Old, New, Borrowed, Blue:
A Perspective on the Evolution of Mobile Platform Security Architectures

N. Asokan
ACM CODASPY ’11

Joint work with Kari Kostiainen, Elena Reshetova, Jan-Erik Ekberg

Feb 22, 2011
Recent interest in smartphone security

Understanding android security
W Enck, M Ongtang... - Security & Privacy, IEEE, 2009 - ieeexplore
50 Published by the IEEE Computer Society • 1540-7993/09/$25.00 ©

Platform Security and Symbian Signed: Foundation for a Secure
B Morris - Symbian Developer Network Report, 2008 - cens.ucla.edu

Virtualization as an enabler for security in mobile devices...
Secure smartphone application platforms: challenges

<table>
<thead>
<tr>
<th></th>
<th>“Feature phones”</th>
<th>PCs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Smartphones</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open software platforms</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Third party software</td>
<td>Java ME</td>
<td>√</td>
</tr>
<tr>
<td>Internet connectivity</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Packet data, WiFi</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Personal data</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Location, contacts,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>communication log</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Risk of monetary loss</td>
<td>√</td>
<td>?</td>
</tr>
<tr>
<td>Premium calls</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Is smartphone platform security different?**
Outline

• A bit of background on requirements for securing mobile phones
• Basics on hardware security enablers
• Comparison of modern mobile (software) platform security architectures
• Discussion: open issues and summary
Background
Platform security requirements for mobile phones

Mobile network operators;
1. Subsidy locks → immutable ID
2. Copy protection → device authentication, app. separation
3. ...

Regulators;
1. RF type approval → secure storage
2. Theft deterrence → immutable ID
3. ...

End users;
1. Reliability → app. separation
2. Theft deterrence → immutable ID
3. Privacy → app. separation
4. ...

Closed → Open
Different Expectations compared to the PC world
Early adoption of hardware and software security

Both IMSI and IMEI require physical protection. **GSM 02.09, 1993**

Physical protection means that manufacturers shall take necessary and sufficient measures to ensure the programming and mechanical security of the IMEI. The manufacturer shall also ensure that the knowledge of how to change the IMEI (where applicable) remains securely under his control.

The IMSI is stored securely within the SIM. **3GPP TS 42.009, 2001**

The IMEI shall not be changed after the ME’s final production process. It shall resist tampering, i.e. manipulation and change by any means (e.g. physical, electrical and software).

**NOTE:** This requirement is valid for new GSM Phase 2 and Release 96, 97, 98 and 99 MEs type approved after 1st June 2002.

Different starting points:

- **~2005** Symbian OS Platform Security
- **~2002** M-Shield Mobile Security Technology
- **~2001** TrustZone® Security Foundation by ARM®
- **~2001** J2ME
Hardware security enablers
Hardware support for platform security

- Public key hash
- Trust root
- Base identity
- Crypto Library
- Boot sequence (ROM)
- Start of boot code
- E.g., serial number

TCB for platform software

Basic elements in immutable storage
Secure bootstrapping

- Code certificate
- Boot code hash
- Trust root
- Base identity
- Crypto Library
- Boot sequence (ROM)
- TCB for platform software
- Launch platform boot code

Ensure only authorized boot image can be loaded
Secure bootstrapping

- Trust root
- Base identity
- Crypto Library
- Boot sequence (ROM)
- Code certificate
  - Boot code hash

TCB for platform software

Launch platform boot code

Validate and execute

Ensure only authorized boot image can be loaded

Hardware security
Securely assign different identities to the device

Identity binding

- Identity certificate
  - Base identity
  - Assigned identity
- Code certificate
  - Boot code hash
- E.g., IMEI, link-layer addresses, ...
- Trust root
- Base identity
- Crypto Library
- Boot sequence (ROM)

Secure boot

TCB for platform software

Launch platform boot code

Validate and accept assigned ID
**Trusted Execution Environment (TrEE): isolation**

