Aritmética Modular

Em algumas situações, nos preocupamos apenas com o resto de um número inteiro quando ele é dividido por um determinado número inteiro positivo. Por exemplo, quando perguntamos que horas serão daqui a 50 horas (em um relógio de 24 horas), nos preocupamos apenas com o resto quando 50 mais a hora atual é dividido por 24. Como estamos interessados apenas nos restos, temos uma notação especial para eles.

Temos uma notação para indicar que dois números inteiros têm o mesmo resto quando eles são divididos pelo número inteiro positivo m.

DEFINIÇÃO 3 Se a e b forem números inteiros e m for um número inteiro positivo, então a é *congruente a* b *módulo* m se m divide $a - b$. Usamos a notação $a \equiv b \pmod{m}$ para indicar que a é congruente a b módulo m. Se a e b não forem congruentes módulo m, escrevemos $a \not\equiv b \pmod{m}$.

A conexão entre as notações usadas quando trabalhamos com restos torna-se clara com o Teorema 3.

TEOREMA 3 Considere a e b como números inteiros e considere m como um número inteiro positivo. Então, $a \equiv b \pmod{m}$ se e somente se $a \bmod m = b \bmod m$.

A demonstração do Teorema 3 será apresentada nos exercícios 11 e 12, no final desta seção.

EXEMPLO 5 Determine se 17 é congruente a 5 módulo 6 e se 24 e 14 são congruentes ao módulo 6.

Solução: Como 6 divide $17 - 5 = 12$, vemos que $17 \equiv 5 \pmod 6$. Entretanto, como $24 - 14 = 10$ não é divisível por 6, temos que $24 \not\equiv 14 \pmod 6$. ◀

O grande matemático alemão Karl Friedrich Gauss desenvolveu o conceito de congruência no final do século XVIII.

A noção de congruência desempenha um importante papel no desenvolvimento da teoria dos números. O Teorema 4 fornece uma maneira útil de trabalhar com congruências.

TEOREMA 4 Considere m como um número inteiro positivo. Os números inteiros a e b são congruentes ao módulo m se e somente se houver um número inteiro k, tal que $a = b + km$.

Demonstração: Se $a \equiv b \pmod m$, então $m \mid (a - b)$. Isto significa que há um número inteiro k tal que $a - b = km$, para que $a = b + km$. Reciprocamente, se houver um número inteiro k, tal que $a = b + km$, então $km = a - b$. Assim, m divide $a - b$, e portanto $a \equiv b$ (mód. m). ◁

O conjunto de todos os números inteiros congruentes a um inteiro a módulo m é chamado de **classe de congruência** de a módulo m. No Capítulo 8 vamos mostrar que há m conjuntos disjuntos de classes de equivalência módulo m e que a união dessas classes de equivalência é o conjunto dos números inteiros.

O Teorema 5 mostra como a congruência funciona com relação à adição e à multiplicação.

TEOREMA 5 Considere m como um número inteiro positivo. Se $a \equiv b \pmod m$ e $c \equiv d \pmod m$, então
$a + c \equiv b + d \pmod m$ e $ac \equiv bd \pmod m$.

Demonstração: Como $a \equiv b \pmod m$ e $c \equiv d \pmod m$, há números inteiros s e t com $b = a + sm$ e $d = c + tm$. Assim,

