A Perspective on the Evolution of Mobile Platform Security Architectures

N. Asokan
Nokia Research Center

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

Padova, July 2012
Recent interest in smartphone security

Understanding android security
W Enck, M Ongtang... - Security & Privacy, IEEE, 2009 - ieeexplore

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
Securing smartphone application platforms: challenges

<table>
<thead>
<tr>
<th>Smartphones</th>
<th>“Feature phones”</th>
<th>PCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open software platforms</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Third party software</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Java ME</td>
<td></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, 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. ...

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

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

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:

- **~2001**
  - J2ME

- **~2002**
  - M-Shield Mobile Security Technology

- **~2005**
  - TrustZone Security Foundation by ARM®

- **~2008**
  - Symbian OS Platform Security
  - Android
Hardware security enablers
Hardware support for platform security

Basic elements in immutable storage

TCB for platform software

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

Hardware security

NA, Kko, JEE, Nokia Research Center 2011-2012
Secure bootstrapping

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

Code certificate: Boot code hash

Validate and execute

TCB for platform software

Launch platform boot code

Ensure only authorized boot image can be loaded
Identity binding

- Identity certificate
  - Base identity
  - Assigned identity

- Code certificate
  - Boot code hash

- Trust root
- Base identity

- Crypto Library
- Boot sequence (ROM)

- Secure boot
  - TCB for platform software

- Validate and accept assigned ID

E.g., IMEI, link-layer addresses, ...

Securely assign different identities to the device

Launch platform boot code

Hardware security
**Trusted execution environment (TEE)**

- **Identity certificate**
  - Base identity
  - Assigned identity

- **Code certificate**
  - Boot code hash
  - TEE code hash

- **Crypto Library**

- **Secure boot**
  - Boot sequence (ROM)

- **Trust root**
  - Base identity

- **Device key**

- **TEE code**

- **TEE API**

**Trusted execution**

- Validate and execute
- Isolated execution
- Basis for secure external storage

**TCB for platform software**

- Launch platform boot code

**Hardware security**

- Authorized execution of arbitrary code, isolated from the OS; access to device key

- Trusted execution environment (TEE)
  - NA, KKo, JEE, Nokia Research Center 2011-2012

- Hardware security
Secure state

- **Trust root**
- **Base identity**
- **Assigned identity**

**Crypto Library**
- **Boot sequence (ROM)**

**Secure boot**

- **Code certificate**
  - **Boot code hash**
  - **TEE code hash**

- **Configuration register(s)**
  - **Device key**
  - **TEE code**

**TEE**
- **Non-vol. memory or counter**
- **Rollback protection for persistent secure storage**

- **Launch platform boot code**
- **TEE API**

**TCB for platform software**

**Identity certificate**
- **Base identity**
- **Assigned identity**

**Securing TEE sessions, authenticated boot**

**Integrity-protected state within the TEE**

- **NA, Kko, JEE, Nokia Research Center 2011**

- Hardware security
Device authentication

- Identity certificate
  - Base identity
  - Assigned identity
- Code certificate
  - Boot code hash
  - TEE code hash
- External trust root
- Device certificate
  - Identity
  - Public device key
- Trust root
- Base identity
- Crypto Library
- Boot sequence (ROM)
- Secure boot
- TCB for platform software
- Configuration register(s)
  - Device key
  - TEE code
  - Non-vol. memory or counter
- TEE API
- Launch platform boot code

Prove device identity or properties to external verifier

Device authentication, secure provisioning, attestation

Hardware security
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
Hardware security architectures (mobile)

ARM TrustZone and TI M-Shield

• Augments central processing unit: “Secure processor mode”

• Isolated execution with on-chip RAM: Very limited (<20kB)

• Secure storage: Typically with write-once E-fuses

• Usually no counters or non-volatile memory: Cost issue
Hardware security architectures (TCG)

• Trusted Platform Module (TPM)
  – Standalone processor on PCs
  – Isolated execution for pre-defined algorithms
  – Arbitrary isolated execution with DRTM (“late launch”)
  – Platform Configuration Registers (PCRs)
  – Monotonic counters

• Mobile Trusted Module (MTM)
  – Mobile variant of TPM
  – Defines interface
  – Implementation alternatives: TrustZone, M-Shield, software
Uses of hardware security

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

• Uses of hardware security (device manufacturer)
  – Device initialization
  – DRM
  – Subsidy lock