- **Identity certificate**
  - Base identity
  - Assigned identity
- **Code certificate**
  - Boot code hash
  - TrEE code hash

- **Validate and execute**
- **Isolated execution**

**TrEE**

- **Device key**
- **TrEE code**

**Secure boot**

- **Trust root**
- **Base identity**
- **Crypto Library**
- **Boot sequence (ROM)**

**TCB for platform software**

- **Launch platform boot code**
- **TrEE API**

**Hardware security**

**Authorized code execution, isolated from the OS**
Trusted Execution Environment (TrEE): integrity

- Identity certificate
  - Base identity
  - Assigned identity
- Code certificate
  - Boot code hash
- Code certificate
  - TrEE code hash

- Trust root
- Base identity
- Crypto Library
- Boot sequence (ROM)

- TCB for platform software
  - Secure boot
  - Launch platform boot code

- TrEE
  - Configuration register(s)
  - Device key
  - TrEE code
  - Non-vol. memory or counter

- Securing TrEE sessions, authenticated boot
- Rollback protection for persistent secure storage

Integrity-protected state information within the TrEE
Device authentication

- Identity certificate
  - Base identity
  - Assigned identity
- Code certificate
  - Boot code hash
- Code certificate
  - TrEE code hash
- External trust root
- Device certificate
  - Identity
  - Public device key
- Configuration register(s)
  - Device key
  - TrEE code
  - Non-vol. memory or counter

Secure boot

- Trust root
- Base identity
- Crypto Library
- Boot sequence (ROM)

TCB for platform software

Launch platform boot code

TrEE API

Prove device identity or properties to external verifier
Hardware platform security features: summary

- **Secure boot**: Ensure only authorized boot image can be loaded
- **Authenticated boot**: Measure and remember what boot image was loaded
- **Identity binding**: Securely assign different identities to the device
- **Secure storage**: Protect confidentiality/integrity of persistent data
- **Isolated execution**: Run authorized code isolated from the device OS
- **Device authentication**: Prove device identity to external verifier
- **Remote attestation**: Prove device configuration/properties to external verifier
Current hardware security architectures

- TI M-Shield, ARM TrustZone
  - TrEE: secure processor mode; isolated execution for small amounts of arbitrary code
  - On-chip RAM, Write-once storage in E-fuse
- Trusted Computing Group Trusted Platform Module (TPM)
  - TrEE: standalone processor; isolated execution for pre-defined algorithms
  - Used with Dynamic Root of Trust (DRTM) for isolated execution of arbitrary code
  - Platform Configuration Registers (PCRs), counters in TPM for recording system state
- Trusted Computing group Mobile Trusted Module (MTM)
  - Mobile variant of TPM which can be implemented on other TrEEs (e.g., TrustZone)
Uses of hardware security

Recap from features
• Secure/authenticated boot
• Identity binding/device authentication
• Secure storage
• Remote attestation

Uses of hardware security:
• Device initialization, DRM, subsidy lock

What about new uses? How can developers make use of hardware security?
Software platform security
Open mobile platforms

• Symbian
  • Descendant of EPOC OS for Psion devices; Platform security since ~2004
  • Still a dominant Smartphone OS

• Java ME
  • Java variant for embedded devices ~2001
  • Most widely deployed mobile application platform

• Android
  • Dominant and rapidly growing Smartphone OS ~2007

• MeeGo
  • Linux-based OS for various device classes by Intel and Nokia ~2010
  • Mobile Simplified Security Framework (MSSF) is a proposal
General model for mobile platform security

Three phases
- Distribution
- Installation
- Run-time enforcement

Common techniques
- Permission-based access control architecture
- Code signing
Distribution

- Developer produces a software package
  - Code
  - Manifest
- May submit to a signer for a trusted signature
- Distributed to device via online stores (typically)
Installation

