Reputation, Authorization and Public Key Infrastructure

Last updated: Sunday, December 18, 2005

© Prof. Amir Herzberg
Computer Science department, Bar Ilan University
“On the Internet, nobody knows you’re a dog”

- This is good!
  - No prejudice
  - No discrimination
  - Size doesn’t matter
  - Don’t need suits and fancy buildings
  - More efficient!!

- But…

The above cartoon by Peter Steiner has been reproduced from *The New Yorker*, (Vol.69 no. 20) only for academic discussion, evaluation, research and complies with copyright law.
“On the Internet, nobody knows you’re a dog”

- This is bad...
  - Fraud
  - Quality
  - Accountability
  - Security

- Solutions:
  - Reputation
  - Identification

The above cartoon by Peter Steiner has been reproduced from *The New Yorker*, (Vol.69 no. 20) only for academic discussion, evaluation, research and complies with copyright law.
Lecture Outline: Public Key Infrastructure

- Public Key Certificates
- Identity in certificates – different approaches:
  - Unique identifier – X.509 Distinguished Names
  - Optional / non-global identifiers
- X.509 Public Key Certificates
- Certificate authorities, hierarchies and cross certification
- Certificate issuing and registration authorities
- Certificate (Path) Validation
- Certificate Revocation
- Conclusions
Consider multiple parties (entities, agents)
With (often) adversarial interests
Ensure (some) interests of some parties
  - Often viewed as preventing threats / risks

How?
  - Deter and punish adversarial behavior
    - Education, Punishment, Incentives
  - Prevent damage in spite of adversarial behavior

This is very general - economy, legal,…
Let’s focus on information (computer) science…
Security by Reputation and Identification

- Deter and punish adversarial behavior / reward good behavior
  - By identification + proof (e.g. to court)
    - Proof of wrongdoing (what was done)
    - Identification of adversary (who’s done it)
  - By reputation (reviews, history)

- Prevent damage in spite of adversarial behavior
  - By identification (white/black lists)
  - By reputation: avoid bad guys (work only with reputable partners)
Relying on Public Keys

- Public keys are very useful
  - Encrypting data and keys
  - Signing documents
- How do we know the public key?
- Initial approach: public keys will be registered in (e.g. X.500) directory/repository
  - Trusted, centralized
  - Subject uniquely identified
  - Directory matches subject to public key
  - Public key authenticated btw directory and relying party
    - Using MAC or signed by directory
  - Possibly other attributes in directory
Public Key Certificates

- **Issuer** (or **Certification Authority – CA**) signs **certificate** binding public key to different attributes of key owner
- **Relying party** validates **certificate** signed by **issuer**
- What are the attributes?
  - Should help the relying party decide on subject
  - Issuer (CA) should be able to validate them (liability!)
  - Usually: include **identity** (ID)

\[
\text{Cert} = \text{Sign}_{\text{CA.s}}(\text{Subject.v, ID, ATTTR})
\]

\[
\text{Cert, Subject.v, Sign}_{\text{Subject.s}}(m)
\]

- **Issuer / CA** (Certificate Authority)
- **Subject** (key owner)
- **Relying Party**

---

Public Key Certificates

- Similar to passport or driver’s license
- Binds a public key to a name and/or other attributes of keyholder, e.g. DNS name for web site
- Signed by a trusted party (Issuer / Certification Authority)
- Allows relying party (Bob, client) to validate name, attributes of key owner (Alice, web site)

**BIU Library**
- Name: Alice
- ID 3783597

**Signature:**
- Issued by:

**Issuer**
- Issuer’s Signature: \( \text{Sign}_{\text{Lib}}(\text{Info}) \)

**Info**
- (DNS) Name
- Attributes
- Public Key
Certificates and Hash Collisions

- Certificates (w/ attributes, encoding) are long
- Use `Hash then Sign` signatures, e.g. RSA_SHA1, RSA_MD5, DSA [always SHA1]
  - \( RSA\_MD5\_Sign_{A.s,A.n}(m) = <m, [MD5(m)]^{A.s} mod A.n> \)
- Problem: collisions recently found for MD5
  - Can find collision for any IV of MD5’s collision function any known prefix of the message
  - Beware of certificates using MD5!
    - known attack: two certs (diff PK) with same signature
    - Open problem: trick CA to sign a collision → fake credentials
- Also notice recent attack on SHA-1 (still \(2^{69}\))
Certification Authorities

- **Certification authority (CA):** binds public key (e.g. *B.v, B.e*) to identifier (e.g. *B.name=`Bob`*).
- **Bob (person, server) registers** *B.v, B.e* with CA.
  - Bob convinces the CA his name is Bob, sends *B.v, B.e*.
  - CA creates certificate binding “Bob” to *B.v, B.e*.
  - Certificate is digitally signed by CA – CA says “*B.e is Bob’s public encryption key*”

