## Technical issues

- **basic concepts**
  - certificate, certification authority, certificate chain (or path), certificate update and revocation

- **PKI components and architectures**
  - CA, RA, repository, archive, users; hierarchical, mesh, and bridge architectures

- **life cycle of a certificate**
  - application, issuance, distribution and use, revocation, expiration

- **key-pair management issues**
  - key-pair generation, private-key protection, management requirements for different key-pair types

- **X.509 certificates and revocation lists**
The need for certificates

- distribution of public keys
  - confidentiality is not needed
  - authenticity is indispensable

- public keys can be distributed via secure out-of-band channels
  - physical contact
  - download public key from website and check its hash value via phone

- these solutions are not always practical and they don’t scale

Basic idea of public-key certificates

- concept invented by Kohnfelder in 1978 in his BSc thesis at MIT
- name and public key is linked together by the digital signature of a trusted entity called certification authority (CA)

- in order to verify a certificate you need to have an authentic copy of the public key of the CA
- advantages:
  - only the CA’s public key need to be distributed via out-of-band channels (scales better)
  - certificates can be distributed without any protection (why?)
Certificate chains

- a single CA cannot issue certificates to everyone in the world
  - practically infeasible
  - a single CA wouldn’t be trusted by everyone

- if there are more CA’s, then the following may happen:
  - you have a public-key certificate [Alice, KAlice, TrustMe, SigTrustMe]
  - you don’t have the public key of TrustMe
  - but you may have a certificate that contains TrustMe’s public key
    [TrustMe, KTrustMe, SuperTrust, SigSuperTrust]
  - …

- a certificate chain is a sequence Cert_1, Cert_2, …, Cert_k of certificates, such that
  - Cert_i = [S_i, KS_i, CA_i, SigCA_i] (i = 1, 2, …, k)
  - S_1 = Alice, S_i = CA_{i-1} (i = 2, …, k)
  - the public key of CA_k is known

Certificate chains (cont’d)

- notes:
  - all CAs in the chain must be trusted by the verifier in order to be able to accept the public key of Alice
  - there may be multiple chains leading to Alice some of which may be trusted

- example:
  - consider two certificate chains leading to A:
    
    \[ [A, KA, CA_1**, SigCA_1] [CA_1, KCA_1, CA_2*, SigCA_2] [CA_2, KCA_2, CA_3*, SigCA_3] \]
    \[ [A, KA, CA_1**, SigCA_1] [CA_1, KCA_1, CA_4*, SigCA_4] \]
  - Red has K_{CA_3}, and trusts CA_1, CA_2, and CA_3
  - Blue has K_{CA_4}, and trusts CA_1, and CA_4
  - Red accepts the first chain, Blue accepts the second chain
Validity periods and revocation

- for security reasons, key-pairs shouldn't be assumed to be valid forever
  - certificates have a scheduled validity period
  - \( \text{Cert} = [S, K_S, \text{valid}_\text{from}, \text{expires}_\text{on}, CA, \text{Sig}_\text{CA}] \)
  - a certificate shouldn't be used outside its validity period …
  - … unless it is to reconfirm an earlier action in the same way as it would have occurred within the validity period (e.g., to verify the signature on an old document)

- if a private key is compromised or suspected to be compromised, then the corresponding certificate needs to be revoked
  - certificate revocation is the dark side of public-key crypto

PKI components: Certification Authority (CA)

- collection of hardware, software, and staff (people)

- main functions:
  - issues certificates for users and other CAs
  - maintains certificate status information and Certificate Revocation Lists (CRLs)
  - publishes currently valid certificates and CRLs
  - maintains archives of status information on expired certificates

- must comply with strict security requirements related to the protection and usage of its private keys (basis of trust)
  - uses tamper resistant Hardware Security Modules that enforce security policies (access and usage control)
  - defines and publishes its certificate issuing policies
  - complies with laws and regulations
  - is subject to regular control (by national supervising authority)
### PKI components: Registration Authority (RA)

