Universidade Federal de Sergipe
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA
MESTRADO PROFISSIONAL EM MATEMÁTICA
EM REDE NACIONAL - PROFMAT

Introdução à Matemática do Criptossistema RSA

Por

Welington Batista Luz

Mestrado Profissional em Matemática - São Cristóvão - SE

Abril de 2013

Universidade Federal de Sergipe
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA
MESTRADO PROFISSIONAL EM MATEMÁTICA
EM REDE NACIONAL - PROFMAT

Welington Batista Luz

Introdução à Matemática do Criptossistema RSA

Trabalho apresentado ao Departamento de Matemática da Universidade Federal de Sergipe como requisito final para a obtenção do título de Mestre em Matemática pelo PROFMAT

Orientador: Profº. Dr. Kalasas Vasconcelos de Araújo

São Cristóvão - Sergipe
Abril de 2013

Catalografica.png

**FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE**

Luz, Welington Batista
L979i Introdução à Matemática do criptossistema RSA / Welington Batista Luz; orientador Kalasas Vasconcelos de Araújo. – São Cristóvão, 2013.
50 f.: il.

Dissertação (Mestrado Profissional em Matemática em Rede Nacional – Profmat) – Universidade Federal de Sergipe, 2013.

1. Criptografia. 2. Álgebra. 3. Algoritmos. 4. Aritmética. I. Araújo, Kalasas Vasconcelos de, orient. II. Título

CDU 501:681.18:004.056

UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA
MESTRADO PROFISSIONAL EM MATEMÁTICA EM REDE NACIONAL

Dissertação submetida à aprovação pelo Programa de Pós-Graduação em Matemática da Universidade Federal de Sergipe, como parte dos requisitos para obtenção do grau de Mestre em Matemática.

Introdução a matemática do criptossistema RSA

por

Wellington Batista Luz

Aprovada pela Banca Examinadora:

Prof. Dr. Kalasas Vasconcelos de Araújo - UFS
Orientador

Prof. Dr. Rodrigo Gondim Neves - UFRPE
Primeiro Examinador

Prof. Dr. Zaqueu Alves Ramos - UFS
Segundo Examinador

São Cristóvão, 11 de abril de 2013

Cidade Universitária "Prof. José Aloísio de Campos" – Av. Marechal Rondon, s/no - Jardim Rosa Elze – Campus de São Critóvão. Tel. (00 55 79) 2105-6986 – Fax (0 xx 55 79) 2105-6566
CEP: 49100-000 - São Cristóvão – Sergipe - Brasil – E-mail: promat_ufs@yahoo.com.br

Sumário

Dedicatória **v**

Agradecimentos **vi**

Epígrafe **viii**

Resumo **ix**

Abstract **x**

Introdução **1**

1 Criptografia **3**

1.1 O que é Criptografia? 3

1.2 Objetivos da Criptografia 4

1.3 Criptografia moderna 6

1.4 Sistema Criptográfico Assimétrico 8

2 Teoria dos Números para o RSA **9**

2.1 Algoritmos 9

2.2 Divisão com resto e divisibilidade 9

2.3 Números primos 10

2.4 Expansão binária de um número inteiro 12

2.5 Algoritmo Euclidiano Estendido 12

2.6 Noções de Aritmética Modular 14

2.6.1 Congruência 14

2.6.2 Sistema Completo de Restos 16

2.6.3 Inversão módulo n 16

2.6.4 Exponenciação rápida 17

2.7 Pequeno Teorema de Fermat 17

3 O Algoritmo RSA **20**

3.1 Generalização de Euler 20

3.2 A magia do RSA – Como a mensagem codificada por Alice é recuperada por Bob . . . 21
3.3 Implementação matemática do Algoritmo RSA . . . 22
3.3.1 Pré-codificação do RSA . . . 22
3.3.2 Codificação do RSA . . . 23
3.4 Decodificação do RSA . . . 25

4 Implementação RSA com MAXIMA **27**

5 Segurança e Criptoanálise do RSA **30**
5.1 Algoritmo da fatoração . . . 30
5.2 Eficiência do Algoritmo da fatoração . . . 32

6 Encontrando primos - Testes de primalidade **33**

A Comandos do MAXIMA para o RSA **36**

B Tabela ASCII imprimíveis maiúsculos **37**

Referências Bibliográficas **38**

Dedicatória

Dedico este trabalho a três pessoas muito especiais:

Meu pai, **Eugênio Batista Luz** (*in memorian*).

Minha mãe, **Judite Cândido Santos** (*in memorian*).

Com a simplicidade de agricultores, eles me ensinaram que é possível vencer o jogo da vida e realizar os sonhos sem perder a alma! Eternas saudades!

Meu avô materno, **Manoel Cândido Santos** (*in memorian*).

Um agricultor, vaqueiro e canoeiro forte e obstinado, de quem eu muito me orgulho!

Welington Batista Luz
abril de 2013

Agradecimentos

Agradeço a Deus por tudo! Deus existe, porque a Matemática existe!

Agradeço à CAPES - Coordenação de Aperfeicoamento de Pessoal de Nivel Superior, pelo financiamento do PROFMAT.

À SBM - Sociedade Brasileira de Matemática, pela competente coordenação do programa, nas pessoas do Prof. Hilário Alencar e do Prof. Marcelo Viana.

Ao IMPA - Instituto de Matemática Pura e Aplicada, pelo apoio humano e institucional ao PROFMAT.

À UFS - Universidade Federal de Sergipe, por ter apoiado e se integrado ao programa, disponibilizando sua estrutura humana, material e logística. Em especial, aos competentes e dedicados professores do Departamento de Matemática da UFS: Paulo Rabelo, Fabio dos Santos, Naldisson dos Santos, Almir Rogério Silva Santos, Anderson dos Santos, Éder Mateus, Evilson da Silva Vieira, Arlúcio Viana, Wagner Ferreira Santos, Natanael Oliveira Dantas, Marco Aurélio Guimarães Monteiro, Danilo Felizardo Barbosa, e Kalasas Vasconcelos de Araújo, este especialmente por aceitar me orientar neste trabalho. E aos professores Rodrigo Godim Neves da UFRPE e Zaqueu Alves Ramos da UFS, por aceitarem participar da minha banca examinadora.

Ao Povo de Aracaju, então representado pelo prefeito Edvaldo Nogueira e pelo Secretário Municipal de Educação e Desporto Antônio Bitencourt Júnior, pela licença para cursar o PROFMAT.

Aos amigos Adailton Novais, Rodolfo Sampaio Cassaca, Geilson Leão e Marcone Augusto, cujas inteligências e cultura científica não comprometem a educação e a simplicidade! Pela ajuda e pela torcida!

Ao amigo Leopoldo Ramos de Oliveira, companheiro de duras jornadas! Pelo exemplo, e o apoio moral e afetivo.

Ao amigo Gerson Silva Ávila, companheiro de duríssimas jornadas e grandes alegrias. Pela torcida e pelo incentivo sincero! Também à sua esposa Eutímia Nascimento Santos Ávila e sua filha e minha afilhada Laura Ávila (Laurinha). E também a sua mãe dona Maria Silva Àvila pelo carinho e consideração de sempre!

Ao amigo e conterrâneo Francisco Araújo Filho e à sua noiva Marinalva dos Santos, pelo companheirismo e pela torcida em todos os desafios.

Ao amigo e conterrâneo Paulo Anderson Santana Rocha, pela confiança, companhei-

rismo, torcida e incentivo!

Aos amigos e conterrâneos Isaias Santos Vilela e Ana Vilela, Edivaldo Santos e Marilene Santos Vilela, pela consideração e torcida sempre.

Aos amigos Ismael Ferreira e sua esposa Lucicleide Cavalcante Ferreira, pela torcida e receptividade calorosa.

Ao amigo Edmundo Lisboa de Araújo e sua esposa Fátima Araújo Lisboa, por ter acreditado em mim desde o primeiro encontro. Você é magnânimo!

À família Simões, na pessoa da minha sogra D. Pureza Simões, em especial à minha esposa Vânia Alves Simões, que suportou com paciência durante dois anos o efeito colateral do PROFMAT. Sem o seu carinho e compreensão as coisas teriam sido bem mais difíceis. Obrigado por tudo! Te amo! E a Thiago Simões, seu filho e meu enteado, pelo respeito e torcida.

À minha prima e madrinha por adoção Maria José Maia de Santana, uma pessoa a quem eu sou muito grato pelo carinho nos tempos difíceis! E também a seu marido Joaquim Reis de Santana e seus filhos Marize Maia de Santana (*in memorian*), Kátia Cilene Maia de Santana, Joaquim Maia de Santana Filho e Mário Reis Maia de Santana. Apesar da distância, estamos juntos! Sei que vocês torcem por mim!

À minha irmã Maria José Batista Luz de Lima e seu esposo João José de Lima. A meu irmão Antonio Batista Luz, o meu primeiro professor de Matemática! E a sua esposa Isabel Cristina Correia Luz e filhos Robson Correira Luz e Rodrigo Correia Luz. Obrigado pelo incentivo, pela ajuda e pelo carinho!

A todos colegas da EMEF Olga Benário do Município de Aracaju-SE, pela torcida, e aos colegas da DFDA-SE-MDA que tiveram que suportar minhas lamentações diariamente. Obrigado pela paciência! E também pela parceria e torcida!

A todos os meus mestres de hoje e de ontem, porque eles são parte da minha vida, dos meus sonhos e do meu sucesso! Infinitamente grato!

Por fim, agradeço aos meus valorosos colegas, companheiros e novos amigos desta dura e gratificante jornada que foi o PROFMAT nesses últimos dois anos: Elisabete Santana de Ávila e Silva, Lúcia Pereira dos Santos Gomes, Marcele Rodrigues Moreno Santos, Evani Machado de Mello, Jimmy Cavalcanti Alves, Isaac Pinheiro Mota, Edvaldo Araújo dos Reis, Elton Jones da Silva Magalhães, Elson Nascimento Lima, César Augusto Vieira Lima, Ávido Sadote de Barros Neto, Luis Anselmo dos S. Vasconcelos, Gilvan Andrade Santos, Carlos Alberto Barreto, André Bispo Calderaro, Márcio Monte Alegre de Souza, Davi Dantas Lima, Sérgio Ricardo dos Santos e José Hélio Barbosa. Com vocês eu aprendi um estranho axioma da matemática do coração: $1 + 1 >>> 2$. Foi uma honra, sorte e felicidade tê-los como meus colegas de mestrado! Sou grato por tudo!!!

Welington Batista Luz - abril de 2013

"Nada é mais importante do que ver as fontes da invenção que são, em minha opinião, mais interessantes do que as próprias invenções".

Gottfried Wilhelm Leibniz

Welington Batista Luz
abril de 2013

Resumo

Este trabalho tem por objetivo apresentar uma introdução à matemática subjacente ao Criptossistema RSA em um nível elementar, como motivação a uma abordagem mais concreta à Teoria dos números. Pretende também servir de introdução à criptografia moderna, uma vez que esta vem se constituindo em uma área altamente estratégica e fecunda tanto na Ciência da Computação quanto na Matemática. Serão abordadas as principais definições, conceitos e resultados referentes à Teoria Clássica dos Números que garantem o entendimento, o funcionamento e a segurança do Criptossistema RSA, e apresentada sua implementação matemática simplificada. Também será apresentada uma implementação prática do RSA utilizando o programa de computação algébrica MAXIMA. O conteúdo deste trabalho servirá de base à confecção de uma cartilha didática de atividades destinada aos alunos do ensino básico, onde serão trabalhados os temas acima citados.

Palavras chaves: Criptografia, RSA, Pequeno Teorema de Fermat, Números Primos, Testes de primalidade, Euler, Euclides, Gauss, Congruência, MAXIMA.

Abstract

This work purpose one introdution for the mathematics behind RSA Cryptossistem on an elementary level, as motivation to a more concrete approach to the theory of numbers. It also seeks to serve as an introduction to modern cryptography, since this is becoming an area of great strategic and fruitful both in Computer Science and in Mathematics. It will address key definitions, concepts and results related to the Classical Theory of Numbers that ensure understanding, operation and security of RSA Cryptographic Algorithm and submitted its implementation simplified mathematics. Also presented will be a practical implementation of RSA using the computer algebra program MAXIMA. The content of this work will serve as a basis for making a booklet of activities aimed at teaching elementary school students, where will be worked the thems mentioned above.

Key words: Cryptography, RSA, Fermat's Little Theorem, Primes, Primality Tests, Euler, Euclid, Gauss, Congruences, MAXIMA.

Introdução

A partir da década de 70 a Criptografia assumiu um papel essencial na garantia da segurança da "sociedade da informação", ao ser massivamente aplicada aos protocolos de comunicação da Internet. Como veremos ao longo deste trabalho, isto não seria possível sem a utilização da Teoria dos Números, uma área da Matemática até então tida como meramente abstrata e sem nenhuma aplicabilidade ao mundo real.

Considerando que a proposta é abordar a Matemática envolvida no RSA sob um ponto-de-vista elementar, destinada especialmente aos alunos do ensino básico, será utilizada uma linguagem mais natural possível, com o cuidado de se manter a exatidão e objetividade que o tema exige. Deve ser alertado que em muitas passagens a concisão foi propositalmente prejudicada em benefício da clareza. Isto pode a princípio parecer contraditório para alguém já familiarizado com o formalismo acadêmico matemático, mas não o é quando se quer assumir a ótica e as limitações teóricas do aluno do ensino básico.

