Teoria de Números

- Números Primos
- Teoremas de Euler e Fermat
- Teste de Primalidade
- Teorema Chinês do Resto
- Logaritmo Discreto

Números Primos

- Primo é um inteiro que só pode ser divido por *1* e por ele mesmo sem resto
- Todo numero inteiro pode ser representado por uma fatoração de primos
- $a = \prod_{p \in P} p^n$
- $n >= 0$
- $12 = 2^2 * 3^1, 91 = 7^1 * 13^1$

Números Primos

- Multiplicação de números inteiros pode ser feita pela adição de fatores primos

$$12 * 18 = (2^2 * 3^1) * (2^1 * 3^2) = 216$$

$$(2^3 * 3^3) = 8 * 27 = 216$$

Números Primos

- Nós podemos saber que um numero divide outro se todo expoente do primo do divisor é ≤ que o do dividendo
- Calcular o MDC de números expressados em notação prima é a multiplicação dos primos pelo menor expoente
- Isso só funciona facilmente para não primos

  $355 = 3 * 5^3$

  $525 = 3 * 5^2 * 7$

- $MDC(355, 525) = 3 * 5^2 = 75$

Teorema de Fermat

- Se *p* é primo e *a* é um inteiro positivo não divisível por *p* então $a^{p-1} \equiv 1 (mod\ p)$
- Requer que *p* e a sejam relativamente primos, ou seja $MDC(a, p) = 1$
- Forma alternativa: $a^p \equiv a (mod\ p)$
- $p=5, a=3 \rightarrow a^p = 3^5 = 243 \equiv 3 (mod\ 5) = a (mod\ p)$

Função Totiente de Euler

- A função é escrita ϕ(*n*) e é definida como a quantidade de números menores que *n* e que são relativamente primos a *n*.
- $\Phi(1) = 1$, $\Phi(35) = 24 \rightarrow \{1,2,3,4,6,8,9,11,12,13,16,17,18,19,22,23,24,26,27,29,31,32,33,34\}$
- $\phi(p) = p - 1$
- $\phi(n) = \phi(pq) = \phi(p) x \phi(q) = (p-1)(q-1)$

Teorema de Euler

- Para todo a e n que são relativamente primos
  $a^{\phi(n)} \equiv 1 \pmod{n}$
- a = 3, n = 10, ϕ(10) = 4
  - $a^{\phi(n)} = 3^4 = 81 \equiv 1 \pmod{10} = 1 \pmod{n}$
- Requer que n e a sejam relativamente primos
- Versão alternativa:
  - $a^{\phi(n)+1} \equiv a \pmod{n}$

Teste de Primalidade

- Saber se um numero é primo é importante para afirmar o teorema de Fermat
- Temos que trabalhar com números das ordem de grandeza de 1024 bits
- Algoritmo de Miller-Rabin
  - Determinístico
  - Probabilístico

Teste de Primalidade

- Saber se um numero é primo é importante para afirmar o teorema de Fermat
- Temos que trabalhar com números das ordem de grandeza de 1024 bits
- Algoritmo de Miller-Rabin
  - Determinístico
  - Probabilístico

Background matemático

- Se *p* é um número primo e *a* < *p* é um inteiro positivo

$a^2 \equiv 1 \ (mod\ p)$ **se e somente se**

a ≡ 1(mod p) OU a ≡ - 1 (mod) p = -1 = p-1

a^2 mod p = (a mod p)^2 = (a mod p) * (a mod p)

Background matemático

- Se $p > 2$ é um número primo

$$p - 1 = 2^k * q, \text{ com } k > 0, q \text{ ímpar}$$

- Para todo $1 < a < p - 1$ uma das duas condições é verdadeira:

$$a^q \bmod p = 1 \text{ OU}$$

$$a^q \bmod p, a^{2q} \bmod p, a^{4q} \bmod p, \ldots,$$

$$a^{(2^k-1)q} \bmod p = -1 = p - 1$$

Background matemático

- Prova

$$a^{q} \bmod p,\ a^{2q} \bmod p,\ a^{4q} \bmod p,\ \ldots,$$

$$a^{(2\hat{}k\text{-}1)q} \bmod p,\ a^{(2\hat{}k)q} \bmod p$$

- Sabemos que $a^{(2\hat{}k)q} \bmod p = 1$ pelo teorema de fermat

Background matemático

- Prova

$$a^q \bmod p,\ a^{2q} \bmod p,\ a^{4q} \bmod p,\ \ldots,$$

$$a^{(2^\wedge k\text{-}1)q} \bmod p,\ a^{(2^\wedge k)q} \bmod p$$

- Sabemos que cada elemento da lista é o quadrado do anterior, então:
  - Ou a^q mod p = 1, assim todos os elementos da lista seriam 1
  - Ou um dos elementos que não o último é -1 = p-1

Algoritmo de Miller-Rabin

Test(n) – para n impar

1. ache k, q inteiros k > 0, q impar | ($n-1=2^k q$)
2. rand(int a) → $1 < a < n-1$
3. Se $a^q \bmod n = 1$ → Inconclusivo
4. para j = 0 ate k -1 faca
5. se $a^{2jq} \bmod n \equiv n - 1$ → Inconclusivo
6. Senão → Composto

Algoritmo Miller-Rabin

- Probabilidade de falha menor que $(1/4)^t$
- t = diferentes valores para a
- Repetindo 10 vezes a probabilidade de ser falso primo é de 10^{-6}
- Tem que ser inconclusivo sempre
- Quanto maior t, mais a certeza de que n é primo

Distribuição de Números Primos

- Todos os pares não são primos
- Primos são espalhados na ordem ln(n)
- A probabilidade de se achar um primo é 0.5 ln (n)
- Para se achar um primos de 200 bits temos que tentar 0.5 ln (2^{200}) = 69 na média
- A certeza do Miller-Rabin pode custar caro

Distribuição de Números Primos

Table 8.1 Primes under 2000

2	101	211	307	401	503	601	701	809	0	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	1999
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																			

Teorema Chinês do Resto

- "É possível reconstruir inteiros a partir de seus resíduos módulo um conjunto de número relativamente primos entre si"
- Permite manipular números potencialmente grandes mod M em termos de tuplas de números menores (relativamente primos entre si)
- Z_{10}, mod 2 e mod 5 como fatores, r_2 =0 e r_5=3 tem como solução única x =8

Teorema Chinês do Resto

- 973 mod 1813 → (mod 37, mod 49)
  - 973 mod 37=11, 973 mod 49=42 → (11,42)
- 678 mod 1813 → (mod 37, mod 49)
  - 678 mod 37=12, 678 mod 49=41 → (12, 41)
- (973 + 678) mod 1813 = (23,34) = 1651

Geradores de Números Aleatórios

- Uso:
  - Geração de chaves
  - Geração de parâmetros
  - Controles de sessão
- Aleatoriedade:
  - Distribuição uniforme de 0 e 1 → fácil
  - Independência → difícil
- Estratégia testes de independência similar a Miller-Rabin

Geradores de Números Aleatórios

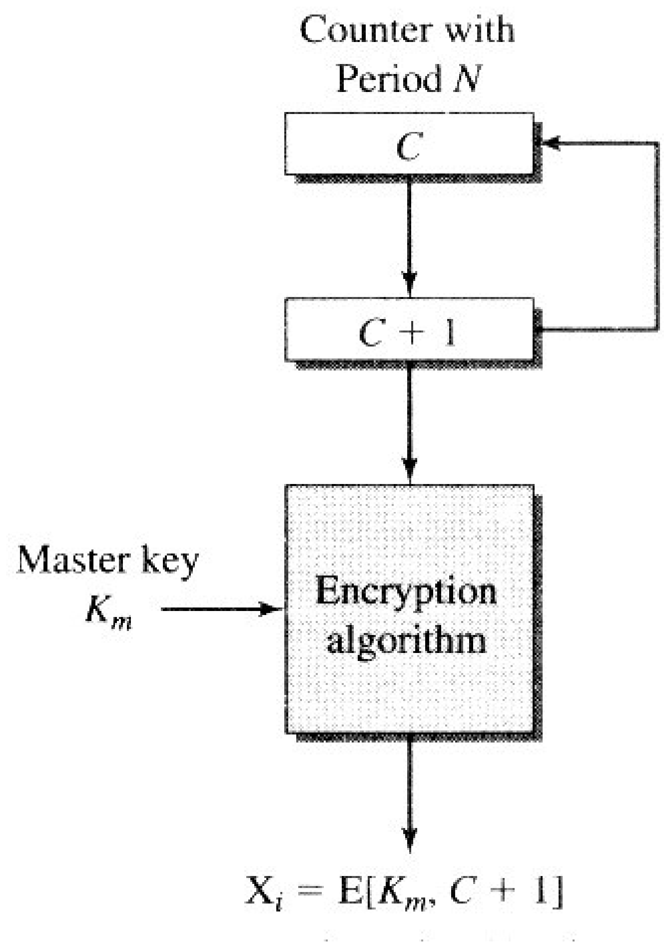
- Não previsibilidade → nonces
- Solução determinística x não determinística
- Geradores Pseudo-Aleatórios:
  - Determinístico (dada um entrada, sempre a mesma saída)
  - Passa por testes de aleatoriedade
  - Aleatoriedade relativa
    - Geradores de Congruência Linear
    - Geradores Criptográficos

Geradores de Congruência Linear

- Modulo m, multiplicador a, incremento c e semente inicial X_0
- $X_{n+1} = (aX_n + c) \bmod m,\ 0 \le X_n < m$
- Dependente na boa escolha de parâmetros
  - m perto ou igual a 2^{31}
  - Um bom a é difícil → um punhado em 2 bilhões pra ter um período próximo a m
    - bom período garante pouca repetição de valores
    - normalmente a = 16807

Geradores Criptográficos

- Cifragem cíclica
  - Bom para chaves de sessão
- DES em OFB com a semente sendo a chave
- ANSI X9.17: 3-DES é um dos mais robustos



Logaritmo Discreto

- Utilizado no Diffie-Hellman e DSA
- $\log_a(b)=x \rightarrow a^x = b$
- É o logaritmo calculado Z_p
- $3^4 \bmod 17 = 13 \rightarrow 3^k = 13 \pmod{17}$
  - 4 é uma solução, mas na verdade inúmeras soluções existem $\rightarrow 4 + 16n = \log_3(13) \bmod 17$
  - Equivalente a $k \equiv 4 \bmod 16$
- Não existe algoritmo eficiente pra isso

Logaritmo Discreto

- Força bruta: elevar a base a maiores potência de k ate achar o valor certo
  - Não existe algoritmo eficiente na computação não-quântica
- Funciona para criptografia, porque é fácil fazer com a exponenciação, mas difícil fazer o logaritmo discreto
- Assimetria equivalente da multiplicação e fatoração de números primos
- Eficiente em outros grupos (curvas elípticas)

Criptografia com Chave Pública

- Criptografia com chave pública NÃO é mais segura que simétrica
- Criptografia com chave pública NÃO surgiu para substituir a simétrica
- Distribuição de chaves NÃO é mais simples na criptografia com chave pública

Criptografia com Chave Pública

- Proposto por Diffie-Hellman (1976)
- Revolução na criptografia
  - Baseada em funções matemáticas
  - Deixa de lado substituição e permutação

