More Public Key Infrastructures

Public Key Infrastructure

- Certificate Revocation
- Trust Models
- Ten Risks of PKI

A Network of Known Hosts
Local Public Key Directory

<table>
<thead>
<tr>
<th>Owner</th>
<th>Alice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>PK_b</td>
</tr>
<tr>
<td>Charlie</td>
<td>PK_c</td>
</tr>
<tr>
<td>Diane</td>
<td>PK_d</td>
</tr>
<tr>
<td>....</td>
<td></td>
</tr>
<tr>
<td>Irene</td>
<td>PK_i</td>
</tr>
<tr>
<td>....</td>
<td></td>
</tr>
<tr>
<td>Mallory</td>
<td>PK_m</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

A Simple But Strong Certificate

<table>
<thead>
<tr>
<th>Alice</th>
<th>[Alice,PK_a]_{PK_c^{-1}}</th>
<th>Charlie</th>
</tr>
</thead>
</table>

In order to extract Alice's verified public key, the certificate holder must know Charlie's public key.

A Weaker, Simple Certificate

<table>
<thead>
<tr>
<th>Alice</th>
<th>PK_a</th>
<th>[Alice,PK_a]_{PK_c^{-1}}</th>
<th>Charlie</th>
</tr>
</thead>
</table>

In order to verify Alice's public key, the certificate holder must know Charlie's public key.
What CA To Trust?
There are two questions you must ask about a CA when trying to validate a certificate:
1. Do I have the public key of the signing CA
2. Do I trust the signing CA

What CA To Trust?
There are two questions you must ask about a principle when trying to decide if you trust them:
1. Am I sure that I have their public key
2. Can I be sure they will not lie

Trust Defined
• "Entity A trusts entity B when assumes that B will behave exactly as A expects"

• Who can I trust to effectively bind a public key to an entity?
Local Public Key Directory
With Trust Relationships

<table>
<thead>
<tr>
<th>Owner: Alice</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>PK₀</td>
<td>No</td>
</tr>
<tr>
<td>Charlie</td>
<td>PK₁</td>
<td>Yes</td>
</tr>
<tr>
<td>Diane</td>
<td>PK₂</td>
<td>Yes</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Irene</td>
<td>PK₃</td>
<td>Yes</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Mallory</td>
<td>PK₄</td>
<td>No</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Certificate Chain

- Only works for a Transitive Trust Model
- Principal desiring to verify a certificate uses a **chain of trusted signers** between a signer they trust and the certificate they desire to verify.
- How best to establish this chain is a matter of extensive study

CA Chaining Verification Example

- **Alice holds public-key for root CA**
- Bob's cert is signed by CA₂
- CA₂ is signed by CA₁
- CA₁ is signed by root

1. Find CA₁'s cert, decrypt with root pk
2. Find CA₂'s cert, decrypt with CA₁'s pk
3. Verify Bob's msg with Bob's pk
PKI Trust Models

- Hierarchy of CAs
- Distributed Trust Architecture
- Web Model
- User-centric Trust
A Certificate Chain

P1 sends signed message with their certificate to P6.
P1 -> P6: (msg, [msg]_{PK1}, (1, [1,PK1]_{PKd}, D))

To verify the msg from P1, P6 must
(1) Acquire and verify AA's public key using root's public key
(2) Acquire and verify A's public key using AA
(3) Acquire and verify D's public key using A

If P1 sends these certificates, P6's job is much easier

A Certificate Chain

The message with a certificate chain appended looks like this:
P1 -> P6: (msg, [msg]_{PK1}, cert(1, D), cert(D, A), cert(A, AA), cert(AA, root))

Distributed Hierarchy

[Diagram of a distributed hierarchy with intermediate CAs and cross certifications]
Distributed Trust
Cross-Certification Options

- Hierarchy root nodes
- Full mesh
- Hub & Spoke

Web Trust Model

- CA Public Keys are pre-installed in Web browsers
- Users can modify these, but probably won't
- Revocation requires globally coordinated software update

