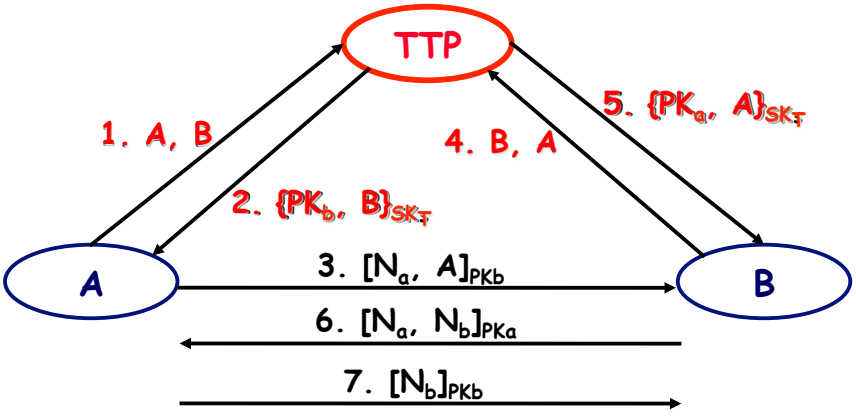
Lecture 13

Public Key Distribution (certification)

PK-based Needham-Schroeder



Here, TTP acts as an "on-line" certification authority (CA) and takes care of revocation

What if?

Alice and Bob have:

- No common mutually trusted TTP(s)

and/or

- No on-line TTP(s)

Public Key Infrastructure (Distribution)

- Problem: How to determine the correct public key of a given entity
  - Binding between IDENTITY and PUBLIC KEY
- Possible attacks
  - Name spoofing: Eve associates Alice's name with Eve's public key
  - Key spoofing: Eve associates Alice's key with Eve's name
  - DoS: Eve associates Alice's name with a nonsensical (bogus) key
- What happens in each case?

Public Key Distribution

- Diffie - Hellman (1976) proposed the "public file" concept
  - universally accessible
  - no unauthorized modification
  - not scalable!

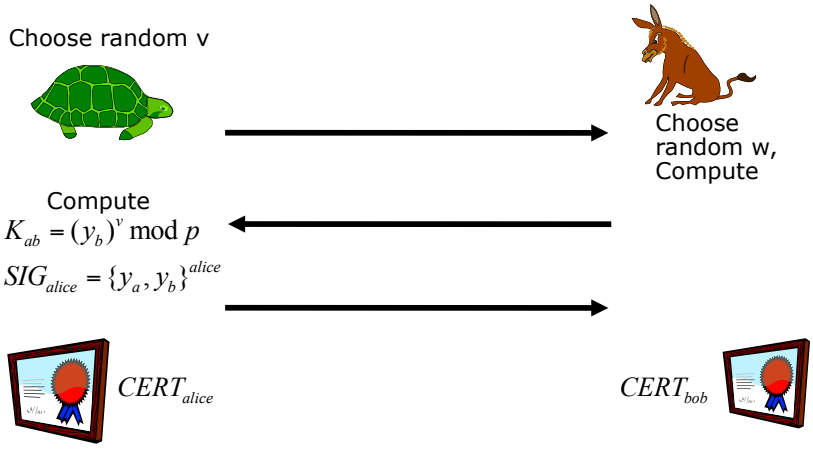
Public Key Distribution

- Popek - Kline (1979) proposed "trusted third parties" (TTPs)
  - TTPs know public keys of the entities and distribute them on-demand basis
  - on-line protocol (a disadvantage)

Certificates

- Kohnfelder (BS Thesis, MIT, 1978) proposed "certificates" as yet another public-key distribution method
- Explicit binding between the public-key and its owner/name
- Issued (digitally signed) by the Certificate Authority (CA)
- Issuance is done off-line

Authenticated Public-Key-based Key Exchange (Station-to-Station or STS Protocol)



Certificates

- Procedure
  - Bob registers at local CA
  - Bob receives his certificate:

    $\{ PK_B, ID_B, \text{issuance\_time}, \text{expiration\_time}, \text{etc.},\ldots\}SK_{CA}$
  - Bob sends certificate to Alice
  - Alice verifies CA's signature
    - PK_{CA} hard-coded in software
  - Alice uses PK_B for encryption and/or verifying signatures

Who issues certificates?

