Universidade Federal de Pernambuco
Centro de Informática

Pós-graduação em Ciência da Computação

Grupos pseudo-livres, primos seguros e criptografia RSA

Marcelo Gama da Silva

Dissertação de Mestrado

Recife
Fevereiro de 2007

Universidade Federal de Pernambuco
Centro de Informática

Marcelo Gama da Silva

Grupos pseudo-livres, primos seguros e criptografia RSA

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: *Prof. PhD. Ruy José Guerra B. de Queiroz*

Recife
Fevereiro de 2007

Silva, Marcelo Gama da
Grupos pseudo-livres, primos seguros e criptografia RSA / Marcelo Gama da Silva. – Recife : O autor, 2007.
xxi, 60 p.: il., fig., tab.

Dissertação (mestrado) – Universidade Federal de Pernambuco. CIN. Ciência da computação, 2007.

Inclui bibliografia.

1. Criptografia. 2. Grupos pseudo - Livres. 3. Primos seguros. I. Título.

652.8 CDD (22.ed.) MEI2007-016

Ao meu Pai (In memorian)

Agradecimentos

Ao Professor Ruy pela paciência, incentivo e orientação prestados.

Aos amigos Igor e Murilo pela força durante o mestrado e a e Francisco Valadares (Chicão) pelo incentivo e pelas discussões sobre algoritmos e complexidade.

Ao Centro de Informática da Universidade Federal de Pernambuco pela oportunidade que me foi dada para realizar este trabalho.

Finalmente, um agradecimento especial à minha esposa (Solange Santos) por toda paciência e dedicação durante minhas quase infindáveis horas de pesquisa e elaboração deste trabalho.

As pessoas passam suas vidas esperando que aconteça algo que vá mudar tudo. Elas buscam poder ou amor, e as respostas às suas maiores perguntas. Acredito que o que elas querem realmente é outra oportunidade. Algum jeito de levar outra vida, aonde todos os erros que cometeram possam ser apagados e elas possam começar tudo novamente, como se nada de ruim tivesse acontecido e todas as suas oportunidades estejam à frente delas.

— ALLIE CLARKE (Personagem da mini-série Taken)

Resumo

Quando um esquema criptográfico é definido sobre um grupo, criptografar mensagens equivale a fazer com que variáveis de alguma equação tomem valores nesse grupo, enquanto que quebrar esse esquema significa "descobrir" quais valores as variáveis tomaram. Portanto, a segurança de tais esquemas está associada à dificuldade de se resolver equações sobre grupos. A utilização de grupos livres seria uma possível solução para esse problema; entretanto, apenas equações "triviais" podem ser resolvidas sobre grupos livres. Além disso, os grupos livres são infinitos, o que não é interessante do ponto de vista computacional.

Uma alternativa foi proposta por Susan Hohenberger em 2003, dando origem à noção de "grupos pseudo-livres", refinada posteriormente por R. Rivest. Informalmente, um grupo pseudo-livre caracteriza-se por não poder ser distinguido, de modo eficiente, de um grupo livre. Do ponto de vista computacional, isto significa que a probabilidade de que se resolva uma equação não trivial sobre um grupo pseudo-livre é desprezível. Dessa forma, encontramos um ambiente adequado para lidarmos com questões de segurança de esquemas criptográficos.

Dois conceitos merecem destaque nesse contexto. O conceito de *grupos pseudo-livres*, como veremos a seguir, é de fundamental importância para a criptografia moderna, enquanto que o conceito de *primos seguros* tem sua relevância associada ao criptossistema RSA.

Este trabalho tem três objetivos principais. Inicialmente estaremos interessados em estudar alguns dos chamados *problemas computacionalmente difíceis* e sua utilização na construção de esquemas criptográficos seguros. Um outro objetivo é o estudo detalhado do *Teorema de Micciancio* sobre grupos pseudo-livres. Finalmente, voltaremos nossas atenções para a geração de primos seguros, pois estes estão diretamente relacionados com a segurança do criptossistema RSA. Em particular, propomos um novo algoritmo para geração de primos seguros que, através de um teorema devido a Euler e Lagrange e da lei de reciprocidade quadrática de Gauss, evita em grande parte os testes de primalidade.

Palavras-chave: RSA, grupos pseudo-livres, primos seguros

Abstract

When a cryptographic scheme is defined over a group, encrypting messages is equivalent to assigning values in this group to variables of some equation, whereas breaking this scheme is "to discover" which values were assigned to those variables. Therefore, the security of such schemes is connected to the hardness to solve equations over groups. The use of free groups would be a possible solution to this problem; however, only "trivial" equations can be solved over free groups. Moreover, free groups are infinite and this is not interesting from a computational perspective.

An alternative solution was proposed by Susan Hohenberger in 2003, originating the notion of "pseudo-free groups", wich was improved by R. Rivest. Informally, a pseudo-free group is characterized by the fact that it cannot be distinguished, in an efficient way, from a free group. From a computational point of view, this implies that the probability of solving a non-trivial equation over a pseudo-free group is negligible. So, this is a suitable environment to deal with questions about security of cryptographic schemes.

Two notions deserve to be pointed. The notion of *pseudo-free groups* is essential to modern cryptography whereas the notion of *safe primes* is relevanto to the RSA cryptosystem.

Our work has three main objectives. We start studying some of so called *computationally hard problems* and its use to develop safe cryptosystems. The second objective is the detailed study of the *Micciancio's theorem* on the pseudo-freeness of the RSA groups. Finally, we will study the generation of safe primes, because these numbers are related to the RSA security. In particular, we propose a new algorithm to generate safe primes wich uses a theorem from Euler and Lagrange and the Gauss's quadratic reciprocity law to avoid most primality tests.

Keywords: RSA, pseudo-free groups, safe primes

Sumário

Lista de Figuras

Lista de Tabelas

Nomenclatura

$\Delta(X,Y)$ distância estatística entre as distribuições X e Y, página 17

$\left(\frac{a}{p}\right)$ símbolo de Legendre de a módulo p, página 14

$\mathbb{Z}_N^*$ grupo dos elementos inversíveis módulo n, página 1

$|A|$ cardinalidade do conjunto A, página 11

$\phi(N)$ valor da funcão de Euler em $N \in \mathbb{N} - \{0\}$, página 8

$\pi(x)$ quantidade de números primos que não excedem x, página 15

$\{0,1\}^*$ cadeias de 0's e 1's de tamanho arbitrário, página 16

$a \equiv b \pmod{n}$ a e b deixam o mesmo resto na divisão por n, página 8

$a \pmod{b}$ resto da divisão de a por b, página 7

$A \leq_{poly} B$ a linguagem B é redutível à linguagem A em tempo polinomial, página 20

$A \simeq B$ as estruturas algébricas A e B são isomorfas, página 14

$b \mid a$ o inteiro b é divisor do inteiro a, página 11

$b \nmid a$ o inteiro b não é divisor do inteiro a, página 11

$o(a)$ ordem de um elemento a em um grupo, página 13

PF problema da fatoração de inteiros, página 23

PLD problema do logarítmo discreto, página 25

PRQ problema dos resíduos quadráticos módulo n, página 24

PRSA problema RSA, página 24

PSQRT problema das raízes quadradas módulo n, página 25

PSS problema da soma de subconjuntos, página 26

RQ_N grupo dos resíduos quadráticos inversíveis módulo N, página 38

CAPÍTULO 1

Introdução

Não deixaremos de explorar e
ao término da nossa exploração
deveremos chegar ao ponto de partida
e conhecer esse lugar pela primeira vez.
– T. S. ELIOT

—

O uso da criptografia teve início com os egípcios, ainda que de forma rudimentar, há cerca de 4000 anos. Desde então esse ramo da matemática aplicada não pára de crescer. Entretanto, foi a partir da década de 1960, com a proliferação dos computadores, que se tornaram cada vez mais comuns a comunicação e o comércio digital e com eles a procura por serviços seguros. Isto deu um novo impulso à criptografia, culminando em 1977 no RSA (desenvolvido por R. **R**ivest, A. **S**hamir e L. **A**dleman), que é um dos esquemas criptográficos mais utilizados atualmente.

Os processos de cifrar e decifrar mensagens são, em um certo sentido, inversos um do outro. Essa necessidade de inversão torna a estrutura algébrica de grupo um possível ambiente para o desenvolvimento da criptografia moderna. Atualmente existem inúmeros algoritmos criptográficos e que têm como suporte os mais variados exemplos de grupos.

Um dos principais criptossistemas utilizados atualmente é o RSA, definido sobre os grupos $\mathbb{Z}_N^*$, onde $N = pq$ é produto de dois primos. Trabalhos recentes indicam que tipos de primos são mais apropriados para garantir uma relativa segurança a esses criptossistemas. Questões sobre a segurança do RSA serão nosso principal objetivo nesse trabalho.

1.1 Organização do trabalho

O presente trabalho está organizado do seguinte modo:

No capítulo 2, *Uma breve viagem pelo mundo da criptografia*, como o próprio nome diz, faremos uma breve discussão sobre a evolução dos esquemas criptográficos. É nesse capítulo que será introduzida a terminologia utilizada atualmente em criptografia. Daremos também uma visão geral dos aspectos mais importantes da criptografia moderna.

O capítulo 3, *Fundamentos teóricos,* tem como objetivo apresentar os principais conceitos teóricos e resultados relacionados necessários para o desenvolvimento desse trabalho.

O capítulo 4 será dedicado ao estudo dos chamados *problemas computacionalmente difíceis*. Nesse capítulo estudaremos a relação entre esses problemas e a segurança de alguns dos criptossistemas associados a eles.

O capítulo 5, por sua vez, trata da noção de *grupos pseudo-livres*. Embora recente, esse conceito mostra-se promissor por permitir uma generalização de muitas das hipóteses criptográficas assumidas atualmente. O ponto principal desse capítulo é a prova de que o grupo $\mathbb{Z}_N^*$, sob certas condições, é pseudo-livre.

No capítulo 6, *Primos seguros,* faremos um estudo de algumas das principais propriedades desses números primos especiais. Os únicos exemplos que se conhece de grupos pseudo-livres são os grupos dos elementos inversíveis de $\mathbb{Z}_N^*$, onde N é produto de dois primos seguros. Isto evidencia a importância desses números para a criptografia RSA. Ainda nesse capítulo serão apresentados algoritmos para a geração de primos seguros, um dos quais é inédito.

O capítulo 7 será dedicado a algumas *Conclusões* relativas ao estudo de criptossistemas e previsões de *Trabalhos futuros*, destacando possíveis desdobramentos desta dissertação.

1.2 Contribuições deste trabalho

Devido à importância já mencionada dos primos seguros para a criptografia RSA, existe atualmente uma grande necessidade de algoritmos rápidos para a geração de tais números. Para isso diversas classes de algoritmos são utilizadas. Entretanto, praticamente todos eles têm nos testes de primalidade seu principal problema. Embora tenha sido provado em 2001 que é possível decidir se um inteiro é primo em tempo polinomial (ver [1]), os algoritmos para isso ainda são bastante ineficientes. A principal contribuição desse trabalho é o desenvolvimento de um novo algoritmo para a geração de primos seguros que evita grande parte dos testes de primalidade utilizados pelos algoritmos tradicionais.

CAPÍTULO 2

Uma breve viagem pelo mundo da criptografia

Eu viajo não para ir a lugar algum, mas para ir.
Eu viajo pelo propósito de viajar.
A grande sedução é se mover.
– ROBERT LOUIS STEVENSON

—

2.1 Introdução

Os primeiros relatos sobre códigos secretos de que se tem conhecimento datam da época de Heródoto (considerado o "pai da história"). Desde então, as técnicas para a produção de escritas secretas, assim como suas aplicações, têm sido desenvolvidas e aperfeiçoadas continuamente. Em princípio, sua utilização era quase totalmente restrita a fins militares, sempre com o intuito de obter alguma vantagem sobre o inimigo; vantagem esta que, em alguns casos, decidiram batalhas. Um exemplo disso, conforme nos conta o próprio Heródoto, em sua obra *As Histórias*, é o desfecho da guerra entre gregos e persas no século V a.C.. A vitória dos gregos só foi possível devido ao conhecimento dos planos do inimigo. Um espião grego, infiltrado entre os persas, enviava informações que eram escritas em tabuletas de madeira e depois cobertas com cera, dando a impressão de que as tabuletas estavam em branco, sendo assim "inofensivas".

A técnica de ocultar uma mensagem é conhecida como *esteganografia*. Esta palavra é derivada das palavras gregas *steganos* (encoberto) e *graphein* (escrita). Foram muitas as formas de esteganografia desenvolvidas, entretanto essa técnica tinha uma grande fraqueza: caso fosse descoberta, o conteúdo da mensagem e, possivelmente, seu emissor, seriam decobertos. Por essa razão, paralelamente ao uso da esteganografia, desenvolveu-se a chamada *criptografia*. A origem desse termo são as palavras gregas *kriptos* (escondido) e *graphein* (escrita). Com o uso de técnicas de criptografia e esteganografia combinadas pode-se, além de esconder uma mensagem, proteger seu significado. Assim a mensagem será ilegível, ainda que tenha sido descoberta. O processo de criptografar uma mensagem é também conhecido como *cifragem* ou *encriptação*.

Os primeiros esquemas criptográficos eram relativamente simples. Um dos mais antigos que conhecemos, segundo [45], é chamado de *Cifra de César*. Este método de cifrar mensagens consiste em substituir uma letra de um alfabeto por outra através de um deslocamento cíclico nesse alfabeto. O esquema a seguir, com deslocamento 3, é o que era usado pelo imperador

Júlio César [45].

CIFRA DE CÉSAR													
Letra do alfabeto	A	B	C	D	E	F	G	H	I	J	K	L	M
Substituir por	D	E	F	G	H	I	J	K	L	M	N	O	P
Letra do alfabeto	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Substituir por	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Tabela 2.1 Cifra de César com deslocamento 3

Utilizando a cifra de César, o texto *IMPERADOR DE ROMA* seria cifrado como

LPSHUDGRU GH URPD

2.2 Definição e objetivos da criptografia

Menezes, Oorschot e Vanstone [30] dão a seguinte definição de criptografia:

Definição 2.1 (Criptografia). É o estudo de técnicas matemáticas relacionadas a aspectos de segurança da informação tais como confidencialidade, integridade de dados, autenticação de entidades e da origem de dados.

Ainda, de acordo com [30], os objetivos de um esquema criptográfico são os descritos como:

(1) **Confidencialidade.** Também chamada de privacidade, significa manter o conteúdo de uma informação fora do alcance de todos, exceto das pessoas autorizadas.

(2) **Integridade de dados.** Tem a função de detectar qualquer alteração não autorizada de mensagens, tais como inserção, remoção ou substituição de cadeias de caracteres.

(3) **Autenticação.** Está relacionada com identificação. Ocorre em dois níveis: autenticação das partes (as partes que estão se comunicando devem identificar-se) e autenticação da origem dos dados.

(4) **Não repudiação.** Assegura que a autoria de uma mensagem não possa ser negada. Dessa forma, o autor de uma mensagem não pode isentar-se de compromissos ou implicações decorrentes do conteúdo de sua mensagem. Como exemplo, digamos que o emissor compromete-se a fazer o pagamento correspondente a aquisição de determinado produto; a característica de não repudiação é o que permite identificar o autor no caso de disputa judicial pelo não cumprimento desse acordo.

2.3 Tipos de cifras

Existem basicamente dois modos de cifrar uma mensagem. Um deles é usando *substituição* onde, como o próprio termo diz, as letras e símbolos de um alfabeto são substituídos por outras letras e símbolos desse ou de outro alfabeto. As cifras de substituição podem ser classificadas em quatro grupos:

Substituição monoalfabética

Nesse tipo de cifragem é estabelecida uma bijeção entre as letras do alfabeto original e as de algum alfabeto (não necessariamente distinto). Cada letra do texto é então substituída pela sua correspondente via essa bijeção.

- A cifra de César é um exemplo de cifra de substituição monoalfabética.

Substituição polialfabética

O texto original é dividido em vários grupos de letras e para cada um desses grupos é feita uma substituição monoalfabética distinta.

- O representante mais conhecido desse modelo de cifragem é a *Cifra de Vigenère* que por muito tempo foi chamada de *a cifra indecifrável* [45].

A técnicas de substituição foram utilizadas durante muitos séculos até que uma invenção de criptoanalistas (aqueles que decifram códigos) árabes, que recebeu o nome de *análise de feqüências*, passou a ser utilizada para decifrar mensagens. Um dos trabalhos pioneiros em criptoanálise usando esse método pode ser encontrado na obra *Um manuscrito sobre decifração de mensagens criptográficas* do estudioso árabe Al Kindi, datado do século IX a.C..

A análise de frequências de um idioma identifica a frequência com que uma letra ou grupo de letras ocorre nesse idioma. Embora não se possa determinar essas frequências com exatidão, a análise de um grande número de textos, escolhidos de forma aleatória, dá uma boa idéia de quais letras ou símbolos ocorrem mais frequentemente. Assim, quando um texto em português é cifrado dessa forma, a letra ou símbolo mais frequente corresponderá, muito provavelmente, à letra A, o segundo ao E e assim por diante. Com essa invenção árabe as cifras de substituição não podiam mais ser consideradas seguras. Sendo assim surgiram novos modelos de cifra de substituição:

Substituição homofônica

Nesse modelo cada letra está associada a várias outras que são utilizadas durante a cifragem. Pode-se, por exemplo, associar a cada letra aproximadamente tantos símbolos quanto for sua freqüência. Dessa forma cada letra terá freqüência aproximadamente igual a 1.

