Capítulo 15

Uma olhada em complexidade e criptografia

15.1 Uma aula de Connecticut na corte do Rei Arthur

Na corte do Rei Arthur[1] residiam 150 cavaleiros e 150 damas-na-espera. O rei decidiu casá-las, mas o problema era que alguns pares se detestavam tanto que eles nem sequer se casariam, quanto mais se falr! O Rei Arthur tentou várias vezes emparelhá-los mas toda vez ele esbarrou em conflitos. Daí ele chamou Merlin o Mago e ordenou-o que encontrasse um emparelhamento no qual todo par estivesse desejando se casar. Agora, Merlin tinha poderes sobrenaturais e ele viu imediatamente que nenhum dos 150! possíveis emparelhamentos era factível, e isso ele disse ao rei. Mas Merlin não era apenas um grande mago, mas um também sujeito suspeito, e o Rei Arthur não confiava realmente nele. "Encontre um emparelhamento ou eu o sentenciarei a ficar preso numa caverna para sempre!" disse Arthur.

Felizmente para Merlin, ele podia usar seus poderes sobrenaturais para varrer a literatura científica futura, e encontrou vários artigos no início do século XX que forneciam a *razão* por que tal emparelhamento não poderia existir. Ele então voltou ao Rei King quando todos os cavaleiros e damas estavam presentes, e pediu a umas certas 56 damas para se colocar de um lado do rei e 95 cavaleiros no outro lado, e perguntou: "Alguma de vocês damas, está querendo casar com algum desses cavaleiros?", e quando todas disseram "Não!", Merlin disse: "Oh Rei, como é que você pode me mandar encontrar um marido para cada uma dessas 56 damas entre os 55 cavaleiros restantes?" Daí o Rei, cuja eduação real de fato incluía o Princípio da Casa-de-Pombos, viu que nesse caso Merlin tinha falado a verdade e graciosamente o dispensou.

Algum tempo passou e o rei notou que nos jantares servidos para os 150 cavaleiros na famosa távola redonda, vizinhos frequentemente discutiam e até mesmo brigavam. Arthur achou isso ruim para a digestão e então uma vez mais ele chamou Merlin e

[1] De L. Lovász–M.D.Plummer: *Matching Theory*, Akadémiai Kiadó - North Holland, Budapest, 1986 (com pequenas modificações), com a grata permissão de Mike Plummer. O material foi desenvolvido como notas de aula na Yale University, New Haven, Connecticut.

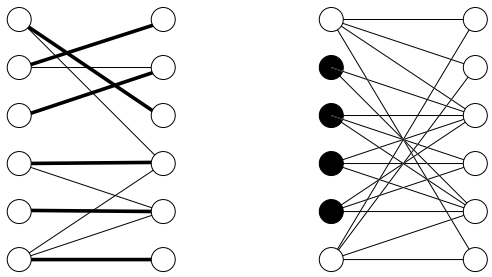

Figura 15.1: Um bigrafor com um emparelhamento perfeito e um sem

o ordenou a encontrar uma maneira de sentar os 150 cavaleiros ao redor da távola de modo que cada um deles deveria sentar entre dois amigos. Novamente, usando seus poderes sobrenaturais Merlin viu imediatamente que nenhuma das 150! formas de assentar resolveria, e isso ele reportou ao rei. Novamente, o rei o ordenou a encontrar um ou explicar por que era impossível. "Oh eu bem que queria que existisse alguma razão simples que eu pudesse lhe dar! Com alguma sorte poderia haver um cavaleiro tendo apenas um amigo, e aí você também poderia ver que imediatamente que o que você me pede é impossível. Mas, aliás!, não há tal razão simples aqui, e eu não posso explicar a vocês mortais por que nenhuma forma de assim assentá-los existe, a menos que você esteja pronto a passar o resto de sua vida ouvindo meus argumentos!" O rei estava naturalmente indesejoso de fazer isso portanto Merlin viveu aprisionado em uma caverna desde então. (Uma grande perda para a matemática aplicada!)

O moral dessa estória é que existem propriedades de grafos que, quando elas se verificam, são facilmente demonstradas que se dão. Se um grafo tem um emparelhamento perfeito, ou um ciclo Hamiltoniano, isso pode ser "provado" facilmente por meio da exibição de um tal emparelhamento ou ciclo. Se um grafo bipartido *não* tem um emparelhamento perfeito, então isso pode ser "provado" exibindo-se um subconjunto X of uma classe de cor que tem menos que $|X|$ vizinhos no outro. O leitor (e o Rei Arthur!) são remetidos à Figura 15.1 na qual o grafo G tem um emparelhamento perfeito (indicado pelas linhas grossas), mas o grafo H não. Para nos convencer (e ao Rei) desse último, considere os quatro pontos pretos e seus vizinhos.

