PKI
Digital Signatures

- Binds a message with the sender’s identity

  **Signing algorithm**: takes a message and a (private) signing key, outputs a signature

  **Verification algorithm**: takes a (public) verification key, a message, and a signature, generates True/False outcome

- Provides:
  - Authentication
  - Data integrity
  - Non-Repudiation (MAC does not provide this)
Digital Signatures and Hashes

- Digital signatures are used with hash functions, hash of a message is signed, instead of the message

- Hash function must be:
  - Pre-image resistant
  - Weak collision resistant
  - Strong collision resistant
Digital Signatures with RSA

**Signing message M**
- Compute $S = M^d \mod n$
- Equivalent to RSA decryption operation
- Requires private key (hence slower)
- RSA is slow, so instead of signing message, compute hash of message (e.g., SHA) and sign hash!

**Verifying signature S**
- Uses public key $(e, n)$
- Compute $S^e \mod n = (M^d \mod n)^e \mod n = M$
- Similar to encryption operation (faster than signing)
Digital Signatures with RSA

Alice (signer)
- M: Message
- S: Signature
- D: Digest
- h(M)
- $D^d \mod n$
- M

Alice’s private key
- $d$

Bob (verifier)
- M
- S
- $S^e \mod n$
- D
- $h(M)$
- true
- Accept

Verifying

Signing
Digital Signatures with Asymm. Encr

**Signing message M**
- Compute hash $h_1 = h(M)$
- Apply decryption function $\text{Signature} = D(h_1)$
- Send $M, \text{Signature}(M)$ to verifier

**Verifying signature S**
- Compute hash $h_2 = h(M)$
- Apply encryption function $h_1 = E(\text{Signature}(M))$
- Verify if $h_1 == h_2$

**DSA** is another popular signing algorithm
- Uses discrete logarithm hardness assumption
Public-Key Infrastructure (PKI)

- Goal: establish **trust** relationship based on **certification** of identity

- A **Certificate** binds identity to public key

- Contents signed by a trusted **Certificate Authority (CA)**
  - Can be verified by anyone using public key of CA

- There is a **chain of trust** established that leads to a CA

- PKI enable authentication, key exchange and digital signatures
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)

- Commonly used standard to represent certificates is **PEM**
Certification Authority

- Well-known entity, trusted by other parties
  - Must implement well-defined, standard processes for **identity vetting**
  - Provides online access to all certificates issued

- Provides **Certificate Revocation Lists (CRL)**: certificates that are no longer valid (e.g., private key was compromised)
  - These must be kept up-to-date!

- Private key of CA requires highest levels of security
  - 2048-bit or higher
  - Certificate of CA is self-signed
  - Private key stored **Hardware Security Module (HSM)**
Obtaining a certificate (1/2)

- Generate a private/public key
  - Several tools exist, e.g., openssl
- Must keep private key secure!
  - Typically encrypt with symm. encryption, e.g., AES/DES
- Create a certificate signing request (CSR) which includes public key as well as identifying information
- CSR sent to CA (e.g., VeriSign)
  - CA must ensure that the requester is indeed the entity in the CSR
  - This is called identity vetting
  - May require physical presence, presenting paper documents
- If identity is verified, CA signs the CSR and the resulting certificate is sent back to requester
Obtaining a certificate (2/2)

Certificate Request

- User Name & other credentials
- User’s Public key

Signed by using CA’s private key

CA

User Certificate

- Serial No.
- User Name
- User’s Email Address
- User’s Public Key
- CA’s Name
- Certificate Class
- Validity
- Digital Signature of CA

Certificate Database

- User 1 certificate
- User 2 certificate
- CRL xyz

Key pair Generation

Public

Private
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
Trust Chain

- Typically a hierarchy of trust is maintained
  - Too much overhead for CA to vet identity of everyone

- CA deals with higher-level organizations
  - E.g., enterprise, university

- Example:
  - Verisign (CA) signs certificate for state of MA
  - State of MA authority signs certificate for UMass System
  - UMass System signs certificate for UMB
  - UMB signs certificates for its employees
  - Employees are vetted by UMB only, chain of trust formed
  - Verification must check all signatures in the chain
Trust Chain Example
**Trust Chain Example**

**To create Diana’s certificate:**
Diana creates and delivers to Edward:

| Name: Diana  | Position: Division Manager | Public key: 17EF83CA ... |

Edward adds:

| Name: Diana  | Position: Division Manager | Public key: 17EF83CA ... | hash value 128C4 |

Edward signs with his private key:

| Name: Diana  | Position: Division Manager | Public key: 17EF83CA ... | hash value 128C4 |

Which is Diana’s certificate.

**To create Delwyn’s certificate:**
Delwyn creates and delivers to Diana:

| Name: Delwyn | Position: Dept Manager | Public key: 3AB3882C ... |

Diana adds:

| Name: Delwyn | Position: Dept Manager | Public key: 3AB3882C ... | hash value 48CFA |

Diana signs with her private key:

| Name: Delwyn | Position: Dept Manager | Public key: 3AB3882C ... | hash value 48CFA |

And appends her certificate:

| Name: Delwyn | Position: Dept Manager | Public key: 3AB3882C ... | hash value 48CFA |
| | Name: Diana  | Position: Division Manager | Public key: 17EF83CA ... | hash value 128C4 |

Which is Delwyn’s certificate.
## Trust Chain Example

**Key to encryptions**

- Encrypted under Betty’s private key
- Encrypted under Camilla’s private key
- Encrypted under Mukesh’s private key
- Encrypted under Delwyn’s private key
- Encrypted under Diana’s private key
- Encrypted under Edward’s private key

<table>
<thead>
<tr>
<th>Name</th>
<th>Position</th>
<th>Public Key</th>
<th>Hash Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew</td>
<td>Worker</td>
<td>7013F82A ...</td>
<td>60206</td>
</tr>
<tr>
<td>Betty</td>
<td>Task Leader</td>
<td>2468ACE0 ...</td>
<td>00002</td>
</tr>
<tr>
<td>Camilla</td>
<td>Group Leader</td>
<td>44082CCA ...</td>
<td>12346</td>
</tr>
<tr>
<td>Mukesh</td>
<td>Project Manager</td>
<td>47F0F008 ...</td>
<td>16802</td>
</tr>
<tr>
<td>Delwyn</td>
<td>Dept Manager</td>
<td>3AB3882C ...</td>
<td>48CFA</td>
</tr>
<tr>
<td>Diana</td>
<td>Division Manager</td>
<td>17EF83CA ...</td>
<td>128C4</td>
</tr>
</tbody>
</table>