Os resultados mais elementares da Aritmética serão considerados como já conhecidos e serão apenas apresentados, a exemplo da Divisibilidade no conjunto dos números inteiros e suas propriedades, o Algoritmo de Euclides Estendido, e a Aritmética Modular, uma vez que se encontram bastante difundidos em excelentes trabalhos introdutórios, a exemplo de Plínio[11], Hefez[9] e Silva[10]. Também o conceito de função será dado como conhecido. Já os resultados que embasam mais diretamente o Algoritmo RSA serão justificados, a exemplo do Pequeno Teorema de Fermat e a Generalização de Euler.

Há aqui o entendimento de que é necessário apresentar logo cêdo e de forma acessível os temas mais avançados da Matemática para o aluno do ensino básico como estratégia de motivação. Também entende-se necessário promover a interdisciplinaridade, tendo em vista a dinâmica cada vez mais intensa no Brasil da pesquisa científica integrada e o conseqüente aumento da demanda por matemáticos de alto nível capazes de interagirem com os demais campos do conhecimento.

No Capítulo 1 será feito um resumido panorama histórico da Criptografia Simétrica, necessário à justificativa da necessidade dos Sistemas Criptográficos Assimétricos, a exemplo do RSA. Em seguida, no Capítulo 2, serão apresentadas as principais definições, conceitos e teoremas referentes à Teoria Clássica dos Números que garantem o entendimento, o funcionamento e a segurança do Algoritmo RSA. No Capítulo 3, com base no que foi fundamentado nos capítulos ateriores, será explicada a implementação matemática

do RSA em linhas gerais, culminando com uma implementação prática e simplificada do Algoritmo RSA. No Capítulo 4 será trabalhada uma aplicação mais realista do RSA com a utilização do MAXIMA.

No Capítulo 5, será abordado o Algoritmo da Fatoração e feita uma análise de sua eficiência, bem como sua importância e limitações na criptoanálise do RSA. E por fim, no Capítulo 6, será feita uma introdução aos testes de primalidade utilizados pelos programas de computação algébrica mais conhecidos. Também será estimada a quantidade de números primos de uma determinada grandeza utilizando-se o Teorema dos Números Primos.

Capítulo 1

Criptografia

1.1 O que é Criptografia?

Criptografia é a ciência da escrita secreta, cuja finalidade é o estudo das técnicas que permitem escrever uma mensagem legível em um modo inteligível apenas para o seu autor ou alguém autorizado por ele.

A mensagem que se pretende esconder com criptografia pode ser um texto, uma frase, uma palavra ou até mesmo um número.

Inicialmente, a Criptografia foi praticada pelas civilizações clássicas, a exemplo da egípcia, grega, romana, chinesa, etc... e depois pelos povos modernos, geralmente ligada aos assuntos do Estado, tais como espionagem, guerras e diplomacia. Atualmente ela se encontra bastante disseminada em diversos campos da atividade humana, especialmente para garantir o sigilo dos nossos dados e conteúdos.

A palavra cripto vem do grego "criptos", que significa escondido, e a palavra grafia vem do grego "graphein", que significa escrever. Portanto, "Criptografia" quer dizer "escrita escondida". Escondida no sentido de que algum "algoritmo criptográfico" é usado para "embaralhar" a mensagem, de modo que ou a posição das suas letras são alteradas, ou as letras são substituídas por outros símbolos. Por exemplo, se na palavra "PROFMAT" trocarmos cada letra pela terceira seguinte na ordem alfabética, obteremos "SURIPDW". Já na "Esteganografia", a escrita é "escondida" no sentido de que ela é ocultada por algum processo físico-químico, a exemplo da conhecida "tinta invisível", onde se escreve com suco de limão sobre uma folha de papel branca. Após esta secar, é só aquecer a folha em contato com uma chama que a escrita aparece magicamente.

Quando substituímos uma única letra de um texto por outra do mesmo alfabeto ou por um símbolo qualquer, temos uma cifra. Já quando substituímos uma palavra inteira por um símbolo, temos um código. No entanto, neste texto, será usado indistintamente os termos cifra e código como sinônimos. Consequentemente, neste trabalho, codificar será sinônimo de cifrar.

Para podermos codificar uma mensagem precisamos de, no mínimo, três coisas: um "alfabeto simbólico" usado para codificar o texto legível; um "algoritmo criptográfico", um conjunto de instruções que nos diz como a mensagem será codificada; e uma "chave", que é uma informação necessária tanto para codificar quanto para "inverter" o algoritmo criptográfico e obter o texto legível original.

No exemplo visto acima, conhecido como Cifra de César, o alfabeto simbólico é o próprio alfabeto da mensagem legível (A, B, C, ..., X, Y, Z). O algoritmo criptográfico é a instrução que manda trocar cada letra da mensagem por uma única letra do mesmo alfabeto, deslocando algumas posições para a direita na ordem alfabética, e a "chave" é o número 3, pois devemos substituir cada letra da mensagem pela terceira seguinte na ordem alfabética, permitindo-nos assim obter a tabela de substituição abaixo e inverter o processo de codificação.

a	b	c	d	e	f	g	h	i	j	k	l	m
D	E	F	G	H	I	J	K	L	M	N	O	P
n	o	p	q	r	s	t	u	v	w	x	y	z
Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Geralmente a chave criptográfica é dada em termos numéricos, e quanto maior o seu tamanho e sua quantidade, mais forte é o algoritmo criptográfico. Por exemplo, na Cifra de César a chave tem no máximo dois dígitos e só há 25 chaves possíveis, o que é obviamente um sistema criptográfico fraco. Mas se admitíssemos substituir qualquer letra do nosso alfabeto (26 letras) por qualquer outra do mesmo alfabeto, teríamos uma cifra permutacional que admitiria $26! = 403291461126605635584000000$ chaves diferentes. No entanto, a aparente força da cifra permutacional é vulnerável à "análise de frequências", uma técnica criada pelos matemáticos árabes. Esta técnica consiste em se contar as frequências dos símbolos em um texto cifrado, com base na constatação de que a frequência de uma letra em um idioma é aproximadamente a mesma em textos diferentes, caso esses textos sejam suficientemente longos e aleatórios. Por exemplo, na língua portuguesa, a letra "a" é a mais frequente. Então é muito provável que em um texto longo cifrado do português, de uma cifra permutacional qualquer, o símbolo mais frequente represente a letra "a" do texto legível.

1.2 Objetivos da Criptografia

O objetivo da Criptografia é proteger o conteúdo de uma mensagem da curiosidade e do interesse de pessoas não autorizadas. Como sabemos, a informação é uma matéria prima e ao mesmo tempo um produto muito caro e estratégico. Informação produz conhecimento, e conhecimento é poder!

Portanto, a Criptografia garante principalmente o sigilo de uma mensagem ou de dados armazenados, isto é, só o autor ou alguém autorizado por ele terá acesso ao conteúdo do texto criptografado. A Criptografia também pode garantir a autenticidade de um documento, quando é usada para criptografar a assinatura digital do seu titular, certificando assim que a assinatura do documento pertençe de fato ao seu remetente.

Imaginemos que alguém está realizando uma compra em um site da internet e precisa enviar à empresa vendedora o seu número de cartão de crédito. Se um "hacker" estiver "fuçando" esta transação e descobrir os dados trafegados, esta pessoa estará em apuros, a menos que os dados enviados estejam criptografados.

Outra situação terrível é alguém ter seu "notebook", "tablet" ou "pendrive" roubados, o que deixaria seus documentos pessoais, agenda telefônica, fotografias, vídeos, etc..., totalmente à disposição de quem estiver na posse do seu equipamento, caso não estejam criptografados.

Pior, imaginemos agora que as informações confidenciais pertencem à empresa ou à instituição em que uma pessoa trabalhe, por exemplo, uma pesquisa de um produto inovador, informações judiciais, policiais, financeiras, etc...? Ou que as informações são um segredo de Estado, como os códigos que acionam uma arma nuclear? Como vemos, são muitos e diversos os usuários da Criptografia e diversas suas aplicações, sejam elas de interesse privado ou público.

Mas a exemplo de quase tudo nesta vida, existem também usuários maléficos da Criptografia: o crime organizado, o terrorismo e os pedófilos. Eles também têm interesse em esconder seus planos do Estado. Em razão disto há um grande debate sobre o acesso dos cidadãos civis aos métodos criptográficos mais seguros e eficientes. O Estado argumenta que a criptografia forte protegerá o crime. Por outro lado, os cidadãos precisam usar a Criptografia para proteger sua privacidade de diversos intrusos, às vezes até do próprio Estado, quando este age de modo arbitrário e invasivo através de agentes públicos corruptos.

Nunca a Criptografia foi tão necessária como nos dias atuais, em que a humanidade está se conectando cada vez mais através da internet, constituindo a tão propalada "sociedade da informação", turbinada cada vez mais pela "convergência digital". Fala-se que em breve até a moeda física, como a conhecemos hoje, vai desaparecer, dando lugar ao "dinheiro eletrônico" viabilizado pelo uso de tecnologias cada vez mais "inteligentes". Fatalmente, todas as nossas transações financeiras serão eletrônicas, o que exigirá um nível elevado e ao mesmo tempo eficiente de criptografia para proteger nossas informações financeiras que fluirão livres pela internet, através de ondas elétricas e eletromagnéticas.

Mais do que uma necessidade tecnológica a serviço do Estado, a Criptografia vem se constituindo tambem no principal assegurador da maior consquista política da modernidade: a privacidade do cidadão!

1.3 Criptografia moderna

A era moderna da Criptografia teve início em 1948, com o nascimento da computação eletrônica e a publicação da Teoria dos Códigos de Claude Shannon, onde ele delineou os fundamentos matemáticos e físicos para a implementação eletrônica de códigos, através da sua tese de doutoramento intitulada "A mathematical theory of communication" (Teoria matemática da comunicação). A partir deste trabalho, a Matemática assumiu definitivamente as bases da Criptografia com a criação de sistemas criptográficos que se utilizam de funções matemáticas e da aplicação maciça dos resultados clássicos e modernos da Teoria dos Números, uma vez que os caracteres de um texto qualquer nada mais são do que números binários manipulados logicamente por um computador, podendo ser submetidos às operações aritméticas clássicas, tais como adição, multiplicação, potenciação, aritmética modular, etc....

O primeiro algoritmo criptográfico para uso geral, padronizado pelo NBS (National Bureau of Standards) do governo dos EUA, foi produzido e publicado pela IBM em 1976, denominado DES (Data Encryption Standard). Até então, os métodos criptográficos eram secretos e de uso apenas estatal ou empresarial. A partir de 2001, o DES foi substituído pelo AES (Advanced Encryption Standard), que é aplicado atualmente nas conexões Wi-Fi que nós usamos em nossos lares.

No entanto, o AES e todos os métodos de cifragem clássicos são "Sistemas Criptográficos Simétricos", ou de "Chave Privada", onde uma mesma chave K é utilizada tanto para codificar quanto para decodificar o texto enviado.

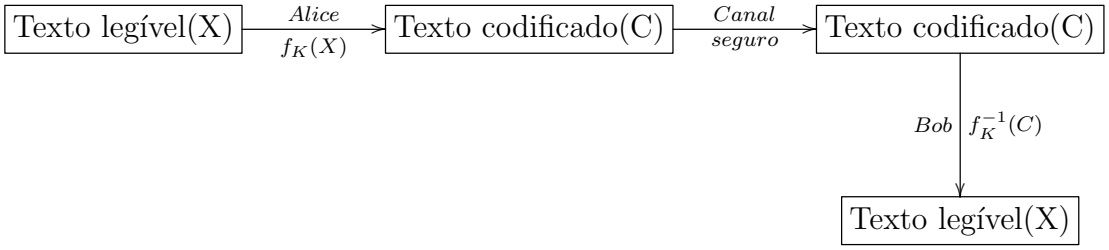
Sejam:

X: texto legível a ser codificado; C: texto codificado a ser transmitido;
K: chave privada;
f: função de codificação (ou algoritmo de codificação);
f^{-1}: função de decodificação (ou algoritmo de decodficação).

Em geral, um Sistema Criptográfico Simétrico, deve atender às seguintes condições mínimas:

1. Todos usuários utilizam a mesma chave K para codificar e decodificar a mensagem.
2. Se $f_K(X) = C$, então $f_K^{-1}(f_K(X)) = f_K^{-1}(C) = X$. Ou seja, ao aplicarmos o algoritmo de decodificação com a chave K à mensagem codificada C, recuperamos a mensagem legível X.
3. O cálculo de $f_K(X)$ e de $f_K^{-1}(C)$ é computacionalmente fácil quando se conhece as chaves K.
4. É "computacionalmente difícil" calcular $f_K^{-1}(C)$ sem o conhecimento de K.

Esquematicamente, temos:



O problema com os Sistemas Criptográficos Simétricos reside na distribuição da chave, que nos métodos implementados eletronicamente são feitos através dos canais eletrônicos (linha telefônica e ondas de rádio), vulneráveis à "escuta" de algum intruso. Se este souber o algoritmo de codificação e descobrir a chave utilizada, tudo estará perdido.

Imaginemos milhões de chaves fluindo desprotegidas pela internet sujeitas à interceptação de milhares de pessoas mal-intencionadas. Uma maneira de contornar este problema seria enviar a chave através de um "canal seguro", por exemplo, os correios ou mesmo um mensageiro particular, o que tornaria todo o processo muito lento e caro. Fica claro portanto que a troca de chaves é o aspecto mais crítico de qualquer sistema criptográfico simétrico.