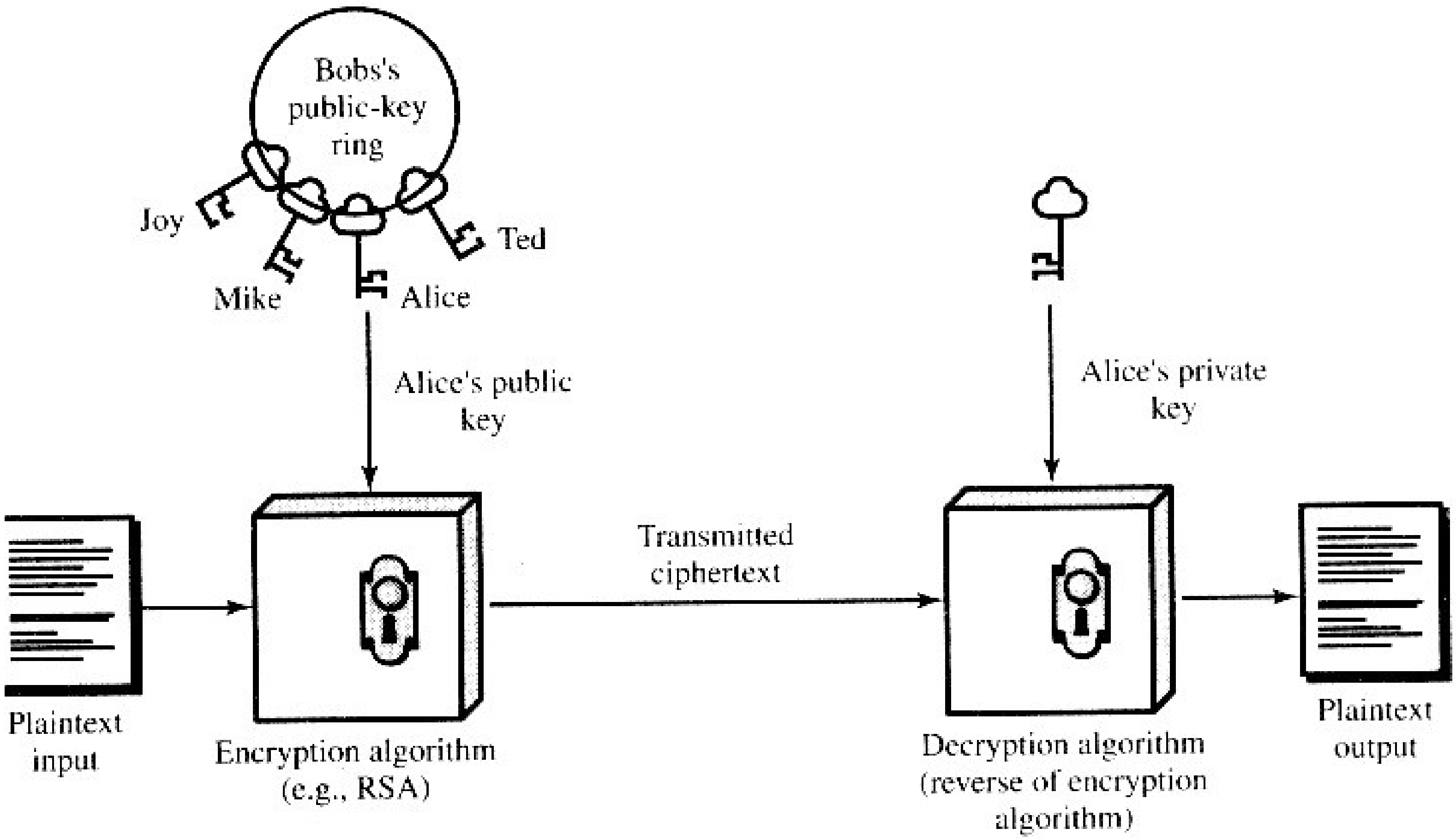
Princípios de Cripto-sistemas de Chave Pública

- Uma chave pública e uma privada
- O que é feito com uma chave poder ser “desfeito” com outra
- Chave assimétrica prove:
  - Confidencialidade, Autenticação, e derivados
- Foi criada para responder ao problema de distribuição de chaves
- Provê assinatura digital

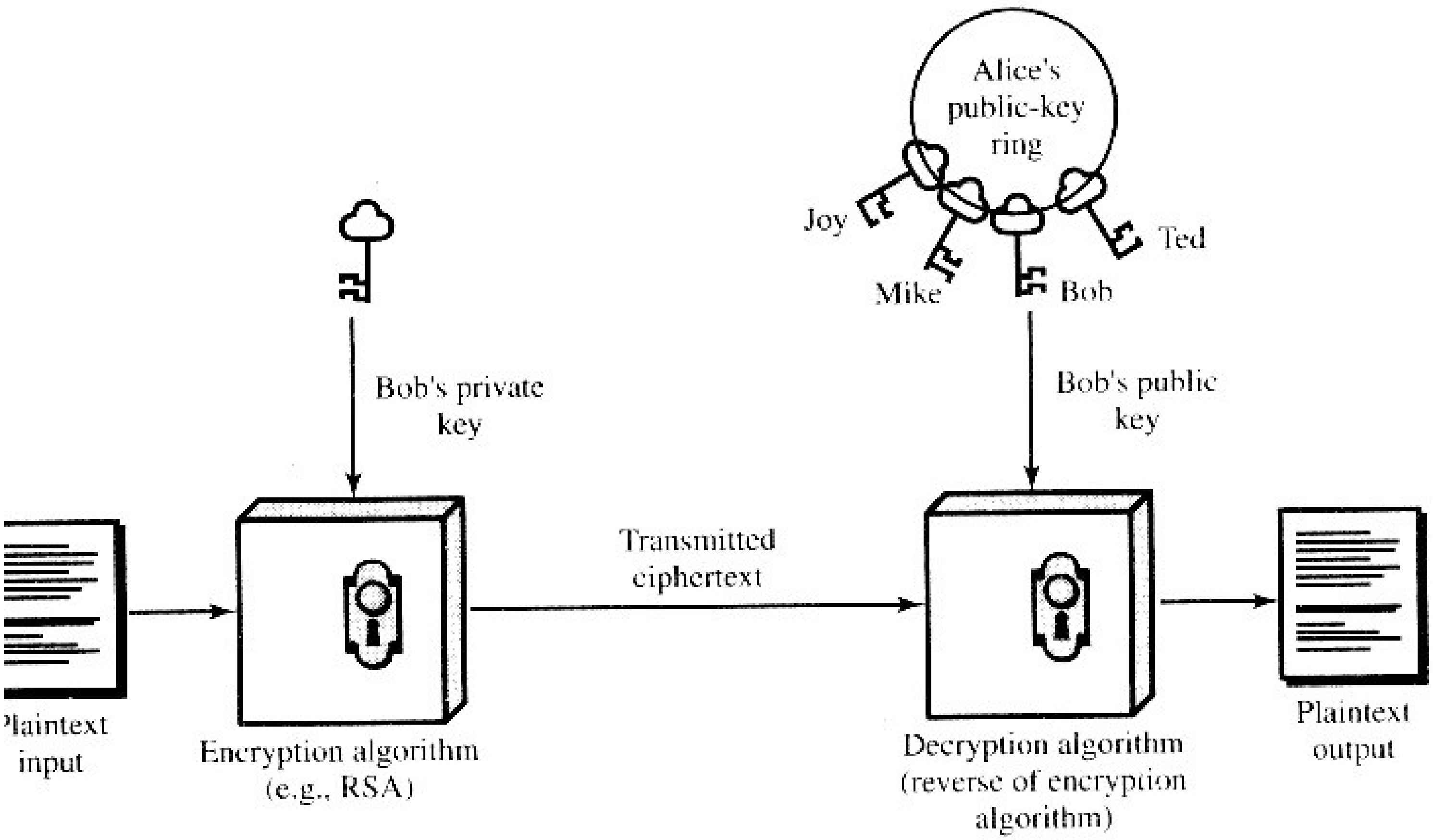
Cripto-sistemas de Chave Pública - Elementos

- Texto claro
- Algoritmo de cifragem
- Par de chaves
- Texto cifrado
- Algoritmo de decifragem
- É computacionalmente impossível determinar a chave privada através da chave pública