- Installer consults local policy and trusted signature
  - identify application
  - grant requested privileges
- Installer may prompt user
Run-time enforcement

- Monitor checks if subject has privileges for requested access
- Resource may perform additional checks
- User may be prompted to authorize access
Software platform security design choices

- Is hardware security used to secure OS bootstrapping?
- How are applications identified at install- and run-time?
- How is a new version of an existing application verified?
- How finely is access control defined?
- What is the basis for granting permissions?
- What is shown to the user?
- When are permissions assigned to a principal?
- How is the integrity of installed applications protected?
- How does a resource declare the policy for accessing it, and how is it enforced?
- How can applications protect the confidentiality and integrity of their data?
## OS bootstrapping

Is hardware security used to secure OS bootstrapping?

<table>
<thead>
<tr>
<th>Symbian</th>
<th>Java ME</th>
<th>Android</th>
<th>MSSF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure boot</td>
<td>Not applicable</td>
<td>No?</td>
<td>Authenticated boot: “Normal mode” vs “Developer mode”</td>
</tr>
</tbody>
</table>
## Application identification

**How are applications identified at install- and run-time?**

<table>
<thead>
<tr>
<th>Symbian</th>
<th>Java ME</th>
<th>Android</th>
<th>MSSF</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Install- and run-time</strong></td>
<td>Install-time</td>
<td>Install-time</td>
<td>Install-time</td>
</tr>
<tr>
<td>• Protected range</td>
<td>• Signing key and midlet attributes</td>
<td>• Signing key</td>
<td>Software source</td>
</tr>
<tr>
<td>SID, VID (managed)</td>
<td></td>
<td>Run-time</td>
<td>(signing key), package name</td>
</tr>
<tr>
<td>• UID (unmanaged)</td>
<td></td>
<td></td>
<td>Run-time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Run-time</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Unix UID and package name (locally</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>unique)</td>
<td></td>
</tr>
</tbody>
</table>

Software Platform security

NA, J-EE Feb 2011
### Application update

**How is a new version of an existing application verified?**

<table>
<thead>
<tr>
<th>Symbian</th>
<th>Java ME</th>
<th>Android</th>
<th>MSSF</th>
</tr>
</thead>
</table>
| • Protected SID, VID: trusted signature  
  • Rest: no controls | • Signed midlets: “same origin” policy  
  • Unsigned midlets: user prompt | “Same origin” policy      | “Same or higher origin” policy |

**Software Platform security**
### Permission granularity

**How finely is access control defined?**

<table>
<thead>
<tr>
<th>Symbian</th>
<th>Java ME</th>
<th>Android</th>
<th>MSSF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed set of “capabilities” (21)</td>
<td>Fine-grained permissions (many)</td>
<td>• Fine-grained permissions (112) • Linux access control</td>
<td>• Fine-grained resource-tokens • Linux access control</td>
</tr>
</tbody>
</table>

**Android and MSSF:** Each application is installed under a separate Linux UID
Permission assignment: prompting

What is shown to the user?

- Usually, nothing,
- Capability descr.
- 21 capabilities

<table>
<thead>
<tr>
<th>Platform</th>
<th>What is shown to the user?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbian</td>
<td>Usually, nothing, Capability descr., 21 capabilities</td>
</tr>
<tr>
<td>Java ME</td>
<td>Function group description, 15 groups</td>
</tr>
<tr>
<td>Android</td>
<td>Permission group description, 11 groups</td>
</tr>
<tr>
<td>MSSF</td>
<td></td>
</tr>
</tbody>
</table>

E.g., NetAccess, PhoneCall, Location, ...

E.g., LOCATION, NETWORK, ACCOUNTS, ...

Example screenshots of different platforms showing permission prompts:

- **Symbian**: Application access window with options like Read user data, Use network or make phone calls, Access Positioning data.
- **Java ME**: Permission group description with options for Gmail, networking, and accounts.
- **Android**: Permission request window with options for location, network communication, and accounts.

