Teoria dos Números e Sistemas Criptográficos

Cid C. de Souza – IC-UNICAMP

21 de fevereiro de 2005

Sistemas Criptográficos de Chave Pública (Seção 31.7)

▷ O sistema pode ser usado para:

- Envio de mensagens *cifradas* que não podem ser entendidas por alguém que esteja "ouvindo" a rede.
- Permite a um participante criar uma *assinatura digital* para incluir em seus documentos.
- A *assinatura* de qualquer participante pode ser facilmente verificada por todos mas não pode ser *forjada* por ninguém.

▷ Todo participante X do sistema tem duas chaves: a **pública** (P_X) e a **secreta** (S_X).

▷ A chave pública de X é conhecida por todos participantes e a sua chave secreta só é conhecida por ele mesmo. Pode-se assumir que as chaves públicas estão todas disponibilizadas em um repositório de domínio público !

Sistemas Criptográficos de Chave Pública (cont.)

- ▷ Seja $\mathcal{D}$ o conjunto de todas possíveis mensagens enviadas por um participante (p. ex., o conjunto de todas seqüências finitas de *bits*).
- ▷ As chaves pública e secreta estão associadas a funções *bijetoras* que podem ser aplicadas a qualquer mensagem de $\mathcal{D}$.
- ▷ A função pública do participante X $(\mathcal{P}_X())$ e a sua função secreta $(\mathcal{P}_X())$ definem permutações em (D) e devem ser **facilmente computáveis** a partir das chaves P_X e S_X, respectivamente.

  As funções pública e secreta do participante X são uma a inversa da outra !

- ▷ Assim, para todo $M \in \mathcal{D}$, tem-se:

$$M = \mathcal{S}_X(\mathcal{P}_X(M)) \qquad (31.37)$$

$$M = \mathcal{P}_X(\mathcal{S}_X(M)) \qquad (31.38)$$

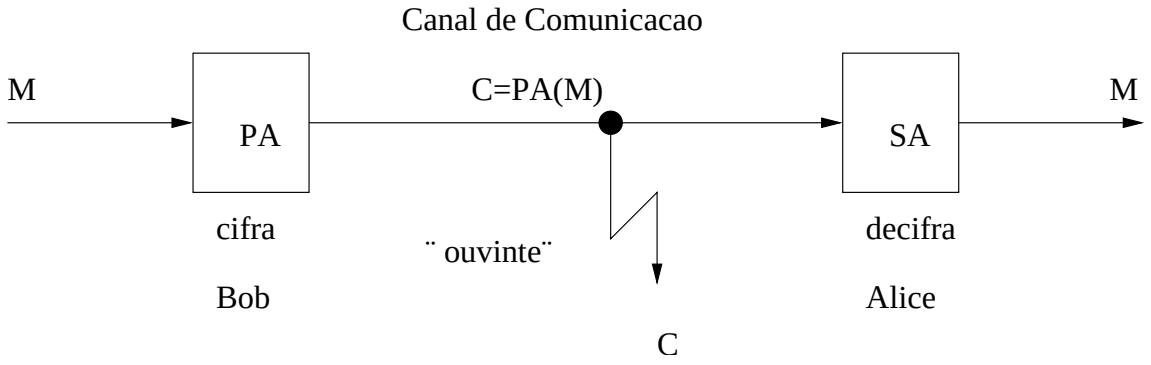
Sistemas Criptográficos de Chave Pública (cont.)

- ▷ Em um sistema de chave pública, é essencial que ninguém exceto o participante X seja capaz de computar a função $\mathcal{S}_X()$ em um tempo computacional aceitável.
- ▷ Esta hipótese deve permanecer válida mesmo considerando que **todos** os demais participantes conhecem P_X (a chave pública de X) e possam computar a função $\mathcal{P}_X()$ eficientemente.

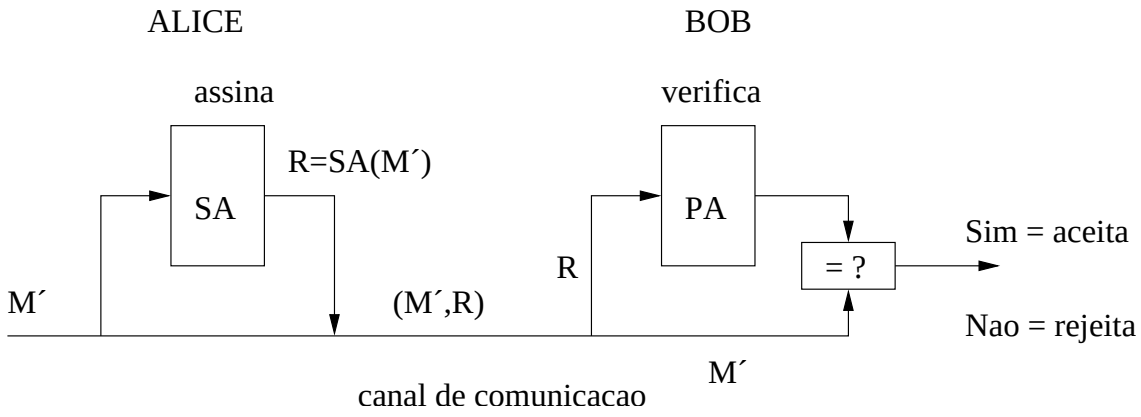
A grande dificuldade de projetar um criptosistema de chave pública que funcione é encontrar como criar um sistema no qual se pode divulgar a transformação $\mathcal{P}_X()$ sem contudo revelar como se computa a função inversa $\mathcal{S}_X()$.

Sistemas Criptográficos de Chave Pública (cont.)

▷ Esquema para cifrar mensagem de *Bob* para *Alice*:



▷ Esquema para uso de assinatura digital de *Alice* para *Bob*:



Teoria dos Números: Introdução (Capítulo 31)

- ▷ O interesse pela Teoria dos Números em Computação cresceu com o surgimento de **sistemas criptográficos** baseados em **grandes números primos**.
  - ◦ Viabilidade = capacidade de gerar grandes números primos;
  - ◦ Segurança = "incapacidade" de fatorar o produto de grandes números primos;
- ▷ **Importante:** a complexidade medida pelo número de operações elementares só é válida enquanto os resultados destas operações têm a mesma magnitude dos números dados na entrada.
- ▷ Para grandes números, a complexidade de uma operação é função do número de *bits* dos operandos (complexidade = # **bit operations**).
- ▷ Para inteiros de β bits, a multiplicação, a divisão e resto da divisão podem ser feitos em $\Theta(\beta^2)$.

Noções Elementares de Teoria dos Números (Seção 31.1)

- ▷ $\mathbb{N}$ e $\mathbb{Z}$: conjunto dos números naturais e inteiros respectivamente.
- ▷ d *divisor* de a : $d \mid a \Rightarrow a = kd$ para algum $k \in \mathbb{Z}$.
- ▷ $d \mid a$ então a é *múltiplo* de d;
- ▷ *Todo inteiro divide 0 (zero)*;
- ▷ $d \mid a \Leftrightarrow (-d) \mid a$; (usaremos só os divisores maiores que 0);
- ▷ Todo inteiro a é divisível por 1 e por ele mesmo. Estes dois divisores são ditos *triviais* e os demais são os *fatores* de a.
- ▷ **Definição:** um inteiro a é *primo* se $a > 1$ e a só contém divisores triviais.
- ▷ **Definição:** um inteiro a é *composto* se $a > 0$ e a tem um fator.

Noções Elementares de Teoria dos Números (cont.)

