Criptografia de Chave Pública

- Aplicações
  - Privacidade, Autenticação: RSA, Curva Elíptica
  - Intercâmbio de chave secreta: Diffie-Hellman
  - Assinatura digital: DSS (DSA)
- Vantagens
  - Não compartilha segredo
  - Provê autenticação
  - Provê não-repúdio
  - Escalável
- Desvantagens
  - Lenta (computacionalmente intensiva)
  - Requer autoridade de certificação (chave pública confiável)

Diffie-Hellman

- É um método para troca segura de chaves
- Inventado em 1976
- O objetivo é permitir a troca de chaves entre duas entidades remotas através de um meio de comunicação não segura
- É baseado na operação de logaritmos discretos

Raiz Primitiva

- O logaritmo discreto é uma função unidirecional
- Logaritmo discreto é baseado na raiz primitiva
- Raízes primitivas de um número primo p são as potências por todos os inteiros de 1 a p-1
- Se a é uma raiz primitiva de p, então

  $a \bmod p, a^2 \bmod p, ..., a^{p-1} \bmod p$

  são distintos e consistem em inteiros de 1 a p-1

Logaritmos Discretos

- Para um inteiro ***b*** uma raiz primitiva ***a*** de um número primo ***p*** é possível encontrar um expoente ***i*** tal que:

  $b = a^i \bmod p \quad \text{onde } 0 \leq i \leq (p-1)$
- O expoente *i* é chamado de logaritmo discreto de *b* na base *a* mod *p*.
- Dado *a*, *i* e *p*, é fácil calcular *b*
- Dado *a*, *b* e *p*, é difícil calcular *i*

Algoritmo Diffie-Hellman

- O algoritmo gera a mesma senha para dois usuários distintos (Alice e Bruno), dado ***p*** primo e **α** uma raiz primitiva de *p:*

Alice		Bruno
sorteia Xa < p		
calcula Ya = α^{Xa} mod p	Ya →	
		sorteia Xb < p
		calcula Yb = α^{Xb} mod p
	Yb ←	calcula K = Ya^{Xb} mod p
calcula K = Yb^{Xa} mod p		

Prova de que ambos K são iguais

Temos: Ya = α^{Xa} mod *p* e Yb = α^{Xb} mod *p*

$K_A = Yb^{Xa} \bmod p$

$(\alpha^{Xb} \bmod p)^{Xa} \bmod p$

$(\alpha^{Xb})^{Xa} \bmod p$

$(\alpha^{Xa})^{Xb} \bmod p$

$(\alpha^{Xa} \bmod p)^{Xb} \bmod p$

$Ya^{Xb} \bmod p$

$= K_B = K$

Exemplo Diffie-Hellman

- p = 97, $\alpha = 5$
- Alice sorteia Xa = 36 e Bruno sorteia Xb = 58
- Alice calcula Ya = 5^{36} = 50 mod 97
- Bruno calcula Yb = 5^{58} = 44 mod 96
- Bruno calcula K = $(Ya)^{Xb}$ mod 97 = 50^{58} = 75 mod 97
- Alice calcula K = $(Yb)^{Xa}$ mod 97 = 44^{36} = 75 mod 97

Diffie-Hellman – Conclusão

- Diffie-Hellman é uma técnica muito usada para troca de chaves
  - SSL (Secure Socket Layer)
  - PGP (Pretty Good Privacy)
- É eficiente
- Porém está sujeito ao ataque do homem no meio na troca de valores públicos Y
- Segurança do Diffie-Hellman:
  - Criptoanálise: conhecidos *q*, α e Y, é preciso calcular o log discreto para obter X (difícil)

Algoritmo RSA

- O algoritmo RSA foi desenvolvido em 1977 pelo Ron Rivest, Adi Shamir e Len Adleman
- RSA é um algoritmo de chave pública
- É baseado em logaritmos discretos
- As senhas são geradas com base em dois números primos grandes (mais de 100 dígitos)
- A segurança é baseada na dificuldade de fatoração de números inteiros