A maioria das propriedades grafo-teóricas que nos interessa têm essa estrutura lógica. Se é fácil provar (certificar, exibir) que a propriedade se verifica, então a propriedade é chamada (no jargão da ciência da computação) uma *propriedade NP* (se você realmente deseja saber, NP é uma abreviação de *Tempo Não-deterministicamente Polinomial*, mas seria difícil explicar de onde essa frase altamente técnica vem). Os dois problemas que Merlin tinha que enfrentar—a existência de um emparelhamento perfeito e a existência de um ciclo Hamiltoniano—são claramente propriedades NP. Mas propriedades NP também aparecem um tanto frequentemente em outras partes da matemática. Uma propriedade NP muito importante dos números naturais é sua *propri-*

edade de ser composto: se um número natural é composto então isso pode ser exibido facilmente mostrando-se uma decomposição $n = ab\ (a, b > 1)$.

As observações que fizemos até aqui explicam como Merlin pode permanecer livre se ele tiver sorte e a tarefa a ele atrib'ıda pelo Rei Arthur tem uma solução. Por exemplo, suponhamos que ele pudesse encontrar um boa maneira de assentar os cavaleiros para o jantar. Ele poderia então convencer o Rei Arthur de que seu plano de assentamento era "bom" perguntando se havia alguém sentando ao lado de algum inimigo seu (ou simplesmente esperar e ver se o jantar ia tranqüilo). Isso mostra que a propriedade do problema correspondente "grafo das amizades" que ele contém um ciclo Hamiltoniano é uma propriedade NP. Mas como ele poderia sobreviver à ira de Arthur no caso do problema do emparelhamento e não no caso do problema do assentamento quando esses problemas *não* têm soluções? O que distingüe a não-existência de um ciclo Hamiltoniano em um grafo da não-existência de um emparelhamento perfeito em um grafo bipartido? De nossa estória, esperamos que a resposta esteja clara: *a não-existência de um emparelhamento perfeito em um grafo bipartido é também uma propriedade NP* (isso é uma implicação principal do Teorema do Emparelhamento 10.3.1), enquanto que a não-existência de um ciclo Hamiltoniano em um grafo não é! (Para ser preciso, nenhuma prova desse último fato é conhecida, mas existe forte evidência a favor dele.)

Portanto, para certas propriedades NP a negação da propriedade é novamente uma propriedade NP. Um teorema asseverando a equivalência de uma propriedade NP com a negação de uma outra propriedade NP é chamado de *boa caracterização*. Existem boas caracterizações famosas por toda a teoria de grafos e alhures.

Muitas propriedades NP são ainda melhores. Enfrentando o problema de casar seus cavaleiros e damas, o próprio Arthur (digamos, após ler este livro) poderia decidir por si próprio se o problema era ou não solúvel: ele poderia rodar o algoritmo descrito na seção 10.4. Muito trabalho, mas provavelmente realizável com a ajuda de pessoas bastante comuns, sem usar os talentos sobrenaturais de Merlin. Propriedades que podem ser decididas eficientemente são chamadas propriedades *na classe P* (aqui P significa *Tempo Polinomial*, uma definição exata porém um tanto técnica da frase "eficientemente"). Muitas outras propriedades simples de grafos discutidas neste livro também pertencem à classe: conectividade, ou a existência de um ciclo. Um dos nossos problemas favoritos, aquele de se decidir se um número é primo, é conjecturado como pertencendo a essa classe também, mas isso não está provado.[2] (O algoritmo que descrevemos na seção 6.10 não chega a se qualificar para a classe P, porque ele usou uma seleção aleatória selection da base a.)

A introdução das noções de Tempo Polinomial e propriedades NP sinalizaram o nascimento da teoria moderna da complexidade. Noções e paradigmas dessa teoria têm penetrado numa grande parte da matemática e suas aplicações. No que se segue descrevemos como idéias de teoria da complexidade podem ser aplicadas em uma das áreas mais importantes de teoria da computação, a saber, criptografia.

[2]N.T. Atualmente já existe uma prova de que o teste de primalidade é um problema P.