![Diagram of certification process]

Bob’s public key

Bob’s identifying information

CA

CA private key

digital signature

certificate for *B.e* (as Bob’s key), signed by CA

B.e
Registration Authority

- CA combines two functions:
  - Validate identity of source of public key
  - Sign public key with attributes (identity, others)
- CA secret key required only to sign cert
- Identify by separate registration authority

Alice, PubA, MAC_k(Alice, PubA) → CA (issuer)
Cert_A = Sign_CA(Alice, PubA)

Alice proves her identity and provides Pub_A.
Registration Authority (RA)

Alice (subject)

k (shared key)
Using Public Key Certificates

- When Alice wants Bob’s public key (to encrypt message to Bob or validate Bob’s signature):
  - gets Bob’s certificate (Bob or elsewhere).
  - apply CA.v (public validation key) to Bob’s certificate, get Bob’s identity, public keys, …
Using Public Key Certificates
(another view)

CA (issuer)

Alice (subject)

Secret signature key $A_{priv}$

Bob (relying party)

CA’s public signature validation key $CA_{pub}$

If $Valid_{CA_{pub}}(c,\{a,p\})$
And $a=Alice$
And $Valid_{p}(s_{m},m)$
then Alice signed $m$
otherwise reject $m$

Alice proves her identity
And provides $A_{pub}$

$Cert_{A} = Sign_{CA_{priv}}(Alice,A_{pub})$

$m$ $Sign_{A_{priv}}(m)$ $\{Alice, A_{pub}\}$

$m$ $s_{m}$ $\{a,p\}$

$c$
What are Certificate Attributes?

- Let $AttrNames$ be the set of attribute names
- Let $Values$ be the set of attribute values
- Let $A = AttrNames \times Values$ be the set of attributes
  - Each attribute is a pair $<n \in AttrNames, v \in Values>$
  - These are called `name-value pairs`.

Attributes can be:
- Properties of the certificate (e.g. $<valid,1-6/2001>$)
- Properties of subject (e.g. $<grade,A>; <job,cop>$)
- Identifiers (e.g. $<email,avi@biu.ac.il>; <ID,5724567>$)
A public key certificate (PKC) is a 4-tuple:
\(<\text{Issuer}_{\text{pub}}, \text{Subject}_{\text{pub}}, \text{Attrs}, \text{Sign}>\), where:

- \(\text{Issuer}_{\text{pub}}, \text{Subject}_{\text{pub}}\) are public keys
- \(\text{Attrs} \subseteq \text{Attributes}\)
- \(\text{Sign}\) is a signature using \(\text{Issuer}_{\text{priv}}\) over \(\text{Subject}_{\text{pub}}\) and \(\text{Attrs}\); namely…
- The certificate is valid if
  \[\text{Valid}_{\text{IssuerPub}}(\text{Sign}, \{\text{Subject}_{\text{pub}}, \text{Attrs}\})\]
Identity Public Key Certificates

- Identity PKC: certificate contains **identifier** (s)
- Examples: name, picture, signature, social security/identity #...
- Use identifiers to:
  - Determine services (e.g. name)
  - Penalize `bad` actions: sue, blacklist, …
- We do **not** refer to using identifiers to identify requestor
  - E.g. public key, picture, fingerprint, signature…
- Desirable properties of identifiers:
  - Unique: no two entities with same identifier
  - Memorable, easy to specify (e.g. allow access to…)
  - Permanent: entity can be identified by identifier
    - Impossible/hard/expensive to have many identifiers
  - Legal or commercial meaning (off-net)
    - Allow use of existing, off-net legal/reputation mechanisms
X.500, X.509 and Distinguished Names

- Distinguished Name (DN): attempt for unique, memorable, permanent and legally-binding identifiers
  - Defined in X.500, X.509
- X.500: ITU’s recommendation (standard) for global, distributed, trusted, on-line directory (phone book)
  - Unique identifier: *Distinguished Name (DN)*
  - Operated by hierarchy of trustworthy directories
  - Never happened – too complex, too revealing
  - Different attributes, including public key
- X.509: authentication related to X.500
  - Initially: Authenticate entity to Directory (PW, Pub key)
    - To maintain entity’s record
  - X.509 certificate binds public key to distinguished name (DN)
  - X.509 is used widely: IETF PKIX, SSL, PGP, S/MIME, IP-Sec, …
# X.509 Public Key Certificates

