pela divisão por 5. Esse conjunto será denotado por $\mathbb{Z}_5$. Em termos gerais, o conjunto quociente dos números inteiros é denotado por:

$$\mathbb{Z}_n = \{\overline{0}, \overline{1}, \overline{2}, \cdots, \overline{n-1}\}$$

Capítulo 3

Inversos Modulares

Nosso objetivo com o decorrer deste capítulo é o de explicar a operação matemática mais importante para para o algoritmo RSA. Para podermos comprendê-la vamos relembrar do conceito ensinado no colégio de inverso multiplicativo, que consiste em obter o número que multiplicado a um número n qualquer resulte em 1. Juntos, nós iremos ver que na operação do inverso modular se segue do mesmo princípio.

3.1 Inversos modulares

Para iniciarmos este capítulo, vamos supor que queremos obter o inverso modular de 6 para o módulo 7, o que nós teremos que fazer então é encontrar qual o número que multiplicado por 6 tem resto 1 quando dividido por 7. Começamos pelo 1, teremos que $6 \cdot 1 = 6$, $6 \equiv 6(mod 7)$. Com 2 o resultado será 12, logo $12 \equiv 5(mod 7)$, que para nós também não serve. Tentando o 3 obtemos 4 e com 4 obtemos 3. Com o 5 nosso retorno será 2. Finalmente quando chegamos ao 6 nós temos que $6 \cdot 6 = 36$, $36 \equiv 1(mod 7)$. Com isso podemos concluir que o inverso multiplicativo de 6 no módulo 7 é o própio 6.

Para simplificar o que foi dito acima, podemos dizer a operação de inverso multiplicativo no módulo n para a consiste em encontar um número a' tal que:

$$a \cdot a' \equiv 1(mod n)$$

3.2 Inexistência e existência de inversos

Antes de começarmos vamos tentar calcular o inverso multiplicativo de 2 no módulo 8, vamos lá:

$$2 \cdot 0 \equiv 0 \not\equiv 1(mod8)$$
$$2 \cdot 1 \equiv 2 \not\equiv 1(mod8)$$
$$2 \cdot 2 \equiv 4 \not\equiv 1(mod8)$$
$$2 \cdot 3 \equiv 6 \not\equiv 1(mod8)$$
$$2 \cdot 4 \equiv 8 \not\equiv 1(mod8)$$
$$2 \cdot 5 \equiv 0 \not\equiv 1(mod8)$$
$$2 \cdot 6 \equiv 2 \not\equiv 1(mod8)$$
$$2 \cdot 7 \equiv 4 \not\equiv 1(mod8)$$

Não encontramos nenhuma resposta pois, simplesmente, não há. Antes que se pergunte o motivo de não tentarmos com números maiores que 7, é válido lembrar que a partir do 8 teríamos a repetição de resultados por conta das congruências.

A operação de inverso multiplicativo só possui resultado em casos onde o número a ao qual queremos calcular o inverso e o m'odulo são *primos entre si*, ou seja, não possuam nenhum fator em comum. Por conta disso usamos os números primos no algoritmo RSA.

Para comprovar o que foi dito acima, vamos tomar um número a, tal que

$$a \cdot a' \equiv 1(modn)$$

isso pode ser traduzido em linguagem humana como n divide $a \cdot a' - 1$. Isso em linguajar matemático pode ser escrito como:

$$a \cdot a' - 1 = n \cdot k$$

como estamos atrás de saber se a e n não possuem fator comum, então há de haver um k inteiro para a equação acima. Nosso primeiro passo para provar isso será de se criar o conjunto $V(a, n)$, esse conjunto é formado por inteiros positivos e pode ser escrito como

$$x \cdot a + y \cdot n$$

Em um primeiro momento este conjunto e esta nova fórmula podem parecer estranhos ao que se via antes, mas se comprovarmos que $1 \in V(a,n)$, concluímos que devem haver dois inteiros x_0 e y_0, ou se preferir a' e k, logo:

$$1 = a \cdot a' - n \cdot k$$

Uma das propiedades deste conjunto é a de n pertencer a ele quando $x = a' = 0$ e $y = k = 1$. Isto significa que os inteiros que podem completar a equação estão entre m e n. Mas para podermos dar essa demonstração como completa, precisamos provar que $m = 1$. Como a e n são primos entre si, basta mostrar que m divide ambos. Vamos começar assumindo que $m \in V(a,m)$, logo devem existir x_1 e y_1 tais que:

$$m = x_1 \cdot a + y_1 \cdot n$$

Caso venhamos a dividir n por m, obtemos:

$$n = m \cdot q + r = (x_1 \cdot a + y_1 \cdot n) \cdot q + r \ \ e \ 0 \leq r < m$$

Que pode ser organizada como:

$$r = (-x_1) \cdot a + (1 - y_1) \cdot n$$

Com isso podemos concluir que $r \in V(a,n)$, como r é o resto da divisão de n por m, de modo que $r = 0$ ou $r \neq 0$. $r \neq 0$ é absurdo por conta de $r < m$, com m divisor de n, por m ser o menor elemento de $V(a,n)$. Logo só nos resta conferir se $r = 0$, o que prova que m divide n. Pode se provar de forma semelhante que m divide a.

3.3 Um teorema e um corolário

Com base em tudo que já foi lido e provado anteriormente neste capítulo podems concluir o seguinte teorema:

Sejam** $a < n$ **inteiros positivos. O resíduo** a **em inverso no módulo** n **se, e somente se,** a **e** n **não possuem fatores primos em comum.

