Security of JavaCard
smart card applets

Erik Poll

University of Nijmegen
http://www.cs.kun.nl/~erikpoll

Contents

- Smart cards
- New generation smart cards
  - smart card applets
  - language level security
  - applet security
- Applet Security

SMART CARDS

Nice cryptography, but

- Where do I keep my private keys?
- Who do I trust to do my en/decryption?

For traditional authentication - face/voice recognition - this is not a problem!

Smart Cards
Card with microprocessor capable of

- storing information
- processing information: en/decryption
  This is what makes a smart card smart:
  stupid cards cannot do this

Eg. bank card, mobile phone SIM

Why use smart cards?

- Private key K never leaves the card
- Card issuer does not have to trust card holder, terminal, or network
Why use smart cards?

- send password unencrypted over net (e.g. rlogin) but can we trust the network?
- send password encrypted over net (e.g. slogin) but can we trust the terminal?
- idem, but user, not terminal, does encryption but can we trust the user?
- use smart card trust no-one

NB smart card security is not perfect

Card can be physically attacked:
- Reading or writing of the chip (memory, bus)
- Analysing timing or power consumption (DPA)

NEW GENERATION SMART CARDS

Eg: Mondex, Java Card, Windows for Smart Cards

Old vs new smart cards

- one program (applet)
- written in chip-specific machine code
- burnt into ROM

- Applet written in high-level language
- compiled into bytecode
- stored in EEPROM
- interpreted on card

- multi-application: several applets on one card
- post-issuance: adding or deleting applets on card

Multi-application

Several applets on one card, possibly interacting

Eg
- credit card + loyalty program
- access to buildings + computer networks
- frequent flyer card + electronic check-in
- all of the above

Post-issuance

Additional applets downloaded onto card after it has been issued, to add or upgrade services
- eg. removing chipper and adding chipknap
- cf. downloading applets in web-browser

Post-issuance download tightly controlled: only trusted - digitally signed - applets are downloaded (using VISA Open Platform), or none at all.
Java Card

A subset of Java
- no threads, doubles, strings, gc optional
- with some extras
- persistent and transient objects
- transaction mechanism
- and increased language-level security
- standard sandbox (cf. web-browsers)
- plus firewall between applets

Java Card smart card

Advantages of new generation
- easier development of applications
  - faster and cheaper
  - high-level language
  - independent of underlying hardware
- more flexibility
  - multi-application
  - post-issuance download?

Disadvantage: Security
- incorrect or malicious applet may interfere with other applets or platform
  - eg a virus on a credit card or mobile phone
- platform can provide basic security against illegal operations
- applet should take care to provide any additional security it requires

Platform level security (platform = VM+OS)
- language level security
  - byte code verification
- OS security
  - firewall

Applet security
- anything beyond this
Context of this work
Verification of JML-annotated Java code, eg

```java
public int squareRoot(int i);
// post: \( \text{result}^2 < i \&\& i < (\text{result}+1)^2 \);
```

What can we do for applets with this approach?

Towards applet security
How to specify “applet security”?
1. Applet correctness
   method does what it should do
2. Applet security policy: access control
   method/data only accessed when allowed
3. Secure information flow
   method does not leak information

1. Applet correctness

1. Applet correctness

But: correctness \( \Rightarrow \) security?
• Limits to the expressivity of specification language
• At least: correct \( \Rightarrow \) secure

In any case: no assumptions on incoming data

No assumptions on incoming data:
```
public int squareRoot(int i);
// post: \( \text{result}^2 < i \&\& i < (\text{result}+1)^2 \);
```

```
but
public int squareRoot(int i);
// post: true;
```

```
2. Applet security policy

Access control for methods
- who may invoke which method when in the smartcard/applet life cycle
and for data
- confidentiality: who may access data
- integrity: who may modify data - modification by authorised party with uncorrupted (digitally signed) data

2. Method access control

Distinguish states in smartcard/applet life cycle. Specify who may do what in which state

```
This can be specified in JML, eg
// @pre: state -- blocked & user -- admin;
```

2. Method access control

- Method access control
  - method invoked when allowed
  - complements correctness
    - method does what it should do

- Maybe temporal logic specifications better for expressing (il)legal access control?

2. Data access control

Annotate any data access with checks

```
... 
// @assert: state == admin;
PIN = newPIN;
... 
verify that these conditions are met
```

Data access conditions already show up in the preconditions of methods?

3. Secure information flow

No sensitive information may be leaked

Traditional approach to information flow:
- distinguish high and low security level variables
- forbid assignments of high to low cq. dependencies of low on high level
- check this by
  - static analysis/type checking, or
  - model checking

3. Secure information flow

Information flow using pre/postconditions:

```
public int m(int i);
// @post: result = f(i, low level variables);
// @signal: (Exception) p(i, low level vars);
```

for some f and p means that no high security level values are leaked.
Practical in real examples?
Conclusion

Smartcard best place to keep private keys and do en/decryption

Security of smartcard application relies on
- Hardware security
- Platform security
- Applet security
- Use scenario

\}

Software

Conclusion

- How do we specify security?
- Correctness ⇒ security?

- Ongoing work:
  - applet case study
  - specification of the JavaCard API using JML

- Why formal methods?
  Needed for security evaluations (Common Criteria)