<table>
<thead>
<tr>
<th>Field</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td></td>
</tr>
<tr>
<td>Certificate serial number</td>
<td></td>
</tr>
<tr>
<td>Signature Algorithm Object Identifier (OID)</td>
<td></td>
</tr>
<tr>
<td>Issuer Distinguished Name (DN)</td>
<td></td>
</tr>
<tr>
<td>Validity period</td>
<td></td>
</tr>
<tr>
<td>Subject (user) Distinguished Name (DN)</td>
<td></td>
</tr>
<tr>
<td>Subject public key information</td>
<td>Public key Value, Algorithm Obj. ID (OID)</td>
</tr>
<tr>
<td>Issuer unique identifier (from version 2)</td>
<td></td>
</tr>
<tr>
<td>Subject unique identifier (from version 2)</td>
<td></td>
</tr>
<tr>
<td>Extensions (from version 3)</td>
<td></td>
</tr>
<tr>
<td>Signature on the above fields</td>
<td></td>
</tr>
</tbody>
</table>

Signed fields
Object Identifiers (OID)

- From Abstract Syntax Notation (ASN.1) standard
- Global, unique identifiers, e.g. for algorithms
- Sequence of numbers, e.g.: 1.16.840.1.45.33
- Top level numbers: 0 – ITU, 1 – ISO, 2 – joint
- Each organization assigns lower-level identifiers
- X.509 use: identify algorithms and extensions.
Distinguished Names (DN)

- **Goal:** unique, memorable, permanent and legally-binding identifiers
- **Ordered sequence of** (pre-defined and other) keywords, and a string value for each of them.
- **Distributed directory, responsibility** → *hierarchical DN*

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Country</td>
</tr>
<tr>
<td>L</td>
<td>Locality name</td>
</tr>
<tr>
<td>O</td>
<td>Organization name</td>
</tr>
<tr>
<td>OU</td>
<td>Organization Unit name</td>
</tr>
<tr>
<td>CN</td>
<td>Common Name</td>
</tr>
</tbody>
</table>
Distinguished Name Hierarchy

Comments:
1. Other keywords Ok
2. No strict usage rules (hierarchy)
3. Often mapped to CA hierarchy …

DN={C=US/L=NY/O=NYPD/OU=soho/CN=John Doe}
Distinguished Names - Problems

- Distinguished Name fields may expose
  - Organizational sensitive information (e.g. position)
  - Privacy, possibly allowing identity theft
- Validation of fields is expensive, intrusive (why should ISP ask for my ID/SSN)?
- And…Is it unique?
  - Yes – by design, each
- Is it memorable?
  - Hope that it is – by structure
  - But: two persons in same organization unit (Julian Jones2/UK/IBM)
- Is it legally-binding?
  - Not directly – see Julian Jones2/UK/IBM…
  - Also: could be subject to identity theft
- Is it permanent?
  - Same person can be in different organizations… or move!
- And… it requires global, hierarchical naming organization…
Distinguished Names – in Practice

- Legally acceptable identifiers in some countries.
- To ensure uniqueness, issuers often place a random string or serial number as part of the DN.
- As of Version 2, X.509 certificates contain additional `unique identifiers` for the subject and issuer.
- As of Version 3, X.509 certificates allow general extensions, that are often used to add identifiers.
- Used to implement *access control / authorization*
  - Which requests to grant and which to deny.
Access Control / Authorization

- Relying party (server) receives request
  - Request: some action, on some object
    - Login, read/write file, transaction, etc.
  - With identity of subject
    - Authenticated: password, key, signature, SSL,…
- Authorization: decide if to grant request based on identity
Sign-On per Application Server

- Each user is given userid/password for each application server
- Application administrator sets user-id and permissions per user
- Sign-on (authentication & authorization) internal to the application

Multiple passwords
Multiple usernames
Confusion!!

ID: PW List
Alice: jak23
Bob: pw1

Access Control List
Alice: RX
Bob: RWX(~/Bob)

File Server

Multiple Administrators
High cost of administration
No overall Security Policy
Single Sign-On (no PKI)

- Idea: separate authentication from authorization
- Authenticate using PW / biometrics / device (e.g. phone)
  - Something you know, something you are, something you have
- Access control: per user, in application or in authorization server

Multiple passwords
Multiple usernames
Confusion!!