Cripto-sistemas de Chave Pública - Cifragem



Cripto-sistemas de Chave Pública - Autenticação

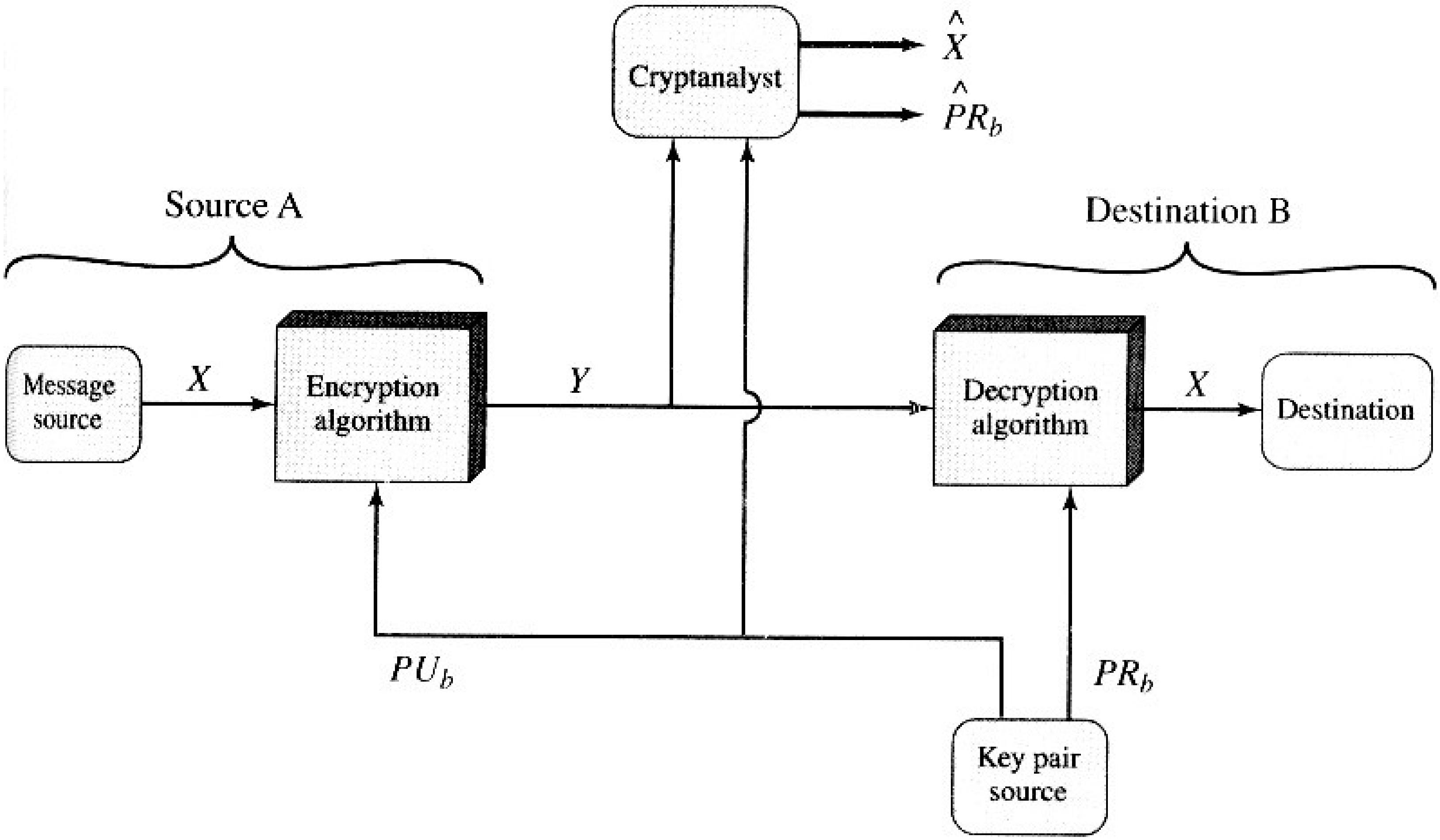


(b) Authentication

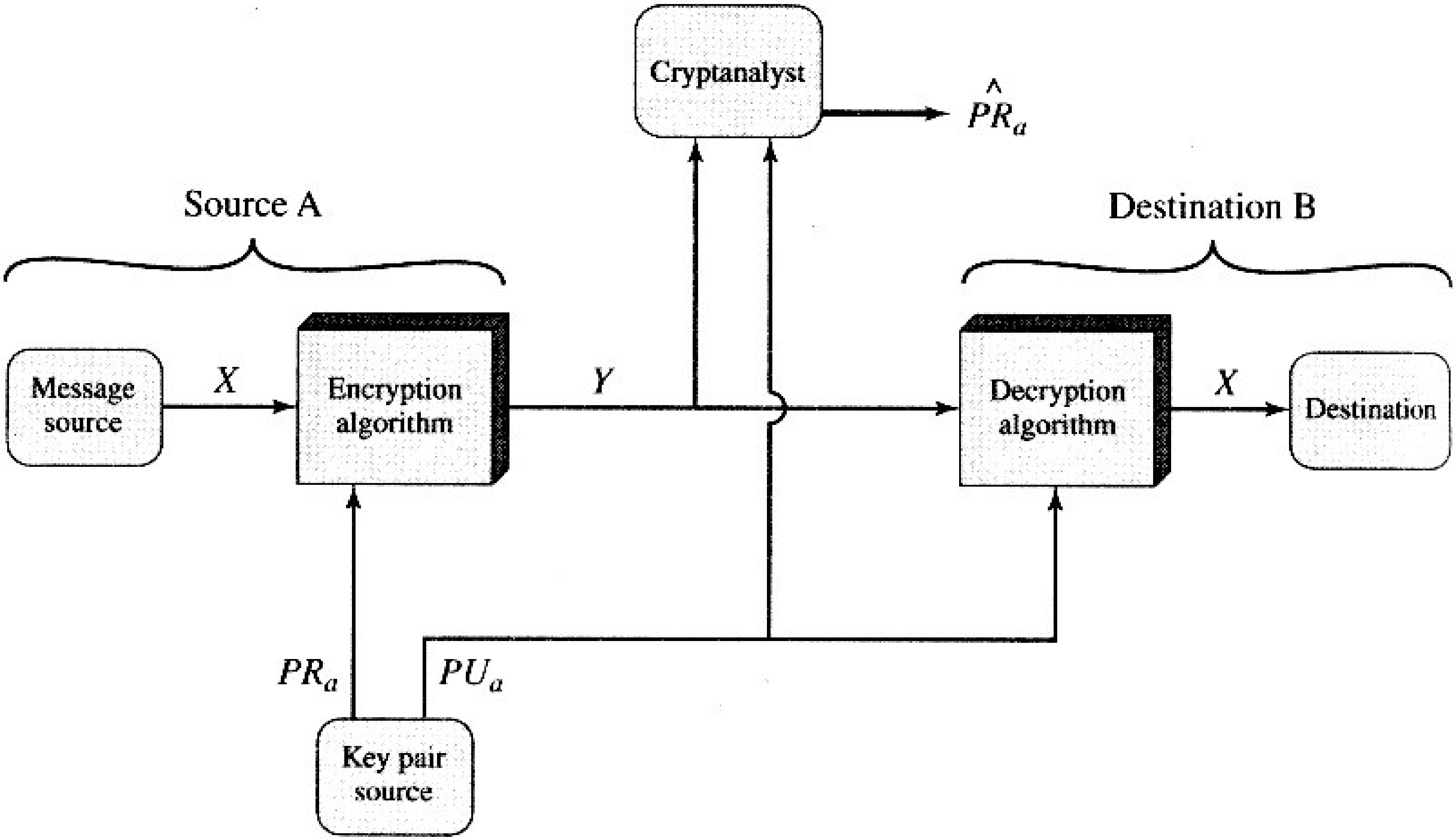
Chave secreta x Chave pública

- Chave Secreta:
  - Funcionamento:
    - Mesmo algoritmo
    - Compartilhamento da chave
  - Segurança:
    - Chave secreta
    - Impossível quebrar sem a chave
- Chave Pública:
  - Funcionamento:
    - Diferentes algoritmos
    - Pares de chaves
  - Segurança:
    - Uma chave secreta
    - Impossível derivar a outra chave
    - Impossível quebrar com uma só chave

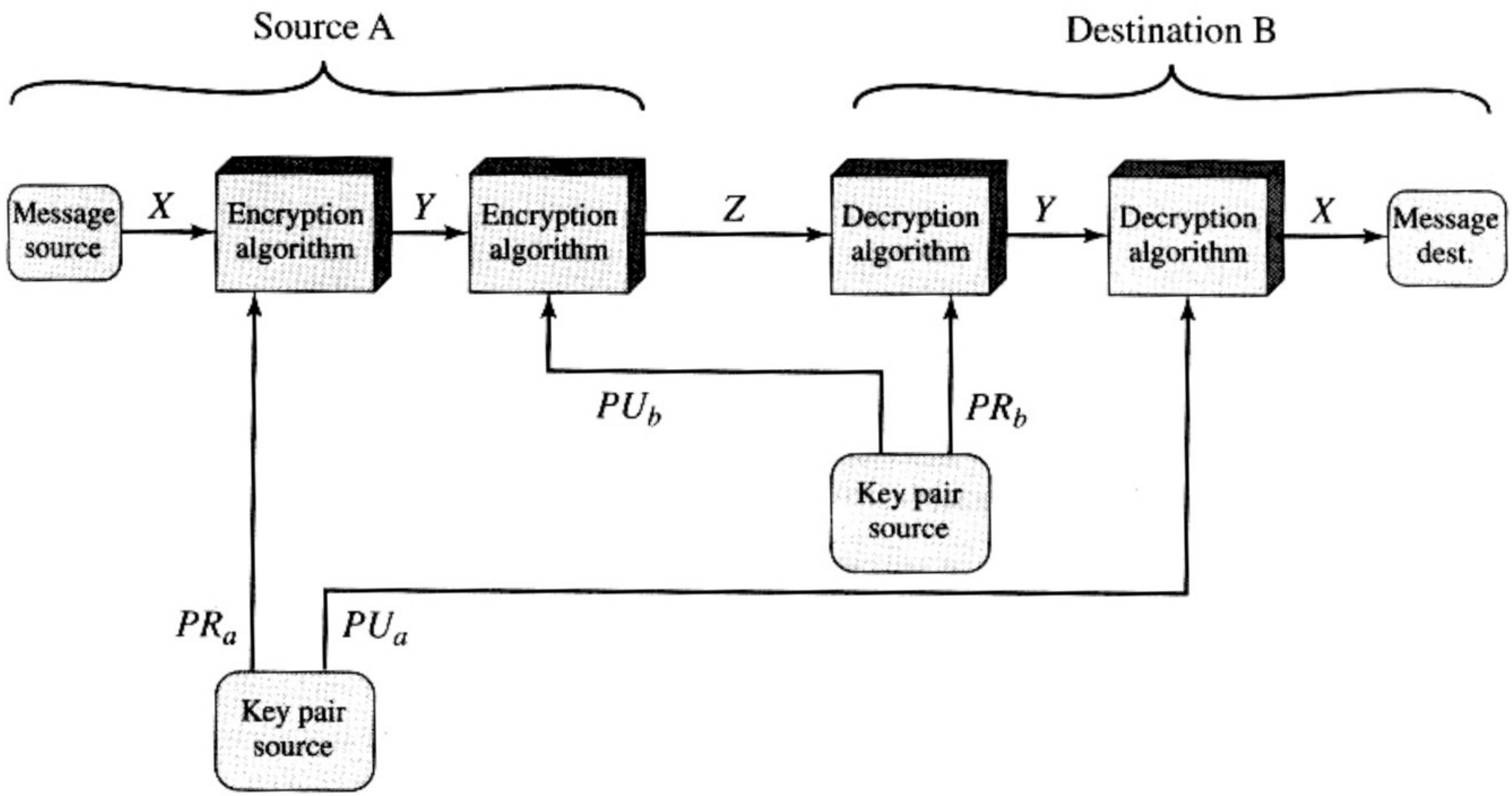
Modelo Cripto-Analítico - Confidencialidade



Modelo Cripto-Analítico - Autenticação



Modelo Cripto-Analítico - Misto



Aplicações de Chave Pública

- Cifragem/Decifragem
- Assinatura Digital
- Troca de Chaves

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Requisitos de Chave Pública

- Fácil (computacionalmente) gerar um par de chaves
- Fácil para o remetente cifrar com a chave pública
- Fácil para o destinatário decifrar com a chave privada
- Impossível determinar Kr a partir de Ku
- Impossível recuperar o texto claro conhecendo Ku e o texto cifrado

Criptoanálise de Chave Pública

- Função de caminho único com "dica"
  – $Y=f(x) \rightarrow$ fácil , $X=f^{-1}(Y) \rightarrow$ impossível
- Fácil quando se conhece a "dica"
- Ataque de força bruta ainda existe
  - Chave pequena -> força bruta
  - Chave grande -> lentidão

RSA

- 1977, Rivest, Shamir e Adelman / MIT
- É o algoritmo mais aceito
  - Base para a Web
  - Base para assinatura digital no Brasil
- Texto claro e texto cifrado são inteiros mod n
- Tamanho do bloco é normalmente 1024 bits (309 dígitos)
- É baseado em exponenciação mod p

RSA - Algoritmo

- Blocos do com valores menores que n
- $C = M^e \bmod n$
- $M = C^d \bmod n = ((M^e)^d) \bmod n = M^{ed} \bmod n$
- Todos conhecem n, o remetente conhece e, o destinatário conhece d
- Chave Pública → (n, e)
- Chave Privada → (n, d)

RSA - Requisitos

- e, d, n são escolhidos pra satisfazer M^{ed} mod n = M para todo M < n
- Para isso "e" e "d" devem ser multiplicativas inversas mod $\phi(n) \rightarrow$ e.d mod $\phi(n) = 1$
  - $e.d \equiv 1 \bmod \phi(n) \rightarrow d \equiv e^{-1} \bmod \phi(n)$
  - $gcd(\phi(n),d)=1$
  - $gcd(\phi(n),e)=1$

RSA – Requisitos Práticos

- p, q primos: privados e escolhidos
- $n = p.q$: público e calculado
- $e \mid gcd(\phi(n),e) = 1$ ˆ $1 < e < \phi(n)$: público e escolhido
- $d \equiv e^{-1}(mod\ \phi(n))$ privado e calculado
- Chave pública (e, n)
- Chave privada (d, n)

RSA na Prática

- $p = 17$ e $q = 11$
- $n =$ porque $= 17 \times 11 = 187$
- $\phi(n) = (p - 1)(q - 1) = 16 \times 10 = 160$
- $e = 7$, $gcd(160, 7) = 1 \text{ ˆ } 1 < 7 < 160$
- $d \mid de \equiv 1 (\mathrm{mod}\ 160) \text{ ˆ } d < 160 \rightarrow d = 23$
  - $23 \times 7 = 161$
- $Ku = \{7, 187\}$, $Kr = \{23, 187\}$

