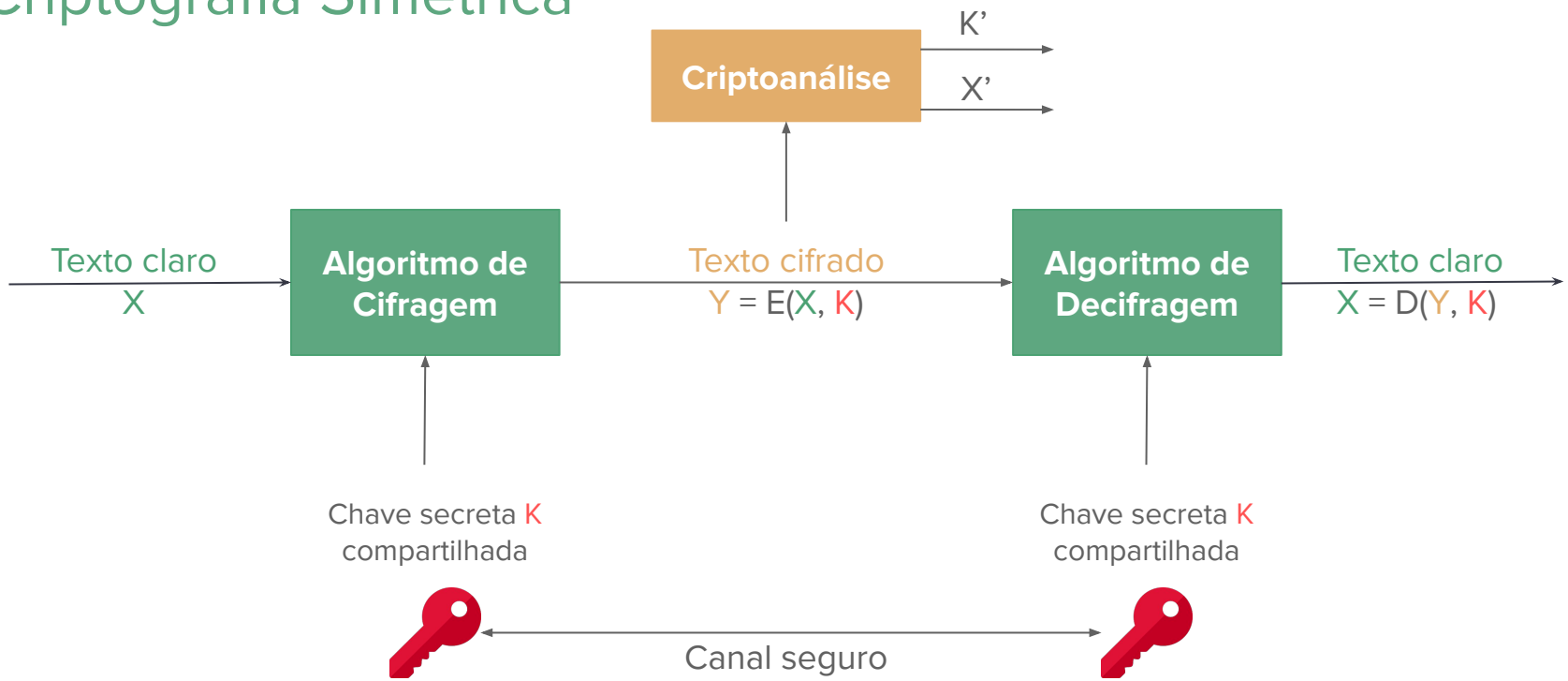
Criptografia Aplicada

Criptografia assimétrica - princípios matemáticos

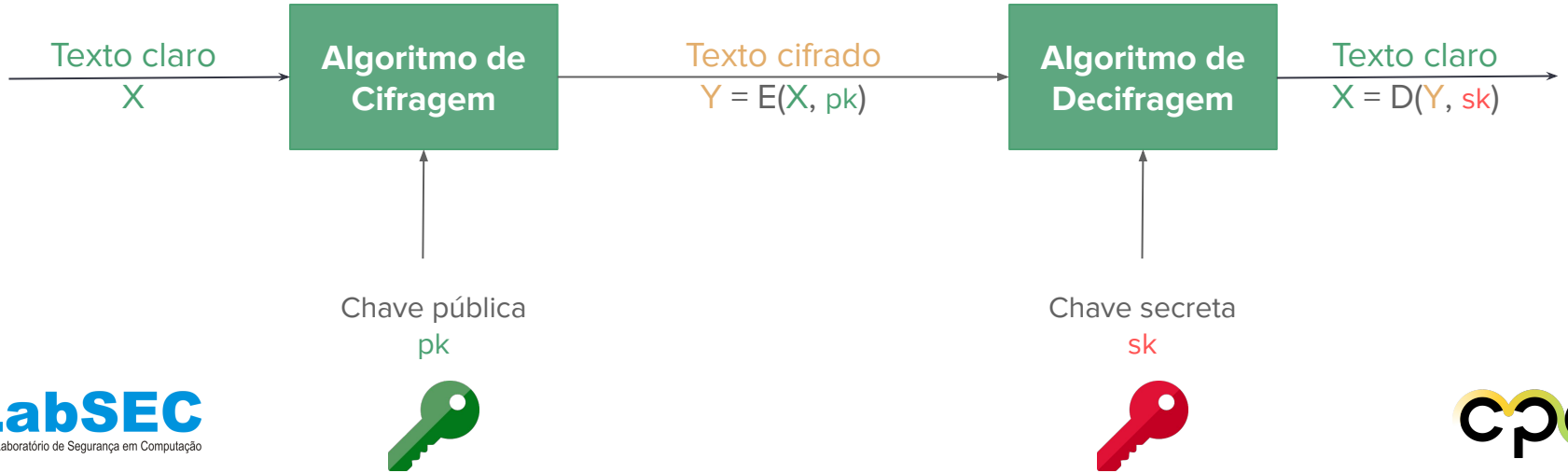
LabSEC
Laboratório de Segurança em Computação

Criptografia Simétrica



Criptografia Assimétrica

- Um par de chaves: uma chave pública e uma privada
- O que é feito com uma chave poder ser “desfeito” com outra
- Para estudar algoritmos assimétricos precisamos de **teoria dos números**



LabSEC
Laboratório de Segurança em Computação

cpqd

Sumário

- Conceitos básicos de teoria de números
- Aritmética modular
- Teorema de Fermat e Euler
- Testes de primalidade
- Problema do logaritmo discreto

Números primos

Definição: um inteiro p > 1 é um número primo se e somente se os seus únicos divisores são 1 e ele mesmo.

- Fundamentais para criptografia assimétrica
- **Fatoração prima**: Todo número inteiro pode ser representado unicamente por uma fatoração de (potências de) primos
  - $12 = 2^2 \times 3$, $91 = 7 \times 13$, $3600 = 2^4 \times 3^2 \times 5^2$
- Não conhecemos nenhum algoritmo de fatoração (não quântico) eficiente o suficiente para fatorar números grandes
  - garante a segurança de esquemas como o RSA.

Table 2.5 Primes Under 2000

2	101	211	307	401	503	601	701	809	907	1009	1103	1201	1301	1409	1511	1601	1709	1801	1901
3	103	223	311	409	509	607	709	811	911	1013	1109	1213	1303	1423	1523	1607	1721	1811	1907