Multiple Administrators
High cost of administration
No overall Security Policy
Single Sign-On with Identity-based PKI

- User types password (or: biometrics), gets private key and cert
  - From single sign-on server, workstation, or personal device
- Authentication: using public key in cert (and signature / SSL)
- Authorization: by identity (e.g. Distinguished Name)
  - Per application or in central authorization server

Identifier (e.g. DN)

Access Control

<table>
<thead>
<tr>
<th>DN</th>
<th>Req</th>
<th>Ok?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dr. A</td>
<td>R/W</td>
<td>Y</td>
</tr>
<tr>
<td>Dr. B</td>
<td>R/W</td>
<td>Y</td>
</tr>
<tr>
<td>Eve</td>
<td>Any</td>
<td>N</td>
</tr>
</tbody>
</table>

Data entry by distinguished name – repetitious, error prone
Relying Party Use of (X.509) Identity Certificates: Identity-based Access Control

Signed Request Or Proposal | Certificate

Identify

User’s Distinguished Name (DN)

Policy

<table>
<thead>
<tr>
<th>DN</th>
<th>Req</th>
<th>Ok?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dr. A</td>
<td>R/W</td>
<td>Y</td>
</tr>
<tr>
<td>Dr. B</td>
<td>R/W</td>
<td>Y</td>
</tr>
<tr>
<td>Dr. C</td>
<td>R/W</td>
<td>Y</td>
</tr>
<tr>
<td>Eve</td>
<td>Any</td>
<td>N</td>
</tr>
</tbody>
</table>

Access control / Business logic

Ok?ReqDN

Y/N

12/18/2005

http://Amir.Herzberg.name
Relying Party Use of (X.509) Identity Certificates: Role-based Access Control

Signed Request Or Proposal

Certificate

Identify

Identity (e.g. Distinguished Name - DN)

Identity to Role

Mapping to role:
- Known user
- Role encoded in DN (e.g. */*/IBM ➞ IBM employee)

Request

Access control / Business logic

Policy

<table>
<thead>
<tr>
<th>Role</th>
<th>Req</th>
<th>Ok?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dr.</td>
<td>R/W</td>
<td>Y</td>
</tr>
<tr>
<td>Other</td>
<td>Any</td>
<td>N</td>
</tr>
</tbody>
</table>

Y/N
Certificate Path

- What if Bob (relying party) does not know Alice’s CA?
- Solution: Certificate Path – a CA known and trusted by Bob certifies Alice’s CA

![Certificate Path Diagram]

<table>
<thead>
<tr>
<th>Alice</th>
<th>CA2</th>
<th>CA1</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issuer DN: CA2</td>
<td>Subject DN: Alice</td>
<td>Alice’s pub. key</td>
<td>Alice is not a CA</td>
</tr>
<tr>
<td>CA2’s Signature</td>
<td>CA1’s Signature</td>
<td>CA2 is a CA</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bob</th>
<th>CA1</th>
<th>CA2</th>
<th>Alice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issuer DN: CA1</td>
<td>Subject DN: CA2</td>
<td>CA2’s pub. key</td>
<td></td>
</tr>
<tr>
<td>CA1’s Signature</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To enforce naming hierarchy, use Naming Constraints Extension (to restrict `lower` CA)

Hierarchical

Mesh:
All CA’s are equal

X.509 Extensions Mechanism

- Used for certificates and Certificate Revocation Lists (CRL – more later)

- Each extension contains:
  - Extension identifier
    - As an OID (Object Identifier – see before)
    - E.g. `Naming constraints`
  - Extension value
    - E.g. `Include C=IL`, `exclude dNSName=*.IBM.COM`
  - Criticality indicator
    - If critical, relying parties MUST understand extension to use certificate
      - E.g. Naming constraints is always marked `critical`
    - If non-critical, Ok to use certificate anyway
X.509v3 & PKIX Standard Extensions

- Authority key identifier (helps esp. if CA has multiple keys!)
- Subject key identifier (helps identify chain of certificates)
- Certification path constraints
  - Basic constraint: subject is CA or end entity, path length
  - Policy and naming constraints (on certs issued by subject)
- Key identifier and usage (signing, encryption, etc.)
- Subject and issuer alternative names (e.g. e-mail)
  - E.g.: subject dNSName in subjectAltName extension
- Certificate policy identifier and qualifiers
  - What is the policy of the CA (and disclaimers)
- Policy mappings
  - How to interpret attributes in certificates issued by subject (if CA)
- Certificate Revocation List (CRL) extensions
  - More on revocation later…
- Other (non-standard) extensions for other attributes
  - E.g. role, clearance, dept, rank, degree, quality, evaluation,…
  - Or: use separate `attribute certificate` (later)
Certificate Validation $\text{Valid}_{CA,v}(c, \{a, p\})$

- Cf. to CA name
- Cf. date/time
- Validate if known
- Cf. to CA ID
- Cf. to subject ID
- Acceptable?
- Valid?