RSA – Cifragrem/Decifragem Prática

- Texto Claro = 88
- 88^7 mod 187 = 11
- Texto cifrado = 11
- 11^{23} mod 187 = 88
- Computacionalmente intensivo de fazer com números grande

RSA - Considerações Computacionais

- Exponenciação mod n requer truques matemáticos
  - $88^7 \bmod n = (88^1 * 88^2 * 88^4) \bmod n$
- O e acaba sendo fixo em primos como: 65537 ($2^{16} + 1$), 17 ou 3, e sofre ataques se utilizado muitas vezes
- d tem que ser grande para evitar força bruta
- Gerar chaves pode ser demorado pois precisamos do M-R várias vezes em um número muito grande

Segurança do RSA

- Força Bruta:
  - Todas as possíveis chaves
  - ↑ Tamanho ↓Eficiência
- Ataques Matemáticos:
  - Todos equivalente a fatorar p.q (achar o $\phi(n)$)
- Ataques de Tempo
  - Adivinhação da chave privada pelo tempo gasto na decifragem

RSA – Ataques matemáticos

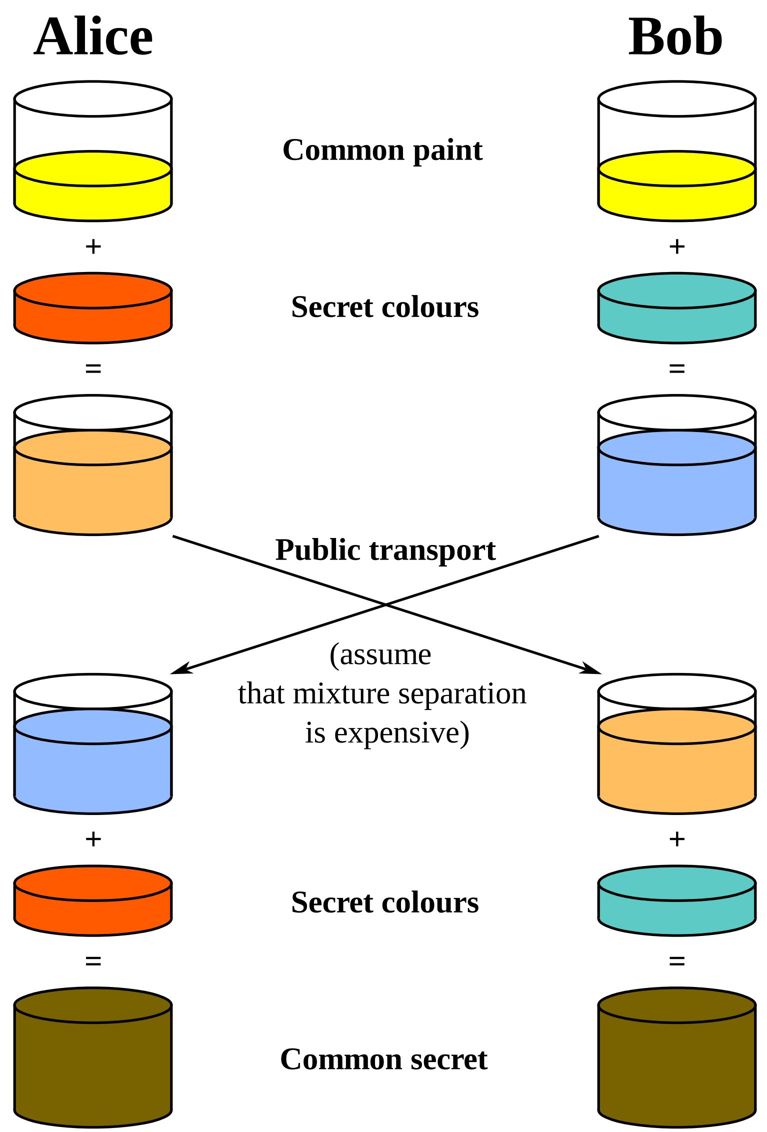
Number of Decimal Digits	Approximate Number of Bits	Date Achieved	MIPS-years	Algorithm
100	332	April 1991	7	Quadratic sieve
110	365	April 1992	75	Quadratic sieve
120	398	June 1993	830	Quadratic sieve
129	428	April 1994	5000	Quadratic sieve
130	431	April 1996	1000	Generalized number field sieve
140	465	February 1999	2000	Generalized number field sieve
155	512	August 1999	8000	Generalized number field sieve
160	530	April 2003	—	Lattice sieve
174	576	December 2003	—	Lattice sieve
200	663	May 2005	—	Lattice sieve

http://en.wikipedia.org/wiki/RSA_Factoring_Challenge#The_prizes_and_records

Troca de Chaves Diffie-Hellman

- Primeiro algoritmo publicado de chave pública
- Objetivo: Troca segura de parâmetros para estabelecer uma chave de sessão
- O algoritmo depende da dificuldade de calcular logaritmos discretos
- Raiz primitiva → $a \bmod p \ldots a^{p-1} \bmod p$
- $b \equiv a^i \pmod{p}$ onde $0 \leq i \leq p$ → $\mathrm{dlog}_{a,p}(b)$

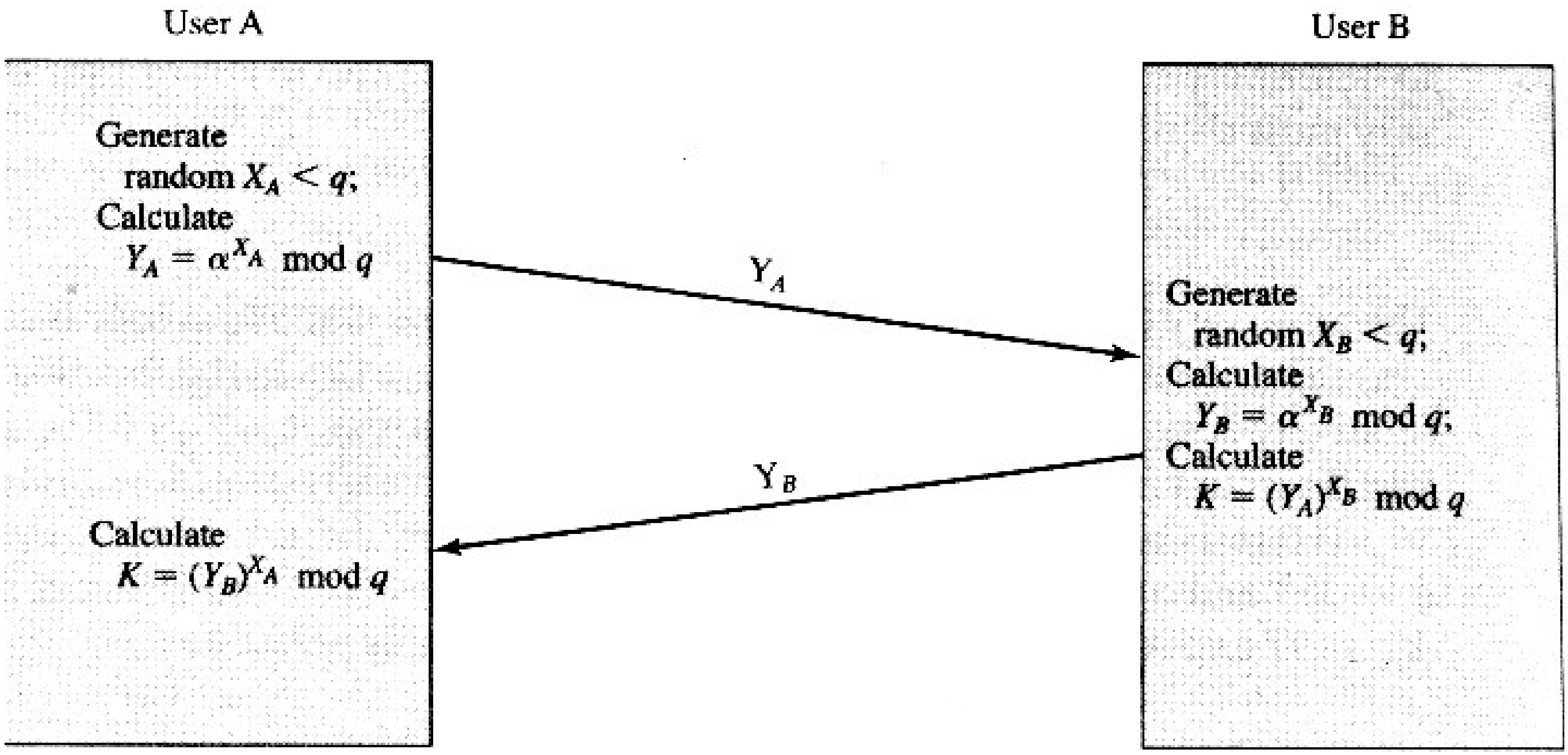
Troca de Chaves Diffie-Hellman



Diffie-Hellman - Algoritmo

- Parâmetros:
  - q numero primo, α raiz primitiva de q → públicos
  - Xa e Xb < q números aleatórios secretos
  - Geração de chave:
    - $Ya = \alpha^{Xa} \bmod q$ e $Yb = \alpha^{Xb} \bmod q$
  - Segredo:
    - $K = (Yb)^{Xa} \bmod q$
    - $K = (Ya)^{Xb} \bmod q$
- O adversários só sabe q, α ,Ya e Yb

Protocolos de Troca da Chaves



Diffie-Hellman - Exemplo

- q = 353, α = 3, Xa = 97 e Xb = 233
- A computa:
  - Ya = 3^{97} mod 353 = 40
- B computa:
  - Yb = 3^{233} mod 353 = 248
- A deriva:
  - K = 248^{97}mod 353 = 160
- B deriva:
  - K = 40^{233} mod 353 = 160

Diffie-Hellman – Ataque MITM

- C gera Xc1, Xc2 e computa Yc1 e Yc2
- C intercepta Ya de A para B, manda como A Yc1 pra B e calcula $K2=(Ya)^{Xc2} \bmod q$
- B recebe Yc1, calcula $K1= (Yc1)^{Xb} \bmod q$
- B manda Yb para A, C intercepta, manda Yc2 pra A e calcula $K1=(Yb)^{Xc1} \bmod q$
- A recebe Yc2 e calcula $K2=(Yc2)^{Xa} \bmod q$
- C atua como proxy

Autenticação de Mensagens

- Garantia de que a mensagem esta íntegra e que foi enviada por alguém válido
- Cifragem garante autenticação
  - Somente as duas partes conhecem o segredo
  - Se B recebe uma mensagem cifrada, então A deve ter enviado
  - Não é prático quando o texto claro não é legível (seqüência aleatória de bits)

Autenticação de Mensagens

- MAC é um algoritmo de verificação que requer uma chave e garante autenticação
  - O modelo mais popular utiliza Hash
  - Outro modelo popular utiliza cifradores de bloco
- Assinatura eletrônica garante autenticação de mensagens
  - Garante integridade e autenticidade da fonte
  - Também garante não repúdio

Autenticação - Ataques

- Mascaramento:
  – Origem fraudulenta
- Modificação de conteúdo:
  – Alteração da carga da mensagem
- Modificação de seqüência:
  – Reordenamento de mensagens
- Modificação de tempo:
  – Replay e delay