5	107	227	313	419	521	613	719	821	919	1019	1117	1217	1307	1427	1531	1609	1723	1823	1913
7	109	229	317	421	523	617	727	823	929	1021	1123	1223	1319	1429	1543	1613	1733	1831	1931
11	113	233	331	431	541	619	733	827	937	1031	1129	1229	1321	1433	1549	1619	1741	1847	1933
13	127	239	337	433	547	631	739	829	941	1033	1151	1231	1327	1439	1553	1621	1747	1861	1949
17	131	241	347	439	557	641	743	839	947	1039	1153	1237	1361	1447	1559	1627	1753	1867	1951
19	137	251	349	443	563	643	751	853	953	1049	1163	1249	1367	1451	1567	1637	1759	1871	1973
23	139	257	353	449	569	647	757	857	967	1051	1171	1259	1373	1453	1571	1657	1777	1873	1979
29	149	263	359	457	571	653	761	859	971	1061	1181	1277	1381	1459	1579	1663	1783	1877	1987
31	151	269	367	461	577	659	769	863	977	1063	1187	1279	1399	1471	1583	1667	1787	1879	1993
37	157	271	373	463	587	661	773	877	983	1069	1193	1283		1481	1597	1669	1789	1889	1997
41	163	277	379	467	593	673	787	881	991	1087		1289		1483		1693			1999
43	167	281	383	479	599	677	797	883	997	1091		1291		1487		1697			
47	173	283	389	487		683		887		1093		1297		1489		1699			
53	179	293	397	491		691				1097				1493					
59	181			499										1499					
61	191																		
67	193																		
71	197																		
73	199																		
79																			
83																			
89																			
97																			

