Infraestrutura de Chave Pública - InfoSec

Conteúdo

- Estabelecimento de chaves simétricas
- Protocolos baseados em autoridades centrais (Kerberos)
- Ataque “Man in the middle”
- Esquemas de distribuição de chaves públicas
- Certificados

Estabelecimento de chaves

- Duas abordagens
  - Transporte de chaves (key transport)
    - Apenas um usuário decide qual é a chave
    - Tipo criptografia simétrica
  - Negociação de chaves (key agreement)
    - Os dois usuários participam da geração da chave
    - Tipo Diffie-Hellman

Distribuição de chaves simétricas

- Abordagem naïve:
  - Estabelecer pares de chaves secretas entre todos usuários
  - Para usuário (A, B, C, D) temos chaves $k_{AB}, k_{AC}, k_{AD}, k_{BC}, \ldots$
- Problemas:
  - Número de pares de chaves é $n \cdot (n-1)/2$
  - Para 1000 usuários, necessárias 499500 chaves
  - É complicado adicionar novos usuários (como?)

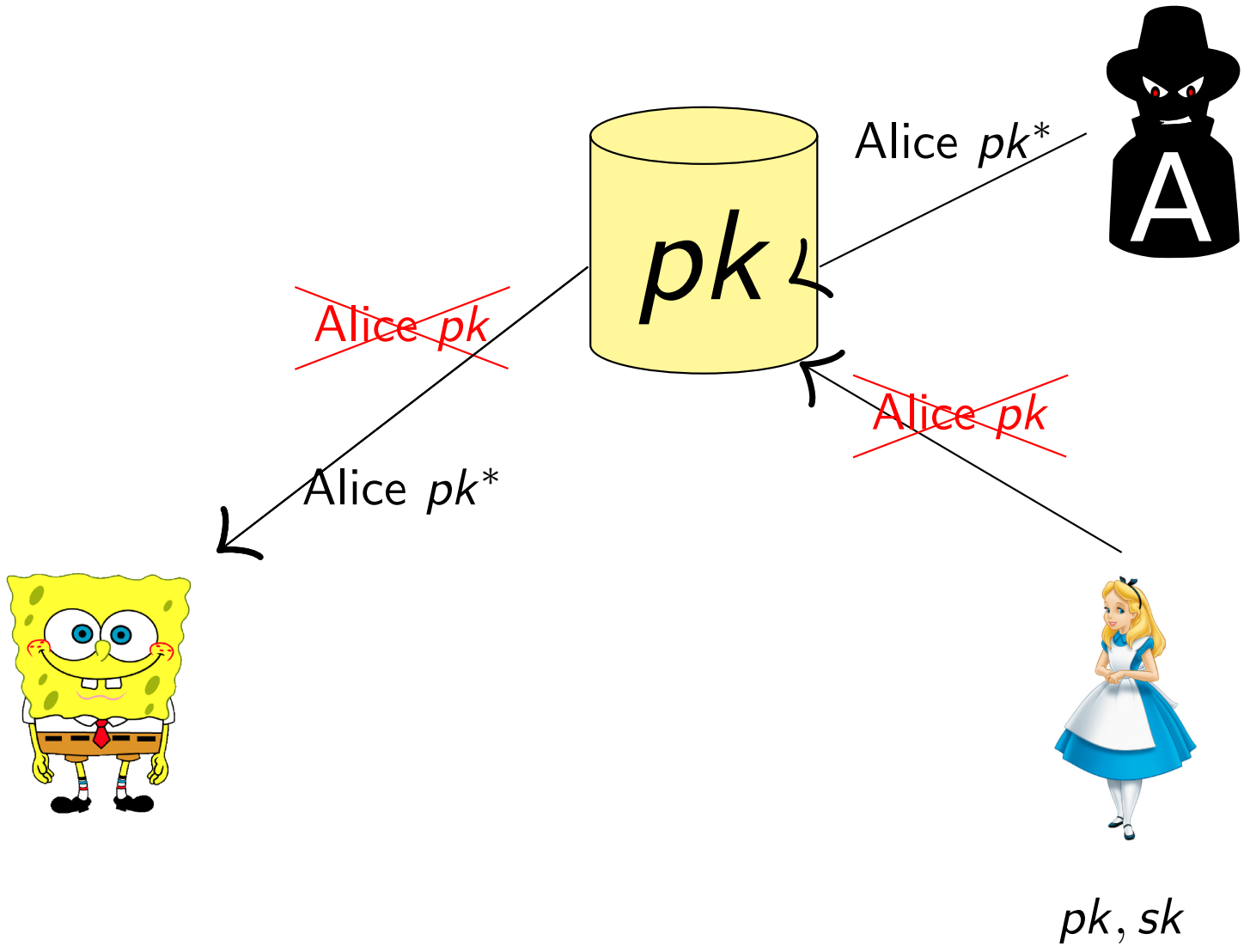
Protocolos baseados em centro de distribuição de chaves (Kerberos)