Esta informação nos será muito útil mais a frente. Outra coisa muito importante para nosso prosseguimento é o seguinte corolário:

Sejam $a < n$ ***inteiros positivos sem fatores comuns, se :***

$$a \cdot b \equiv a \cdot c(mod n)$$

então:

$$b \equiv c(mod n)$$

Nos capítulos seguintes poderemos ver como iremos aplicar estas propiedades, que nos serão muito úteis para agilizar nossos cálculos.

Capítulo 4

Teorema chinês do resto

O teorema chinês do resto tem como principal objetivo resolver congruências que a primeira vista parecem não possuir solução. Ao longo deste e do próximo capítulo iremos ver como ele pode vir a facilitar operações, ao ponto de tornar coisas que pareciam impossíveis possíveis.

4.1 Introdução a técnica

Para sermos iniciados nesta técnica, vamos analisar o seguinte problema: Qual o menor inteiro que possui resto 1 na divisão por 3 e resto 2 na divisão por 5. Podemos vir a tranformar esse problema nas seguintes equações:

$$n = 3q_1 + 1 \ e \ n = 5q_2 + 2$$

Essas equações também podem ser denotadas em forma modular como:

$$n \equiv 1(mod3) \ e \ n \equiv 2(mod5)$$

Essa saída modular nos deixou com apenas uma variável, mas ainda não resolveu ao nosso problema. Para fazermos isso vamos substituir n por $5q_2 + 2$, montando a seguine equação modular:

$$5q_2 + 2 \equiv 1(mod3)$$

Como $5 \equiv 2(mod3)$, substituímos:

$$2q_2 + 2 \equiv 1(mod3)$$

Feito isso, passamos 2 para o outro lado da equação

$$2q_2 \equiv -1(mod3)$$

Como $-1 \equiv 2(mod3)$, nós substitímos novamente, e depois dividimos a equação por 2, e obtemos

$$q_2 \equiv 1(mod3)$$

Com isso, concluímos que

$$q_2 \equiv q_3 + 1(mod3)$$

Sei que parece que mais uma equação só serve para tornar a resolução mais complexa, mas vamos a reorganizar como

$$q_2 = 3q_3 + 1$$

Agora substituímos

$$n = 5(3q_3 + 1) + 2 = 15q_3 + 7$$

Feito isso, vamos por o 3 em evidência em todos os lugares, obtendo:

$$n = 3(5q_3) + 3(2) + 1 = 3(5q_3 + 2) + 1$$

Este procedimento foi feito apenas para provar que a equação deixa resto 1 se dividida por 3, de forma análoga, abaixo é mostrado como ela deixa resto 2 quando dividida por 5.

$$n = 5(3q_3) + 5(1) + 2 = 5(3q_3 + 1) + 2$$

Após tudo isso feito ainda não possuímos a solução final, mas já sabemos que é um número da forma $15q_3 + 7$, substituindo q_3 or 0, iremos obter 7, que é o resultado procurado.

4.2 Provas ao teorema

O teorema chinês do resto é um procedimento tomado para resolver sistema de congruências, como o descrito acima. Ele foi descrito pela primeira vez pelo Manual de aritmética do mestre Sun, por volta do século III d.C.

Para ver a definição formal desse teorema, vamos considerar o sistema

$$x \equiv a(mod n)$$
$$x \equiv b(mod m)$$

nele, n e m são inteiros diferentes entre si. Tomemos x_0 como um número cappaz de satisfazer ambas as congruência de forma simultânea e teremos:

$$x_0 \equiv a(mod m)$$
$$x_0 \equiv b(mod n)$$

Para podermos juntar ambas as equações converteremos uma em equação, nesse caso teremos

$$x_0 = a + m \cdot k, \textit{ com } k \textit{ sendo um inteiro qualquer}$$

Feito isso, chegaremos em

$$a + m \cdot k \equiv b(mod n)$$

que pode ser substituída por

$$m \cdot k \equiv (b - a)(mod n)$$

Agora vamos supor que m e n são primos entre si. Pelo teorema apresentado no capítuo sobre inversos multiplicativos nós já sabemos que eles possuem inverso multiplicativo um para o outro. Tomemos m' como o inverso de m no módulo n. Multipplicando toda a congruência por m' obtemos

$$k \equiv m' \cdot (b - a)(mod n)$$

que pode ser escrita como:

$$k \equiv m' \cdot (b - a) + n \cdot t, \textit{ para um inteiro } t \textit{ qualquer}$$

Substituindo a parte de k, nós obtemos

$$x_0 \equiv a + m(m' \cdot (b - a) + n \cdot t)$$

Podemos ver agora que para qualquer t, $a + m(m' \cdot (b - a) + n \cdot t$ é parte da solução da congruência, sabendo disso, agora podemos descrever o teorema em si, que será feito na próxima seção.

4.3 O teorema chinês do resto

***Teorema chinês do resto* - Sejam m e n inteiros positivos primos entre si. Se a e b são inteiros quaisquer, então o sistema**

$$x \equiv a(modm)$$
$$x \equiv b(modn)$$

sempre tem solução e qualquer uma de suas soluções pode ser escrita na forma

$$a + m \cdot (m' \cdot (b - a) + n \cdot n)$$

onde t é um inteiro qualquer e m' é o inverso de m no módulo n.

Capítulo 5

Potenciação

Ao longo deste capítulo vamos estudar como tornar as operações de potenciação e a obtençao de seus restos calcaláveis de forma simpls e rápida. Para isso vamos dispor de algumas artimanhas matemáticas, como o famoso *Teorema de Fermat* e o Teorema chinês do resto.