Software Platform security
## Permission assignment: timing

When are permissions assigned to a principal?

<table>
<thead>
<tr>
<th>Symbian</th>
<th>Java ME</th>
<th>Android</th>
<th>MSSF</th>
</tr>
</thead>
<tbody>
<tr>
<td>install-time assignment</td>
<td>run-time prompts</td>
<td>install-time assignment</td>
<td>install-time assignment, run-time privilege shedding possible</td>
</tr>
</tbody>
</table>

*Symbian and MSSF*: Permissions of app loading a DLL $\subseteq$ Permissions of DLL
Application integrity

How is the integrity of installed applications protected?

<table>
<thead>
<tr>
<th>Symbian</th>
<th>Java ME</th>
<th>Android</th>
<th>MSSF</th>
</tr>
</thead>
<tbody>
<tr>
<td>dedicated directory</td>
<td>java sandboxing</td>
<td>java sandboxing, Linux access control</td>
<td>• IMA, Smack</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Offline protection with EVM and TrEE</td>
</tr>
</tbody>
</table>

Integrity Measurement Architecture (IMA)

→ store hash of file (in extended attribute `security.ima`) and verify on launch

Extended Validation Module (EVM)

→ store MAC of all extended attributes (in `security.evm`) and verify on access
Discussion
Recurring themes: borrowed and adapted

Software platform security architectures
- Permission-based platform security architecture
  - VAX /VMS privileges (circa 1970s), but applied to programs
- Code signing
  - Mid ‘90s

Hardware enablers
- Hardware-support for platform security
  - Cambridge CAP etc. (circa 1970s); extended to Trusted Execution Environments
- Hardware-assisted secure storage; Secure and authenticated boot
  - TCPA and TCG (late 1990s), academic research projects (mid 1990s)
  - extended (private secure storage for applications), adapted (normal vs. developer mode in MSSF)
Open issues

• Permission granularity:
  • Coarse-grained permissions vs. principle of least privilege
  • But fine-grained permissions vs. user/developer confusion

• Permission assignment
  • Is it sensible to let end users make policy assignment decisions?

• Centralized vetting for appropriateness
  • Can a central authority decide what is offensive (obscene?, violent?, intolerant?)
  • Can there be crowd-sourced or “clique-sourced” alternatives? [Chia et al]

• Colluding applications
  • How to detect/prevent applications from pooling their privileges? [Capkun et al]
How can developers make use of hardware security?

A credential platform that leverages on-board trusted execution environments

Secure yet inexpensive
On-board user credentials

An open credential platform that leverages on-board trusted execution environments.

Secure yet inexpensive
ObC Architecture

On Trusted Execution Environments (TrEEs) with

- Secure execution (within TrEE)
- Secure storage (secret key OPK in TrEE)
- Certified device keypair (PK_{dev}/SK_{dev} in TrEE)
ObC Architecture

On Trusted Execution Environments (TrEEs) with

- Secure execution (within TrEE)
- Secure storage (secret key OPK in TrEE)
- Certified device keypair (PK_{dev}/SK_{dev} in TrEE)

\[ \text{Credential} = \text{program} + \text{secret} \]
ObC Architecture

On Trusted Execution Environments (TrEES) with

- Secure execution (within TrEE)
- Secure storage (secret key OPK in TrEE)
- Certified device keypair (PK_{dev}/Sk_{dev} in TrEE)

\[ \text{Credential} = \text{program} + \text{secret} \]

More in ACM ASIACCS ’09 paper
Implementations for different OSs/TrEES

Available for researchers to experiment with
Summary

• Strong push for mobile phone security from the early days on
• Widespread deployment of hardware and software platform security
  • concepts were borrowed from old systems but adapted with new twists
• Several open issues remain

On-board Credentials opens up hardware security features for new uses
• Prototype available for researchers to experiment with