CA: **Certification Authority**

e.g. GlobalSign, VeriSign, Thawte, etc.

look into your browser...

- Trustworthy (at least to its users/clients)
- Off-line operation (usually)
- Has a well-known long-term certificate
- May store client certificates
- Very secure: physically and electronically

How does it work?

- A public/private key-pair is generated by user
- User requests certificate via local application (e.g., web browser)
  - Good idea to prove knowledge of private key as part of the certificate request. Why?
- Public key and "name" usually part of a PK certificate
- Private keys only used for small amount of data (signing, encryption of session keys)
- Symmetric keys (e.g., RC5, AES) used for bulk data encryption

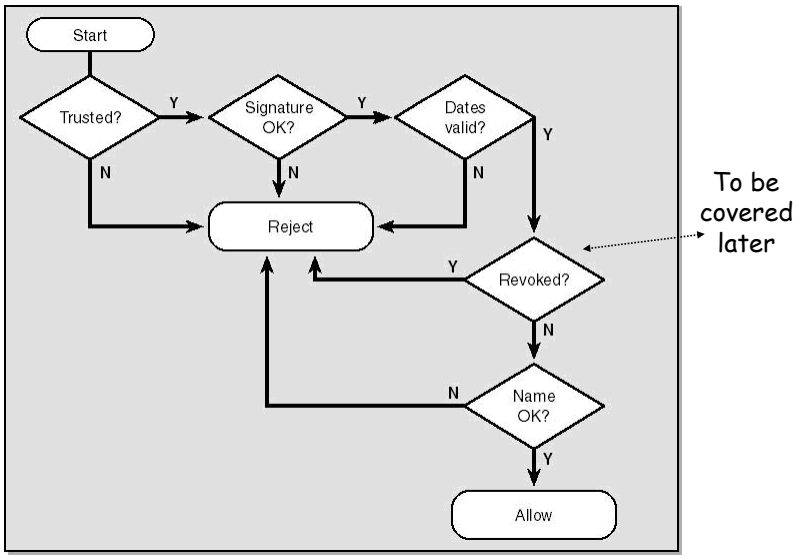
CA

- CA checks that requesting user is who he claims to be (in the certificate request)
- CA's own certificate is signed by a higher-level CA. Root CA's certificate is self-signed and his identity/name is "well-known"
- CA is a critical part of the system and must operate in a secure and predictable way according to some policy

Who needs them?

- Alice's certificate is checked by whomever wants to: 1) verify her signatures, and/or 2) encrypt data for her.
- A verifier must:
  - know the public key of the CA(s)
  - trust all CAs involved
- Certificate checking is: verification of the signature and validity
- Validity: expiration + revocation checking

Verifying a certificate (assuming common CA)



BTW:

- Certificate types
  - PK (Identity) certificates
    - Bind PK to some identity string
  - Attribute certificates
    - Bind PK to arbitrary attribute information, e.g., authorization, group membership
- We concentrate on former

What are PK certificates good for?

- Secure channels in TLS / SSL for web servers
- Signed and/or encrypted email (PGP,S/MIME)
- Authentication (e.g., SSH with RSA)
- Code signing!
- Encrypting files (EFS in Windows/2000)
- IPSec: encryption/authentication at the network layer

Components of a certification system

- Request and issue certificates (different categories) with verification of identity
- Storage of certificates
- Publishing/distribution of certificates (LDAP, HTTP)
- Pre-installation of root certificates in a trusted environment
- Support by OS platforms, applications and services
- Maintenance of database of issued certificates (no private keys!)
- Helpdesk (information, lost + compromised private keys)
- Advertising revoked certificates (and support for applications to perform revocation checking)
- Storage "guidelines" for private keys