15.2 Criptografia clássica

Desde que a escrita foi inventada as pessoas têm se interessado não apenas em usá-la para se comunicar com seus parceiros, mas também em tentar esconder de seus adversários o conteúdo de sua mensagem. Isso leva à criptografia (ou criptologia), a ciência da comunicação secreta.

A situação básica é que uma parte, digamos Rei Arthur, deseja enviar uma mensagem ao Rei Bela. Existe, entretanto, o perigo que o diabólico Caesar Caligula intercepte a mensagem e fique sabendo de coisas que ele não deveria saber. A mensagem, compreensível até para Caligula, é chamada de o *texto pleno*. Para proteger seu conteúdo, Rei Arthur *encripta* sua mensagem. Quando Rei Bela a recebe, ele tem que *decriptá*-la de modo a ser capaz de ler a mensagem. Para que os Reis sejam capazes de *encriptar* e *decriptar* a mensagem, eles têm que saber algo que Caesar não sabe: essa informação é a *chave*.

Muitos criptossistemas têm sido usados na história; a maioria deles, na verdade, acabam sendo inseguros, especialmente se o adversário pode usar ferramentas poderosas de computação para quebrá-lo.

Talvez o método mais simples é o *código de substituição*: substituimos cada letra do alfabeto por uma outra letra. A *chave* é a tabela que contém para cada letra a letra a ser substituída por ela. Enquanto que uma mensagem encriptada dessa maneira parece totalmente embaralhada, códigos de substituição são na verdade fáceis de quebrar. Resolver o exercício 15.1 deverá tornar claro como o comprimento e as posições das palavras podem ser usadas para desvendar o significado original das letras, se a separação entre palavras são preservadas (i.e., se "espaço em branco"não for substituído por um outro caracter). Porém mesmo se a separação em palavras for ocultada, uma análise da freqüência de várias letras fornece informação suficiente para quebrar o código de substituição.

Folha-de-uso-único. Existe um outro método simples e frequentemente utilizadao, que é muito mais seguro: o uso de "folha-de-uso-único". Esse método é muito seguro; ele foi usado, por exemplo, durante a Segunda Grande Guerra para comunicação entre o presidente americano e o primeiro-ministro britânico. Sua desvantagem é que ele requer uma chave muito longa ("folha"), que só pode ser usada uma vez.

Uma folha-de-uso-único é uma cadeia aleatoriamente gerada de 0's e 1's. Digamos, aqui está uma:

1100011100001000011001010010010010110011001010110000111011000001 0

Ambos os Reis Arthur e Bela conhecem essa seqüência (ela foi enviada com boa antecipação por um mensageiro). Agora o Rei Arthur quer enviar a seguinte mensagem ao Rei Bela:

ATAQUE SEGUNDA

Primeiramente, ele tem que convertê-la para 0's e 1's. Não está claro que reis medievais tinham o conhecimento de fazer isso, mas o leitor deve ser capaz de pensar de várias maneiras: usando códigos ASCII, ou Unicodes das letras, por exemplo. Mas queremos manter as coisas simples, portanto numeramos as letras de 1 a 26, e então escrevemos

a representação binária dos números, colocando 0's na frente de modo que obtemos uma cadeia de comprimento 5 para cada letra. Por conseguinte, temos "00001" para A, "00010" para B, etc. Usamos "00000" para "espaço em branco". A mensagem acima fica:

00001100100000110001101010010100000100110010100111101010111000001

Isso pode parecer suficientemente oculto, mas Caligula (ou, na verdade, um dos excelentes cientistas gregos que ele mantém na sua corte como escravos) poderia facilmente adivinhar o que isso representa. Para codificá-la, Arthur adiciona bit-a-bit a folha-de-uso-único à mensagem. Ao primeiro bit da mensagem (que é 0) ele adiciona o primeiro bit da folha (1) e escreve o primeiro bit da segunda mensagem codificada: $0 \oplus 1 = 1$. Ele calcula o segundo, o terceiro, etc. bits de modo semelhante: $0 \oplus 1 = 1$, $0 \oplus 0 = 0$, $0 \oplus 0 = 0$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$,... (Note que ele usa essa estranha soma do corpo de 2-elementos, para a qual $1 \oplus 1 = 0$.) Uma outra maneira de dizer o que o Rei Arthur faz é a seguinte: se o k-ésimo bit da folha é 1, ele inverte o k-ésimo bit do texto; caso contrário, ele o deixa como estava.