É neste contexto que aparecem os sistemas criptográficos assimétricos, ou de chave pública, sugeridos pelos norte-americanos Whitfield Diffie e Martin Hellman em um trabalho revolucionário intitulado "New Directions in Cryptography" (Novas direções em Criptografia), publicado de 1976. Na criptografia assimétrica são usadas duas chaves, uma pública e outra privada, sendo que esta não transita pelos canais eletrônicos, ficando armazenada no computador do destinatário da mensagem, desde o momento em que é criada por ele, conforme veremos adiante.

Mas a característica principal da criptografia assimétrica é a implementação de uma "função bijetiva de mão única", ou seja uma função fácil de se computar em um sentido mas difícil de ser computada no sentido inverso, caso não se conheça a chave secreta. Existem vários processos assimétricos, a exemplo do famoso brinquedo Cubo Mágico. Sabemos que é muito fácil misturar as cores do Cubo girando suas faces, mas também sabemos que é "muito difícil" retornar as faces às suas cores originais, a menos que já se conheça os movimentos certos para tal. Mas será que um processo análogo poderia ser implementado aritmeticamente a ponto de se poder construir uma criptografia assimétrica forte? Felizmente a resposta é sim!

Inspirados por esta ideia, os pesquisadores do MIT Ronald Rivest, Adi Shamir e Leonard Adleman inventaram em 1978 o Criptosistema RSA, sendo este o primeiro sistema criptográfico de chave pública utilizado e ainda o mais seguro e o mais difundido atualmente. Por exemplo, o RSA é usado em SSL (Secure Socket Leyer), como uma

"camada" de segurança do http (Hypertext Transfer Protocol), também denominado de "https" (Hypertext Transfer Protocol Secure), e que aparece no início do endereço da URL (endereço eletrônico do site) quando estamos fazendo uma compra com cartão de crédito em uma loja eletrônica na internet, combinado com o DES.

1.4 Sistema Criptográfico Assimétrico

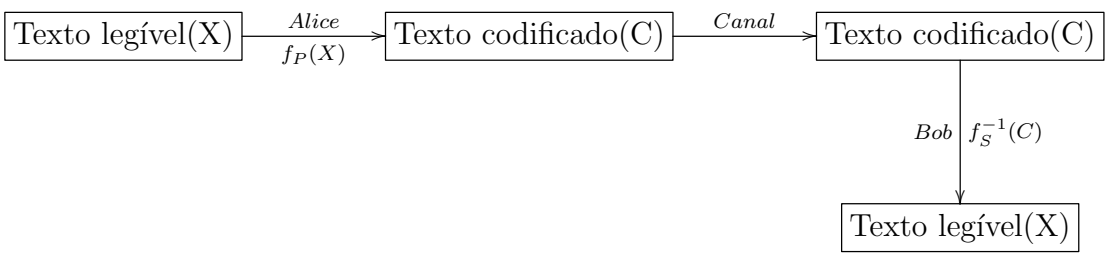
Sejam:

X: texto legível a ser codificado; C: texto codificado a ser transmitido;
P: chave pública; S: chave privada;
f: função de codificação (ou algoritmo de codificação);
f^{-1}: função de decodificação (ou algoritmo de decodficação);

De forma geral, os sistemas criptográficos assimétricos, ou de chave pública, devem atender às seguintes condições mínimas:

1. Cada usuário de um Sistema Criptográfico Assimétrico deve possuir um par de chaves (P, S).
2. Se $f_P(X) = C$, então $f_S^{-1}(f_P(X)) = f_S^{-1}(C) = X$. Ou seja, ao aplicarmos o algoritmo de decodificação com a chave privada S à mensagem codificada C, recuperamos a mensagem legível X.
3. É computacionalmente fácil calcular o par de chaves (P, S).
4. É computacionalmente difícil calcular a chave privada S a partir do conhecimento da chave pública P.
5. O cálculo de $f_P(X)$ e de $f_S^{-1}(C)$ é computacionalmente fácil quando se conhece as chaves S e P.
6. É computacionalmente difícil calcular $f_S^{-1}(C)$ sem o conhecimento de S.

Esquematicamente, temos:



O Criptosistema RSA foi o primeiro a atender os critérios acima para uma sistema criptográfico de chave pública, conforme veremos adiante.

Capítulo 2

Teoria dos Números para o RSA

2.1 Algoritmos

Como falaremos bastante sobre algoritmos nas páginas seguintes, convém sabermos logo o que eles realmente são. A palavra "algoritmo" originou-se do nome do grande matemático árabe Al Khowarismi (780 - 850), e significa, segundo a definição de Donald Knuth em [8], p.4, "(...) um conjunto finito de instruções em uma dada sequência de operações a fim de se resolver um problema específico, devendo apresentar cinco características essenciais: finitude, clareza, entrada, saída e efetividade". Além de ser finito em um tempo razoável, o algoritmo deve ser claro e cumprir realmente o que se quer com sua implementação.

Grande parte da Matemática consiste na implementação de algoritmos, a exemplo do Algoritmo de Euclides para o cálculo do máximo divisor comum. E qualquer programa de computador nada mais é do que sequências enormes concatenadas de algoritmos escritos em alguma linguagem que o computador pode "compreender" e implementar.

2.2 Divisão com resto e divisibilidade

O resultado que segue nos acompanha desde o quinto ano do ensino fundamental e é uma das ferramentas matemáticas mais importantes que existem. Euclides de Alexandria (360 — 295 a.C.) foi a primeira pessoa a dar uma demonstração formal para ele na sua monumental obra "Os elementos". Este teorema fundamenta o que é conhecido como Algoritmo da Divisão Euclidiana, ou ainda Algoritmo da Divisão com Resto. Euclides só conhecia e trabalhava com números positivos (medidas de segmentos de reta), mas aqui consideraremos também números inteiros negativos como dividendos.

i) "Sejam n e $d > 0$ inteiros. Então existem únicos q e r inteiros tal que $n = q \cdot d + r$, com $0 \leq r < d$." (Euclides)

Exemplos:

a) Sejam $n = 20$ e $d = 3$, temos $20 = 6 \cdot 3 + 2$.

b) Se $n = -10$ e $d = 7$, temos –10 $= (-2) \cdot 7 + 4$.

Vamos lembrar que, quando estamos "dividindo" através deste algoritmo, estimamos valores máximos para os quocientes, subtraímos o resultado do dividendo e obtemos restos, até que o último resto seja menor do que o divisor. Aí a divisão pára e temos portanto o nosso quociente e resto "únicos". Isto irá sempre acontecer, pois a sequência dos restos obtida é formada por números inteiros positivos e decrescentes.

Quando $r = 0$, temos que $n = q \cdot d$. Neste caso dizemos que d é divisor de n, ou que d divide n, ou que n é múltiplo de d, ou que n é divisível por d. Também dizemos neste caso que n pode ser fatorado nos fatores q e d. Exemplo: 5 é divisor de –20, pois $-20 = (-4) \cdot 5 + 0$. Da mesma forma, -4 é divisor de -20. Também poderíamos dizer que –20 é múltiplo (ou divisível) de 5 e de –4.

ii) "Sejam a, b, c, m e n inteiros, onde c divide a e c divide b. Então c divide $m \cdot a + n \cdot b$."

Exemplos:

a) Como 2 divide 8 e 2 divide 30, então 2 divide $5 \cdot 8 + 7 \cdot 30$.

b) Como 5 divide 15 e 5 divide 35, então 5 divide $12 \cdot 15 - 9 \cdot 35$.

2.3 Números primos

A palavra "primo" vem do latim "primus", e significa "os que vêm primeiro". O sentido de "primo" em Aritmética é de que todos os demais números inteiros, que não são primos, são formados pela multiplicação de números primos, como nos garante o Teorema Fundamental da Aritmética (Euclides/Gauss).

Por desempenhar um papel importante no algoritmo RSA, os números primos e algumas de suas propriedades fundamentais serão revisadas e justificadas a partir de agora.

Chamamos de primo ao número inteiro $p > 1$ que só possui os divisores 1 e p. Um número inteiro $p > 1$ que não é primo é dito composto.

Exemplos:

a) 3, 7 e 23 são primos.

b) 9, 84 e 11235 são compostos.

Os números primos menores do que 100 são:

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97.$$

iii) "Existem infinitos números primos." (Euclides)

Vamos supor que exista uma quantidade finita $p_1, p_2, \ldots, p_k$ de números primos. Formemos com eles o número $n = p_1 \cdot p_2 \cdot p_3 \cdot \ldots \cdot p_k + 1$. É obvio que um dos primos $p_1, p_2, \ldots, p_k$ deve dividir n. Este fator primo que divide n também divide o produto $p_1 \cdot p_2 \cdot p_3 \cdot \ldots \cdot p_k$. Mas isto é um absurdo, pois, pelo resultado **ii)**, este primo deve dividir a diferença $n - p_1 \cdot p_2 \cdot p_3 \cdot p_k = 1$, ou seja, deve dividir 1. Portanto, devem existir mais primos do que os $p_1, p_2, \ldots, p_k$. Isto equivale a afirmar que existem infinitos números primos! Lembremos que os gregos da época de Euclides não concebiam o "infinito".

Em outras palavras, este resultado nos diz que não pode haver uma "tabela periódica" dos números naturais, ou seja, finitos números que "gerem" (pela multiplicação) todos os outros números naturais. Isto é, os "elementos químicos" da Aritmética são infinitos!

iv) "Para todo inteiro positivo k, existem k números inteiros consecutivos todos compostos"

Consideremos a sequência de k números consecutivos:

$$(k+1)! + 2, (k+1)! + 3, (k+1)! + 4, (k+1)! + 5, \ldots, (k+1)! + k, (k+1)! + (k+1).$$

Cada termo dela é claramente um número composto, pois sao divisíveis por cada número inteiro de 2 a $k + 1$.

Exemplos:

a) 2 divide $2! + 2$.

b) 2 divide $3! + 2$ e 3 divide $3! + 3$.

c) 2 divide $4! + 2$, 3 divide $4! + 3$ e 4 divide $4! + 4$.

Este resultado nos informa que há intervalos muito longos (tão longo quanto queiramos) entre dois números primos. Ou seja, intervalos arbitrariamente grandes formados apenas por números compostos. Isso nos dá uma ideia da ordem de grandeza com que os números primos podem estar afastados quando avançamos na sequência dos números naturais. O que não quer dizer que não possa haver também intervalos contendo muitos números primos mesmo após intervalos longos de números compostos como estes.

2.4 Expansão binária de um número inteiro

v) Consideremos dois números positivos n e b, com $b > 1$.
Então n pode ser escrito na forma:

$$n = a_k \cdot b^k + a_{k-1} \cdot b^{k-1} + \ldots + a_2 \cdot b^2 + a_1 \cdot b + a_0,$$

onde $0 \leq a_k < b$, $0 \leq a_{k-1} < b, \ldots, 0 \leq a_2 < b$, $0 \leq a_1 < b$, $0 \leq a_0 < b$ **são únicos** e $a_k \neq 0$.

Aplicando o Algoritmo Euclidiano da Divisão, temos: (I) $n = q_1 \cdot b + a_0$, onde $0 \leq a_0 < b$. Se $q_1 \geq b$, podemos dividi-lo por b, obtendo: (II) $q_1 = q_2 \cdot b + a_1$, onde $0 \leq a_1 < b$. Substituindo (II) em (I), temos: $n = (q_2 \cdot b + a_1) \cdot d + a_0 = q_2 \cdot b_2 + a_1 \cdot b + a_0$, onde $0 \leq a_0, a_1 < b$. Continuando assim até que $q_k < b$, para um k inteiro positivo, quando paramos de dividir e obtemos $n = q_k \cdot b^k + a_{k-1} \cdot b^{k-1} + \ldots + a_2 \cdot b^2 + a^1 \cdot b + a_0$. Como q_k é o último resto, fazemos $q_k = a_k$ e, portanto:

$$n = a_k \cdot b^k + a_{k-1} \cdot b^{k-1} + \ldots + a_2 \cdot b^2 + a_1 \cdot b + a_0$$

onde $0 \leq a_k, a_{k-1}, \ldots, a_2, a_1, a_0 < b$ são "únicos".

Neste caso dizemos que $a_k, a_{k-1}, \ldots, a_2, a_1, a_0$ são os algarismos de n na base b.

O número n também pode ser representado como $n = (a_k a_{k-1} a_2 a_1 a_0)_b$

Para obtermos a expansão de um número n em uma base qualquer b, aplicamos o algoritmo da divisão euclidiana sucessivamente entre os quocientes obtidos e a base b. Os algarismos de n serão os restos obtidos na ordem contrária.

Quando a base $b = 2$, dizemos que a expansão de n é binária. Quando $b = 10$ dizemos que é decimal. Neste caso, não é preciso indicar a base subscrita. Por exemplo $(32)_{10} = 32$.

Exemplo: Determinar a expansão binária do número 37.

Inicialmente dividimos 37 por 2, obtendo $37 = 18 \cdot 2 + 1$. Em seguida dividimos 18 por 2, obtendo $18 = 9 \cdot 2 + 0$. E prosseguimos dividindo os quocientes obtidos por 2 até obter um quociente menor do que 2, quando não poderemos mais continuar a divisão com resto. O numeral na base 2 é formado pelo último quociente e pelos restos tomados na ordem inversa do processo de divisão. Continuando, teríamos as igualdades: $9 = 4 \cdot 2 + 1$, $4 = 2 \cdot 2 + 0$, $2 = 1 \cdot 2 + 0$. Logo $37 = (100101)_2$.