5.1 Restos na potenciação

Vamos começar tentando uma coisa que aparentemente é complexa, mas se converterá em uma operação bem simples: Calcular o resto da divisão de 10^{135} por 7. Podemos fazer da forma tradicional, mas dividir um número tão alto não seria nada prático.

O que faremos é tomar uma propiedade da multiplicação e da potenciação emprestadas, a do elemento neutro, nesse caso o 1. O que faremos é calcular em qual potência 10 é congruente a 1 no módulo 7. Logo teremos a tabela:

$$10^1 \equiv 3(mod 7)$$
$$10^2 \equiv 2(mod 7)$$
$$10^3 \equiv 6(mod 7)$$
$$10^4 \equiv 4(mod 7)$$
$$10^5 \equiv 5(mod 7)$$
$$10^6 \equiv 1(mod 7)$$

Como sabemos agora que 10^6 é o número que queríamos, vamos decompor o 135 em razão de 6 e teremos que $135 = (6 \cdot 22) + 3$, essa expressão nos levará a seguinte congruência:

$$10^{135} \equiv (10^6)^{22} \cdot 10^3 \equiv (1)^{22} \cdot 10^3 \equiv 10^3 \equiv 6(mod 7)$$

Agora nós vamos deixar essa operação um pouco mais complexa, ao passo que vamos calcular o resto por 31 de 2^{124512}. Vamos pelo mesmos caminho que anteriormente, buscando a potência de 2 que é congruente a 1 no módulo 31. Obtemos:

$$2^1 \equiv 2(mod 31)$$
$$2^2 \equiv 4(mod 31)$$
$$2^3 \equiv 8(mod 31)$$
$$2^4 \equiv 16(mod 31)$$
$$2^5 \equiv 1(mod 31)$$

Vamos dividir 124512 por 5 e obteremos 4016 com resto 2, obtendo assim que $2^{124512} \equiv 2^2 \equiv 4(mod 31)$.

Tornando um pouco mais difícil, podemos calcular o resto de $2^{1398765}$, é descobrir o resto de 13^{98765} por 5, podemos dizer que $13^{98765} \equiv 3^{98765}(mod 5)$ como se sabe que $3^4 = 81 \equiv 1(mod 5)$, podemos usar isso em nosso favor, pois teremos $3^{98765} \equiv 3^{4 \cdot 24691+1} \equiv 3(mod 5)$, logo o resultado de 13^{98765} é um número da forma $5q'+3$. Como isso, nós podemos dizer que $2^{1398765} \equiv 2^{5q'+3} \equiv 2^{5q'} \cdot 2^3 \equiv 1^{q'} \cdot 2^3 \equiv 8(mod 31)$.

5.2 O teorema de Fermat

***Teorema de Fermat* - Se p é um número primo e a é um inteiro não divisível por p, então:**

$$a^{p-1} \equiv 1(mod p)$$

Embora esse seja denominado como o pequeno teorema de Fermat, ele possui suma importância para o algoritmo RSA clássico. Vamos apresentar abaixo uma de suas demonstrações.

Sabemos que os possíveis resíduos no módulo p são todos os inteiros entre 1 e $p-1$. Vamos multiplicá-los por a, obtendo assim:

$$a \cdot 1, a \cdot 2, a \cdot 3, ..., a \cdot (p-1)$$

Vamos levar em conta que $r_1 \equiv a \cdot 1(modp)$,$r_2 = a \cdot 2(modp)$, e assim por diante até $r_{p-1} = a \cdot (p-1)(modp)$. Tomemos par r_k e r_l um par de inteiros k e l que está entre 1 e $p-1$. Com isso teremos:

$$a \cdot k \equiv k \equiv l \equiv a \cdot l(modp)$$

que equivale à:

$$a \cdot k \equiv a \cdot l(modp)$$

Se viermos a cancelar pela equivalência, obteremos que $k \equiv l(modp)$, mas sendo k e l positivos, inteiros e menores que p, estes só podem ser congruentes se forem iguais, logo se:

$$r_k = r_l \ então \ k = l$$

Isto demonstra que $r_1, r_2, r_3, ...r_{p-1}$ são $p-1$ resíduos não nulos de módulo p, que serão $1, 2, 3, ..., p-1$, o que nos permite dizer que a primeira sequência não é nada além de um reordenamento da segunda. Com isso podemos dizer que:

$$r_1 \cdot r_2 \cdot r_3 \cdot ... \cdot r_{p-1} = 1 \cdot 2 \cdot 3 \cdot ... \cdot p-1$$

Sabendo disso vemos que:

$$a^{p-1}(1 \cdot 2 \cdot 3 \cdot ... \cdot p-1) \equiv (1 \cdot 2 \cdot 3 \cdot ... \cdot p-1)(modp)$$

E apenas cortando os fatores iguais:

$$a^{p-1} \equiv 1(modp)$$

Provando assim o Teorema de Fermat.

5.3 Aplicando o teorema de Fermat

Antes de prosseguirmos para o próximo capítulo, vamos utilizar o Teorema de Fermat para resolver uma congruência. Neste caso vamos tentar descobrir quem é congruente a 3^{10342} no módulo 1033. Como 1033 é primo nós podemos usar o teorema de Fermat. Neste caso teremos que:

$$3^{1032} \equiv 1(mod 1033)$$

O que faremos agora consiste em "dividir" 1034 por 1032, de forma a obter o resto da divisão. e com isso vamos veirficar que:

$$1034^2 \equiv 2^2 \equiv 4(mod 1033)$$

e com essa simplificação chegamos à:

$$3^{1034} \equiv 3^{1032} \cdot q + 4 \equiv (3^{1032})^q + 3^4(mod 1033)$$

Agora com a simples aplicação do Teorema de Fermat, podemos chegar a conclusão que:

$$3^{1034^2} \equiv 1 \cdot 81(mod 1033)$$

verificando assim que 3^{1034^2} deixa resto 81 na divisão por 1033.