Daí Arthur calcula a mensagem codificada:

11001011100010110111111111011000010100000000000010111001000010000 11

Ele envia isso ao Rei Bela, que, por sua vez, usando a folha-de-uso-único, pode facilmente inverter novamente os bits apropriados, e recuperar a mensagem original.

Mas Caligula não conhece a folha-de-uso-único (nem tampouco seus excelentes cientistas), portanto ele não tem idéia sobre quais bits foram invertidos, e portanto ele fica impotente. A mensagem está segura.

Pode ser caro assegurar que Emissor e Receptor ambos tenham tal chave em comum; mas note que a chave pode ser enviada num momento mais seguro e por um método completamente diferente daquele usado para enviar a mensagem (além do mais, pode ser possível concordar sobre uma chave mesmo sem de fato pass-la; mas isso nos levaria longe demais em criptografia).

Uma vez que os Reis conseguiram passar a chave um para o outro, é tentador reusá-la; afinal de contas, se Bela encripta sua resposta por meio da mesma folha, ela ainda está completamente aparentemente-aleatória. Resolvendo os exercícios 15.2–15.3 você verá que isso não é uma boa idéia, e verá também algumas outras fragilidades do método da folha-de-uso-único.

15.1 Para a mensagem abaixo, os Reis usaram o código de substituição. Caligula interceptou a mensagem e um tanto facilmente a quebrou. Você pode fazer isso também?

U GXUAY LS ZXMEKW AMG TGGTIY HMD TAMGXSD LSSY, FEG GXSA LUGX HEKK HMDIS. FSKT

15.2 Uma certa vez, Arthur cometeu um erro de usar a folha-de-uso-único deslocada: o primeiro bit do texto pleno ele codificou usando o segundo bit da folha, o segundo bit do texto pleno ele codificou usando o terceiro bit da folha, etc. Ele notou seu erro depois que ele enviou a mensagem. Receando que Bela não entenderia sua mensage, ele a codificou novamente (agora corretamente) usando a mesma folha-de-uso-único, e a enviou a Bela por um outro mensageiro, explicando o que aconteceu.

Caligula interceptou ambas as mensagens, e foi capaz de recuperar o texto pleno. Como?

15.3 Os Reis estavam ficando com poucas folhas-de-uso-único, e portanto Bela teve que usar as mesmas folhas para codificar sua resposta que foram usadas para a mensagem de Arthur. Caligula interceptou ambas as mensagens, e foi capaz de reconstruir ambos os textos plenos. Como?

15.3 Como guardar o último movimento em xadrez?

A criptografia moderna começou no final dos anos 1970 com a idéia de que não é apenas a falta de informação que pode proteger nossa mensagem contra um araponga não-autorizado, mas também a *complexidade computacional* de processá-la. A idéia pode ser ilustrada pelo seguinte exemplo simples.

Alice e Bob estão jogando xadrez por telefone. Eles querem interromper o jogo durante a noite; como é que eles podem fazê-lo de tal modo que a pessoa que vai jogar não obtenha a vantagem imprópria de ser capaz de pensar sobre sua jogada durante a noite toda? Num torneio, o último movimento não é feito no tabuleiro, mas somente escrito num papel, colocado num envelope, e entregue ao juiz. Na manhã seguinte, o envelope é aberto, e o outro jogador toma conhecimento de qual foi o movimento ao mesmo tempo em que seu relógio começa a rodar. Mas agora os dois jogadores não têm juiz, não têm envelope, e nenhum contato exceto a linha telefônica. O jogador fazendo o último movimento (digamos, Alice) tem que enviar a Bob alguma mensagem. A manhã seguinte (ou sempre que eles continuam o jogo) ela tem que dar alguma informação adicional, alguma "chave", que permite a Bob reconstruir o movimento. Bob não deveria ser capaz de reconstruir o movimento de Alice sem a chave; Alice não deveria ser capaz de mudar de opinião da noite para o dia e modificar seu movimento.

Certamente isso parece ser impossível! Se ela dá informação suficiente inicialmente para determinar univocamente seu movimento, Bob conhecerá o movimento cedo demais; se a informação dada inicialmente permite vários movimentos, então ela pode pensar sobre ele durante a noite, verificar qual o melhor entre eles, e dar a informação remanescente, a "chave" em conformidade.

Se medirmos informação no sentido da teoria clássica da informação, então não há saída para esse dilema. Mas complexidade vem ao nosso socorro: não é suficiente *comunicar* a informação, é preciso também que ela seja *processada*.