<table>
<thead>
<tr>
<th>Issuer Distinguished Name (DN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validity period</td>
</tr>
<tr>
<td>Subject (user) Distinguished Name (DN)</td>
</tr>
<tr>
<td>Subject public key</td>
</tr>
<tr>
<td>Issuer unique identifier</td>
</tr>
<tr>
<td>Subject unique identifier</td>
</tr>
</tbody>
</table>

- Key Usage extension(s)
- Name Constraints extension
- Basic constraint: Cert_len [for CA>0]
- Policy (ID) Constraints Mappings
- SubjectAltName ext.
- E-mail DNS URI

Signature on the above fields

12/18/2005
Realities of CA Hierarchies

- Used mostly within an organization
  - E.g. with Lotus Notes
  - Or: an external CA certifies the `root CA` of an organization

- CA interoperability via hierarchy is difficult
  - Motivation, liability, different policies
  - Effort: US Federal Bridge CA

- Relying parties often simply trust each CA; example – list of CA’s in browsers

- Or: Cross-certification (upon need)
X.509 Cross Certification

• NTT JP allows IBM JP to certify IBMers
• IBM JP allows IBM to certify all IBMers, except of IBM JP
• How? Using Naming Constraint Extension (always critical)…
• NTT JP allows IBM JP to certify IBMers
• IBM JP allows IBM to certify all IBMers, except of IBM JP
• How? Using Naming Constraint Extension (always critical)…
Certificate Path Validation

- By relying party (or a trusted path validation service)
- Local validation of each certificate
  - Validity periods, key usage, revocations.
- Verify chain of distinguished names and identifiers
- Verify each certificate:
  - Signed by previous public key
  - Certificate path below all `basic constraint` length limits
  - Verify name constraints (permissible name space)
  - Perform any policy mapping and verify policy constraints
Certificate Path Discovery

- Offline problem: given set of (locally valid) public key certificates, is there a valid certificate path to Alice’s certificate?
- Online problem: same, but collect more certificates as needed.
Offline Certificate Path Discovery

- Given set of PKCs, is there a valid certificate path to Alice’s certificate?
- A public key certificate (PKC) is a 4-tuple: \(<\text{IssuerPK}, \text{SubjectPK}, \text{Attrs}, \text{Sign}>\)
- Simplify:
  - All PKCs locally valid
  - No path length constraints
  - Name, policy constraints – none/trivial/ignored
  - Only remaining relevant attributes are:
    - DN = Distinguished Name
    - CA = Y if this is a CA, N if not a CA
- Defines a graph…
Certificate Path Graph

- Vertices $V$: <pub_key, DN, CA_flag>
  - $CA\_flag$=CA for a CA, $N$ – just end-entity
- Edges $E$: connect from <p,n,CA> to <p’,n’,f> if there is a certificate: Signed by $p$, issuer DN = $n$, subject DN = $n’$, subject PK = $p’$, CA flag = $f$
- Example: Bob initial graph contains only the public key and Distinguished Name of $CA_B$, the CA Bob trusts:

\[
P_{CA_b, CA_B, CA}
\]
Certificate Path Graph

- Vertices \( V: \text{<pub_key, DN, CA\_flag>} \)
  - \( \text{CA\_flag}=1 \) for a CA, 0 – just end-entity
- Edges \( E: \) connect from \( \text{<p,n,CA>} \) to \( \text{<p',n',f>} \) if there is a certificate: Signed by \( p \), issuer DN = \( n \), subject DN = \( n' \), subject PK = \( p' \), CA flag = \( f \)
- Example: After Bob receives also cross-certificate signed by his CA for Alice’s CA, with properties: \( \{\text{DN}=CA_A, \text{CA}=Y\}\):
Certificate Path Graph

- Vertices $V$: $<\text{pub_key}, \text{DN}, \text{CA\_flag}>$
  - $\text{CA\_flag}=1$ for a CA, 0 – just end-entity
- Edges $E$: connect from $<p, n, 1>$ to $<p', n', f>$ if there is a certificate: Signed by $p$, issuer DN = $n$, subject DN = $n'$, subject PK = $p'$, CA flag = $f$
- Example: After Bob receives also the certificate from Alice’s CA to Alice, with properties $\{\text{DN= Alice, CA=N}\}$:

\[\text{CA}_{Bpub} , \text{CA}_B, \text{CA} \rightarrow \text{CA}_{Apub} , \text{CA}_A, \text{CA} \rightarrow \text{Alice}_{pub} , \text{Alice}, \text{N}\]
Certificate Path Graph