▷ **Módulos e Classes de Equivalência de $\mathbb{Z}$:**

Teorema da Divisão (31.1):
Para todo a inteiro e n inteiro positivo, existe um único par de inteiros (q, r), onde $0 \leq r < n$, tal que $a = qn + r$.

▷ $q = \lfloor \frac{a}{n} \rfloor$ é o *quociente* e $r = a \bmod n$ é o *resto* da divisão.

▷ **Definição**: equivalência módulo n:

$$a \equiv b \pmod{n} \Rightarrow (a \bmod n) = (b \bmod n)$$

▷ *Exemplo*: $-18 \equiv 47 \equiv 2 \pmod{5}$.

▷ *Idéia*: particionar $\mathbb{Z}$ em múltiplos e não-múltiplos de n.

Noções Elementares de Teoria dos Números (cont.)

▷ **Classes de Equivalência módulo** n: $[a]_n = \{a + kn : k \in \mathbb{Z}\}$.

- *Exemplo*:
  $[5]_7 = \{\ldots, -9, -2, 5, 12, 19, 26, \ldots\} = [-2]_7 = [33]_7$,
  $[-1]_n = [n-1]_n, \ldots$
- $\mathbb{Z}_n = \{[a]_n : 0 \leq a \leq n-1\}$ $\qquad (\ldots = \{0, 1, \ldots, n-1\})$

▷ **Divisores Comuns e Maior Divisor Comum** (`mdc`):

- $d \mid a$ e $d \mid b$ então d é *divisor comum* de a e b.
- $\texttt{mdc}(a, b) \doteq \max\{d \ : \ d \mid a \text{ e } d \mid b\}$.
- $\texttt{mdc}(0, 0) \doteq 0$ $\qquad$ (**definição !**).

Noções Elementares de Teoria dos Números (cont.)

▷ Fatos:

- $d \mid a$ e $d \mid b$ então $d \mid (a+b)$ e $d \mid (a-b)$; (31.5)
- *mais geral*: se $d \mid a$ e $d \mid b$ então $d \mid (ax+by),\ \forall\ x, y \in \mathbb{Z}$; (31.6)
- se $a \mid b$ então $|a| \leq |b|$ ou $b = 0$, logo:
  $a \mid b$ e $b \mid a \Rightarrow a = b$ ou $a = -b$. (31.7)

▷ Propriedades do `mdc`:

- $\texttt{mdc}(a, b) = \texttt{mdc}(b, a)$ (31.8)
- $\texttt{mdc}(a, b) = \texttt{mdc}(-a, b)$ (31.9)
- $\texttt{mdc}(a, b) = \texttt{mdc}(|a|, |b|)$ (31.10)
- $\texttt{mdc}(a, 0) = |a|$ (31.11)
- $\texttt{mdc}(a, ka) = |a|$ para todo $k \in \mathbb{Z}$ (31.12)

Noções Elementares de Teoria dos Números (cont.)

- ▷ **Teorema 31.2 (caracterização do `mdc`)**: sejam a e b dois inteiros não simultaneamente nulos. Então, $d = \texttt{mdc}(a, b)$ é dado por: $\min\{ax + by\}$ com $ax + by$ positivo e $x, y \in \mathbb{Z}$.
- ▷ **Corolário 31.3**: a e b inteiros, $d \mid a$ e $d \mid b \Rightarrow d \mid mdc(a, b)$.
- ▷ **Corolário 31.4**: a e b inteiros,
  $n \geq 0$ inteiro $\Rightarrow \texttt{mdc}(an, bn) = n \times \texttt{mdc}(a, b)$.
- ▷ **Corolário 31.5**: Sejam n, a e b três inteiros **positivos**. Se $n \mid ab$ e $\texttt{mdc}(a, n) = 1$ então $n \mid b$.

Noções Elementares de Teoria dos Números (cont.)

Definição: a e b são *primos relativos* se $\texttt{mdc}(a, b) = 1$.

Exemplo: 15 e 34.

Teorema 31.6: Sejam a, b e p inteiros. Se $\texttt{mdc}(a, p) = 1$ e $\texttt{mdc}(b, p) = 1$ então $\texttt{mdc}(ab, p) = 1$.

Teorema 31.7: para **primo** p e todos inteiros a e b, se $p \mid ab$ então $p \mid a$ ou $p \mid b$.

Teorema 31.8 (Unicidade da Fatoração):
um inteiro a pode ser escrito de maneira única como um produto da forma $a = p_1^{e_1} p_2^{e_2} p_3^{e_3} \ldots p_r^{e_r}$, onde todo p_i é primo e satisfaz $p_i < p_{i+1}$.

Exemplo: $2940 = 2^2.3^1.5^1.7^2$

Algoritmo de Euclides (Seção 31.2)

▷ **Nota:** o $\texttt{mdc}(a, b)$ será calculado para a e b não-negativos já que $\texttt{mdc}(a, b) = \texttt{mdc}(|a|, |b|)$.

▷ *Alternativa*: fatorar a e b em mnúmeros primos

- $a = p_1^{e_1} p_2^{e_2} p_3^{e_3} \ldots p_r^{e_r}$;
- $b = p_1^{f_1} p_2^{f_2} p_3^{f_3} \ldots p_r^{f_r}$;
- $\texttt{mdc}(a, b) = p_1^{\min\{e_1, f_1\}} p_2^{\min\{e_2, f_2\}} \ldots p_r^{\min\{e_r, f_r\}}$;

▷ *Dificuldade:* algoritmos conhecidos para fatoração em números primos não rodam em tempo polinomial.

Teorema 31.9 (recursão): Para todo par de inteiros $a \geq 0$ e $b > 0$, $\texttt{mdc}(a, b) = \texttt{mdc}(b, a \texttt{ mod } b)$.

Algoritmo de Euclides (cont.)

EUCLIDES(a, b)

Entrada: inteiros a e b maiores ou iguais a zero.

Saída: `mdc`(a,b)

Se $b = 0$ **então Retorne** a.

Se não EUCLIDES$(b, a$ `mod` $b)$.

FIM

▷ *Exemplo*:

`EUCLIDES`(2940,126) $\Rightarrow$ `EUCLIDES`(126,42) $\Rightarrow$ `EUCLIDES`(42,0)=42.

▷ *Corretude*:

- Teorema 31.9 e equação (31.11).
- como $b > a$ `mod` b o segundo parâmetro diminui estritamente ao londo das iterações e o algoritmo pára !

Algoritmo de Euclides (cont.)

▷ *Análise de Complexidade:*
a complexidade de pior caso será dada em função dos tamanhos de a e b. Vamos assumir que $a > b \geq 0$.

- *O que acontece se* $b > a \geq 0$ *? E se* $b = a$ *?*

▷ O tempo de execução é proporcional ao número de chamadas recursivas feitas pelo algoritmo a qual está relacionada com ... **os números de Fibonacci !!!**

- $F_0 = 0$, $F_1 = 1$, $F_i = F_{i-1} + F_{i-2}$ quando $i \geq 2$.
- Seqüência de Fibonacci: $\{0, 1, 1, 2, 3, 5, 8, 13, \ldots\}$.
- *razão áurea*: $\phi = (1 + \sqrt{5})/2 = 1.618\ldots$ (conjugada $\hat{\phi} = (1 - \sqrt{5})/2 = -.618\ldots$).
- $F_i = (\phi^i - \hat{\phi}^i)/\sqrt{5}$.

Algoritmo de Euclides (cont.)

Lema 31.10: Se $a > b \geq 0$ e EUCLIDES(a, b) executa $k \geq 1$ chamadas recursivas, então $a \geq F_{k+2}$ e $b \geq F_{k+1}$.