Funções de Autenticação

- Autenticadores:
  - Cifragem
  - Message Authentication Codes
  - Funções HASH

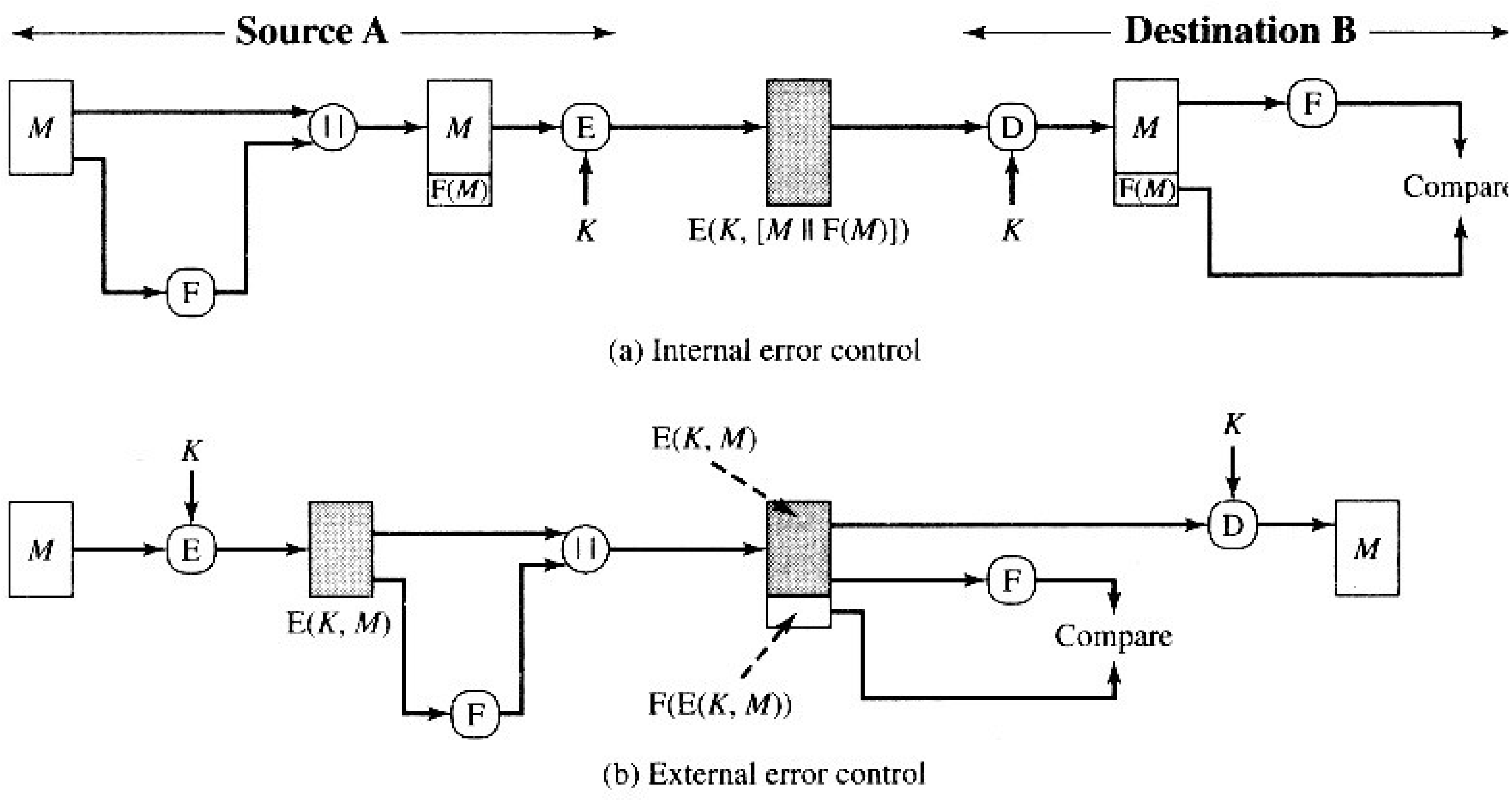
Autenticação - Cifragem

- Provê autenticação usando algoritmos criptográficos
- Autenticação por cifragem pode ser dividida em:
  - Simétrica
  - Assimétrica
- A autenticação é baseada na manutenção dos segredos
- Chaves que devem ser protegidas

Autenticação – Cifragem Simétrica

- Somente A e B compartilham a chave K
- Se um texto recebido por A decifra para uma mensagem inteligível usando K, A pode inferir que a mensagem veio de B
- Senão for legível, deve conter alguma estrutura que seja facilmente reconhecida:
  - Detecção de erro
  - Hash

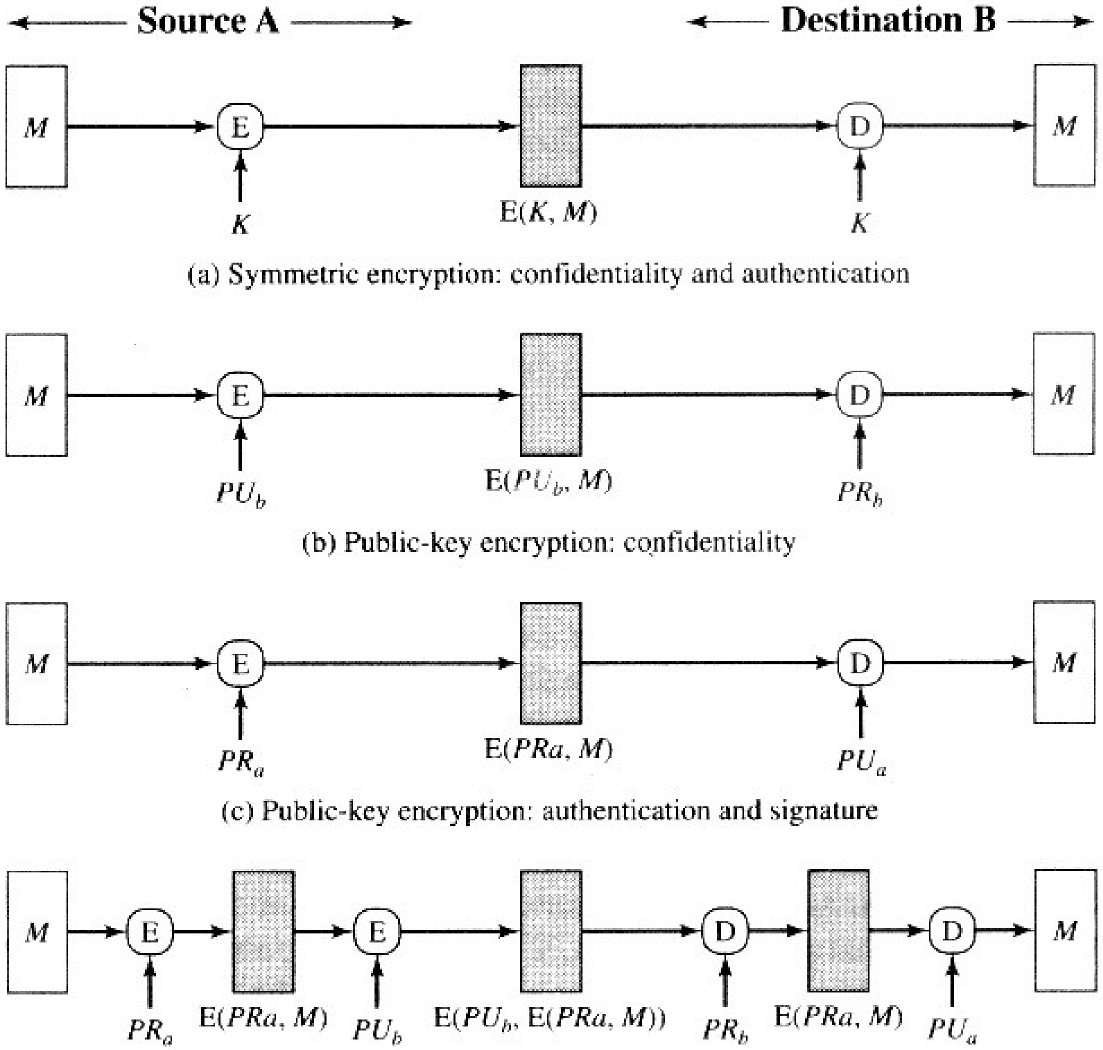
Autenticação – Cifragem Simétrica com Integridade



Autenticação – Cifragem Assimétrica

- Autenticação pelo uso da chave privada
- A operação pode ser desfeita pela chave pública
- Se relacionarmos B com a chave pública que decifra uma mensagem recebida por A, este autentica B pela posse da chave privada
- Integridade normalmente feita por HASH
    - Alta complexidade de cifragem para textos grandes

Autenticação – Cifragem em Exemplos



(a) Symmetric encryption: confidentiality and authentication

(b) Public-key encryption: confidentiality

(c) Public-key encryption: authentication and signature

(d) Public-key encryption: confidentiality, authentication, and signature

Autenticação – Propriedades da Cifragem

A → B: E(*K*, *M*)

- Provides confidentiality
  —Only A and B share *K*
- Provides a degree of authentication
  —Could come only from A
  —Has not been altered in transit
  —Requires some formatting/redundancy
- Does not provide signature
  —Receiver could forge message
  —Sender could deny message

(a) Symmetric encryption

Autenticação – Propriedades da Cifragem

A → B: E(PU_b, M)

- Provides confidentiality
  —Only B has PR_b to decrypt
- Provides no authentication
  —Any party could use PU_b to encrypt message and claim to be A

(b) Public-key (asymmetric) encryption: confidentiality

A → B: E(PR_a, M)

- Provides authentication and signature
  —Only A has PR_a to encrypt
  —Has not been altered in transit
  —Requires some formatting/redundancy
  —Any party can use PU_a to verify signature

(c) Public-key encryption: authentication and signature

Autenticação – Propriedades da Cifragem

A → B: $E(PU_b, E(PR_a, M))$

- Provides confidentiality because of PU_b
- Provides authentication and signature because of PR_a

