Digital Signatures, Public Key Certificates, X509

Ahmet Burak Can
Hacettepe University
abc@hacettepe.edu.tr

Digital Signatures: The Problem

- Real-life examples for signatures:
  - A person pays by credit card and signs a bill; the seller verifies that the signature on the bill is the same with the signature on the card
  - Contracts: are valid if they are signed.
- Can we have a similar service in the electronic world?

Digital Signatures

- **Digital Signature**: a data string which associates a message with some originating entity.
- **Digital Signature Scheme**:
  - a **signing algorithm**: takes a message and a (private) signing key, outputs a signature
  - a **verification algorithm**: takes a (public) key verification key, a message, and a signature
- **Provides**:
  - Authentication
  - Data integrity
  - Non-Repudiation

Digital Signatures and Hash

- Digital signatures are generally used with hash functions, hash of a message is signed, instead of the message.
  - Since public key encryption is costly, signing hash digest is more efficient than signing the whole message.
  - So, a digital signature generally uses
    - A hash function: MD5, SHA-1, RIPEMD
    - A public key encryption algorithm: RSA, El-gamal
RSA Signatures

Key generation (as in RSA encryption):
- Select 2 large prime numbers of about the same size, p and q
- Compute \( n = pq \)
  - Since p and q are prime, \( \varphi(n) = (q - 1)(p - 1) \)
- Select a random integer \( e, \ 1 < e < \varphi(n) \), such that \( \gcd(e, \varphi(n)) = 1 \)
- Compute \( d, 1 < d < \varphi(n) \), such that \( ed \equiv 1 \pmod{\varphi(n)} \)

Public key: \((e, n)\)
Secret key: \(d\)

RSA Signatures (cont.)

Signing message \( M \)
- Verify \( 0 < M < n \)
- Compute \( S = M^d \mod n \)

Verifying signature \( S \)
- Use public key \((e, n)\)
- Compute \( S^e \mod n = (M^d \mod n)^e \mod n = M \)

Note: In practice, a hash of the message is signed and not the message itself.

Public Keys and Trust

ALICE
Public Key: \( P_A \)
Secret key: \( S_A \)

BOB
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
  - For example, put the public key to your web site
  - How to ensure the announcement is not forged by an adversary?

- **Publicly available directory**: can obtain greater security by registering keys with a public directory
  - How to implement the directory?
Public-Key Certificates

- A public key certificate binds identity to public key.
- Certificates are issued and signed by an entity called public key or certification authority (CA).
- Certificates can be verified by anyone who knows CA’s public-key.
- CA’s private key remains secret
- CA’s certificate must be accessible.
- Certificates allow key exchange without real-time access to public-key authority.

Public Key Infrastructure

- A system to securely distribute & manage public keys.
- Important for wide-area trust management (e.g., for e-commerce)
- Ideally consists of
  - a certification authority
  - certificate repositories
  - a certificate revocation mechanism (CRLs, etc.)
- Many models possible:
  - monopoly
  - delegated
  - oligarchy
  - anarchy

Monopoly Model

- Single organization is the certificate authority (CA) for everyone
- Shortcomings:
  - no such universally-trusted organization
  - requires everyone to authenticate physically with the same CA
  - compromise recovery is difficult (due to single embedded public key)
  - once established, CA can abuse its position (excessive pricing, etc.)
  - requires perfect security at CA
- CA may trust registration authorities (RAs) to check identities in order to do the initial authentication
  - Solves the problem of physically meeting the CA.

Delegated CAs

- Root CA certifies lower-level CAs to certify others
- All verifiers trust the root CA & verify certificate chains beginning at the root (i.e., the root CA is the trust anchor of all verifiers)
- Example: A national PKI, where a root CA certifies institutions, ISPs, universities who in turn certify their members
- Limitations are similar to monopoly with RAs
Oligarchy

- Many root CAs exist trusted by verifiers
- **The model of web security**
- Solves the problems of single authority (e.g., excessive pricing)
- Disadvantages:
  - n security-sensitive sites instead of one. Compromise of any one compromises the whole system
  - Users can easily be tricked into trusting fake CAs. (depending on implementation)

Anarchy

- Each user decides whom to trust & how to authenticate their public keys
- Certificates issued by arbitrary parties can be stored in public databases, which can be searched to find a path of trust to a desired party
- Works well for informal, non-sensitive applications
  - For example, in PGP, each person creates its public key certificate and distributes it to his/her friends

Revocation

- Mechanisms to cancel certificates compromised before expiration
- **Certificate Revocation List (CRL)**: list of revoked certificates, published periodically (mostly daily) by the CA
- **Delta CRLs**: Only the changes since the last issue are published
- **Online Revocation Servers**: No CRL is published. Verifier queries a central server to check if a certificate has been revoked.

X.509 Authentication Service

- Part of X.500 directory service standards.
  - Started 1988

- Defines framework for authentication services:
  - Defines that public keys stored as certificates in a public directory.
  - Certificates are issued and signed by certification authority.

- Used by numerous applications and protocols: SSL, IPSec.
Contents of X.509 Certificates

- 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)

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

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: A company (CA) that provides certificates
  ◦ Commercial companies obtain certificates from CAs.
- Example:
  ◦ See certificates accepted by your browser, if you use netscape: preferences/security and privacy/certificates

CA Hierarchy

- X509 entities have different CAs; in this case CAs how is a certificate verified?
  ◦ Start with the subject
  ◦ CAs must form a hierarchy
  ◦ Certificate's linking members of hierarchy are used to validate other CAs
  ◦ Each CA has certificates for clients (forward) and parent (backward)
  ◦ Each client trusts parent's certificates
Problems with X509

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

Problems with X509 – 2

- CRL 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