5.4 Teorema de Fermat para potências compostas

Embora aplicar o teorema de Fermat diretamente sobre os números compostos não seja possível, nós ainda podemos resolver a estas congruências com o auxílio do teorema chinês d restos, como veremos a seguir.

Para que possamos entender como resolver este problema com números compostos vamos tentar resolver um problema númerico, nesse caso o cálculo do módulo de 2^{6754} por 1155.

Nosso primeiro passo é fatorar o 1155. Ao fim da fatoração vamos obter que $1155 = 3 \cdot 5 \cdot 7 \cdot 11$. Em seguida vamos aplicar o teorema de Fermat a cada um dos primos, obtendo assim:

$$2^2 \equiv 1(mod3)$$
$$2^4 \equiv 1(mod5)$$
$$2^6 \equiv 1(mod7)$$
$$2^{10} \equiv 1(mod11)$$

Agora dividimos 6754 por $p-1$ para cada um dos múltiplos:

$$6754 = 2 \cdot 3377$$
$$6754 = 4 \cdot 1688 + 2$$
$$6754 = 6 \cdot 1125 + 4$$
$$6754 = 10 \cdot 675 + 4$$

Em seguida substituímos nas congruências e as reduzimos

$$2^{6754} \equiv 2^{3377^2} \equiv 1(mod3)$$
$$2^{6754} \equiv 2^{1688^4} \cdot 2^2 \equiv 1 \cdot 4 \equiv 4(mod5)$$
$$2^{6754} \equiv 2^{1125^6} \cdot 2^4 \equiv 1 \cdot 16 \equiv 2(mod7)$$
$$2^{6754} \equiv 2^{675^{10}} \cdot 2^4 \equiv 1 \cdot 16 \equiv 5(mod11).$$

Logo, nossa tarefa consiste em resolver o sistema

$$x \equiv 1(mod3)$$
$$x \equiv 4(mod5)$$
$$x \equiv 2(mod7)$$
$$x \equiv 5(mod11)$$

Podemos resolver esse sistema usando o algoritmo chinês, vamos começar substituindo na primeira congruência, onde $x = 3y + 1$, em seguida substituímos x por y na segunda congruência, tornando-a $3y + 1 \equiv 4(mod5)$, que equivale a $y \equiv 1(mod5)$ como 3 é inversível no módulo 5 ele pode ser anulado na equação. Com isso temos $x = 4 + 15z$ que se substiutindo na terceira equação e resolvendo obtemos $z \equiv 5(mod7)$, que significa que $x = 79 + 105t$. Finalmente substituindo na última equação, teremos que $t \equiv 6(mod11)$, o que resulta em $x = 709 + 1155u$. Concluímos com isso que $26754 \equiv 709(mod1155)$.

Capítulo 6

Aplicando a criptografia RSA

A criptografia RSA tem suma importância para toda a comunicação moderna. Ela é tão importante que a descoberta de uma forma de se desencriptá-la colocaria em risco a sociedade como a conhecemos. Ao longo deste capítulo nós vamos ver como é seu funcionamento, unindo todos os conteúdos já vistos anteriormente.

6.1 Preparando-se para criptografar

Para que o algoritmo RSA possa encriptar de forma eficiente, nós precisaremos seguir uma série de passos necessários para que o RSA funcione, mas que ainda não são parte do algoritmo.

O primeiro passo é a conversão das letras da mensagem em números. A essa etapa nós chamaremos de pré-codificação. Para que o RSA venha a funcionar precisamos seguir uma tabela como a apresentada abaixo:

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	X	Y	W	Z
23	24	25	26	27	28	29	30	31	32	33	34	35

Para representar espaços vamos usar o 99. Avisamos que esta é uma tabela apenas com finalidade didático, e, por isso há vários caracteres desconsiderados. Como exemplo vamos pré-encriptar o poema Amor, de Oswald de Andrade. O texto do poema a ser pré-encriptado é o seguinte:

Amor
Humor.

Como primeiro passo vamos converter todas as letras em números, resultando em:

10 22 24 27 99 17 30 22 24 27

Feito isso, nós agrupamos o conjunto em um bloco único de caracteres

10222427991730222427

Atente-se ao fato de todo o caractere convertido possuir sempre o mesmo número de algarismos. Isso é útil para evitar ambiguidades na fase de desencriptação.

Nosso próximo passo nesta fase que antecede a encriptação consiste em definir quais serão os primos p e q. Para nosso exemplo vamos usar $p = 17$ e $q = 23$, como mencionado na Introdução desta obra, temos que $n = pq$, logo $n = 391$.

O último passo da pré-encriptação consiste em quebrar o número que obtemos acima em blocos menores. Essas blocos devem obedecer a duas regras básicas, devem ser menores que n, ou no nosso exemplo 391, pois iremos trabalhar com módulos de 391 durante a encriptação, e não podem se iniciar por 0, para não haver ambiguidades na desincriptação. Vejamos como a nossa mensagem fica quando pré-encriptada.