Imagem: W. Stallings. *Cryptography and network security*. Cap 2.4

LabSEC
Laboratório de Segurança em Computação

cpqd

Máximo divisor comum

Definição: um número *d* é divisor de *a* se $a = d * n$, onde a, d, n são inteiros.

Exemplo: os divisores positivos de 24 são 1, 2, 3, 4, 6, 8, 12, e 24.

Definição: o máximo divisor comum entre *a* e *b* é o maior inteiro que divide simultaneamente *a* e *b*.

Exemplo:

- $360 = 2^3 * 3^2 * 5$ e $84 = 2^2 * 3 * 7$
- $mdc(360, 84) = 12 = 2^2 * 3$

Máximo divisor comum

- O cálculo do mdc é importante para diversos algoritmos criptográficos
  - Se a e b são “relativamente primos”, isso significa que mdc(a,b) = 1
- A técnica de fatoração para encontrar o mdc só funciona para números compostos, cuja fatoração contém primos conhecidos/pequenos.
- Como calculamos mdc de números grandes?
  - Algoritmo de Euclides
  - mdc(a, b) = mdc(b, a mod b)
  - **Exemplo**: mdc(360, 84) = mdc(84, 24) = mdc(24, 12) = mdc(12, 0) = 12

Algoritmo de Euclides

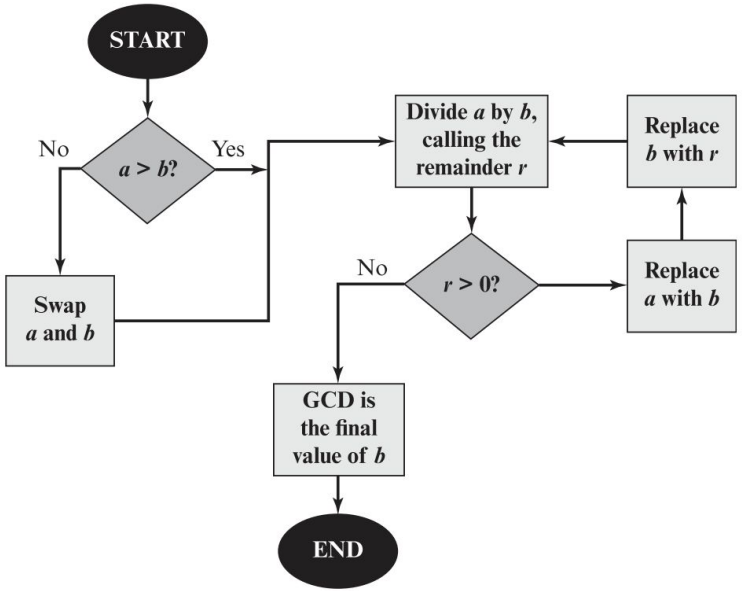


Figure 2.2 Euclidean Algorithm

Algoritmo de Euclides:

sejam a = 360 e b = 84:

a = q × b + r

360 = (4) × 84 + 24
84 = (3) × 24 + 12
24 = (2) × 12 + 0

Imagem: W. Stallings. *Cryptography and network security*. Cap 2.2

Problema da fatoração

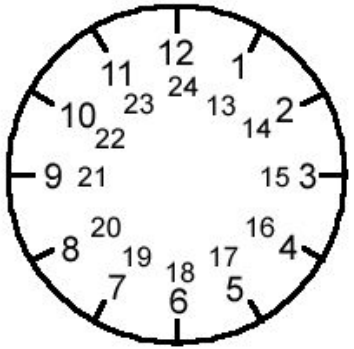
- Considere *n* um número composto
- Para fatorar um número pequeno, podemos utilizar a técnica da escola: tentar dividir pelos primeiros números primos até atingir √n
- Para números maiores, existem algoritmos mais sofisticados
- Para números suficientemente grandes, não se conhece nenhum algoritmo eficiente para fatoração
    - a não ser que se tenha um computador quântico grande o suficiente para executar o algoritmo de Shor
- A dificuldade presumida deste problema é importante para os algoritmos usados em criptografia, como a criptografia de chave pública RSA

Sumário

- Conceitos básicos de teoria de números
- **Aritmética modular**
- Teorema de Fermat e Euler
- Testes de primalidade
- Problema do logaritmo discreto

Mod e congruência

- **a mod n** é o inteiro **r** tal que a = qn+r, 0 <= r < n
  - exemplo: **11 mod 7** = 4 pois 11 = 1 x 7 + 4
  - *a mod n* é o resto da divisão de *a* por *n*
  - *n* é chamado de módulo, r de resto
  - *mod n* mapeia qualquer número inteiro *a* para um número do conjunto {0, 1, ..., n-1}
  - **Obs.**: se a mod n = 0, então a é divisível por n
- **a ≡ b (mod n)** se *a mod n = b mod n*
  - 24 ≡ 9 (mod 5) (r = 4)
  - Obs.: se a ≡ b (mod n), então b ≡ a (mod n)



LabSEC
Laboratório de Segurança em Computação

cpqd

O conjunto $\mathbb{Z}_n$ e aritmética modular

- $\mathbb{Z}_n$ é o conjunto de inteiros módulo n
  - ou seja, $\mathbb{Z}_n = \{0, 1, ..., n-1\}$
- Podemos executar operações restritas em $\mathbb{Z}_n$ (aritmética modular)
  - [(a mod n) + (b mod n)] mod n = (a + b) mod n
  - [(a mod n) - (b mod n)] mod n = (a - b) mod n
  - [(a mod n) * (b mod n)] mod n = (a * b) mod n

$$
\begin{aligned}
&11 \bmod 8 = 3; 15 \bmod 8 = 7 \\
&[(11 \bmod 8) + (15 \bmod 8)] \bmod 8 = 10 \bmod 8 = 2 \\
&(11 + 15) \bmod 8 = 26 \bmod 8 = 2 \\
&[(11 \bmod 8) - (15 \bmod 8)] \bmod 8 = -4 \bmod 8 = 4 \\
&(11 - 15) \bmod 8 = -4 \bmod 8 = 4 \\
&[(11 \bmod 8) \times (15 \bmod 8)] \bmod 8 = 21 \bmod 8 = 5 \\
&(11 \times 15) \bmod 8 = 165 \bmod 8 = 5
\end{aligned}
$$

Imagem: W. Stallings. *Cryptography and network security*. Cap 2.3

26 | 8
2 3

-4 | 8
4 -1

165 | 8
5 20

LabSEC
Laboratório de Segurança em Computação

cpqd

O conjunto $\mathbb{Z}_n$ e aritmética modular

- Exponenciação é feita com multiplicações sucessivas, como nos números inteiros com aritmética "tradicional"

$121 \equiv 4 \pmod{13}$

$121 \mid 13$ $\quad$ $4 \mid 13$
$4 \quad 9$ $\quad$ $4 \quad 0$

To find $11^7 \bmod 13$, we can proceed as follows:

$$11^2 = 121 \equiv 4 \pmod{13}$$
$$11^4 = (11^2)^2 \equiv 4^2 \equiv 3 \pmod{13}$$
$$11^7 = 11 \times 11^2 \times 11^4$$
$$11^7 \equiv 11 \times 4 \times 3 \equiv 132 \equiv 2 \pmod{13}$$