2.5 Algoritmo Euclidiano Estendido

vi) "Se a e b são inteiros e $a = b \cdot q + r$, onde q e r são inteiros, então $mdc(a, b) = mdc(b, r)$"

Onde $mdc(a, b)$ é o maior divisor comum de a e b.

Isto é fácil ver porque, de acordo com o resultado **ii)**, como $a = b \cdot q + r$, os divisores de b e r dividem a, e os divisores de a e b também dividem r em razão de $a - b \cdot q = r$. Logo, o maior divisor de a e b também é o maior divisor de b e r.

Este fato nos dá uma maneira bastante eficiente para calcular o mdc entre dois números inteiros, realizando divisões sucessivas, o que é conhecido como Algoritmo de Euclides. Primeiro vamos aplicar o Algoritmo da Divisão com Resto de Euclides, onde o dividendo é 876 e o divisor é 597. Em seguida vamos dividir 597 pelo resto da divisão anterior, e assim sucessivamente até obtermos um resto zero. Isto será sempre possível, pois os restos obtidos neste processo formam uma sequências decrescente de números inteiros positivos.

Fazendo assim, obteremos a sequencia de igualdades:

$876 = 1 \cdot (597) + 279 \rightarrow$ Pelo teorema acima: $mdc(876, 597) = mdc(597, 279)$
$597 = 2 \cdot (279) + 39 \rightarrow$ Idem: $mdc(597, 279) = mdc(279, 39)$
$279 = 7 \cdot (39) + 6 \rightarrow$ Idem: $mdc(279, 39) = mdc(39, 6)$
$39 = 6 \cdot (6) + 3 \rightarrow$ Idem: $mdc(39, 6) = mdc(6, 3)$
$6 = 2 \cdot (3) + 0 \rightarrow$ Idem: $mdc(6, 3) = mdc(3, 0) = 3$.

Logo, temos:
$mdc(876, 597) = mdc(597, 279) = mdc(279, 39) = mdc(39, 6) = mdc(6, 3) = mdc(3, 0) = 3$.

Ou seja, o mdc é o último resto não-nulo obtido na sequência de divisões euclideanas acima. No entanto, é mais comum expressarmos essas divisões no diagrama abaixo:

	1	2	7	6	2
876	597	279	39	6	3
279	39	6	3	0	

Onde na linha superior temos os quocientes. Na do meio, os dividendos, e na de baixo, os restos. Apesar de ter sido inventado a mais de 2.000 anos, o Algoritmo de Euclides ainda é o mais eficiente quando se trata de calcular o mdc entre dois números. Segundo Silva em [10], p.65, uma avaliação importante da sua eficiência computacional foi feita pelo matemático Gabriel Lamé (1795 – 1870). O Teorema de Lamé garante que a quantidade de divisões necessárias para se determinar o mdc entre dois números positivos é, no máximo, cinco vezes a quantidade de algarismos do menor deles. Por exemplo, para se calcular o mdc entre 8579324 e 521 deveremos fazer, no máximo, $5 \cdot 3 = 15$ divisões, uma vez que o número 521 tem três algarismos.

Outro resultado importante para o RSA é o Teorema de Bachet-Bézout:

vii) "Se $mdc(a, b) = d$, então existem números inteiros x e y tais que $a \cdot x + b \cdot y = d$ "

Para encontrarmos os números x e y, aplicamos o Algoritmo de Euclides Estendido, que consiste em reescrever as igualdades das divisões acima, isolando-se os restos obtidos e substituindo-se, a partir da penúltima igualdade, as relações resultantes anteriores nas seguintes até se obter a relação $a \cdot x + b \cdot y = d$. Tomando-se como base o exemplo anterior, onde $mdc(876, 597) = 3 = 876 \cdot x + 597 \cdot y$, teremos:

$$876 = 1 \cdot (597) + 279 \Longrightarrow 876 - 597 = 279 \text{ (I)}$$
$$597 = 2 \cdot (279) + 39 \Longrightarrow 597 - 2 \cdot (279) = 39 \text{ (II)}$$
$$279 = 7 \cdot (39) + 6 \Longrightarrow 279 - 7 \cdot (39) = 6 \text{ (III)}$$
$$39 = 6 \cdot (6) + 3 \Longrightarrow 39 - 6 \cdot (6) = 3 \text{ (IV)}$$
$$6 = 2 \cdot (3) + 0$$

Substituindo (III) em (IV): $39 - 6 \cdot [279 - 7 \cdot (39)] = 3 \Longrightarrow 39 \cdot (43) - 6 \cdot (279) = 3$ (V)
Substituindo (II) em (V): $[597 - 2 \cdot (279)] \cdot (43) - 6 \cdot (279) = 3 \Longrightarrow 43 \cdot (597) - 92 \cdot (279) = 3$ (VI)
Substituindo (I) em (VI): $43 \cdot (597) - 92 \cdot (876 - 597) = 3 \Longrightarrow (-92) \cdot (876) + 135 \cdot (597) = 3$.

Logo $x = -92$ e $y = 135$.

Também importante para o Algoritmo RSA é o seguinte resultado:

viii) "Se a e b são relativamente primos e dividem c, então o produto $a \cdot b$ divide c".

Pois como a divide c, podemos escrever $c = a \cdot k$, para um k inteiro. E como b também divide c mas é relativamente primo com a, temos que b tem que dividir k, ou seja, $k = b \cdot t$, para um t inteiro. Daí, $c = a \cdot (b \cdot t) = (a \cdot b) \cdot t$. Logo $a \cdot b$ divide c.

Dizer que a e b são relativamente primos (ou primos entre si) é o mesmo que afirmar $mdc(a, b) = 1$, ou seja eles não possuem divisores em comum além do número 1.

Exemplo: Como 3 divide 120 e 5 divide 120, e $mdc(3, 5) = 1$, então $3 \cdot 5 = 15$ divide 120.

2.6 Noções de Aritmética Modular

2.6.1 Congruência

A aritmética modular é popularmente conhecida como a "aritmética do relógio", onde o número total de horas é o "módulo", um número natural positivo que chamaremos de n. Por exemplo, consideremos um relógio de 12 horas. Se agora são 8 horas da manhã e queremos saber que horas serão daqui a 10 horas, fazemos $8 + 10 = 18$ e a soma dividimos por 12, que nos dá um resto 6. Isto é, serão 6 horas da tarde. Portanto a adição na aritmética modular consiste em somar, depois dividir pelo módulo e obter o resto.

Em sua obra monumental sobre Teoria dos Números "Disquisitiones Arithmeticae" (Investigações aritméticas) publicada em 1801, o grande Gauss (1777 - 1875) introduziu a notação de congruência, conforme a definição abaixo:

Sejam a, b e n númeoros inteiros, com $n > 0$. Dizemos que a é congruente a b módulo n se ocorrer que $a - b = k \cdot n$, para k inteiro, ou seja, se n dividir a diferença $a - b$. Neste caso, escrevemos:

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

É fácil ver que isto ocorrerá se a e b deixarem o mesmo resto quando divididos por n, ou seja: $a = s \cdot n + r$ e $b = t \cdot n + r$, com s e t inteiros, o que nós dará $a - s \cdot n = r$ e $b - t \cdot n = r$. Logo $a - s \cdot n = b - t \cdot n$. Daí, teremos $a - b = (s - t) \cdot n$, ou ainda $a - b = k \cdot n$, onde $s - t = k$.

No exemplo acima, temos que $18 \equiv 6 \ (mod \ 12)$, pois 12 divide $18 - 6 = 12$.

Da mesma forma $39 \equiv 3 \ (mod \ 12)$, pois $39 - 3 = 36 = 3 \cdot 12$, e $72 \equiv 0 \ (mod \ 12)$, pois 12 divide $72 - 0 = 72 = 6 \cdot 12$.

Se a não é congruente a b módulo n, escrevemos $a \not\equiv b \ (mod \ n)$.

É fácil ver que todo múltiplo de 12 é congruente a 0 módulo 12.

Se o nosso relógio for de "5 horas", ou seja, o módulo $n = 5$ e quisermos saber a que horas equivale a potência 3^4, é só calcularmos essa potência e depois dividirmos o resultado por 5. O resto obtido nesta conta será a hora procurada. Ou seja, $3^4 = 81 = 16 \cdot 5 + 1$, o que nos dá 1 hora. Na forma de congruência, fica $3^4 \equiv 1 \ (mod \ 5)$.

ix) Valem as Propriedades:

Sejam a, b, c e d inteiros.

1) $a \equiv a \ (mod \ n)$ (reflexiva);
2) Se $a \equiv b \ (mod \ n)$ então $b \equiv a \ (mod \ n)$ (simétrica);
3) Se $a \equiv b \ (mod \ n)$ e $b \equiv c \ (mod \ n)$, então $a \equiv c \ (mod \ n)$ (transitiva);
4) Se $a \equiv b \ (mod \ n)$ e $c \equiv d \ (mod \ n)$, então $a + c \equiv b + d \ (mod \ n)$; $a - c \equiv b - d \ (mod \ n)$ e $a \cdot c \equiv b \cdot d \ (mod \ n)$;
5) Se $a \equiv b \ (mod \ n)$, então $a^k \equiv b^k \ (mod \ n)$, para k inteiro e $k \geq 0$;
6) Se $a \equiv b \ (mod \ n)$, então $a \pm c \equiv b \pm c \ (mod \ n)$;
7) Se $a \equiv b \ (mod \ n)$, então $a \cdot c \equiv b \cdot c \ (mod \ n)$;
8) Se $a \cdot x \equiv b \cdot x \ (mod \ n)$ e $mdc(x, n) = 1$ com x inteiro, então $a \equiv b \ (mod \ n)$. Ou seja, podemos "cortar" o x dos dois lados da congruência.

2.6.2 Sistema Completo de Restos

Considerando um relógio de 12 horas, qualquer número de horas, por maior que seja deverá ser congruente a uma das horas 0, 1, 2, 3, ..., 10, 11. Ou seja, cada uma dessas horas vai representar um conjunto de horas que deixam o mesmo resto quando dividida por 12. Obviamente haverá apenas 12 desses conjuntos. Por exemplo, $247 = 20 \cdot 12 + 7$. Ou ainda $247 \equiv 7 \ (mod\ 12)$. Vamos ver isto com mais generalidade.

Sejam a e n inteiros, onde $n > 0$. Pelo Algoritmo de Euclides para a divisão, temos que $a = q \cdot n + r$, onde q e r são inteiros únicos e $0 \leq r < n$. Ou seja, todo número inteiro quando dividido por n deixa um único resto $0 \leq r < n$. Este resto também é chamado de resíduo de a módulo n.

O Algoritmo de Euclides da divisão nos garante que, quando dividimos um número inteiro a qualquer por n, os únicos resíduos (restos) possíveis são $0, 1, 2, \ldots, n - 1$. Dizemos que $r_1, r_2, r_3, \ldots, r_n$ formam um sistema completo de restos (resíduos) módulo n, se qualquer r_i acima é congruente a um dos n números $0, 1, 2, \ldots, n - 1$, e são incongruentes dois a dois.

Sejam $r_1, r_2, r_3, \ldots, r_n$ um sistema completo de restos módulo n e a um inteiro relativamente primo com n. Temos que $a \cdot r_1, a \cdot r_2, a \cdot r_3, \ldots, a \cdot r_n$ também forma uma sistema completo de restos módulo n, pois supondo que $a \cdot r_i \equiv a \cdot r_j \ (mod\ n)$, e como $mdc(a, n) = 1$, temos que $r_i \equiv r_j \ (mod\ n)$, pela propriedade **ix), item 8)**. Ou seja, $i = j$, pois $0 \leq r_1, r_2, r_3, \ldots, r_n < n$ é um sistema completo de restos módulo n. Portanto, podemos escrever:

$$a \cdot r_1 \equiv r_i \ (mod\ n); a \cdot r_2 \equiv r_j \ (mod\ n); \ldots \equiv \ldots; a \cdot r_n \equiv r_k \ (mod\ n).$$

onde $r_i, r_j, \ldots, r_k$ são distintos dois a dois e estão em correspondência biunívoca com os números $0, 1, 2, \ldots, n - 1$.

Por exemplo, temos que 0, 1, 2, 3, 4, formam um sistema completo de resíduos módulo 5. Se multiplicarmos todos eles por 2, obteremos 0, 2, 4, 6, 8. Calculando módulo 5, temos 0, 2, 4, 1, 3. E tem que ser assim, pois, conforme ficou comprovado acima, ao multiplicarmos todos os elementos do sistema completo de restos por um número relativamente primo com o módulo, os produtos são incongruentes, e portanto iguais a um dos elementos 0, 1, 2, 3, 4, módulo cinco. Podemos ver isto como uma permutação dos elementos do sistema completo de restos.

2.6.3 Inversão módulo n

x) Se $mdc(a, n) = 1$, então a é inversível módulo n, ou seja, existe o inverso $b = a^{-1}$ tal que $a \cdot a^{-1} \equiv 1 \ (mod\ n)$, onde $0 < a < n$ e $0 < b < n$.