$$b + d = (a + sm) + (c + tm) = (a + c) + m(s + t)$$

e

$$bd = (a + sm)(c + tm) = ac + m(at + cs + stm).$$

Assim,

$$a + c \equiv b + d \pmod m \quad \text{e} \quad ac \equiv bd \pmod m.$$ ◁

Links

KARL FRIEDRICH GAUSS (1777–1855) Karl Friedrich Gauss, filho de um pedreiro, foi uma criança-prodígio. Ele demonstrou seu potencial com 10 anos de idade, quando rapidamente resolveu um problema dado pelo professor para manter a sala em silêncio. O professor pediu aos estudantes que encontrassem a soma dos primeiros 100 números inteiros positivos. Gauss viu que a soma poderia ser encontrada formando 50 pares, cada um com a soma 101: $1 + 100$, $2 + 99, \ldots, 50 + 51$. Este brilhantismo atraiu o patrocínio de benfeitores, incluindo o Duque Ferdinand de Brunswick, que tornou possível a Gauss freqüentar a Caroline College e a Universidade de Göttingen. Enquanto estudante, ele inventou o método dos mínimos quadrados, que é usado para estimar o valor mais verossímil de uma variável a partir de resultados experimentais. Em 1796, Gauss fez uma descoberta fundamental na geometria, desenvolvendo um assunto que não era discutido desde os tempos remotos. Ele mostrou que um polígono regular com 17 lados poderia ser construído usando-se apenas uma régua e um compasso. Em 1799, Gauss apresentou a primeira regra rigorosa do Teorema Fundamental de Álgebra, que estabelece que um polinômio de grau n tem exatamente n raízes (contando multiplicidades). Ele alcançou fama mundial quando calculou corretamente a órbita do primeiro asteróide descoberto, Ceres, usando dados limitados.

Gauss foi chamado de Príncipe da Matemática pelos seus matemáticos contemporâneos. Embora seja conhecido por suas descobertas em geometria, álgebra, análise, astronomia e física, ele tinha um interesse especial na teoria dos números, o que pode ser visto na declaração: "A matemática é a rainha das ciências, e a teoria dos números, a rainha da matemática". Gauss deixou os fundamentos para a teoria dos números moderna com a publicação de seu livro, *Disquisitiones Arithmeticae,* em 1801.

EXEMPLO 6 Como $7 \equiv 2 \pmod 5$ e $11 \equiv 1 \pmod 5$, temos que, a partir do Teorema 5,

$$18 = 7 + 11 \equiv 2 + 1 = 3 \pmod 5$$

e que

$$77 = 7 \cdot 11 \equiv 2 \cdot 1 = 2 \pmod 5.$$ ◀

Precisaremos do corolário a seguir do Teorema 5 da Seção 4.4.

COROLÁRIO 2 Considere m como um número inteiro positivo e a e b como números inteiros. Então

$$(a + b) \bmod m = ((a \bmod m) + (b \bmod m)) \bmod m$$

e

$$ab \bmod m = ((a \bmod m)(b \bmod m)) \bmod m.$$

Demonstração: Pelas definições de **mod** m e pela definição de congruência módulo m, sabemos que $a \equiv (a \bmod m) \pmod m$ e $b \equiv (b \bmod m) \pmod m$. Assim, o Teorema 5 nos diz que

$$a + b \equiv (a \bmod m) + (b \bmod m) \pmod m$$

e

$$ab \equiv (a \bmod m)(b \bmod m) \pmod m.$$

As equações deste corolário seguem a partir das duas últimas congruências do Teorema 3. ◁

Devemos ter cuidado ao trabalhar com as congruências. Algumas propriedades que pensamos ser verdadeiras não são válidas. Por exemplo, se $ac \equiv bc \pmod m$, a congruência $a \equiv b \pmod m$ pode ser falsa. Da mesma forma, se $a \equiv b \pmod m$ e $c \equiv d \pmod m$, a congruência $a^c \equiv b^d \pmod m$ pode também ser falsa. (Veja o Exercício 23.)

Aplicações das Congruências

A teoria dos números tem aplicações em diversas áreas. Introduziremos três aplicações nesta seção: o uso de congruências para determinar a localização da memória de arquivos de computador, a construção de números pseudo-aleatórios e a codificação de sistemas com base na aritmética modular.

EXEMPLO 7 **Funções de Hashing** O computador central em uma companhia de seguros administra os registros para cada um de seus clientes. Como a localização da memória pode ser determinada para que os registros dos clientes sejam recuperados rapidamente? A solução para este problema é usar uma escolha apropriada: **a função de hashing.** Os registros são identificados usando-se uma **chave,** que identifica unicamente cada registro do cliente. Por exemplo, os registros de cliente são geralmente identificados usando-se um número de Segurança Social do cliente como a chave. Uma função de hashing h determina a localização da memória $h(k)$ para o registro que tem k como sua chave.

Links

Na prática, muitas funções de hashing diferentes são usadas. Uma das mais comuns é a função

$$h(k) = k \bmod m$$

em que m é o número de localizações da memória disponíveis.

As funções de hashing podem ser facilmente avaliadas para que os arquivos possam ser encontrados rapidamente. A função de hashing $h(k) = k$ **mod** m tem esta característica; para encontrar $h(k)$, precisamos apenas computar o resto quando k é dividido por m. Além disso, a função de hashing deve ser sobrejetiva para que todas as localizações de memória sejam utilizadas. A função $h(k) = k$ **mod** m também satisfaz esta propriedade.

