Integrating the Belgian e-ID into Android

Gauthier Van Damme  Karel Wouters  Danny De Cock  Dries Schellekens

Katholieke Universiteit Leuven  ESAT/SCD/IBBT-COSIC

Droidcon, 2011
Outline

1. The Belgian e-ID card
   - The Belgian e-ID card
   - The open source Belgian e-ID card

2. Building a secure application on Android
   - Secure Elements in Mobile Phones
   - Working with Secure Elements in Android

3. Belgian e-ID for Android
   - Belgian e-ID for Android
   - Opportunities & pitfalls for mobile e-ID
Outline

1. The Belgian e-ID card
   - The Belgian e-ID card
   - The open source Belgian e-ID card

2. Building a secure application on Android
   - Secure Elements in Mobile Phones
   - Working with Secure Elements in Android

3. Belgian e-ID for Android
   - Belgian e-ID for Android
   - Opportunities & pitfalls for mobile e-ID
The Belgian e-ID card

- March 31st, 2003: first Belgian e-ID cards issued
- Today: almost 10 million cards active
- Printed information provide for normal citizen identification
- Java Card application for secure data & key storage:
  - PKI based on X.509 v3 certificates
  - Two key pairs per citizen are defined inside this PKI
    - Authentication key pair for client authentication
    - Non-Repudiation key pair for file signature
  - Key usage is PIN protected
The e-ID Quick Key Toolset

Belgian e-ID application is closed source and rigid ⇒ Open Source version developed for innovation & security purposes

- Can be written on any Java Card
- Enables creation of e-ID clones
- Cryptographic keys of course different

See: http://code.google.com/p/eid-quick-key-toolset/
Outline

1. The Belgian e-ID card
   - The Belgian e-ID card
   - The open source Belgian e-ID card

2. Building a secure application on Android
   - Secure Elements in Mobile Phones
   - Working with Secure Elements in Android

3. Belgian e-ID for Android
   - Belgian e-ID for Android
   - Opportunities & pitfalls for mobile e-ID
Secure Elements in Mobile Phones

- Mid-1990s: idea of Secure Element (SE) accessible by mobile phone applications
  - Securely store data and keys
- Mostly Java Cards as compatible platform
- Applications managed by one entity
- 3 Secure Element versions found today:
  - In SIM, controlled by MNO
  - Embedded in phone, controlled by OEM
  - On microSD card, controlled by card distributor
Java Card 2.2.x as Secure Element

- Compliant to the ISO 7816 standard:
  ▶ Communication done through APDUs
  ▶ Predefined file types & structure
  ▶ Defines an extensible set of card commands
- Primitive subset of the Java environment:
  ▶ Data types: boolean, short, string, arrays
  ▶ Packages, classes, interfaces, and exceptions
  ▶ Dedicated libraries for cryptographic operations
  ▶ No garbage collection
  ▶ No multithreading
- Class files converted to ‘cap’ files, loaded on Java Card
  → According to Global Platform specifications for card management
  → Each applet unique AID on card
Working with Java Cards in Android

1. Develop your Java Card applet & load it into the SE
   ▶ Load it into your SE using program of your choice (e.g. GPShell)
   ▶ It will respond to APDUs coming from your mobile phone
     ▶ e.g. Verify PIN, Sign Data, Encrypt Data, etc.

2. Have a smart card driver installed on the phone

3. Develop your android application
   ▶ Need library to access driver
   ▶ Have a SmartCardClient to initialise a smart card connection

   ```java
   private static SmartcardClient smartcard;
   
   try {
       smartcard = new SmartcardClient(this, connectionListener);
   } catch (SecurityException e) {
       Log.e(LOG_TAG, "Binding not allowed, uses-permission SMARTCARD?");
       return;
   }
   
   catch (Exception e) {
       Log.e(LOG_TAG, "Exception: " + e.getMessage());
   }
   ```
Working with Java Cards in Android (2)

- Connect to your application on the smart card
- Start communicating with your application

```java
ICardChannel cardChannel;
try {
    cardChannel = smartcard.openLogicalChannel("Mobile Security Card 00 00",
        new byte[] { (byte)0xD2, 0x76, 0x00, 0x01, 0x18, 0x00, 0x02,
            (byte)0xFF, 0x49, 0x50, 0x25, (byte)0x89, 
            (byte)0xC0, 0x01, (byte)0x9B, 0x01});
    byte[] respApdu = cardChannel.transmit(
        new byte[] { (byte)0x90, 0x10, 0x00, 0x00, 0x00 });
    cardChannel.close();
    Log.i(LOG_TAG, "Smart Card response: " + respApdu);
} catch (CardException e) {
    return;
}
```

- Don’t forget to clean up smart card connection

```java
@Override
protected void onDestroy() {
    if (smartcard != null) {
        smartcard.shutdown();
    }
    super.onDestroy();
}
```
Outline

1. The Belgian e-ID card
   - The Belgian e-ID card
   - The open source Belgian e-ID card

2. Building a secure application on Android
   - Secure Elements in Mobile Phones
   - Working with Secure Elements in Android

3. Belgian e-ID for Android
   - Belgian e-ID for Android
   - Opportunities & pitfalls for mobile e-ID
Belgian e-ID for Android

- Used Giesecke & Devrient secure microSD card as Java Card
- Used open source Smart Card driver & library
- Source code & further info found at: http://code.google.com/p/seek-for-android/
- Contains:
  - Full e-ID clone on microSD card
  - Read functionalities
  - Data & file signing
  - Data loading and verification
Opportunities & pitfalls for mobile e-ID

Authenticated cryptographic primitives & ID could enable:
+ Signing documents & mails easily & everywhere
+ Secure authentication to web and other servers
+ Visual citizen identification: visual/NFC/...
+ Direct picture or video watermarking
+ Setting up encrypted communication channels (e.g. in VoIP)

Unfortunately:
- No visual security measures can be implemented
- In case of malware on phone: security & card blocking risk
  → ‘Always-on’ connection!
Future Work

Current & Future work:

- Extend e-ID for anonymous petition signing on mobile phone
- Use e-ID in mobile banking application

Questions?

Acknowledgment:

- Giesecke & Devrient for their support
- Belgian Government (FEDICT) for the close collaboration