Seja $mdc(a, n) = 1$, temos pelo resultado **vii)** que existem x e y inteiros tal que

$a \cdot x + n \cdot y = 1$, o que nos permite escrever $a \cdot x = 1 - n \cdot y = 1 + (-y) \cdot n$, isto é $a \cdot x \equiv 1 \ (mod\ n)$. E x é o inverso de a módulo n, que indicamos $x = a^{-1}$. E mais, o inverso de a módulo n é único. Pois se existisse outro inverso b^{-1}, teríamos $a \cdot b^{-1} \equiv 1 \ (mod\ n)$, e portanto $a \cdot a^{-1} \equiv a \cdot b^{-1} \ (mod\ n)$. Como mdc(a,n)=1, podemos cancelar a nos dois lados da congruência, o que nos dá $a^{-1} \equiv b^{-1} \ (mod\ n)$. E como $b^{-1} < n$, temos que $a^{-1} = b^{-1}$.

Por exemplo, como $mdc(3, 10) = 1$, já sabemos que 3 é inversível módulo 10, isto é $3 \cdot x \equiv 1 \ (mod\ 10)$. E que esta congruência possui uma única solução módulo 10, onde $0 < x < 10$. E esta é precisamente 7, pois $3 \cdot (7) = 21 \equiv 1 \ (mod\ 10)$. Ou seja, 7 é o inverso de 3 módulo 10.

Uma consequência direta do que acabamos de ver é que se n for primo, todos os elementos não nulos do sistema completo de restos módulo n são inversíveis, isto é, 1,2,3, ..., $n - 1$ são inversíveis, pois todos são relativamente primos com n.

2.6.4 Exponenciação rápida

Vamos calcular 6^{73} módulo 100. Pelo que já vimos até aqui, deveríamos calcular a potência 6^{73} e depois dividirmos o resultado por 100, obtendo um determinado resto, que seria o resultado. Mas pela nossa experiência com potências, calcular 6^{73} é um trabalho estupendo. Então faremos esse cálculo de uma forma surpreendentemente "rápida". Escrevemos primeiro a expansão binária do expoente: $73 = 1 + 2^3 + 2^6$. Portanto, temos $6^{73} = 6^{1+2^3+2^6} = 6^1 \cdot 6^{2^3} \cdot 6^{2^6}$. Calculando os quadrados sucessivos de 6 e dividindo pelo módulo 100, temos $6^2 = 36$, $6^{2^2} = (6^2)^2 = (36)^2 \equiv 1296 \equiv 96 \ (mod\ 100)$, $6^{2^3} = (6^{2^2})^2 \equiv (96)^2 \equiv 9216 \equiv 16 \ (mo\ 100)$, $6^{2^4} = (6^{2^3})^2 \equiv (16)^2 \ (mod\ 100) \equiv 256 \equiv 56 \ (mod\ 100)$, $6^{2^5} = (6^{2^4})^2 \equiv (56)^2 \equiv 3116 \equiv 36 \ (mod\ 100)$, $6^{2^6} = (6^{2^5})^2 \equiv (36)^2 \equiv 1296 \equiv 96 \ (mod\ 100)$. Logo $6^{73} = 6^1 \cdot 6^{2^3} \cdot 6^{2^6} = 6 \cdot 16 \cdot 96 \equiv 9216 \equiv 16 \ (mod\ 100)$.

Desta forma calculamos apenas seis quadrados e dois produtos para obter o resultado. Se tivéssemos feito $6^{73} = 6 \cdot 6 \cdot 6 \cdot \ldots \cdot 6$ (73 fatores), teríamos que computar 72 multiplicações!

Podemos calcular qualquer potência pela exponenciação rápida usando a recorrência:

$\boldsymbol{a^{2^{i+1}} = (a^{2^i})^2}$, onde $\boldsymbol{i}$ é um número natural maior que zero.

2.7 Pequeno Teorema de Fermat

Estudando os "números perfeitos", aqueles números inteiros que são iguais à soma dos seus divisores próprios, Fermat descobriu e conjecturou este resultado de grande importância na teoria dos números, e que não era conhecido dos gregos. Euclides já havia provado n'Os Elementos, que os números pares da forma $2^{n-1} \cdot (2^n - 1)$ são números perfeitos sempre que $2^n - 1$ for primo.

Quando Fermat calculou as diversas potências de 2 módulo 5, ele percebeu o seguinte padrão:

n	1	2	3	4	5	6	7	8	9	10	11	12	13
$\mathbf{2^n}$	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192
$\mathbf{2^n}$ **(mod 5)**	2	4	3	1	2	4	3	1	2	4	3	1	2

Podemos observar que $2^5 \equiv 2\ (mod\ 5)$ e $2^4 \equiv 1\ (mod\ 5)$.

Da mesma forma, quando ele calculou as potências de 2 módulo 7 obteve $2^7 \equiv 2\ (mod\ 7)$ e $2^6 \equiv 1\ (mod\ 7)$. E de forma geral, podemos verificar que $a^p \equiv a\ (mod\ p)$ com p primo e $a^{p\ -\ 1} \equiv 1\ (mod\ p)$ onde $mdc(a, p) = 1$.

Observemos o comportamento aleatório das potências de 2 módulo 5, bem diferente das potências de 2 normais, onde se pode deduzir facilmente o valor do expoente pelo valor da potência dada. Nas potências módulo n qualquer isto não é possível, pois as mesmas não obedecem um padrão de crescimento estável, principalmente se o módulo n for um número "grande". Bennet em [3], p.162, afirma que as potências modulares são inclusive utilizadas em "geradores congruenciais"na obtenção de números aleatórios para utilização no método Monte Carlo.

Vamos portanto justificar o Pequeno Teorema de Fermat.

Já sabemos que quando dividimos um número qualquer por 5, os únicos restos possíveis são 0, 1, 2, 3, e 4. Vamos multiplicar um número relativamente primo com 5, por exemplo 2, apenas pelos restos não nulos, isto é $2{\cdot}(1), 2{\cdot}(2), 2{\cdot}(3), 2{\cdot}(4)$, obtendo 2, 4, 6, 8. Tomando o módulo 5, obtemos a sequência 2, 4, 1 e 3. Ou seja, obtivemos os mesmos restos não nulos em uma ordem diferente, o que já era esperado, de acordo com a Subseção 2.6.2. Portanto temos que o produto 1.2.3.4 é congruente ao produto $2{\cdot}4{\cdot}1{\cdot}3$, o que nos autoriza a escrever $2{\cdot}4{\cdot}1{\cdot}3 \equiv 1{\cdot}2{\cdot}3{\cdot}4\ (mod\ 5)$, ou ainda $2{\cdot}(1){\cdot}2{\cdot}(2){\cdot}2{\cdot}(3){\cdot}2{\cdot}(4) \equiv 1{\cdot}2{\cdot}3{\cdot}4\ (mod\ 5)$. Daí temos $2^4 \cdot (1 \cdot 2 \cdot 3 \cdot 4) \equiv 1 \cdot 2 \cdot 3 \cdot 4\ (mod\ 5)$. E como os números 1, 2, 3, e 4 são relativamente primos com 5, podemos "cortá-los" dos dois lados da congruência, obtendo $2^4 \equiv 1\ (mod\ 5)$.

Consideremos agora um primo p e um inteiro a qualquer onde $mdc(a, p) = 1$. Multiplicando a pelos restos não nulos módulo p, temos:

$a \cdot (1) \cdot a \cdot (2) \cdot a \cdot (3) \cdot \ldots \cdot a \cdot (p\ -\ 1) \equiv 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (p - 1)(mod\ p)$. O que nos dá $a^{p\ -\ 1} \cdot [1 \cdot 2 \cdot 3 \cdot \ldots \cdot (p\ -\ 1)] \equiv 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (p - 1)(mod\ p)$. E como 1, 2, 3, ..., (p – 1) são relativamente primos com p, temos o Pequeno Teorema de Fermat na forma provada por Euler:

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

O que equivale a dizer que p divide $a^{p-1} - 1$.

Exemplos:

a) $3^4 \equiv 1 \ (mod\ 5)$

b) $25^6 \equiv 1 \ (mod\ 7)$

c) Sabemos que $6972593^{88} - 1$ é divisível por 89, mesmo sem realizarmos o astronômico cálculo, pois 6972593 e 89 são primos, e portanto primos entre si.

A forma deste teorema originalmente conjecturada por Fermat é:

"Se p é primo e a é um inteiro positivo, então $a^p \equiv a \ (mod\ p)$"

Mas isto é uma consequência direta da forma anterior, pois, se p dividir $a^p - 1$, então p divide o pruduto $a \cdot (a^p - 1)$, ou seja, p divide $a^p - a$ e daí $a^p \equiv a \ (mod\ p)$. E caso p não divida a, pelo Pequeno Teorema de Fermat/Euler, p divide $a^p - 1$. Daí p divide o produto $a \cdot (a^p - 1)$, isto é, p divide $a^p - a$, e portanto $a^p \equiv a \ (mod\ p)$.

Exemplos:

a) $6^7 \equiv 6 \ (mod\ 7)$

b) $15^{93} \equiv 15 \ (mod\ 93)$

Capítulo 3

O Algoritmo RSA

3.1 Generalização de Euler

Para justificarmos porque o RSA realmente funciona, vamos conhecer antes um resultado preliminar que é a generalização de Euler ao Pequeno Teorema de Fermat quando $n = p \cdot q$.

"Sejam p e q dois números primos distintos e seja X um número inteiro, tal que p e q não dividem X, então $X^{(p-1)\cdot(q-1)} \equiv 1 \ (mod\ p \cdot q)$."

De acordo com o resultado **viii)**, é suficiente mostrarmos que:

$$X^{(p-1)\cdot(q-1)} \equiv 1 \ (mod\ p) \ \ e \ \ X^{(p-1)\cdot(q-1)} \equiv 1 \ (mod\ q)$$

Pois se p divide $X^{(p-1)\cdot(q-1)} - 1$ e q divide $X^{(p-1)\cdot(q-1)} - 1$, teremos que $p.q$ divide $X^{(p-1)\cdot(q-1)} - 1$, pois $mdc(p, q) = 1$.

Pelo Pequeno Teorema de Fermat/Euler, temos que $X^{p-1} \equiv 1 \ (mod\ p)$, pois $mdc(X, p) = 1$. Elevando esta congruência à potência $q - 1$, temos $X^{(p-1)\cdot(q-1)} \equiv (1)^{(q-1)} \ (mod\ p)$, ou seja, $X^{(p-1)\cdot(q-1)} \equiv 1 \ (mod\ p)$. Da mesma forma se prova que $X^{(p-1)\cdot(q-1)} \equiv 1 \ (mod\ q)$. E portanto, segue o resultado $X^{(p-1)\cdot(q-1)} \equiv 1 \ (mod\ p \cdot q)$.

Observação: De fato, a Generalização de Euler para o Pequeno Teorema de Fermat é o resultado que nos garante que se $mdc(X, n) = 1$, então $X^{\varphi(n)} \equiv 1 \ (mod\ n)$, onde $\varphi(n)$ é a função totiente de Euler. Para a implementação do RSA nos basta que $\varphi(n) = \varphi(p \cdot q) = (p - 1) \cdot (q - 1)$, onde p e q são primos. Maiores detalhes ver [11].

3.2 A magia do RSA – Como a mensagem codificada por Alice é recuperada por Bob

Veremos a seguir a "fórmula mágica" que Rivest, Shamir e Adleman criaram usando como "ingredientes" apenas Aritmética Modular, números primos, o Teorema de Euler e complexidade computacional, atendendo perfeitamente às exigências de um Sistema Criptográfico Assimétrico, segundo os postulados de Diffie e Hellman.

A base do RSA é o fato matemático descoberto por Euler visto anteriormente, originado do Pequeno Teorema de Fermat. O que Rivest, Shamir e Adleman fizeram foi combinar esses fatos matemáticos de forma coerente e eficiente de modo a criar um algoritmo de grande complexidade reversa, ou seja muito difícil de ser desmontado conhecendo-se a chave pública apenas. A prova de que o RSA realmente funciona é dada a seguir.

Vamos supor que Alice queira enviar uma mensagem M criptografada com o RSA para seu namorado Bob. Usando uma tabela de conversão conveniente, transformaremos a mensagem M em um número inteiro positivo X, fazendo cada letra da mensagem corresponder a um número inteiro, uma vez que na memória do computador, um texto ou qualquer informação nada mais é do que um números binários. Ou seja, no computador, a máxima de Pitágoras (572 - 479 a.C.) é totalmente verdadeira: "Tudo são números".

Segue enfim a "fórmula mágica" do Criptosistema RSA:

1º) Bob precisa gerar sua chave pública para que Alice ou qualquer pessoa a use para codificar sua mensagem X. Bob faz isso escolhendo um par de números inteiros (n, e), onde n é o produto de dois primos ímpares "grandes" distintos p e q escolhidos aleatoriamente, ou seja, $n = p.q$, e $1 < e < (p-1)\cdot(q-1)$ com $mdc[e, (p-1)\cdot(q-1)] = 1$. O número n é chamado de módulo RSA, e e é chamado de expoente de codificação.

2º) Já a chave privada (n, d) para que Bob decodifique a mensagem enviada por Alice, ele a gera a partir de n e e, onde $1 < d < (p-1)\cdot(q-1)$ e $d\cdot e \equiv 1\, mod[(p-1)\cdot(q-1)]$. E como $mdc[e, (p-1)\cdot(q-1)] = 1$, conforme vimos no resultado **x)**, o número d existe, sendo o inverso multiplicativo de e módulo $(p-1)\cdot(q-1)$. O número d é chamado de expoente de decodificação.