- Vertices $V$: <$pub\_key, DN, CA\_flag$>
  - $CA\_flag=CA$ for a CA, $N$ – just end-entity
- Edges $E$: connect from <$p,n,CA$> to <$p',n',f$> if there is a certificate:
  - Signed by public key $p$
  - With issuer DN = $n$
  - With subject DN = $n'$
  - With subject PK = $p'$
  - With CA flag = $f$
- Question: let $V'\subseteq V$ be trusted CA’s. Is there a path from a vertex in $V'$ to <$p,n,f$>? Find shortest path!
- Answer: use BFS; work=$O(|E|)$. 
Observations

- Length constraint usually not used / ignored
  - Except for length>0 for CA which is critical…
    - Yet ignored at early versions of IE…
  - If used, need to be reflected in graph → more complex
  - Relying party should decide acceptable length
  - Select shortest path (BFS)

- This graph/policy is *monotonic* – more certificates only add validity

- `Internal` distinguished names and other identifiers are not significant!
Identity PKI Risks

- To relying parties
  - Fraudulent identity
  - Disputes (claims of fraudulent identity)
- To identified entity - identity theft
- To CA – liability
  - Potentially unbounded – unknown application ➔ high costs
  - Limit by stating policy (of issuing, use, liability)
  - In reality, often extreme disclaimers of liability
  - Even without disclaimers: how to prove negligence???
- Main threats:
  - Exposure of the CA private signing key
  - Issuing certificate for false identity
    - Esp. a problem with remote issuing
X.509 Privilege Management Infrastructure

- Idea: specific properties/privileges of subject (not ID) ➔ limited use and risk ➔ easier, cheaper to issue
  - Also: issued by other entity, e.g. school

- Complementary to X.509 PKI
  - Idea: present ID card and signed certificate (grades) identifying you by ID/name/cert/PK/...


- Attribute Certificate: properties of subject (holder)
- Attribute Authority: issuer of Attribute certificate
- SOA (Source Of Authority): root AA
## X.509 Attribute Certificates (v2, 2000)

<table>
<thead>
<tr>
<th>Signed fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
</tr>
<tr>
<td>Subject(Holder): serial, name, or hash</td>
</tr>
<tr>
<td>Issuer: serial, name, or hash</td>
</tr>
<tr>
<td>Signature OID</td>
</tr>
<tr>
<td>Serial number</td>
</tr>
<tr>
<td>Validity period</td>
</tr>
<tr>
<td>Attributes</td>
</tr>
<tr>
<td>Issuer unique ID</td>
</tr>
<tr>
<td>Extensions (from version 3)</td>
</tr>
<tr>
<td>Signature on the above fields</td>
</tr>
</tbody>
</table>

Signed fields:

- **Version**
- **Subject(Holder): serial, name, or hash**
- **Issuer: serial, name, or hash**
- **Signature OID**
- **Serial number**
- **Validity period**
- **Attributes**
- **Issuer unique ID**
- **Extensions (from version 3)**
- **Signature on the above fields**
Difference between PKC and AC

PKC is passport and AC is visa

Public Key Certificate (PKC)
- Version
- Serial Number
- Signature ID
- Subject
- Issuer
- Validity Period
- Subject Public Key Info
- Extensions
- Signature

Attribute Certificate (AC)
- Version
- Serial Number
- Signature ID
- Holder
- Issuer
- Validity Period
- Attributes
- Extensions
- Signature

Public Key PKC binds a subject and a public key

No Public Key AC binds a holder and attributes
Binding PKC and AC

Public Key Certificate (PKC)
- Version
- Serial Number
- Signature ID
- Subject
- Issuer
- Validity Period
- Subject Public Key Info
- Extensions
- Signature

Attribute Certificate (AC)
- Version
- Serial Number
- Signature ID
- Holder
- Issuer
- Validity Period
- Attributes
- Extensions
- Signature
Binding AC to Object by Hash

Object
(program/applet, document/picture, public key, PKC,...)
Access Control with AC: Privilege as Attribute

Signed Request Or Proposal

PKC+AC

Validate Signature

Request ∈ Privileges ?

Issuer of AC must know privilege → inflexible, centralized…
Better: use access control…

Confirm PKC, AC (signatures, issuers)

PK

Request, signature
Role-based Access Control with AC
Case I: Role as Attribute

Signed Request Or Proposal

Confirm Role, PK

PKC+AC

PK

Request, signature

Validate Signature

Access control / Business logic

Y/N
Role-based Access Control with AC
Case II: Role derived from Attributes

Signed Request Or Proposal

Confirm and Extract Attrs and PK

Trust Policy (Rules)

Trust Manager: \{attrs\} → Role

Validate Signature

Access control / Business logic

Y/N

Request, signature

PKC+AC

PK

Attrs

Role

http://Amir.Herzberg.name

12/18/2005
Trust Policy and Rules

- How to define, evaluate trust policy?
- Many works, no clear answer, open issues
- We’ll just show some examples

