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 web site 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 CAs, then the following may happen:
  - You have a public-key certificate [Alice, $K_{Alice}$, TrustMe, $\text{Sig}_{\text{TrustMe}}$]
  - You don’t have the public key of TrustMe
  - But you may obtain a certificate that contains TrustMe’s public key [TrustMe, $K_{\text{TrustMe}}$, SuperTrust, $\text{Sig}_{\text{SuperTrust}}$]
    - ...

- A certificate chain is a sequence $\text{Cert}_1, \text{Cert}_2, \ldots, \text{Cert}_k$ of certificates, such that
  - $\text{Cert}_i = [S_i, K_{S_i}, \text{CA}_i, \text{Sig}_{\text{CA}_i}]$ (i = 1, 2, ..., k)
  - $S_1$ = Alice, $S_i$ = $\text{CA}_{i-1}$ (i = 2, ..., k)
  - The public key of $\text{CA}_k$ 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:
    
    \[
    [A, K_A, \text{CA}_1**, \text{Sig}_{\text{CA}_1}] [\text{CA}_1, K_{\text{CA}_1}, \text{CA}_2**, \text{Sig}_{\text{CA}_2}] [\text{CA}_2, K_{\text{CA}_2}, \text{CA}_3**, \text{Sig}_{\text{CA}_3}] \\
    [A, K_A, \text{CA}_1**, \text{Sig}_{\text{CA}_1}] [\text{CA}_1, K_{\text{CA}_1}, \text{CA}_4**, \text{Sig}_{\text{CA}_4}] 
    \]
  - Red has $K_{\text{CA}_3}$, and trusts $\text{CA}_1$, $\text{CA}_2$, and $\text{CA}_3$
  - Blue has $K_{\text{CA}_4}$, and trusts $\text{CA}_1$, and $\text{CA}_4$
  - 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
  - Cert = [S, Ks, valid_from, expires_on, CA, SigCA]
  - 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
  - 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 knows 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

- certificate application
- validation of application
- certificate issuance
- acceptance of certificate by subscriber
- distribution and use of certificates
- certificate expiration and renewal
- certificate revocation (if needed)
- certificate suspension (if needed)

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 (if provided) 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 may 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?
  - advantage:
    - easier to archive signed document with the corresponding certificate chain (and certificate status information)

- other means
  - web
  - e-mail
  - DNS

**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).

<table>
<thead>
<tr>
<th>issuer's name</th>
<th>CRL issue time/date</th>
</tr>
</thead>
<tbody>
<tr>
<td>certificate serial # revocation time/date</td>
<td></td>
</tr>
<tr>
<td>certificate serial # revocation time/date</td>
<td></td>
</tr>
<tr>
<td>certificate serial # revocation time/date</td>
<td></td>
</tr>
<tr>
<td>issuer's signature</td>
<td></td>
</tr>
</tbody>
</table>

The CA's private key is used for digital signature process.

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.

Key compromise: revocation

- Issue of CRLi
- Revocation requested
- Issue of CRLi+1

(a) (b) (c) (d) (e)
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
  - a tamper resistant hardware module or token (e.g., smart card, PCMCIA card, ...)
  - an 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 the same 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)**

```
Common Name = Roy Mills
Title = CEO
Telephone = +1 212 222 2222
E-mail = rmills@sharons.com
```

```
RDN: {Country = USA}
DN: {C = USA}
RDN: {Org = Sharon's Steel}
DN: {C = USA, O = Sharon’s Steel}
RDN: {CN = Roy Mills, Title = CEO}
DN: {C = USA, O = Sharon’s Steel, CN = 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)
                          ANSI
                          Sharon's Steel Corp.
                          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) md5withRSAencyption (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
- delegation and limitations
  - 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

**X.509 v3 certificate format**

- same as in v2
- Extensions
- Signature of issuer

---

Technical issues / X.509 certificates and CRLs

© Levente Buttyán
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 easily find the issuing CA's public key (needed to verify the certificate)
  - can be an explicit key id
  - or 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
- may contain:
  - community and applicability restrictions
    - e.g., Sharon's CA issues certs for Sharon's employees and for signing e-mails only
  - identification and authentication policy
    - practices followed by the CA to authenticate certificate subjects
  - key management policy
    - the measures taken by the CA to protect its own crypto keys
  - operational policy
    - e.g., specifies the frequency at which the CA issues CRLs
  - local security policy
    - the measures taken by the CA to protect its computing environment
  - legal provisions
    - a statement of limitations of liability
  - policy administration
    - identification of the policy defining authority and indication of how the policy definition is maintained and published
- 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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version of CRL format</td>
<td>version 1 or 2</td>
</tr>
<tr>
<td>Signature Alg ID</td>
<td>CRL and CRL Entry Extensions are allowed only in version 2</td>
</tr>
<tr>
<td>CRL issuer’s X.500 name</td>
<td></td>
</tr>
<tr>
<td>Date/time of this update</td>
<td></td>
</tr>
<tr>
<td>Date/time of next update</td>
<td></td>
</tr>
<tr>
<td>Revoked certificate</td>
<td></td>
</tr>
<tr>
<td>Revoked certificate</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>CRL Extensions</td>
<td></td>
</tr>
<tr>
<td>Certificate Serial #</td>
<td></td>
</tr>
<tr>
<td>Revocation Time</td>
<td></td>
</tr>
<tr>
<td>CRL Entry Extensions</td>
<td></td>
</tr>
<tr>
<td>Signature of CRL issuer</td>
<td></td>
</tr>
</tbody>
</table>

### Extensions

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

The extension mechanism is the same as for X.509 v3 certificates (including the critical/non-critical flag).
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 storage*) overhead, CRLs shouldn’t be very large
- 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

- 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

* CRLs may be archived together with signed documents
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 the one that issued the certificates on the CRL
- 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.