Substituição poligrâmica

Aqui as letras não são mais substituídas individualmente, mas em grupos maiores. Por exemplo, pode-se substituir um par de letras por outro par de letras. Nesse caso, dispomos de $26 \times 26 = 676$ símbolos distintos.

Outro modelo de cifragem é formado pelas chamadas cifras de *transposição*. Estas cifras consistem em permutar as letras e símbolos que compõem uma mensagem segundo alguma regra conhecida pelo emissor e pelo receptor dessa mensagem. Mesmo para um texto pequeno, existe um número muito grande de transposições possíveis. O texto

CIFRASDETRANSPOSIÇÃO

por exemplo, admite cerca de $2,5 \times 10^{16}$ transposições distintas, o que torma extremamente difícil saber que

TORDCASOINRASFIAPSEC

é uma dessas transposições.

2.4 Cifragem simétrica $\times$ cifragem assimétrica

A informação que permite decifrar uma mensagem é conhecida como *chave*. Na cifra de César, por exemplo, a chave é saber o deslocamento que foi feito. Numa transposição a chave é a regra utilizada para permutar as letras.

De acordo com o tipo de chave utilizada, um esquema criptográfico pode ser *simétrico* ou *assimétrico*. Em um esquema simétrico a mesma chave é utilizada tanto para cifrar, quanto para decifrar uma dada mensagem. Considere, por exemplo, a seguinte "regra" de cifragem:

Adicione, se necessário, um caractere ao texto para que este contenha um número par de caracteres. Divida o texto em pares de caracteres consecutivos. Em seguida, permute os caracteres de cada par.

Com esse esquema, o texto "SIMÉTRICOS" é cifrado como "ISEMRTCISO". É fácil notar que o mesmo esquema reverte o processo.

Até metade da década de 1970 os esquemas de cifragem utilizados eram simétricos. Tais sistemas eram confiáveis, entretanto surgiu um problema no estabelecimento das chaves. Como duas pessoas quaisquer, que poderiam estar em quaisquer pontos do planeta, concordariam

sobre a chave que seria utilizada? Como fazer isto mantendo absoluto segredo? Este é chamado de *problema da distribuição (ou troca) de chaves*. Como diz Singh [45]:

> *Todo problema de distribuição (de chaves) é uma clássica situação de beco sem saída. Se duas pessoas querem trocar mensagens secretas, o remetente deve cifrá-las. Para cifrar a mensagem secreta ele deve usar uma chave, também secreta. Daí surge o problema de transmitir uma chave secreta ao receptor. Resumindo, antes que duas pessoas possam partilhar um segredo (a mensagem), elas devem partilhar outro segredo (a chave).*

O problema da distribuição de chaves era, até então, o grande problema da criptografia. Isto durou até o ano de 1976 quando W. Diffie e M. Hellman publicaram o artigo *New Directions in Cryptography* [13], onde descreviam uma solução para o problema da troca de chaves.

Embora a solução seja simples, a idéia levou séculos para ser concebida. Este é o modelo por eles proposto:

Esquema Diffie-Hellman para troca de chaves

Digamos que duas pessoas, A e B, queiram estabelecer uma chave comum para troca de mensagens cifradas.

(1) A e B escolhem, de comum acordo, um primo p e um gerador g do grupo cíclico $\mathbb{Z}_p^*$, ambos não necessariamente secretos.

(2) A escolhe (secretamente) um inteiro x e envia para B o número $g^x \pmod p$.

(3) B escolhe (secretamente) um inteiro y e envia para A o número $g^y \pmod p$.

(4) Ao receber $g^y \pmod p$ de B, A calcula a "chave"

$$K_A = (g^y)^x \pmod p = g^{xy} \pmod p = K.$$

Depois disso x não é mais necessário.

(5) Ao receber $g^x \pmod p$ de A, B calcula a "mesma chave"

$$K_B = (g^x)^y \pmod p = g^{xy} \pmod p = K.$$

Depois disso y não é mais necessário.

Durante o processo os valores de $g^x \pmod p$ e $g^y \pmod p$ são públicos. Nesse caso, o conhecimento de x (ou de y) dá a quem o possui o valor da chave K. Caso um indivíduo conheça, por exemplo, o valor de x, ele pode calcular a chave K do mesmo modo que A.

O problema de se encontrar o valor de x a partir de g e $g^x \pmod p$ é conhecido como *Problema do logarítmo discreto* (*PLD*), que é considerado um problema computacionalmente difícil. No capítulo 4 estudaremos, em mais detalhes, a noção de computacionalmente difícil e alguns problemas relacionados.

Ainda em seu artigo, Diffie e Hellman propõem a chamada criptografia assimétrica ou de chave pública. Embora não dispusessem de um modelo concreto, o conceito de criptografia assimétrica já era, por si mesmo, revolucionário. Nesse modelo, o *receptor* produz duas chaves. Uma delas é mantida em segredo (chave privada) enquanto que a outra deve ser conhecida por qualquer um que deseje enviar-lhe uma mensagem (chave pública). A idéia é que uma mensagem cifrada com a chave pública pode ser decifrada *apenas* pelo receptor com sua chave privada.

Não demorou muito até que o primeiro modelo de sistema de chave pública fosse concebido. Em 1977, Ronald **R**ivest, Adi **S**hamir e Leonard **A**dleman anunciaram a criação do esquema criptográfico que foi batizado de RSA [41]. Embora existam outros criptossistemas de chave pública, tais como o *ElGamal, Rabin, Merkle-Hellman, Goldwasser-Micali*, o RSA tornou-se o mais amplamente usado entre todos. A seguir é dada uma descrição de seu funcionamento:

Sistema de chave pública RSA

• **Parâmetros**

(1) São escolhidos dois números primos ímpares p e q. Em seguida, calculam-se os números

$$N = pq \quad \text{e} \quad \phi(N) = (p-1)(q-1).$$

(2) Escolhe-se um inteiro d tal que

$$1 < d < \phi(N) \quad \text{e} \quad mdc(d, \phi(N)) = 1.$$

(3) Calcula-se o (único) inteiro e, entre 1 e $\phi(N)$, tal que

$$d.e \equiv 1 \pmod{\phi(N)}.$$

(4) Publica-se o par (N, e) que será a chave pública do sistema e mantém-se em segredo a chave privada d.

• **Funcionamento**

Suponha que o indivíduo A deseja cifar uma mensagem $\mathscr{M}$ e enviá-la para B. Inicialmente $\mathscr{M}$ é convertida em um inteiro M por algum processo.

(1) A calcula $C = M^e \pmod{N}$ e o envia para o receptor B.

(2) O receptor B, ao receber C, calcula $C^d \pmod{N} = M^{ed} \pmod{N} = M$.

A última igualdade decorre das propriedades da função ϕ de Euler (ver 3.5).

Observe que, em parte, a segurança do RSA está associada à dificuldade de alguém não autorizado conseguir descobrir o valor do parâmetro d. Trataremos de questões como essa no capítulo 4, quando estudarmos alguns problemas computacionalmentes difíceis associados ao RSA.

2.5 Assinaturas digitais

Um outro aspecto importante da criptografia moderna é a chamada *assinatura digital*. Em muitas das transações realizadas por meios eletrônicos, as partes envolvidas precisam certificar-se sobre a autoria das mensagens.

Quando tratamos de documentos escritos ou impressos, a assinatura feita "à mão" é o que assegura a autoria do documento. Embora medidas de segurança sejam adotadas, é possível que alguma assinatura forjada não seja detectada. A abordagem utilizada para lidar com questões de autenticidade de assinaturas digitais, assim como no caso das cifragens, é baseda na chamada complexidade computacional. Informalmente, segundo Goldreich [17], um esquema de assinaturas digitais tem três pré-requisitos:

(1) Cada usuário deve ser capaz de gerar, de modo eficiente, sua própria assinatura.

(2) Cada usuário deve ser capaz de verificar, de modo eficiente, se uma dada assinatura pertence a determinado usuário.

(3) O processo de forjar uma assinatura não pode ser executado de modo eficiente.

Nos ítens anteriores, a afirmação de que algo pode ser executado de forma eficiente significa dizer que a computação necessária para essa tarefa pode ser executada em tempo polinomial probabilístico (ver 4.1.3).

A utilização de criptografia de chave pública proporciona um esquema de assinatura digital de simples implementação.

Assinatura digital utilizando criptografia de chave pública RSA

Suponha que um usuário A deseja enviar uma mensagem assinada para outro usuário B. Digamos que suas chaves públicas sejam, respectivamente, (N_A, e_A) e (N_B, e_B), enquanto que d_A e d_B são suas chaves privadas.

(1) Dada a mensagem M, A calcula $S = M^{d_A} \pmod{N_A}$.

(2) A cifra S com a chave pública de B, produzindo $C = S^{e_B} \pmod{N_B}$ que é enviado para B.

Note que A é o único usuário que conhece d_A, sendo, portanto, o único capaz de produzir S. Assim, S é uma *assinatura* da mensagem M, produzida por A. Quando decifra C com sua chave privada d_B, B obtém

$$C^{d_B} \pmod{N_B} = (S^{e_B})^{d_B} \pmod{N_B} = S.$$

Nesse ponto, como a chave pública e_A é a única que pode reverter d_A, B calcula

$$S^{e_A} \pmod{N_A} = (M^{d_A})^{e_A} \pmod{N_A} = M.$$

Ao obter a mensagem *legível* M, B tem a certeza de que esta foi escrita por A.

Finalizamos este capítulo fazendo algumas considerações sobre a segurança dos sistemas de criptografia de chave pública RSA.

Considere uma instância do RSA com parâmetros p, q (primos ímpares distintos), $N = pq$, $\phi(N) = (p-1)(q-1)$, e, d, com $ed = 1 \pmod{\phi(N)}$. Um modo óbvio de conseguir decifrar uma mensagem é obtendo o parâmetro d. Como o valor de e é público, pode-se tentar resolver a equação $ed = 1 \pmod{\phi(N)}$. Para isso é necessário conhecer o valor de $\phi(N)$. Caso os primos p e q sejam conhecidos, pode-se facilmente calcular esse valor, pois

$$\phi(N) = (p-1)(q-1) = pq - p - q + 1 = N + 1 - (p+q).$$

Sendo $N = pq$ público, p e q podem ser obtidos se N puder ser fatorado. Entretanto, ainda não se conhece um algoritmo eficiente para fatoração de inteiros. Este é, como veremos no capítulo 4, mais um problema computacionalmente difícil. Isto significa que o tempo necessário para fatorar o inteiro N é muitas vezes maior do que o tempo de "vida útil" da mensagem. Portanto, pode-se dizer que a segurança do RSA reside, pelo menos em parte, no fato que fatorar inteiros tem custo computacional muito elevado.

CAPÍTULO 3

Fundamentos teóricos

Deus fez os números naturais;
todo o resto é obra dos homens.
– LEOPOLD KRONECKER

—

Este capítulo tem como principal objetivo, descrever os principais resultados teóricos que serão úteis nos capítulos posteriores, bem como fixar a notação a ser utilizada.

3.1 Cardinalidade de conjuntos

Dado um conjunto A, sua cardinalidade indica o número de elementos que fazem parte desse conjunto. Usaremos a notação $|A|$ para indicar cardinalidade.

3.2 Divisibilidade

Dizemos que o inteiro a é *divisível* pelo inteiro b (ou que b é *divisor* de a ou ainda que a é *múltiplo* de b) quando existe um inteiro c tal que $a = b \cdot c$. Usaremos a notação $b \mid a$ para indicar que b é divisor de a; caso contrário, escreveremos $b \nmid a$.

3.3 Congruências

A noção de congruência foi introduzida por Gauss em sua notável obra *Disquisitiones Arithmeticae* [16]. Dois inteiros a e b são ditos *congruentes* módulo o inteiro positivo n quando esses deixam o mesmo resto na divisão por n. Equivalentemente, dois inteiros a e b são congruentes módulo n quando $a - b$ é múltiplo de n.

A notação $a \equiv b \pmod{n}$, criada por Gauss, é utilizada até hoje para indicar congruência de inteiros. Escrevemos $a \pmod{n} = b$ para indicar que b é o resto da divisão de a por n.

3.4 O pequeno teorema de Fermat

Em uma carta a seu amigo Frénicle de Bessy, datada de 18 de Outubro de 1640, Pierre de Fermat afirma:

p divide $a^{p-1}-1$ sempre que p é primo, com a e p primos entre si.

Em notação moderna temos (ver [11])

Teorema 3.1 (Pequeno teorema de Fermat (PTF)). *Se p é um número primo e $mdc(a,p)=1$, então $a^{p-1} \equiv 1 \pmod{p}$.*

A primeira prova desse resultado foi publicada em 1736 por L. Euler no artigo *Theorematum Quorundam ad Numeros Primos Spectantium Demonstratio*.

O menor inteiro positivo r tal que $a^r \equiv 1 \pmod{p}$ é chamado de *ordem de a módulo p*.

3.5 O teorema de Euler

A função de Euler, $\phi : \mathbb{N} - \{0\} \to \mathbb{N}$, é definida por

$$\phi(n) = |\{x \in \mathbb{N} : 1 \leq x < n \text{ e } \text{mdc}(x,n) = 1\}|,$$

ou seja, $\phi(n)$ é a quantidade de inteiros entre 1 e n que são primos com n. Conhecendo a fatoração de n em fatores primos, $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$, tem-se

$$\phi(n) = n\left(1-\frac{1}{p_1}\right)\left(1-\frac{1}{p_2}\right)\ldots\left(1-\frac{1}{p_k}\right).$$

Vale o seguinte resultado (ver [11]):

Teorema 3.2 (Euler). *Se $mdc(a,n)=1$, então $a^{\phi(n)} \equiv 1 \pmod{n}$.*

Observando que, para p primo, tem-se $\phi(p) = p-1$, podemos dizer que o Teorema de Euler é uma generalização do Pequeno teorema de Fermat.

Uma outra consequência do Teorema de Euler que permite a decifragem do criptossistema RSA é

Corolário 3.1. *Se $mdc(a,n)=1$ e $x \equiv y \pmod{\phi(n)}$, então $a^x \equiv a^y \pmod{n}$.*

3.6 O teorema chinês dos restos

Utilizado em alguns esquemas criptográficos, este resultado estabelece condições para que um sistema de congruências tenha solução (ver [11]).

Teorema 3.3 (Teorema chinês dos restos (TCR)). *Se os inteiros $n_1, n_2, \ldots, n_k$ são dois a dois coprimos, isto é, $mdc(n_i, n_j) = 1$ para $i \neq j$, então o sistema*

$$\begin{cases} x \equiv a_1 \pmod{n_1} \\ x \equiv a_2 \pmod{n_2} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ x \equiv a_k \pmod{n_k} \end{cases}$$

admite uma única solução módulo $n = n_1 n_2 \ldots n_k$. Se b_i é tal que $bn/n_i \equiv 1 \pmod{n_i}$, então essa solução é dada por $x \equiv \sum_{i=1}^{k} a_i b_i n/n_i \pmod{n}$.

3.7 Grupos

Um conjunto não vazio G munido de uma operação $*$ é um grupo se as três condições a seguir são satisfeitas:

Associatividade: $(a*b)*c = a*(b*c)$, $\forall a, b, c \in G$.

Existência de unidade: $\exists e \in G$ tal que $e*a = a*e = a$, $\forall a \in G$.

Existência de inversos: $\forall a \in G$, $\exists b \in G$ tal que $a*b = b*a = e$. Usamos a notação $b = a^{-1}$.

A *ordem* de um grupo G é a sua cardinalidade e será denotada por $|G|$ ou $o(G)$. A ordem de um elemento $a \in G$, denotada por $o(a)$, é o menor inteiro positivo n tal que $a^n = e$. Sabe-se que em um grupo finito a ordem de qualquer elemento é um divisor da ordem do grupo. Esse resultado é uma consequência do *Teorema de Lagrange* que veremos a seguir.

Um *subgrupo* de um grupo G é um subconjunto não vazio $H \subseteq G$ que também é um grupo (com a mesma operação de G). Vale o seguinte resultado (ver [11]):

Teorema 3.4 (Lagrange). *Sejam G um grupo finito e H um subgrupo de G. Então $o(H)$ é um divisor de $o(G)$.*

Um grupo G é *cíclico* se existe algum elemento $g \in G$ tal que para cada $a \in G$ existe um inteiro $n(a)$ tal que $a = g^{n(a)}$. Um tal g é chamado de *gerador* do grupo.

Sabe-se que, se p é primo, então o grupo multiplicativo $\mathbb{Z}_p^* = \mathbb{Z}/p\mathbb{Z} - \{0\}$ é cíclico.

O produto cartesiano de dois grupos cíclicos finitos, com operação coordenada a coordenada, ainda é um grupo cíclico, desde que suas ordens sejam números primos entre si.

Dois grupos G_1 e G_2 são *isomorfos* quando existe uma aplicação bijetiva $f : G_1 \to G_2$ que preserva as operações dos grupos, ou seja, $f(a * b) = f(a) \circ f(b)$, onde $*$ é a operação do grupo G_1 e $\circ$ é a operação de G_2. Escrevemos $G_1 \simeq G_2$ para indicar que os grupos G_1 e G_2 são *isomorfos*.

Existe uma forma equivalente do TCR no contexto de teoria dos grupos [23]:

Teorema 3.5. *Se os inteiros $n_1, n_2, \ldots, n_k$ são dois a dois coprimos e $n = n_1 n_2 \ldots n_k$, então $\mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^* \times \cdots \times \mathbb{Z}_{n_k}^* \simeq \mathbb{Z}_n^*$.*

3.8 Resíduos quadráticos

Um inteiro $a \in \{1, 2, \ldots n-1\}$ e primo com a é *resíduo quadrático* módulo n quando existe um inteiro $x \in \{1, 2, \ldots n-1\}$ tal que $x^2 \equiv a \pmod{n}$.