Por exemplo, quando $m = 111$, o registro do cliente com o número de Seguro Social 064212848 é projetado para a localização da memória 14, porque

$$h(064212848) = 064212848 \textbf{ mod } 111 = 14.$$

Da mesma forma, como

$$h(037149212) = 037149212 \textbf{ mod } 111 = 65,$$

o registro do cliente com o número de Seguro Social 037149212 é projetado para a localização da memória 65.

Como a função de hashing não é injetora (porque há mais chaves possíveis que localizações da memória), mais de um arquivo pode ser projetado para a localização. Quando isso acontece, dizemos que ocorreu uma **colisão**. Uma maneira de resolver uma colisão é determinar o primeiro local vago seguido daquele já ocupado que foi projetado pela função de hashing. Por exemplo, depois de designar duas tarefas anteriores, determinamos a localização 15 para o registro do cliente com o número de Seguro Social 107405723. Para verificar isto, primeiro note que $h(k)$ mapeia este número para a localização 14, porque

$$h(107405723) = 107405723 \textbf{ mod } 111 = 14,$$

mas este local já está ocupado (pelo arquivo do cliente com o número de Seguro Social 064212848). Entretanto, a localização 15, o primeiro local subseqüente à localização 14, está livre.

Há muitas maneiras mais sofisticadas para resolver colisões que são mais eficientes que este simples método que descrevemos. Elas serão discutidas nas referências às funções de hashing dadas no final deste livro. ◀

EXEMPLO 8 **Números Pseudo-aleatórios** Números escolhidos aleatoriamente são geralmente necessários em simulações de computação. Diferentes métodos foram desenvolvidos para gerar números que têm propriedades de escolher aleatoriamente números. Como os números gerados por métodos sistêmicos não são verdadeiramente aleatórios, eles são chamados de **números pseudo-aleatórios.**

Links

O procedimento mais comum usado para gerar esses números é o **método congruente linear.** Escolhemos quatro números inteiros: o **módulo** m, o **multiplicador** a, o **acréscimo** c e a **origem** x_0, com $2 \leq a < m$, $0 \leq c < m$ e $0 \leq x_0 < m$. Construímos uma seqüência de números pseudo-aleatórios $\{x_n\}$, com $0 \leq x_n < m$ para todo n, usando sucessivamente a congruência

$$x_{n+1} = (ax_n + c) \textbf{ mod } m.$$

(Este é um exemplo de definição recursiva, a ser discutida na Seção 4.3, quando mostraremos que tais seqüências são bem definidas.)

Muitas experiências com computadores necessitam de números pseudo-aleatórios entre 0 e 1. Para gerá-los, dividimos os números gerados com um gerador de congruência linear pelo módulo, ou seja, usamos os números x_n/m.

Por exemplo, a seqüência de números pseudo-aleatórios gerados escolhendo-se $m = 9$, $a = 7$, $c = 4$ e $x_0 = 3$, pode ser encontrada a seguir:

$$x_1 = 7x_0 + 4 \bmod 9 = 7 \cdot 3 + 4 \bmod 9 = 25 \bmod 9 = 7$$
$$x_2 = 7x_1 + 4 \bmod 9 = 7 \cdot 7 + 4 \bmod 9 = 53 \bmod 9 = 8$$
$$x_3 = 7x_2 + 4 \bmod 9 = 7 \cdot 8 + 4 \bmod 9 = 60 \bmod 9 = 6$$
$$x_4 = 7x_3 + 4 \bmod 9 = 7 \cdot 6 + 4 \bmod 9 = 46 \bmod 9 = 1$$
$$x_5 = 7x_4 + 4 \bmod 9 = 7 \cdot 1 + 4 \bmod 9 = 11 \bmod 9 = 2$$
$$x_6 = 7x_5 + 4 \bmod 9 = 7 \cdot 2 + 4 \bmod 9 = 18 \bmod 9 = 0$$
$$x_7 = 7x_6 + 4 \bmod 9 = 7 \cdot 0 + 4 \bmod 9 = 4 \bmod 9 = 4$$
$$x_8 = 7x_7 + 4 \bmod 9 = 7 \cdot 4 + 4 \bmod 9 = 32 \bmod 9 = 5$$
$$x_9 = 7x_8 + 4 \bmod 9 = 7 \cdot 5 + 4 \bmod 9 = 39 \bmod 9 = 3$$

Como $x_9 = x_0$ e cada termo depende apenas de seu termo anterior, a seqüência abaixo é gerada:

3, 7, 8, 6, 1, 2, 0, 4, 5, 3, 7, 8, 6, 1, 2, 0, 4, 5, 3, ...