- Approves certificate applications, but doesn't itself issue certificates (RA is not CA)
  - Identifies and authenticates subscribers, provides collected attribute information to CA
  - Authorizes requests for key-pair or certificate generation or recovery from back-up
  - Authorizes requests for certificate suspension or revocation
  - Authorizes requests for certificate suspension or revocation
- Distributes personal tokens (which contain issued private key) and collects obsolete tokens

### PKI components: repositories and archives

- **Repository**
  - A directory service for the distribution and management of certificates and certificate status information (e.g., CRLs)
  - Typically based on the X.500 directory standard (or a subset, such as the Lightweight Directory Access Protocol (LDAP))
  - Format of directory entries must be generally agreed on (standards)

- **Archive**
  - Long term storage of status information of expired certificates
  - Used to check if an old certificate was valid at a given time in the past (e.g., in case of disputes)
  - Integrity of the archive must be strongly protected
**PKI components: users**

- organizations or individuals that use the PKI, but do not issue certificates

- two types:
  - certificate holders
    - have certificate(s) (own key pairs)
    - can use their own private key(s) (e.g., to sign documents)
  - relying parties
    - relies on certificates (and other components of the PKI) to verify if a public key belongs to a given entity and is valid
  - individuals and organizations may be both certificate holders and relying parties

**PKI architectures: hierarchical**

- top-down hierarchy
- certificate chains always start at the root CA
- every relying party must know the public key of the root CA
PKI architectures: mesh

- CAs cross certify each other
- a relying party know the public key of a CA near itself (usually the one that issued its certificate)
- a path needs to be found and verified from that CA to the target certificate

PKI architectures: bridge

- bridge CA connects enterprise PKIs (regardless of their architecture)
- relying parties need the same information as before (public key of their root or nearest CA)
Life cycle of a certificate

Applying for a certificate

- subscriber registers with the CA
  - establishment of a relationship between the subscriber and the CA
  - general subscriber information is provided to the CA
    - e.g., name, address, ...

- subscriber requests a certificate from the CA
  - certificate request contains more specific information regarding the requested certificate
    - e.g., type of certificate, public-key, other specific fields requested for the certificate
  - may not be a conscious action (e.g., employee of a company)
  - in case of a third party CA, a subscriber must always explicitly request the issuance of a certificate and explicitly accept the issued certificate
Validation of application

- the CA needs to verify
  - the identity of the subscriber (subject authentication)
  - that the public-key and other subscriber information originates from the subscriber and have not been tampered with in transit (public-key verification)

- subject authentication
  - method depends on the type of the requested certificate (assurance level)
  - for high assurance level certificates, usually personal presence is required
    - subject presents identification documents
    - CA may obtain further information from third party databases
    - most reliable form of authentication
  - low assurance level certificates can be obtained via an entirely on-line process

- public-key verification
  - if the CA generates the key-pair, then the problem is trivial
  - if the public key is provided by the subscriber, then a certificate issued by another CA may attest that the given public key really belongs to the given subscriber

Certificate issuance

- certificate is signed by a signing device using the CA’s private key
- a copy of the certificate is forwarded to the subscriber
- a confirmation of acceptance is returned by the subscriber (if needed)
- a copy of the certificate is sent to the repository (directory service)
- a copy of the certificate is archived
- transaction is logged in an audit journal
Distribution of certificates

- via the repository (directory) service
- individually
  - the signer usually has a copy of her certificate
  - attach the certificate (chain) to the digitally signed document
- potential disadvantages:
  - waste of bandwidth or storage space, as the verifier may already have a copy of the certificate
  - multiple chains may exist from the verifier to the signer; which one to attach?
- other means
  - DNS
  - web
  - e-mail

Certificate revocation

- sometimes certificates need to be revoked before their expiration time
  - detected or suspected key compromise
  - change of data contained by the certificate (e.g., name, e-mail)
  - change of subject-CA relationship (e.g., employee leaves the company)
- who can request a revocation?
  - the subscriber is authorized to request the revocation of her own certificate
  - officers of the CA are also authorized to revoke a certificate under well-specified circumstances
  - other people may be authorized (e.g., employer)
  - in any case, the requesting party is authenticated by the CA and a log is generated
  - (RA may have the responsibility to approve revocation requests)
Certificate Revocation Lists (CRL)

- a CRL is a time-stamped list of revoked certificates signed by the CA and made available to certificate users (e.g., published regularly in the directory or on a web site)

CRLs (cont’d)

- the CA issues CRLs regularly (hourly, daily, or weekly)
- a new CRL is issued even if no new revocations happened since the last CRL (why?)

