Secure Elements for You and Me: A Model for Programmable Secure Hardware in Mobile Ecosystems

Alexandra Dmitrienko
Fraunhofer Institute for Secure Information Technology, Darmstadt, Germany

Joined work with

Marcos da Silva Ramos, Andre Rein
Fraunhofer SIT, Darmstadt, Germany

Stephan Heuser, Thien Duc Nguyen, Ahmad-Reza Sadeghi
Technische Universität Darmstadt, Germany
Motivation

• Secure hardware can drastically improve security of mobile apps
  • Provides isolated execution environment for security sensitive app routines
Secure Hardware

- Processor-based Trusted Execution Environments (TEEs)
  - ARM TrustZone, TI M-Shield
- Secure Element (SE) on UICC (or SIM) cards
- Embedded SE (eSE)
  - Standalone or NFC-based
- Secure Element (SE) on an SD memory card
Mobile Payments Apps with Secure Hardware Support

- SE-supported apps from OS vendors
  - Google Wallet and Apple Pay

- SIM-based apps from Mobile Network Operators (MNOs)
  - Vodafone payment app, MyWallet from Telekom, etc.

- But, “[...] there is a major shift towards cloud based software solutions away from hardware based technologies [...]” (Advanced Payments Report 2014, Edgar Dunn & Company)
  - E.g., solutions of PayPal and Visa rely on cloud-based services
Other Security Sensitive Apps

- Online banking, mobile ticketing, access control applications, etc.
  - Typically rely on software-based security mechanisms
    - SmartCard emulated in software
    - OS-level isolation and access control
Why?

Secure hardware is not freely programmable
Problem Description and Challenges

- Widely deployed Secure Elements (SE) and Trusted Execution Environments (TEEs) are controlled by their stakeholders.
- No effective business models exist to allow third party app developers to interact with SE/TEE stakeholders.
- Access control to secure hardware APIs is controlled either by their stakeholders or by OS vendors, but not by app developers.
Our Goal

A framework to allow third party developers to develop for secure hardware and to deploy their code

- Incentives for secure hardware stakeholders to allow such access
- Simplified interaction of third party developers and secure hardware stakeholders
- Access control to secure hardware APIs independent from OS vendors
Core Ideas

Market place code distribution ecosystem
• Bridges small and mid-size app developers and large hardware stakeholders

DRM for secure hardware code
• Hardware stakeholders grant installation rights (to users) by issuing installation tokens
• Financial incentives for hardware stakeholders

Developer-centric access control to secure hardware APIs
• Distribute access policy in installation tokens
Involved Parties

Developer D
• Develops mobile apps and applets, trustlets or trusted apps

App Market M
• Usual app market where mobile apps are published

Applet Market S
• Maintained by the stakeholder of secure hardware

Secure Element SE
• Secure hardware which can run applets, trustlets, etc.

Mobile Host
• Mobile device, such as a smartphone or tablet
System Architecture

(1) Applet certification: submit applet \( a \) and policy \( P \)

(2) Publish app \( A \)

(3) Download and install app

(4) Download applet

(5) Download token \( T_a \)

(6) Install applet: transfer token \( T_a \), applet \( a \)

App Market \( M \)

Developer \( D \)

Mobile Host \( H \)

Secure Element \( E \)

Applet Market \( S \)

Certify and publish (encrypted) applet
Policy \( P \) integrated into token

Verify token \( T_a \), extract policy \( P \), decrypt and install applet \( a \)
Applet Installation

1. Install app
2. Detect applet dependency
3. Invoke applet installer
4. Download $T_{a,a}$
5. $T_{a,a}$
6. Add policy $P$
7. Install applet

Mobile Host $H$
- Mobile app
- App installer
- Access Control Enforcer

Secure Element $E$
- ARA-M applet
- Applet
- Applet Manager
Applet Invocation

1. Invoke applet $a$
2. Fetch policy $P$ for applet $a$
3. Verify policy $P$
4. Invoke applet $a$
5. Invoke applet $a$
Associated Challenge

- Market-driven code provisioning scheme may stipulate rapid development of various SE applets
- Limited resources of hardware-based SEs limit number of installed applets
- Current resource quota mechanisms for SEs are not effective in case of market-driven code provisioning

Our solution: SE resource management
SE Resource Management

Applet Manager performs resource management by
- monitoring available SE resources
- maintaining statistics of applet usage
- de-installing of rarely used applets in case SE is out of resources
- installing applets on-demand in case previously de-installed applet is invoked

- On-demand applet installation
  - Transparent to mobile applications invoking the applet
  - May impact runtime performance
# Implementation

<table>
<thead>
<tr>
<th>Module</th>
<th>Size (LoC)</th>
<th>Language</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applet Manager</td>
<td>791</td>
<td>Java/Android</td>
<td>SpongyCastle Crypto API</td>
</tr>
<tr>
<td>jCardSim4Android*</td>
<td>4923</td>
<td>Java/Android</td>
<td>SmartCardIO</td>
</tr>
<tr>
<td>SmartCardIO**</td>
<td>728</td>
<td>Java/Android</td>
<td></td>
</tr>
<tr>
<td>Applet Installer</td>
<td>1124</td>
<td>Java/Android</td>
<td>SpongyCastle Crypto API, Communication API</td>
</tr>
<tr>
<td>SE Stakeholder</td>
<td>1390</td>
<td>Java 6</td>
<td>BouncyCastle Crypto API, Communication API</td>
</tr>
<tr>
<td>Developer</td>
<td>656</td>
<td>Java 6</td>
<td>BouncyCastle Crypto API, Communication API</td>
</tr>
<tr>
<td>Communication API</td>
<td>545</td>
<td>Java 6, Java/Android</td>
<td></td>
</tr>
</tbody>
</table>

*Port of open-source project jCardSim ([http://jcardsim.org/](http://jcardsim.org/)) to Android

** Port of [javax.smartcardio](http://jcardsim.org/) classes from Open-JDK v7 to Android
External SE Deployment

- Host on mobile device
- Emulated SE on a smartwatch (or any other wearable)
## Performance Evaluation

<table>
<thead>
<tr>
<th></th>
<th>Applet installation*, ms</th>
<th>Applet de-installation*, ms</th>
<th>Applet execution**, ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>H and E on the same smartphone</td>
<td>46.27 ± 19.19</td>
<td>15.76 ± 7.83</td>
<td>38.43 ± 18.47</td>
</tr>
<tr>
<td>H on the smartphone, E on the smartwatch</td>
<td>415.27 ± 78.00</td>
<td>205.36 ± 72.54</td>
<td>150.34 ± 50.07</td>
</tr>
<tr>
<td>Hardware-based SE</td>
<td></td>
<td></td>
<td>24.47 ± 1.86</td>
</tr>
</tbody>
</table>

* Applet size 10953 Bytes
** Send 4 bytes to the applet and reply with 10 bytes

- Smartphone: Samsung Galaxy S3 (Android 4.4)
- Smartwatch: Samsung Galaxy Gear SM-V700 (Android 4.2)
- Hardware-based SE: Mobile Security Card SE 1.0 (G&D)
Summary and Further Work

Our model for programmable secure hardware

- Provides incentives for SE/TEE stakeholders
- Bridges small-size app developers and large stakeholders
- Developer-centric access control to secure hardware APIs compatible to Global Platform specifications

Further work:

- Support for Trusted Execution Environments
- Support for multiple types of secure hardware on a single platform