Imagem: W. Stallings. *Cryptography and network security*. Cap 2.3

O conjunto $\mathbb{Z}_n$ e aritmética modular

Table 2.2 Arithmetic Modulo 8

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

(a) Addition modulo 8

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

(b) Multiplication modulo 8

Imagens: W. Stallings. *Cryptography and network security*. Cap 2.3

LabSEC
Laboratório de Segurança em Computação

cpqd

Inversas

- A inversa aditiva de *a* em $\mathbb{Z}_n$ é
  - um inteiro $x \in \mathbb{Z}_n$ tal que $a+x \equiv 0 \pmod n$
  - chamamos de -a
  - **exemplo**: inversa aditiva de 6 em $\mathbb{Z}_8$ é 2
- A inversa multiplicativa de *a módulo n* é
  - um inteiro $x \in \mathbb{Z}_n$ tal que $a \cdot x \equiv 1 \pmod n$
  - se x existe, ele é único e chamamos de a^{-1}
  - **exemplo**: inversa multiplicativa de 3 em $\mathbb{Z}_8$ é 3.
- Nem todo número tem inversa multiplicativa em $\mathbb{Z}_n$
  - *a* é inversível sse mdc(a,n) = 1
  - precisaremos de um número com inversa quando estudarmos a geração de chaves no RSA

Table 2.2 Arithmetic Modulo 8

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

(a) Addition modulo 8

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

(b) Multiplication modulo 8

w	$-w$	w^{-1}
0	0	—
1	7	1
2	6	—
3	5	3
4	4	—
5	3	5
6	2	—
7	1	7

(c) Additive and multiplicative inverse modulo 8

LabSEC
Laboratório de Segurança em Computação

Imagens: W. Stallings. *Cryptography and network security*. Cap 2.3

cpqd

Algoritmo de Euclides Estendido

- Se mdc(a,n) = 1, sabemos que a^{-1} existe em $\mathbb{Z}_n$
  - além disso, sabemos também que mdc(a,n) = ax + ny
  - se mdc(a,n) = 1 = ax + ny, o valor de x é o a^{-1}
- Para encontrá-lo, utilizamos o algoritmo de euclides estendido
- **Exemplo**: seja a = 5 e n = 8, queremos encontrar a inversa de 5 em $\mathbb{Z}_8$

Algoritmo de Euclides:

$n = q \times a + r$

$8 = 1 \times 5 + 3$

$5 = 1 \times 3 + 2$

$3 = 1 \times 2 + 1$

$2 = 2 \times 1 + 0$

Algoritmo de Euclides estendido:

$mdc(5,8) = 1 = 3 - 1 \times 2$

$= 3 - 1 \times [5 - 1 \times 3]$ (substitui 2)

$= 3 - 1 \times 5 + 1 \times 3$ (distributiva)

$= 2 \times 3 - 1 \times 5$ (juntar)

$= 2 \times [8 - 1 \times 5] - 1 \times 5$ (substitui 3)

$= 2 \times 8 - 2 \times 5 - 1 \times 5$ (distributiva)

$= 2 \times 8 - 3 \times 5$ (juntar)

inversa de 5 é $-3 \bmod 8 = 5$

LabSEC
Laboratório de Segurança em Computação

cpqd

Sumário

- Conceitos básicos de teoria de números
- Aritmética modular
- **Teorema de Fermat e Euler**
- Testes de primalidade
- Problema do logaritmo discreto

(Pequeno) Teorema de Fermat

Teorema de Fermat: Se *p* é primo e *a* é um inteiro positivo não divisível por *p* então

$$a^{p-1} \equiv 1 \pmod{p}$$

- Requer que **a** e **p** sejam relativamente primos
  - mdc(a,p) = 1
- Forma alternativa: *p* é primo e *a* é um inteiro positivo, então $a^p \equiv a \pmod{p}$
- **Exemplo**: p = 5, a = 3
  - $a^{p-1} = 3^4 = 81 \equiv 1 \pmod{5}$
  - $a^p = 3^5 = 243 \equiv 3 \pmod{5} = a \pmod{p}$
  - $3^{402} \pmod{5}$
  - $(3^4)^{100} \times 3^2 \equiv 1^{100} \times 3^2 \equiv 9 \equiv 4 \pmod{5}$