Portanto aqui está uma solução para o problema, usando teoria elementar dos números! (Muitos outros esquemas podem ser desenhados.) Alice e Bob concordam em codificar todo movimento como um número de 4-dígitos (digamos, '11' significa 'K', '6' significa 'f', e '3' significa ele próprio, de modo que '1163' significa 'Kf3'). Até aí, isso é apenas notação.

A seguir, Alice estende os quatro dígitos descrevendo seu movimento para um número primo $p = 1163\ldots$ com 200 dígits. Ela também gera um outro primo q com 201 dígitos e calcula o produto $N = pq$ (isso levaria bastante tempo no papel, mas é trivial usando um computador pessoal). O resultado é um número com 400 ou 401 dígitos; ela envia esse número a Bob.

Na manhã seguinte, ela envia ambos os fatores primos p e q a Bob. Ele reconstrói o movimento de Alice a partir dos primeiros quatro dígitos do menor primo. Para ter certeza de que Alice não estava trapaceando, ele deveria conferir que p e q são primos

e que seu produto é N.

Vamos argumentar que esse protocolo faz a tarefa.

Primeiro, Alice não pode mudar de opinião durante a noite. Isso é porque o número N contém toda a informação sobre seu movimento: isso está codificado como os primeiros quatro dígitos do menor fator primo de N. Portanto Alice se compromete com o movimento ao enviar N.

Mas exatamente porque o número N contém toda a informação sobre o movimento de Alice, Bob parece ter a vantagem, e ele de fato teria se ele tivesse tempo ilimitado ou computadores incrivelmente poderosos. O que ele tem que fazer é encontrar os fatores primos do número N. Mas como N tem 400 dígitos (ou mais), isso é uma tarefa desanimadoramente difícil com a tecnologia atual.

Alice pode trapacear enviando um par diferente (p', q') de primos na manhã seguinte? Não, porque Bob pode facilmente calcular o produto $p'q'$, e conferir se esse é de fato o número N que foi enviado na noite anterior. (Note o papel do Teorema 6.3.1 da *unicidade* da fatoração prima.)

Toda a informação sobre o movimento de Alice está codificada nos primeiros 4 dígitos do menor fator primo p. Poderíamos dizer que o resto de p e o outro fator primo q servem como uma "caixa de depósito": eles escondem essa informação de Bob, e podem ser abertos apenas se a chave apropriada (a fatoração de N) estiver disponível. O ingrediente crucial desse esquema é *complexidade*: a dificuldade computacional para encontrar a fatoração de um inteiro.

Com a expansão da comunicação eletrônica nos negócios, muitas soluções da correspondência e do comércio tradicional têm que ser substituídas por versões eletrônicas. Vimos uma "caixa de depósito" eletrônica acima. Outros esquemas (similares ou mais complicados) podem ser encontrados para senhas eletrônicas, autorização, autenticação, assinaturas, impressão de marca-d'água, etc. Esses esquemas são extremamente importantes em segurança computacional, criptografia, caixas automáticos, e muitos outros campos. Os protocolos são frequentemente baseados em teoria simples dos números; na próxima seção discutimos (uma versão muito simplificada de) um deles.

15.4 Motivada pelo método da folha-de-uso-único, Alice sugere o seguinte protocolo para guardar o último movimento no jogo de xadrez deles: à noite, ela encripta o seu movimento (talvez com outro texto adicionado, para torná-lo razoavelmente longo) usando uma seqüência 0-1 aleatoriamente gerada como a chave (tal qual no método da folha-de-uso-único). Na manhã seguinte ela envia a chave a Bob, de modo que ele pode decriptar a mensagem. Bob deveria aceitar essa sugestão?

15.5 Alice modifica sua sugestão da seguinte maneira: ao invés da seqüência 0-1 aleatória, ela oferece usar um texto aleatório, porém significativo, como a chave. Para quem isso seria vantajoso?

15.4 Como verificar uma senha—sem aprendê-la?

Em um banco, um caixa automático funciona por nome e senha. Esse sistema é seguro desde que a senha seja mantida secreta. Mas existe um ponto fraco: o computador do

banco tem que armazenar a senha, e o administrador desse computador pode aprendê-la e mais tarde fazer mal uso dela.

Teoria da complexidade provê um esquema onde o banco pode verificar que o cliente realmente sabe a senha—sem armazená-la ele próprio! À primeira vista isso parece impossível—tal qual era o problema com guardar o último movimento em xadrez. E a solução (pelo menos aquela que discutimos aqui) usa o mesmo tipo de construção que nosso exemplo do xadrez por telefone.