Esta seqüência contém nove números diferentes antes de qualquer repetição.

A maioria dos computadores usa o gerador de congruência linear para gerar números pseudo-aleatórios. Geralmente, um gerador de congruência linear com acréscimo $c = 0$ é usado. Esse gerador é chamado de **gerador multiplicativo puro.** Por exemplo, o gerador multiplicativo puro com módulo $2^{31} - 1$ e multiplicador $7^5 = 16\,807$ é muito usado. Com esses valores, podemos mostrar que $2^{31} - 2$ números são gerados antes de qualquer repetição. ◀

Criptologia

As congruências têm muitas aplicações em matemática discreta e ciência da computação. Discussões sobre essas aplicações podem ser encontradas nas leituras sugeridas no final do livro. Uma das aplicações mais importantes da congruência envolve a **criptologia,** que é o estudo das mensagens secretas. Um dos usos mais antigos conhecido da criptologia foi com Júlio César. Ele fez mensagens secretas substituindo cada letra pelas três letras à frente no alfabeto (sendo as últimas três letras do alfabeto substituídas pelas três primeiras). Por exemplo, usando este esquema, a letra *B* é enviada como *E* e a letra *X*, como *A*. Este é um exemplo de **codificação,** ou seja, o processo de montar uma mensagem secreta.

Para expressar matematicamente o processo de codificação de César, primeiro substitua cada letra por um número inteiro de 0 a 25, com base em sua posição no alfabeto. Por exemplo, substitua *A* por 0, *K* por 10 e *Z* por 25. O método de codificação de César pode ser representado pela função f, que projeta para o número inteiro não negativo p, $p \leq 25$, o inteiro $f(p)$ do conjunto $\{0, 1, 2, \ldots, 25\}$ com

$$f(p) = (p + 3) \bmod 26.$$

Na versão codificada da mensagem, a letra representada por p é substituída pela letra representada por $(p + 3) \bmod 26$.

EXEMPLO 9 Qual é a mensagem secreta produzida a partir da mensagem "MEET YOU IN THE PARK" usando a criptografia César?

Solução: Primeiro troque as letras da mensagem por números. Isto produz

12 4 4 19 24 14 20 8 13 19 7 4 15 0 17 10.

Agora substitua cada um desses números p por $f(p) = (p + 3)$ **mod** 26. Isto equivale a

15 7 7 22 1 17 23 11 16 22 10 7 18 3 20 13.

Transcrevendo de volta às letras, temos a mensagem codificada "PHHW BRX LQ WKH SDUN." ◀

Para recuperar a mensagem original a partir de uma mensagem secreta codificada pelo código de César, a função f^{-1}, a inversa de f, é usada. Note que a função f^{-1} projeta um número inteiro p de $\{0, 1, 2, \ldots, 25\}$ para $f^{-1}(p) = (p - 3)$ **mod** 26. Em outras palavras, para encontrar a mensagem original, cada letra é substituída pelas três letras anteriores no alfabeto, com as primeiras três letras referindo-se às três últimas letras do alfabeto. O processo de determinar a mensagem original a partir da mensagem codificada é chamado de **decodificação.**

Há várias maneiras de construir o código de César. Por exemplo, em vez de substituir cada letra por uma que esteja 3 posições à frente, podemos substituir cada letra por k letras à frente, de tal forma que

$$f(p) = (p + k) \textbf{ mod } 26.$$

Esse código é chamado de **código de substituição.** Note que a decodificação pode ser realizada usando-se

$$f^{-1}(p) = (p - k) \textbf{ mod } 26.$$

É óbvio que o método de César e os códigos de substituição não fornecem um alto nível de segurança. Há várias maneiras de melhorar este método. Uma aproximação que aumenta um pouco a segurança é usar a função de forma

$$f(p) = (ap + b) \textbf{ mod } 26,$$

em que a e b são números inteiros, escolhidos para que f seja uma bijetora. (Tal função é chamada de *transformação afim*.) Isto fornece diversos sistemas de codificações possíveis. O uso de um desses sistemas é ilustrado no Exemplo 10.