CA Security

- Must minimize risk of CA private key being compromised
- Best to have an off-line CA
  - Requests may come in electronically but not processed in real time
- Microsoft recommends using CA hierarchy where root CA is off-line and signing CA are on-line
- In addition, using tamper-resistant hardware for the CA would help (should be impossible to extract private key)

Mapping personal certificates into accounts/names

- Certificate must map "one-to-one" into an account/name for the sake of authentication
- In some systems, mapping are based upon X.509 naming attributes from the Subject field
- Example: Verisign issues certificate as CN=Full Name (account)
- Account/name is local to the issuing domain

Storage of private key

- The problem of having the user to manage the private key (user support, key loss or compromise)
- Modern OS's offers Protected Storage which saves private keys (encrypted).
- Applications take advantage of this; Browsers sometimes save private keys encrypted in its configuration directory
- users who mix applications or platforms must manually import / export private keys via PFX files.

Key lengths

- Strong encryption has been adopted since the relaxation of US export laws
- 512-bit RSA and 56-bit DES are not safe
- Root CA should have an (RSA) key length of >= 2048 bits given its importance and typical lifetime of 3-5 years
- A personal (RSA) certificate should have key length of >= 1536 bits

Naming comes first!

- Cannot have certificates without a comprehensive naming scheme
- Cannot have PKI without a comprehensive distribution/ access method
- X.509 uses X.500 naming
- X.500 Distinguished Names (DNs) contain a subset of:
  - **C** **Country**
  - **SP** **State/Province**
  - **L** **Locality**
  - **O** **Organization**
  - **OU** **Organizational Unit**
  - **CN** **Common Name**

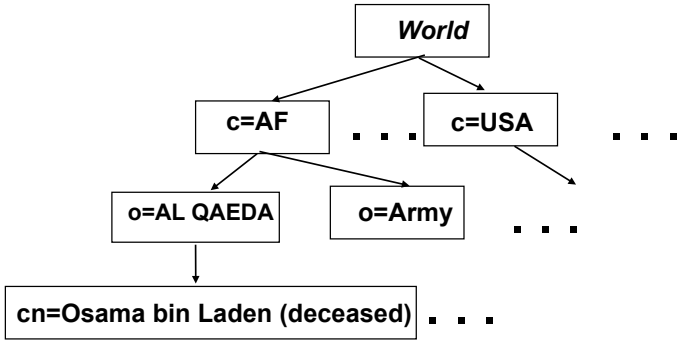
X.500

- ISO standard for directory services
- global, distributed
- first solid version in 1988. (second in 1993.)
- documentation - several RFC's

X.500

- data model:
  - based on hierarchical namespace
  - Directory Information Tree (DIT)
  - geographically organized
  - entry is defined with its dn (Distinguished Name)
- searching:
  - you must select a location in DIT to base your search
  - a one-level search or a subtree search
  - subtree search can be slow

X.500 - DIT



dn: cn=Osama bin Laden, o=Al Qaeda, c=AF

X.500

- accessible through:
  - telnet (client programs known as dua, dish, ...)
  - WWW interface
    For example: http://www.dante.net:8888/
- hard to use and very heavy ...

  ... thus LDAP was developed

LDAP

- LDAP - **Lightweight Directory Access Protocol**
- LDAP v2 - RFC 1777, RFC 1778
- LDAP v3 - RFC 1779
- developed to make X.500 easier to use
- provides basic X.500 functions
- referral model instead original chaining
  - server informs client to ask another server (without asking question on the behalf of client)
- LDAP URL format:
  - ldap://server_address/dn

(ldap://ldap.uci.edu/cn=Kasper Rasmussen,o=UCI,c=US)

Some relevant standards

- The IETF reference site
  - http://ietf.org/html.charters/wg-dir.html#Security_Area
- Public-Key Infrastructure (X.509, PKIX)
  - RFC 2459 (X.509 v3 + v2 CRL)
- LDAP v2 for certificate and CRL storage
  - RFC 2587
- Guidelines & practices
  - RFC 2527
- S/MIME v3
  - RFC 2632 & 2633
- TLS 1.0 / SSL v3
  - RFC 2246