Suponha que a senha seja um número primo de 100-dígitos p (isso é, obviamente, longo demais para uso cotidiano, mas ilustra melhor a idéia). Quando o cliente escolhe a senha, ele escolhe um outro primo q com 101 dígitos, forma o produto $N = pq$ dos dois primos, e dá ao banco o número N. Quando o caixa automático é usado, o cliente dá seu nome e a senha p. O computador do banco verifica se p é ou não um divisor de N; em caso positivo, ele aceita p como uma senha apropriada. A divisão de um número de 200 dígitos por um número de 100 dígitos é uma tarefa trivial para um computador.

Vamos assumir que o administrador do sistema aprende o número N armazenado juntamente com os arquivos de nosso cliente. Para usar esse número de modo a se passar pelo cliente, ele tem que encontrar um número de 100-dígitos que seja um divisor de N; mas isso é essencialmente o mesmo problema que encontrar a fatoração prima de N, e isso é desesperadoramente difícil. Portanto—muito embora toda a informação necessária esteja contida no número N—a complexidade computacional do problema da fatoração protege a senha do cliente!

15.5 Como encontrar esses primos?

Em nossos dois exemplos simples de "criptografia moderna", assim como em quase todos os outros, se precisa de números primos grandes. Sabemos que existem números primos arbitrariamente grandes (Teorema 6.4.1), Mas existe algum com 200 dígitos, começando com 1163 (ou quaisquer outros 4 dígitos dados)? Maple encontrou (em uns poucos segundos em um laptop!) o menor dos tais números primos:

116300000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000371

O menor inteiro com 200 dígitos começando com 1163 é $1163 \cdot 10^{196}$. Esse obviamente não é um primo, mas acima encontramos um primo muito próximo a ele. Deve haver zilhões de tais primos! De fato, um cálculo muito semelhante ao que fizemos na seção 6.4 mostra que o número de primos que Alice pode escolher daí é cerca de $1.95 \cdot 10^{193}$.

Isso é um bocado de possibilidades, mas como achar uma? Não seria bom usar o primo acima (o menor elegível): Bob poderia adivinhar isso e então descobrir o movimento de Alice. O que Alice pode fazer é preencher os 196 dígitos remanescentes aleatoriamente, e então testar se o número que ela obtém é um primo. Se não é, ela pode jogá-lo fora e tentar novamente. Como calculamos na seção 6.4, um em cada 460 números com 200 dígitos é um primo, portanto na média em 460 tentativas ela pega um

primo. Isso parece um bocado de tentativas, mas obviamente ela usa um computador; aqui está um que calculamos para você com esse método (em uns poucos segundos, novamente):

116314671287655576327990970455966069082836547600668873814489354662474360419891104680411103886895880574571557248000956963917403338545841859353548862232378231757755986473965270112717709727838946541458 9

Portanto vemos que no esquema de "envelope" acima, ambos os fatos computacionais mencionados na seção 6.10 desempenham um papel crucial: é fácil testar se um número é um primo (e através disso é fácil computar a encriptação), mas é difícil encontrar os fatores primos de um número composto (e portanto é difícil quebrar o criptossistema).

15.6 Criptografia de chave pública

Sistemas criptográficos usados na vida real são mais complexos que aqueles descritos na seção anterior—mas eles são baseados em princípios similares. Nesta seção esboçamos a matemática por trás dos sistemas mais comummente usados, o código RSA (cujo nome é uma homenagem aos seus inventores, Rivest, Shamir, e Adleman).

O protocolo. Alice gera dois números primos de 100-dígitos, p e q e calcula seu produto $m = pq$. Então ela gera dois números de 200-dígitos d e e tais que $(p-1)(q-1)$ seja um divisor de $ed - 1$. (Retornaremos à questão de como isso é feito.)

Os números m e e ela publica na sua página na internet (ou no catálogo telefônico), mas os fatores primos p e q e o número d permanecem seus segredos fortemente guardados. O número d é chamado sua *chave privada*, e o número e, sua *chave pública* (os números p e q ela pode até esquecer—eles não serão necessários para operar o sistema, mas simplesmente para inicializá-lo).

Suponha que Bob queira enviar uma mensagem a Alice. Ele escreve a mensagem como um número x (vimos anteriormente como fazer isso). Esse número x tem que ser um inteiro não-negativo menor que m (se a mensagem é mais longa, ele pode simplesmente quebrá-la em pedaços menores).