- advantages:
  - CRLs can be distributed in the same way as certificates
  - no need for trusted servers and secure communication links

- disadvantage:
  - time granularity is limited to CRL issue period
    - key is suspected to be compromised now, but certificate users will be aware of that only when the next periodic CRL is issued

Broadcast CRLs

- periodic publishing of CRLs → pull model
- an alternative → push model
  - CA broadcasts CRLs to certificate using systems as new revocations are posted

- advantage
  - critical revocations can be distributed very quickly

- disadvantages
  - needs reliable distribution methods (why?)
  - potentially large overhead due to broadcast if all revocations are reported in this way
  - absence of standards

- push and pull can be combined! (how?)

Immediate revocation

- the CA can operate a trusted on-line server that can be queried for the revocation status of a given certificate in real-time
  - server’s response must be authenticated and its freshness must be ensured
  - server should be highly available to users
  - revocation requests must be quickly processed by the CA

- example: OCSP - Online Certificate Status Protocol (RFC 2560)
Key-pair generation

- on the key owner’s system
  - possibly in the hardware (smart card) or software module where the private key will be stored later
  - preferable for digital signature keys (easier to ensure non-repudiation as the private key never leaves the key owner’s system)
- on the CA’s system
  - private key should be securely transported to the key owner’s system
  - higher quality keys can be generated (more resources, stronger controls)
  - preferable for encryption keys (if private key needs to be backed up or archived)

Private-key protection

- protection of the private-key from unauthorized access is of paramount importance
- the private key is typically stored in
  - tamper resistant hardware module or token (e.g., smart card, PCMCIA card, …)
  - encrypted file within a computer or regular data storage media (e.g., CF card, USB key, …)
- access to the key needs to be protected via one or more authentication mechanisms
  - typically, passwords and PINs
    - can be used directly in case of hardware tokens
    - encryption keys can be derived from them in case of encrypted files
  - biometric checks
Management requirements for different key types

- RSA has the interesting property that one key pair can be used for both encryption and digital signature
- however, such double use of key-pairs is not advisable; users should have different key-pairs for different applications
- the main reason is in the difference in key management requirements
  - digital signature
    - private key should never leave the key owner’s system
    - private key doesn’t need back up and archive (why?)
    - public key (certificate) needs to be archived
  - encryption
    - private key often needs to be backed up and archived (why?)
    - public key usually doesn’t need to be archived
→ the two applications have conflicting requirements

Management requirements … (cont’d)

- more reasons
  - in general, an encryption key pair may not necessarily be used for digital signature (and vice versa)
    - it is better to design a system assuming that different algorithms (and thus key-pairs) will be used for digital signature and encryption
  - implementations that support encryption may be subject to more strict export controls
    - length of encryption key is often limited
    - if the same key is used for digital signature, then the digital signature key is smaller than it could be
  - key escrow
    - private keys used for encryption may be made available for government officials for escrow purposes
    - digital signature keys should not be disclosed in this way
Basic X.509 certificate format (v1 and v2)

- X.509 version (currently 1, 2, or 3)
- unique identifier of this certificate assigned by the issuing CA
- object identifier of the signature algorithm used to sign this certificate
- start and expiry date of the certificate
- value of the public key together with the object identifier of the algorithm with which the key should be used
- bit strings used to make the CA's and the subjects name unambiguous in case the same name has been reassigned to different entities through time (optional, version 2 only)
- signature of the issuing CA