Pode-se provar que se n é produto de primos distintos, então $(\mathbb{Z}/n\mathbb{Z})^*$ é cíclico [30]. Assim conclui-se que apenas metade de seus elementos são resíduos quadráticos módulo n. O conjunto dos elementos de $\mathbb{Z}_n^*$ que são resíduos quadráticos módulo n é denotado por RQ_n.

Para identificar resíduos quadráticos são definidos os símbolos de Legendre e de Jacobi:

Definição 3.1 (Símbolo de Legendre). Dado um primo p e um inteiro a não divisível por p, o *simbolo de Legendre* de a módulo p é:

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

Um inteiro é resíduo quadrático módulo p se, e somente se, seu símbolo de Legendre módulo p é igual a 1.

Definição 3.2 (Símbolo de Jacobi). Dado um inteiro $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$ e um inteiro a primo com n, o *simbolo de Jacobi* de a módulo n é definido em termos do símbolo de Legendre por:

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{\alpha_1} \left(\frac{a}{p_2}\right)^{\alpha_2} \cdots \left(\frac{a}{p_k}\right)^{\alpha_k}$$

Diferentemente do que ocorre com o símbolo de Legendre, pode ocorrer que um inteiro não seja resíduo quadrático módulo n, mesmo que seu símbolo de Jacobi módulo n seja igual a 1.

3.9 A lei de reciprocidade quadrática de Gauss

Um dos mais belos teoremas de toda a matemática relaciona os símbolos de legendre $\left(\frac{p}{q}\right)$ e $\left(\frac{q}{p}\right)$, onde p e q são primos ímpares distintos [23].

Teorema 3.6 (Lei de reciprocidade quadrática de Gauss). *Se p e q são primos ímpares distintos, então*

$$\left(\frac{p}{q}\right) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}} \left(\frac{q}{p}\right)$$

3.10 Teorema dos números primos

O teorema que mostraremos a seguir foi conjecturado por Legendre em 1798 e foi provado pela primeira vez por *Hadamard* e *de la Vallée Poussin* no ano de 1896. Uma conexão entre esse teorema e a *Hipótese de Riemann* (um dos principais problemas em aberto relacionado com diversas áreas da Matemática, Computação e Física), bem como a importância dos números primos para a criptografia, faz com que ele esteja sempre em destaque.

Teorema 3.7 (Teorema dos números primos). *Seja $\pi(x)$ a quantidade de números primos que não excedem o real x. Tem-se*

$$\lim_{x\to\infty} \frac{\pi(x)}{x/\ln x} = 1$$

O teorema dos números primos pode ser interpretado como: $\pi(x)$ é assintoticamente igual à função $x/\ln x$. Uma excelente introdução aos temas abordados até agora pode ser encontrada em [20].

3.11 Alfabetos, cadeias e linguagens

Um *alfabeto* é um conjunto finito de símbolos. Os elementos de um alfabeto recebem o nome de *letras*. Uma *cadeia de caracteres* (ou *string*) sobre um alfabeto é uma sequência finita de letras desse alfabeto concatenadas. O conjunto de todas as cadeias definidas sobre um alfabeto Σ é denotado por Σ^*. Frequentemente utiliza-se $\Sigma = \{0,1\}$.

O *comprimento* de uma cadeia w, denotado por $|w|$, é o número de letras concatenadas para formar essa cadeia (contadas as repetições). Um conjunto de palavras definidas sobre um alfabeto será chamado de *linguagem*.

3.12 Máquinas de Turing

Uma *máquina de Turing* é um dispositivo teórico, criado pelo matemático Alan Turing em 1936, que simula a lógica de algoritmos. Informalmente, uma máquina de Turing consiste de [47]:

1. Uma fita *infinita* dividida em células adjacentes.
2. Um *cabeçote* que pode ler, escrever ou apagar símbolos de uma alfabeto finito sobre a fita, bem como mover-se para a esquerda ou para a direita sobre a fita.
3. Uma tabela de instruções indicando que ação será executada a partir da configuração da máquina e do símbolo lido na fita. Essa tabela também é chamada de *função de transição*.
4. Um *registrador* que armazena o estado da máquina.

Formalmente, temos:

Definição 3.3. Um máquina de Turing M é uma 7-upla $(Q, \Sigma, \Gamma, \delta, q_0, q_{ac}, q_{rej})$, onde

1. Q é o conjunto (finito) de estados possíveis da máquina.
2. Σ é um alfabeto que contém os símbolos que são entradas para M. Esse alfabeto não contém o símbolo $\square$ (espaço em branco).
3. Γ é o alfabeto da fita, onde $\square \in \Gamma$ e $\Sigma \subseteq \Gamma$.
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{E, D\}$ é a função de transição, onde E e D indicam, respectivamente, movimento do cabeçote para a esquerda ou para a direita.
5. $q_0 \in Q$ é o estado inicial de M.
6. q_{ac} é o estado de *aceitação*.
7. q_{rej} é o estado de *rejeição* ($q_{ac} \neq q_{rej}$).

Atingindo um dos estados q_{ac} ou q_{rej} a máquina M pára. Caso isto não ocorra, M entra em *loop*.

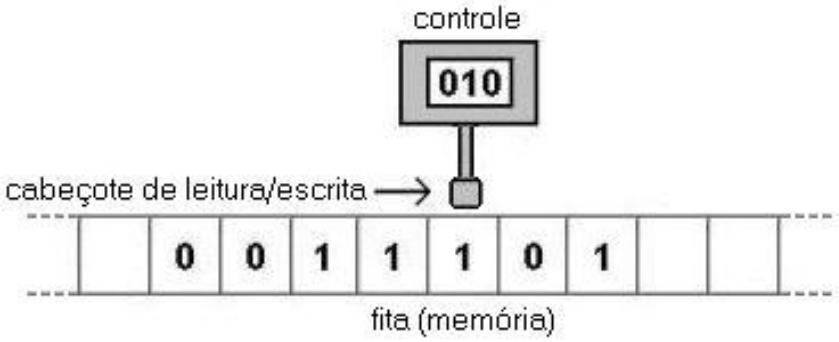

Figura 3.1 Máquina de Turing

3.13 Distância estatística

Definição 3.4 (Distância estatística). Sejam X e Y duas variáveis aleatórias discretas definidas sobre um conjunto finito A. A *distância estatística* entre essas duas distribuições é definida por

$$\Delta(X,Y) = \frac{1}{2}\sum_{a\in A} |Pr[X=a] - Pr[Y=a]|\,.$$

Informalmente, $\Delta(X,Y)$ é o maior valor possível para a diferença entre as probabilidades que essas duas distribuições podem atribuir a um mesmo evento.

Um resultado importante sobre distância estatística é dado a seguir (ver [18]):

Teorema 3.8. *Seja $p : A \to \{0,1\}$ uma função. Tem-se*

$$|Pr[p(X)=a] - Pr[p(Y)=a]| \leq \Delta(X,Y).$$

CAPÍTULO 4

Problemas computacionalmente difíceis

Se não existem soluções para grandes problemas,
o ideal é adaptar-se e preparar-se para conviver com eles...
– CARLOS A. C. COELHO (Denúncia vadia)

—

A noção de "problema computacionalmente difícil" é de fundamental importância para a criptografia moderna. Informalmente, a segurança de um esquema criptográfico depende da dificuldade encontrada por alguém não autorizado para inverter a função usada na cifragem. Essas funções são conhecidas como *funções unidirecionais*. Veremos neste capítulo diversas classes de problemas computacionalmente difíceis e seus respectivos criptossistemas associados.

4.1 Classes de complexidade

O ponto de principal importância em qualquer esquema criptográfico é oferecer extrema dificuldade a qualquer pessoa que tente decifrar uma mensagem que não lhe é destinada. Sendo assim, é natural a busca por problemas computacionalmente difíceis de serem resolvidos e utilizá-los como base para esquemas criptográficos.

Na teoria qualquer esquema criptográfico, por envolver um número finito de possibilidades de cifragem, pode ser quebrado. Entretanto, isto pode ter custo computacional bastante elevado.

Pensaremos em um dado problema como sendo uma cadeia de caracteres que o descreve e chamaremos de linguagem a um conjunto de cadeias que possui certa propriedade em comum. Nesse contexto, a complexidade de uma linguagem (e do problema associado) é medida pela quantidade de recursos (normalmente o número de operações realizadas) para decidir se uma determinada cadeia pertence a essa linguagem. Veremos a seguir uma breve descrição das principais classes de complexidade.

4.1.1 Classe de complexidade P

Uma linguagem L é reconhecida em *tempo polinomial determinístico* se existe uma máquina de Turing M, que retorna os valores 0 ou 1, e um polinômio $p : \mathbb{N} \to \mathbb{N}_+$ tais que

(1) Sobre uma entrada x, M pára após no máximo $p(|x|)$ passos, onde $|x|$ é o comprimento da cadeia x;

(2) $M(x) = 1$ se, e somente se, $x \in L$.

A classe de complexidade P é o conjunto das linguagens reconhecidas em tempo polinomial determinístico. De outro modo, podemos dizer que P é o conjunto dos problemas de decisão que podem ser resolvidos em tempo polinomial.

4.1.2 Classe de complexidade NP

Uma linguagem L está em NP quando existe uma relação binária $R_L \subseteq \{0,1\}^* \times \{0,1\}^*$ e um polinômio $p(.)$ tal que R_L pode ser reconhecida em tempo polinomial determinístico e $x \in L$ se, e somente se, existe $y \in \{0,1\}^*$ tal que $(x,y) \in R_L$ e $|y| \leq p(|x|)$. Um tal y é chamado de *testemunha* da pertinência de x a L.

A definição acima pode ser interpretada do seguinte modo:

Uma linguagem L está em NP se existem "provas curtas" de pertinência a L que podem ser eficientemente verificadas . O termo "prova curta" está associado à condição

$$\exists y \in \{0,1\}^* \text{ tal que } (x,y) \in L \text{ com } |y| \leq p(|x|).$$

A eficiência vem do fato que isto pode ser verificado em tempo polinomial.

Uma outra maneira de interpretar NP é como sendo o conjunto de todos os problemas de decisão que podem ser resolvidos por uma máquina de Turing não-determinística em tempo polinomial.

Pode-se (a grosso modo) dizer que P é a classe dos problemas cujas soluções são "fáceis de encontrar" e NP é a classe dos problemas cujas soluções são "fáceis de serem verificadas". Muitos dos teóricos em teoria da complexidade acreditam que P $\neq$NP (ver [9]). Este é o mais famoso problema em aberto em Ciência da Computação.

NP-Completude

Dizemos que uma linguagem L é NP-completa se L está em NP e se toda linguagem em NP é redutível a L em tempo polinomial. Uma linguagem L' é redutível a uma linguagem L se existe uma função computável f tal que $x \in L'$ se, e somente se, $f(x) \in L$; para indicar esse fato usamos a notação $L \leq_{poly} L'$.

Alguns problemas NP-completos conhecidos são [15, 24, 52]:

Satisfatibilidade de uma fórmula proposicional (SAT)

Chama-se *literal* uma variável proposicional ou a negação de uma variável proposicional. Uma disjunção de literais recebe o nome de *cláusula*. Uma fórmula proposicional está na *forma normal conjuntiva* (FNC) se esta é uma conjunção de cláusulas. Sabe-se que toda fórmula proposicional é equivalente a uma fórmula na forma normal conjuntiva.

Chamaremos de SAT ao conjunto de todas as fórmulas na forma normal conjuntiva que são satisfatíveis, ou seja, existe uma atribuição de valores às variáveis proposicionais que tornam a fórmula verdadeira. O problema SAT consiste, por definição, em verificar se uma expressão booleana na FNC é satisfatível. Em 1971 Stephen Cook provou que o SAT é NP-completo.

Problema da mochila (knapsack problem)

Dado um conjunto de ítens, cada um tendo um peso p_i e um valor v_i, determinar a quantidade de cada ítem que se pode tomar de modo que o peso total não exceda um dado valor p e o valor total seja o máximo possível.

Problema do caixeiro viajante

Um caixeiro viajante, que desloca-se entre pares de cidades a um custo que depende apenas do par escolhido, recebe um prêmio por cada cidade visitada e paga uma multa por cada cidade que deixa de visitar. Deseja-se estabelecer uma rota que minimize os custos de viagem e a soma das multas pagas, além de permitir que ele receba um dado prêmio pré-fixado.

Problema da clique máxima em um grafo

Uma *clique*[1] em um grafo não-direcionado é um conjunto C de vértices que são dois a dois adjacentes. Dizemos que uma clique C é *maximal* se não existe outra clique que a contenha. Quando não existe uma clique C_1 que seja maior (em quantidade de vértices) que C, então C é *máxima*. O problema de encontrar uma clique máxima em um grafo não-direcionado é NP-completo.

Problema da soma de subconjuntos

Dado um conjunto X de inteiros e um inteiro s, existe um subconjunto não vazio de X cuja soma de seus elementos seja s?

[1] Aqui o termo clique tem um significado pouco conhecido. Significa conspiração; subgrupo organizado no interior de um grupo cujos membros partilham preferências específicas; "panelinha".

Coloração de grafos (usando no mínimo 3 cores)

Assumiremos que *colorir* um grafo significa atribuir cores aos seus vértices de modo que dois vértices adjacentes tenham cores distintas. O problema da coloração de grafos está associado a diversos problemas de ordem prática como, por exemplo, programação, registro de alocação em compiladores e reconhecimento de padrões.

Determinar se é possível colorir os vértices de um grafo usando no máximo k cores é um problema NP-completo.

Problemas NP-difíceis

Um problema é chamado de NP-difícil quando existe algum problema NP-completo que pode ser reduzido a ele em tempo polinomial. O mais conhecido dos problemas NP-difíceis é o *problema da parada*. Este foi um dos primeiros problemas que se provou ser indecidível. A prova desse fato, dada por Alan Turing, foi publicada em 1936 [47].

Problema da parada

Dada a descrição de um programa e uma entrada finita, decidir se o programa pára ou entra em loop para aquela entrada.

Turing provou que não existe um algoritmo geral para resolver o problema da parada para todos os possíveis pares programa-entrada.

4.1.3 Classe de complexidade BPP

Dizemos que uma linguagem L é reconhecida por uma máquina de Turing probabilística de tempo polinomial M se existe uma constante $k > \frac{2}{3}$ tal que as duas condições a seguir são satisfeitas:

(1) $x \in L \Rightarrow Pr[M(x) = 1] \geq k$

(2) $x \notin L \Rightarrow Pr[M(x) = 0] \geq k$

O conjunto das linguagens reconhecidas por máquinas de Turing probabilísticas de tempo polinomial é denotado por BPP (Bounded-error Probabilistic Polinomial Time). Esta definição pode ser interpretada como:

(1) Se uma cadeia x está em L, existe uma alta probabilidade (dependendo de k) de que M decida que x está em L.

(2) Analogamente, se uma dada cadeia x não está em L, existe uma alta probabilidade de que M decida que x não está em L.

As condições (1) e (2) combinadas equivalem a: "Existe uma alta probabilidade de que M tome a decisão correta sobre a pertinência ou não de x à linguagem L".

4.2 Problemas computacionalmente difíceis

A intratabilidade de alguns problemas computacionais garante uma relativa segurança aos esquemas criptográficos assimétricos. Um problema é dito *computacionalmente difícil* quando não existe algoritmo que possa solucioná-lo em tempo polinomial ou, caso exista um tal algoritmo, encontrar uma solução é infactível.

Existem diversos problemas que são considerados computacionalmente difíceis, alguns dos quais estudaremos a seguir (ver [30], cap. 3):

4.2.1 Problema 1: Fatoração de inteiros (PF)

Dado um inteiro $n > 1$, o Teorema Fundamental da Aritmética afirma que n pode ser escrito de modo único, exceto pela ordem dos fatores, na forma

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k},$$

onde $p_1, p_2, \dots, p_k$ são números primos distintos e $\alpha_1, \alpha_2, \dots, \alpha_k \in \mathbb{N}^*$.

Dizemos que $a \in \mathbb{N}$ é um fator não trivial do inteiro n quando $n = ab$, com $1 < a, b < n$. Então o PF é descrito como

Definição 4.1 (Problema da fatoração de inteiros). Dado um inteiro n, encontrar um de seus fatores não triviais.

Em 2001, M. Agrawal, N. Kayal e N. Saxena provaram que o problema de decidir se um dado inteiro é primo pode ser resolvido em tempo polinomial [1]. Entretanto, ainda não sabemos se o mesmo ocorre com o problema de encontrar fatores não triviais de um dado inteiro.

O problema da fatoração de inteiros foi atacado por importantes matemáticos, dentre os quais está Gauss que em sua obra *Disquisitiones Arithmeticae* [16] escreveu:

> O problema de distinguir os números primos dos compostos e de resolver um inteiro em seus fatores primos é conhecido como sendo um dos mais importantes e úteis em aritmética. Ele tem ocupado o tempo e a sabedoria de antigos e modernos matemáticos de tal maneira que seria desnecessário discutir o problema em toda sua extensão...
> Além disso, a dignidade da própria ciência parece requerer a solução de um tão elegante e celebrado problema.

Estas palavras foram escritas por Gauss aproximadamente 175 anos antes que os testes de primalidade e o problema de fatoração de inteiros fossem aplicados em criptografia. Portanto, se esse problema era importante nos dias de Gauss, ele se torna ainda mais importante nos dias de hoje.

