Voting using Java Card smart cards
(a case study)

Martijn Oostdijk
(joint work with C-B. Breunesse & B. Jacobs)
University of Nijmegen
1. Context
2. Toys & Setup
3. Voting
4. Implementation
5. Demo
6. Results & Conclusions
• Loop Group: Traditionally, interest in program correctness, semantics, logic
• Semantics of Java in PVS theorem prover implemented in Loop Tool
• VerifiCard: Application of Loop Tool to smart card application
• Case study driven research
• Problem: No experience with smart card applications
• Security is interesting
Toys

Schlumberger Palmera (10x)

Gemplus GemXPresso IS (2x)

Java iButton (+ adapters) (20x)

Gemplus GCR410 (2x)
Linux machine with Sun’s Java SDK and additional APIs, smart card terminal attached to serial port:
Electronic voting is an interesting case study because:

- Many aspects of security involved:
  - Confidentiality
  - Authentication
  - Integrity
  - Non-repudiation

- Distributed application over the Internet

- Untrusted clients: Smart cards as TCB
Voting 1

- Applet
- Host Application
- Internet
- Server

- Personal vote list
- Voter
- "Secure" feedback channel

- id
- privA
- vote thread
- GUI

- pubS
- virus

- pubA
- privS
**Personal vote list**

**Stemnummers voor id**

<table>
<thead>
<tr>
<th>Party</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>pvda</td>
<td>227</td>
</tr>
<tr>
<td>vvd</td>
<td>72</td>
</tr>
<tr>
<td>cda</td>
<td>242</td>
</tr>
<tr>
<td>d66</td>
<td>16</td>
</tr>
<tr>
<td>groenlinks</td>
<td>235</td>
</tr>
<tr>
<td>sp</td>
<td>96</td>
</tr>
<tr>
<td>christenunie</td>
<td>83</td>
</tr>
<tr>
<td>sgp</td>
<td>46</td>
</tr>
</tbody>
</table>
Voting 2

- Applet $A$ sends $id$ to host application $H$
- $H$ sends vote $v$ to $A$
- $A$ returns RSA/SHA signature $s$ of $v$ (using $\text{priv}_A$)
- $H$ generates a new session key $\text{des}$, encrypts it with $\text{pub}_S$, and sends it to $S$
- $H$ sends $(id, \ldots, v, s)$ encrypted with $\text{des}$ to $S$
- $S$ checks the signature
1. \( A \rightarrow H : \text{id} \)

2. \( H \rightarrow A : v \)

3. \( A \rightarrow H : \{\partial(v)\}_{\text{priv}_A} =: s \)

4. \( H \rightarrow S : \{\text{des}\}_{\text{pub}_S} \)

5. \( H \rightarrow S : \{\text{id}, \ldots, \langle v \rangle_s\}_{\text{des}} =: m \)

6. \( S \rightarrow H : \text{ack/deny} \)
Implementation

- Loading the applet onto the card: Visa OP
- Generating the keys $\text{pub}_A$ and $\text{priv}_A$
- Initialization: INS_SET_ID, INS_SET_PRIVATE_EXP, INS_SET_MODULUS
- Voting: INS_GET_ID, INS_SIGN
- Terminal uses threads to keep GUI responsive
Implementation: process method

... case INS_SET_MODULUS:
    if (modulus!=null)
        ISOException.throwIt(ISO7816.SW_CONDITIONS_NOT_SATISFIED);
    else {
        modulus = new byte[lc];
        readBuffer(apdu,modulus);
    }
    break;

case INS_SIGN:
    if (modulus==null || private_exp==null)
        ISOException.throwIt(ISO7816.SW_CONDITIONS_NOT_SATISFIED);
    else
        sign(apdu);
    break;

...
Implementation: sign method

MessageDigest digester;
Cipher encrypter;
...
byte[] buffer = apdu.getBuffer();
short lc = (short)(buffer[ISO7816.OFFSET_LC] & 0x00FF);
if (lc!=BLOCK_SIZE)
    ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
readBuffer(apdu,plaintext);
digester.doFinal(plaintext,(short)0,BLOCK_SIZE,hashtext,(short)0);
Util.arrayCopy(hashtext,(short)0,
paddedhashtext,(short)(BLOCK_SIZE-hashtext.length),
    (short)hashtext.length);
encrypter.doFinal(paddedhashtext,
    (short)0,BLOCK_SIZE,ciphertext,(short)0);
writeBuffer(ciphertext,apdu);
...
Problems/Results

- Crypto export restrictions: Sun’s JCE doesn’t come with RSA
- Differences JC 2.0 and 2.1: iButtons use crypto processor directly
- Threaded terminal: Correct? Not part of TCB
- Patent pending for personal vote lists
Conclusions

• Even though it’s Java, it’s still very low level

• Applet is small enough to be formally specified

• Security verification requires higher level reasoning?

• Future work: Visa OP, GSM, other case studies...