O próximo passo é o mais complicado: Bob calcula o resto de x^e módulo m. Como ambos x e e são inteiros enormes (200 dígitos), o número x^d tem mais que 10^{200} dígitos - não poderíamos sequer escrevê-lo no papel, imagine calculá-lo! Felizmente, não temos que calcular esse número, apenas seu resto quando dividido por m. Esse ainda é um número grande - mas pelo menos pode ser escrito no papel em 2-3 lines.

Portanto, seja r esse resto; esse é enviado a Alice. Quando ela o recebe, ela pode decriptá-lo usando sua chave privada d realizando essencialmente o mesmo procedimento que Bob realizou: ela calcula o resto de r^d módulo m. E—uma magia negra da teoria dos números, até você ver as explicações—esse resto é justamente o texto pleno.

E se Alice quiser enviar uma mensagem a Bob? Ele também precisa de passar pelo trabalho de gerar suas chaves privada e pública. Ele tem que pegar dois primos p' e q', calcular seu produto m', selecionar dois inteiros positivos d' e e' de modo que

$(p'-1)(q'-1)$ seja um divisor de $e'd'-1$, e finalmente publicar m' e e'. Então Alice pode enviá-lo uma mensagem segura.

A magia negra matemática por trás do protocolo. O fato-chave da matemática que usamos é o "Pequeno" Teorema de Fermat 6.5.1. Lembre-se que x é o texto pleno (escrito como um inteiro) e a mensagem encriptada r é o resto de x^e módulo m. Portanto podemos escrever

$$r \equiv x^e \pmod{m}.$$

Para decriptar, Alice eleva isso à d-ésima potência, para obter

$$r^d \equiv x^{ed} \pmod{m}.$$

Para ser mais preciso, Alice calcula o resto x' de r^d módulo m, que é o mesmo que o resto de x^{ed} módulo m. Queremos mostrar que esse resto é precisamente x. Como $0 \leq x < m$, basta argumentar que $x^{ed} - x$ é divisível por m. Como $m = pq$ é o produto de dois primos distintos, basta provar que $x^{ed} - x$ é divisível por ambos p e q.

Vamos considerar divisibilidade por p, por exemplo. A principal propriedade de e e d é que $ed-1$ é divisível por $(p-1)(q-1)$, e daí também por $p-1$. Isso quer dizer que podemos escrever $ed = (p-1)l + 1$, onde l é um inteiro positivo. Temos

$$x^{ed} - x = x\big(x^{(p-1)l} - 1\big).$$

Aqui $x^{(p-1)l}-1$ é divisível por $x^{p-1}-1$ (veja o exercício 6.6), e portanto $x\left(x^{(p-1)l}-1\right)$ é divisível por $x^p - x$, o que por sua vez é divisível por p pelo Teorema de Fermat.

Como fazer todo esse cálculo? Já discutimos como encontrar primos, e Alice pode seguir o método descrito na seção 6.10.

A próxima questão é o cálculo das duas chaves e e d. Uma delas, digamos e, Alice pode escolher aleatoriamente, da faixa $1..(p-1)(q-1)-1$. Ela tem que verificar que ele é primo em relação a $(p-1)(q-1)$; isso pode ser feito eficientemente com a ajuda do Algoritmo Euclideano discutido na seção 6.6. Se o número que ela escolheu não é primo em relação a $(p-1)(q-1)$, ela simplesmente o joga fora, e tenta um outro. Isso é semelhante ao método que usamos para encontrar um primo, e não é difícil ver que ela encontrará um número bom em não mais tentativas que ela pode encontrar um primo.

Mas se ela finalmente consegue e o Algoritmo Euclideano mostra que ela encontrou um número e primo em relação a $(p-1)(q-1)$, então (como na seção 6.6) ele também dá dois inteiros u e v tais que

$$eu - (p-1)(q-1)v = 1.$$

Portanto $eu - 1$ é divisível por $(p-1)(q-1)$. Suponha que d denote o resto de u módulo $(p-1)(q-1)$, então $ed-1$ também é divisível por $(p-1)(q-1)$, e portanto encontramos uma chave privada adequada d.

Finalmente, como calcular o resto de x^e módulo m, quando simplesmente escrever x^e no papel preencheria o universo? Isso é feito da mesma maneira como descrito na seção 6.10.