Teorema 3.11 (Teorema de Lamé): Para todo $k \geq 1$, se $a > b \geq 0$ e $b < F_{k+1}$, então EUCLIDES(a, b) faz menos que k chamadas recursivas.

- O limite superior dado no Teorema 31.11 é o melhor possível.
- *Pior caso*: a e b são números de Fibonacci consecutivos. EUCLIDES(F_{k+1}, F_k) faz $k - 1$ recursões (notar que, pela definição de F_k e pelo Teorema da divisão, F_{k+1} `mod` $F_k = F_{k-1}$!).
- número de chamadas recursivas do EUCLIDES é $O(\log b)$.
- se a e b são inteiros de β bits, EUCLIDES faz $O(\beta)$ operações aritméticas (mod) e $O(\beta^3)$ *bit-operations*.

Algoritmo de Euclides Estendido

- $d = \texttt{mdc}(a, b) = ax + by$. *Pergunta*: quem são x e y ?
- Estes coeficientes serão importantes para o cálculo de *inversos multiplicativos* módulo n.
- Pelo Teorema 31.9, se $d = \texttt{mdc}(a, b)$ e $d' = \texttt{mdc}(b, a \texttt{ mod } b)$ então $d = d'$.
- Pelo Teorema 31.2, existem inteiros x, y, x' e y' tais que

$$d = ax + by \quad (1)$$

$$d' = bx' + (a \texttt{ mod } b)y' \quad (2)$$

- Pelo Teorema da divisão (33.1), sabemos que $(a \texttt{ mod } b) = a - \lfloor \frac{a}{b} \rfloor b$. Substituindo-se em (2) têm-se:

$$d' = ay' + b(x' - \lfloor \frac{a}{b} \rfloor y') = d$$

Logo, em (1) escolhemos $x = y'$ e $y = x' - \lfloor \frac{a}{b} \rfloor y'$.

Algoritmo de Euclides Estendido

X-EUCLIDES(a, b)

Entrada: inteiros não negativos a e b.

Saída: `mdc`(a,b) e x e y tais que `mdc`$(a, b) = ax + by$

Se $b = 0$ **então Retorne** $(a, 1, 0)$.

$(d', x', y') \leftarrow$ X-EUCLIDES$(b, a$ `mod` $b)$.

$(d, x, y) \leftarrow (d', y', x' - \lfloor \frac{a}{b} \rfloor y')$

Retorne (d, x, y).

FIM

Exemplo:
X-EUCLIDES(99,78):

a	b	$\lfloor \frac{a}{b} \rfloor$	d	x	y
99	78	1	3	-11	14
78	21	3	3	3	-11
21	15	1	3	-2	3
15	6	2	3	1	-2
6	3	2	3	0	1
3	0	–	3	1	0

Aritmética Modular (Seção 31.3)

▷ *Definição:* Seja S um conjunto e $\oplus$ uma operação binária sobre S. $(S, \oplus)$ é um **grupo** se:

1. **Fecho:** a e $b \in S$ então $a \oplus b \in S$.
2. **Identidade:** $\exists e \in S$ tal que $\forall a \in S$, $a \oplus e = e \oplus a = a$.
3. **Associatividade:** $\forall a, b$ e $c \in S$, $(a \oplus b) \oplus c = a \oplus (b \oplus c)$.
4. **Inverso:** $\forall a \in S$, existe um **único** elemento $b \in S$ tal que $a \oplus b = b \oplus a = e$.

▷ *Exemplo:* $(\mathbb{Z}, +)$ tendo 0 como identidade e $(-a)$ como inverso de $a, \forall a \in \mathbb{Z}$.

▷ *Definição:* um grupo $(S, \oplus)$ é dito ser **abeliano** se a operação $\oplus$ for comutativa ($a \oplus b = b \oplus a$ para todo a e b em S).

▷ *Definição:* um grupo $(S, \oplus)$ é **finito** se $|S| < \infty$.

Aritmética Modular (cont.)

▷ Grupos definidos pela adição e multiplicação modular ...

▷ *Propriedades:* sejam $a \equiv a' \pmod{n}$ e $b \equiv b' \pmod{n}$

- $a + b \equiv a' + b' \pmod{n}$.
- $ab \equiv a'b' \pmod{n}$.

▷ *Definição:* adição e multiplicação módulo n:

- $[a]_n +_n [b]_n = [a+b]_n$.
- $[a]_n \cdot_n [b]_n = [ab]_n$.

▷ *Definição:* o **grupo aditivo módulo** n é dado por $(\mathbb{Z}_n, +_n)$ (0 é a identidade para este grupo).

▷ *Definição:* o **grupo multiplicativo módulo** n é dado por $(\mathbb{Z}_n^*, \cdot_n)$, onde $\mathbb{Z}_n^*$ é o conjunto de elementos de $\mathbb{Z}_n$ que são primos relativos de n, i.e. ,

$$\mathbb{Z}_n^* = \{[a]_n \in \mathbb{Z}_n \ : \ \mathtt{mdc}(a, n) = 1\}.$$

Aritmética Modular (cont.)

- ▷ *Exemplo (grupo multiplicativo)*: $\mathbb{Z}_{22}^* = \{1, 3, 5, 7, 9, 13, 15, 17, 19, 21\}$.
- ▷ *Observação 1:* 1 é a identidade para o grupo multiplicativo.
- ▷ *Observação 2:* $\mathbb{Z}_n^*$ está bem definido pois $\texttt{mdc}(a, n) = \texttt{mdc}(a + kn, n)$, i.e., $a \in \mathbb{Z}_n^*$ então todo $b \in [a]_n$ também está.

$+_6$	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

↑ $(\mathbb{Z}_6, +_6)$

$\cdot_{15}$	1	2	4	7	8	11	13	14
1	1	2	4	7	8	11	13	14
2	2	4	8	14	1	7	11	13
4	4	8	1	13	2	14	7	11
7	7	14	13	4	11	2	1	8
8	8	1	2	11	4	13	14	7
11	11	7	14	2	13	1	8	4
13	13	11	7	1	14	8	4	2
14	14	13	11	8	7	4	2	1

$(\mathbb{Z}_{15}^*, \cdot_{15})$ →

Aritmética Modular (cont.)

▷ **Teorema 31.12:** $(\mathbb{Z}_n, +_n)$ é um grupo abeliano finito.

▷ **Teorema 31.13:** $(\mathbb{Z}_n^*, .n)$ é um grupo abeliano finito.

▷ *Definição:* o **elemento inverso** (multiplicativo) de um elemento a é denotado por $(a^{-1} \bmod n)$. A **divisão** em $\mathbb{Z}_n^*$ é definida pela equação:

$$a/b \equiv ab^{-1} \ (\texttt{mod}\ n).$$

Exemplo: em $\mathbb{Z}_{15}^*$, $7^{-1} \equiv 13 \ (\texttt{mod}\ 15)$ pois $7.13 = 91 \equiv 1 \ (\texttt{mod}\ 15)$.
Logo, $4/7 \equiv 4.13 \equiv 7 \ (\texttt{mod}\ 15)$.

▷ O tamanho de $\mathbb{Z}_n^*$ é dado pela função ϕ de **Euler**, definida por:

$$\phi(n) = n \prod_{p \mid n} (1 - \frac{1}{p}),$$

onde p é um primo que divide n (incluindo n se for o caso).

Aritmética Modular (cont.)