102 — 224 — 279 — 91 — 7 — 30 — 222 — 42 — 7

Perceba que não há relação entre nenhum dos números obtidos com um caractere específico, o que torna impossível a associação de um número a uma letra por frequência de aparecimento.

6.2 Codificando e decodificando mensagens

Encerrada a fase de pré-codificando vamos agora codificar nossas mensagens. Manteremos os valores e exemplos da seção anterior a fim de facilitar a compreensão.

6.2.1 Codificando uma mensagem

A esta altura nós já conhecemos o número n, que em nosso exemplo possui o valor de 391. Vamos chamar o bloco codificado que iremos encriptar de b, lembrando que b é um número inteiro menor que n. Vamos chamar

o bloco após a codificação de $C(b)$. Para obtermos $C(b)$ devemos aplicar a seguinte fórmula:

$$C(b) \equiv b^3 \pmod{n}$$

Podemos para facilitar dizer que $C(b)$ é o resíduo de b^3 pelo módulo n. Vamos a uma demonstração prática com o primeiro bloco de nossa mensagem, que possui o valor 102. Para simplificar o nosso trabalho vamos utilizar as operações modulares

$$102^3 \equiv 24276 \equiv 34 \pmod{391}$$

Faremos o mesmo procedimento para todos nossos blocos

$$224^3 \equiv 11239424 \equiv 129 \pmod{391}$$
$$279^3 \equiv 21717639 \equiv 326 \pmod{391}$$
$$91^3 \equiv 753571 \equiv 114 \pmod{391}$$
$$7^3 \equiv 343 \equiv 343 \pmod{391}$$
$$30^3 \equiv 27000 \equiv 21 \pmod{391}$$
$$222^3 \equiv 10941048 \equiv 86 \pmod{391}$$
$$42^3 \equiv 74088 \equiv 189 \pmod{391}$$
$$7^3 \equiv 343 \equiv 343 \pmod{391}$$

Portanto, "Amor Humor", encriptado pelo RSA com as chaves privadas 17 e 23 e a pública 391 é:

$$34 — 129 — 326 — 114 — 343 — 21 — 86 — 189 — 343$$

6.2.2 Decodificando uma mensagem

Para podermos desencriptar uma mensagem nós precisamos de dois números. O primeiro é a nossa chave pública n. O segundo número é d, onde temos que $3d$ é congruente a 1 para o módulo $(p-1)(q-1)$. Mais a frente explicaremos o motivo de se adicionar este número. Para resumir o que dissemos neste parágrafo, temos que:

$$3d \equiv 1 \pmod{(p-1)(q-1)}$$

Agora que estamos de posse de d, podemos usar o par (n, d) para desencriptar a mensagem, onde a é o bloco encriptado e $D(a)$ a mensagem desencriptada, usando a fórmula:

$$D(a) \equiv a^d \pmod{n}$$

Note que na função acima nós assumimos o compromisso de que $D(C(b)) = b$. Para comprová-la vamos na próxima sessão fazer sua demonstração. Neste momento vamos apenas aplicá-la ao nosso exemplo. Sabemos que o primeiro passo consiste em calcular d. Vamos tomar que p e q deixam resto 5 na divisão por 6. Com isso podemos afirmar que:

$$(p-1)(q-1) \equiv 4 \cdot 4 \equiv 16 \equiv 4 \equiv -2 \pmod{6}$$

$$(p-1)(q-1) = 6 \cdot k - 2$$

No entanto, podemos dizer que $6 \cdot k - 2 \equiv 4 \cdot k - 1 \pmod{3}$.Podendo dizer que assim que d é igual a $4 \cdot k - 1$. Feito isso vamos aos números primos de nosso exemplo: 17 e 23, com eles iremos obter:

$$(p-1)(q-1) = 16 \cdot 22 = 352 = 6 \cdot 58 + 4 = 6 \cdot 59 - 2$$

Com isso obtemos que $k = 59$. Aplicando k, nós teremos que:

$$d = 4 \cdot 59 - 1 = 235$$

Agora que já conhecemos a d podemos decodificar a mensagem. Vamos fazer isso em nosso primeiro bloco codificado, que possui o valor 34. Para achar a resposta precisaremos calcular $D(34) \equiv 34^{235} \pmod{391}$. Esse cálculo seria praticamente impossível sem o uso dos Teoremas chinês do resto e de Fermat.

Nosso primeiro passo será o de calcular 34^{235} nos módulos 17 e 23, que são os primos resultantes da fatoração de n. Neste caso, começamos com:

$$34 \equiv 0 \pmod{17}$$

$$34 \equiv 11 \pmod{23}$$

Assim teremos que $34^{235} \equiv 0^{235} \equiv 0 \pmod{17}$. Aplicando o teorema de Fermat na outra congrência teremos:

$$11^{235} \equiv (11^{22})^{10} \cdot 11^{15} \equiv 11^{15} \pmod{23}$$

Como $11 \equiv -12 \equiv -4 \cdot 3 \pmod{23}$, nós podemos afirmar que:

$$11^{235} \equiv 11^{15} \equiv -4^{15} \cdot 3^{15} \pmod{23}$$

Com isso, teremos:

$$415 \equiv 230 \equiv (2^{11})^2 \cdot 2^8 \equiv 2^8 \equiv 3 \pmod{23}$$
$$315 \equiv 3^{11} \cdot 3^4 \equiv 3^4 \equiv 12 \pmod{23}$$

