Revocation Methods

**Explicit:**

- **CRL - Certificate Revocation List**
  - Sources: CRL-DP, indirect CRL, dynamic CRL-DP
  - Delta-CRL, windowed CRL, etc.
  - Certificate Revocation Tree (CRT) and other Authenticated Data Structures
- **OCSP – On-line Certificate Status Protocol**

**Implicit:**

- **CRS - Certificate Revocation System**
Open Questions

• Consistency between CRL and OCSP responses
  • It is possible to have a certificate with two different statuses.

• If OCSP is more timely and provides the same information as CRLs, do we still need CRLs?

• Which method should come first - OCSP or to CRL?
Revocation Methods

Explicit:

• CRL - Certificate Revocation List
  • Sources: CRL-DP, indirect CRL, dynamic CRL-DP
  • Delta-CRL, windowed CRL, etc.
  • Certificate Revocation Tree (CRT) and other Authenticated Data Structures

• OCSP – On-line Certificate Status Protocol

Implicit:

• CRS - Certificate Revocation System
Implicit Revocation: Certificate Revocation System (CRS)

• Proposed by Micali (1996)
• Aims to improve CRL communication costs
• Basic idea: CA periodically refreshes valid certificates
• Uses off-line/on-line signature scheme to reduce update cost
One-Way Hash Chains

• Versatile cryptographic primitive

• Construction:
  1. Pick random number $Y_N$ and a public hash function $H()$
  2. Compute $N$ values $Y_{N-1},...,Y_0$ such that $Y_{i-1} = H(Y_i)$
  3. Secret $\text{ROOT}=Y_N$, public $\text{ANCHOR}=Y_0$

• Properties:
  • Use in reverse order of construction: $Y_0, Y_1, ..., Y_N$
  • Hard to compute $Y_i$ from $Y_j$ (if $j<i$), easy to compute $Y_j$ from $Y_i$
  • For example: easy to compute $Y_1$ from $Y_2$ since $Y_1=H(Y_2)$
  • But, Infeasible to compute $Y_2$ from $Y_1$
  • Verifier can efficiently authenticate $Y_j$ knowing $Y_i$ ($j<i$) by verifying whether: $Y_j = H^{i-j}(Y_i) = H(H(...H(Y_i)...))$
  • This method is robust to missing values
CRS: Creation of a Certificate

- Two new parameters included in each PKC: $Y_0$ and $N_0$

\[
Y_0 = H^{\text{MAX}}(Y_{\text{MAX}})
\]

\[
N_0 = H(N_1)
\]

- $[Y_{\text{MAX}}, N_1]$ -- per-PKC secrets stored by CA

- $H()$ -- public one-way function, e.g., SHA-2
CRS Example:
Certificate issued for a year, refreshed daily

- If Alice's certificate is valid:
  • \( \text{UPD}_i = Y_i \) and
  • \( Y_o = H^i(Y_i) \) \( \leftarrow \) verifier can easily check this
  • Also, note that: \( Y_i = H^{MAX-i}(Y_{MAX}) \)

- If her certificate is revoked, \( \text{UPD}_i = N_1 \)

- \( Y_0 \) and \( N_0 \) are distinct for each certificate

NOTE: \( i=0 \) at issuance date
Lecture 13

Access Control

[lecture slides are adapted from previous slides by Prof. Gene Tsudik]
Recall: Security Services

- **Confidentiality**: to assure information privacy and secrecy
- **Authentication**: to assert who created or sent data
- **Integrity**: to show that data has not been altered
- **Access Control**: prevent misuse of resources = control access to them
e.g., files, directories, accounts, printers, computers, IoT devices, etc.
- **Availability**: to offer access to resources, permanence, non-erasure
Access Control (AC)

• A “language” for expressing access control policies: who can access what, how and when ...

• Enforcement of access control
  • Identify all resources (objects) and their granularity
  • Identify all potential users (subjects)
  • Specify rules for subject/object interaction
  • Guard them in real time
Model and Terminology

- **Subjects**: users or processes

- **Objects**: resources (files, memory, printers, routers, plotters, disks, processes, etc., etc., ...)
Focus of Access Control

• What a subject is allowed to do

• What may be done with an object
Access Modes

- **“Look” at an object, e.g.:**
  - Read file
  - Check printer queue
  - Read screen
  - Query database
  - Turn on/use microphone, etc., etc.