- ▷ *Exemplo:* $\phi(22) = 22(1 - 1/2)(1 - 1/11) = 22(1/2)(10/11) = 10$.
- ▷ Para um número primo p tem-se que $\phi(p) = p - 1$.
- ▷ *Definição:* Se $(S, \oplus)$ é um grupo, $S' \subseteq S$ e $(S', \oplus)$ também é um grupo então $(S', \oplus)$ é um **subgrupo** de $(S, \oplus)$.
- ▷ *Exemplo:* os números pares formam um subgrupo de $(\mathbb{Z}, +)$.
- ▷ **Teorema 31.14:** Um subconjunto fechado de um grupo **finito** é um subgrupo.
- ▷ *Exemplo:* $\{0, 2, 4, 6\}$ é um subgrupo de $(\mathbb{Z}_8, +_8)$.
- ▷ **Teorema de Lagrange (31.15):** se $(S', \oplus)$ é um subgrupo de um grupo finito $(S, \oplus)$ então $|S'|$ é um divisor de $|S|$.

Aritmética Modular (cont.)

▷ **Corolário 31.16:** Se S' é um subgrupo próprio de um grupo finito S, então $|S'| \leq |S|/2$.

▷ **Observação:** usado na análise do teste de primalidade de Miller-Rabin ...

▷ *Criando subgrupos de um grupo finito usando o teorema 31.14:* escolha um elemento $a \in S$ e gere todos elementos que se obtém de a usando a operação de grupo (este é o **subgrupo gerado por** a).

$$a^{(k)} = \bigoplus_{i=1}^{k} a = \underbrace{a \oplus a \oplus \ldots \oplus a}_{k}, \text{ para } k \geq 1.$$

▷ *Exemplo:* $(\mathbb{Z}_6, +_6)$ e $a = 2$: $a^{(1)}, a^{(2)}, \ldots = 2, 4, 0, 2, 4, 0, 2 \ldots$.

Para $(\mathbb{Z}_n, +_n)$, tem-se $a^{(k)} = ka \bmod n$.

Para $(\mathbb{Z}_n^*, .n)$, tem-se $a^{(k)} = a^k \bmod n$.

Aritmética Modular (cont.)

▷ *Definição:* o **subgrupo gerado por** a, denotado por $\langle a \rangle$ (ou $(\langle a \rangle, \oplus)$), é dado por

$$\langle a \rangle = \{a^{(k)} \ : \ k \geq 1\}.$$

▷ Fatos diversos:

- $\langle a \rangle$ é finito.
- $\langle a \rangle$ é fechado pois, pela associatividade, $a^{(i)} \oplus a^{(j)} = a^{(i+j)}$.
- pelo Teorema 31.14, $\langle a \rangle$ é um subgrupo de S.
- *Exemplos:* para o $\mathbb{Z}_7^*$ tem-se

$$\langle 1 \rangle = \{1\}$$
$$\langle 2 \rangle = \{1, 2, 4\}$$
$$\langle 3 \rangle = \{1, 2, 3, 4, 5, 6\}.$$

Aritmética Modular (cont.)

▷ **Teorema 31.17**: para todo grupo finito $(S, \oplus)$ e todo $a \in S$ tem-se: $\texttt{ord}(a) = |\langle a \rangle|$.

▷ **Corolário 31.18**: A seqüência $a^{(1)}, a^{(2)}, \ldots$ é periódica com período $t = \texttt{ord}(a)$, i.e., $a^{(i)} = a^{(j)} \Leftrightarrow i \equiv j (\text{mod } t)$.

Nota: $a^{(0)} = e$.

▷ **Corolário 31.19**: Se $(S, \oplus)$ é um grupo finito com identidade e, então, para todo $a \in S$ tem-se que

$$a^{(|S|)} = e.$$

Resolução de Equações Lineares Modulares (Seção 31.4)

▷ Encontrar as raízes da equação $\boxed{ax \equiv b \ (\texttt{mod}\ n)}$ (31.22), $n > 0$.

▷ *Observações:*

- $\langle a \rangle$ subgrupo de $(\mathbb{Z}_n, +_n)$ gerado por a
- A equação (31.22) tem solução $\Leftrightarrow b \in \langle a \rangle$.
- $|\langle a \rangle|$ é divisor de n (Lagrange, Teorema 31.5).

▷ **Teorema 31.20**: para todo inteiro a e n, se $d = \texttt{mdc}(a, n)$ então, em $\mathbb{Z}_n$, $\langle a \rangle = \langle d \rangle = \{0, d, 2d, 3d, \ldots, ((n/d) - 1)d\}$. Portanto, tem-se que $|\langle a \rangle| = n/d$.

▷ **Corolário 31.21**: a equação $ax \equiv b \ (\texttt{mod}\ n)$ tem solução se e somente se $\texttt{mdc}(a, n) \mid b$ (i.e., $= d \mid b$).

Equações Lineares Modulares (cont.)

▷ **Corolário 31.22**: a equação $ax \equiv b \pmod{n}$ ou não tem solução ou tem $d = \mathtt{mdc}(a, n)$ soluções distintas módulo n.

▷ **Teorema 31.23**: seja $d = \mathtt{mdc}(a, n) = ax' + ny'$ (x' e y' são inteiros obtidos, p.ex., por X-EUCLIDES(a, n)). Se $d \mid b$, então, a equação $ax \equiv b \pmod{n}$ tem uma solução cujo valor é $\boxed{x_0 = x'(b/d) \bmod n}$.

▷ *Exemplo*: $14x \equiv 28 \pmod{105}$.
$a = 14$, $b = 28$, $n = 105$, $d = 14.(-7) + 105.(1) = 7$, $n/d = 15$.
$x' = -7$, $x_0 = (-7).(28/7) \pmod{105} = (-28) \bmod 105 = 77$.

Equações Lineares Modulares (cont.)

▷ **Teorema 31.24**: suponha que $ax \equiv b \pmod{n}$ tem solução e que x_0 é uma solução para esta equação. Então a equação tem exatamente $d = \texttt{mdc}(a, n)$ soluções distintas, módulo n, dadas por:

$$x_i = x_0 + i(n/d), \text{ para } i = 0, 1, \ldots, d-1.$$

Modular-Linear-Equation-Solver(a, b, n)
Entrada: inteiros a, b e n, com $n > 0$.
Saída: $x \in \mathbb{Z}_n^*$ satisfazendo $ax \equiv b \pmod{n}$.
$(d, x', y') \leftarrow$ X-EUCLIDES(a, n)
Se $d \mid b$ **então**
$x_0 \leftarrow x'(b/d) \pmod{n}$
Para $i = 0$ **até** $d - 1$ **faça**
imprima $x_0 + i(n/d) \pmod{n}$
se não imprima "não há solução".
FIM

Equações Lineares Modulares (cont.)

▷ Exemplo:

$9x \equiv 12 \pmod{15}$.
$a = 9$, $b = 12$, $n = 15$, $d = 9.(2) + 15.(-1) = 3$, $n/d = 5$.
$x' = 2$, $x_0 = (2).(12/3) \pmod{15} = 8$.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$[9i]_{15}$	0	9	3	12	6	0	9	3	12	6	0	9	3	12	6
				↑					↑					↑	

Equações Lineares Modulares (cont.)

▷ **Corolário 31.25**: Sejam a e n dois inteiros tais que $n > 1$ e $\texttt{mdc}(a, n) = 1$. Então, $ax \equiv b \ (\texttt{mod}\ n)$ tem **uma única** solução módulo n.

▷ **Corolário 31.26**: Sejam a e n dois inteiros tais que $n > 1$ e $\texttt{mdc}(a, n) = 1$. Então, $ax \equiv 1 \ (\texttt{mod}\ n)$ tem **uma única** solução módulo n e esta solução será o **inverso multiplicativo** de a em $\mathbb{Z}_n^*$.