4.2.2 Problema 2: Problema RSA (PRSA)

O PRSA foi inicialmente proposto por R. Rivest, A. Shamir e L. Adleman, em um artigo [41] no ano de 1978. Considere uma mensagem $M \in \{0,1,\ldots,n-1\}$. Utilizando o RSA, M é cifrada como $C = M^e \pmod{n}$. Nesse contexto, o PRSA é descrito como

Definição 4.2 (Problema RSA). Dados os inteiros $e > 1$, $C \in \{0,1,\ldots,n-1\}$, encontrar um inteiro $M \in \{0,1,\ldots,n-1\}$ tal que $C = M^e \pmod{n}$.

Portanto, o PRSA consiste em encontrar raízes e-ésimas módulo o inteiro composto n. Suponha que conhecemos a fatoração de $n = p.q$. Podemos facilmente calcular $\phi(n)$:

$$\phi(n) = (p-1)(q-1) = pq - p - q + 1 = n - (p+q) + 1.$$

Assim, podemos encontrar a chave privada d, bastando, para isso, calcular o único inteiro d com $1 < d < \phi(n)$ e $de = 1 \pmod{\phi(n)}$. Com isso conclui-se que PRSA $\leq_{poly}$ PF.

4.2.3 Problema 3: Resíduos quadráticos (PRQ)

Seja p um número primo. Um inteiro a, não divisível por p, é dito ser um resíduo quadrático módulo p quando existe um inteiro x tal que $x^2 \equiv a \pmod{p}$. Como visto no capítulo anterior, o símbolo de Legendre $\left(\frac{a}{p}\right)$ é utilizado para indicar se isto ocorre ou não. O símbolo de Jacobi $\left(\frac{a}{n}\right)$ generaliza o símbolo de Legendre para um inteiro composto n. Podemos agora enunciar o PRQ:

Definição 4.3 (Problema dos resíduos quadráticos). Dado um inteiro composto n e um inteiro a, com $mdc(a,n) = 1$ e $\left(\frac{a}{n}\right) = 1$, decidir se a é um resíduo quadrático módulo n.

Embora existam algoritmos eficientes para calcular $\left(\frac{a}{p}\right)$ quando p é primo e assim decidir se a é ou não resíduo quadrático módulo p, não se conhece algoritmo eficiente que resolva o PRQ quando n é composto se não dispomos de uma fatoração de n como produto de primos.

Caso a fatoração de n em fatores primos seja conhecida, digamos $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$, então

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{\alpha_1} \left(\frac{a}{p_2}\right)^{\alpha_2} \cdots \left(\frac{a}{p_k}\right)^{\alpha_k}.$$

Portanto, PRQ$\leq_{poly} PF$.

4.2.4 Problema 4: Raízes quadradas módulo n (PSQRT)

O problema de encontrar raízes quadradas módulo n pode ser enunciado como segue:

Definição 4.4 (Problema das raízes quadradas módulo n). Dados os inteiros $n > 1$, composto, e $a \in \{2, 3, \ldots, -2\}$, encontrar $x \in \{2, 3, \ldots, n-2\}$ tal que $x^2 \equiv a \pmod{n}$.

Aqui o problema é um pouco diferente do problema anterior. Enquanto no PRQ estamos interessados apenas em saber se o inteiro a é ou não resíduo quadrático módulo n, no PSQRT podemos dispor dessa informação, mas queremos encontrar uma de suas raízes quadradas módulo n.

Novamente, temos um problema que pode ser resolvido de modo eficiente para n primo ou potência de primo. Para $n = p$ primo, se g é um gerador de $\mathbb{Z}_p^*$ e $a \in \{1, 2, 3, \ldots, p-1\}$, então $a \equiv g^r \pmod{p}$, para algum inteiro $r \geq 0$. Assim, a será ou não resíduo quadrático módulo p, dependendo de r ser par ou ímpar. Sendo r par, a equação $x^2 \equiv a \pmod{p}$ tem as soluções $x \equiv \pm g^{r/2} \pmod{p}$. Em [30] pode-se encontrar um algoritmo eficiente para determinar essas soluções.

Também aqui é possível encontrar uma relação entre esse problema e o PF. De fato, pode-se provar que o PSQRT e o PF são computacionalmente equivalentes ([30], Cap. 3).

4.2.5 Problema 5: Logaritmo discreto (PLD)

Seja G um grupo cíclico de ordem n e seja g um gerador desse grupo. Dado $b \in G$, existe um único inteiro x, com $0 \leq x < n$ tal que $b = g^x$. Dizemos que x é o *logaritmo discreto* de b na base g e escrevemos $x = log_g b$. Para enunciar o PLD tomaremos $G = \mathbb{Z}_p^*$, com p primo.

Definição 4.5 (Problema do logaritmo discreto). Dados os inteiros p, g, b, com p primo, g um gerador de $\mathbb{Z}_p^*$ e $\text{mdc}(b, n) = 1$, encontrar um inteiro x tal que $g^x \equiv b \pmod{p}$.

Sabe-se que a dificuldade do PLD é independente do gerador escolhido. Existem classes de algoritmos que resolvem o PLD em casos particulares, como no caso em que a ordem do grupo $\mathbb{Z}_p^*$ tem apenas fatores primos pequenos.

4.2.6 Problema 6: Soma de subconjuntos (PSS)

Esse problema pode ser caracterizado da seguinte forma:

Definição 4.6 (Problema da soma de subconjuntos). Dado um conjunto $A = \{a_1, a_2, \ldots, a_n\}$ de inteiros positivos e um inteiro $s > 0$, determinar se existe um subconjunto A' de A de modo que $\sum_{x \in A'} x = s$.

Existe um modo equivalente de definir esse problema:

Definição 4.7 (Problema da soma de subconjuntos (versão 2)). Dados os inteiros positivos $a_1, a_2, \ldots, a_n, s$, determinar se existe $(x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ tal que $\sum_{i=1}^{n} a_i x_i = s$.

O PSS é um problema de decisão. Sua versão computacional é descrita como:

Definição 4.8 (Problema da soma de subconjuntos (versão computacional)). Dados os inteiros positivos $a_1, a_2, \ldots, a_n, s$, determinar $(x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ tal que $\sum_{i=1}^{n} a_i x_i = s$.

É conhecido que o PSS é um problema NP-completo e que sua versão computacional é NP-difícil [30].

4.3 Problemas computacionalmente difíceis e criptossistemas associados

Vimos na seção anterior alguns dos principais problemas NP-difíceis. Estudaremos nesta seção o conceito de função unidirecional e de criptossistema e veremos exemplos de esquemas criptográficos associados aos problemas estudados anteriormente.

4.3.1 Funções unidirecionais

Informalmente, uma função f é chamada de *unidirecional* ou de *mão única* quando esta é fácil de calcular mas é difícil de inverter, ou seja, dado um elemento x no domínio de f, pode-se facilmente computar $f(x)$ mas, dado y no conjunto imagem de f, é extremamente difícil determinar x tal que $f(x) = y$.

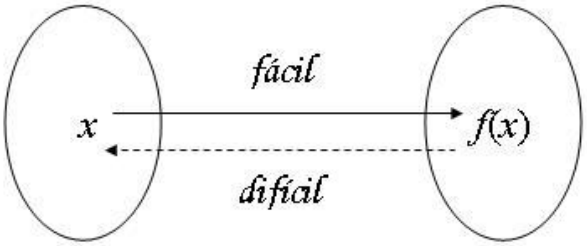

Figura 4.1 Funções unidirecionais

Uma definição rigorosa do conceito de função unidirecional, depende da noção de função desprezível:

Definição 4.9 (Função desprezível). Uma função $\nu : \mathbb{R} \to \mathbb{R}_+$ é dita *desprezível* se para toda constante $c \geq 0$ existe um inteiro k_c tal que $\nu(k) < k^{-c}$, para todo $k \geq k_c$.

Exemplo 4.1. A função $f(k) = a^{-k}$, com $a > 1$, é desprezível. Sabemos do cálculo diferencial que, dada qualquer constante $c \geq 0$, tem-se $\lim_{k\to\infty} \frac{k^c}{a^k} = 0$. Assim, para k suficientemente grande, tem-se $\frac{k^c}{a^k} < 1$, ou seja, $a^{-k} < k^{-c}$.

Existem algumas definições de funções unidirecionais, dependendo do rigor com se quer tratar cada problema. Optamos pela definição de *função unidirecional forte*, conforme [17].

Definição 4.10 (Função unidirecional forte). Uma função $f : \{0,1\}^* \to \{0,1\}^*$ é chamada de unidirecional (forte) se valem as seguintes condições:

1. f **é fácil de computar:** Existe um algoritmo determinístico de tempo polinomial A que, tomando x como entrada, produz $f(x)$.
2. f **é difícil de inverter:** Para todo algoritmo probabilístico de tempo polinomial A', existe uma função desprezível $\nu_{A'}$ tal que para todo inteiro k suficientemente grande

$$Pr\left[A'(f(U_k), 1^k) \in f^{-1}(f(U_k))\right] < \nu_{A'}(k).$$

Aqui U_k representa uma variável unformemente distribuída sobre $\{0,1\}^k$ e 1^k é um parâmetro que indica o tamanho k de uma cadeia em $\{0,1\}^*$ que deve ser produzida como saída pelo algoritmo A'. Assim, se f é unidirecional, a probabilidade que A' produza uma pré-imagem para $f(x)$ para um dado x é desprezível.

Para uso em esquemas criptográficos é mais conveniente pensar em coleções de funções unidirecionais.

Definição 4.11 (Coleção de funções unidirecionais fortes). Seja I um conjunto de índices e para cada $i \in I$ sejam D_i e R_i conjuntos finitos. Uma coleção de funções unidirecionais fortes é um conjunto $F = \{f : D_i \to R_i\}_{i\in I}$ satisfazendo às seguintes condições:

1. Existe um algoritmo probabilístico de tempo polinomial S_1 que tomando como entrada $a \in \{0,1\}^k$ produz um índice $i \in I$.
2. Existe um algoritmo probabilístico de tempo polinomial S_2 que tomando como entrada $i \in I$ produz $x \in D_i$.
3. Existe um algoritmo probabilístico de tempo polinomial A_1 tal que para $i \in I$ e $x \in D_i$, tem-se $A_1(i,x) = f_i(x)$.
4. Para todo algoritmo probabilístico de tempo polinomial A, existe uma função desprezível ν_A tal que, para todo k suficientemente grande, tem-se

$$Pr\left[A(f_i(U_k), 1^k) \in f_i^{-1}(f_i(U_k))\right] < \nu_A(k).$$

Funções unidirecionais por si só não são úteis para criptografia. A razão disso é que um esquema criptográfico deve permitir que o receptor tenha acesso ao conteúdo original da mensagem. Sendo assim, são definidas as funções *trapdoor*. Uma função trapdoor unidirecional é uma função unidirecional $f : X \to Y$ com a propriedade adicional de que, dada uma informação extra (chamada de informação trapdoor), pode-se, a partir de $y \in Im(f)$, calcular $x \in Dom(f)$ tal que $f(x) = y$. Formalmente temos:

Definição 4.12 (Função trapdoor unidirecional). Uma função trapdoor unidirecional é uma função unidirecional $f : X \to Y$ tal que existe um polinômio $p : \mathbb{R} \to \mathbb{R}_+$ e um algoritmo probabilístico de tempo polinomial B tal que para todo inteiro positivo k existe $t_k \in \{0,1\}^*$ com $|t_k| \leq p(k)$ e para todo $x \in X$, $B(f(x), t_k) = y$ e $f(y) = f(x)$.

Exemplo 4.2. Dados os números primos $p = 3904501$ e $q = 13487$, calcule seu produto $n = pq = 52660004987$. Defina a função $f : \mathbb{N} \to \mathbb{N}$ por $f(x) = x^2 \pmod{n}$. Conhecendo $f(x) = 52119910234 \pmod{n}$, não é tão simples o cálculo de um valor para x (aqui tem-se $x = 10^{12}$). Um computador rápido pode auxiliar nessa tarefa. Entretanto, se n tiver algo em torno de 200 algarismos (na base 10), isso pode levar alguns milênios!

Agora, suponhamos que alguma informação extra (informação trapdoor) nos é fornecida. Por exemplo, digamos que são dados os fatores primos de n. Nesse caso, resolvemos as congruências $x^2 \equiv c \pmod{p}$ e $x^2 \equiv c \pmod{q}$, que são relativamente mais fáceis de resolver, e usamos o TCR para encontrar uma solução para a congruência módulo n.

Como antes, é conveniente utilizar famílias de funções trapdoor:

Definição 4.13 (Coleção de funções trapdoor unidirecionais). Seja I um conjunto de índices e para cada $i \in I$ sejam D_i e R_i conjuntos finitos. Uma coleção de funções trapdoor unidirecionais fortes é um conjunto $F = \{f : D_i \to R_i\}_{i \in I}$ satisfazendo às seguintes condições:

1. Existe um polinômio $p : \mathbb{N} \to \mathbb{N}$ e um algoritmo de tempo polinomial S_1 que tomando como entrada $a \in \{0,1\}^k$ produz pares (i, t_i), onde $i \in I$ e $|t_i < p(k)|$ (t_i é chamado de trapdoor de i).
2. Existe um algoritmo de tempo polinomial S_2 que tomando como entrada $i \in I$ produz $x \in D_i$.
3. Existe um algoritmo de tempo polinomial A_1 tal que para $i \in I$ e $x \in D_i$, $A_1(i,x) = f_i(x)$.
4. Para todo algoritmo probabilístico de tempo polinomial A, existe uma função desprezível ν_A tal que, para todo k suficientemente grande, tem-se

$$Pr\left[A(f_i(U_k), 1^k) \in f_i^{-1}(f_i(U_k))\right] < \nu_A(k).$$

4.3.2 Criptossistemas

Como visto no capítulo 1, existem basicamente dois tipos de esquemas criptográficos. Um modelo é formado pelos esquemas *simétricos,* ou de *chave secreta,* que utilizam uma única chave tanto para cifragem quanto para decifragem. O outro modelo é composto dos esquemas *assimétricos,* ou de *chave pública*, cujas chaves para cifragem e decifragem são distintas. Embora sejam raramente encontrados, existem uns poucos modelos de criptossistemas além desse dois, como o modelo sem chaves baseado no logaritmo discreto apresentado em [48]. Iniciamos com uma definição formal do conceito de criptossistema.

Definição 4.14 (Criptossistema simétrico). Um *esquema criptográfico simétrico* (ou de *chave secreta*) é uma quádrupla $ES = (\mathcal{K}, \mathcal{F}, \mathcal{E}, \mathcal{D})$ onde

1. $\mathcal{K}$ é um algoritmo aleatorizado, de tempo polinomial, que retorna cadeias de caracteres. Denotaremos por $Keys(ES)$ o conjunto de todas as cadeias que têm probabilidade não nula de serem produzidas por $\mathcal{K}$. Os elementos $K \in Keys(ES)$ são chamados de *chaves*.

2. $\mathcal{F} = \{ f_K : \{0,1\}^* \to \{0,1\}^* \}_{K \in Keys(ES)}$ é uma família de funções trapdoor unidirecionais.

3. $\mathcal{E}$ é um algoritmo de tempo polinomial que toma como entrada uma chave $K \in Keys(ES)$ e uma mensagem $M \in \{0,1\}^*$ e retorna o *texto cifrado* $\mathcal{E}(K,M) = f_K(M) \in \{0,1\}^*$, com $f_K \in \mathcal{F}$. Chamaremos $\mathcal{E}$ de *algoritmo de cifragem*.

4. $\mathcal{D}$ é um algoritmo de tempo polinomial que toma como entrada uma chave $K \in Keys(ES)$ e um texto cifrado $C \in \{0,1\}^*$ e retorna um elemento $\mathcal{D}(K,C) \in \{0,1\}^*$. Chamaremos $\mathcal{D}$ de *algoritmo de decifragem*.

5. Dado $K \in Keys(ES)$, vale a relação $\mathcal{D}(K, \mathcal{E}(K,M)) = M$.

A última condição na definição acima mostra que o esquema funciona se usarmos a mesma chave K tanto na cifragem quanto na decifragem, indicando tratar-se de um esquema simétrico.

Definição 4.15 (Criptossistema assimétrico). Um esquema criptográfico *assimétrico* (ou de *chave pública*) é uma quádrupla $EA = (\mathcal{K}, \mathcal{F}, \mathcal{E}, \mathcal{D})$ onde

1. $\mathcal{K}$ é um algoritmo aleatorizado de tempo polinomial que, tomando como entrada um *parâmetro de segurança* inteiro positivo k, retorna pares de cadeias de $\{0,1\}^*$. Vamos denotar por $Keys(EA)$ o conjunto de todos os pares de cadeias que têm probabilidade não nula de serem produzidos por $\mathcal{K}$. Dado um elemento $(K_e, K_d) \in Keys(EA)$, chamamos K_e de *chave pública* e K_d de *chave privada* do sistema.

2. $\mathcal{F} = \{ f_K : \{0,1\}^* \to \{0,1\}^* \}_K$, sendo K o conjunto das coordenadas dos elementos de $Keys(EA)$, é uma família de funções trapdoor unidirecionais.

3. $\mathcal{E}$ é um algoritmo de tempo polinomial que toma como entrada a coordenada K_e de um par $(K_e, K_d) \in Keys(EA)$ e uma mensagem $M \in \{0,1\}^*$ e retorna o *texto cifrado* $\mathcal{E}(K_e, M) = f_{K_e}(M) \in \{0,1\}^*$, onde $f_{K_e} \in \mathcal{F}$. Chamaremos $\mathcal{E}$ de *algoritmo de cifragem*.