Concluindo assim:

$$11235 \equiv -415 \cdot 315 \equiv -3 \cdot 12 \equiv 10 \pmod{23}$$

Temos assim as congruências $34^{235} \equiv 0 \pmod{17}$ e $34^{235} \equiv 10 \pmod{23}$. Com isso podemos aplicar o teorema chinês do resto no sistema:

$$x \equiv 0 \pmod{17}$$
$$x \equiv 10 \pmod{23})$$

Com ele nós iremos obter que

$$10 + 23y \equiv 0 \pmod{17})$$

Obtendo assim:

$$6y \equiv 7 \pmod{17}$$

Porém, 3 é o inverso de 6 no módulo 17, e por isso teremos

$$y \equiv 3 \cdot 7 \equiv 4 \pmod{17}$$

Com isso iremos chegar até $x = 10 + 23y = 10 + 234 = 102$. Caso você venha a conferir na seção sobre codificação de mensagens poderá confirmar o resultado. Os demais blocos podem ser decodificados da mesma forma, apenas não serão mostradas nesta obra por necessitar de muitos passos, o que tornaria este capítulo inutilmente mais longo.

6.3 Provando a funcionalidade do RSA

Ao longo dessa seção vamos provar que o RSA funciona no processo de decodificação, para podermos fazer isso tudo o que teremos que fazer é provar que:

$$b \equiv DC(b) \pmod{n}$$

Nós já vimos que $C(b) \equiv b^3 \pmod{n}$ e $D(a) \equiv a^d \pmod{n}$. Se combinarmos ambos as congruência iremos obter

$$D(C(b)) \equiv D(b^3) \equiv b^{3d} \pmod{n}$$

Logo o que nós temos de provar é que $b^{3d} \equiv b \pmod{n}$. Como por definição $3d \equiv 1 \pmod{(p-1)(q-1)}$, o que nos leva até:

$$3d = 1 + k(p-1)(q-1)$$

Pelo Teorema Chinês do Resto e levando em conta a expressão para obter $3d$ temos que:

$$b^{3d} \equiv b(b^{p-1})^{k(q-1)}$$

Tomando que p não divide b, nós podemos vir usar o Teorema de Fermat, de modo a obter:

$$b^{p-1} \equiv 1 \pmod{p}$$

Obtendo assim:

$$b^{3d} \equiv b \cdot (1)^{k(q-1)} \equiv b \pmod{p}$$

Mesmo que tenhamos que b seja múltiplo de p, nós teremos que b e b^{3d} são congruentes a 0, logo nessa caso também é válida a congruência, tendo assim:

$$b^{3d} \equiv b \pmod{p}$$

Pelo mesmo método podemos obter q, obtendo o par:

$$b^{3d} \equiv b \pmod{p}$$
$$b^{3d} \equiv b \pmod{q}$$

Veja que b é solução de:

$$x \equiv b \pmod{p}$$
$$x \equiv b \pmod{q}$$

Pelo Teorema Chinês do resto esse sistema tem solução igual:

$$b + p \cdot q \cdot t$$

Onde $t \in \mathbb{Z}$. Logo como provamos anteriormente temos que:

$$b^{3d} \equiv b + p \cdot q \cdot k$$

Para algum inteiro k. Isso equivale a $b^{3d} \equiv b \pmod{p}$. Isso comprova aque $b = D(C(b))$.

6.4 Discutindo a segurança do RSA

Antes de mudarmos o foco deste trabalho, vamos nos prestar atenção no que tange a segurança do RSA. Vamos supor que alguém, que vamos denominar Irineu, esteja com uma escuta em nossas mensagens, tendo assim acesso tanto a mensagem codificada quanto a chave pública n. Vamos lembrar que n é a multiplicação dos primos p e q. Sabendo disso bastaria para Irineu fatorar n para obter p e q e depois descobrir d para poder decodificar a mensagem, como já foi explicado neste capítulo.

Isto pode parecer muito simples, mas como já mostramos na seção sobre fatoração que não há um algoritmo conhecido que possa fazer isso de forma eficiente. O que ocorre é que um algoritmo que faça a fatoração de forma eficiente pode vir a surgir a qualquer momento, do ponto que não há nenhuma prova matemática de que esse algoritmo não exista.

O que iremos fazer nos próximos capítulos consiste em propor uma variação da criptografia RSA que não seja completamente vulnerável caso tal algoritmo seja descoberto, a *Criptografia RSA Gaussiana*.

Capítulo 7

Inteiros e Primos de Gauss

Até o momento apenas os números inteiros foram neste projeto, mas para podermos entender a RSA Gaussiana é necessário conhecer os inteiros gaussianos. Neste capítulos vamos apresentar os inteiros e os primos gaussianos e suas propiedades básicas.

7.1 Inteiros de Gauss e suas propiedades

O inteiros gaussianos, que a partir de agora iremos nos referenciar por $Z[i]$, são um subconjuntos dos números complexos, relembrando que os números complexos são os números de forma $a+b$i, onde a eb são reais e i é a $\sqrt{-1}$. A diferença entre o conjunto $Z[i]$ e o conjunto C reside no fato de em $Z[i]$ a e b serem números inteiros.

Por $Z[i]$ estar contido em C, as operações deste conjunto podem ser realizadas, por exemplo, se tomarmos $z_1 = a + b$i e $z_2 = c + d$i nós iremos obter:

$$z_1 + z_2 = (a + c) + (b + d)\boldsymbol{i}$$
$$z_1 \cdot z_2 = (ac - bd) + (ad + bc)\boldsymbol{i}$$

Outra propiedade herdada é a dos elementos neutros, o $0 = 0 + 0$i continua sendo o elemento neutro da adição. O $1 = 1 + 0$i também continua sendo o elemento neutro da multiplicação. As propiedades associativa da adição e da multiplicaçõ, comutativa da adiçõ e multiplicaçõ e distributiva também são herdadas do conjunto C.