EXEMPLO 10 Qual letra substitui a letra K quando a função $f(p) = (7p + 3)$ **mod** 26 é usada para codificação?

Solução: Primeiro, note que 10 representa K. Então, usando a função de codificação específica, temos que $f(10) = (7 \cdot 10 + 3)$ **mod** $26 = 21$. Como 21 representa V, K é substituída por V na mensagem codificada. ◀

Links

O método de codificação de César e sua construção funcionam pela substituição de cada letra do alfabeto por outra. Métodos de codificação desse tipo são vulneráveis a ataques baseados na freqüência de ocorrência das letras na mensagem. Métodos de codificação mais sofisticados são baseados na substituição de blocos de letras por outros blocos de letras. Há várias técnicas baseadas em aritmética modular para codificar blocos de letras. Outras discussões sobre o assunto podem ser encontradas nas leituras sugeridas no fim deste livro.

Criptografia de Chave Pública

Na Seção 3.4, introduzimos métodos para codificar mensagens com base em congruências. Quando esses métodos de codificação são usados, as mensagens, que são seqüências de caracteres, são transcritas em números. Então, o número para cada caractere é transformado em outro número, se usarmos uma substituição ou uma transformação afim módulo 26. Esses métodos são exemplos de **criptossistemas de chave privada.** Conhecer a chave de codificação permite que encontremos rapidamente a chave de decodificação. Por exemplo, quando um código de substituição é usado com chave de codificação k, um número p representando uma letra é enviado a

$$c = (p + k) \bmod 26.$$

A decodificação é realizada pela substituição de $-k$; ou seja,

$$p = (c - k) \bmod 26.$$

Quando um criptossistema de chave privada é usado, um par de pessoas que desejam se comunicar em segredo precisa ter uma chave separada. Como qualquer um que conheça essa chave pode facilmente codificar e decodificar as mensagens, essas duas pessoas precisam trocar a chave para segurança.

Em meados da década de 70, criptologistas introduziram o conceito de **criptossistemas de chave pública.** Quando tais criptossistemas são usados, saber como enviar a alguém uma mensagem não ajuda na decodificação de mensagens enviadas a essa pessoa. Em tal sistema, toda pessoa pode ter uma chave pública. Apenas as chaves de decodificação são mantidas em segredo e apenas o destinatário da mensagem pode decodificá-la, porque a chave de codificação não permite que alguém encontre a chave de decodificação sem ter um trabalho muito grande (algo como 2 bilhões de anos em tempo de computação).

O Criptossistema RSA

Em 1976, três pesquisadores do M.I.T. — Ronald Rivest, Adi Shamir e Leonard Adleman — apresentaram ao mundo um criptossistema de chave pública, conhecido como **sistema RSA,** a partir das iniciais de seus inventores. (Como geralmente acontece com as descobertas criptográficas, o sistema RSA já tinha sido descoberto em uma pesquisa secreta do governo. Clifford Cocks, trabalhando secretamente para o governo do Reino Unido, descobriu este criptossistema em 1973, mas sua invenção ficou desconhecida até o final da década de 90. Um excelente material sobre essa antiga descoberta, assim como o trabalho de Rivest, Shamir e Adleman, pode ser encontrado em [Si99].) O criptossistema RSA é baseado em potenciação modular do produto de dois números primos grandes, o que pode ser feito rapidamente usando o Algoritmo 5 da Seção 3.6. Cada indivíduo tem uma chave de codificação que consiste em um módulo $n = pq$, em que p e q são números primos grandes, com 200 dígitos cada, e um expoente e que é relativamente primo a $(p - 1)(q - 1)$. Para produzir uma chave que possa ser usada na decodificação, dois números primos grandes devem ser encontrados. Isto pode ser feito rapidamente em um computador usando-se os testes de probabilidade de primalidade, mencionados anteriormente nesta seção. Entretanto, o produto desses números primos $n = pq$, com aproximadamente 400 dígitos, não pode ser fatorado em uma quantidade razoável de tempo. Como veremos, esta é uma razão importante do porquê a decodificação não pode ser feita rapidamente sem uma chave de decodificação separada.