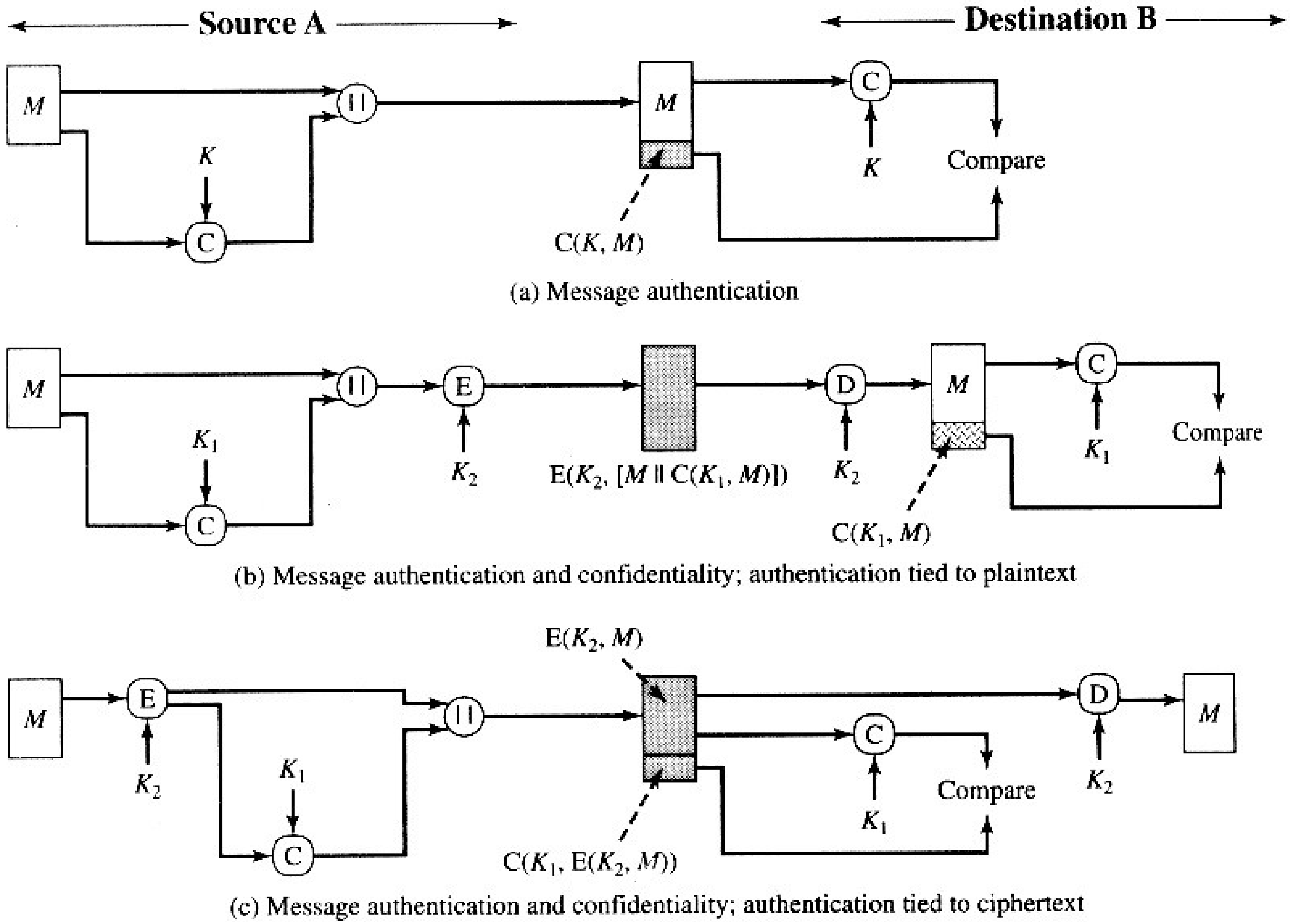
Autenticação – Códigos de Autenticação de Mensagens

- Resumo da mensagem baseado em chave simétrica
  - MAC = C(K,M)
- É similar a cifragem mas não tem reversão
- Calcula-se dos dois lados usando os mesmos parâmetros para confirmar

Códigos de Autenticação de Mensagens - Quando Usar

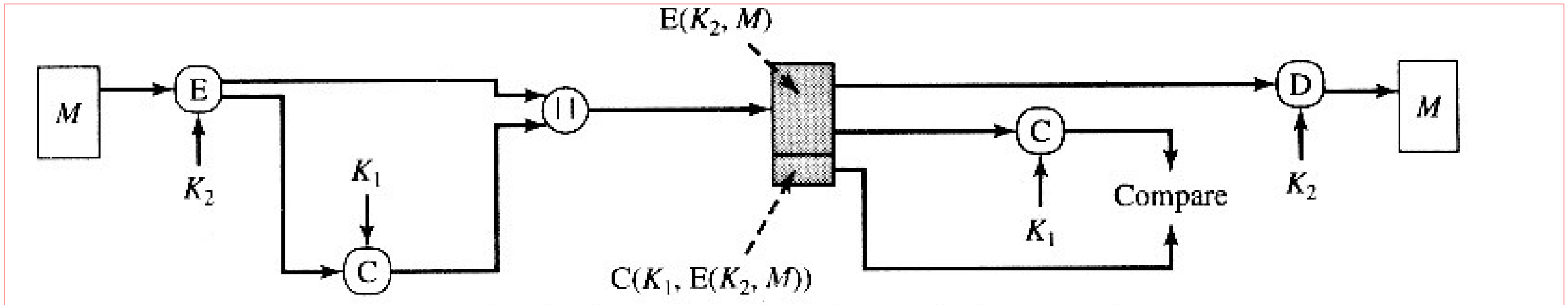
- Mensagem enviada a vários destinatários, somente um verifica a integridade
- Mensagem muito grande para ser cifrada (processo lento), usa-se checagem MAC seletiva
- Verificação de integridade de programas
- Não é necessário sigilo
- Autenticação após decifragem
- <u>Verificação de integridade autêntica andes da decifragem.</u>

Códigos de Autenticação de Mensagens - Uso



Códigos de Autenticação de Mensagens - Uso

Encrypt and Mac!

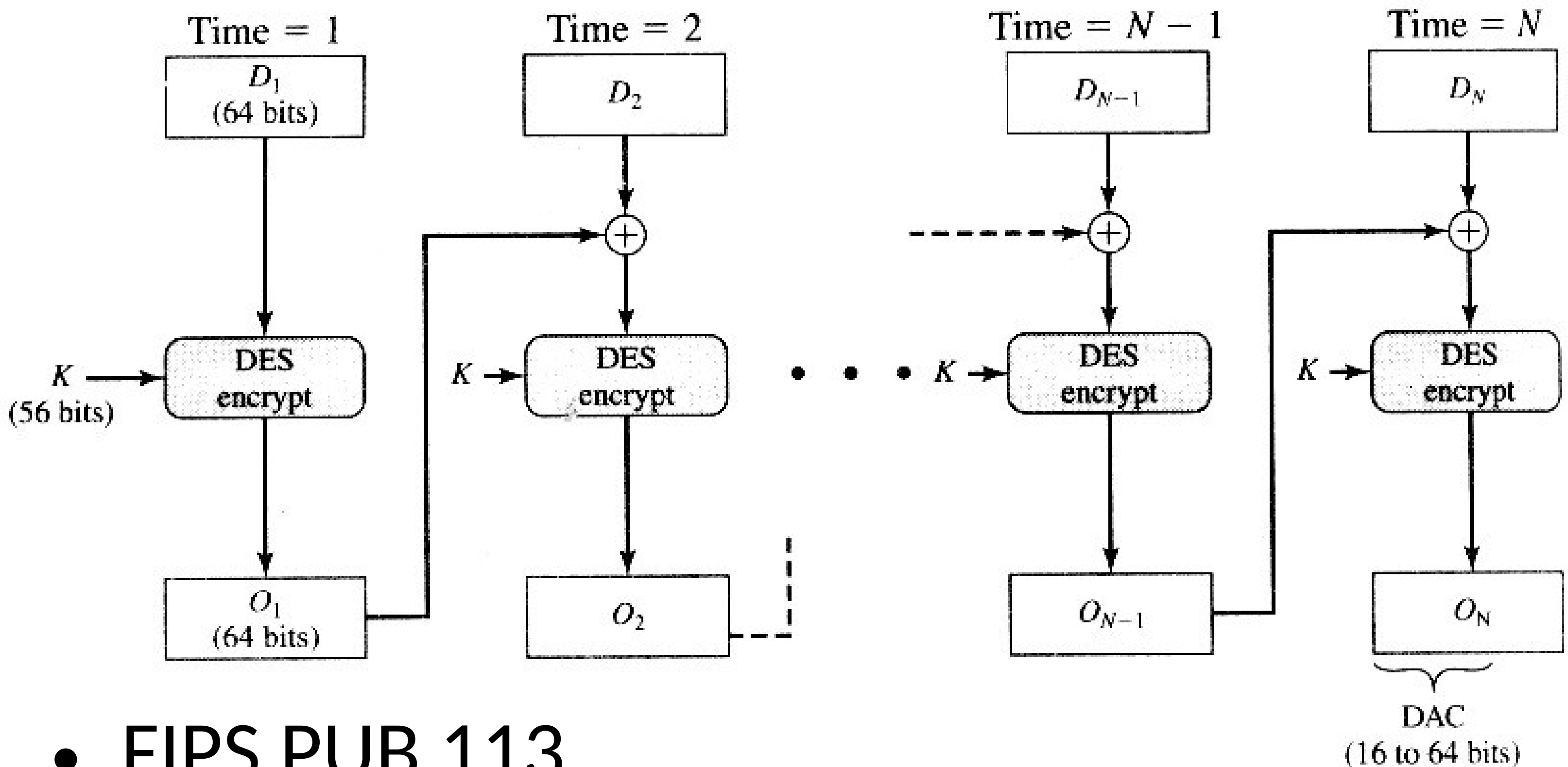


(c) Message authentication and confidentiality; authentication tied to ciphertext

MAC – Descrição/Requisitos

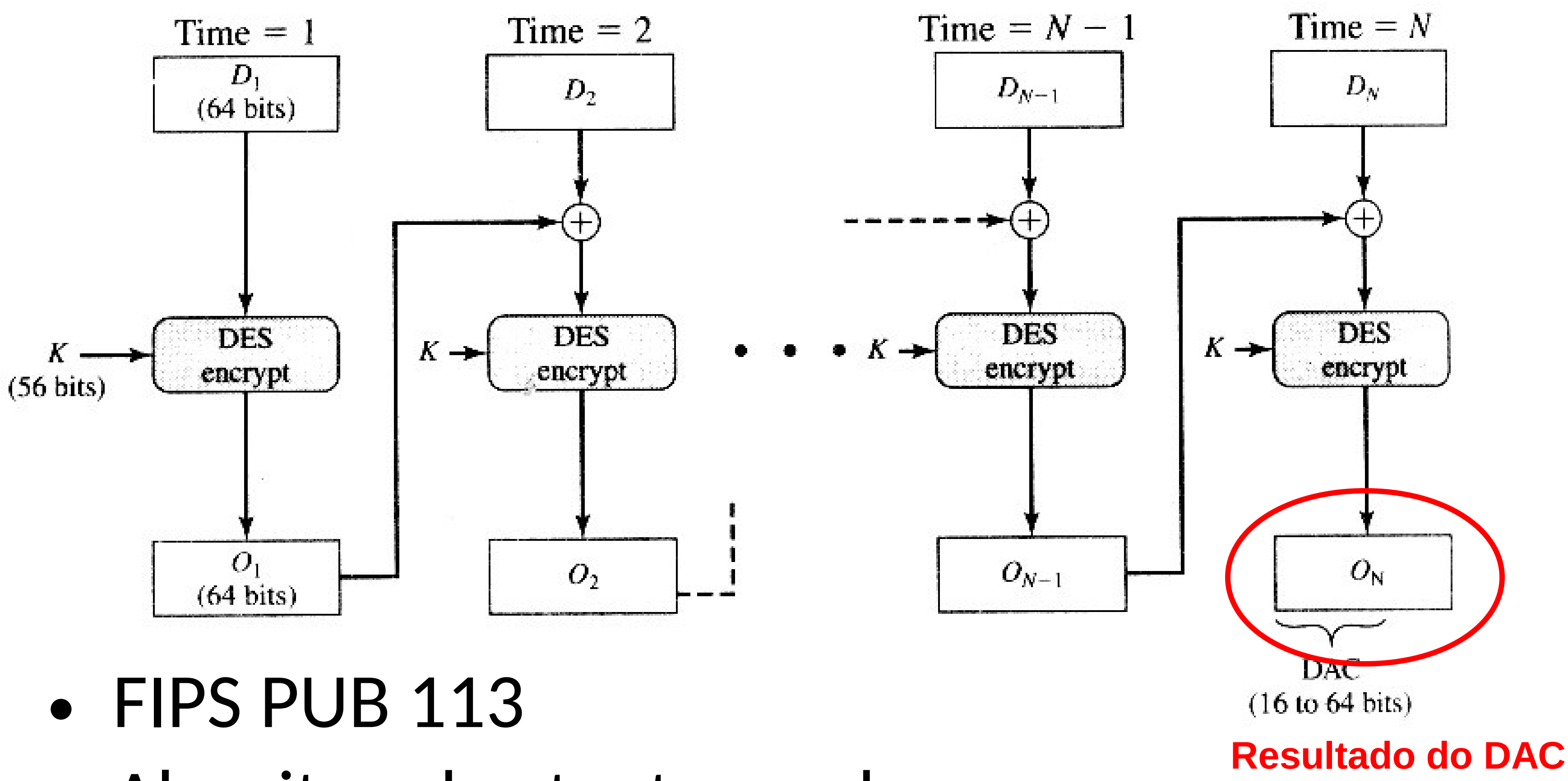
- Função de caminho único
- Requerimentos:
  - C(K,M') = C(K,M) impossível para M,M' escolhido
  - Distribuição uniforme C(K,M') = C(K,M) $\rightarrow$ Probabilidade = 2^{-n}, n = tamanho do MAC
  - Efeito avalanche