▷ O *inverso multiplicativo* de $a \ \texttt{mod}\ n$, denotado por $(a^{-1} \ \texttt{mod}\ n)$, é facilmente computável usando o algoritmo X-EUCLIDES(a, n, x', y'). Neste caso tem-se: $(a^{-1} \ \texttt{mod}\ n) = x'$.

Teorema Chinês do Resto (Seção 31.5)

▷ Sun-Tsu matemático chinês no ano 100 d.c.:

> Encontre os inteiros x que ao serem divididos por 3, 5, e 7 dão resto 2, 3 e 2 respectivamente.

▷ *Formalizando*: $x \equiv 2 \pmod 3$, $x \equiv 3 \pmod 5$ e $x \equiv 2 \pmod 7$.
Qual o valor de x ?

▷ *Resposta*: $x = 23 + 105k$, k inteiro.

▷ O *Teorema Chinês do Resto* generaliza este resultado fornecendo uma correspondência entre um sistema de equações módulo um conjunto de números primos entre si dois a dois (p.ex., 3, 5 e 7) e uma equação módulo o produto destes números (neste caso, 105).

Teorema Chinês do Resto (Teorema 31.27)

Seja $n = n_1.n_2.\ldots.n_k$ onde $\texttt{mdc}(n_i, n_j) = 1$ para todo $1 \leq i < j \leq k$. Considere a relação dada por:

$$a \leftrightarrow (a_1, a_2, \ldots, a_k) \qquad (31.25)$$

onde $a \in \mathbb{Z}_n, a_i \in \mathbb{Z}_{n_i}$ e $a_i = a \bmod n_i$, $\forall\ i = 1, \ldots, k$. **Então**, o mapeamento (31.25) é uma bijeção de $\mathbb{Z}_n$ no produto cartesiano $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2} \times \ldots \mathbb{Z}_{n_k}$. Operações realizadas sobre elementos de $\mathbb{Z}_n$ podem ser realizadas de maneira equivalente nas k-tuplas correspondentes. Para tanto, as operações deverão ser realizadas de forma independente sobre as k coordenadas. Ou seja, para a e b dados por $a \leftrightarrow (a_1, a_2, \ldots, a_k)$ e $b \leftrightarrow (b_1, b_2, \ldots, b_k)$ tem-se que:

$$(a + b) \bmod n \leftrightarrow ((a_1 + b_1) \bmod n_1, \ldots, (a_k + b_k) \bmod n_k)\,, \qquad (31.26)$$

$$(a - b) \bmod n \leftrightarrow ((a_1 - b_1) \bmod n_1, \ldots, (a_k - b_k) \bmod n_k)\,, \qquad (31.27)$$

$$(ab) \bmod n \leftrightarrow ((a_1 b_1) \bmod n_1, \ldots, (a_k b_k) \bmod n_k)\,. \qquad (31.28)$$

Teorema Chinês do Resto (Prova)

▷ *Resultado preliminar:* (Exercício 31.1-6)
Se a e b são inteiros tais que $a \mid b$ e $b > 0$, então

$$(x \bmod b) \bmod a = x \bmod a.$$

▷ *Defina*: $m_i = n/n_i$ para $1, \ldots, k$

▷ *Defina*: $c_i = m_i(m_i^{-1} \bmod n_i)$ para $i = 1, \ldots, k$ (31.29)
Quem garante a existência de $(m_i^{-1} \bmod n_i)$?

▷ *Defina*: $a \equiv (a_1c_1 + a_2c_2 + \ldots + a_kc_k) \pmod n$ (31.30)

▷ *Quanto valem* $(c_i \bmod n_i)$ *e* $(c_j \bmod n_i)$ *para* $i \neq j$ *?*
Qual a representação de c_i *no produto cartesiano* $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2} \times \ldots \mathbb{Z}_{n_k}$ *?*

▷ *Quanto vale* $(a \bmod n_i)$ *para* a *dado por (31.30) ?*
Nota: quero mostrar que $a \leftrightarrow (a_1, a_2, \ldots, a_k)$!

▷ Mostrar corretude de (31.26)-(31.27): exercício 31.1-6 de novo ! □

Teorema Chinês do Resto (cont.)

Corolário 31.28: Se $n = n_1.n_2.\ldots.n_k$ onde $\texttt{mdc}(n_i, n_j) = 1$ para todo $1 \leq i < j \leq k$, então para todos inteiros $a_1, a_2, \ldots, a_k$, o sistema de equações:

$$x \equiv a_i \,(\texttt{mod}\ n_i)\ ,\ \forall\ i = 1, \ldots, k\ ,$$

tem uma **única** solução **módulo** n.

Corolário 31.29: Se todo par de inteiros do conjunto $\{n_1, n_2, \ldots, n_k\}$ é primo relativo e $n = n_1.n_2.\ldots.n_k$, então para todo par de inteiros x e a,

$x \equiv a \,(\texttt{mod}\ n_i)$ para todo $i = 1, \ldots, k \iff x \equiv a \,(\texttt{mod}\ n)$.

Teorema Chinês do Resto (cont.)

▷ *Exemplo*: $a \equiv 2 \ (\texttt{mod}\ 5)$ e $a \equiv 3 \ (\texttt{mod}\ 13)$. Quanto vale a ?

▷ Tem-se que: $n = 5.13 = 65$, $a_1 = 2$, $n_1 = m_2 = 5$,
$a_2 = 3$ e $n_2 = m_1 = 13$.

▷ Inversos multiplicativos: $m_1^{-1} \ \texttt{mod}\ n_1 = 13^{-1} \ \texttt{mod}\ 5 = 2$,
$m_2^{-1} \ \texttt{mod}\ n_2 = 5^{-1} \ \texttt{mod}\ 13 = 8$.

▷ Valores dos c_i's: $c_1 = m_1.(m_1^{-1} \ \texttt{mod}\ n_1) = 13.2 = 26$,
$c_2 = m_2.(m_2^{-1} \ \texttt{mod}\ n_2) = 5.8 = 40$.

▷ Logo: $a \equiv a_1c_1 + a_2c_2 \ (\texttt{mod}\ 65)$
$a \equiv 2.26 + 3.40 \ (\texttt{mod}\ 65)$
$a \equiv 52 + 120 \ (\texttt{mod}\ 65) = 42$.

Teorema Chinês do Resto (cont.)

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	40	15	55	30	5	45	20	60	35	10	50	25
1	26	1	41	16	56	31	6	46	21	61	36	11	51
2	52	27	2	42	17	57	32	7	47	22	62	37	12
3	13	53	28	3	43	18	58	33	8	48	23	63	38
4	39	14	54	29	4	44	19	59	34	9	49	24	64

A representação cartesiana frequentemente pode ser usada para projetar algoritmos mais rápidos porque trabalhar com os sistemas $\mathbb{Z}_{n_i}$ pode ser mais eficiente (em termos de operações de *bits*) do que trabalhar módulo n !

Exponenciação Modular (Seção 31.6)

▷ *Pergunta*: como calcular de modo eficiente as potências de $a \bmod n$ para $a \in \mathbb{Z}_n^*$?

▷ *Exemplo 1*: $a = 3$ e $n = 7$

i	0	1	2	3	4	5	6	7	8	9	10	11 ...
$3^i \bmod 7$	1	3	2	6	4	5	1	3	2	6	4	5 ...

▷ *Exemplo 2*: $a = 2$ e $n = 7$

i	0	1	2	3	4	5	6	7	8	9	10	11 ...
$3^i \bmod 7$	1	2	4	1	2	4	1	2	4	1	2	4 ...

