National update: Simple Signing Device

Presentation of SSD on TF-EMC2
on 22.9.2010 Copenhagen
TOC

• Background
• Reasons
• Solution
Us – Haka Group

- Manne Miettinen
- Mikael Linden
- Arto Tuomi
- Timo Mustonen
- Janne Lauros

- Mika Suvanto has left the building
Haka metadata process

- Changes to metadata are usually new idp/sp, certificates, contact information
- Metadata change request usually 3-5 times a week
- Metadata release is triggered usually by ValidUntil, agreed idp/dp deployment, Certificate change
- Usually 1-.2 times a week
- Not something you would call dynamic
Haka metadata process

- View we use 'daily' for making updates to next metadata to be released
- View we use for 'weekly' metadata releases

Following resources/homeOrgs may have changed since last Haka metadata update:

- [Finnish Online University of Applied Sciences Portal](#)

Published metadata file last updated: 6.9.2010 09:37:45

Following resources/homeOrgs may have changed since last Haka/Kalmar metadata update:

- [Finnish Online University of Applied Sciences Portal](#)

Published metadata file last updated: 6.9.2010 09:37:45
Haka metadata process

- What I as operator feel is good about it
  - The release process is very static
    - No moving parts. It is very hard to break the system. Basically, what the client sees, is a file sitting on address [http://haka.funet.fi/fed/haka-metadata.xml](http://haka.funet.fi/fed/haka-metadata.xml) and that is what it actually is too, no automation.

- What I as operator feel is bad about it
  - The release process is very static
    - ValidUntil.. Having low ValidUntil value just does not suite the model. Of course fixing this does not cause major overhaul to system but does require some modifications.
How did we handle the key then??

Currently we are storing the private key in a java keystore protected by alias and password combination hardcoded to php code…

.. Which is okay if we trust our staff never to make any mistakes but..
In an Audit.. How do we

- Explain the origin of the key. Has it been generated by nn on a machine named zz. How safe was the process? Is it even unique?
- Who are the people that have had access to the key in plain text during it’s lifecycle?
- Where the key is actually located? Has it been sent in emails, copied to usb sticks?
- We cannot answer convincingly to any of those questions
SSD – Goals

• How to ensure that anyone (e.g. federation operator staff, laptop/server support staff, intruders) does not accidentally or intentionally compromise the private key?

• How to monitor the use of private key (i.e. who has signed what) without introducing tedious ceremonies?

• How to spot if the private key is compromised? Again, without tedious ceremonies?

• And how to do this with minimum effort (there aren’t that many of us)
SSD – Solution

- Facilities
- Processes
- Roles
- And the HSModule and supporting Sw
SSD – Roles

• **Basic User**
  - Is the metadata operator, has access to private key operation.

• **Master User**
  - Master User can grant Basic User rights to SSD.

• **Facilities Officer**
  - Is the only one allowed to enter HSM room and handle the SSD.

• **Key Officers 1 & 2**
  - Are responsible for storing the divided backup of private key. The only role that can be mixed with other roles.
SSD – Facilities

- Two rooms with restricted access
- Users having Master/Basic User role do not have clearance to HSM Room
- Only user having Facilities officer role is allowed to HSM Room
SSD – Facilities

- In operation room we have secure laptop. The laptop has networking disabled.
- In HSM room we SSD cards in secure cabinet.
- Laptop is connected to SSD cards with usb readers. Wiring goes through wall.
SSD – What does it consist of?

- Smart Card
- SSD Application on the smart card
- Applications running on the laptop
  - SSD Maintenance
  - SSD Metadatatool
SSD – Smart Card

Gemalto TOP GX4
FIPS140-2 Level 3 Certified
Java Card 2.2.1
Workable tools
Slow
Cheap (~15€/pcs)
SSD – The Option II

IBM 4764


FIPS140-2 Level 4 Certified

Fast

Expensive – approx. 8600$
# FIPS-2 Physical Security Requirements

<table>
<thead>
<tr>
<th></th>
<th>General Requirements for all Embodiments</th>
<th>Single-Chip Cryptographic Modules</th>
<th>Multiple-Chip Embedded Cryptographic Modules</th>
<th>Multiple-Chip Standalone Cryptographic Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Security Level 1</strong></td>
<td>Production-grade components (with standard passivation).</td>
<td>No additional requirements.</td>
<td>If applicable, production-grade enclosure or removable cover.</td>
<td>Production-grade enclosure.</td>
</tr>
<tr>
<td><strong>Security Level 2</strong></td>
<td>Evidence of tampering (e.g., cover, enclosure, or seal).</td>
<td>Opaque tamper-evident coating on chip or enclosure.</td>
<td>Opaque tamper-evident encapsulating material or enclosure with tamper-evident seals or pick-resistant locks for doors or removable covers.</td>
<td>Opaque enclosure with tamper-evident seals or pick-resistant locks for doors or removable covers.</td>
</tr>
<tr>
<td><strong>Security Level 3</strong></td>
<td>Automatic zeroization when accessing the maintenance access interface. Tamper response and zeroization circuitry. Protected vents.</td>
<td>Hard opaque tamper-evident coating on chip or strong removal-resistant and penetration resistant enclosure.</td>
<td>Hard opaque potting material encapsulation of multiple chip circuitry embodiment or applicable Multiple-Chip Standalone Security Level 3 requirements.</td>
<td>Hard opaque potting material encapsulation of multiple chip circuitry embodiment or strong enclosure with removal/penetration attempts causing serious damage.</td>
</tr>
</tbody>
</table>
SSD – Application