Grupo multiplicativo

- Se *n* é primo, todo valor a $\in \mathbb{Z}_n$, a $\neq 0$ tem inversa
- O **grupo multiplicativo** de $\mathbb{Z}_n$ é definido como
  $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \mid mdc(a,n) = 1\}$
  - $\mathbb{Z}_5^* = \{1, 2, 3, 4\}$
  - $\mathbb{Z}_8^* = \{1, 3, 5, 7\}$
  - $\mathbb{Z}_{35}^* = \{1, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18, 19, 22, 23, 24, 26, 27, 29, 31, 32, 33, 34\}$
- O tamanho de $\mathbb{Z}_n^*$ é dado por $\phi(n)$, a função totiente de Euler:
  $\phi(n) = |\mathbb{Z}_n^*|$
  - Exemplo: $\phi(5) = 4$, $\phi(8) = 4$, $\phi(37) = 36$, $\phi(35) = 24$
- Se n é primo, $\phi(n) = (n-1)$
- n = pq é o produto de dois primos p e q, então
  $\phi(n) = \phi(p)\phi(q) = (p-1)(q-1)$
  - usaremos essa função quando estudarmos RSA

Multiplicação em $\mathbb{Z}_5$

$\otimes_5$	**0**	**1**	**2**	**3**	**4**
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Teorema de Euler

Teorema de Euler: se *a* e *n* são relativamente primos, então

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

- Note que esse teorema não assume *n* primo!
- Forma alternativa: $a^{\phi(n)+1} \equiv a \pmod{n}$
- **Exemplo**: a = 3, n = 10, mdc(3,10) = 1, $\phi(10) = \phi(2)\phi(5) = 1x4 = 4$
  - $a^{\phi(n)} = 3^4 = 81 \equiv 1 \pmod{10}$
  - Versão alternativa: $3^5 = 243 \equiv 3 \pmod{10}$

Aplicações

- A contrapositiva do teorema de Fermat é utilizada em testes de primalidade
- O teorema de Euler é fundamental na cifragem e decifragem do esquema RSA

Sumário

- Conceitos básicos de teoria de números
- Aritmética modular
- Teorema de Fermat e Euler
- **Testes de primalidade**
- Problema do logaritmo discreto

Testes de primalidade

- Saber se um número é primo é importante para criptografia
  - Algoritmos com o RSA precisam de números primos "aleatórios" grandes para o funcionamento correto e seguro
- Euclides (365-300 BC) provou o teorema de que existem infinitos números primos
- Temos que trabalhar com números das ordem de grandeza de 1024 bits
  - Um nº aleatório tem Pr de 1/ ln 2^{1024} ≅ 1/710 de ser primo
- Na prática, geramos um número grande ímpar e testamos se ele é primo
  - utilizamos algoritmos probabilísticos e eficientes
  - existe a possibilidade de o algoritmo responder que é primo mesmo quando não é
  - ao rodar o algoritmo com o mesmo número diversas vezes, a probabilidade de erro reduz bastante

Teste de Fermat

- **Teorema de Fermat**: Se *p* é primo e *a* é um inteiro positivo não divisível por *p* então

$$a^{p-1} \equiv 1 \pmod{p}$$

- Teste de primalidade: se $2^{n-1} \equiv 1 \pmod{n}$, o número n é primo
  - Note que esse é o *converso* do teorema de Fermat, não foi provado e nem sempre é verdade!
  - esse teste é condição necessária mas não suficiente para garantirmos que *n* é primo
  - se *n* falhar no teste, temos certeza que ele é composto
  - se *n* passar no teste, ele é provavelmente primo
  - **Cuidado**, existem exceções: os pseudoprimos
    - $2^{560} \equiv 1 \pmod{561}$ mas $561 = 3 \times 11 \times 17$

Algoritmo de Miller-Rabin

Começamos com algumas propriedades:

- Para qualquer número ímpar $n \geq 3$, podemos escrever $\mathbf{n - 1 = 2^k m}$, onde $k > 0$ e m é a parte ímpar do número.
  - Note que, se *n* é ímpar, então n-1 é par, estamos dividindo n-1 por 2 *k vezes*, até chegar em *m*.
- Seja *p* um primo maior que 2, podemos escrever $p-1 = 2^k m$.
- Seja *a* qualquer inteiro $1 < a < p-1$, uma das duas condições é verdadeira (ver prova no Stallings):
  - $a^m \equiv 1 \pmod p$ (ou seja, $a^m \bmod p = 1$); ou
  - um dos valores $a^m, a^{2m}, a^{4m}, \ldots, a^{(2^{k-1})m}$ será congruente à -1 modulo p
    - ou seja, $a^{(2^i)m} \equiv -1 \pmod p$, para algum $0 < i < k$
    - Lembrete: $-1 \bmod p = p-1$

- Se fizermos esse teste com um número *n* e:
  - nenhuma dessas condições for satisfeita, *n* é **com certeza composto**;
  - se alguma das condições for satisfeita, *n* não necessariamente é primo.
  - **Exemplo**: n = 2047, n-1 = 2x1023, $2^{1023} \equiv 1 \pmod{2047}$ mas 2047 = 23x89 é composto.

Algoritmo de Miller-Rabin

Miller-Rabin(n)

1. Escreva n - 1 = 2^k m, onde *m* é ímpar
2. selecione um número aleatório *a*, 1 < a < n-1
3. Se a^m mod n = 1
   return "provavelmente primo"
4. para i = 0 até k - 1:
   Se $a^{(2^i)m}$ mod n = n-1
   return "provavelmente primo"
5. **return "composto"**

Exemplo: n = 29

1. n-1 = 28 = 2^2*7, então k = 2 e m = 7
2. testamos a = 10
3. 10^7 mod 29 = 17
4. $(10^7)^2$ mod 29 = 28, portanto provavelmente primo

Podemos testar mais uma vez com a=2

1. 2^7 mod 29 = 12
2. $(2^7)^2$ mod 29 = 28, provavelmente primo

- Podemos testar com todos os valores de a entre 1 e 28 e todos eles vão dar provavelmente primo
- compatível com o fato de que 29 é realmente primo.

Algoritmo: W. Stallings. *Cryptography and network security*. Cap 2.6
D. Stinson e M. Paterson. *Cryptography: Theory and Practice. Cap 6.4*

LabSEC
Laboratório de Segurança em Computação

cpqd

Algoritmo de Miller-Rabin

- Esse é um teste **probabilístico**:
  - Eficiente
  - Nunca responde “composto” se for primo
  - Mas pode responder “primo” e ser composto
  - Probabilidade de erro: no máx ¼
- Podemos diminuir a probabilidade de erro rodando ele *t* vezes com valores aleatórios de *a* $(1 < a < n-1)$
- Se *n* é composto, a maioria dos valores de ***a*** falhariam o teste

Sumário

- Conceitos básicos de teoria de números
- Aritmética modular
- Teorema de Fermat e Euler
- Testes de primalidade
- **Problema do logaritmo discreto**

A função log

- O que é log?
  - é a operação inversa da exponenciação
  - $\log_a b$ nos dá o valor x tal que $a^x = b$
- Por exemplo:
  - a = 3, b = 81, sabemos que $\log_3 81 = 4$ já que $3^4 = 81$

Potências na aritmética modular

- Considere as potências de 7 em $\mathbb{Z}_{19}$
- Qual o valor de x tal que $7^x \equiv 1 \pmod{19}$?
  - **Solução**: $7^3 = 343 \equiv 1 \pmod{19}$
  - entretanto, existem outras soluções:
    $7^3 \equiv 1 \pmod{19}$, $7^9 \equiv 1 \pmod{19}$,
    $7^{12} \equiv 1 \pmod{19}$, $7^{15} \equiv 1 \pmod{19}$,
    $7^{18} \equiv 1 \pmod{19}$