3º) Para codificar sua mensagem X, tal que $0 < X < n$, Alice toma a chave pública (n, e) de Bob, que está disponível para qualquer pessoa, calcula a potência X^e e determina o seu resto módulo n, obtendo assim a mensagem cifrada C, ou seja $C \equiv X^e\ (mod\ n)$, em seguida envia C a Bob, através de um canal de comunicação.

4º) Quando Bob receber a mensagem cifrada C de Alice, e usando sua chave secreta (n, d), ele calcula a potência C^d e determina o seu resto módulo n, isto é $X \equiv C^d\ (mod\ n)$, recuperando assim a mensagem X original que Alice lhe enviou.

Este último passo é o mais importante, pois ele nos garante que a mensagem original X enviada por Alice será recuperada por Bob. Vamos justificá-lo então.

Elevando os dois membros da congruência $C \equiv X^e \ (mod\ n)$ à potência d, teremos: $C^d \equiv (X^e)^d \ (mod\ n)$, ou ainda $C^d \equiv X^{e \cdot d} \ (mod\ p \cdot q)$. E como $e \cdot d \equiv 1 \ [mod\ (p\ -\ 1) \cdot (q\ -\ 1)]$, teremos que $e \cdot d = k \cdot (p\ -\ 1) \cdot (q\ -\ 1) + 1$, para um k inteiro. Logo, $C^d \equiv X^{[k \cdot (p\ -\ 1) \cdot (q\ -\ 1) + 1]} \ (mod\ n) \equiv [(X^{(p\ -\ 1) \cdot (q\ -\ 1)}]^k \cdot X \ (mod\ p \cdot q)$. E pelo resultado da Seção 3.1, $X^{(p\ -\ 1) \cdot (q\ -\ 1)} \equiv 1 \ (mod\ p \cdot q)$, temos que $C^d \equiv (1)^k \cdot X \ (mod\ p \cdot q)$. Logo, $C^d \equiv X \ (mod\ p \cdot q)$ ou $X \equiv C^d \ (mod\ n)$, onde $mdc(X, n) = 1$.

Só nos resta mostrar agora se quando $0 < X < n$ e $mdc(X, n) \neq 1$, ou seja, quando p ou q dividir X, se $X \equiv C^d \ (mod\ n)$. Caso p divida X, também dividirá $X \cdot (X^{k \cdot (p-1) \cdot (q-1)} - 1)$, para um k inteiro. Ou seja, p dividirá $X^{k \cdot (p-1) \cdot (q-1) + 1} - X$. De modo análogo, se q dividir X, também dividirá $X^{k \cdot (p-1) \cdot (q-1) + 1} - X$. E como $mdc(p, q) = 1$, temos, pela propriedade **viii)**, que $p \cdot q$ divide $X^{k \cdot (p-1) \cdot (q-1) + 1} - X$, o que equivale a $X^{k \cdot (p-1) \cdot (q-1) + 1} \equiv X \ (mod\ p \cdot q)$, ou seja, $C^d \equiv X \ (mod\ n)$.

E como $C^d \ (mod\ n)$ é um inteiro positivo menor que n, temos que realmente o X encontrado nesse proceso é mesmo o número inteiro X anterior à codificação, uma vez que só haverá igualdade neste caso se C^d for congruente a X módulo n, pois os inteiros $0, 1, 2, \cdots, n - 1$, formam um sistema completo de restos.

Portanto, a condição $0 < X < n$ nos garante que a mensagem X será recuperada quando aplicarmos a função de decodificação $X_i \equiv C^d \ (mod\ n)$.

3.3 Implementação matemática do Algoritmo RSA

A abordagem a seguir é uma expansão e adaptação da excelente implementação do RSA feita por Coutinho [16], Capítulo 6, com as modificações necessárias à sua integração a este trabalho e à sua clareza didática.

3.3.1 Pré-codificação do RSA

1º) Alice pode usar a tabela abaixo para converter letras em números, de modo que ela poderá codificar qualquer texto utilizando as ferramentas matemáticas vistas anteriormente.

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

É interessante que Alice utilize números de dois dígitos para cada letra a fim de evitar ambiguidades, tipo: não saberíamos se 15 seria AE ou O, caso Alice começasse

enumerando o alfabeto com 1, 2, 3, ... até 26.

Ela deverá substituir o espaço entre duas palavras por 99. Por exemplo, vamos supor que Alice vai enviar a mensagem **"AMO O PROFMAT"** para Bob. De acordo com a tabela acima a mensagem ficaria assim pré-codificada:

$$X = 1022249924999252724152210 29$$

2º) Bob deve definir os "parâmetros" do RSA, que são os números primos p e q que formam o módulo RSA, $n = p \cdot q$. Se ele escolher $p = 11$ e $q = 23$, teremos $n = p \cdot q = 11 \cdot 23 = 253$.

3º) Alice deverá "quebrar" o texto X em blocos. O tamanho de cada bloco apenas precisa ser menor que n. Então Alice pode quebrar a mensagem numérica acima como:

$$102–22–49–92–49–92–52–72–41–52–210 - 29$$

Que pode ser organizada na seguinte tabela de blocos:

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}
102	22	49	92	49	92	52	72	41	52	210	29

Alice também precisa ter o cuidado para que nenhum bloco começe com zero, pois na hora de decodificar poderia haver confusão. Por exemplo, não dá pra distinguir 029 de 29 quando se está operando com esses números.

Observemos também que ao quebrar a mensagem X em blocos aleatórios, Alice evita que a frequência de cada letra se mantenha ao longo do processo de cifragem.

4º) Bob deve escolher o expoente de codificação e, onde $1 \leq e \leq (p - 1) \cdot (q - 1)$ e $mdc[e, (p - 1) \cdot (q - 1)] = 1$. Como $(p - 1) \cdot (q - 1) = 10 \cdot 22 = 220$, temos que $mdc(e, 220) = 1$. Ele pode escolher $e = 3$, que é o menor possível, o que torna o processo de codificação mais "rápido", pois mdc(3,220)= 1, aplicando o Algoritmo de Euclides.

A chave pública de Bob é portanto o par $(n, e) = (253, 3)$ e que fica à disposição de qualquer pessoa que queira enviar uma mensagem para Bob.

Feito isto, podemos passar à fase de Codificação.

3.3.2 Codificação do RSA

Seja um "texto-bloco" X_i, onde $i = 1, 2, 3, ..., 12$.

De acordo com a seção 3.1., Alice deve codificar o texto X_i como:

$C_i \equiv X_i^e \ (mod\ n)$, onde C_i é o texto cifrado.

Alice pode usar aqui a exponenciação rápida vista no subseção 2.6.4 para facilitar seus cálculos. Tomando o primeiro bloco X_1 da mensagem pré-codificada no item anterior, temos:

$C_1 \equiv X_1^e \ (mod\ n)$. Como $X_1 = 102$:

$C_1 \equiv 102^3 \equiv (102)^{2+1} \equiv 102^2 \cdot 102 \equiv (10404) \cdot 102 \equiv (31) \cdot 102 \equiv 3162 \equiv 126 \ (mod\ 253)$.

Ou seja, o bloco $X_1 = 102$ foi codificado como 126.

Fazendo o mesmo com os demais blocos da mensagem criptografada, Alice obterá:

$C_2 \equiv X_2^e \ (mod\ n)$

$C_2 \equiv 22^3 \equiv 22^{2+1} \equiv (22^2) \cdot 22 \equiv (484) \cdot 22 \equiv (231) \cdot 22 \equiv 5082 \equiv 22 \ (mod\ 253)$.

Ou seja, o bloco $X_2 = 22$ foi codificado como 22.

$C_3 \equiv X_3^e \ (mod\ n)$

$C_3 \equiv 49^3 \equiv 49^{2+1} \equiv (49^2) \cdot 49 \equiv (2401) \cdot 49 \equiv (124) \cdot 49 \equiv 6076 \equiv 4 \ (mod\ 253)$

Ou seja, os blocos $X_3 = X_5 = 49$ foram codificados como 49.

$C_4 \equiv X_4^e \ (mod\ n)$

$C_4 \equiv 92^3 \equiv 92^{2+1} \equiv (92^2) \cdot 92 \equiv (8464) \cdot 92 \equiv (115) \cdot 92 \equiv 10580 \equiv 207$.

Ou seja, os blocos $X_4 = X_6 = 92$ foram codificados como 207.

$C_7 \equiv X_7^e \ (mod\ n)$

$C_7 \equiv 52^3 \equiv 52^{2+1} \equiv (52^2) \cdot 52 \equiv (2704) \cdot 52 \equiv (174) \cdot 52 \equiv 9048 \equiv 193 \ (mod\ 253)$.

Ou seja, os blocos $X_7 = X_{10} = 52$ foram codificados como 193.

$C_8 \equiv X_8^e \ (mod\ n)$

$C_8 \equiv 72^3 \equiv 72^{2+1} \equiv (72^2) \cdot 72 \equiv (5184) \cdot 72 \equiv (124) \cdot 72 \equiv 8928 \equiv 73 \ (mod\ 253)$.

Ou seja, o bloco $X_8 = 72$ foi codificado como 73.

$C_9 \equiv X_9^e \ (mod\ n)$

$C_9 \equiv 41^3 \equiv 41^{2+1} \equiv (41^2) \cdot 41 \equiv (1681) \cdot 41 \equiv (163) \cdot 41 \equiv 6683 \equiv 105 \ (mod\ 253)$.

Ou seja, o bloco $X_9 = 41$ foi codificado como 105.

$C_{11} \equiv X_{11}^e \ (mod\ n)$

$C_{11} \equiv 210^3 \equiv 210^{2+1} \equiv (210^2) \cdot 210 \equiv (44100) \cdot 210 \equiv (78) \cdot 210 \equiv 16380 \equiv 188\ (mod\ 253)$.

Ou seja, o bloco $X_{11} = 210$ foi codificado como 188.

$C_{12} \equiv X_{12}^e\ (mod\ n)$
$C_{12} \equiv 29^3 \equiv 29^{2+1} \equiv (29^2) \cdot 29 \equiv (841) \cdot 29 \equiv (82) \cdot 29 \equiv 2378 \equiv 101\ (mod\ 253)$.

Ou seja, o bloco $X_{12} = 29$ foi codificado como 101.

Então, a mensagem ficará cifrada da seguinte forma:

C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}
126	22	4	207	4	207	193	73	105	193	188	101

Observamos aqui que Bob tem de manter os blocos cifrados separados! Caso contrário, ele não poderá mais recuperar a mensagem original enviada por Alice, pois não saberia a quais números aplicar corretamente a função de decodificação $X \equiv C^d\ (mod\ n)$.

3.4 Decodificação do RSA

Para decodificar a mensagem enviada por Alice, Bob precisará do número n e do expoente de decodificação d, onde $e \cdot d \equiv 1\ mod[(p-1) \cdot (q-1)]$ e $1 \leq d \leq (p-1) \cdot (q-1)$, de acordo com o item 3.1. Ou seja, a chave privada é o par (n, d), que deve ficar em posse apenas de Bob. Na verdade, apenas d deve ser sigiloso, pois n já foi divulgado publicamente.

Como $e = 3$, $p = 11$ e $q = 23$, teremos:

$e \cdot d \equiv 1\ [mod\ (p-1) \cdot (q-1)] \Longrightarrow 3 \cdot d \equiv 1\ [mod (11-1) \cdot (23-1)] \Longrightarrow 3 \cdot d \equiv 1\ (mod\ 220)$. Logo d existe pois mdc (3, 220)= 1.

Fazendo $3 \cdot d = 220 \cdot k + 1$, temos $3 \cdot d - 220 \cdot k = 1$. Aplicando agora o Algoritmo Euclideano Estendido, temos:

	73	3
220	3	1
1	0	

$220 = 3 \cdot (73) + 1 \Longrightarrow (-73) \cdot 3 + (1) \cdot 220 = 1 \Longrightarrow (-73) \cdot 3 \equiv 1\ (mod\ 220)$. E como $-73 \equiv 147\ (mod\ 220)$, temos que $(147) \cdot 3 \equiv 1\ (mod\ 220)$. Logo $d = 147$.

Observamos que, ao contrário do processo de codificação, onde é conveniente trabalhar com um expoente de codificação e pequeno, no nosso caso $e = 3$, o expoente de

decodificação $d = 147$ torna o processo de tentativa de ataque ao RSA difícil, pois o atacante terá que testar vários candidatos a expoentes de decodificação grandes até obter um texto-bloco X_i.

Consideremos o primeiro bloco codificado $C_1 = 126$. Escrevendo $d = 147$ na base 2, ou seja, $147 = 2^7 + 2^4 + 2 + 1$ e aplicando a função de decodificação $X \equiv C^d \ (mod\ n)$, teremos:

$X_1 \equiv C_1^d \ (mod\ n)$

$X_1 \equiv 126^{147} \equiv 126^{2^7+2^4+2+1} \equiv (126^{2^7}) \cdot (126^{2^4}) \cdot (126^2) \cdot (126) \ (mod\ 253)$.

Antes de prosseguirmos vamos calcular as potências obtidas acima por exponenciação rápida, separadamente.

$126^2 \equiv 15876 \equiv 190 \ (mod\ 253)$

$126^{2^2} \equiv (126^2)^2 \equiv 190^2 \equiv 36100 \equiv 174 \ (mod\ 253)$.

$126^{2^3} \equiv (126^{2^2})^2 \equiv 174^2 \equiv 30276 \equiv 169 \ (mod\ 253)$.

$126^{2^4} \equiv (126^{2^3})^2 \equiv 169^2 \equiv 28561 \equiv 225 \ (mod\ 253)$.