Codificação RSA

No método de codificação RSA, as mensagens são transcritas em seqüências de números inteiros. Isto pode ser feito transcrevendo cada letra em um número inteiro, como foi feito com o código de César. Esses números inteiros são agrupados na forma de inteiros grandes, cada um representando um bloco de letras. A codificação é feita transformando o número inteiro M, que representa o texto total (a mensagem original), em um número inteiro C, que representa o texto codificado (a mensagem codificada), usando-se a função

$$C = M^e \bmod n.$$

(Para realizar a codificação, usamos um algoritmo para potenciação modular rápida, assim como o Algoritmo 5 da Seção 3.6.) Deixamos a mensagem codificada como blocos de números e os enviamos ao devido destinatário.

O Exemplo 11 ilustra como a codificação RSA é realizada. Por motivos de ordem prática, usamos números primos pequenos p e q no exemplo, em vez de primos com 100 dígitos ou mais. Embora o código descrito neste exemplo não seja seguro, ele ilustra as técnicas usadas no código RSA.

EXEMPLO 11 Codifique a mensagem STOP usando o criptossistema RSA com $p = 43$ e $q = 59$, para que $n = 43 \cdot 59 = 2537$, com $e = 13$. Note que

$$\text{mdc}(e, (p-1)(q-1)) = \text{mdc}(13, 42 \cdot 58) = 1.$$

Solução: Transcrevemos as letras de STOP em equivalentes numéricos e então agrupamos os números em blocos de quatro. Obtemos

1819 1415.

Links

RONALD RIVEST (Nascido em 1948) Ronald Rivest recebeu seu doutorado em Yale, em 1969, e seu Ph.D. em ciência da computação em Stanford, em 1974. Ele é professor de ciência da computação no M.I.T. e foi um dos fundadores da Segurança de Dados RSA, que mantém a patente do criptossistema RSA que ele inventou juntamente com Adi Shamir e Leonard Adleman. As áreas em que Rivest tem trabalhado, além da criptografia, incluem inteligência artificial, design de VLSI e algoritmos. Ele é co-autor de um texto popular sobre algoritmos ([CoLeRiSt01]).

ADI SHAMIR (Nascido em 1952) Adi Shamir nasceu em Tel Aviv, em Israel. Ele obteve seu título de bacharel pela Universidade de Tel Aviv (1972) e seu Ph.D. pelo Weizmann Institute of Science (1977). Shamir foi pesquisador-assistente na Universidade de Warwick e professor-assistente no M.I.T. Atualmente, é professor do Departamento de Matemática Aplicada do Weizmann Institute e lidera um grupo de estudo sobre segurança computacional. As contribuições de Shamir à criptografia, além do criptossistema RSA, incluem criptossistemas de craqueamento de pacotes, análise criptográfica do Data Encryption Standard (DES) e a criação de muitos protocolos de criptografia.

LEONARD ADLEMAN (Nascido em 1945) Leonard Adleman nasceu em São Francisco, Califórnia. Recebeu seu título de bacharel em matemática (1968) e seu Ph.D. em ciência da computação (1976) pela Universidade da Califórnia, em Berkeley. Adleman foi membro da faculdade de matemática do M.I.T. de 1976 a 1980, onde foi co-inventor do criptossistema RSA, e em 1980 conquistou um cargo no departamento de ciência da computação na University of Southern California (USC). Ele foi indicado a uma cadeira na USC em 1985. Adleman trabalha com segurança computacional, complexidade computacional, imunologia e biologia molecular. Ele inventou o termo "vírus de computador". O trabalho recente de Adleman em computar o DNA tem despertado grande interesse. Ele foi consultor técnico do filme *Quebra de sigilo*, no qual a segurança computacional desenvolve um importante papel.