• How can developers make use of hardware security?
Software platform security
Open mobile platforms

- Java ME ~2001
  - For “feature phones”
  - 3 billion devices!
  - Not supported by most smartphone platforms

- Symbian ~2004
  - First “smartphone” OS
  - App development in C++ (Qt)

- Android ~2007
  - Linux-based OS
  - App development in Java

- MeeGo ~2010
  - Linux-based OS
  - App development in C++ (Qt)
  - MSSF (Intel Tizen)

- Windows Phone ~2010
  - App development in .NET
Mobile platform security model

• Common techniques
  – Application signing
  – Permission-based access control architecture
  – Application isolation

• Three phases
  1. Distribution
  2. Installation
  3. Run-time enforcement
**Distribution**

- Developer produces a software package
  - Code
  - Manifest

- May submit to a signer for a trusted signature

- Distributed to device via on-line 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? (e.g., secure boot, secure storage)
• 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>
<th>Windows Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure boot</td>
<td>Not applicable</td>
<td>Typically no</td>
<td>Authenticated boot: “Normal mode” vs “Developer mode”</td>
<td>Secure boot</td>
</tr>
</tbody>
</table>

NA, KKo, JEE, Nokia Research Center 2011-2012
## Application identification

### How are applications identified at install and runtime?

<table>
<thead>
<tr>
<th>Symbian</th>
<th>Java ME</th>
<th>Android</th>
<th>MSSF</th>
<th>Windows Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install and runtime:</td>
<td>Install:</td>
<td>Install:</td>
<td>Install:</td>
<td>Install and runtime:</td>
</tr>
<tr>
<td></td>
<td>• Protected range SID and VID</td>
<td>• Signing key</td>
<td>• Software source (signing key)</td>
<td>• Unique ID (assigned by marketplace)</td>
</tr>
<tr>
<td></td>
<td>(managed)</td>
<td></td>
<td>• Package name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• UID (unmanaged)</td>
<td></td>
<td>• Unix UID</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Package name</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(locally unique)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Runtime:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Signing key</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Software Platform security**

NA, KKo, JEE, Nokia Research Center 2011-2012
## Application update

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

<table>
<thead>
<tr>
<th>Platform</th>
<th>Symbian</th>
<th>Java ME</th>
<th>Android</th>
<th>MSSF</th>
<th>Windows Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protected SID/VID</td>
<td>• trusted signature</td>
<td>Signed midlets:</td>
<td>“Same origin” policy</td>
<td>“Same or higher origin” policy</td>
<td>Trusted signature</td>
</tr>
<tr>
<td>Rest</td>
<td>• no controls</td>
<td>“same-origin” policy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unsigned midlets:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• user prompt</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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>
<th>Windows Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed set of “capabilities” (21)</td>
<td>Fine-grained permissions (many)</td>
<td>Fine-grained permissions (112)</td>
<td>Fine-grained resource-tokens</td>
<td>Fixed set of “capabilities” (16)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Linux access control</td>
<td>Linux access control</td>
<td></td>
</tr>
</tbody>
</table>

Android and MSSF: Each application is installed under a separate Linux UID
# Permission assignment (basis)

What is the basis for granting permissions?

<table>
<thead>
<tr>
<th>Software Platform</th>
<th>Symbian</th>
<th>Java ME</th>
<th>Android</th>
<th>MSSF</th>
<th>Windows Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4 categories</td>
<td>Trusted signatures for protection domains</td>
<td>4 protection levels</td>
<td>Trusted signatures</td>
<td>Trusted signatures</td>
</tr>
<tr>
<td></td>
<td>Trusted signature (also user prompts)</td>
<td>4 permission modes</td>
<td></td>
<td>Local policy file</td>
<td>(user prompt for location)</td>
</tr>
</tbody>
</table>

**Symbian**

- Blanket, Session, One-shot, No

**Java ME**

- Trusted signatures for protection domains

**Android**

- 4 protection levels

**MSSF**

- Trusted signatures

**Windows Phone**

- Trusted signatures

---

**User**

- System, Restricted, Manufacturer

**Normal (automatic)**

- Dangerous (user-granted)

**Signature (developer-controlled)**

- SystemOrSignature (Google-controlled)
Permission assignment (user prompting)

<table>
<thead>
<tr>
<th>Symbian</th>
<th>Java ME</th>
<th>Android</th>
<th>Windows Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capability description</td>
<td>Function group description</td>
<td>Permission group description</td>
<td>User prompted only for location capability</td>
</tr>
<tr>
<td>• 21 capabilities</td>
<td>• 15 groups</td>
<td>• 11 groups</td>
<td></td>
</tr>
</tbody>
</table>

What is shown to the user?