Observe que todo inteiro n tem no conjunto $Z[i]$ uma notação na forma $n+0$i que pode ser suprimida. Feito isso vamos nos ater aos critérios de divisibilidade em $Z[i]$. Vamos fatorar o número 5, que no conjunto Z é primo:

$$(1+2\boldsymbol{i}).(1-2\boldsymbol{i}) = 1-2\boldsymbol{i}+2\boldsymbol{i}-4\boldsymbol{i}^2 = 1-4(-1) = 5$$

Preste atenção ao fato de que os números primos do conjunto inteiro não são necessariamente primos do conjunto gaussiano.

Vamos definir o que vem a ser divisibilidade em $Z[i]$ para que possamos progredir. Vamos supor que x e y perteçam a $Z[i]$ e sejam diferentes entre si e diferentes de 0. Dizemos que y divide x e indicamos na forma de $y|x$, se e somente se existe um inteiro gaussiano w tal que $x = yw$. Tome de exemplo $(1+\mathbf{i})|2$, pois $2 = (1+\mathbf{i})(1-\mathbf{i})$ e $(1+\mathbf{i})|(1-\mathbf{i})$, pois $1+\mathbf{i} = \mathbf{i}(1-\mathbf{i})$.

Como os resultados das fatorações não se equivalem, seria muito útil a nós saber se a definição de divisibilidade é a mesma tanto nos inteiros quanto nos gaussianos. Para podermos checar isso vamos supor que x e y existem em Z e que $y|x$ em $Z[i]$. Com isso deve existir em $Z[i]$ um número $w = c+d\mathbf{i}$ tal que $x = wy$, ou seja $x = (c+d\mathbf{i})y = cy+d\mathbf{i}y)$. Com isso temos que $x = cy$ e $0 = dy$, o que implica em $d = 0$ e $w = c$, o que faz de w um membro do conjunto Z. Logo $x = wy = cy$ e com isso concluímos que se $y|x$ em $Z[i]$, ele também o faz em Z.

Sabemos que ± 1 dividem qualquer elemento da conjunto inteiro, analogamente ± 1 e $\pm\mathbf{i}$ fazem isso no conjunto complexo e gaussiano, esses números são as unidades básicas do conjunto. Sendo w uma unidade de inteiro gaussiano, e x e y inteiros gaussianos tais que $x = wy$ dizemos que x e y são elementos associados.

Agora que sabemos sobre os elementos associados e as unidades básicas podemos definir os primos gaussianos. Um *primo gaussiano* é um inteiro gaussiano que é diviível apenas pelos seus elementos associados e pelas unidades de $Z[i]$. Além de possuir primos, assim com em Z eles são infinitos, isso lhe será mostrado mais a frente.

7.2 Fatoração única

A fatoração única é a propiedade base de toda a Teoria de números, para que possamos construir um algoritmo RSA gaussiano tal qual desejamos se torna necessária que essa propiedade esteja presente no conjunto de inteiros gaussianos. Para podermos entender se isso é viável ou não, antes devemos conhecer que a *norma* de um número gaussiano $x = a + b$i é igual a $a^2 + b^2$, a função da norma é verificar relações de semelhança e diferença no conjunto gaussiano e seu símbolo N(x).

Antes de provarmos a fatoração única, provemos que todo o inteiro de Gauss com norma maior que 1 pode ser escrito como produto e um ou mais primos de Gauss. Se $N(x) = 2$, como 2 é primo e a norma multiplicativa temos que 2 é primo. Da mesma forma podemos estender para $N(x) > 2$, se x é primo a fatoração será imediata, se x não for primos nós teremos que $x = a \cdot b \Rightarrow N(x) = N(a) \cdot N(b)$, com $N(a), N(b) > 1$, logo $N(a), N(b) < N(x)$. Podemos supor que $N(y) < N(x)$, y é fatorável. Logo a, b e x tambem são fatoráveis.

Agora iremos provar a fatoração única, para isso, vamos considerar as fatorações $p_1 \cdot p_2 \cdot ... \cdot p_n$ e $q_1 \cdot q_2 \cdot ... \cdot q_m$, sendo ϵ uma unidade que implica em que a sequencia(p_i) seja uma permutação, exceto em casos de multiplicação por unidade, de (q_i). Se $max(m; n) = 1$, o resultado será imediato. Supondo que ele vale se $max(n'; m') < max(m; n)$, pelo Lema de Euclides, que diz que se n é um número inteiro e divide um produto ab e é primo entre si com um fator, então n divide o outro fator, vemos que para algum $i, p_n|q_i$.

Para não perdermos a generalidade vamos tomar que $i = m$. Como p_n q_m são primos, então $q_m = \epsilon' p_n$, com ϵ' sendo uma unidade. Logo $p_1 \cdot p_2 \cdot ... \cdot p_n = \epsilon' q_1 \cdot q_2 \cdot ... \cdot q_m \Leftrightarrow p_1 \cdot p_2 \cdot ... \cdot p_{n-1} = \epsilon\epsilon' q_1 \cdot q_2 \cdot ... \cdot q_{m-1}$. Como $p_1 \cdot p_2 \cdot ... \cdot p_n$ é uma permutação de $q_1 \cdot q_2 \cdot ... \cdot q_m$, exceto em casos de multiplicação por unidades, fica provada por indução a fatoração única dos inteiros gaussianos.