$126^{2^5} \equiv (126^{2^4})^2 \equiv 225^2 \equiv 50625 \equiv 25 \ (mod\ 253)$.

$126^{2^6} \equiv (126^{2^5})^2 \equiv 125^2 \equiv 625 \equiv 119 \ (mod\ 253)$.

$126^{2^7} \equiv (126^{2^6})^2 \equiv 119^2 \equiv 14161 \equiv 246 \ (mod\ 253)$.

Portanto, temos:

$X_1 \equiv 126^{147} \equiv (126^{2^7}) \cdot (126^{2^4}) \cdot (126^2) \cdot (126) \equiv (246) \cdot (225) \cdot (190) \cdot (126) \equiv (55350) \cdot (23940) \equiv (196) \cdot (158) \equiv 30968 \equiv 102 \ (mod\ 253)$. (Viva Rivest, Shamir e Adleman !!!)

Se Bob aplicar $X_i \equiv C_i^d \ (mod\ n)$ aos demais blocos da mensagem cifrada, obterá os blocos numéricos respectivos da mensagem original X enviada por Alice. Depois é só Bob usar a tabela do item 1º), subseção 3.3.1, no sentido inverso e trocar os números pelas letras, obtendo assim a mensagem em forma de texto legível enviada por Alice.

Capítulo 4

Implementação RSA com MAXIMA

Trabalharemos a seguir dois exemplos mais realistas de implementação do RSA utilizado como ferramenta de apoio didático o programa de computação algébrica wxMAXIMA, versão 12.04.0. Maiores detalhes e comandos ver Apêndice A.

Exemplo 1: Achar primos de 10 dígitos para usar no módulo RSA, codificar e decodificar a palavra ARACAJU usando a tabela ASCII contida no Apêndice B.

Seja o módulo RSA $n = p \cdot q$. Façamos inicialmente $p = 2351174217$.

No MAXIMA:

primep(2351174217) retorna "false". Então fazemos em seguida *next_prime*(2351174217), que reorna 2351174351.

Em seguida, no MAXIMA, fazemos:

primep(2351174351), que retorna "true". Logo 2351174351 é primo com uma grande probabilidade.

Portanto, fazemos $p = 2351174351$.

Do mesmo modo, fazemos inicialmente $q = 1201429113$.

No MAXIMA:

primep(1201429113), que retorna "false".
Então fazemos em seguida *next_prime*(1201429113), que retorna 1201429123.

Depois fazemos *primep*(1201429123), que retorna "true". Logo 1201429123 é primo com grande probabilidade.

E então fazemos $q = 1201429123$.

Portanto, o módulo RSA $n = p \cdot q = 2351174351 \cdot 1201429123 = 2824769338542024173$.

Uma observação importante! Para testar se o módulo RSA encontrado é"bom", ou seja, difícil de ser fatorado em um tempo"baixo", podemos utilizar o comando factor(n). Se esta ação retornar os fatores de n em um tempo "curto", significa que devemos descartar os parâmetros p e q e procurar outros até que a fatoração de n seja "difícil".

Vamos calcular agora o expoente de codificação e.

Fazendo $(p\ -\ 1) \cdot (q\ -\ 1) = 2351174350 \cdot 1201429122 = 2824769334989420700$.

Procuramos agora um número inteiro e que satisfaça $1 < e < (p\ -\ 1) \cdot (q\ -\ 1)$ e $mdc[e, (p\ -\ 1) \cdot (q\ -\ 1)] = 1$.

No MAXIMA:

$gcd(3, 2824769334989420700)$ reorna 3;
$gcd(5, 2824769334989420700)$ retorna 5;
$gcd(7, 2824769334989420700)$ retorna 7;
$gcd(11, 2824769334989420700)$ retorna 1.

Logo, podemos usar 11 como expoente de exponenciação. Daí, $e = 11$.

Portando a chave pública é $(n, e) = (2824769338542024173, 11)$, que iremos usar para codificar as mensagens a serem enviadas a um destinatário.

Convertendo a palavra ARACAJU em um número decimal de acordo com a tabela ASCII constante no Apêndice B, temos: X = 65826567657485.

Vamos agora codificar a mensagem X.

Para isso fazemos no MAXIMA:

$power\_mod\ (65826567657485, 11, 2824769338542024173)$, que retorna 1093258905866990787. Então a nossa mensagem codificada é C = 1093258905866990787.

Para decodificar a mensagem C recebida, o destinatário terá que calcular primeiro sua chave privada, que é o expoente de decodificação d, onde $e \cdot d = 1\ [mod\ (p\ -\ 1) \cdot (q\ -\ 1)]$ e $1 \leqslant d \leqslant (p\ -\ 1) \cdot (q\ -\ 1)$.

No MAXIMA, fazemos:

$inv\_mod(11, 2824769338542024173)$, que retorna 1283986061358827591.
Ou seja $d = 1283986061358827591$.

Agora, para decodificar a mensagem C, basta calcularmos $X \equiv C^d \ (mod\ n)$.

No MAXIMA, fazemos:

$power\_mod(1093258905866990787, 1283986061358827591, 2824769338542024173)$, que retorna 65826567657485. Aplicando inversamente a tabela ASCII obtemos a palavra ARACAJU.

Exemplo 2: Uma mensagem foi codificada com a chave pública e a tabela ASCII do exemplo anterior, e corresponde a C = 392551868987068804 - 2113855424116769378 - 948229051171291444 - 726877796189145130. Decodificar a mensagem.

Como a mensagem foi codificada em blocos, basta decodificarmos cada bloco separadamente utilizando a chave privada d calculada no exemplo anterior, aplicando

$$X \equiv C^d \ (mod\ n).$$

No MAXIMA, fazemos:

$power\_mod(392551868987068804, 1283986061358827591, 2824769338542024173)$, que retorna $793285787386698283 = X_1$

$power\_mod(2113855424116769378, 1283986061358827591, 2824769338542024173)$, que retorna $793265678269687384 = X_2$

$power\_mod(948229051171291444, 1283986061358827591, 2824769338542024173)$, que retorna $653278653267827380 = X_3$

$power\_mod(726877796189145130, 1283986061358827591, 2824769338542024173)$, que retorna $8479718265707365 = X_4$

Concatenando os blocos decodificados, obtemos a mensagem X procurada. Assim, temos:

X = 793285787386698283793265678269687384653278653267827380847971826570736 5.

Que após aplicarmos a tabela ASCII inversamente, obtemos:

X = **O UNIVERSO ACREDITA NA CRIPTOGRAFIA**. (Julian Assange)

Capítulo 5

Segurança e Criptoanálise do RSA

5.1 Algoritmo da fatoração

Segundo Terada [7], p.21, "Criptoanálise" é o conjunto de técnicas matemáticas que têm por objetivo atacar um código a fim de desvendar o significado do texto criptografado, através da descoberta do algoritmo criptográfico, da sua chave de decodificação ou de um ataque ao texto cifrado. O criptoanalista portanto nada mais é do que um "quebrador" de códigos, e a Ciência da Escrita Secreta tem sido a sítese do confronto entre criptógrafos e criptoanalistas. A estratégia mais importante usada pelos criptoanalistas para atacar o RSA é a tentativa de fatorar $n = p \cdot q$, pois o expoente de decodificação d pode ser deduzido se conhecermos os números primos p e q, uma vez que e já é de conhecimento público, onde $e \cdot d \equiv 1\ [mod\ (p-1) \cdot (q-1)]$ e $1 \leq d \leq (p-1) \cdot (q-1)$. Então, se alguém conseguisse fatorar $n = p \cdot q$ e consequentemente obtivesse d, qualquer mensagem enviada de Alice para Bob poderia ser facilmente decodificada, bastando aplicar $X_i \equiv C_i^d\ (mod\ n)$ ao bloco criptografado C_i, conforme visto anteriormente. Mas fatorar n é praticamente uma tarefa impossível de ser realizada em um tempo humanamente viável e de uma maneira eficiente, a menos que o módulo n RSA tenha fatores primos pequenos. Para nos convencermos disto, vejamos o seguinte resultado:

"Se n não é primo, então n possui, necessariamente, um divisor primo p, tal que $p \leq \sqrt{n}$"

Como estamos supondo n é composto, então o seu menor divisor é um primo p, se não fosse existiria outro primo $p_1 < p$, onde $p = k \cdot p_1$, e neste caso p_1 também dividiria n. Mas isto não pode ser pois estamos supondo que p é o menor divisor de n. Logo p é o menor divisor de n. Então podemos escrever $n = p \cdot c$, onde c é um número composto e $1 < p, c < n$. Temos que $p \leq c$. Multiplicando esta desigualdade por p, temos $p^2 \leq p \cdot c$, ou seja, $p^2 \leq n$, ou ainda $p \leq \sqrt{n}$.

Outra maneira equivalente de enunciar este resultado é:

"Se um número inteiro positivo n não possui nenhum divisor primo p, tal que $p \leq \sqrt{n}$, então n é primo".

Com base nos resultados acima, podemos esboçar um algoritmo simples que nos diz se um determinado número inteiro positivo n é ou não primo.

Algoritmo da fatoração

1. **Entrada**: Um número inteiro positivo n
2. **Saída**: n é composto ou n é primo
3. **Instruções**: Tente dividir n por 2. Se for divisível páre, e dois é um fator de n. Senão tente dividir n por 3. Se for divisível, páre, e três é um fator de n. Se não for, tente dividir n por 5... Faça isso até encontrar um divisor de n. Neste caso n será composto. Ou até que o candidato a divisor de n seja maior do que $\sqrt{n}$. Se acontece este último caso, n será primo!

Para saber se um número inteiro n é ou não divisível por outro número a, podemos usar as regrinhas de divisibilidade ou simplesmente aplicar o algoritmo da divisão euclidiana. Se o resto for zero, neste caso, n será divisível por a, e a será um fator de n.

Por exemplo, vamos verificar se o número 173 é ou não primo usando o algoritmo "Divisões Sucessivas". Teremos então que tentar dividir 173 por 2, 3, 5, ... até no máximo 13, pois $\sqrt{173} = 13,152\ldots$ Assim temos que 173 não é divisível por 2, pois é ímpar. Não é divisível por 3, pois a soma dos seus algarismos dá 11. Não é divisível por 5, pois o seu algarismo das unidades não é zero. Não é divisível por 7, pois deixa resto 5. Não é divisível por 11, pois deixa resto 8, e também não é divisível por 13, pois deixa resto 4. Logo podemos concluir que 173 é primo!

Os resultados acima fundamentam o nosso tão conhecido Algoritmo da Fatoração que nós aprendemos no sexto ano do ensino fundamental. Se o número dado n for composto, obteremos necessariamente um menor fator primo menor do que $\sqrt{n}$. Se continuarmos aplicando o algoritmo da fatoração aos demais fatores de n, teremos no final a decomposição prima do número dado. O Teorema Fundamental da Aritmética garante que "todo número inteiro positivo ou é primo ou pode ser decomposto em um produto único de fatores primos, a menos da ordem desses fatores."

Por exemplo, 120 é divisível por 2, então $120 = 2 \cdot 60$. Como 60 também é divisível por 2, temos $120 = 2 \cdot 2 \cdot 30$. Trinta também é divisível por 2, daí $120 = 2 \cdot 2 \cdot 2 \cdot 15$. Como 15 é divisível por 3, temos $120 = 2 \cdot 2 \cdot 2 \cdot 3 \cdot 5 = 3 \cdot 2 \cdot 5 \cdot 2 \cdot 2 = 5 \cdot 2 \cdot 3 \cdot 2 \cdot 2 = 2 \cdot 3 \cdot 3 \cdot 5$.

5.2 Eficiência do Algoritmo da fatoração

Vamos analisar agora se o algoritmo das divisões sucessivas é eficiente quando se trata de fatorar números inteiros grandes . Digamos que o módulo n que queremos fatorar tem 100 algarismos, onde seus dois fatores primos são da ordem de 10^{50}, ou seja, considerando-se sua maior parcela 10^{100} e dispensando-se as demais, teremos $\sqrt{10^{100}} = 10^{50}$. Ou seja, teríamos que realizar 10^{50} divisões. Se cada divisão gastasse um tempo de 10^{-10} segundos, que é um tempo muito menor do que o mais rápido computador que existe atualmente, levaríamos $10^{50} \cdot 10^{-10} = 10^{40}$ segundos. Como um ano tem, em média, $12 \cdot 30 \cdot 24 \cdot 60 \cdot 60 = 31.104.000$ segundos, nós gastaríamos $10^{40} \div 31.104.000 \cong 10^{32}$ anos $= 100.000.000.000.000.000.000.000.000.000.000$ anos! Que é cem sextilhões de vezes mais tempo do que a idade do universo, segundo os astrofísicos estimada em 13 bilhões de anos.

Isto nos mostra que o algoritmo da fatoração é extremamente ineficiente quando se trata de fatorar números inteiros grandes, pois a quantidade de operações de divisão cresce exponencialmente com relação à quantidade de dígitos do número que se quer fatorar. Ainda assim, o algoritmo das divisões sucessivas não é totalmente inútil, pois podemos esperar que o número inteiro dado, mesmo muito grande, tenha um fator primo "pequeno", da ordem de 10^{10} algarismos. Isto pode ocorrer por descuido do programador do algoritmo RSA sob ataque.