X.500 names

- in X.509 v1 and v2, X.500 names are used to identify subjects and issuers
- it is assumed that the subject and the issuer both have an X.500 directory entry (they are registered in the directory)
- X.500 directory entries are logically organized in a tree (Directory Information Tree - DIT)
- each entry (except the root) has a distinguished name (DN)
- the DN for an entry is constructed by joining
  - the DN of the parent in the DIT, and
  - a relative distinguished name (RDN)
    - a collection of attribute values that distinguishes this entry from other children of its parent
    - usually, the collection consists of a single attribute value
**X.500 names (an example)**

```
root
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Canada</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>USA</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>US government</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Danielle's Machine Makers</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Canadian government</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
  |                   root

RDN: \(\text{Country} = \text{USA}\)
DN: \(\{C = \text{USA}\}\)
RDN: \(\text{Org} = \text{Sharon's Steel}\)
DN: \(\{C = \text{USA}, O = \text{Sharon's Steel}\}\)
RDN: \(\text{CN} = \text{Roy Mills Title = CEO}\)
DN: \(\{C = \text{USA}, O = \text{Sharon's Steel}, CN = \text{Roy Mills Title = CEO}\}\)
```

**Object registration**

- International standards describe how different objects (e.g., algorithms) should be identified and registered.
- Object identification works on the basis of a hierarchical structure of different value assigning authorities (e.g., national standard organizations).
- Each authority is responsible for managing its sub-tree.
- Example: Sharon's Steel has a super hash function.

```
0 (itu-t) 1 (iso) 2 (joint-iso-itu-t) 16 (country) 840 (us) 1 (organization) 15678 (sharons) 1 (policies) 2 (algorithms) 66 (sharons-hash)
```

Object id of Sharon's super hash function: 2 16 840 1 15678 2 66
Object registration (more examples)

- **DSA digital signature with SHA-1**
  
  **object id:**
  
  iso (1) member-body (2) us (840) x9-57 (10040) x9algorithm (4)
  
  dsa-with-sha1 (3)
  
  **source of spec:**
  
  ANSI X9.57

- **RSA digital signature with MD5**
  
  **object id:**
  
  iso (1) member-body (2) us (840) rsadsi (113549) pkcs (1) pkcs-1 (1)
  
  md5withRSAencryption (4)
  
  **source of spec:**
  
  RSA Data Security Inc. PKCS #1

Defects of X.509 v1 and v2

- **multiple certificates per subject**
  
  - the same subject needs different certificates for different key-pairs
  
  - X.509 v1 and v2 cannot distinguish different certificates conveniently
    (only via serial number)

- **additional subject identifying information**
  
  - X.500 DN doesn't contain enough information to identify the subject

- **application specific name forms**
  
  - some applications need to identify users by using application specific name-forms
  
  - e.g., for e-mail, the public key should be bound to an e-mail address

- **certification policies**
  
  - different certificates are issued under different policies
  
  - certificate users need to know the level of assurance that they can have in a given certificate

- **certification paths**
  
  - when CA X issues a certificate to CA Y, X may want to recognize only a subset of the certificates issued by Y and its subordinate CAs
  
  - there's a need to limit the length of certificate chains
Technical issues / X.509 certificates and CRLs

X.509 v3 certificate format

```
+---------------------------------+------------------+
| Ext type                        | Critical/Non-critical |
| Ext value                       |                  |
+---------------------------------+------------------+
| Ext type                        |                  |
| Ext value                       |                  |
| Critical/Non-critical           |                  |
+---------------------------------+------------------+------------------+
| Ext type                        |                  |
| Ext value                       |                  |
| Critical/Non-critical           |                  |
+---------------------------------+------------------+
```

Extensions

Signature of issuer

X.509 v3 certificate format (cont’d)

- extension types must be registered (see object registration)
  - communities can define their own extension types
  - most important extension types are standardized

- critical / non-critical flag
  - non-critical:
    - if you don’t know this ext type, you can safely ignore this extension
    - e.g.: e-mail address or alternative name
  - critical:
    - it is safe to use this certificate only if you recognize this extension type (you shouldn’t ignore this extension)
    - e.g.: an extension that limits the type of applications where the certificate should be used

- critical extensions lead to interoperability problems
  → most extensions should be flagged non-critical
### Standard certificate extensions

- **Key and policy information**
  - These extensions are used to convey additional information about the subject and the issuer keys (e.g., key identifier)
  - Help to find certificate chains

- **Subject and issuer attributes**
  - These extensions support alternative names and convey more attribute information (e.g., postal address, phone number)

- **Certification path constraints**
  - These extensions help to limit the length of certificate chains

- **Extensions related to CRLs**
  - ...