Algoritmo RSA

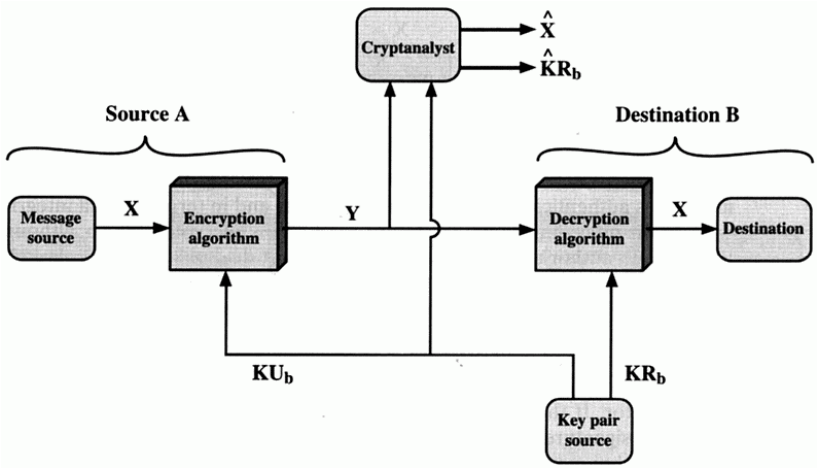
- Geração do par de chaves (pública/privada)
  - Selecionar ***p*** e ***q*** , ambos números primos
  - Calcular ***n*** = $p \times q$
  - Calcular $\phi(n) = (p - 1)(q - 1)$ [quociente de Euler]
  - Selecionar inteiro ***e*** , primo relativo a $\phi(n)$
  - Calcular ***d*** = $e^{-1} \bmod \phi(n)$ [ou $de = 1 \bmod \phi(n)$]
  - Chave Pública: **KU = {*e*, *n*}**
  - Chave Privada: **KR = {*d*, *n*}**
- Encriptação: $C = M^e \bmod n$, $M < n$
- Decriptação: $M = C^d \bmod n$

Algoritmo RSA

- Para criptografar a mensagem M:
- $C = M^e \bmod n$
- Para decriptografar a mensagem cifrada C:

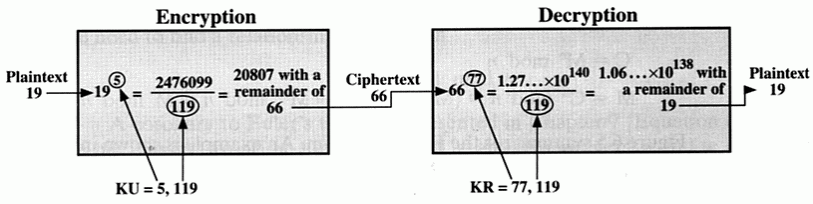
  $M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$
- Ambos os lados deve conhecer n
- A senha pública KU é formada por {e, n}
- A senha secreta KR é formada por {d, n}
- O algoritmo funciona porque $M^{ed} = M \bmod n$ quando $ed = 1 \bmod \phi(n)$

Criptografia RSA



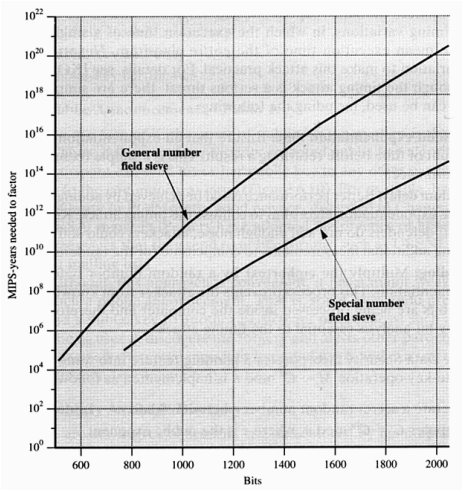
Exemplo RSA

- Primos $p = 7$ e $q = 19$. $n = pq = 119$, $\phi(n) = 6 \times 18 = 108$
- Senha pública do destinatário: $\{e, n\} = \{5, 119\}$
- Senha secreta do destinatário: $\{d, n\} = \{77, 119\}$
- Mensagem M: 19
- Mensagem cifrada C: 66