- **“Change” an object, e.g.:**
  - Write/append/erase file
  - Print on a printer
  - Display on screen
  - Use speakers (audio out)
  - Send packets via WiFi/Bluetooth, etc., etc.
Access Modes: Bell-Lapadula model

execute, read, append, and write

<table>
<thead>
<tr>
<th></th>
<th>Execute</th>
<th>Read</th>
<th>Append</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observe</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Alter</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
UNIX/Linux/*x Operating Systems

- **execute**: execute (program) file, search directory

- **read**: read from file, list directory

- **write**: write (re-write or append) file, create or rename file in directory
Example: Windows NT/2000 (NTFS)

- execute
- read
- write
- delete
- change permission
- change ownership
AC Types

Who is in charge of setting AC policy?

• **Discretionary**: resource owner

• **Mandatory**: system-wide policy
Access Control Structures

i. Access Control Matrix

ii. Capabilities

iii. Access Control Lists
## Access Control Matrix

<table>
<thead>
<tr>
<th>Subject</th>
<th>Bill.doc</th>
<th>Edit.exe</th>
<th>Fun.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>{0}</td>
<td>{execute}</td>
<td>{execute,read}</td>
</tr>
<tr>
<td>Bob</td>
<td>{read,write}</td>
<td>{execute}</td>
<td>{execute,read,write}</td>
</tr>
</tbody>
</table>
Access Control Lists 1/2

Keep access rights to an object with that object:

- **ACL for bill.doc:**
  - Bob: read, write

- **ACL for edit.exe:**
  - Alice: execute;
  - Bob: execute

- **ACL for fun.com:**
  - Alice: execute, read;
  - Bill: execute, read, write

- As many ACLs as there are objects
- Each ACL either signed or stored in protected place
Access Control Lists 2/2

• Managing access rights can be difficult

• Groups can be helpful ...

• Groups simplify definition of access control policies
Access Control Lists

S1 → S2 → S3

O1 → O2 → O3 → O4

G1

X

O5
Capabilities 1/2

- Capabilities are associated with discretionary access control
  - Reason: difficult to get full view of who has permission to access an object
  - Very difficult to revoke a capability – owners and objects have to keep track of all issued capabilities

- As many capabilities as there are (subject/object) pairs
- Each capability either signed or otherwise protected
- Hard to revoke in a distributed setting
Capabilities 2/2

Keep access rights with the subject:

• Alice's capabilities:
  • [edit.exe:execute];
  • [fun.com:execute,read]

• Bob's capabilities:
  • [bill.doc:read,write]
  • [edit.exe:execute]
  • [fun.com:execute,read,write]
In Summary

- **Centralized Systems:**
  - ACLs are better

- **Distributed Systems:**
  - Capabilities are better
Example: Android Security/Permissions

Facebook needs access to additional permissions (marked as NEW):

Your messages
NEW: Read your text messages (SMS or MMS)

Your personal information
NEW: Add or modify calendar events and send emails to guests without host's knowledge, read calendar events plus confidential information, read your own contact card

Network communication
NEW: Connect and disconnect from Wi-Fi
Full network access

Default
Com.sec.android.provider.badge.permission.READ

[Buttons: SKIP ACCEPT]
Android Security Model

• Application-level permissions model
  • Controls access to app components
  • Controls access to system resources
  • Specified by the app writers and seen by the users

• Kernel-level sandboxing and isolation
  • Isolate apps from each other and the system
  • Prevent bypass of application-level controls
  • Relies on Linux Discretionary Access Control (DAC)
  • Normally invisible to the users and app writers
Discretionary Access Control (DAC)

• Typical form of access control in Linux and many other Unix-derived OS-s
• Access to data is entirely at the discretion of the owner/creator of the data
• Some processes (e.g., uid 0) can override and some objects (e.g., sockets) are unchecked
• Based on user & group identity
ROLE BASED ACCESS CONTROL (RBAC)
RBAC Basics

• Users are associated with roles

• Roles are associated with permissions

• A user has permission only if s/he has a role associated with that permission
Example: Cops (aka Police Officers) (User/Permission Association)
Example: RBAC
Example: RBAC
Example: RBAC

Here RBAC doesn’t work ...

38
Example: Alice becomes a Cop

Alice

Cop

- station
- uniform
- weapons

Alice

- station
- uniform
- weapons