- Ideia: usar um "autoridade confiável" central
- Key Distribution Center (KDC)
  - Autoridade central compartilha uma chave com cada usuário
  - Usuários (A, B, C, D), chaves: $k_{A,KDC}, k_{B,KDC}, k_{C,KDC}, k_{D,KDC}$
- Comunicação entre A e B exige comunicação com KDC primeiro
  - KDC gera k_{sessao} aleatória
  - KDC encripta: $y_A = e_{k_A}(k_{sessao})$ e $y_B = e_{k_B}(k_{sessao})$
  - KDC envia y_A e y_B para A
  - A obtém chave $k_{sessao} = e_{k_A}^{-1}(y_A)$
  - A envia texto cifrado com k_{sessao} e y_B para B
- Para um usuário novo E, basta se cadastrar em KDC e obter uma key encryption key (KEK) $k_{E,KDC}$
- Problemas:
  - KDC é um ponto único de falha

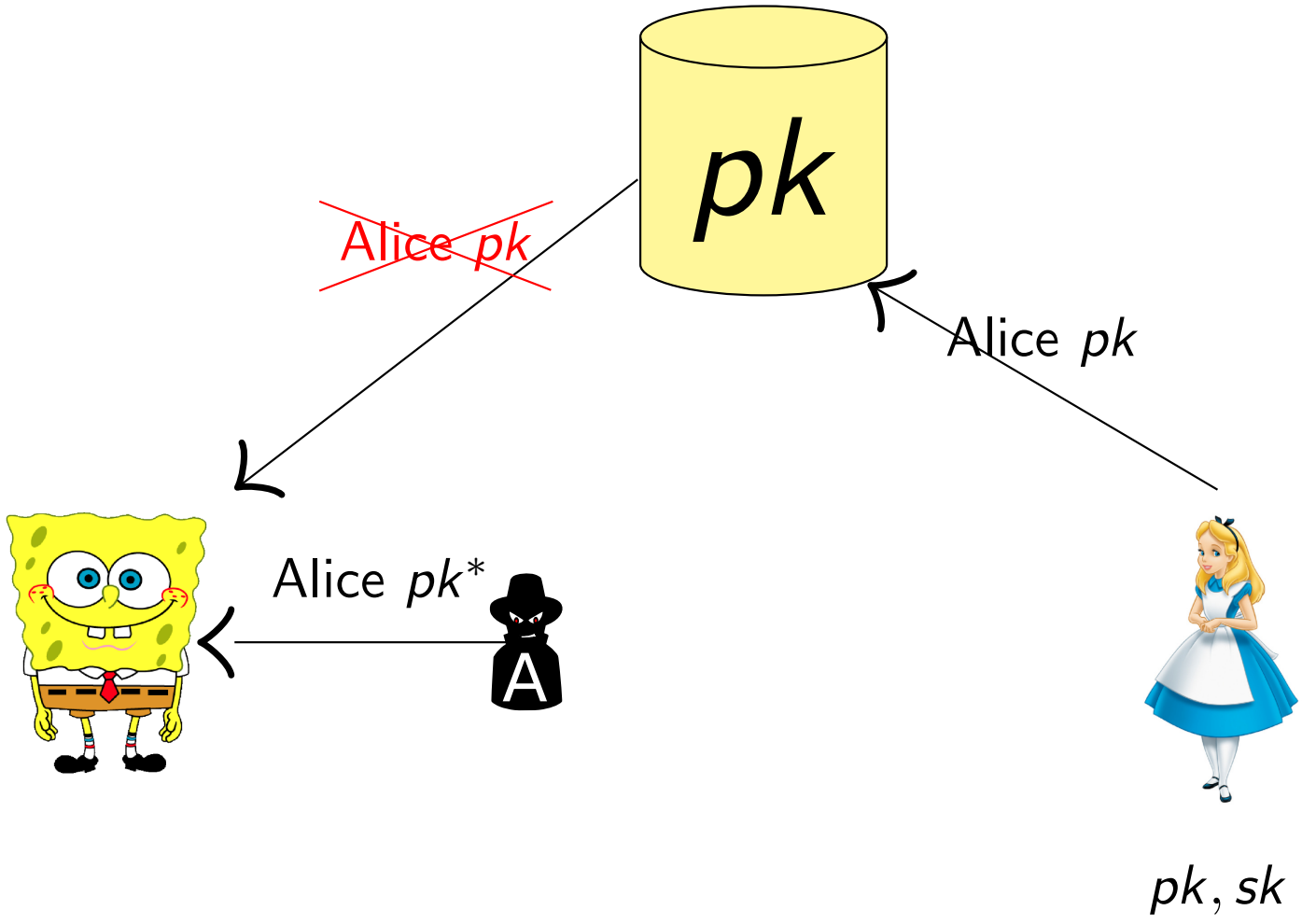
Estabelecimento de chaves assimétricas

- Meta: troca de chaves com meio inseguro
- Abordagens:
  - Transporte de chaves: autoridade central
  - Negociação de chaves: Diffie-Hellman
- Problema: Man-in-the-Middle attack

Man-in-the-Middle attack



Distribuição de chaves públicas



Usar assinaturas para distribuição segura de chaves

- Ideia: usar autoridade confiável com uma chave pública conhecida por todos
  - CA = certifying authority = autoridade certificadora
  - Chave pública pk_{CA}
  - Chave privada sk_{CA}

Usar assinaturas para distribuição segura de chaves

- Alice pede para o CA assinar o contrato $(Alice, pk)$
  - Certificado da CA sobre Alice consiste na assinatura do par "dados de identidade da Alice" ID_{Alice} e "chave pública da Alice" pk_{Alice} com a chave privada sk_{CA} da autoridade certificadora
  - $cert_{CA \to Alice} = Sign_{sk_{CA}}(ID_{Alice}, pk_{Alice})$
- CA deve verificar identidade de Alice fora da comunicação (por canal seguro)

