

# Cryptography CS 555



## Lecture 18: Distributing Public Keys, X509, PGP

Department of Computer Sciences  
Purdue University

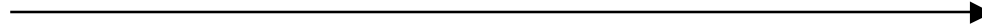
Cristina Nita-Rotaru

Spring 2005/Lecture 18

# Public Keys and Trust



Public Key:  $P_A$   
Secret key:  $S_A$



Public Key:  $P_B$   
Secret key:  $S_B$

- How are public keys stored
- How to obtain the public key?
- How does Bob know or 'trusts' that  $P_A$  is Alice's public key?

# Distribution of Public Keys

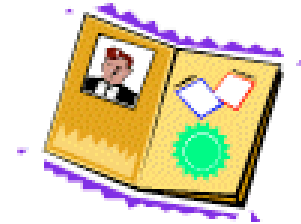
- **Public announcement:** users distribute public keys to recipients or broadcast to community at large
- **Publicly available directory:** can obtain greater security by registering keys with a public directory
- Both approaches have problems, and are vulnerable to forgeries



# X.509 Authentication Service

- Part of X.500 directory service standards.
- Defines framework for authentication services:
  - Defines that public keys stored as **certificates** in a public directory.
  - Certificates are **issued and signed** by an entity called **certification authority (CA)**.
- Used by numerous applications and protocols: SSL, IPSec.
- Started 1988

# Public-Key Certificates



- Certificates allow key exchange without real-time access to public-key authority
- A certificate binds identity to public key
- Contents signed by a trusted Public-Key or Certificate Authority (CA)
- Can be verified by anyone who knows the public-key authorities public-key
- A commonly used standard to store certificates is PEM.

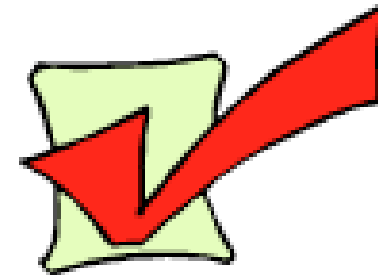
# X.509 Certificates

- Certificates contain:
  - version (1, 2, or 3)
  - serial number (unique within CA) identifying certificate
  - signature algorithm identifier
  - issuer X.500 name (CA)
  - period of validity (from - to dates)
  - subject X.500 name (name of owner)
  - subject public-key info (algorithm, parameters, key)
  - issuer unique identifier (v2+)
  - subject unique identifier (v2+)
  - extension fields (v3)
  - signature (of hash of all fields in certificate)

# How to Obtain a Certificate?

- For a particular application you can define your own CA (libraries like openssl provide the necessary tools)
- Many companies define their own CA.
- Verisign: company that provides certificates; commercial companies obtain certificates;
- Private key remains secret and certificate must be accessible.
- Example: see certificates accepted by your browser, if you use netscape: preferences/security and privacy/certificates

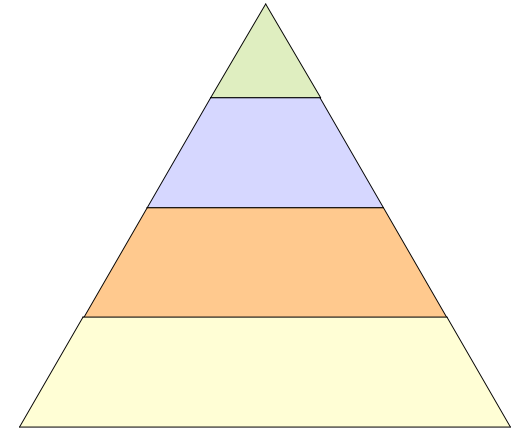
# Validity of Certificates



- Certificates are valid if:
  - Signature of CA verifies
  - Dates of the certificate are valid
  - Certificate was not revoked
- Certificates can be revoked before expiration if
  - user's private key is compromised
  - user is no longer certified by this CA
  - CA's certificate is compromised
- CA maintains a list of revoked certificates: **Certificate Revocation List (CRL)**
- Users should check certificates with CA's CRL

# CA Hierarchy

- If everybody has the same CA then they are assumed to know its public key, so they can verify each other's certificate. Not scalable.
- Other approach: entities have different CAs; in this case CAs how is a certificate verified?
  - CAs must form a hierarchy
  - certificates linking members of hierarchy are used to validate other CAs
  - each CA has certificates for clients (forward) and parent (backward)
  - each client trusts parents certificates



# CAs and Trust

- Certificates are trusted if signature of CA verifies
- Chain of CA's can be formed, head CA is called root CA
- In order to verify the signature, in the end the public key of the root CA should be obtain. When is that valid?
- “You just trust the root CA”.
- TRUST is CENTRALIZED (one CA) or HIERARCHICAL (more CAs.)

# Problems with X509

- Management of certificates
- Assumptions about validity of certificates:
  - detection of secret key disclosure
  - time delay for certificate revocation
  - time delay for distribution of revoked certificates
  - amount of data distributed periodically by CA

# Problems with X509 (2)

- CRLs have several problems
  - Protocols must check CRLs to make sure that the certificate is still valid
  - In practice protocols do not really check CRLs, delay between revocation and detection of revocation
  - CRL is not suitable for time-critical applications
  - time-validity of CRL is typically 24 hours
  - Validity of certificates is usually years

# Detection of Secret Key Disclosure

- Time between disclosure and detection may be in hours or days, time needed for abuse may be counted in milliseconds
- Owner is responsible for private key usage until requesting CA to revoke appropriate certificate
- There is no trusted way to identify place or time of signature creation

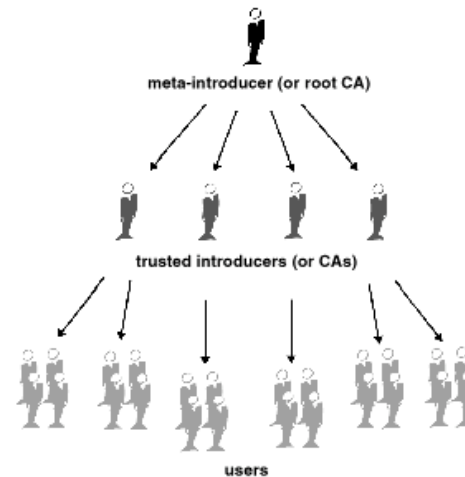
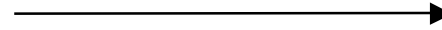
# PGP

- PGP (Pretty Good Privacy) is a secure email application
- Mail is encrypted and signed using public keys
- What's different? The way the keys are authenticated, trust about the keys is built.
- Trust is not centralized.
- <http://www.pgpi.org/>



# Trust Models

- Direct Trust
- Hierarchical trust
- Web of trust: combination of both



# PGP Web of Trust

- Any user can act as a CA
- Certificate is only valid if the receiving party recognize the validator as a trusted introducer
- Each user stores:
  - Its own public/private keys
  - Keys of entities that interacts with
  - whether or not the user considers a particular key to be valid
  - the level of trust the user places on the key that the key's owner can serve as certifier of others' keys

# Problems

- Key revocation of a key, a user needs to issue a revoked certificate and then distribute it as broad as possible.
- Does not scale for large, open communities
- Does not really accomodate for more formalised security needs, for instance for non-repudiation purposes towards a third party

# Next Lecture...

- Authentication protocols
- Network Authentication services: Kerberos