E.g., Read user data, Use network, Access positioning, ...

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

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

Only LOCATION
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>
<th>Windows Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install-time assignment</td>
<td>Run-time prompts</td>
<td>Install-time assignment</td>
<td>Install-time assignment</td>
<td>Install-time assignment</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Run-time privilege shedding possible</td>
<td></td>
</tr>
</tbody>
</table>

Symbian and MSSF: Permissions of app loading a DLL is a subset of permissions of DLL
How is the integrity of installed applications protected?

<table>
<thead>
<tr>
<th>Symbian</th>
<th>Java ME</th>
<th>Android</th>
<th>MSSF</th>
<th>Windows Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dedicated directory</td>
<td>Java sandboxing</td>
<td>Java sandboxing</td>
<td>IMA, Smack</td>
<td>.NET sandboxing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Linux access control</td>
<td>Offline protection with EVM and TEE</td>
<td></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
## Access control policy

How does a resource declare the policy for accessing it?
How is it enforced?

<table>
<thead>
<tr>
<th>Symbian</th>
<th>Java ME</th>
<th>Android</th>
<th>MSSF</th>
<th>Windows Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declare in code</td>
<td>[System resources] Enforced by VM</td>
<td>Declare in manifest Enforced by VM</td>
<td>Declare in manifest Enforced by SMack or via libcreds</td>
<td>[System resources] Enforced by VM</td>
</tr>
<tr>
<td>Enforced by IPC framework or code</td>
<td>Enforced by VM</td>
<td>Enforced by VM</td>
<td>Enforced by VM</td>
<td>Enforced by VM</td>
</tr>
</tbody>
</table>
Application data protection

How can applications protect the confidentiality and integrity of their data?

<table>
<thead>
<tr>
<th>Symbian</th>
<th>Java ME</th>
<th>Android</th>
<th>MSSF</th>
<th>Windows Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime:</td>
<td>Runtime:</td>
<td>Runtime:</td>
<td>Runtime:</td>
<td>Runtime:</td>
</tr>
<tr>
<td>• private directory</td>
<td>• private record stores</td>
<td>• dedicated UID</td>
<td>• fine-grained data caging</td>
<td>• private directory</td>
</tr>
<tr>
<td>Off-line:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• private secure storage</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Discussion
Recurring themes (hardware enablers)

• Hardware-support for platform security
  − Cambridge CAP etc. (~1970’s)
    → Extended to Trusted Execution Environments

• Hardware-assisted secure storage

• Secure and authenticated boot
  − Academic research projects (mid 1990’s)
  − TCPA and TCG (late 1990’s)
    → Extended (private secure storage for applications)
    → Adapted (normal vs. developer mode in MSSF)
Recurring themes (software platforms)

• Permission-based platform security architectures
  − VAX /VMS privileges for user (~1970’s)
  → Adapted for applications
  − Code signing (mid 1990’s)
  → Used for application installation

• Application Signing
• Application/process isolation
Open issues

• Permission granularity
  – Coarse-grained permissions vs. principle of least privilege
  – Fine-grained permissions vs. user/developer confusion

• Permission assignment
  – Is it sensible to let end users make policy assignment decisions? [Chia et al, WWW ’12]

• Centralized vetting for appropriateness
  – Can central authority decide what is offensive?
  – Can there be crowd-sourced alternatives? [Chia et al, Nordsec ’10, Amini et al, CMU ’12]

• Colluding applications
  – How to detect/prevent applications from pooling their privileges? [Marforio et al, ETHZ ’11] [Schlegel et al, NDSS ’11]
Summary

• Mobile phone security
  – Requirements: operators, regulators, user expectations
  – Closed → open
  – Early adaptation of hardware security mechanisms

• Platform security architecture
  1. Application signing
  2. Permission based access control
  3. Application isolation
  – Many features borrowed or adapted

• Open issues remain…

• This tutorial is based on an earlier survey paper [Kostiainen et al., CODASPY 2011]; expanded version in preparation.