### Key and policy information extensions

- **Authority Key Identifier**
  - Can be used to distinguish different signing keys of the issuing CA
    - Explicit key id
    - A pointer to another certificate, where the signing key is certified by another CA (pointer can be the name of the CA and serial number of the certificate)

- **Subject Key Identifier**
  - Enables different keys used by the same subject to be distinguished conveniently

- **Key Usage**
  - Indicates the purpose for which the key should be used (e.g., digital signature, non-repudiation, key-encryption, DH key agreement, data encryption, certificate signing, CRL signing)
  - Usually flagged as critical

- **Private-Key Usage Period**
  - Validity of a certificate can be longer than the period in which the private key is effectively used for signing
  - This extension indicates the usage period of the private key

- **Certificate Policies**
  - Identifies policies under which the CA issued the certificate
Certificate policies

- A certificate policy is a named set of rules and practices that are applied by the CA when issuing certificates and that should be followed by the certificate users when they use certificates.
- It may contain:
  - Community and applicability restrictions
  - Identification and authentication policy
  - Key management policy
  - Operational policy
  - Local security policy
  - Legal provisions
  - Policy administration
- Certificate policies need to be registered (see object registration).

Subject and issuer attribute extensions

- **Subject Alternative Name**
  - This extension can carry an alternative name of the subject (e-mail address, domain name, web URL, ...)

- **Issuer Alternative Name**
  - Same as for subject

- **Subject Directory Attributes**
  - Provides general means for conveying additional information about the subject
  - Contains X.500 attribute values (e.g., Phone = +1 212 222 2222)
Certification path constraint extensions

- **Basic Constraints**
  - indicates whether the subject can act as a CA
  - if so, then it may also specify the length of a certificate chain that may stem from this certificate

- **Name Constraints**
  - restricts the name space that will be considered acceptable in subsequent certificates in any chain stemming from this certificate
  - example:
    - subject is bme.hu
    - the subject name of any further certificate should end with bme.hu
    - acceptable names: hit.bme.hu, vik.bme.hu
    - non-acceptable names: crysys.hu, epfl.ch

- **Policy Constraints**
  - restrictions on the policies that may be used by the CAs that issued the certificates following this certificate in the chain

X.509 CRL format

- Version of CRL format
- Signature Alg ID
- CRL issuer's X.500 name
- Date/time of this update
- Date/time of next update
- Revoked certificate
- Revoked certificate
- ... (CRL Extensions)
- Certificate Serial #
- Revocation Time
- CRL Entry Extensions

- Signature of CRL issuer

CRL and CRL Entry Extensions are allowed only in version 2

The extension mechanism is the same as for X.509 v3 certificates (including the critical/non-critical flag)
Extensions

- general extensions
- CRL distribution points
- Delta-CRLs
- Indirect CRLs
- Certificate suspension

General extensions

- **CRL Number** (CRL extension)
  - helps a certificate user to see if any past CRLs has been missed
  - also needed to support Delta-CRLs

- **Reason Code** (CRL entry extension)
  - gives a reason of the revocation: Key Compromise, CA Compromise, Affiliation Change, Superceded, Cessation of Operation, Certificate Hold

- **Invalidity Date** (CRL entry extension)
  - indicates a date at which the revoked compromised key was known to still be good

- **Authority Key Identifier** (same as for X.509 v3)