4. $\mathcal{D}$ é um algoritmo de tempo polinomial que toma como entrada a coordenada K_d de um par $(K_e, K_d) \in Keys(EA)$ e um texto cifrado $C \in \{0,1\}^*$ e retorna um elemento $\mathcal{D}(K_d, C) = f_{K_d}(C) \in \{0,1\}^*$, onde $f_{K_d} \in \mathcal{F}$. Chamaremos $\mathcal{D}$ de *algoritmo de decifragem*.

5. Dado $(K_e, K_d) \in Keys(ES)$, vale a relação $\mathcal{D}(K_d, \mathcal{E}(K_e, M)) = M$.

A última condição nesta definição mostra que duas chaves K_e e K_d, possivelmente distintas, são utilizadas, indicando tratar-se de um esquema assimétrico.

Veremos a seguir alguns exemplos de criptossistemas baseados nos problemas estudados na seção anterior.

4.3.2.1 Criptossistema RSA

Trata-se do primeiro modelo de criptossistema de chave pública. O RSA foi proposto por Ronald Rivest, Adi Shamir e Leonard Adleman em 1977 e tem como suporte o PF. Seu funcionamento é descrito a seguir:

1. *Geração de chaves:*
   - Gere dois primos ímpares distintos p e q.
   - Calcule $n = pq$.
   - Gere um inteiro e satisfazendo $1 < e < \phi(n)$ e $\text{mdc}(e, \phi(n)) = 1$ e defina a chave pública $K_E = (n, e)$
   - Calcule o menor inteiro positivo d tal que $ed \equiv 1 \pmod{\phi(n)}$ e fixe a chave privada $K_D = d$.
2. *Cifragem:*
   - Dada a mensagem $M < n$, calcule $C = M^e \pmod{n}$.
3. *Decifragem:*
   - $M = C^d \pmod{n}$.

Sendo $ed = 1 \pmod{\phi(n)}$, tem-se $C^d \pmod{n} = M^{ed} \pmod{n} = M \pmod{n} = M$. Portanto, o processo de decifragem funciona corretamente. Este resultado ainda continua válido quando $\text{mdc}(M, n) \neq 1$ (ver [30]).

4.3.2.2 Criptossistema de Rabin

Este criptossistema de chave pública tem como base o PSQRT e foi proposto por Michael O. Rabin em janeiro de 1979 [37]. Para esse sistema temos:

1. *Geração de chaves:*
   - Gere dois primos ímpares distintos p e q.
   - Calcule $n = pq$ e defina a chave pública $K_e = n$.
   - Fixe a chave privada $K_d = (p, q)$.
2. *Cifragem:*
   - Dada a mensagem $M < n$, calcule $C = M^2 \pmod{n}$.
3. *Decifragem:*
   - Calcule as duas soluções das congruências $x = M^2 \pmod{p}$ e $y = M^2 \pmod{q}$.
   - Utilize o TCR para calcular as quatro soluções de $C = M^2 \pmod{n}$.
   - Uma das quatro soluções é a procurada.

4.3.2.3 Criptossistema Goldwasser-Micali

Este esquema criptográfico foi desenvolvido por Shafi Goldwasser e Silvio Micali no ano de 1982 e apoia-se no PRQ. Seu destaque deve-se ao fato de ter sido o primeiro criptossistema probabilístico de chave pública provado ser seguro sob as hipóteses criptográficas padrões. Um ponto negativo é sua ineficiência, pois o texto cifrado pode ser centenas (ou até milhares) de vezes maior que o texto original. Vamos à sua descrição:

1. *Geração de chaves:*
   - Gere dois primos ímpares distintos p e q e calcule $n = pq$.
   - Gere um inteiro y primo com n e que não é resíduo quadrático módulo n.
   - Defina a chave pública $K_e = (n, y)$.
   - Fixe a chave privada $K_d = (p, q)$.
2. *Cifragem:*
   - Dada a mensagem $M = m_1 m_2 \cdots m_r$, onde cada m_i é um bit de M, escolha (de forma aleatoria) um inteiro x_i menor que n e primo com n, para cada bit m_i de M.
   - Defina $e(m_i) = \begin{cases} y x_i^2 \pmod{n} & \text{se } m_i = 1 \\ x_i^2 \pmod{n} & \text{se } m_i = 0 \end{cases}$
   - Faça $C = e(m_1)e(m_2)\cdots e(m_r)$.

3. *Decifragem:*

   - Para cada i, com $1 \leq i \leq n$, faça $\begin{cases} m_i = 0 & \text{se } e(m_i) \text{ for resíduo quadrático módulo } n \\ m_i = 1 & \text{caso contrário} \end{cases}$

4.3.2.4 Criptossistema ElGamal

Este é mais um criptossistema de chave pública. O esquema, que foi descrito por Taher Elgamal em 1984, baseia-se no PLD e pode ser definido sobre qualquer grupo cíclico finito. Segue a sua descrição:

1. *Geração de chaves:*

   - Escolha um gerador g desse grupo.
   - Sendo $n = |G|$, escolha $x \in \{0, 1, \ldots, n-1\}$ e calcule $h = g^x$.
   - Defina chave pública $K_e = (G, n, g, h)$.
   - Fixe a chave privada $K_d = x$.

2. *Cifragem:*

   - Dada uma mensagem M, convertê-la em um elemento de $m \in G$.
   - Escolha aleatoriamente $y \in \{0, 1, \ldots n-1\}$ e calcule $c_1 = g^y$ e $c_2 = m \cdot h^y$.
   - Faça $C = (c_1, c_2)$.

3. *Decifragem:*

   - Calcule $c_2 \cdot (c_1^x)^{-1} = m \cdot h^y \cdot (g^{xy})^{-1} = m \cdot h^y \cdot (h^y)^{-1} = m$.

4.3.2.5 Criptossistema Naccache-Stern

Este sistema de chave pública determinístico tem como base o PSS e foi desenvolvido por David Naccache e Jacques Stern em 1997. Sua descrição é:

1. *Geração de chaves:*

   - Gere um primo grande p e um inteiro positivo r.
   - Para $0 \leq i \leq r$, faça $p_i = i$-ésimo número primo (com $p_0 = 2$).
   - Escolha um inteiro $s < p-1$ tal que $\text{mdc}(p-1, s) = 1$.
   - Calcule $v_i = \sqrt[s]{p_i} \pmod p$ (pode ser necessário calcular em extensões de $\mathbb{Z}_p$) e faça $v = (v_0, v_1, \ldots v_r)$.
   - Defina a chave pública $K_e = (p, r, v)$.

- Fixe a chave privada $K_d = s$.

2. *Cifragem:*
   - Dada a mensagem $M = m_1 m_2 \cdots m_n$, onde cada m_i é um bit de M, calcule $C = \prod_{i=0}^{n} v_i^{m_i}$.
3. *Decifragem:*
   - Calcule $\sum_{i=0}^{r} \left[2^i \cdot (p_i - 1)^{-1} \pmod{p} \times (\text{mdc}(p_i, C^s) - 1)\right]$.

Note que, como $C^s = \prod_{i=0}^{n} v_i^{sm_i} = \prod_{i=0}^{n} p_i^{m_i}$, tem-se $\text{mdc}(p_i, C^s) = 1$ ou $\text{mdc}(p_i, C^s) = p_i$, dependendo de ocorrer $m_i = 0$ ou $m_i = 1$, respectivamente. Assim,

$$\sum_{i=0}^{r} \left[2^i \cdot (p_i - 1)^{-1} (\text{mod } p) \times (\text{mdc}(p_i, C^s) - 1)\right] = \sum_{m_i=1} 2^i = M.$$

4.4 Hipóteses RSA

Nesta seção estudaremos a chamada hipótese RSA (HRSA) e uma de suas variantes, a hipótese RSA forte (HRSAF). Em linhas gerais, a HRSA está relacionada com a segurança do criptossistema RSA e sustenta-se na intratabilidade do PF.

- **Hipótese RSA**
  É assumido que o problema a seguir (PRSA) é infactível:

  Dado um inteiro $n = pq$, que é produto de dois primos ímpares distintos aleatoriamente gerados, dado um inteiro $r > 1$ e dado um inteiro $z \in \mathbb{Z}_n^*$, também aleatoriamente gerado, encontrar $y \in \mathbb{Z}_n^*$ tal que $y^r = z$.

De fato, atualmente assume-se um pouco mais que isso:

- **Hipótese RSA forte**
  O problema a seguir (PRSAF) é infactível:

  Dado um inteiro $n = pq$, que é produto de dois primos ímpares distintos aleatoriamente gerados e dado um inteiro $z \in \mathbb{Z}_n^*$, também aleatoriamente gerado, encontrar um inteiro $r > 1$ e $y \in \mathbb{Z}_n^*$ tais que $y^r = z$.

O termo “forte” vem do fato que a HRSAF é um pouco mais flexível, permitindo a escolha de dois parâmetros (y e r). Entretanto, essa flexibilidade não torna o problema menos intratável que a HRSA [42].

O fato de o PRSA ser, aparentemente, tão intratável quanto o PF é a principal justificativa para assumirmos as HRSA's. O meio mais eficiente de se quebrar o criptossistema RSA que se conhece atualmente é fatorar o *módulo n*. Isso permite calcular $\phi(n)$ e assim determinar a chave privada d ao resolver a congruência $ed \equiv 1 \pmod{\phi(n)}$. Sabe-se que o problema de se determinar a chave privada d é equivalente a fatorar n, mas não existe prova de que essa é a única maneira de resolver o PRSA. De fato, existem fortes evidências de que o PF e o PRSA não sejam equivalentes [5].

No capítulo seguinte estudaremos a noção de grupos pseudo-livres que foi o ponto de partida para a formulação de uma nova hipótese RSA. Esta generaliza as anteriores, além de outras formulações sobre segurança de várias classes de criptossistemas e por essa razão foi denominada de *Hipótese RSA super-forte.*

CAPÍTULO 5

Grupos pseudo-livres e hipótese RSA super-forte

The theory of groups is a branch of mathematics in wich one does something to something and then compares the result with the result obtained from doing the same thing to something else, or something else to the same thing.
– JAMES R. NEWMAN (The world of mathematics)

—

O conceito de grupos pseudo-livres, criado recentemente, permite estabelecer uma nova versão de hipótese RSA. Esta nova hipótese não só generaliza as anteriores, mas contém em si muitas outras hipóteses criptográficas usadas atualmente.

A noção de grupo pseudo-livre foi inicialmente proposta por Susan Hohenberger [21], no ano de 2003, e refinada por R. Rivest [40]. Informalmente, a proposta é que, dado um grupo G, seja possível, sob certas condições, assegurar que a probabilidade de se encontrar soluções, nesse grupo, para certos tipos de equações (ou sistemas de equações) seja desprezível. É de particular interesse que, dado um esquema criptográfico sobre um grupo G, seja improvável que um adversário consiga resolver, em tempo razoável, as equações usadas para cifrar mensagens. Entretanto, durante algum tempo uma questão permaneceu sem resposta: *Existem grupos pseudo-livres?*

Estudaremos neste capítulo a resposta dada em 2005 por D. Micciancio em [32] onde é provado que, sob certas condições, o grupo $\mathbb{Z}_N^*$ é pseudo-livre. Este é o primeiro exemplo de tais grupos que se conhece.

5.1 Grupos RSA

Estaremos particularmente interessados no grupo $\mathbb{Z}_N^*$ das classes de intreiros relativamente primos com N. Em sua representação computacional, seus elementos são inteiros do conjunto $\{0,1,2,\ldots,N-1\}$.

Definição 5.1 (Grupos RSA). O grupo multiplicativo $\mathbb{Z}_N^*$, onde $N = PQ$ é produto de dois primos ímpares distintos, é chamado de *Grupo RSA.*

Por razões de segurança contra certos tipos de ataques conhecidos ao criptossistema RSA é comum a utilização de um tipo especial de primos:

Definição 5.2 (Primo seguro). Um primo P é dito *seguro* quando $p = (P-1)/2$ é primo.

Portanto, se p é um primo e $P = 2p+1$ também é primo, então este último é chamado de primo seguro. Infelizmente, ainda não se sabe se existem infinitos primos seguros.

5.2 Grupos livres

Seja A um conjunto não vazio e sejam a_i, com $i \in I$, seus elementos, onde I é um conjunto (não necessariamente finito) de índices. Diremos que A é o *alfabeto* ao qual pertencem as *letras* a_i. Os símbolos da forma $a_i^n = a_i a_i \dots a_i$, formados pela justaposição de letras iguais (com $n \in \mathbb{N}$), serão chamados de *sílabas* e qualquer cadeia finita $w = a_1^{n_1} a_2^{n_2} \dots a_k^{n_k}$, formada pela justaposição de sílabas, será uma *palavra*. Definiremos ainda a *palavra vazia* como sendo aquela que não contém sílabas e a denotaremos por 1.

Chamaremos de *contrações elementares* às seguintes operações com palavras:

- Substituição de uma ocorrência de $a_i^m a_i^n$ por a_i^{m+n}.
- Substituição de uma ocorrência de a_i^0 pela palavra vazia 1.

Dizemos que uma palavra w está na *forma reduzida* se não é possível fazer nenhuma contração elementar em w. Evidentemente, qualquer palavra pode ser expressa na forma reduzida através de um número finito de contrações elementares.

Sejam A um alfabeto e $F(A)$ o conjunto de todas as palavras sobre A que estão na forma reduzida. Existe um modo natural de dar a $F(A)$ uma estrutura de grupo:

- Dadas duas palavras w_1 e w_2, defina $w_1.w_2$ como sendo a forma reduzida da palavra obtida pela justaposição $w_1 w_2$ dessas duas palavras.
- A inversa da palavra $w = a_1^{n_1} a_2^{n_2} \dots a_k^{n_k}$ é definida por $w^{-1} = a_k^{-n_k} \dots a_2^{-n_2} a_1^{-n_1}$.

Definição 5.3. O grupo $F(A)$ é chamado de *grupo livre gerado por A.*

No que segue será necessário que tenhamos grupos abelianos (ou comutativos). Isto pode ser feito definindo-se uma certa relação de equivalência no conjunto $F(A)$ e tomando o quociente por essa relação. Mais precisamente, tomamos o comutador de $F(A)$, que é o grupo $\mathscr{C}$ gerado por $\{a_i a_j a_i^{-1} a_j^{-1} : a_i, a_j \in F(A)\}$ e definimos em $F(A)$ a relação de equivalência $a \sim b \Leftrightarrow ab^{-1} \in \mathscr{C}$. Finalmente, tomamos o quociente $F(A)/\sim$. Obtemos um grupo livre abeliano que denotaremos por $\mathscr{F}(A)$.

Exemplo 5.1. O conjunto dos polinômios na variável X e coeficientes em $\mathbb{Z}$, denotado por $\mathbb{Z}[X]$, com a operação de adição de polinômios é um grupo livre gerado pelo conjunto $A = \{1, X, X^2, X^3, \ldots\}$.

A característica que nos interessa em um grupo livre é o fato de que não existem relações não triviais entre seus geradores (ver [26]).

Exemplo 5.2. No exemplo anterior, como o grupo é aditivo, não existem relações não triviais entre os geradores $1, X, X^2, X^3, \ldots$

5.3 Grupos pseudo-livres

A noção de grupo pseudo-livre foi inicialmente proposta por Susan Hohenberger em sua dissertação [21] em 2003 e melhorada por Rivest em [40]. Informalmente, a proposta é que, dado um grupo G, seja possível, sob certas condições, assegurar que a probabilidade de se encontrar soluções, nesse grupo, para certos tipos de equações (ou sistemas de equações) seja desprezível. Dessa forma, a característica desejável em um grupo pseudo-livre é que este tenha, pelo menos probabilisticamente, um comportamento semelhante a um grupo livre. É de particular interesse que, dado um esquema criptográfico sobre G, seja improvável que um adversário consiga resolver em tempo razoável as equações usadas para cifrar mensagens.

Definição 5.4. Seja $(G, \circ, ()^{-1}, 1)$ um grupo com operação $\circ$, inversos dados pela função $()^{-1}$ e identidade 1. Um *grupo computacional* associado a G, denotado por $< G >$, é uma aplicação $< . >: G \to \{0,1\}^*$ tal que as seguintes operações podem ser executadas em tempo polinomial:

- Testar pertinência a $< G >$, ou seja, determinar se uma dada string x é uma representação válida de um elemento de G.
- Dados $< x >$ e $< y >$, calcular $< x \circ y >$.
- Dado $< x >$, calcular $< x^{-1} >$.
- Calcular a representação da identidade $< 1 >$.
- Sortear um elemento $< g > \in < G >$ (mas não necessariamente com distribuição uniforme de probablidade).

Podemos agora definir grupo pseudo-livre:

Definição 5.5. Sejam

• $\mathscr{N} = \mathscr{N}(k)$ um conjunto de produtos de dois primos seguros, onde k é um parâmetro de segurança (relativo ao tamanho dos primos),
• $\mathscr{G} = \{G_N\}_{N \in \mathscr{N}}$ uma família de grupos computacionais com índices em $\mathscr{N}$,
• S um conjunto de tamanho polinomial $|S| = p(k)$,
• $\mathscr{A}$ um algoritmo de tempo polinomial probabilístico.

Dizemos que a família $\mathscr{G}$ é pseudo-livre se vale o seguinte:

> Seja $N \in \mathscr{N}$ um índice aleatoriamente escolhido e seja $\alpha : S \to G_N$ uma função definindo $|S|$ elementos aleatoriamente escolhidos, de acordo com o procedimento de amostragem adotado para o grupo G_N. Então a probabilidade de que $\mathscr{A}(N, \alpha) = (E, \xi)$ produza uma equação E sobre o grupo abeliano livre gerado por S e com variáveis X, juntamente com uma solução $\xi : X \to G_N$ para E_α (equação obtida de E, fazendo $a \mapsto \alpha(a)$) sobre G_N, é desprezível.