- Sejam *a* e *n* inteiros relativamente primos, então existe pelo menos um inteiro *m* que satisfaça $a^m \equiv 1 \pmod{n}$
  - o menor inteiro positivo *m* que satisfaça essa equação é chamado de **ordem de a (mod n)**
  - se *a* tem ordem $\phi(n)$, dizemos que *a* é uma **raiz primitiva módulo n**
- **Exemplo:** considere a = 2 e n = 19
  - a *ordem* de 2 (mod 19) é 18
  - 18 é o menor expoente tal que $2^{18} \equiv 1 \pmod{19}$
  - ou seja, 2 é uma *raiz primitiva módulo 19*

Potências na aritmética modular

- Seja *a* uma raiz primitiva e *n* um **número primo**, sabemos que $\phi(n) = n-1$.
  - temos que $a^{\phi(n)} = a^{n-1} \equiv 1 \pmod{n}$
  - Ao calcular $a^x \pmod{n}$ para todo valor $1 \leq x \leq n-1$, produziremos todos os valores de $\mathbb{Z}_n^*$
  - ou seja, podemos gerar todo o grupo multiplicativo $\mathbb{Z}_n^*$ a partir de uma raiz primitiva.
- **Exemplo:** considere raiz primitiva a = 2 e n = 19
  - Observe o que acontece ao calcularmos $2^x \pmod{19}$ para todo valor $1 \leq x \leq 18$
    - geramos todos os números em $\mathbb{Z}_{19}^*$
  - Em $\mathbb{Z}_{19}$ temos as raízes primitivas 2, 3, 10, 13, 14, 15.

Table 2.7 Powers of Integers, Modulo 19

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

LabSEC
Laboratório de Segurança em Computação

Imagens: W. Stallings. *Cryptography and network security*. Cap 2.8

cpqd

Logaritmo discreto

- Para qualquer inteiro *b* e raiz primitiva *a* de um número primo *n*, podemos encontrar um único expoente x tal que $b = a^x \bmod n$
  - esse expoente é o **logaritmo discreto** do número *b* para a base *a (mod n)*
  - representamos esse valor como $\text{dlog}_{a,n}(b)$
- Exemplo:
  - $\text{dlog}_{2,19}(4) = 2$ pois $4 = 2^2 \bmod 19$
  - $\text{dlog}_{2,19}(18) = 9$ pois $18 = 2^9 \bmod 19$

Table 2.7 Powers of Integers, Modulo 19

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

Imagens: W. Stallings. *Cryptography and network security*. Cap 2.8

Problema do logaritmo discreto

- Considere a equação: $b = a^x \bmod p$
  - dado a, x e p, é fácil e eficiente calcular b, basta fazer exponenciação
- Problema: dado b, a e p, geralmente é muito difícil encontrar x
  - Acredita-se que a dificuldade é da mesma ordem de magnitude da dificuldade de fatorar os números requeridos para o RSA
- **Força bruta**: elevar a base a à diversas potências x até achar o valor b
  - Como $1 \leq x \leq p-1$, se p for grande o suficiente, força bruta é computacionalmente inviável
- O problema do logaritmo discreto é a base para muitos criptossistemas
  - Protocolo Diffie-Hellman, algoritmo de assinatura DSA, ElGamal, curvas elipticas, etc.
  - Nesses casos, utilizamos o grupo $\mathbb{Z}_p^*$, onde p é um primo grande

Resumo

- Conceitos básicos de teoria de números
  - Números primos, MDC, problema da fatoração
- Aritmética modular
  - aplicar as operações tradicionais e operar o mod n no final
- Teorema de Fermat e Euler
  - propriedades importantes da exponenciação modular
- Testes de primalidade
- Problema do logaritmo discreto
  - dados a, b e p, encontrar o x tal que $b = a^x \bmod p$

Referências

- W. Stallings. *Cryptography and network security*. 7a edição.
  - mdc e primos: 2.2,
  - números primos e teste de primalidade: 2.4, 2.6
  - aritmética modular: 2.3, 2.5
  - logaritmo discreto: 2.8
- D. Stinson e M. Paterson. *Cryptography: Theory and Practice*. 4a edição.
  - teste de primalidade: 6.4
  - logaritmo discreto: 7.1