DAC – MAC baseado em DES



- FIPS PUB 113
- Algoritmo bastante usado

DAC – MAC baseado em DES



- FIPS PUB 113
- Algoritmo bastante usado

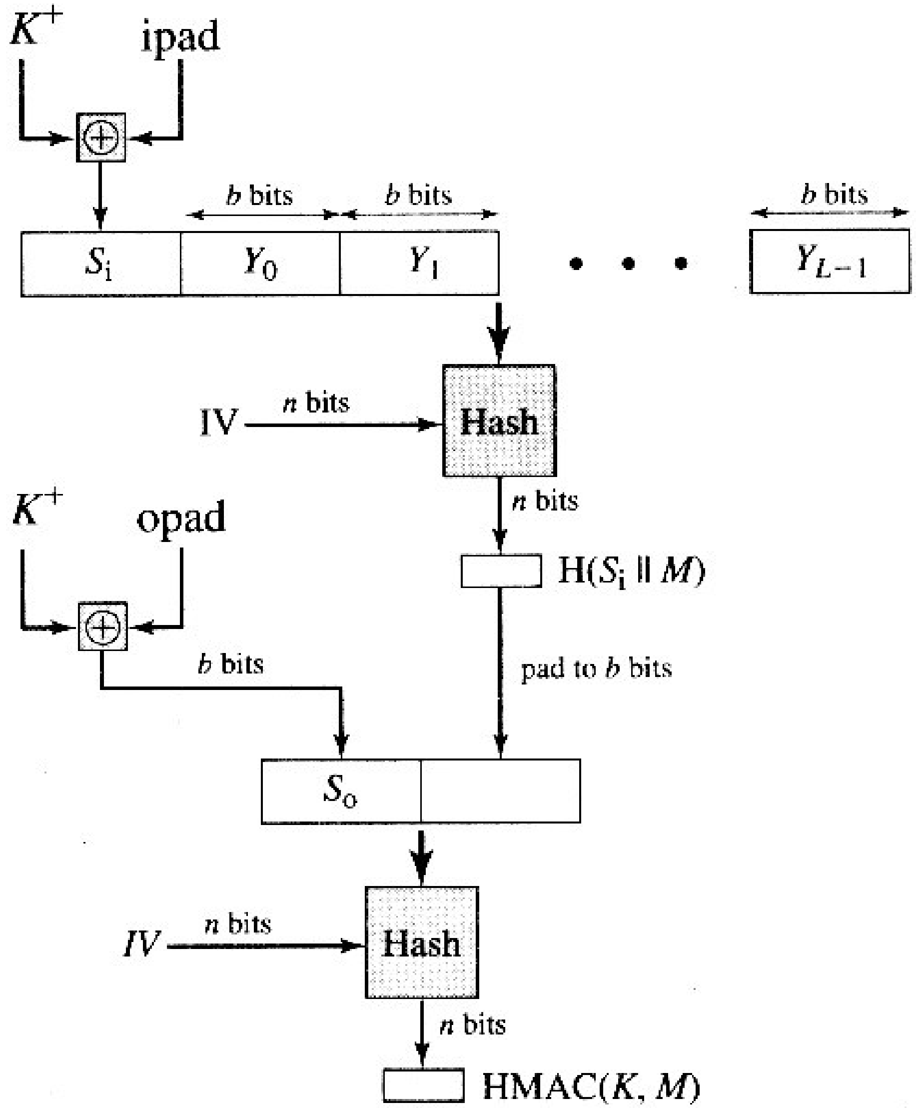
HMAC

- MAC baseado em função HASH
- Objetivos:
  - Mais rápido que cifragem
  - Funções HASH amplamente disponíveis
- RFC 2104 /FIPS 198 → como adicionar um chave a um HASH
- Usado em SSL e IPSEC

HMAC – Objetivos de Projeto

- Usar funções HASH sem modificação
- Permitir trocar a função HASH
- Preservar a performance do HASH
- Usar chave de maneira simples
- Ter toda analise criptográfica baseada no função HASH

HMAC - Estrutura



HMAC - Estrutura

- $HMAC(K,M)=H[(K^{+} \otimes opad)||H[(K^{+} \otimes ipad)||M]]$
  - b→ tamanho do bloco em bits
  - K→ Chave, K^{+}→ Chave estendida até b
  - ipad = 0x36 repetido b/8
  - opad = 0x5C repetido b/8
  - IV → valor de inicialização do HASH
  - opad e ipad geram bits alternados na chave

HMAC Pseudo-Código

```
function hmac (key, message)
    if (length(key) > blocksize) then
        key = hash(key) // keys longer than blocksize are shortened
    end if
    if (length(key) < blocksize) then
        key = [0x00 * (blocksize - length(key)) || key] // keys shorter than blocksize are zero-padded
    end if
      o_key_pad = [0x5c * blocksize] ⊕ key // Where blocksize is that of the underlying hash function
       i_key_pad = [0x36 * blocksize] ⊕ key
      return hash(o_key_pad ∥ hash(i_key_pad ∥ message))
end function
```

Assinatura Digital

- É um mecanismo de autenticação que possibilita o criador da mensagem ser identificado
- Prova de não-repúdio
- Pode ser direta ou arbitrada

Assinatura Digital - Requisitos

- A assinatura deve depender de cada bit da mensagem
- Deve usar algo único do criador
- Deve ser fácil de produzir, reconhecer e verificar
- Dever ser computacionalmente não forjável
- Dever ser possível reter uma cópia

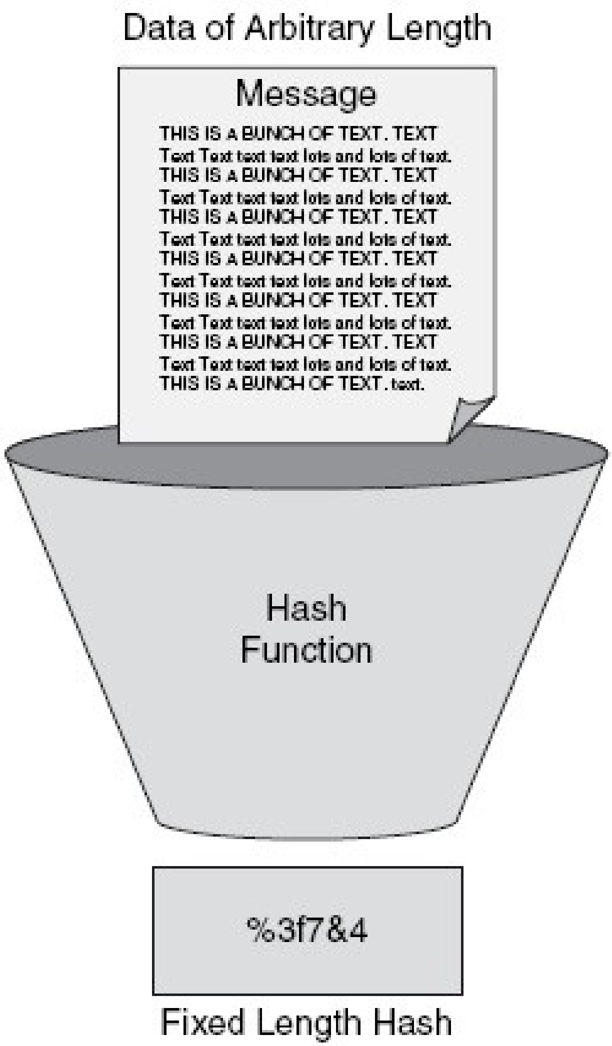
Assinatura Digital Direta

- Envolve só origem e destino
- Cifragem do hash com a chave privada
- Validade atrelada a chave privada
- Negar é alegar a perda da chave
- É normalmente incluído carimbo de tempo

Assinatura Digital Arbitrada

- Tentar resolver o problema da assinatura direta
- Envolve origem, destino e arbitro
- O arbitro checa a mensagem e assina junto dando o seu carimbo de tempo
- O arbitro provê uma prova de verificação
- O arbitro deve ser confiável por ambos

Integridade – Funções HASH



Integridade – Funções HASH

- São similares a MAC mas não tem chaves
- Provê propriedades como efeito avalanche
- Prove uma camada de integridade diferente da autenticação