Apesar de terem sido idealizados teoricamente, durante algum tempo não era conhecido se de fato existiam grupos ou famílias de grupos pseudo-livres.

5.4 O grupo RSA é pseudo-livre

De agora em diante denotaremos por $\mathscr{N}$ o conjunto dos produtos de dois primos seguros e assumiremos alguma distribuição de probabilidade sobre $\mathscr{N}$. Assumiremos ainda, nesta seção, a hipótese RSA forte e também que o procedimento de amostragem sobre $\mathbb{Z}_N^*$, com $N \in \mathscr{N}$, é feito escolhendo-se uniformemente resíduos quadráticos módulo N, isto é, elementos do grupo cíclico $RQ_N = \{a^2 \pmod N : 1 \leq a \leq N-1 \text{ e } \text{mdc}(a, N) = 1\} \subset \mathbb{Z}_N^*$. O principal resultado deste capítulo é:

Teorema 5.1 (Micciancio). *Denote por $\mathscr{N}$ o conjunto dos produtos de dois primos seguros escolhidos de acordo com uma distribuição de probabilidade de modo que o PRSAF sobre $N \in \mathscr{N}$ é difícil. Então a família de grupos computacionais $\{\mathbb{Z}_N^* : N \in \mathscr{N}\}$, com amostragem uniforme sobre RQ_N, é pseudo-livre.*

A prova desse teorema será feita por redução ao absurdo e será dividida em três partes. Inicialmente veremos alguns resultados que serão utilizados durante a demonstração. Nas duas últimas partes assumimos o PRSAF e será mostrado como a negação da tese pode ser usada para resolver o PRSAF, o que nos levará a uma contradição.

5.4.1 Resultados preliminares

Começaremos destacando a importância de que na escolha de elementos $\gamma \in RQ_N$, os escolhidos sejam geradores do grupo RQ_N, pois do contrário veremos que o PRSAF pode ser facilmente resolvido para a instância (N, γ). A condição para que γ seja um gerador desse grupo é dada pelo resultado a seguir.

Lema 5.1. *Seja $N = PQ$ produto de dois primos seguros e seja $\gamma \in RQ_N$. Então γ é gerador de RQ_N se, e somente se, $mdc(\gamma - 1, N) = 1$.*

Demonstração. Faça $P = 2p+1$ e $Q = 2q+1$, com p e q primos distintos. Pelo teorema chinês dos restos, a aplicação $\psi(\gamma) = (\gamma_p, \gamma_q) = (\gamma \pmod P, \gamma \pmod Q)$ define um isomorfismo entre RQ_N e $RQ_P \times RQ_Q$. Sejam $o(\gamma_p)$ e $o(\gamma_q)$ as ordens de γ_p e γ_q em RQ_P e RQ_Q, respectivamente. Temos

$$\gamma_p \in RQ_P \Rightarrow o(\gamma_p) \mid o(RQ_P) = p \Rightarrow o(\gamma_p) \in \{1, p\}.$$

Analogamente, $o(\gamma_q) \in \{1, q\}$. Assim,

$$o(\gamma) = o(\gamma_p)o(\gamma_q) \in \{1, p, q, pq\}.$$

Um elemento $\gamma \in RQ_N$ é gerador de RQ_N se, e somente se, $o(\gamma) = |RQ_N| = |RQ_P|.|RQ_Q| = pq$. Seja $d = mdc(\gamma - 1, N)$. Devemos provar que $o(\gamma) = pq$ se, e somente se, $d = 1$. Como $d \mid N$, temos $d \in \{1, p, q, pq\}$. Vamos analisar cada caso:

(1) $d = P$
Como $d = mdc(\gamma - 1, N)$ e $N = PQ$, então $P \mid \gamma - 1$ e teremos $\gamma_p = \gamma \pmod P = 1$. Assim, $o(\gamma) = o(\gamma_p)o(\gamma_q) = o(\gamma_q) \in \{1, q\}$ e γ não pode ser gerador de RQ_N.

(2) $d = Q$
Este caso é análogo ao anterior.

(3) $d = PQ$
Nesse caso, $d = N$ e, como $d \mid \gamma - 1$, teremos $\gamma = 1$ em $\mathbb{Z}_N^*$. Assim γ não pode ser gerador de RQ_N.

Portanto, $d = mdc(\gamma - 1, N) = 1$ é condição necessária para que γ seja gerador de RQ_N.

Por outro lado, se γ não é gerador de RQ_N, teremos $o(\gamma) = o(\gamma_p)o(\gamma_q) \neq pq$. Assim, $o(\gamma_p) = 1$ ou $o(\gamma_q) = 1$. Caso ocorra $o(\gamma_p) = 1$, então $\gamma \pmod P = \gamma_p = 1$. Portanto, $P \mid \gamma - 1$ e, como $P \mid N$, teremos $mdc(\gamma - 1, N) \neq 1$. O caso em que $o(\gamma_q) = 1$ é análogo.

□

Agora veremos como se pode resolver o PRSAF quando γ não é gerador de RQ_N. Dados $N = PQ$ e $\gamma \in RQ_N$, calculamos $d = mdc(\gamma - 1, N)$. Como $d \neq 1$, temos três casos a considerar:

(1) $d = N = PQ$
Teremos $d \mid \gamma - 1$ e assim $\gamma = 1 \pmod N$. Então $\xi = 1$ e $e = 3$, por exemplo, são tais que $\xi^e = \gamma$.

(2) $d = P$
Calculamos $\phi(N) = (P-1)(Q-1) = (d-1)(N/d-1)$. Fazendo $\xi = \gamma$ e $e = \phi(N)+1$, teremos $\xi^e = \gamma^{\phi(N)+1} = \gamma^{\phi(N)}.\gamma = \gamma \pmod N$.

(3) $d = Q$
Este caso é análogo ao anterior.

Um outro resultado importante para a demonstração do Teorema 5.1 é dado a seguir:

Lema 5.2. *Sejam G um grupo cíclico finito e γ um dos seus geradores. Seja $Y(v) = \gamma^v$ a distribuição obtida escolhendo-se $v \in \{0,1,\ldots,B-1\}$ uniformemente ao acaso. Então a distância estatística entre Y e a distribuição uniforme sobre G é no máximo $|G|/2B$.*

Demonstração. Seja $v \in \{0,1,\ldots,B-1\}$. Como γ é gerador de G, temos $G = \{1,\gamma,\ldots,\gamma^{|G|-1}\}$ e $\gamma^v = \gamma^i$ se, e somente se, $v = i \pmod{|G|}$. Assim,

$$P[\gamma^v = \gamma^i] = P[v = i \pmod{|G|}] = \frac{\lceil (B-i)/|G| \rceil}{B}$$

pois, dado $i \in \{0,1,\ldots,|G|-1\}$, a quantidade de elementos $v \in \{0,1,\ldots,B-1\}$ tais que $v = i \pmod{|G|}$ é $\lceil (B-i)/|G| \rceil$. Nesse caso, dado $i \in \{0,1,\ldots,|G|-1\}$, temos

$$\begin{aligned}\left| P[\gamma^v = \gamma^i] - \frac{1}{|G|} \right| &= \left| \frac{1}{B} \left\lceil \frac{B-i}{|G|} \right\rceil - \frac{1}{|G|} \right| \\ &= \frac{1}{B} \left| \left\lceil \frac{B-i}{|G|} \right\rceil - \frac{B}{|G|} \right| \\ &\leq \frac{1}{B}\end{aligned}$$

pois, como $0 \leq \frac{B-i}{|G|} \leq \frac{B}{|G|}$, temos $0 \leq \left\lceil \frac{B-i}{|G|} \right\rceil \leq \frac{B}{|G|} + 1$ e assim $\left| \left\lceil \frac{B-i}{|G|} \right\rceil - \frac{B}{|G|} \right| \leq 1$.

Finalmente, a distância estatística entre Y e a distribuição uniforme U sobre G é

$$\begin{aligned}\Delta(Y,U) &= \frac{1}{2}\sum_{i=0}^{|G|-1}\left|P[Y=\gamma^i]-P[U=\gamma^i]\right| \\ &= \frac{1}{2}\sum_{i=0}^{|G|-1}\left|P[\gamma^y=\gamma^i]-\frac{1}{|G|}\right| \\ &\leq \frac{1}{2}\sum_{i=0}^{|G|-1}\frac{1}{B} \\ &= \frac{|G|}{2B}.\end{aligned}$$

□

Assumir que a família $\{\mathbb{Z}_N^*\}_{N\in\mathscr{N}}$ não é pseudo-livre, equivale a assumir a existência de um algoritmo de tempo polinomial probabilístico $\mathscr{A}$ que, sob a entrada $N \in \mathscr{N}(k)$ e elementos $\alpha : A \to RQ_N$ (A de tamanho polinomial $|A| = p(k)$), escolhidos aleatoriamente, produz uma equação $E : w_1 = w_2$, com conjunto de variáveis X, que não pode ser satisfeita sobre o grupo livre $\mathscr{F}(A)$, juntamente com uma solução $\xi : X \to \mathbb{Z}_N^*$ para E_α sobre o grupo $\mathbb{Z}_N^*$.

Dado $a \in A$, escolha $v_a \in \{0,\ldots,B-1\}$ (onde $B = N|A|\lambda$) com distribuição de probabilidade uniforme, onde $\lambda = \lambda(k)$ é uma função super-polinomial de k, e faça $\alpha(a) = \gamma^{v_a}$. Pelo Lema 5.2, a distância estatística entre a distribuição α e a uniforme sobre RQ_N é no máximo $|RQ_N|/2B = |RQ_N|/2N|A|\lambda \leq 1/|A|\lambda$. Como os valores de $\alpha(a)$ são independentemente escolhidos, segue que

$$\begin{aligned}\Delta(\alpha,U) &= \frac{1}{2}\sum_{a\in A}|P[x=\alpha(a)]-P[x=a]| \\ &\leq \frac{1}{2}\sum_{a\in A}\frac{1}{|A|\lambda} \\ &= \frac{1}{2\lambda}.\end{aligned}$$

5.4.2 Prova do teorema principal

Como dito anteriormente, a prova do Teorema 5.1 será divida em algumas etapas. Vimos alguns resultados que serão utilizados durante a demonstração. De agora em diante vamos assumir que a família $\mathbb{Z}_{\mathbb{N}}{}^*$, com $N \in \mathscr{N}$ não é pseudo-livre. Veremos então como essa hipótese leva à solução do PRSAF sobre RQ_N. A redução será feita em dois passos:

5.4.2.1 Primeiro passo da redução

Transformar $E : w_1 = w_2$ em uma nova equação E' com apenas uma variável e que também não possa ser satisfeita sobre o grupo $\mathscr{F}(A)$ e transformar a solução ξ de E_α em uma solução ξ' para E'_α.

Lema 5.3. *Dado um grupo computacional G, existe um algoritmo de tempo polinomial que, tomando como entrada uma equação E (com conjunto de constantes A e de variáveis X) e uma atribuição de valores $\xi : X \to G$, produz uma equação de uma variável E'_α e um elemento $\xi' \in G$ tal que*

(i) se E não é satisfatível sobre o grupo livre $\mathscr{F}(A)$, então o mesmo ocorre com E'.

(ii) para qualquer atribuição $\alpha : A \to G$, se ξ é solução de E_α, então ξ' é uma solução de E'_α.

Demonstração. Fixe G, E e ξ como no enunciado. Vamos assumir, sem perda de generalidade, que E é dada na forma canônica $E : \prod_{x\in X} x^{e_x} = \prod_{a\in A} a^{d_a}$. Utilizando o algoritmo euclideano estendido podemos calcular $e = mdc(e_x : x \in X)$ e inteiros b_x tais que $\sum_{x\in X} b_x e_x = e$. Defina a saída do algoritmo como sendo

$$E' : x^e = \prod_{a\in A} a^{d_a}, \quad \xi' = \prod_{x\in X} \xi(x) x^{e_x/e}.$$

Devemos provar que E' e ξ' satisfazem às propriedades (i) e (ii).

(i) Assuma que E' tem uma solução $\delta' \in \mathscr{F}(A)$. Temos $(\delta')^e = \prod_{a\in A} a^{d_a}$. Para $x \in X$ defina $\delta(x) = (\delta')^{b_x}$. Assim,

$$\prod_{x\in X} \delta(x)^{e_x} = \prod_{x\in X} (\delta')^{b_x e_x} = (\delta')^{\sum_{x\in X} b_x e_x} = (\delta')^e = \prod_{a\in A} a^{d_a},$$

ou seja, E seria satisfatível sobre $\mathscr{F}(A)$, contrariando a hipótese.

(ii) Fixe $\alpha : A \to G$ e suponha que $\xi : X \to G$ é uma solução para a equação E_α. Temos $\prod_{x\in X} \xi(x)^{e_x} = \prod_{a\in A} \alpha(a)^{d_a}$ em G. Então

$$(\xi')^e = \left(\prod_{x\in X} \xi(x)^{e_x/e}\right)^e = \prod_{x\in X} \xi(x)^{e_x} = \prod_{a\in A} \alpha(a)^{d_a}.$$

Portanto, ξ' é uma solução de E'_α sobre G.

□

5.4.2.2 Segundo passo da redução

Usar a equação E' e a solução ξ' de E'_α para resolver o PRSAF sobre RQ (PRSAF-RQ).

A equação $E' : x^e = \prod_{a\in A} a^{d_a}$ terá solução sobre o grupo livre $\mathscr{F}(A)$ se, e somente se, e divide $d = mdc(d_a : a \in A)$. Como E não deve ser satisfeita sobre $\mathscr{F}(A)$ devemos assumir, devido ao Lema 5.3, que $e \nmid d$. Sendo $\alpha(a) = \gamma^{\nu_a}$, temos

$$(\xi')^e = \prod_{a\in A} \alpha(a)^{d_a} = \prod_{a\in A} \gamma^{\nu_a d_a} = \gamma^{\sum_{a\in A} \nu_a d_a}.$$

Caso seja necessário, trocaremos ξ' por $(\xi')^{-1}$ e γ por γ^{-1} (isto equivale a trocar o sinal de e e de todos os d_a, respectivamente), de modo que tenhamos $e \geq 0$ e $d = \sum_{a\in A} \nu_a d_a \geq 0$.

Fixaremos nossa atenção em e e em $mdc(e, pq)$ e veremos como resolver o PRSAF sobre RQ_N, assumindo a HRSAF e que $\{\mathbb{Z}_{\mathbb{N}}{}^*\}_{N\in\mathscr{N}}$ não é pseudo-livre. Precisaremos de mais dois lemas:

Lema 5.4. *Dados $\alpha : A \to \mathbb{Z}_N$, $e = 0$ e $\{d_a : a \in A\}$ tal que $e \nmid mdc(d_a : a \in A)$, a probabilidade condicional de que ocorra $d = 0$ é de pelo menos 1/2.*

Lema 5.5. *Dados α, $mdc(e, pq) = 1$ e $\{d_a : a \in A\}$ tais que $e \nmid mdc(d_a : a \in A)$, a probabilidade condicional de que e divida d é no máximo 2/3.*

Podemos agora completar a redução:

Lema 5.6. *Se $\{\mathbb{Z}_N^*\}_{N\in\mathscr{N}}$ não é pseudo-livre, então o PRSAF sobre RQ_N pode ser resolvido.*

Demonstração. Assuma que $\{\mathbb{Z}_N\}_{N\in\mathscr{N}}$ não é pseudo-livre. Dado $N \in \mathscr{N}$, existe um algoritmo de tempo polinomial $\mathscr{A}$ que produz uma equação $E : \prod_{x\in X} x^{e_x} = \prod_{a\in A} a^{d_a}$ insatisfatível sobre $\mathscr{F}(A)$ (onde A é um conjunto de tamanho polinomial $|A| = p(k)$) e uma solução $\xi : X \to \mathbb{Z}_N^*$ para E_α. Pelo Lema 5.3 obtemos, a partir de E e de ξ, uma equação $E' : x^e = \prod_{a\in A} a^{d_a}$, em uma variável, insatisfatível sobre $\mathscr{F}(A)$, juntamente com uma solução $\xi' \in \mathbb{Z}_N^*$ de E'_α.

Dado $a \in A$, escolhemos $\nu_a \in \{0, 1, \cdots, N|A|\lambda - 1\}$ uniformemente ao acaso e fazemos $\alpha(a) = \gamma^{\nu_a}$, onde γ é um gerador de RQ_N. Vimos antereiormente que podemos assumir que $e \geq 0$ e $d = \sum_{a\in A} \nu_a d_a \geq 0$. Vamos analisar os diversos casos que podem ocorrer, dependendo dos valores de e e $mdc(e, pq)$.

Caso 1: $e \neq 0$ e $mdc(e, pq) = pq$.
Temos $pq \mid e$ e, como $o(\gamma) = pq$, segue que $\gamma^{e+1} = \gamma^e.\gamma = \gamma \pmod N$. Assim, $(\gamma, e+1)$ é uma solução para a instância (N, γ) do PRSAF-RQ.

Caso 2: $e \neq 0$ e $mdc(e, pq) \in \{p, q\}$.
Temos $o(\gamma^e) = pq/mdc(e, pq) \in \{p, q\}$. Assim, γ^e não é gerador de RQ_N e, pelo Lema 5.1, $mdc(\gamma^e - 1, N) \neq 1$. Além disso, como $o(\gamma^e) \neq 1$, temos $\gamma^e \neq 1 \pmod N$ e assim $N \nmid \gamma^e - 1$, o que nos dá $mdc(\gamma^e - 1, N) \neq N$. Como $N = PQ$, segue que $g = mdc(\gamma^e - 1, N) \in \{P, Q\}$. Com isso podemos calcular

$$\phi(N) = (P-1)(Q-1) = (g-1)(N/g-1).$$

Então $\gamma^{\phi(N)+1} = \gamma^{\phi(N)}.\gamma = \gamma \pmod N$, o que resolve o PRSAF-RQ para a instância (N, γ).