- **Issuer Alternative Name** (same as for X.509 v3)
Issues related to CRL size

- when verifying a certificate, the appropriate CRL needs to be fetched and verified
- to avoid communication and processing overhead, CRLs shouldn’t be very large
  - size of a CRL depends on
    - size of the population
    - certificate validity period (expired certs need not be kept on CRL)
  - reducing the validity period is undesirable
    - more user inconvenience
    - higher demand on archive resources
  → population size must be limited
- a useful technique is the following (used in early versions of X.509)
  - each CA maintains two CRLs
    - one for revoked end-user certificates
    - another for revoked CA certificates (very short CRL, usually empty)
  - in a certificate chain, there’s only one end-user certificate and multiple CA certificates → a potentially long CRL need to be processed only for the verification of the end-user certificate
- growing end-user population is still a problem

CRL distribution points

- in the current version of X.509, this problem is solved by
  - allowing to arbitrarily partition the population
  - associating a CRL distribution point to each partition
    - the CRL distribution point is not a CA; it doesn’t issue CRLs
  - inserting a pointer in the certificate to the CRL distribution point where revocation of this certificate may appear (certificate extension)

- supporting extensions:
  - Certificate Distribution Points (certificate extension)
    - identifies the CRL distribution points where a revocation of this certificate can appear
    - identifies the CRL issuer (if not the same as the certificate issuer, see Indirect CRLs later)
    - can be an X.500 name, a web URL, an e-mail address ...
  - Issuing Distribution Point (CRL extension)
    - gives the name of the CRL distribution point for this CRL
    - signed by the CA that issued the CRL (together with other entries of the CRL)
    - prevents attackers from substituting an empty CRL obtained from distribution point A in place of a non-empty CRL at distribution point B
Delta CRLs

- another mechanism to reduce the size of CRLs
- a delta-CRL is a digitally signed list of changes that have occurred since the issuance of the last complete CRL
  - reduces communication overhead
  - certificate using systems should be capable of maintaining their own database of certificate revocation information
  - the delta-CRL is used to update these local databases
- supporting extension:
  - Delta CRL Indicator (CRL extension)
    - identifies the CRL as being a delta-CRL only
    - carries the CRL number of the base CRL (the complete CRL to which the changes should be applied)

Indirect CRLs

- it is possible that the CRL is issued by a different CA than that which issued the certificates concerned
- thus, one CRL can contain revoked certificates issued by different CAs
- advantage:
  - a CRL can be created that contains ALL revoked CA certificates (not only those issued by a given CA)
  - when verifying a certificate chain, the user needs to fetch only two CRLs
    - the above indirect CRL (to verify the revocation status of every CA in the chain)
    - the end-user CRL of the last CA in the chain (to verify the status of the target certificate)
- supporting extensions:
  - CRL Distribution Points (certificate extension)
    - identifies the CRL issuer that issues CRLs on which a revocation of this certificate can appear
  - Certificate Issuer (CRL entry extension)
    - indicates who was the issuer of this revoked certificate
Certificate suspension

- sometimes it is not clear whether a certificate should be revoked or not
- examples:
  - an unusually high value e-banking transaction
    - Alice pays her bills using e-banking; she transfers a rather small amount from her account every month
    - once Alice decides to buy a car; she transfers a huge amount from her account
    - this is suspicious!
  - two transactions in a short time but far apart from each other
    - Alice uses a digital check system, where checks are signed by her smart card
    - the bank receives two checks one signed at 10:17 in the US, and another signed at 10:35 on the same day in Germany
    - this is suspicious too!
- supporting extension:
  - Reason Code (CRL entry ext) = Certificate Hold

Attribute certificates

- identity certificates bind a public key to a subject
- attribute certificates bind one or more attribute values to a subject
- attribute certificates can be used for flexible authorization
- why should public-key certificates be used with caution for authorization?
  - the authority that is appropriate to certify identity of a person associated with a public key may not be appropriate to certify authorization information
  - differences in dynamics of the two types
    - the person authorized to perform a particular function may vary monthly, weekly, or even daily
    - public-key certificates are typically valid for a year or more
- X.509 supports attribute certificates
  - they have a similar structure and managed in a similar way (including revocation) as public-key certificates