Geração de certificado pelo cliente

Certificate Generation with User-Provided Keys

Alice		**CA**
generate $k_{pr,A}, k_{pub,A}$		
	$\xrightarrow{\text{RQST}(k_{pub,A}, ID_A)}$	
		verify ID_A
		$s_A = \text{sig}_{k_{pr,CA}}(k_{pub,A}, ID_A)$
		$\text{Cert}_A = [(k_{pub,A}, ID_A), s_A]$
	$\xleftarrow{\text{Cert}_A}$	

Geração de certificado pela CA

Certificate Generation with CA-Generated Keys

Alice		**CA**
request certificate	$\xrightarrow{\text{RQST}(ID_A)}$	
		verify ID_A generate $k_{pr,A}, k_{pub,A}$ $s_A = \text{sig}_{k_{pr},CA}(k_{pub,A}, ID_A)$ $\text{Cert}_A = [(k_{pub,A}, ID_A), s_A]$
	$\xleftarrow{\text{Cert}_A, k_{pr,A}}$	

Usar assinaturas para distribuição segura de chaves

- Bob obtém par "ID_{Alice}, pk" e o certificado $cert_{CA \to Alice}$
  - e verifica que $Vrfy_{PK_{CA}}((ID_{Alice}, pk_{Alice}), cert_{CA \to Alice}) = 1$
- Bob agora tem certeza que pk é a chave pública de Alice
  - Depende da confiabilidade de CA
    - CA deve ser honesto e verificar apropriadamente a identidade de Alice
  - Depende da segurança da chave privada de CA

Diffie-Hellman com certificados

Diffie–Hellman Key Exchange with Certificates

Alice		**Bob**
$a = k_{pr,A}$		$b = k_{pr,B}$
$A = k_{pub,A} \equiv \alpha^a \bmod p$		$B = k_{pub,B} \equiv \alpha^B \bmod p$
$\text{Cert}_A = [(A, ID_A), s_A]$		$\text{Cert}_B = [(B, ID_B), s_B]$
	$\text{Cert}_A \longrightarrow$	
	$\longleftarrow \text{Cert}_B$	
verify certificate:		verify certificate:
$\text{ver}_{k_{pub,CA}}(\text{Cert}_B)$		$\text{ver}_{k_{pub,CA}}(\text{Cert}_A)$
compute session key:		compute session key:
$k_{AB} \equiv B^a \bmod p$		$k_{AB} \equiv A^b \bmod p$