Caso 3: $e = 0$.
Pelo Lema 4, $d = 0$ com probabilidade pelo menos 1/2. Como $(\xi')^e = \gamma^{\sum_{a \in A} v_a d_a} = \gamma^d$, temos

$$\gamma^{d+1} = \gamma^d.\gamma = (\xi')^0.\gamma = \gamma$$

com probabilidade pelo menos 1/2. Assim, teremos uma probabilidade não desprezível de resolver o PRSAF-RQ para a instância (N, γ).

Caso 4: $mdc(e, pq) = 1$.
Pelo Lema 5, existe uma probabilidade de no máximo 2/3 de que e divida d. Assim, a probabilidade de ocorrer $e \nmid d$ é de pelo menos 1/3. Vamos construir recursivamente um algoritmo que, tomando (γ, ξ, e, d) como entrada, onde

$$\xi^e = \gamma^d, \quad e \neq 0, \quad e \nmid d,$$

produz uma saída (θ, c, h) tal que

$$\theta^{c^{h+1}} = \gamma^{c^h}, \quad |c| \geq 2, \quad c^{h+1} \mid e, \quad c^h \mid d$$

e h é o menor possível.

Observamos que, nessas condições, $\delta = (\theta^c/\gamma)^{c^{h-1}}$ é um resíduo quadrático módulo N:

• Para c par faça $c = 2r\,(r \in \mathbb{Z})$. Temos $\theta^c = (\theta^r)^2 \in RQ_N$. Como $\gamma \in RQ_N$, segue que $(\theta^c/\gamma)^{c^{h-1}} \in RQ_N$.

• Para c ímpar faça $c^{h+1} = 2r + 1\,(r \in \mathbb{Z})$. Obtemos $\theta^{c^{h+1}} = (\theta^r)^2\,\theta$. Assim, $\theta \in RQ_N$ se, e somente se, $\theta^{c^{h+1}} \in RQ_N$. Como $\theta^{c^{h+1}} = \gamma^{c^h} \in RQ_N$, temos $\theta \in RQ_N$. Portanto, $\delta \in RQ_N$.

Caso tenhamos $\theta^c = \gamma$, então (θ, c) é uma solução para a instância (N, γ) do PRSAF-RQ. Dessa forma, vamos assumir que $\theta^c \neq \gamma$. Como $(\theta^c/\gamma)^{c^h} = \theta^{c^{h+1}}/\gamma^{c^h} = 1$ e h é o menor possível, temos $\delta = (\theta^c/\gamma)^{c^h-1} \neq 1$. Vamos mostrar como resolver o PRSAF-RQ para os possíveis valores de δ.

• δ é gerador de RQ_N:
Como $\delta^c = (\theta^c/\gamma)^{c^h} = 1$, segue que $o(\delta) \mid c$. Sendo γ gerador de RQ_N, temos $o(\gamma) = o(\delta)$. Assim, $\gamma^c = 1$ e $\gamma^{c+1} = \gamma$. Portanto $(\gamma, c+1)$ resolve o PRSAF-RQ.

• δ não é gerador de RQ_N:
Pelo Lema 1, $mdc(\delta - 1, N) \neq 1$. Como $\delta \neq 1$, temos também $mdc(\delta - 1, N) \neq N$. Assim, $g = mdc(\delta - 1, N) \in \{P, Q\}$. Podemos calcular $\phi(N) = (P-1)(Q-1) = (g-1)(N/g-1)$. Como antes, $(\gamma, \phi(N)+1)$ resolve o PRSAF-RQ.

Para concluir a prova de que se $\{\mathbb{Z}_N\}_{N \in \mathscr{N}}$ não é pseudo-livre, resolvemos o PRSAF-RQ resta construir o algoritmo que produz (θ, c, h).

Descrição do algoritmo $\mathbb{A}(\gamma, \xi, e, d)$:

• Se $d \nmid e$, calcule $d_1 = mdc(e, d)$ e inteiros r e s com $d_1 = er + ds$. Chame $\mathbb{A}(\gamma, \xi^s \gamma^r, e, d_1)$.
• Caso contrário (isto é, $d \mid e$), calcule $c = e/d$.
  – Se $d = c^h$ é potência de c, retorne (ξ, c, h)
  – Caso contrário, seja h o maior expoente tal que $c^h \mid d$. Chame $\mathbb{A}(\gamma, \xi^{d/c^h}, c^{h+1}, d)$.

Devemos provar que o algoritmo $\mathbb{A}$ funciona corretamente e que d ou e decresce a cada iteração. Lembramos que a entrada (γ, ξ, e, d) deve satisfazer às condições

$$\xi^e = \gamma^d, \quad e \neq 0, \quad e \nmid d.$$

• **Caso** $d \nmid e$:
Fazendo $\xi_1 = \xi^s \gamma^r$, $e_1 = e$ e $d_1 = mdc(e, d)$, temos

$$\xi_1^{e_1} = (\xi^s \gamma^r)^e = (\xi^e)^s \gamma^{re} = \gamma^{er+ds} = \gamma^{d_1}$$

Além disso, $e_1 = e \neq 0$ e, como $e \nmid d$, temos $e_1 \nmid d_1$. Se $d = d_1$, $d = d_1 = mdc(e, d) \mid e$, o que não pode ocorrer. Assim $d_1 < d$, o que garante que o algoritmo pára.

• **Caso** $d \mid e$ e $d = c^h$:
Inicialmente devemos verificar se o quociente $c = e/d$ está definido, o que ocorre se $d \neq 0$. Caso seja $d = 0$, teremos $e \mid d$, o que é proibido por hipótese. Temos ainda $|c| > 1$, pois $e \neq 0$ e $e \nmid d$. Nesse caso o algoritmo termina com saída $\theta (= \xi), c, h$. Além disso, a saída satisfaz às condições estabelecidas, pois

$$\theta^{c^{h+1}} = (\theta^{c^h})^c = \xi^{dc} = \xi^e = \gamma^d = \gamma^{c^h}.$$

• **Caso** $d \mid e$ e d não é potência de c:
Faça $c = e/d$. Novamente, temos $d \neq 0$ e, como $|c| > 1$, podemos calcular $h = max\{j : c^j \mid d\}$.

Assim d/c^h é inteiro. O algoritmo será chamado recursivamente com entrada $(\gamma, \xi_1, e_1, d_1)$, onde $\xi_1 = \xi^{d/c^h}$, $e_1 = c^{h+1}$ e $d_1 = d$. Assim,

$$\xi_1^{e_1} = \xi^{dc^{h+1}/c^h} = \xi^{dc} = \xi^e = \gamma^d = \gamma^{d_1}.$$

Como h é o maior expoente tal que $c^h \mid d$, temos $c^{h+1} \nmid d$. Assim, $e_1 \nmid d_1$. Finalmente, se $e_1 = e$, teríamos $c^h = e_1/c = e/c = d$, ou seja, d seria potência de c (o que não pode ocorrer, por hipótese). Portanto, $e_1 < e$ e o algoritmo deve parar.

Isto conclui a prova do Teorema 5.1. Assim, a família $\{\mathbb{Z}_N^*\}_{N \in \mathcal{N}}$ é pseudo-livre. □

Vimos neste capítulo que a família de grupos $\{\mathbb{Z}_N^*\}_{N \in \mathcal{N}}$, onde N é produto de primos seguros, é pseudo livre. Dessa forma, fica evidenciada a importância dessa classe de números primos para a criptografia RSA.

No capítulo seguinte estudaremos em mais detalhes os primos seguros. Veremos algumas de suas principais propriedades, além de alguns algoritmos para geração de tais números.

CAPÍTULO 6

Primos seguros

A vida é ou uma aventura audaciosa, ou não é nada.
A segurança é geralmente uma superstição.
Ela não existe na natureza.
– HELEN KELLER

—

6.1 Introdução

Vimos no capítulo anterior que utilizar primos seguros (ou *primos fortes*) é de fundamental importância para obter um esquema criptográfico RSA seguro. Existem algoritmos de fatoração, como o *Pollard* $p-1$, que pressupõem que o número a ser fatorado possui um fator primo p tal que os fatores primos de $p-1$ são pequenos. Sendo p ímpar, 2 é um fator de $p-1$. Podemos adotar o "caso extremo" que é tomar $(p-1)/2$ primo.

Em matemática esses números são conhecidos como *primos de Sophie Germain*. Sua importância para a matemática deve-se, em parte, à sua relação com o *Último Teorema de Fermat (UTF)*. A afirmação que $x^n+y^n=z^n$, com $n \nmid xyz$, não tem solução em inteiros positivos é conhecida como o *primeiro caso do UTF*. Aproximadamente no ano de 1825 Marie-Sophie Germain provou o seguinte resultado:

Teorema 6.1 (Sophie Germain). *Seja q um primo ímpar. Suponha que existe um primo p tal que:*

i) $x^q+y^q+z^q \equiv 0 \pmod{p} \Rightarrow x \equiv 0 \pmod{p}$ *ou* $y \equiv 0 \pmod{p}$ *ou* $z \equiv 0 \pmod{p}$,

ii) $x^q \equiv q \pmod{p}$ *não tem solução.*

Então $x^q+y^q=z^q$ (com $q \nmid xyz$) não tem solução inteira.

É possível provar que as hipóteses desse teorema são satisfeitas se $q=2p+1$. Assim, os primos da forma $2p+1$, com p primo, ficaram conhecidos como primos de Sophie Germain.

Antes desse teorema, foram provados apenas casos isolados do UTF. Sua importância reside no fato deste ter sido o primeiro resultado de caráter geral sobre o UTF. A prova do UTF por A.

Wiles em 1995 fez com que diminuisse um pouco o interesse (em matemática) por esses primos.

Não se sabe se existem infinitos primos seguros. Entretanto, argumentos heurísticos e evidências numéricas indicam a existência de infinitos primos dessa forma [54]. Conjectura-se que o número de primos seguros menores que um dado inteiro $N > 2$ é assintoticamente igual a

$$2C \int_2^N \frac{dx}{\ln x \ln(2x+1)} \simeq \frac{2CN}{(\ln N)^2},$$

onde $C = \prod_{q>2} \frac{q(q-2)}{(q-1)^2} \simeq 0.6601618158$ e o produto é tomado sobre todos os primos $q > 2$.

A estimativa dada por essa integral é surpreendentemente precisa como pode ser visto na tabela a seguir:

N	Valor correto (V_C)	Valor estimado (V_E)	Erro $= 100\% \times \lvert V_C - V_E \rvert / V_C$
10^3	37	39	5,41
10^4	190	195	2,63
10^5	1.171	1.166	0,43
10^6	7.746	7.811	0,84
10^7	56.032	56.128	0,17
10^8	423.140	423.295	0,04
10^9	3.308.859	3.307.888	0,03
10^{10}	26.569.515	26.568.824	0,003
10^{11}	218.116.524	218.116.102	0,0002

Tabela 6.1 Primos seguros entre 2 e 10^N

6.2 Algoritmos para a geração de primos seguros

Nesta seção assumiremos que é dado um algoritmo $\mathbb{A}$ que fornece primos. Sua entrada será o número k de bits do primo a ser gerado e sua complexidade será denotada por $f(k)$. Estudaremos alguns algoritmos que, utilizando $\mathbb{A}$, produzem primos seguros. Iniciaremos com um algoritmo ingênuo e faremos sucessivos refinamentos até obter um algoritmo baseado na Lei de reciprocidade quadrática e em um teorema de Euler e Lagrange sobre primos seguros e números de Mersenne. De início observamos que primos seguros são de uma forma especial:

Teorema 6.2. *Se $p > 3$ e $q = 2p+1$ são ambos primos, então $p \equiv 2 \pmod 3$.*

Demonstração. Como $p > 3$, temos $p \equiv 1 \pmod 3$ ou $p \equiv 2 \pmod 3$. Se $p \equiv 1 \pmod 3$, então $q = 2p + 1 \equiv 0 \pmod 3$, ou seja, $q = 3$. Isto não pode ocorrer, pois $q > p > 3$. Assim, tem-se $p \equiv 2 \pmod 3$. □

6.2.1 Algoritmo ingênuo

Vamos assumir que $\mathbb{A}$ produz apenas primos $p \equiv 2 \pmod 3$. O algoritmo mais óbvio para geração de primos seguros é dado por:

Algoritmo *Ingenuo*
Entrada k: inteiro positivo
Saída p: primo seguro com k bits
1 **faça**
2 $\quad p \leftarrow \mathbb{A}(k)$
3 **enquanto** $(p-1)/2$ é composto
4 **devolva** p
fim

Algoritmo 1 Geração de primos seguros: Algoritmo ingênuo

Sabemos, pelo teorema dos números primos, que a quantidade de primos que não excedem N é aproximadamente $N/\ln N$. Então a probabilidade de que um inteiro de k bits, escolhido aleatoriamente, seja primo é dada por

$$p_1(k) = \frac{1}{2^{k-2}}\left(\frac{2^k}{k\ln 2} - \frac{2^{k-1}}{(k-1)\ln 2}\right) = \frac{4}{k\ln 2} - \frac{2}{(k-1)\ln 2} \simeq \frac{2}{k\ln 2},$$

pois o intervalo contendo esses primos tem comprimento $2^{k-1} - 2^{k-2} = 2^{k-2}$. Isto significa que, no pior caso, encontramos um primo a cada $\frac{1}{2}k\ln 2$ execuções de $\mathbb{A}$. Como $(p-1)/2$ tem algo em torno de $k-1$ bits, a complexidade do algoritmo 1 pode então ser avaliada em aproximadamente

$$C_1(k) = \frac{f(k)}{p_1(k-1)} \simeq \frac{f(k)k\ln 2}{2}$$

6.2.2 Reduzindo a complexidade por um fator 2

O algoritmo da seção anterior verifica se um dado primo p é seguro. Uma simples mudança, proposta por D. Naccache [34], verifica a cada execução de $\mathbb{A}$ se p e $2p+1$ são seguros:

Algoritmo *Dupla_velocidade*
Entrada k: inteiro positivo
Saída p: primo seguro com k ou $k+1$ bits
1 **faça**
2 $\quad p \leftarrow \mathbb{A}(k)$
3 **enquanto** $(p-1)/2$ e $2p+1$ são compostos
4 **se** $(p-1)/2$ é primo **então**
5 $\quad$ **devolva** p
6 **senão**
7 $\quad$ **devolva** 2p+1
8 **fim se**
fim

Algoritmo 2 Geração de primos seguros: Dupla_velocidade

A probabilidade de que $(p-1)/2$ ou $2p+1$ seja primo é

$$p_2(k) = 1-(1-p_1(k-1))(1-p_1(k+1)) \simeq 2p_1(k),$$

desde que o algoritmo $\mathbb{A}$ gere primos com l e $l+1$ bits com aproximadamente a mesma probabilidade. Agora podemos estimar a complexidade do algoritmo 2:

$$C_2(k) = \frac{f(k)}{2p_1(k)} \simeq \frac{f(k)k\ln 2}{4} = \frac{1}{2}C_1(k).$$

6.2.3 Um crivo combinado

Vimos na seção 6.2.1 que os candidatos a primos seguros p devem satisfazer à condição $p \equiv 2 \pmod 3$. Caso contrário, $p \equiv 1 \pmod 3$ e, sendo p ímpar, $(p-1)/2$ seria múltiplo de 3. M. J. Wiener mostrou como se pode acelerar ainda mais o processo de obtenção de primos seguros [49]:

Sejam p e r primos ímpares, com $r < p$. Se $p \equiv 1 \pmod r$, então $(p-1)/2 \equiv 0 \pmod r$ e, caso ocorra $p \equiv (r-1)/2 \pmod r$, então $q = 2p+1 \equiv 0 \pmod r$. Modificamos o algoritmo $\mathbb{A}$ para que elimine os candidatos que são congruentes a 1 ou $(r-1)/2$ módulo r para todos os primos $r \leq B$, onde B é uma constante pré-fixada.

Finalmente, para os primos sobreviventes, usamos o algoritmo 2 para dobrar a chance de sucesso.

6.2.4 Um teorema de Euler e Lagrange

O teorema a seguir, enunciado por Euler em 1750 e provado por Lagrange em 1775, será utilizado para aumentar a eficiência na geração de primos seguros:

Teorema 6.3 (Euler - Lagrange). *Seja $p \equiv 3 \pmod 4$ um primo. Então $q = 2p+1$ é primo se, e somente se, q divide o número de Mersenne $M_p = 2^p - 1$.*

Demonstração. Suponha que q é primo. Como $p \equiv 3 \pmod 4$, podemos escrever $p = 4a+3$ para algum inteiro a. Temos $q = 2p+1 = 8a+7$ e daí $q^2 - 1 \equiv 0 \pmod{16}$. Calculando o símbolo de Legendre de 2 módulo q, encontramos $\left(\frac{2}{q}\right) = (-1)^{\frac{(q^2-1)}{8}} = 1$. Assim 2 é resíduo quadrático módulo q e existe um inteiro c tal que $c^2 \equiv 2 \pmod q$. Portanto,

$$2^p = 2^{\frac{q-1}{2}} \equiv (c^2)^{\frac{q-1}{2}} = c^{q-1} \equiv 1 \pmod q,$$

pois c é primo com q. Segue que $M_p = 2^p - 1 \equiv 0 \pmod q$, ou seja, q divide M_p.