Assinaturas, etc. Há muitas outras coisas úteis que esse sistema pode fazer. Por exemplo, suponha que Alice recebe uma mensagem de Bob conforme descrito acima. Como ela pode saber que a mensagem de fato veio de Bob? Somente porque ela está assinada "Bob", ela poderia ter vindo de qualquer pessoa. Mas Bob pode fazer o seguinte. Primeiro, ele encripta a mensagem com sua chave privada, então adiciona "Bob", e a encripta novamente com a chave pública de Alice. Quando Alice a recebe, ela pode decriptá-la com sua chave privada. Ela verá uma mensagem ainda encriptada, assinada "Bob". Ela pode cortar fora a assinatura, buscar a chave pública de Bob no catálogo, e usá-la para decriptar a mensagem.

Alguém poderia ter falsificado essa mensagem? Não, porque o araponga teria que usar a chave privada de Bob para encriptar a mensagem (usando qualquer chave significaria que Alice obtém lixo quando ela decripta a mensagem por meio da chave pública de Bob, e ela saberia imediatamente que é uma farsa).

Pode-se usar truques semelhantes para implementar muitos outros dispositivos eletrônicos, usando o sistema de chave pública RSA: autenticação, impressão de marca-d'água, etc.

Segurança. A segurança do protocolo RSA é uma questão difícil, e desde sua concepção em 1977, milhares de pesquisadores a têm investigado. O fato de que nenhum ataque tem sido geralmente bem-sucedido é um bom sinal; mas infelizmente nenhuma prova exata de sua segurança foi encontrada (e parece que a matemática atual carece das ferramentas para prover tal prova).

Podemos dar, entretanto, pelo menos alguns argumentos que suportam sua segurança. Suponha que você intercepte a mensagem de Bob, e deseje decifrá-la. Você conhece o resto r (essa é a mensagem interceptada). Você também conhece a chave pública de Alice e, e o número m. Poder-se-ia pensar em duas linhas de ataque: ou você pode descobrir a chave privada dela d e então decriptar a mensagem tal qual ela faz, ou você poderia até certo ponto mais diretamente encontrar o inteiro x, conhecendo o resto de x^e módulo m.

Infelizmente não há qualquer teorema enunciando que qualquer uma dessas duas é impossível em menos que tempo astronômico. Mas se poderia justificar a segurança do sistema com o seguinte fato: *se se pode quebrar o sistema RSA, então se pode usar o mesmo algoritmo para encontrar os fatores primos de* m (veja o exercício 15.6). O problema da fatoração tem sido estudado por muitas pessoas e nenhum método eficiente foi encontrado, o que torna a segurança do RSA bastante provável.

15.6 Suponha que Bob desenvolve um algoritmo que pode quebrar o RSA da primeira, e mais direta forma descrita acima: sabendo a chave pública de Alice m e e, ele pode encontrar sua chave privada d.

(a) Mostre que ele pode usar isso para encontrar o número $(p-1)(q-1)$;

(b) disso, ele pode encontrar a fatoração prima $m = pq$.

O mundo real. Quão prático poderia ser um sistema complicado como esse? Parece que somente uns poucos matemáticos poderia mesmo usá-lo. Mas na realidade você provavelmente o usou centenas de vezes! RSA é usado em SSL (*Secure Socket Layer*), que por sua vez é usado em http's (http seguro). A qualquer momento que você visita

um “portal seguro” da internet (para ler seu e-mail ou para encomendar uma mercadoria), seu computador gera uma chave pública e uma privada para você, e as usa para tornar seguro que seu número de cartão de crédito e outros dados pessoais permaneçam secretos. Não é preciso envolver você nisso de forma algum—tudo o que você nota é que a conexão é um pouco mais lenta.

Na prática, os dois primos de 100 dígitos não são considerados suficientemente seguros. Aplicações comerciais usam mais que duas vezes esse tamanho, aplicações militares, mais que 4 vezes.

Enquanto que os cálculos cabeludos de elevar o texto pleno x a um exponente que somente ele tem centenas de dígitos são surpreendentemente eficientes, e seria ainda demasiado lento encriptar e decriptar cada mensagem dessa maneira. Uma maneira de evitar isso é enviar, como uma primeira mensagem, a chave a um sistema mais simples de encriptação (pense numa folha-de-uso-único, embora se use um sistema mais eficiente na prática, como DES, o *Digital Encryption Standard*). Essa chave é então usada por alguns minutos para codificar as mensagens indo e voltando, e então jogada fora. A idéia é que numa sessão curta, o número de mensagens codificadas não é suficiente para um araponga quebrar o sistema.