- Trusted Banking Authority
  - AA or better credit rating
  - Good Bank

- Trusted Consumer Organization
  - Good Service
Dynamic Authorities

- Trusted Banking Authority
- Trustworthy Banking Authority
- Trusted Consumer Organization
- Good Bank
- AA or better credit rating
- Good Service
- Trustworthy Banking Authority
Negative Attributes

- Negative review is often very important
- Problem: will not be provided by subject…
- Possible conflicts

- Trustworthy Banking Authority
- Trusted Banking Authority
- Trusted Consumer Organization
- AA or better credit rating
- Good Bank
- Exclude: Bad Service
Challenges of reputation based policies

- False positive reports ("ballot stuffing")
  - Self-voting, "Sybil attack", Amazon’s glitch
  - Votes Inc.
  - Collusions

- False negative reports ("bad mouthing")
  - ... Amazon glitch again...
  - Spy vs. Spy: who’s the villain?

- Some solutions...
  - Controlled membership (barriers to entrance: ID, $, CPU)
  - Proofs of wrong-doings (validate randomly, aggregate)
    - $Sign_{A,s}(email-message, “not spam”)$
    - $Sign_{compute,s}(“input=“, x,”output=“, f(x))$
  - Evidence of actual money transfer (and payment of fees...)
    - By Payment Service Provider (PSP), marketplace
Multiple-party certificate example

My Friends

Consumer

Marketplace (e.g. eBay, Visa)

Real Consumer

Real Transaction

Good Transaction

5

Good Vendor

Exclude: Bad Service

Include: Good Merchant

Good Merchant
PKC + AC + Collect + Extract + ...
Privacy of Credentials

- Another motivation for attr certs:
  - Not all relying parties need all attributes
  - Don’t send! ⇒ improve privacy & shorter cert
- Solution 1: send only necessary attr cert
- Solution 2: cert contains only hash of attr
  - $Cert_A = \text{Sign}_{CA.s}(A.s, "Alice", h(A.grades), h(A.jpg))$
  - Use: Alice sends $Cert_A, A\text{.grades}$ to apply to job
  - And: Alice sends $Cert_A, A\text{.jpg}$ to chat or enter door
  - Can we use this method to hide gender?
    - With $\text{Sign}_{CA.s}(A.s, "Alice", h(A.gender), h(A.jpg))$
Privacy for short info: Commitment

- We can hide even one bit, using `salt`…
  - Namely: \( \text{Sign}_{CA_s}(A.s, ”Alice”, h(A.gender,salt)) \)
  - To prove gender: send \( \text{Cert}_A, A.gender, salt \)
  - If hash is a random oracle…
  - Or if hash is `exposure resilient` function (we do not define)

- This is special case of commitment scheme
  - Functions: \( \text{commit}(m), \text{decommit}(m), \text{valid}(m,c,d) \)
  - Such that: \( \text{valid}(m,\text{commit}(m),\text{decommit}(m))=\text{True} \)
  - And two (`competing`) security properties:
    - Privacy: \( \text{commit}(m) \) reveals no info on
    - Can’t find \( c,d,d’,m,m’ \) s.t. \( \text{valid}(c,d,m), \text{valid}(c,d’,m’), m\neq m’ \)
  - Often commitment is defined with public key
Using encryption for commitment

- Solution 3: cert contains encryption of attr
  - $Cert_A = \text{Sign}_{CA, A.s}(A.s, A.e, "Alice", E_e(A.grades, r), E_e(A.jpg, r'))$
  - Where $r, r'$ are the random bits used to encrypt
  - Alice sends $Cert_A, A.grades, r$ to expose grades
  - Alice sends $Cert_A, A.jpg, r'$ to expose picture

- Secure if decryption process is deterministic
  - Each ciphertext has at most one decryption
  - Property of all proposed cryptosystems
    - Therefore a safe assumption

- A very useful trick...
  - Such `commitment` property may also hold for symmetric encryption, but not always... so use only PKCs.
Advanced Privacy of Credentials

- We’ve seen how to selectively display credentials…
- Q: is this enough for privacy?
- Not necessarily…
  - In identify certs: identity still exposed
  - The mere existence of extension type may reveal
  - Reuse of same cert, PK identifies: same person
  - I’ll show my rank only to an authorized officer
    - But officer has same policy… how will show first?
- Solutions exist… but not today
Certificate Revocation

- Reasons for revoking certificate
  - Key compromise
  - CA compromise
  - Affiliation changed (changing DN or other attribute)
  - Superseded (replaced)
  - Cessation – not longer needed