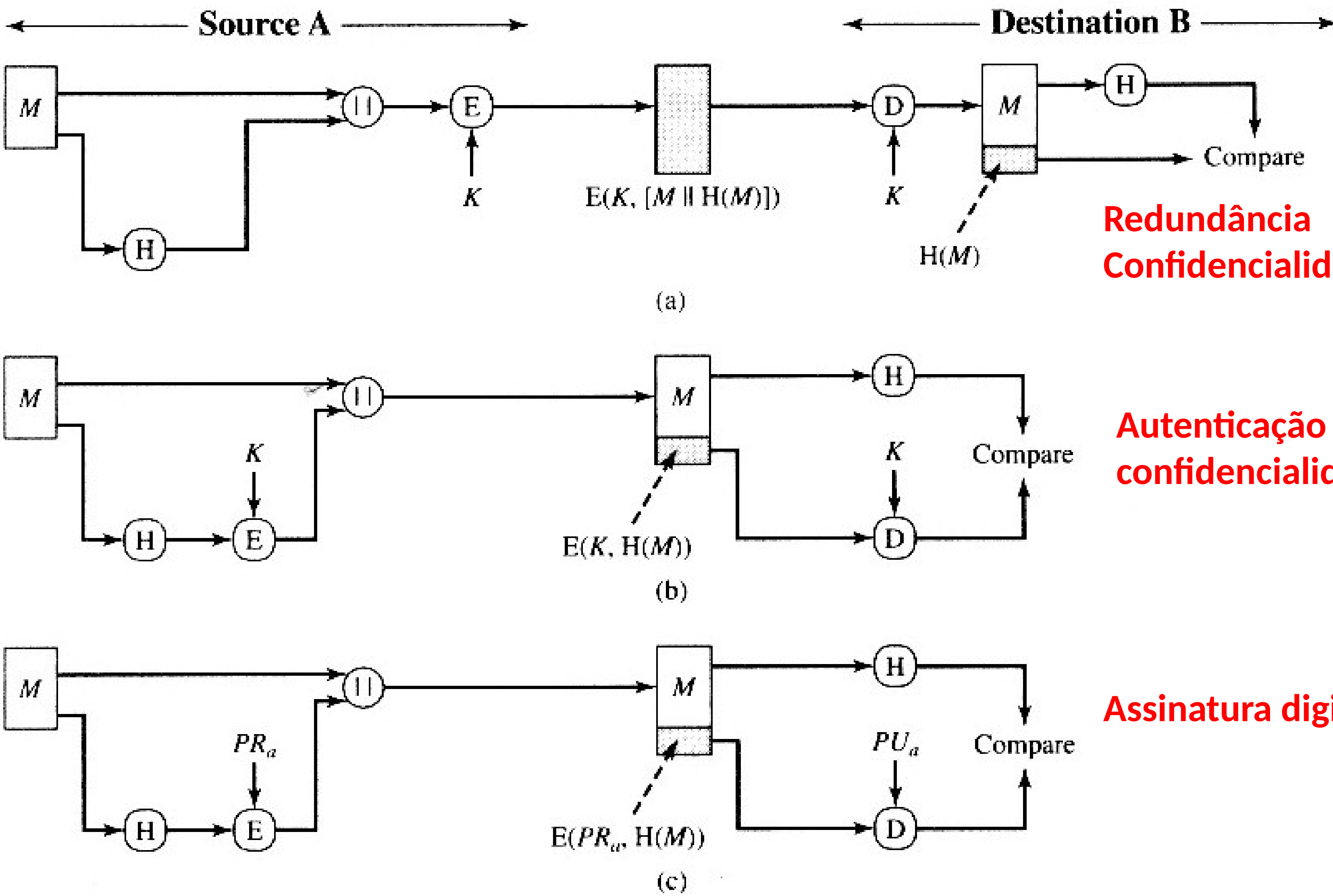
Integridade – Funções HASH

- Computacionalmente não praticável achar:
  - Um dado que coincida com um hash pré-especificado (não é inversível)
  - Dois dados que tenham o mesmo hash (colisão)
- Melhor algoritmo para ambos: força bruta

HASH – Descrição/Requisitos

- Função de caminho único, M variável, H(M) Fixo
- Produz uma impressão digital de um arquivo
- Requisitos:
  - Fácil de computar H(M) para qualquer M
  - É impossível achar M tendo H(M) → caminho único
  - Dado M1 e M2 não deve ser possível computar H(M1) = H(M2) para M1 ≠ M2
- Pseudo-aleatoriedade

Funções HASH - Usos

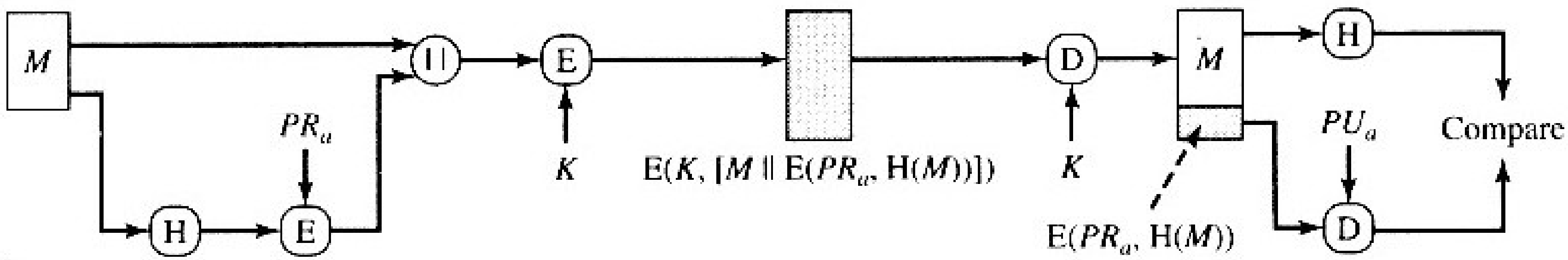


Redundância
Confidencialidade

Autenticação sem confidencialidade

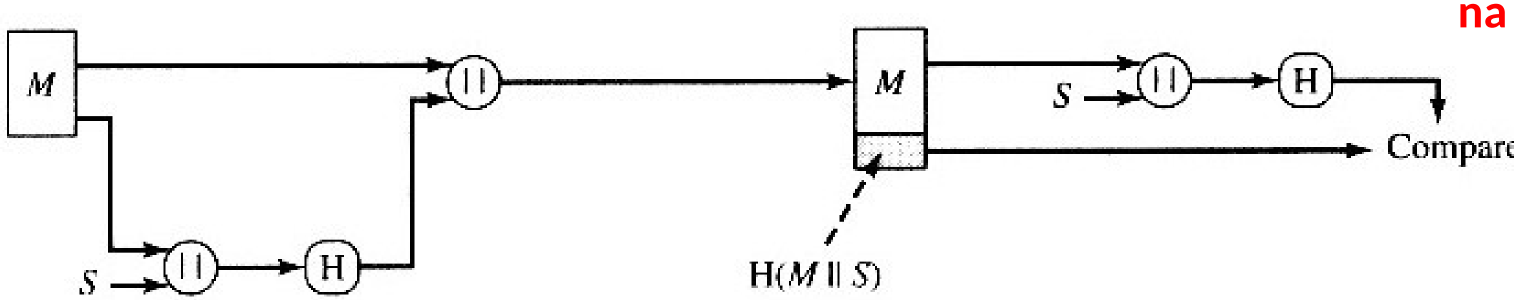
Assinatura digital

Funções HASH - Usos



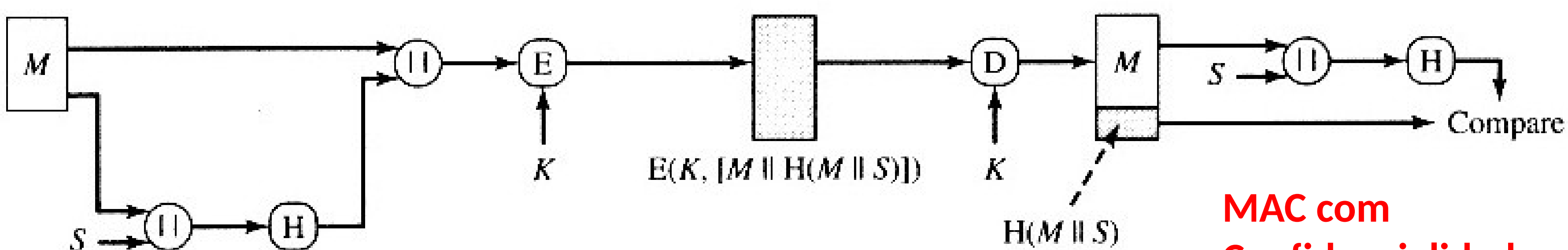
(d)

Confidencialidade na assinatura



(e)

MAC



(f)

MAC com Confidencialidade

Funções HASH – Quando Usar

- Uso em funções MAC
- Verificação de integridade
- Indexação de arquivos (estruturas de dados)
- Armazenamento de senhas
  - salted password hashing
  - *Token* de acesso

Paradoxo do Aniversário

- Consedere uma sala com 30 alunos
- Professor escolhe uma data
  - ~8% de chance de um aluno ter nascido nesta data.
- Professor solicita data de nascimento dos alunos
  - 70% de chance de dois alunos terem a mesma data de nascimento

Obs: Ano com 365 dias

Paradoxo do Aniversário

- Em um grupo de 23 pessoas existe uma probabilidade de 50% para que duas destas façam aniversário no mesmo dia.
- A chance de encontrar um valor repetido em um conjunto de 0 a N-1 excede 50% depois de aprox. $\sqrt{N}$ tentativas

Paradoxo do Aniversário

- A está preparado para assinar x
- Atacante gera $2^{m/2}$ variações de uma mensagem x com o mesmo significado (m é o tamanho do hash)
- Atacante gera $2^{m/2}$ variações fraudulentas y
  - A probabilidade de encontrar algum y com hash igual ao de algum x é maior que 50%
  - Se oferece a versão variada (x) para assinatura e se usa a versão fraudulenta (algum y).

Paradoxo do Aniversário M2^{37}

Dear Anthony,

$\begin{Bmatrix}\text{This letter is}\\ \text{I am writing}\end{Bmatrix}$ to introduce $\begin{Bmatrix}\text{you to}\\ \text{to you}\end{Bmatrix}$ $\begin{Bmatrix}\text{Mr.}\\ \text{--}\end{Bmatrix}$ Alfred $\begin{Bmatrix}\text{P.}\\ \text{--}\end{Bmatrix}$

Barton, the $\begin{Bmatrix}\text{new}\\ \text{newly appointed}\end{Bmatrix}$ $\begin{Bmatrix}\text{chief}\\ \text{senior}\end{Bmatrix}$ jewellery buyer for $\begin{Bmatrix}\text{our}\\ \text{the}\end{Bmatrix}$

Northern $\begin{Bmatrix}\text{European}\\ \text{Europe}\end{Bmatrix}$ $\begin{Bmatrix}\text{area}\\ \text{division}\end{Bmatrix}$ He $\begin{Bmatrix}\text{will take}\\ \text{has taken}\end{Bmatrix}$ over $\begin{Bmatrix}\text{the}\\ \text{--}\end{Bmatrix}$

responsibility for $\begin{Bmatrix}\text{all}\\ \text{the whole of}\end{Bmatrix}$ our interests in $\begin{Bmatrix}\text{watches and jewellery}\\ \text{jewellery and watches}\end{Bmatrix}$

in the $\begin{Bmatrix}\text{area}\\ \text{region}\end{Bmatrix}$ Please $\begin{Bmatrix}\text{afford}\\ \text{give}\end{Bmatrix}$ him $\begin{Bmatrix}\text{every}\\ \text{all the}\end{Bmatrix}$ help he $\begin{Bmatrix}\text{may need}\\ \text{needs}\end{Bmatrix}$

to $\begin{Bmatrix}\text{seek out}\\ \text{find}\end{Bmatrix}$ the most $\begin{Bmatrix}\text{modern}\\ \text{up to date}\end{Bmatrix}$ lines for the $\begin{Bmatrix}\text{top}\\ \text{high}\end{Bmatrix}$ end of the

market. He is $\begin{Bmatrix}\text{empowered}\\ \text{authorized}\end{Bmatrix}$ to receive on our behalf $\begin{Bmatrix}\text{samples}\\ \text{specimens}\end{Bmatrix}$ of the

$\begin{Bmatrix}\text{latest}\\ \text{newest}\end{Bmatrix}$ $\begin{Bmatrix}\text{watch and jewellery}\\ \text{jewellery and watch}\end{Bmatrix}$ products, $\begin{Bmatrix}\text{up}\\ \text{subject}\end{Bmatrix}$ to a $\begin{Bmatrix}\text{limit}\\ \text{maximum}\end{Bmatrix}$

of ten thousand dollars. He will $\begin{Bmatrix}\text{carry}\\ \text{hold}\end{Bmatrix}$ a signed copy of this $\begin{Bmatrix}\text{letter}\\ \text{document}\end{Bmatrix}$

Resistência de Hashes

- Para tamanho de hash m:

Técnica	Esforço
Reversão (h -> y |H(y) = h)	2^m
Colisão fraca (x -> y != x ^ H(y) = H(y))	2^m
Colisão forte (x, y | H(x) = H(y))	2^(m/2)