Voting using Java Card smart cards (a case study)

Martijn Oostdijk
(joint work with C-B. Breunesse & B. Jacobs)
University of Nijmegen

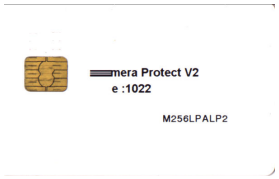
Outline

1. Context
2. Toys & Setup
3. Voting
4. Implementation
5. Demo
6. Results & Conclusions

Context

- 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)

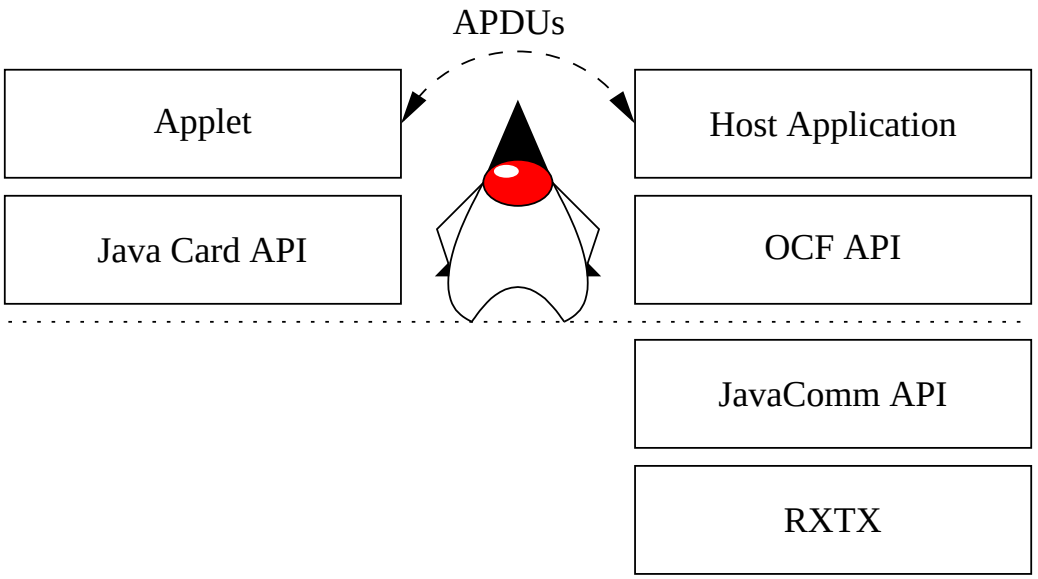
Gemplus GemXPresso IS (2x)

Java iButton (+ adapters) (20x)

Gemplus GCR410 (2x)

Setup

Linux machine with Sun's Java SDK and additional APIs, smart card terminal attached to serial port:

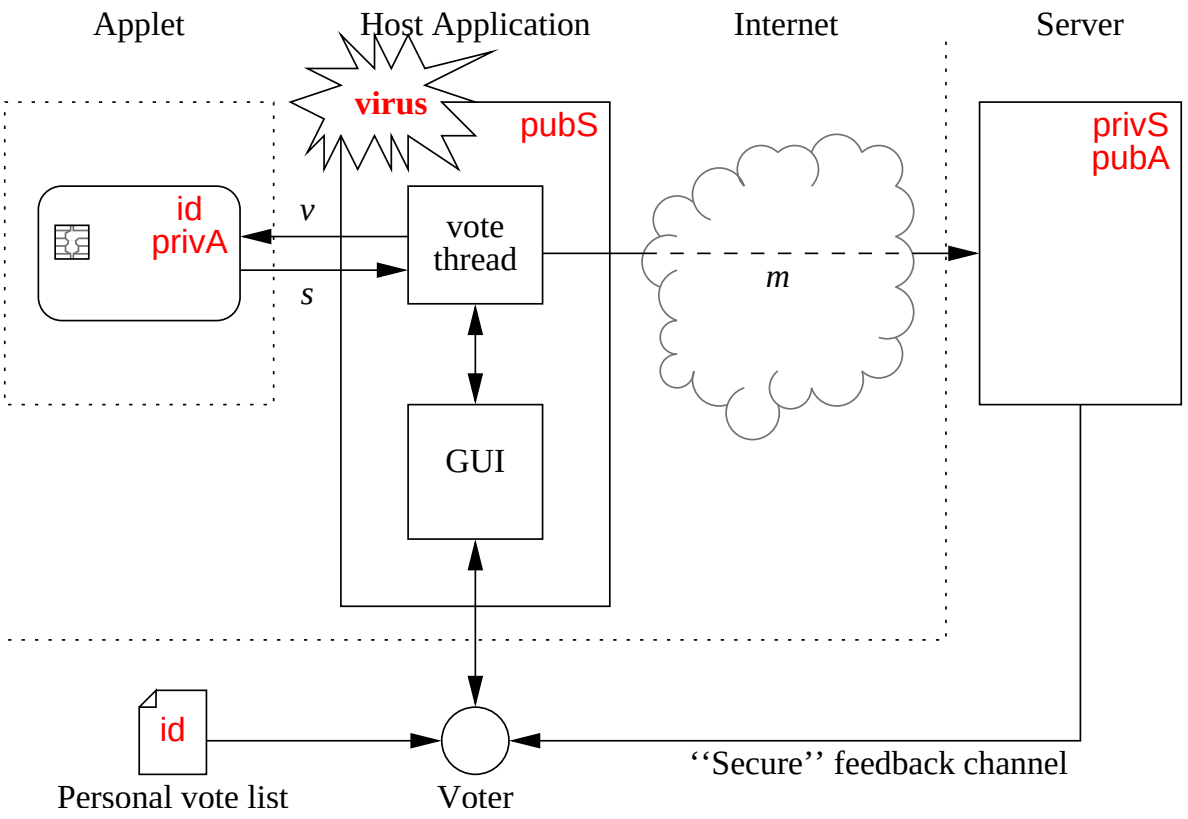


Why voting?

Electronic voting is an interesting case study because:

- Many aspects of security involved:
  - Confidentiality
  - Authentication
  - Integrity
  - Non-non-repudiation
- Distributed application over the Internet
- Untrusted clients: Smart cards as TCB

Voting 1


Applet
Host Application
Internet
Server
virus
pubS
privS
pubA
id
privA
v
s
vote
thread
m
GUI
id
Personal vote list
Voter
‘‘Secure’’ feedback channel

Personal vote list

Stemnummers voor id

pvda	227
vvd	72
cda	242
d66	16
groenlinks	235
sp	96
christenunie	83
sgp	46

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 priv_A)
- H generates a new session key des, encrypts it with pub_S, and sends it to S
- H sends $(\text{id}, \ldots, v, s)$ encrypted with des to S
- S checks the signature

Voting 3

1. $A \rightarrow H \;:\; \mathsf{id}$
2. $H \rightarrow A \;:\; v$
3. $A \rightarrow H \;:\; \{\partial(v)\}_{\mathsf{priv}_A} =: s$
4. $H \rightarrow S \;:\; \{\mathsf{des}\}_{\mathsf{pub}_S}$
5. $H \rightarrow S \;:\; \{\mathsf{id}, \ldots, \langle v \rangle_s\}_{\mathsf{des}} =: m$
6. $S \rightarrow H \;:\; \mathsf{ack}/\mathsf{deny}$

Implementation

- Loading the applet onto the card: Visa OP
- Generating the keys pub_A and 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...