Segurança do RSA

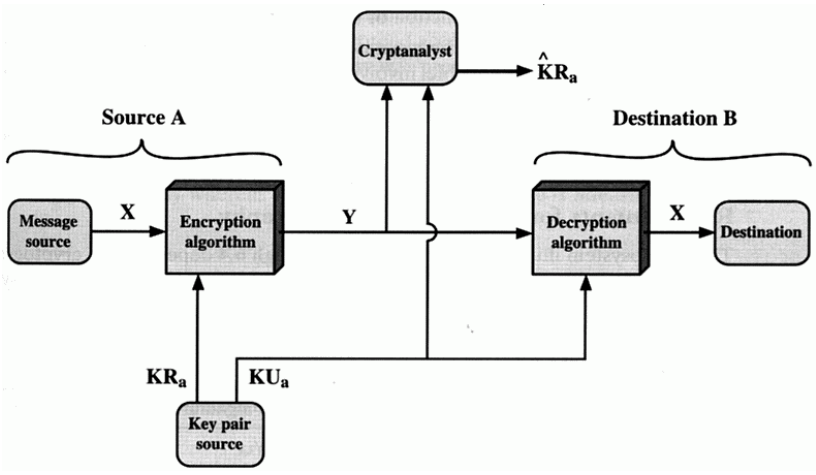
- Criptoanálise: conhecendo *e* e *n*, é preciso fazer a fatoração de *n*, para obter os dois primos *p* e *q* e calcular *d*
- Fatoração é uma tarefa demorada
- Pentium 200Mhz é uma máquina de 50 MIPS



Assinatura digital usando RSA

- O algoritmo RSA pode ser usado para assinar digitalmente um documento
- A assinatura garante a autenticidade
- A assinatura é gerada com base na senha secreta do assinante
- Desta forma um documento assinado só poderia ter sido gerado pelo dono da senha

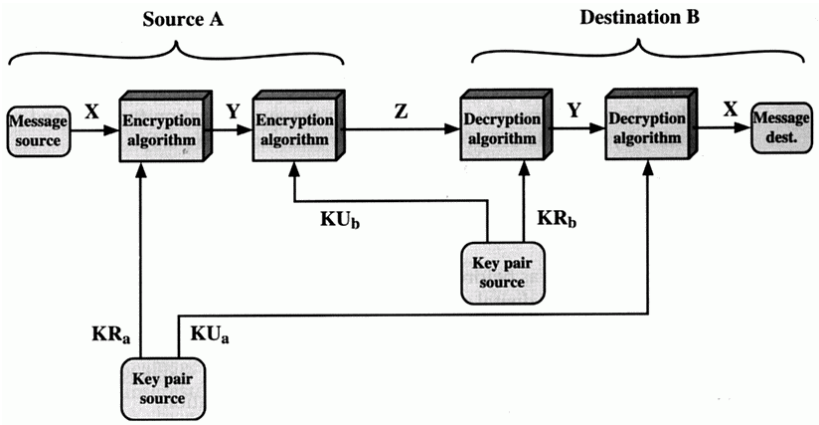
Assinatura digital usando RSA



Criptografia e Assinatura digital

- Com o RSA é possível ainda criptografar e assinar digitalmente
- Assim a autenticidade e a confidencialidade são garantidas simultaneamente
- Duas operações de criptografia são executadas em seqüência no documento original:
  - Uma com a senha secreta do assinante
  - Outra com a senha pública do destinatário

Criptografia e Assinatura digital



Distribuição de Chaves

- A distribuição de chaves é um possível ponto de falha também em um sistema de chave pública
- Um usuário C pode gerar uma par KR/KU em nome de B e enviar a chave pública para A
- A, ao gerar uma mensagem pensando que é para B, está gerando uma mensagem que na verdade somente C tem a chave para ler

Autoridade de Certificação

- Autoridades de certificação são usadas para distribuir chaves públicas garantindo a sua autenticidade
- A CA (Certificate Authority) é uma entidade confiável e reconhecida (VeriSign, Thawte, ValiCert, GlobalSign, Entrust, BelSign)
- A CA emite certificado digital que inclui a chave pública de uma entidade, com dados para identificação confiável desta e assinado digitalmente com a chave privada da CA

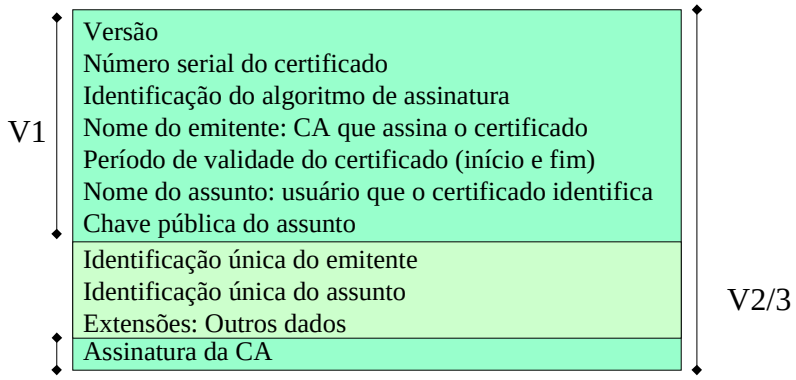
Autoridade de Certificação

- A chave pública da CA deve ser muito bem conhecida e amplamente disponível, pois é usada por quem recebe um certificado vindo da CA, para validá-lo (autenticar assinatura)
- Exemplo de conteúdo de Certificado Digital:

Nome do indivíduo e/ou organização
Chave pública do detentor
Data de validade do certificado
Numeração de controle do certificado
Identificação da CA
Assinatura digital da CA

Certificado X.509

- Padrão ITU-T para certificado: X.509
- Usado em S/MIME, IPSec, SSL/TLS, SET



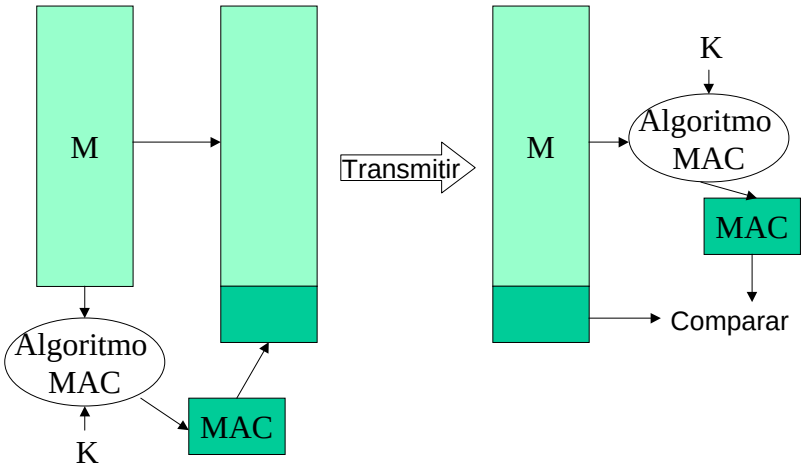
Autenticação de Mensagem

- Criptografia do conteúdo protege mensagens contra interceptação (ataque passivo)
- Criptografia e técnicas similares podem ser usadas para autenticação de mensagens: garantir e validar sua integridade contra falsificação (ataque ativo)
- Autenticação simples de mensagem na criptografia convencional:
  - Incluir na mensagem informações de código de correção de erro e controle de seqüência e tempo

Autenticação de Mensagem

- Técnicas de autenticação de mensagem sem criptografar toda a mensagem:
  - Message Authentication Code (MAC)
  - Função de Hash Unidirecional
- MAC
  - Uso de uma **chave secreta K_{AB}** para gerar um pequeno bloco de dados conhecido como código de autenticação da mensagem, anexado a esta
  - $MAC_M = F(K_{AB}, M)$
  - O receptor gera o mesmo código e compara

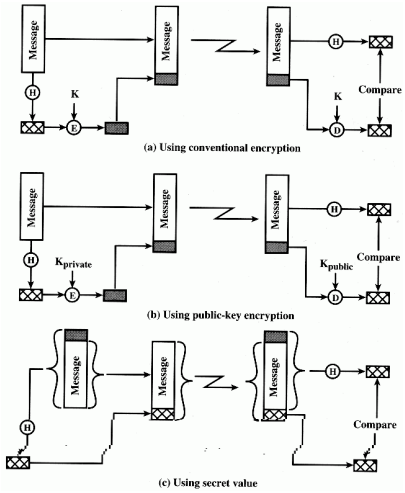
MAC



Algoritmos Hash Unidirecionais

- Algoritmo Hash Unidirecional:
  - Toma uma mensagem arbitrária M e gera uma compilação da mensagem (*message digest*) de tamanho fixo H(M) como saída (como nos MACs)
  - O algoritmo não precisa ser reversível
  - Diferente do MAC, a função hash não usa uma chave secreta como parâmetro
  - A autenticação da mensagem passa a ser baseada na autenticação segura do *digest*:
    - Adicionar de um valor secreto a M antes de gerar H(M)
    - Criptografia convecional do message digest
    - Assinatura do digest com a chave privada do emissor