7.3 Primos de Gauss

Neste capítulo veremos quem são os números considerados primos em $Z[i]$, os famosos primos de Gauss. Observe que se $N(\pi)$ é primo em Z, nós teremos π sendo primo em $Z[i]$, de acordo com a demontração de fatoração única.

Atente-se ao fato de que todo o primo π divide $N(\pi)$, portanto ele deve dividir ao menos um fator primo em Z de $N(\pi)$. Caso π venha a dividir dois fatores distintos x e y, ambos primos em Z, nós teríamos que $x|1$, o que seria um absurdo. Com isso é possível se concluir que todo o primo de Gauss divide 1 e somente 1 primo inteiro positivo(além de se oposto negativo).

Considerando o caso acima e tomando o número primo com um inteiro positivo p, nós temos três casos que podemos prestar atenção. O primeiro ocorre quando p é par, sendo que nesse caso $p = 2$. Nesse caso obtemos como primos gaussianos $1 + i$,$1 - i$,$-1 + i$e$-1 - i$

Caso tomemos $p \equiv 3(mod4)$ sempre viremos a obter números primos. Já no caso de $p \equiv 1(mod4)$, apenas os casos em que $a^2+b^2 = p$ resultam em números primos gaussianos.

Como toda a criptografia de chave pública necessita de um conjunto de chaves, foi-se definido que os números primos gaussianos viriam a ser as chaves para a criptografia RSA Gaussiana. No próximo capítulo será apresentado o que já está feito neste algoritmo e o que ficará como implicação futura para desenvolvimento.

Nosso próximo passo consiste na encriptação, embora ela ainda não possua um algoritmo concluÃdo, assim como a desencriptação ela já possui um esboço

Capítulo 8

RSA Gaussiano e Conclusões

Chegamos ao último capítlo desta obra, aqui será debatido sobre tudo o que foi alcançado até o momento no que diz respeito ao RSA Gaussiano, veremos como sua forma é planejada e o que terá de ser feito no futuro para que este algoritmo se torne uma opção entre os algoritmos criptográficos.

8.1 O RSA Gaussiano e o que falta para ele ser utilizável

Ao longo desse seção lhe será apresentado como o RSA Gaussiano deverá vir a funcionar. Para iniciarmos, teremos que, assim como na criptografia RSA fazer uma pré-encriptação vindo a transformar todas as letras em número inteiros, da mesma forma que já ocorre.

A segunda parte do processo, que é a encriptação depende de algumas garantias as quais ainda não possuímos. Embora saibamos que a propiedade da fatoração única é válida para o conjunto $Z[i]$. Um dos meios para que possamos seguir consiste em definir a operação de congruência modular para o conjunto gaussiano, também se fazem necessárias propiedades análogas ao Teorema de Fermat e ao Teorema chinês do resto para que possamos seguir a mesma linha de encriptação do algoritmo RSA.

Para o RSA Gaussiano é planejada a encriptação com a fórmula:

$$C'(a) \equiv a^3 (mod n')$$

onde a seria o número a ser encriptado, n' seria a chave pública gaussiana, derivada de p' e q', que são números primos gaussianos. Os números p' e q' são as chaves privadas de encriptação.

Para a desencriptação, embora ainda não tenhamos uma prova de seu funcionamento pelos motivos já citados acima, ela é planejada em um primeiro momento pelo número d', que pode manter sua fórmula similar a do RSA ou não. Caso ele não precise de mudanças por conta do conjunto gaussiano, deverá ser da seguinte fórmula:

$$3d' \equiv 1 (mod((p'-1)(q'-1)))$$

Supondo que a fórmula de d' acima seja válida, teremos que para podermos concluir desencriptação a fórmula análoga a original, que seria:

$$D'(b) \equiv b^{d'} (mod n')$$

Nesta fórmula temos que b é um número encriptado pelo RSA Gaussiano, d' a constante cuja fórmula foi mostrada anteriormente, n' a chave pública e $D'(b)$ o resultado da desencriptação.

Com isso concluímos este projeto, aguardando que muito em breve tudo o que ficou como trabalho futuro aqui venha a ser realizado, muito obrigado por sua atenção a este projeto e esperamos que ele venha a ser uma fonte de inspiração para seus projetos futuros.

Referências Bibliográficas

[Cou07] Coutinho, Severino Collier: Criptografia. Rio de Janeiro, IMPA/SBM, Programa de Iniciação Científica da OBMEP, 2007.

[DH76] Diffie, Whitfield e Martin Hellman: New directions in cryptography. IEEE transactions on Information Theory, 22(6):644–654, 1976.

[Gau15] Gauss, Carl Friedrich: Methodus nova integralium valores per approximationem inveniendi. apvd Henricvm Dieterich, 1815.

[mil] Millennium Problems — Clay Mathematics Institute. `http://www.claymath.org/millennium-problems`. Acessado em 15/11/2016.

[PR13] Pacci, Daniel Campolina e Camila Takeuti Vaz Rodrigues: Inteiros De Gauss. 2013.

[Rie59] Riemann, Bernhard: Ueber die Anzahl der Primzahlen unter einer gegebenen Grosse. Ges. Math. Werke und Wissenschaftlicher Nachlas, 2:145–155, 1859.

[RSA78] Rivest, Ronald L, Adi Shamir e Leonard Adleman: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2):120–126, 1978.

[SF09] Sinkov, Abraham e Todd Feil: Elementary cryptanalysis, volume 22. MAA, 2009.

[Sha49] Shannon, Claude E: Communication theory of secrecy systems. Bell system technical journal, 28(4):656–715, 1949.