- How to inform relying parties?
  - Do not inform – wait for end of (short?) validity period
  - Distribute *Certificate Revocation List (CRL)*
  - Ask - Online Certificate Status Protocol (OCSP)
### X.509 CRL Format

<table>
<thead>
<tr>
<th>Signed fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version of CRL format</td>
</tr>
<tr>
<td>Signature Algorithm Object Identifier (OID)</td>
</tr>
<tr>
<td>CRL Issuer Distinguished Name (DN)</td>
</tr>
<tr>
<td>This update (date/time)</td>
</tr>
<tr>
<td>Next update (date/time) - optional</td>
</tr>
<tr>
<td>Subject (user) Distinguished Name (DN)</td>
</tr>
</tbody>
</table>

#### CRL Entry

<table>
<thead>
<tr>
<th>Certificate Serial Number</th>
<th>Revocation Date</th>
<th>CRL entry extensions</th>
</tr>
</thead>
</table>

#### CRL Extensions

Signature on the above fields
Revocation is Difficult

- If CRLs contain all revoked certificates (which did not expire)… it may be huge!
- CRLs are (also) not immediate
  - Who is responsible until CRL is distributed?
  - What is the impact on non-repudiation?

Solutions:
- Online Certificate Status Protocol (OCSP)
- More efficient CRL schemes (usually CRL extensions)
  - CRL distribution point – split certificates to several CRLs
  - Authorities Revocation List (ARL): list only revoked CAs
  - Delta CRL – only new revocations since last `base CRL`
  - Certificate Revocation Tree (more later)
- Short validity for certificates – no CRLs
Preventing Invalidation of Signatures

Problem: public key revoked… are signatures invalid??
- If not… revoking does not prevent signing with stolen key
- If yes… signer can revoke key (claim exposed) to deny signing
- Fair solutions: signatures validated before revocation remain valid

Solution 1: time-stamping of validation and revocation
- Signature is valid if time of validation < time of revocation
- Third-party evidence (timestamp) for time of validation, revocation
Goal: non-repudiated proof of document creation date
- Proof document/signature/revocation existed at/before date
- If signature on contract existed (was validated) before public key was revoked, then contract remains valid!

Timestamp signed by Time-stamping Authority

Hash document to protect confidentiality

Author

$h(doc)\ (paper,\ invention)$

Relying Party

$h(doc,\ sign),\ certificate(PK)$

Key Owner

Revocation(PK)

Time-Stamping Authority (TSA)
Preventing Invalidation of Signatures

- Problem: public key revoked… are signatures invalid??
  - If not… revoking does not prevent signing with stolen key
  - If yes… signer can revoke key (claim exposed) to deny signing
  - Fair solutions: signatures validated before revocation remain valid

- Solution 1: time-stamping of signature and revocation
  - Signature is valid if time of signature < time of revocation
  - Third-party evidence (timestamp) for time of signature, revocation

- Solution 2: limited, short validity periods for keys
  - Divide time into *periods*, e.g. day / month
  - Different keys for each period *i*: \( \text{Priv}[i], \text{Pub}[i] \)
  - Exposure of \( \text{Priv}[i] \) does not enable signing with \( \text{Priv}[i'], i' < i \)
  - \( \Rightarrow \) Even period *i*’s key is revoked, previous signatures are Ok
Short-Term Certificates

- Idea: short validity period of certificates, so no need to revoke them
- Concern: overhead of signing many certificates each (short) period
- Solutions:
  - Extend many certs with one signature: hash tree
    - \( \text{Sign}_{CA,s}(\text{date, valid: } h(h(\text{cert}_A), h(\text{cert}_B), ...)) \)
  - Certificate revocation tree:
    - \( \text{Sign}_{CA,s}(\text{date, all except: } h(h(\text{cert}_A), h(\text{cert}_B), ...)) \)
  - Certificates includes a hash chain, e.g. for Jan 2005:
    - \( \text{Cert}_A = \text{Sign}_{CA,s}(A.s, ”Alice”, 2005, h^{(11)}(x)) \)
    - And for Feb 2005: \( \text{Cert}_A, h^{(10)}(x) \)
    - Validate incoming \( \text{Cert}_A, h_{10} \) by \( h^{(11)}(x) = h(h_{10}) \)
    - Security based on random choice of x and h being one-way
Conclusion

- Public Key Certificates link between a public key and (attributes of) its owner
- X.509 focus is on Identity PKC
- Identity PKC are natural – we are used to ID cards
- But even X.509 added non-identity attributes:
  - In extensions of the X.509 PKC
  - In attribute certificates