▷ *Notação*: $\texttt{ord}(a)_n$ é a *ordem* de a em $\mathbb{Z}_n^*$.
Por exemplo, $\texttt{ord}(2)_7 = 3$.

Exponenciação Modular (cont.)

▷ **Teorema 31.30 (Euler):** Para todo número inteiro $n > 1$, $a^{\phi(n)} \equiv 1 \,(\texttt{mod}\ \ n)$ para todo $a \in \mathbb{Z}_n^*$. (31.32)

▷ **Teorema 31.31 (Fermat):** Para todo número p primo, tem-se que $a^{p-1} \equiv 1 \,(\texttt{mod}\ p)$ para todo $a \in \mathbb{Z}_p^*$. (31.33)

▷ *Definição:* $g \in \mathbb{Z}_n^*$ e $\texttt{ord}(g) = \phi(n)$ então todo elemento de $\mathbb{Z}_n^*$ é uma potência de g módulo n e g é dito ser uma **raiz primitiva** ou **gerador** de $\mathbb{Z}_n^*$. Se $\mathbb{Z}_n^*$ tiver uma raiz primitiva, este grupo é dito ser **cíclico**. *Exemplo*: 3 é raiz primitiva de $\mathbb{Z}_7^*$.

▷ **Teorema 31.32** (Niven e Zuckerman): Os valores de $n > 1$ para os quais $\mathbb{Z}_n^*$ é cíclico são 2, 4, p^e e $2p^e$, para todos os números primos ímpares p e todos os inteiros positivos e.

Exponenciação Modular (cont.)

▷ *Definição:* se g uma **raiz primitiva** de $\mathbb{Z}_n^*$ e a um elemento qualquer deste grupo, então existe um z tal que $g^z \equiv a \,(\texttt{mod}\ n)$. Este z é o **logaritmo discreto** ou **índice** de a módulo n na base g, também denotado por $\texttt{ind}_{n,g}(a)$.

▷ **Teorema 31.33**: se g é uma raiz primitiva de $\mathbb{Z}_n^*$ então $g^x \equiv g^y \,(\texttt{mod}\ n)$ se e somente se $x \equiv y \,(\texttt{mod}\ \phi(n))$.

▷ Aplicação do logaritmo discreto ...

▷ **Teorema 31.34**: se p é um número primo **ímpar** e $e \geq 1$, então

$$x^2 \equiv 1 \,(\texttt{mod}\ p^e) \qquad (31.34)$$

tem somente duas soluções dadas por $x = 1$ e $x = -1$.

Exponenciação Modular (cont.)

- ▷ *Definição:* $x^2 \equiv 1 \pmod{n}$ e $x \neq \pm 1$ então x é uma **raiz quadrada não trivial de 1, módulo** n.
- ▷ *Exemplo*: 9 é raiz quadrada não trivial de 1, módulo 80.
- ▷ Um resultado importante para a prova de corretude do algoritmo de Miller-Rabin para o teste de primalidade ...
- ▷ **Corolário 31.35**: se existe uma raiz quadrada não trivial de 1 módulo n, então n é um **número composto**.

Exponenciação Modular (cont.)

▷ *Pergunta*: como calcular eficientemente as potências de $a^b \bmod n$?

▷ A exponenciação modular é uma operação fundamental em algoritmos de teste de primalidade e para o sistema criptográfico do RSA !

▷ *Método ingênuo*:

faz $d \leftarrow 1$ e repete b vezes a operação $d \leftarrow (a.d) \bmod n$.

Complexidade: $O(b)$ operações aritméticas e $O(b\beta^2)$ operações de bits se a, b e n tem β bits.

▷ *Técnica mais eficiente*: "repeated squaring".

▷ Supor que $b = b_k 2^k + b_k 2^k + \ldots + b_1 2^1 + b_0 2^0$ onde $k = \lceil \log b \rceil$.

▷ *Representação binária de b*: $\langle b_k, b_{k-1}, \ldots, b_1, b_0 \rangle$.

Exponenciação Modular (cont.)

▷ *Definição*: $c_{-1} = 0$ e $c_i = 2c_{i-1} + b_{k-i}$ para todo $i \in \{0, \ldots, k\}$.

▷ Por esta definição, $c_i = \sum_{j=0}^{i} 2^j b_{k-i-j}$ e, portanto, $c_k = b$.

▷ Dado $a^{c_{i-1}}$ como calcular a^{c_i} ?

$$a^{c_i} = a^{2c_{i-1}+b_{k-i}} = (a^{c_i})^2 . a^{b_{k-i}}$$

Portanto, $a^{c_i} = (a^{c_i})^2 . a \pmod{n}$ se $b_{k-i} = 1$ e
$a^{c_i} = (a^{c_i})^2 \pmod{n}$ se $b_{k-i} = 0$.

▷ *Conclusão:* armazenando o valor de $(a^{c_{i-1}} \bmod n)$ para $i = 1, \ldots, k$, posso calcular $(a^{c_i} \bmod n)$ simplesmente **elevando** $(a^{c_{i-1}} \bmod n)$ **ao quadrado** e computando o resultado em módulo n. Se b_{k-i} for um, uma multiplicação extra por a (módulo n) ainda é necessária.
Como $b = c_k$, ao final de k iterações teremos o valor de $a^{c_k} \bmod n \equiv a^b \bmod n$ como procurado !

Exponenciação Modular (cont.)

EXPONENCIAÇÃO-MODULAR(a, b, n)

1. $c \leftarrow 0$;
2. $d \leftarrow 1$;
3. seja $\langle b_k, b_{k-1}, \ldots, b_1, b_0 \rangle$ a representação binária de b;
4. **Para** $i \leftarrow 0$ **até** k **faça**
5. $\quad c \leftarrow 2c + b_{k-i}$;
6. $\quad d \leftarrow (d.d)$ `mod` n;
7. $\quad$ **Se** $b_{k-i} = 1$ **então**
8. $\quad\quad d \leftarrow (d.a)$ `mod` n;
9. **fim-para**
10. **Retorne** d.

▷ *Complexidade*: $O(\beta)$ operações aritméticas e $O(\beta^3)$ operações de bits se a, b e n tem β bits.

Exponenciação Modular (cont.)

▷ *Exemplo*: $a = 7$, $b = 560 = \langle 1000110000 \rangle$, $k = 9$ e $n = 561$.

i	0	1	2	3	4	5	6	7	8	9
$k-i$	9	8	7	6	5	4	3	2	1	0
b_{k-i}	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
d	7	49	157	526	160	241	298	166	67	1

▷ *Nota*: o valor de c não precisa ser efetivamente calculado neste algoritmo !

O sistema criptográfico RSA (Seção 31.7)

▷ *Lembrando*: sistemas de chave pública

- $\mathcal{D}$: o domínio das mensagens;
- $M = S_A(P_A(M)),$ para todo $M \in \mathcal{D}$ (31.37)
- $M = P_A(S_A(M)),$ para todo $M \in \mathcal{D}$ (31.38)

▷ *Criando as chaves no RSA*: $M \in \mathbb{Z}_n$.

Passo 1: Escolher dois números primos p e q **grandes** (512 bits).

Passo 2: Calcular $n = pq$.

Passo 3: Escolher um inteiro ímpar e pequeno tal que $\texttt{mdc}(e, \phi(n)) = 1$.

Passo 4: Calcular d, o inverso multiplicativo de e módulo $\phi(n)$.

Passo 5: Tornar pública a chave $P_A = (e, n)$.

Passo 6: Manter secreta a chave $S_A = (d, n)$.

O sistema criptográfico RSA (cont.)