Certificate Revocation

1. Adds significant complexity in infrastructure
   - Certificates are *fire and forget*
   - Lists must be maintained
2. Adds complexity in implementation
   - It adds a second step in certificate verification
3. Time
   - Revocation will likely NEVER be instantaneous
   - If it is not instantaneous, how fast is fast enough?
Certificate Revocation Options

1. Certificate Revocation Lists (CRLs)
2. Authority Revocation Lists
3. CRL Distribution Points (Partitioned CRLs)
4. Delta CRLs
5. Indirect CRLs
6. Enhanced CRL Distribution Points and Redirect CRLs
7. Certificate Revocation Trees (CRTs)

Certificate Revocation Lists

- Signed data structure
- Contains a list of revoked certificates
- CR sign is usually the same entity that signed the certificate that the corresponding CRL revokes
- CRL is a signed list, not a database; cannot be updated

Certificate Revocation Lists

- Fields
  - Version
  - Signature
  - Issuer
  - Date/time issued
  - Date/time expires
  - List of revoked certificates
  - Extensions
Certificate Revocation Lists

- Extensions
  - Reason
  - Certificate Issuer
  - Hold Instruction Code
  - Invalidity Date

Authority Revocation Lists

- CRL for CA certificate revocation
  Should be very rarely used

CRL Distribution Points

- CRL for CA domains
  - Subdivides info about certificates
    issued by one CA into multiple CRLs

Enhanced CRL Distribution Points

- Dynamic CRL partitioning
- Allows reassociation of certificate & CRL after the certificate is issued.
- Can be accomplished by having the originally associated CRL redirect inquiries
Delta CRLs
- Used to enhance timeliness without significantly impacting performance
- Allows incremental postings of CR information
- Each Delta version contains ALL changes to the base CRL, so when a new Delta is received, old Deltas can be discarded

Indirect CRLs
- Enables single CRL for multiple CAs
- No specification for how to accumulate CR info from CAs to the issuing CA

Certificate Revocation Trees
- Based on hash tree
- Dramatically reduces storage requirement
- Does not use CRL format

Other Revocation Option
- Short-lived certificate
- Problems with this?
## Risks of PKI

<table>
<thead>
<tr>
<th>Question</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Who do we trust and for what?</td>
<td>Trust isn't transitive</td>
</tr>
<tr>
<td>2. Who is using my key?</td>
<td>Protecting your private key is a non-trivial problem.</td>
</tr>
<tr>
<td>3. How secure is the verifying computer?</td>
<td>A hacker may add a key to your trusted CA list.</td>
</tr>
<tr>
<td>4. Which John Robinson is it?</td>
<td>Name confusion can happen.</td>
</tr>
<tr>
<td>5. Is the CA an authority?</td>
<td>Particularly dangerous when certs contain access info.</td>
</tr>
<tr>
<td>6. Is the user part of the security design?</td>
<td>Users should be able to manually audit the cert process</td>
</tr>
</tbody>
</table>

## Risks of PKI

<table>
<thead>
<tr>
<th>Question</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>6. Was it one CA or a CA plus a Registration Authority?</td>
<td>Again, relates to certs with access control embedded.</td>
</tr>
<tr>
<td>7. How did the CA identify the certificate holder?</td>
<td>Through the snail mail doesn't get it. Email is worse.</td>
</tr>
<tr>
<td>8. How secure are the certificate practices?</td>
<td>PKI has lots of rules. Vendors get sloppy. Recipe for disaster.</td>
</tr>
<tr>
<td>9. Why are we using the CA process anyway?</td>
<td>PKI can not solve many authentication problems, e.g. Single Sign on</td>
</tr>
</tbody>
</table>

## PKI Utilization

- Must have PKI-enabled applications
- What applications need PKI?
  - Messaging (e.g. Email)
  - E-Commerce
  - Network infrastructure
- Clients request public keys from the service
- Clients software is frequently proprietary
PKI Challenges

1. Authentication: It's hard to prove identity
2. Heterogeneity
3. Distributed architecture
   - Requires trust
   - Vulnerable to attack
4. Expense
5. Think of PKI as a massive database management system

Review

- Certificate Revocation
- Trust Models
- Ten Risks of PKI