Codificamos cada bloco usando a função

$$C = M^{13} \bmod 2537.$$

Computações que usam multiplicações modulares mostram que $1819^{13} \bmod 2537 = 2081$ e $1415^{13} \bmod 2537 = 2182$. A mensagem codificada é 2081 2182. ◀

Decodificação RSA

A mensagem de texto puro pode ser rapidamente recuperada quando a chave de decodificação d, um inverso de e módulo $(p-1)(q-1)$, é conhecida. [Tal que um inverso exista, já que mdc(e, $(p-1)(q-1)) = 1$.] Para verificar isso, note que se $de \equiv 1 \pmod{(p-1)(q-1)}$, há um número inteiro k, tal que $de = 1 + k(p-1)(q-1)$. Portanto,

$$C^d \equiv (M^e)^d = M^{de} = M^{1+k(p-1)(q-1)} \pmod{n}.$$

Pelo Pequeno Teorema de Fermat [supondo que mdc(M, p) = mdc(M, q) = 1, que é mantido, com exceção de casos raros], temos que $M^{p-1} \equiv 1 \pmod{p}$ e $M^{q-1} \equiv 1 \pmod{q}$. Conseqüentemente,

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

e

$$C^d \equiv M \cdot (M^{q-1})^{k(p-1)} \equiv M \cdot 1 \equiv M \pmod{q}.$$

Como mdc(p, q) = 1, temos que, pelo Teorema Chinês do Resto,

$$C^d \equiv M \pmod{pq}.$$

O Exemplo 12 ilustra como decodificar mensagens enviadas usando o criptossistema RSA.

EXEMPLO 12 Recebemos a mensagem codificada 0981 0461. Qual é a mensagem decodificada se ela tiver sido codificada usando-se o código RSA do Exemplo 11?

Solução: A mensagem foi codificada usando-se o criptossistema RSA com $n = 43 \cdot 59$ e expoente 13. Como mostra o Exercício 4, $d = 937$ é um inverso de 13 módulo $42 \cdot 58 = 2436$. Usamos 937 como nosso expoente de decodificação. Conseqüentemente, para decodificar um bloco C, computamos

$$P = C^{937} \bmod 2537.$$

Para decodificar a mensagem, usamos um algoritmo de potenciação modular rápido para computar $0981^{937} \bmod 2537 = 0704$ e $0461^{937} \bmod 2537 = 1115$. Conseqüentemente, a versão numérica da mensagem original é 0704 1115. Traduzindo de volta às letras, vemos que a mensagem é HELP. ◀

RSA como um Sistema de Chave Pública

Links

Por que o criptossistema RSA é adequado para a criptografia de chave pública? Quando sabemos a fatoração do módulo n, ou seja, quando conhecemos p e q, podemos usar o algoritmo de Euclides para encontrar rapidamente um expoente d inverso a e módulo $(p-1)(q-1)$. Isto nos permite decodificar mensagens enviadas usando nossa chave. Entretanto, não há método conhecido para decodificar mensagens que não seja baseado em encontrar uma fatoração de n, ou então fugir

da fatoração de n. Acredita-se que a fatoração seja um problema difícil, oposto a encontrar os primos grandes p e q, o que pode ser feito rapidamente. O método de fatoração mais eficiente conhecido (como de 2005) requer bilhões de anos para fatorar números inteiros com 400 dígitos. Conseqüentemente, quando p e q são números primos com 200 dígitos, as mensagens codificadas usando-se $n = pq$ como módulo não podem ser encontradas em um tempo razoável, a menos que os números primos p e q sejam conhecidos.

A pesquisa ativa está a caminho de encontrar novas maneiras para tornar mais eficiente a fatoração de números inteiros. Aqueles que eram considerados, recentemente e muitos anos atrás, muito grandes para serem fatorados em um tempo razoável, agora podem ser fatorados rapidamente. Números inteiros com mais de 100 dígitos, assim como alguns com mais de 150 dígitos, têm sido fatorados em um trabalho em equipe. Quando novas técnicas de fatoração forem descobertas, será necessário usar números primos maiores para assegurar o segredo das mensagens. Infelizmente, as mensagens que foram consideradas seguras anteriormente podem ser salvas e decodificadas por vários destinatários quando o fator de $n = pq$ torna-se acessível na chave usada pela codificação RSA.

O método RSA é agora mundialmente utilizado. Entretanto, os criptossistemas mais usados são os de chave privada. O uso da criptografia de chave pública, via sistema RSA, está crescendo. Entretanto, há aplicações que usam sistemas de chaves privadas e públicas. Por exemplo, um criptossistema de chave pública, como o RSA, pode ser usado para distribuir chaves privadas a pares de indivíduos quando eles desejam se comunicar. Essas pessoas usam, então, um sistema de chave privada para codificar e decodificar mensagens.