▷ *As funções P_A e S_A*:

$$P_A(M) = M^e \bmod n \tag{31.39}$$

$$S_A(M) = M^d \bmod n \tag{31.40}$$

▷ *Complexidades:* para $\log e = O(1)$, $\log d \leq \beta$ e $\log n \leq \beta$

- Pública: $O(\beta^2)$.
- Secreta: $O(\beta^3)$.

▷ **Teorema 31.36** (*corretude do RSA*)
As equações (31.39) e (31.40) do RSA definem transformações inversas de $\mathbb{Z}_n$ satisfazendo às equações (31.37) e (31.38).

O sistema criptográfico RSA (cont.)

▷ *Prova de corretude do RSA*:

- $P(S(M)) = S(P(M)) = M^{ed} \pmod{n}$;
- $\phi(n) = (p-1)(q-1)$;
- $ed \equiv 1 \pmod{\phi(n)}$;
- $ed = 1 + k(p-1)(q-1)$;
- Para $M \not\equiv 0 \pmod{p}$, usar Teorema de Fermat e concluir que $M^{ed} \equiv M \pmod{p}$;
- Repetir passo anterior para provar que $M^{ed} \equiv M \pmod{q}$;
- Teorema Chinês do Resto implica que $M^{ed} \equiv M \pmod{n}$.

□

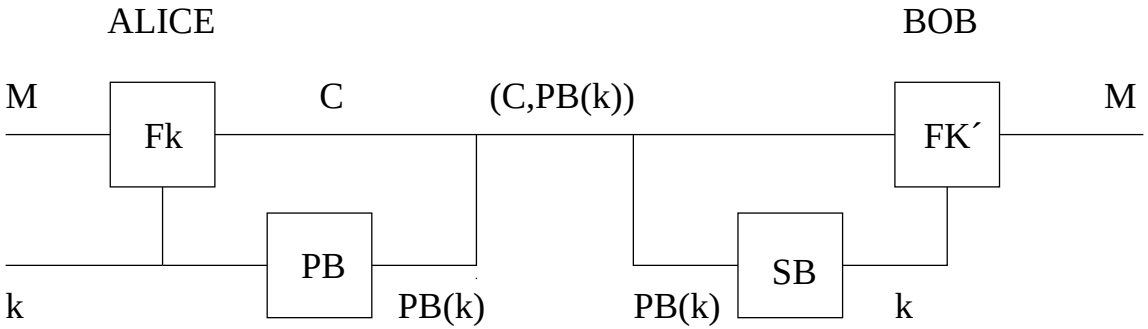
O sistema criptográfico RSA (cont.)

- ▷ Observação 1: se o problema de fatoração de números inteiros for fácil, então é fácil *quebrar* o RSA. A inversa permanece *em aberto* !
- ▷ Observação 2: implementações atuais do RSA usam valores para n que são inteiros de 768 a 2048 bits !
- ▷ Observação 3: Por eficiência, o RSA é usado comumente em sistemas criptográficos **híbridos** junto com sistemas de chave não-pública.

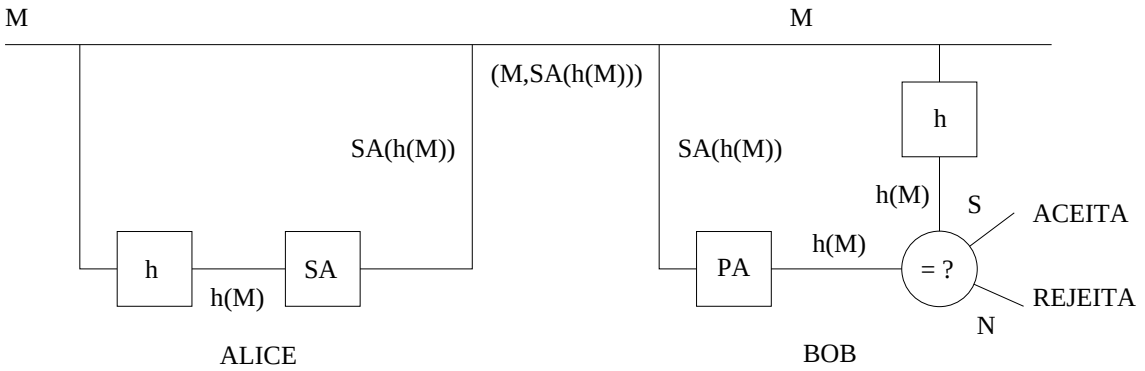
  Envio de mensagens cifradas por chave não pública k e função F_k com inversa dada por F_k^{-1} ambas muito rápidas. Normalmente k é bem menor que a mensagem M e cifrar/decifrar com o RSA apenas a chave k é muito mais rápido que cifrar/decifrar M.
- ▷ *Observação 4*: Assinatura digital **híbrida** usando função de *hashing* h facilmente computável e tal que seja praticamente inviável achar M e M' em $\mathbb{Z}_n$ satisfazendo $h(M) = h(M')$. $|h(M)|$ pequeno: 160 bits.

O sistema criptográfico RSA (cont.)

▷ RSA híbrido: esquema de cifragem/decifragem



▷ Assinatura digital usando *one way hashing function*



Testes de primalidade (Seção 31.8)

▷ *Pergunta*: Quão difícil é encontrar um número primo **grande** ?

▷ A densidade dos números primos é alta !

Teorema 31.37: Seja $\pi(n)$ a quantidade de números primos $\leq n$. Então, $\lim_{n\to\infty} \dfrac{\pi(n)}{n \,/\ln\, n} = 1$.

▷ $\pi(10) = 4$ pois 2, 3, 5 e 7 são todos os primos ≤ 10.

▷ A aproximação dada pelo Teorema 31.37 é bastante boa. Por exemplo, para $n = 10^9$ tem-se:

$$\pi(n) = 50.847.534 \quad \text{e} \quad \frac{n}{\ln\, n} \approx 48.254.942,$$

um erro de menos de 6%.

▷ A probabilidade de um número aleatório n ser primo pode ser estimada em $1/\ln n$.

Testes de primalidade (cont.)

- ▷ Assim, deve-se examinar aproximadamente $\ln n$ inteiros próximos de n para se achar um primo com a mesma ordem de grandeza que n.
- ▷ *Exemplo*: para encontrar um primo de 512 bits, devem ser testados aproximadamente $\ln 2^{512} \approx 355$ números aleatórios de 512 bits cada. Na verdade, este número cai a metade se considerarmos que só os números ímpares precisam ser testados.
- ▷ *O método da divisão trivial* (para testar se n é primo):
  Fazer $i = 2$ e `continua` = `True`. Enquanto (`continua` e $i \leq \lfloor \sqrt{n} \rfloor$), `continua` $\leftarrow$ (n não é divisível por i) e $i \leftarrow i+1$. Retornar `continua`.
- ▷ *Complexidade*: se $|n| = \beta$ e cada divisão levar tempo constante, no pior caso, o número de divisões é $\Theta(\sqrt{n}) = \Theta(2^{\beta/2})$. Portanto, é exponencial no tamanho de n e impraticável para grandes valores !
  Vantagem: para n composto, o método retorna a fatoração de n.

Testes de primalidade (cont.)

- ▷ Fatorar n é bem mais difícil do que dizer simplesmente se n é primo ou não. A segurança do RSA se baseia na dificuldade de se fatorar e sua viabilidade de implementação, na facilidade de gerar grandes números primos.
- ▷ *Definição*: $\mathbb{Z}_n^+ = \{1, 2, \ldots, n-1\}$
- ▷ *Observação*: se n é primo, $\mathbb{Z}_n^+ = \mathbb{Z}_n^*$.
- ▷ *Definição*: n é um **pseudoprimo base** a se n é composto e

$$a^{n-1} \equiv 1 \ (\texttt{mod}\ n) \qquad (31.38)$$

- ▷ O Teorema de Fermat garante que, se n é primo, (31.38) é satisfeita para **todo** a em $\mathbb{Z}_n^+$. Portanto, se acharmos um a (testemunha) que **não** satisfaz (31.38) podemos afirmar **com certeza** que n é composto.