Por outro lado, assuma que q divide M_p. Se q não for primo, seja r seu menor fator primo. Temos $r^2 \leq q$. Como r divide q e q divide M_p, temos $2^p \equiv 1 \pmod r$. Assim, a ordem de 2 módulo r divide p. Como $2 \neq 1 \pmod q$, a ordem de 2 módulo r é igual a p. Pelo PTF, temos $2^{r-1} \equiv 1 \pmod r$ e a ordem de 2 módulo r divide $r-1$, ou seja, p divide $r-1$. Assim, $p < r$. Segue que $p^2 < r^2 \leq q = 2p+1$. Mas

$$p^2 \leq 2p+1 \Rightarrow p^2 - 2p + 1 \leq 2 \Rightarrow p - 1 \leq \sqrt{2} \Rightarrow p \leq 1 + \sqrt{2} \Rightarrow p = 2.$$

Essa contradição mostra que $q = 2p+1$ é primo. □

Podemos utilizar o teorema de Euler-Lagrange para acelerar o processo de obtenção de primos seguros:

- O preço que se paga por isso é descartar todos os primos $p \equiv 1 \pmod 4$ e com eles, possivelmente, algum primo seguro $q = 2p+1$. Isto ocorre, por exemplo, com $p = 41$ e $q = 83$.
  As condições $p \equiv 3 \pmod 4$ e $p \equiv 2 \pmod 3$ equivalem, pelo TCR, a $p \equiv 11 \pmod{12}$. Dessa forma, podemos nos concentrar nos primos da forma $p = 12a + 11$.

- O ganho está no teste de primalidade para $2p+1$. Devemos apenas verificar se $2p+1$ divide M_p, ou seja, se vale a relação $2^p \equiv 1 \pmod{2p+1}$.

6.2.5 Utilizando resíduos quadráticos

Sejam p e q primos ímpares com $q = 2p+1$. Como

$$\left(\frac{q}{p}\right) = \left(\frac{2p+1}{p}\right) = \left(\frac{1}{p}\right) = 1,$$

q é resíduo quadrático módulo p. A lei de reciprocidade quadrática de Gauss estabelece uma relação entre os símbolos de Legendre $\left(\frac{p}{q}\right)$ e $\left(\frac{q}{p}\right)$:

$$\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right)(-1)^{(\frac{p-1}{2})(\frac{q-1}{2})}.$$

Sabendo que $\left(\frac{q}{p}\right) = 1$, podemos determinar $\left(\frac{p}{q}\right)$:

$$\left(\frac{p}{q}\right) = (-1)^{(\frac{p-1}{2})(\frac{q-1}{2})} = (-1)^{\frac{p(p-1)}{2}} = (-1)^{\frac{p-1}{2}}.$$

Com a intenção de manter os resultados obtidos na seção anterior, assumiremos $p \equiv 3 \pmod 4$. Temos $(p-1)/2 \equiv 1 \pmod 2$ e assim $\left(\frac{p}{q}\right) = -1$. Nesse caso, p não é resíduo quadrático módulo q. Com essa nova informação construiremos um algoritmo mais eficiente para a geração de primos seguros:

Dado um primo p, eliminamos $(p-1)/2$ se este for resíduo quadrático módulo p, sem que seja necessário testar sua primalidade.

Existem algoritmos bastante eficientes para o cálculo de $\left(\frac{a}{p}\right)$, com p primo, que têm custo significativamente mais baixo que testar primalidade [7].

6.2.6 Um novo algoritmo

Nesta seção combinaremos os algoritmos e teoremas das seções anteriores, produzindo assim um novo algoritmo para a geração de primos seguros. A característica principal desse algoritmo é o fato de que, sob certas circunstâncias, é possível evitar os testes de primalidade (com seus altos custos computacionais), utilizando o teorema de Euler-Lagrange e a Lei de reciprocidade quadrática.

Inicialmente assumimos que é dado um algoritmo $\mathbb{A}$ que produz primos p, de k bits, com $p \equiv 3 \pmod 4$, $p \pmod r \neq 1$ e $p \pmod r \neq (r-1)/2$ para todo primo $r \leq B$, onde B é uma constante pré-fixada (ver seções 6.2.3 e 6.2.4). O algoritmo a seguir, batizado de ELRQ (por utilizar o Teorema de Euler-Lagrange e a Lei de reciprocidade quadrática), sintetiza as idéias estudadas neste capítulo:

Algoritmo *ELRQ*
Entrada k: inteiro positivo
Saída p: primo seguro com k ou $k+1$ bits
1 **faça**
2 $\quad p \leftarrow \mathbb{A}(k)$
3 $\quad r \leftarrow 2^p \pmod{2p+1}$
4 **enquanto** $r \neq 1$ e $\left(\left(\frac{(p-1)/2}{p}\right) = 1 \text{ ou } \frac{p-1}{2} \text{ é composto }\right)$
5 **se** r=1 **então**
6 $\quad$ **devolva** 2p+1
7 **senão**
8 $\quad$ **devolva** p
9 **fim se**
fim

Algoritmo 3 Geração de primos seguros: ELRQ

Algumas observações devem ser feitas a respeito da linha 4 do algoritmo ELRQ:

- Caso a condição $r = 1$ seja satisfeita, o Teorema de Euler-Lagrange assegura a primalidade de $2p+1$ que será, portanto, um primo seguro.
- É importante testar a condição $\left(\frac{(p-1)/2}{p}\right) = 1$ antes de verificar se $(p-1)/2$ é primo. Se essa condição for verdadeira, o número $(p-1)/2$ será descartado sem que seja necessário testar sua primalidade.

CAPÍTULO 7

Conclusão e trabalhos futuros

7.1 Conclusão

Há muito a criptografia ganhou "status" de ciência. Entretanto, sua consolidação como tema relevante na nossa sociedade só ocorreu quando a comunicação segura entres entidades dos mais variados setores tornou-se indispensável. Embora nem sempre se perceba, lidamos com criptografia quase todo o tempo. Quando acessamos nossa conta bancária, quando fazemos compras com cartão de crédito ou, simplesmente, acessando alguns sites na internet, diversos esquemas criptográficos são acionados para garantir a privacidade de nossas ações.

Infelizmente, nenhum esquema criptográfico é totalmente seguro. Precisamos garantir que, mesmo dispondo de poderosos recursos computacionais, um adversário não terá sucesso (pelo menos probabilisticamente) em decifrar mensagens que não lhe são destinadas. As hipóteses baseadas em problemas computacionalmente difíceis são a garantia (teórica) de que estamos razoavelmente seguros. O surgimento do conceito de grupos pseudo-livres traz consigo uma nova e mais geral maneira de lidar com quesões relativas à segurança de criptossistemas assimétricos.

Sabe-se, há algum tempo, que a utilização de primos seguros é um pré-requisito importante para a segurança do RSA. O teorema de Micciancio torna ainda mais evidente essa necessidade. Assim, a busca por algoritmos eficientes para a geração de primos seguros está fortemente associada à utilização do RSA. O algoritmo ELRQ por nós proposto é uma alternativa eficiente para a geração de primos seguros.

7.2 Problemas em aberto

No contexto de grupos pseudo-livres algumas questões importantes permanecem sem resposta como, por exemplo:

- *Existem outros grupos pseudo-livres além de $\mathbb{Z}_N^*$, onde N é produto de dois primos seguros?*

  Dos diversos tipos de grupos que são utilizados em criptografia, até o presente momento apenas um é reconhecido como sendo pseudo-livre. Isto limita muito o raio de aplicação das hipóteses de segurança estabelecidas por esse conceito.

- *O resultado de Micciancio permanece válido se N não é produto de primos seguros?*

  Uma resposta afirmativa para esse problema permitiria que fosse utilizada uma classe de números primos maior que a classe dos primos seguros em criptografia RSA.

- *Pode-se deduzir que $\mathbb{Z}_N^*$ é pseudo-livre assumindo que fatorar é difícil ao invés de assumir que o problema RSA forte é difícil?*

  A importância de uma resposta afirmativa para esse problema reside no fato que muitos sistemas criptográficos tomam como hipótese de segurança o problema da fatoração de inteiros que é considerado computacionalmente difícil.

7.3 Trabalhos futuros

Possíveis continuações no estudo do tema "grupos pseudo-livres" poderiam dar-se em duas frentes:

- **Teórica:** Busca por novos exemplos de grupos pseudo-livres. Dentre os principais exemplos de grupos utilizados em criptografia estão os grupos de curvas elípticas sobre corpos finitos e uma versão do resultado de Micciancio para tais grupos seria de grande importância.
  Um outro ponto importante seria a busca por grupos pseudo-livres não abelianos (isto é, não comutativos), uma vez que a não comutatividade oferece uma dificuldade adicional para a quebra de criptossistemas sobre grupos não abelianos.

- **Prática:** Busca por algoritmos eficientes para a geração de primos seguros, uma vez que os únicos grupos pseudo-livres conhecidos até o presente momento têm como base tais números primos.

Referências Bibliográficas

[1] M. Agrawal, N. Kayal and N. Saxena, *Primes in P*. Ann. of Math. (2), 160:2 (2004) pp. 781-793.

[2] K. Ambos-Spies, *On the relative complexity of hard problems for complexity classes without complete problems*. TCS, 64 pp. 43-61, 1989.

[3] E. Bach, *Discrete logarithms and factoring*. Technical Report CSD-84-186, University of California at Berkeley, 1984.

[4] T. Baker, J. Gill, and R. Solovay, *Relativizations of the P =? NP question*. SICOMP: SIAM Journal on Computing, 1975.

[5] D. Boneh and R. Venkatesan, *Breaking RSA may be easier than factoring.* In Proceedings Eurocrypt '98, Lecture Notes in Computer Science, Vol. 1233, Springer-Verlag, pp. 59-71, 1998.

[6] D. Boneh, *Twenty years of attacks on the RSA cryptosystem.* Notices Amer. Math. Soc. 46 (1999), pp. 203-213.

[7] D.M. Bressoud, *Factorization and primality testing*. Undergraduate Texts in Mathematics, Springer-Verlag, 1989.

[8] H. Cohen, *A course in computational algebraic number theory*. Graduate Texts in Mathematics 138, Springer-Verlag, 1993.

[9] S. Cook, *The P versus NP Problem.* Manuscript prepared for the Clay Mathematics Institute for the Millennium Prize Problems, April 2000 (revised November 2000).

[10] S. C. Coutinho, *Primalidade em tempo polinomial: Uma introdução ao algoritmo AKS*. Coleção Iniciação Científica, SBM, 2004.

[11] S. C. Coutinho, *Números inteiros e Criptografia RSA.* IMPA, Rio de Janeiro-RJ, 2005.

[12] H. Delfs and H. Knebl, *Introduction to Cryptography*. Springer-Verlag, 2002.

[13] W. Diffie and M. Hellman, *New Directions in Cryptography*. IEEE Transactions of Information Theory, vol. IT-22 (1976), pp. 644-654.

[14] T. El Gamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*. IEEE Trans. Inform. Theory, 31:469-472, 1985.

[15] M. Garey and D. Johnson, *Computers and Intractability: a Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[16] C. F. Gauss, *Disquisitiones Arithmeticae*. Braunschweig, 1801. English Edition, Springer-Verlag, New York, 1986.

[17] O. Goldreich, *Foundations of Cryptography: Basic Tools*. Cambridge Univ. Press, 2001.

[18] O. Goldreich, *Randomized Methods in Computation*. Lecture Notes, Spring 2001. Available at http://www.wisdom.weizmann.ac.il/õded/rnd.html

[19] S. Goldwasser and M. Bellare, *Lecture notes on cryptography*. Lecture notes for a summer course on cryptography at the Massachusetts Institute of Technology (MIT), 1996-2001.

[20] G. H. Hardy and E. M. Wright, *An introduction to the theory of numbers*. Oxford University Press, 1979.

[21] S. Hohenberger, *The cryptographic impact of groups with infeasible inversion*. Master's thesis, EECS Dept., MIT, June 2003.

[22] S. Homer and A. Selman, *Computability and Complexity Theory*. Springer, 2000.

[23] K. Ireland and M. Rosen, *A Classical Introduction to Modern Number Theory*. Graduate Texts in Mathematics, Springer, 1990.

[24] R. M. Karp, *Reducibility Among Combinatorial Problems*. In Complexity of Computer Computations, Proc. Sympos. IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y.. New York: Plenum, p.85-103, 1972.

[25] N. Koblitz, *Algebraic aspects of cryptography*. Algorithms and Computation in Mathematics, 3., Springer-Verlag, Berlin, 1998.

[26] S. Lang, *Algebra*. Graduate Texts in Mathematics, Springer, 2002.

[27] R. C. Lyndon, *Equations in free groups*. Trans. Amer. Math. Soc.,96: 445-457, 1960.

[28] U. Maurer and S. Wolf, *The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms*. SIAM Journal on Computing, 28(5):1689-1721, 1999.

[29] K. McCurley, *The Discrete Logarithm Problem in C*. Pomerance, ed., Cryptology and Computational Number Theory, American Mathematical Society, Proceedings of Symposia in Applied Mathematics, Vol. 42, 1990, pp. 49-74.

[30] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of applied cryptography*. CRC Press Series on Discrete Mathematics and its Applications., CRC Press, Boca Raton, FL, 1997.

[31] R. Merkle, *Secure Communications over Insecure Channels.* Communications of the ACM, vol. 21, No. 4 (1978), pp. 294-499.

[32] D. Micciancio, *The RSA Group is Pseudo-Free.* In Advances in Cryptology - Eurocrypt 2005, Aarhus, Denmark. May 2005. LNCS 3494, Springer, pp. 387-403.

[33] R. Mollin, *RSA and Public-Key Cryptography*. Chapman and Hall/CRC, 2003.

[34] D. Naccache, *Double-speed safe prime generation*. Crypto Eprint Archive entry 2003:175, http://eprint.iacr.org/2003.

[35] C. Papadimitriou, *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.

[36] G. Pólya, *Heuristic reasoning in the theory of numbers*. Amer. Math. Monthly 66 (1959), 375-384.

[37] M. O. Rabin, *Digitalized Signatures and Public-Key Functions as Intractable as Factorization*. MIT Laboratory for Computer Science, January 1979.

[38] P. Ribenboim, *The new book of prime number records.* 3rd ed., Springer-Verlag, New York, 1995.

[39] H. Riesel, *Prime numbers and computer methods for factorization.* Progress in Mathematics 126, Birkhäuser, 2nd edition, 1994.

[40] R. L. Rivest, *On the notion of pseudo-free groups*. In M. Naor, editor, Theory of cryptography conference - Proceedings of TCC 2004, volume 2951 of Lecture Notes in Computer Science, pp. 505-521, Cambridge, MA, USA, Feb. 2003. Springer.

[41] R.L. Rivest, A. Shamir and L. Adelman, *A method for obtaining digital signature and public key cryptosystem.* Comm. ACM 21 (2) (1978), pp. 120-126.

[42] R. L. Rivest and B. Kaliski. *The RSA problem*. Encyclopedia of Cryptography and Security, 2003.

[43] A. Selman, *Complexity Theory Retrospective.* Springer, New York, 1988.

[44] V. Shoup, *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2005.

[45] S. Singh, *The code book. The Evolution of Secrecy from Mary, Queen of Scots to Quantum Cryptography*. Anchor Books, New York, 1999.

[46] M. Sipser, *The history and status of the P versus NP question.* In Proceedings of the 24th ACM Symposium on the Theory of Computing, pp. 603-618. ACM, New York, 1992.

[47] M. Sipser, *Introduction to the Theory of Computation, second edition*. Course Technology (Thomson), 2005.

[48] D. Welsh, *Codes and Cryptography*. Oxford University Press, Oxford, 1988.

[49] M. J. Wiener, *Safe Prime Generation with a Combined Sieve*. Crypto Eprint Archive entry 2003:186, http://eprint.iacr.org/2003.

[50] S. Y. Yan, *Primality Testing and Integer Factorization in Public-Key Cryptography*. Kluwer Academic Publishers, 2004.

[51] S. Yates, *Sophie Germain primes*. The Mathematical Heritage of C. F. Gauss (G. M. Rassias, ed.), World Scientific, 1991, pp. 882-886.

Internet

[52] Wikipedia, The Free Encyclopedia, *Complexity classes P and NP*.
http://en.wikipedia.org/wiki/Complexity_classes_P_and_NP

[53] Wikipedia, The Free Encyclopedia, *NP-Hard*.
http://en.wikipedia.org/wiki/NP-hard

[54] Wikipedia, The Free Encyclopedia, *Sophie Germain primes*.
http://en.wikipedia.org/wiki/Sophie_Germain_prime

[55] Wikipedia, The Free Encyclopedia, *Rabin cryptosystem.*
http://en.wikipedia.org/wiki/Rabin_cryptosystem

[56] Wikipedia, The Free Encyclopedia, *Goldwasser-Micali cryptosystem.*
http://en.wikipedia.org/wiki/Goldwasser-Micali_cryptosystem

Dissertação de Mestrado apresentada por **Marcelo Gama da Silva** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título, **"Grupos Pseudo-Livres, Primos Seguros e Criptografia RSA"**, orientada pelo **Prof. Ruy José Guerra Barretto de Queiroz** e aprovada pela Banca Examinadora formada pelos professores:

Prof. Anjolina Grisi de Oliveira
Centro de Informática / UFPE

Prof. Ricardo Menezes Campello de Souza
Departamento de Eletrônica e Sistemas / UFPE

Prof. Ruy José Guerra Barretto de Queiroz
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 26 de fevereiro de 2007.

Prof. FRANCISCO DE ASSIS TENÓRIO DE CARVALHO
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Este volume foi tipografado em LaTeX na classe UFPEThesis (www.cin.ufpe.br/~paguso/ufpethesis).