Algoritmos Hash Unidirecionais



Algoritmos Hash Unidirecionais

- Exemplo de um hash simples de *n* bits:
  - Tomar a mensagem em blocos de *n* bits
  - Fazer XOR do bit *i* (de 1 a *n*) de todos os blocos
  - $C_i = b_{i1} \oplus b_{i2} \oplus ... \oplus b_{in}$

	Bit 1	Bit 2		Bit n
Bloco 1	b_{11}	b_{21}		b_{n1}
Bloco 2	b_{12}	b_{22}		b_{n2}
	...	...	...	...
Bloco m	b_{1m}	b_{2m}		b_{nm}
	C_1	C_2		C_n

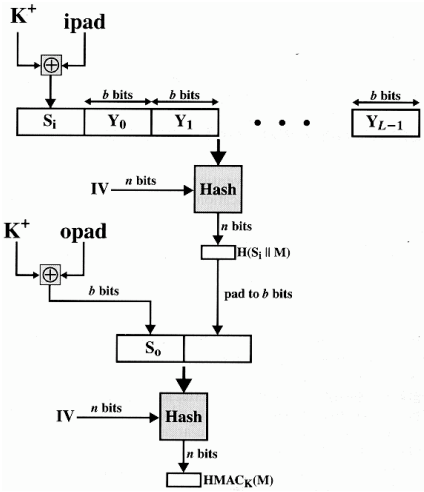
Algoritmos Hash Unidirecionais

- Algoritmos Hash Seguros
  - SHA-1, Secure Hash Algorithm-1: desenvolvido pelo NIST (National Institute of Standards and Technology), EUA
    - Digest de 160 bits
    - Propriedade principal: todo bit do código hash é função de todos os bits da mensagem de entrada
  - MD4 e MD5, Message Digest Algorithm #4, #5: desenvolvido pela RSA (128 bits)
  - RIPEMD, RACE Integrity Primitives Evaluation (RIPE) MD: projeto europeu RIPE. RIPEMD-160, RIPEMD-256 (hash de 160 e 256 bits)

MAC & Hash

- MAC a partir de uma função hash
  - Idéia: desenvolver um MAC derivado de um hash unidirecional, introduzindo uma chave secreta
- Padrão mais aceito é o HMAC. Princípios:
  - Usar uma função hash existente (ex: SHA-1), sem modificações (como se fosse uma "caixa preta")
  - Deixar a função hash facilmente substituível, caso se deseje usar outra função mais conveniente
  - Preservar o desempenho original da função hash, sem introduzir degradação significante
  - Tratar de forma simples a chave secreta usada

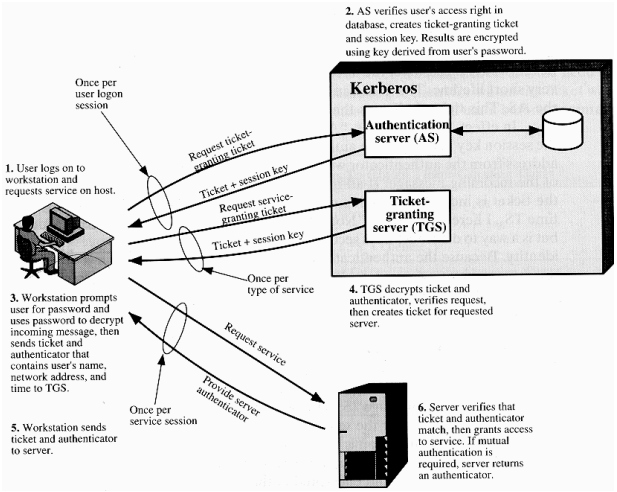
HMAC



Autenticação de Usuário

- Kerberos
  - Parte do Projeto Athena, do MIT
  - Problema: em um ambiente distribuído aberto, é necessário autenticar requisições e restringir acesso a usuários autorizados
  - Idéia do Kerberos: é difícil garantir a segurança de muitos servidores em uma rede, mas é viável garantir alta segurança de um único servidor
  - Kerberos: servidor de autenticação central que contém e valida a senha (chave) e autorizações de todos os usuários e servidores da rede
  - Baseado em criptografia convencional

Kerberos


2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.
Kerberos
Authentication server (AS)
Ticket-granting server (TGS)
Once per user logon session
Request ticket-granting ticket
Ticket + session key
Request service-granting ticket
Ticket + session key
Once per type of service
1. User logs on to workstation and requests service on host.
3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.
4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.
Request service
Provide server authenticator
Once per service session
5. Workstation sends ticket and authenticator to server.
6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.