Testes de primalidade (cont.)

▷ Mas a inversa é *quase sempre* verdadeira no sentido que, quando n é composto, praticamente qualquer $a \in \mathbb{Z}_n^+$ não satisfaz (31.38).

▷ A rotina `PSEUDOPRIMO` abaixo declara um número n como sendo primo sempre que ele primo ou pseudoprimo base 2.

`PSEUDOPRIMO`(n)
1. Se `EXPONENCIAÇÃO-MODULAR`$(2, n-1, n) \not\equiv 1$ (`mod` n)
2. **Retornar** `COMPOSTO` ▷ Com certeza !
3. **Retornar** `PRIMO` ▷ Tomara !

▷ *Pergunta*: a rotina `PSEUDOPRIMO` erra muito ?

Testes de primalidade (cont.)

▷ *Resposta 1*: ela só pode cometer erro ao declarar um número primo. Mas ela erra muito pouco. Por exemplo, só existem 22 valores para os quais ela erra entre os 10.000 primeiros números inteiros (341, 561, 645, 1105, ...).

▷ *Resposta 2*: pode-se provar que para um número de β bits escolhido aleatoriamente, a probabilidade da rotina cometer um erro vai para zero à medida que $\beta \to \infty$.

▷ *Resposta 3*: estudos sobre a quantidade de pseudoprimos base 2 de um tamanho fixo indicam que um número de 512 bits escolhido aleatoriamente classificado como primo pela rotina tem probabilidade inferior a $1/10^{20}$ de ser um pseudoprimo base 2. Para 1024 bits, essa probabilidade cai para $1/10^{41}$.

Testes de primalidade (cont.)

▷ *Pergunta*: é possível eliminar **completamente** os erros forçando a rotina a verificar outros pseudoprimos base a para $a \neq 2$?

Resposta: infelizmente não ! Existem números compostos que satisfazem $a^{n-1} \equiv 1 \pmod{n}$ para todo a em $\mathbb{Z}_n^*$. Estes números são chamados de **números de Carmichael**.

▷ Os primeiros três **números de Carmichael** são: 561, 1105, 1729. Estes números são **raríssimos**. Só existem 255 entre os 100.000.000 primeiros números inteiros.

▷ *Observação*: (exercício 31.8-2) Pode-se mostrar que os números de Carmichael não são divisíveis por nenhum quadrado de número primo e que eles devem ser o produto de três números primos. Por isso eles são tão raros.

Testes de primalidade: Miller-Rabin

▷ O algoritmo de Miller-Rabin melhora o algoritmo `PSEUDOPRIMO` através de 2 modificações:

- vários valores da base $\underline{a}$ são gerados aleatoriamente e testados.
- Durante o cálculo da exponenciação modular, verifica-se se alguma raiz não-trivial da unidade módulo n é encontrada. Se for, o algoritmo retorna que n é `COMPOSTO`.

▷ O procedimento auxiliar `TESTEMUNHA`(a, n) retorna `Verdadeiro` se e somente se a base $\underline{a}$ pode ser usada para provar que n é composto.

▷ Assumir que $n - 1 = 2^t u$, onde $t \geq 1$ e u é ímpar.

Ou seja, a representação binária de $n-1$ é a representação binária do número ímpar u seguida de t zeros.

Testes de primalidade: Miller-Rabin (cont.)

▷ Desta forma, $a^{n-1} \equiv (a^u)^{2^t} \pmod{n}$ de modo que $a^{n-1} \bmod n$ pode ser computado primeiramente computando $a^u \bmod n$ e depois elevando-se o resultado ao quadrado t vezes sucessivamente.

```
TESTEMUNHA(a, n)
1.  seja n − 1 = 2^t u, onde t ≥ 1 e u é ímpar
2.  x_0 ← EXPONENCIAÇÃO-MODULAR(a, u, n)
3.  Para i ← 1 até t faça
4.      x_i ← x_{i−1}^2 mod n;
5.      Se x_i = 1 e x_{i−1} ≠ 1 e x_{i−1} ≠ n − 1
6.      então Retorne VERDADEIRO
7.  fim-para
8.  Se x_t ≠ 1 então Retorne VERDADEIRO
9.  Retorne FALSO.
```

Testes de primalidade: Miller-Rabin (cont.)

▷ Análise do algoritmo `TESTEMUNHA`:

- Na linha 8, foi descoberto que $x_t = a^{n-1} \neq 1 \bmod n$. Como $a \in \mathbb{Z}_n^*$, o Teorema de Fermat garante que n é composto.
- Nas linhas 5 e 6, achou-se uma raiz não trivial de 1 módulo n em $\mathbb{Z}_n^*$. Pelo Corolário 31.35, n tem que ser composto.

Conclusão: sempre que a rotina retornar `VERDADEIRO`, a base a fornece uma prova de que n é de fato composto !

MILLER-RABIN(n, s) ▷ $n \geq 2$ e ímpar

1. **Para** $j = 1$ **até** s **faça**
2. $\quad a \leftarrow$ `RANDOM`$(1, n-1)$
3. $\quad$ **Se** `TESTEMUNHA`(a, n) Retorne `COMPOSTO` ▷ *com certeza !*
4. **fim-para**
5. Retorne `PRIMO` ▷ *é quase certo !*

Testes de primalidade: Miller-Rabin (cont.)

▷ *Complexidade*: $O(s\beta)$ operações aritméticas e $O(s\beta^3)$ operações de bit.

▷ *Exemplo*: $n = 561$ (número de Carmichael)

Tem-se que $n - 1 = 560 = 2^4.35$, logo $u = 35$ e $t = 4$. Se usarmos $a = 7$, `TESTEMUNHA` calculará $x_0 \equiv a^{35} \equiv 241$ (`mod` 561). Na seqüência teremos $x = \langle 241, 298, 166, 67, 1\rangle$.

Portanto, no último passo de elevação ao quadrado, descobriu-se uma raiz não trivial de 1 módulo 561 já que $a^{280} \equiv 67$ (`mod` 561) e $a^{560} \equiv 1$ (`mod` 561). Assim, a base 7 mostra que 561 é `COMPOSTO`.

i	0	1	2	3	4	5	6	7	8	9
$k-i$	9	8	7	6	5	4	3	2	1	0
b_{k-i}	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
d	7	49	157	526	160	241	298	166	67	1

Testes de primalidade: Miller-Rabin (cont.)

▷ Se o algoritmo de Miller-Rabin retorna `PRIMO` existe uma chance pequena de que ele tenha errado. Ao contrário do algoritmo `PSEUDOPRIMO` a probabilidade de erro não depende de n.

Não existem entradas *ruins* para o algoritmo de Miller-Rabin !

▷ O sucesso do algoritmo depende do número ($\underline{s}$) de bases testadas e das bases ($\underline{a}$) propriamente ditas.

▷ **Teorema 31.38** Se n é um número ímpar composto, então o número de testemunhas de que n é de fato composto é pelo menos $(n-1)/2$.

▷ **Teorema 31.39** Para todo inteiro ímpar $n > 2$ e todo inteiro positivo s, a probabilidade de que `MILLER-RABIN`(n, s) erre é, no máximo, 2^{-s}.