- Lifecycle consists of
  - Initializiation phase
  - Active phase
  - Inactive phase
SSD – Application

• Initialization phase functionality
  – Create Master Pin
  – Create Key Pair
  – Export Key Pair
  – Import Key Pair
  – Finalize Initialization
SSD – Application

- Active phase functionality
  - Add User
  - Remove User
  - Login User
  - Export Public Key
  - Prv Key Operation
SSD – Application

- Inactive (and Active) phase functionality
  - Get Serial Number
  - Get Status
  - Get Log
  - Get Sw Version
SSD – Application

- Simplified Lifecycle
SSD – Metadatatool

- Runs on secure laptop
- User can sign xml files
  - Requires User Id and Pin code
SSD – Maintenance tool

- Runs on a secure laptop
  - Initialize SSD
  - Manage users
  - View SSD internal log
  - Create CSR
SSD – Maintenance tool

- Internal Log

## SSD Log

<table>
<thead>
<tr>
<th>Entry Id</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>KEYPAIR_CREATED</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>USER_ADDED</td>
<td>User 0 added</td>
</tr>
<tr>
<td>2</td>
<td>USER_ADDED</td>
<td>User 1 added</td>
</tr>
<tr>
<td>3</td>
<td>SHA1PKCS1DEC_OPERATION</td>
<td>User 1 decrypted 4b5d6d682154868d9408d0d7ed3d4fd5b905131c</td>
</tr>
<tr>
<td>4</td>
<td>SHA1PKCS1DEC_OPERATION</td>
<td>User 1 decrypted fe48c5dea89e3d5a1d2a727c7627d0e727b063e</td>
</tr>
<tr>
<td>5</td>
<td>SHA1PKCS1DEC_OPERATION</td>
<td>User 1 decrypted fe48c5dea89e3d5a1d2a727c7627d0e727b063e</td>
</tr>
<tr>
<td>6</td>
<td>SHA1PKCS1DEC_OPERATION</td>
<td>User 0 decrypted fcee60fcad8e442e745979d2f78be8004d27bb95</td>
</tr>
<tr>
<td>7</td>
<td>SHA1PKCS1DEC_OPERATION</td>
<td>User 0 decrypted fcee60fcad8e442e745979d2f78be8004d27bb95</td>
</tr>
<tr>
<td>8</td>
<td>SHA1PKCS1DEC_OPERATION</td>
<td>User 0 decrypted fcee60fcad8e442e745979d2f78be8004d27bb95</td>
</tr>
<tr>
<td>9</td>
<td>SHA1PKCS1DEC_OPERATION</td>
<td>User 0 decrypted fcee60fcad8e442e745979d2f78be8004d27bb95</td>
</tr>
<tr>
<td>10</td>
<td>SHA1PKCS1DEC_OPERATION</td>
<td>User 0 decrypted fcee60fcad8e442e745979d2f78be8004d27bb95</td>
</tr>
<tr>
<td>11</td>
<td>SHA1PKCS1DEC_OPERATION</td>
<td>User 0 decrypted 4b5d6d682154868d9408d0d7ed3d4fd5b905131c</td>
</tr>
<tr>
<td>12</td>
<td>SHA1PKCS1DEC_OPERATION</td>
<td>User 0 decrypted fcee60fcad8e442e745979d2f78be8004d27bb95</td>
</tr>
<tr>
<td>13</td>
<td>SHA1PKCS1DEC_OPERATION</td>
<td>User 0 decrypted fcee60fcad8e442e745979d2f78be8004d27bb95</td>
</tr>
<tr>
<td>14</td>
<td>SHA1PKCS1DEC_OPERATION</td>
<td>User 0 decrypted fcee60fcad8e442e745979d2f78be8004d27bb95</td>
</tr>
<tr>
<td>15</td>
<td>SHA1PKCS1DEC_OPERATION</td>
<td>User 0 decrypted fcee60fcad8e442e745979d2f78be8004d27bb95</td>
</tr>
<tr>
<td>16</td>
<td>SHA1PKCS1DEC_OPERATION</td>
<td>User 0 decrypted fcee60fcad8e442e745979d2f78be8004d27bb95</td>
</tr>
<tr>
<td>17</td>
<td>SHA1PKCS1DEC_OPERATION</td>
<td>User 0 decrypted fcee60fcad8e442e745979d2f78be8004d27bb95</td>
</tr>
<tr>
<td>18</td>
<td>SHA1PKCS1DEC_OPERATION</td>
<td>User 0 decrypted fcee60fcad8e442e745979d2f78be8004d27bb95</td>
</tr>
</tbody>
</table>
SSD – Initializing SSD

- Facilities officer moves the SSD to cabinet in HSM Room..

- SSD Maintenance tool is launched in operation room to start process
SSD – Initializing SSD

- Next->next type of initialization
- Backup of the key can be created only during initialization process
- Backup is not mandatory
- Key officers get halves of the key encrypted by passphrases..
SSD – Signing with SSD

- Signing requires presence of only the Basic User.
- Every prv key operation is recorded to internal log of the SSD.
SSD – So we have a system such that

- No person has to be in contact with plain text private key
- It is very unlikely that hamfisted operator could compromise the key by mistake
- It is also unlikely that one disgruntled employee is able to take a copy of the key
- We are able to sign without tedious ceremonies
SSD – what else

- Offline is offline..
- Limited number of log entries..
- This is result of first round and I expect we have to modify concept a bit in the future (which actually is not bad but fun)
- We have now fairly safe way of creating and using private key in controlled fashion, but is metadata signing the right application for it ?-)