- s_A e s_B são as assinaturas de certificação do CA

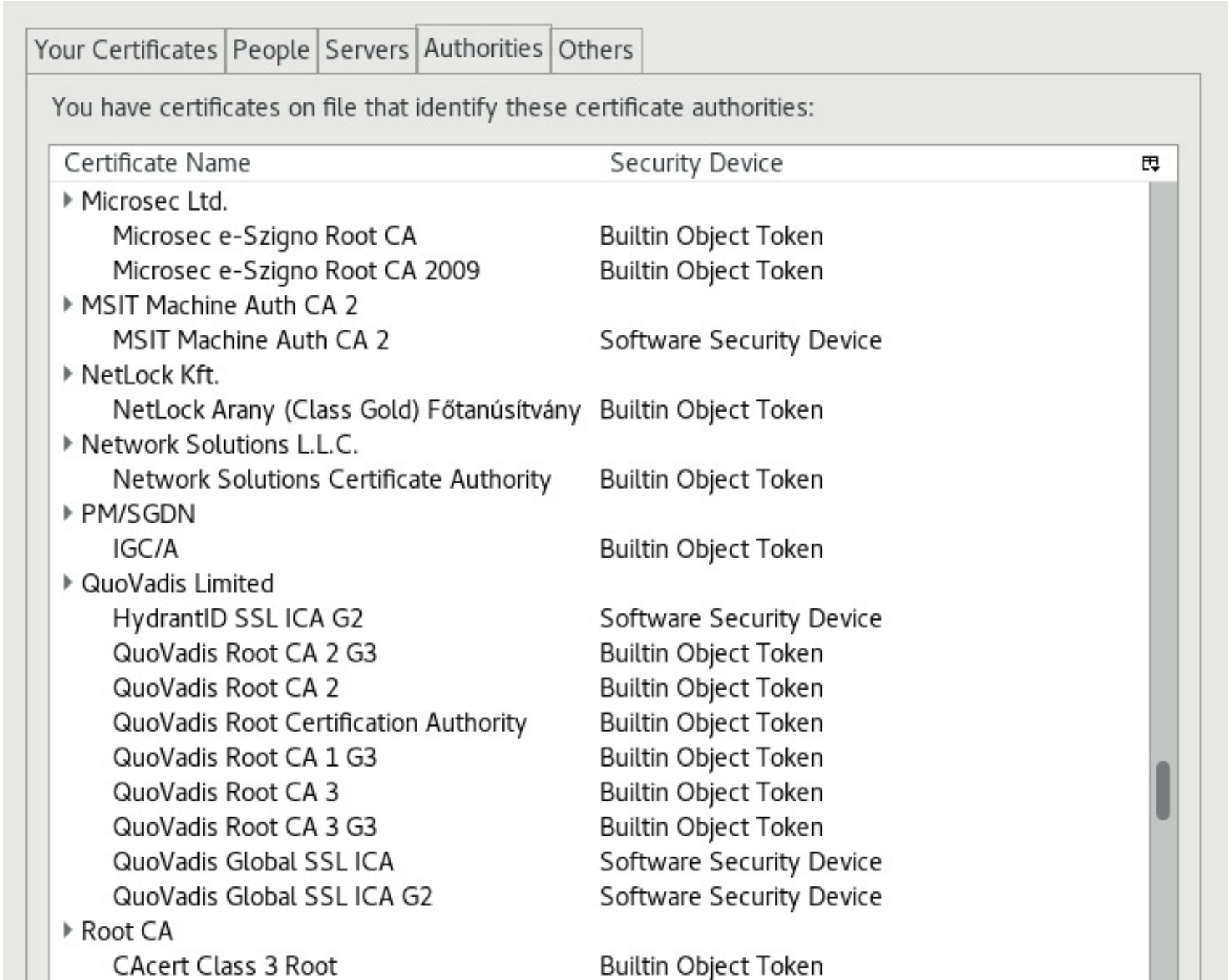
Problema cíclico?

- Com Bob obtém pk_{CA} em primeiro lugar?
- Opções:
  - Roots of Trust
  - Web of Trust
  - Repositórios de chaves públicas
    - MIT PGP keyserver

Roots of Trust: chaves de CAs

- Pontos de origem de confiança (Roots of Trust)
- Bob só precisa obter de maneira segura um número pequeno de chaves públicas de CAs
- Necessário garantir distribuição segura somente para essas poucas chaves públicas iniciais
- Opções
  - Distribuir como parte do sistema operacional
  - Instalar com os navegadores Web

Roots of Trust



Web of Trust: Teia de confiança

- PGP, GnuPG, OpenPGP...
- Obter chaves públicas de meus amigos pessoalmente
  - Partes que assinam chaves
- Obter cerficados na minha chave pública de meus amigos
- Se A conhecer pk_B e B emitiu certificado para C então C pode mandar esse certificado para A

Repositório de chaves públicas

- Armazenar certificados em um repositório central
  - Exemplo: servidor de chaves PGP do MIT
- Para achar a chave pública de Alice
  - Obter todas as chaves públicas para Alice e certificados nessas chaves
  - Procurar por um certificado assinado por alguém que você confia cuja chave pública você já tem

PKI na prática

- Não funciona tão bem quanto na teoria
  - Proliferação de CAs raiz
  - Revocação
  - Outros problemas