Segundo Paulo Ribenboim em [17], p.107, até hoje não se descobriu nenhum algoritmo eficiente para se fatorar um número inteiro grande, digamos com 200 algarismos. Não se sabe ainda se este algoritmo pode existir ou não. Mesmo com toda a Matemática e toda a tecnologia computacional avançada, é um problema em aberto fatorar com eficiência um número inteiro muito grande. Esta é uma questão muito séria, uma vez que é possível alguém conseguir a proeza em segredo, comprometendo totalmente a segurança do RSA e consequentemente de suas aplicações.

Apesar desta clássica dificuldade em se fatorar um número inteiro grande, algums sucessos têm ocorrido com a utilização do Crivo Quadrático e do Crivo de Corpos, usando-se computação paralela e distribuida entre centenas de computadores de usuários da internet. Aplicando esses métodos, já foi possível fatorar números de até 200 dígitos decimais, embora em meses de processamento. Ver [7], p.129.

Capítulo 6

Encontrando primos - Testes de primalidade

Nos exemplos anteriores foram utilizados números primos "pequenos" na composição do módulo $n = p \cdot q$ do Algoritmo RSA. Mas para garantir uma segurança satisfatória para o módulo RSA, o ideal é que usemos números primos da ordem de 100 dígitos ou mais cada um. Conforme visto no item anterior, isso dificultará uma tentativa de fatorar n e obter o expoente de decodificação d. Na prática, para obter números primos muito grandes utilizamos programas de computação algébrica, a exemplo do MAXIMA.

Mas como é possível o MAXIMA, ou outro programa de computação algébrica qualquer, encontrar números primos gigantes, da ordem de 100 dígitos para serem usados no algoritmo RSA, se já sabemos que é impossível determinar se um número desta ordem é ou não primo, mesmo com o mais rápido dos computadores? Será que os programas de computação algébrica implementam alguma fórmula especial que nos forneçe o enésimo número primo? Bom, essa fórmula realmente existe, conforme podemos verificar em [17], p.118, mas é uma fórmula de grande complexidade computacional, totalmente inútil para fins práticos. Na verdade, os programas de computação algébrica implementam critérios de primalidade que consistem em se testar alguma propriedade dos números primos que os caracterize com uma grande probabilidade, ou de forma determinística com base em alguma propriedade específica que determinados números primos apresentam, a exemplo dos números de Fermat e Mersenne.

Os testes de primalidade mais eficientes que são usados em programas de computação algébrica são probabilísticos, e se baseiam no famoso Pequeno Teorema de Fermat/Euler:

> **"Seja p um número primo. Se a é um número inteiro tal que $mdc(a, p) = 1$, então $a^{p-1} \equiv 1 \ (mod \ p)$"**

Grandes matemáticos, a exemplo de Leibniz (1646 - 1716), acreditavam que se um número natural n qualquer satisfizesse a congruência $a^{n-1} \equiv 1 \ (mod \, n)$, com $mdc(a, n) =$

1, então n seria primo. Ou seja, pensava-se que a recíproca do Pequeno Teorema de Fermat também fosse verdadeira, o que não é verdade. Por exemplo $7^{24} \equiv 1 \ (mod\ 25)$ e 25 é composto. Os números compostos ímpares n que satisfazem o Pequeno Teorema de Fermat são chamados de "pseudoprimos" na base a. E há ainda os números de Carmichael, que são pseudoprimos para todas as bases a relativamente primas com n, o que faz os números de Carmichael se "camuflarem" totalmente como se fossem números primos para esta propriedade. O menor número de Carmichael é $561 = 3 \cdot 11 \cdot 17$, onde $a^{560} \equiv 1 \ (mod\ 561)$, para todo a relativamente primo com 561. Segundo S.Shokranian em [6], p.66, foi provado por Alford, Granville e Pomerance em 1994 que existem infinitos números de Carmichael.

Conforme Coutinho em [14], p.106, os pseudoprimos e os números de Carmichael são bastante raros quando se testa números até a ordem de um bilhão. Por exemplo, há 50.847.534 primos de 1 até um bilhão, mas apenas 5587 pseudoprimos para a base 2. Ou seja, aproximadamente 0,01% dentre primos destes são pseudoprimos. Se forem usadas duas bases, por exemplo 2 e 3, conclui-se que há somente 1272 pseudoprimos para essas bases, ou seja 0,0025%. Aumentando-se o número de bases a serem testadas, pode-se chegar a uma precisão ainda maior neste tipo de teste de primalidade. Claro que esse cuidado vai depender da importância da aplicação RSA que está sendo implementada. Quanto maior a segurança das informações que se quer proteger, maior deve ser o cuidado com a precisão de um teste de primalidade probabilístico.

Um teste de primalidade probabilístico utilizado na maioria dos programas de computação algébrica e que se baseia no Pequeno Teorema de Fermat é o Teste de Miller-Rabin. Rabin provou que há uma probabilidade menor que $\frac{1}{4}$ de este teste acusar um pseudoprimo quando se testa uma base a aleatória. Ver Buchmann [13], p.153. Então, para k bases distintas testadas, há uma probabilidade menor que $(\frac{1}{4})^k$ de n ser composto. Por exemplo, testando-se 10 bases, há uma probabilidade menor que $(\frac{1}{4})^{10} = \frac{1}{2^{20}} \simeq 0,000000954$ de o número testado não ser primo. No entanto, o teste de de Miller-Rabin fundamenta-se na Hipótese de Riemman Generalizada, que asegura ser suficiene testar um número "pequeno" de bases para se garantir a primalidade de um número. Ver Ribenboim [17], p.101.

Há também os testes determinísticos que finalizam afirmando com toda certeza se um número natural é ou não primo, embora eles sejam bem mais lentos na sua implementação computacional. Um exemplo é o teste de Lucas-Lehmer que testa a primalidade de um número de Mersenne. Ver [14], Cap.09.

Interessante observar que a contrapositiva do Pequeno Teorema de Fermat constitui-se em um teste de composição bastante eficaz e determinístico. Isto é:

"Se $a^{n-1} \not\equiv 1 \ (mod\ n)$, onde $mdc(a,p) = 1$, então n não é primo".

Ou ainda:

"Se $a^{n-1} \not\equiv 1 \ (mod\ n)$, onde $mdc(a,p) = 1$, então n é composto"

Isto é, se a^{n-1} não é congruente a 1 módulo n, então n é composto. Desta forma, podemos saber se um determinado número inteiro muito grande é composto sem no entanto conhecermos seus fatores. Por exemplo, $3^{340} \equiv 56\ (mod\ 341)$. Logo podemos ter certeza de que 341 é composto.

Com a onipresença da internet em nossas vidas e a grande utilização do Criptosistema RSA em seus protocolos de comunicação, podemos nos perguntar se há números primos grandes o suficiente para que sejam gerados módulos RSA seguros para atender a todos os habitantes da terra, de modo que possamos estar certos de que duas pessoas quisquer não escolherão o mesmo par de números primos, ou de que esta possibilidade é remota. E uma vez que os números primos vão ficando mais raros e mais afastados à medida que avançamos na sequência dos números naturais, podemos nos perguntar também se há números primos suficientemente grandes, digamos com 100 algarismos, para que possamos obter um múdulo RSA de 200 dígitos, o que é considerado atualmente um nível de segurança bastante satisfatório para a maioria das aplicações do RSA.

O Teorema dos Números Primos de Gauss-Hadamard-Poussin nos garante que a quantidade de primos de 1 até um dado natural n, denotada por $\pi(n)$, é próxima e sempre menor do que o quociente de n por $ln\ n$ quando n tende ao infinito, ou seja:

$$\pi(n) \cong \frac{n}{ln\ n}, \text{ para } n \to \infty.$$

Dessa forma temos que a quatidade de primos de 100 algarismos é igual à quantidade de primos até 10^{100} menos a quantidade de primos até 10^{99}. Aplicando o Teorema dos Números Primos, temos:

$$\frac{10^{100}}{ln\ 10^{100}} \ - \ \frac{10^{99}}{ln\ 10^{99}} \cong 3,9 \cdot 10^{97}$$

Supondo que o Planeta Terra tivesse 10 bilhões de habitantes, o que é claramente uma estimativa exagerada, a quantidade de números primos de 100 algarismos para cada habitante, seria:

$$\frac{3,9 \cdot 10^{97}}{10^{10}} = 3,9 \cdot 10^{87}$$

Que é aproximadamente $2 \cdot 10^{87}$ pares distintos de primos para cada um dos 10 bilhões de habitantes da suposta população terráquea!

Apêndice A

Comandos do MAXIMA para o RSA

O MAXIMA é um poderoso programa de computação algébrica que, dentre muitas coisas, possui valiosas ferramentas para se explorar os principais elementos da Teoria dos Números ligados às aplicações ao Criptosistema RSA. Sua versão gráfica wxMAXIMA 12.04.0 pode ser obtida gratuitamente no saite:

`http://wxmaxima.sourceforge.net/wiki/index.php/Main_Page`

Seguem abaixo os comandos básicos usados neste trabalho.

- `primep(n)`: Se retornar "false", n é composto. Se retornar "true", n é primo com grande probabilidade.(Miller-Rabin)
- `factor(n)`: Retorna a fatoração de n.
- `next_prime(n)`: Retorna o menor primo maior que n.
- `prev_prime(n)`: Retorna o maior primo menor que n.
- `inv_mod(n,m)`: Retorna o inverso de n módulo m. Retorna "false" se n módulo m for zero.
- `remainder(D,q)`: Retorna o resto da divisão de D por q.
- `power_mod(a,n,m)`: Retorna o resto da divisão de a^n por m.
- `gcd(n,m)`: Retorna o mdc de n e m.
- `float(n)`: Retorna o valor decimal aproximado de n.

Apêndice B

Tabela ASCII imprimíveis maiúsculos

Letra	Rep. decimal	Rep. binária	Letra	Rep. decimal	Rep. binária
Espaço	32	0101 1010	N	78	0100 1110
A	65	0100 0001	O	79	0100 1111
B	66	0100 0010	P	80	0101 0000
C	67	0100 0011	Q	81	0101 0001
D	68	0100 0100	R	82	0101 0010
E	69	0100 0101	S	83	0101 0011
F	70	0100 0110	T	84	0101 0100
G	71	0100 0111	U	85	0101 0101
H	72	0100 1000	V	86	0101 0110
I	73	0100 1001	W	87	0101 0111
J	74	0100 1010	X	88	0101 1000
K	75	0100 1011	Y	89	0101 1001
L	76	0100 1100	Z	90	0101 1010
M	77	0100 1101			

Referências Bibliográficas

[1] CRATO, Nuno. *A Matemática das Coisas*, Editora Gradiva, Lisboa-Portugal, 10.ed, 2010.

[2] RANGEL, Ricardo. *Passado e futuro da Era da Informação*, Editora Nova Fronteira, Rio de Janeiro-RJ,1.ed.,1999.

[3] BENNET, Deborah J., *Aleatoriedade*, Editora Martins Fontes, São Paulo,1.ed.,2003.

[4] SINGH, Simon. *O Livro dos Códigos*, Editora Record, Rio de Janeiro-RJ,3.ed,2003.

[5] KHAN, David. *The Codebreakers - The story of secret writing*, Editora New American Library-NY-USA, 1.ed.,1973.

[6] SHOKRANIAN, Salahodin. *Criptografia para iniciantes*, Editora Ciência Moderna,Brasilia-DF, 2.ed., 2012.

[7] TERADA, Routo. *Segurança de Dados: Criptografia em redes de computadores*, Editora Edgard Blucher,São Paulo-SP, 2.ed., 2008.

[8] KNUTH, Donald E., *The art of computer programming*, Editora Addison-Wesly Publishing Company, INC,Menlo Park-Califonia-USA, Volume1, 2.ed., 1973.

[9] HEFEZ, Abramo. *Elementos de aritmética - Coleção Textos Universitáros*, Editora da SBM, Rio de Janeiro-RJ, 2.ed.,2006.

[10] SILVA, Valdir Vilmar da. *Números - Construção e propriedades*, Editora da UFG, Goiânia-GO, 1.ed.,2003.

[11] SANTOS, José Plínio de Oliveira. *Introdução à Teoria dos Números*, Editora do IMPA, Rio de Janeiro-RJ,3.ed.,2010.

[12] SAUTOY, Marcus du. *A música dos números primos*, Editora Jorge Zahar, Rio de Janeiro-RJ,1.ed.,2007.

[13] BUCHMANN, Johannes. *Introdução à Criptografia*, Editora Berkeley, São Paulo-SP,1.ed,2002.

[14] COUTINHO, S.C. *Números inteiros e Criptografia RSA*, Editora do IMPA, Rio de Janeiro-RJ,1.ed,2007.

[15] COUTINHO, S.C. *Primalidade em tempo polinomial*, Editora da SBM, Janeiro-RJ,1.ed,2004.

[16] COUTINHO, S.C. *Criptografia - Programa de Iniciação científica OBMEP*, Editora da SBM, Janeiro-RJ,1.ed.,2008.

[17] RINBEMBOIM, Paulo. *Números primos: mistérios e recordes*, Editora do IMPA, Janeiro-RJ,1.ed.,2001.

[18] LOVÁZ, L., PELIKÁN J. e VESZTERGOMBI K. *Matemática Discreta*, Editora da SBM, Janeiro-RJ,1.ed.,2003.

[19] LAURITZEN, Niels. *Concrete abstract algebra*, Editora Cambridge University Press, Cambridge-UK,5.ed.,2009.

[20] STEWART, Ian. *Concepts of modern mathematics*, Editora Dover Publications, INC, New York-NY-USA,1.ed.,1995.