A Perspective on the Evolution of Mobile Platform Security Architectures

N. Asokan
Nokia Research Center

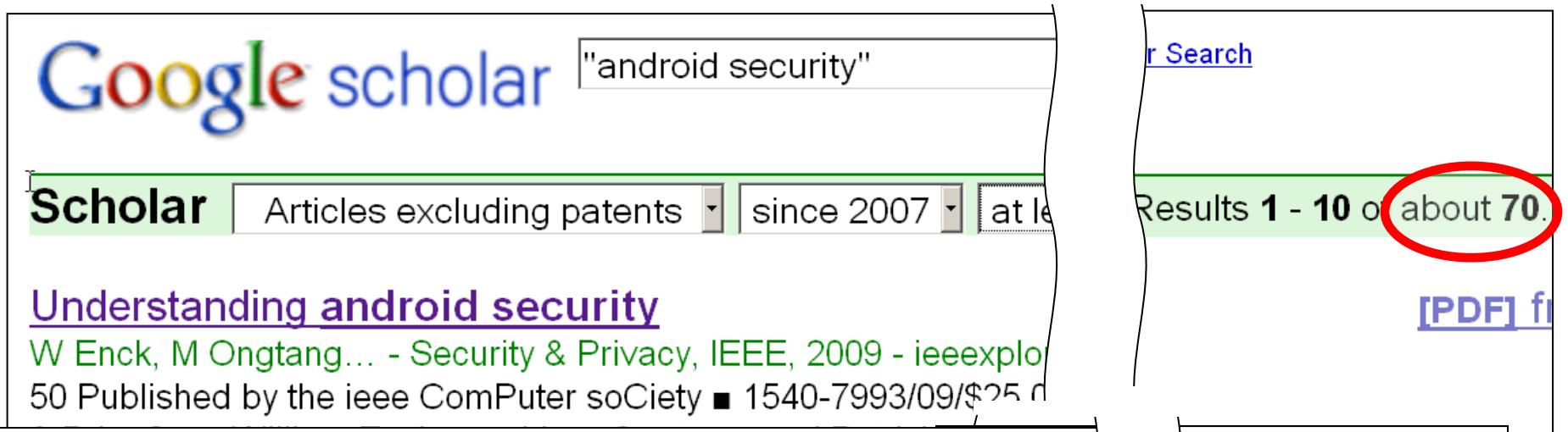
NOKIA

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

ETH Zürich Computer Science Colloquium

Mar 21, 2011

Recent interest in smartphone security



Securing smartphone application platforms: challenges

Smartphones	"Feature phones"	PCs
Open software platforms Third party software	√ Java ME	√
Internet connectivity Packet data, WiFi	√	√
Personal data Location, contacts, communication log	√	√
Risk of monetary loss Premium calls	√	?

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

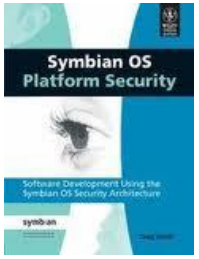
The IMEI shall not be changed after the ME's final production process. It shall resist tampering, i.e. manipulation and change, by any means (e.g. physical, electrical and software).

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:

widespread use of hardware and software platform security

~2005



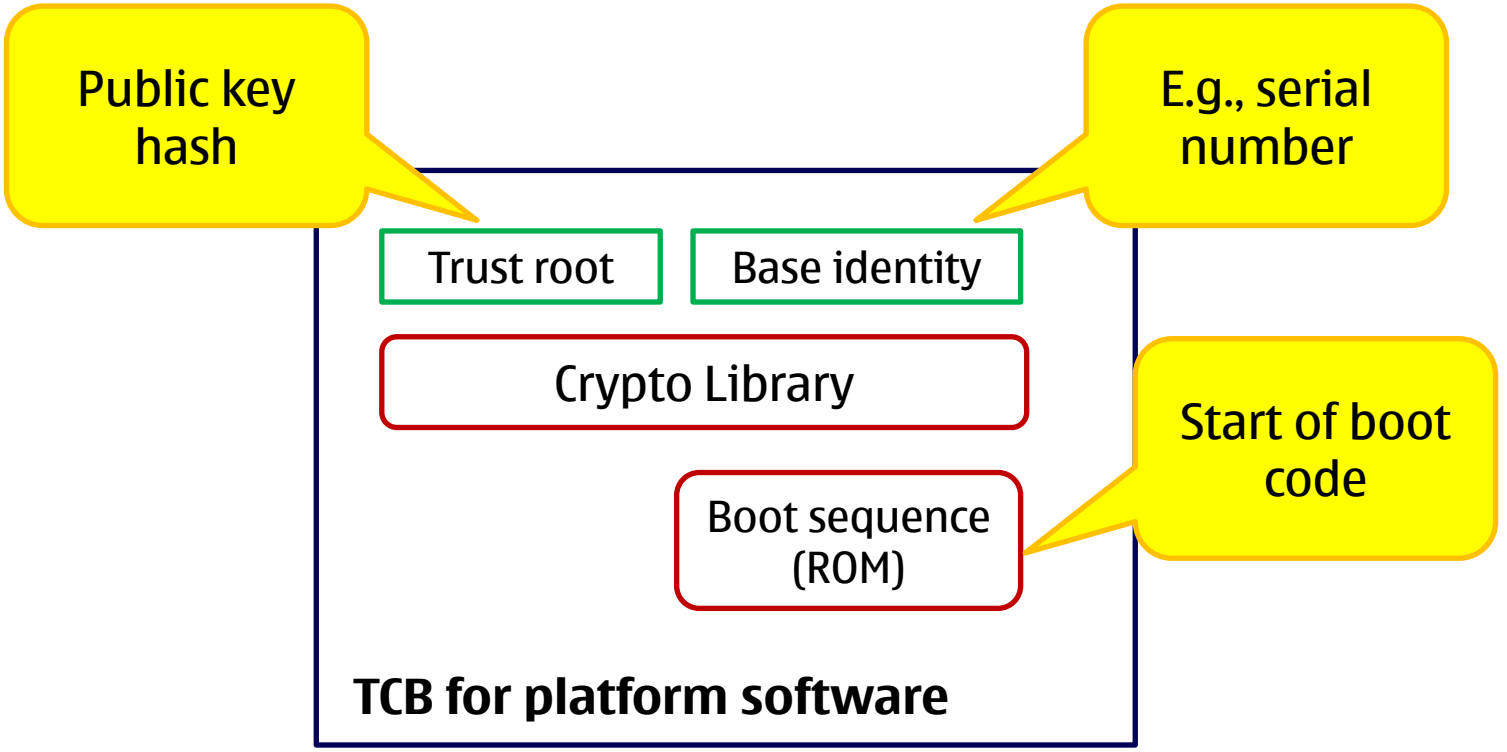
~2002

TrustZone®
Security Foundation by ARM®

~2001

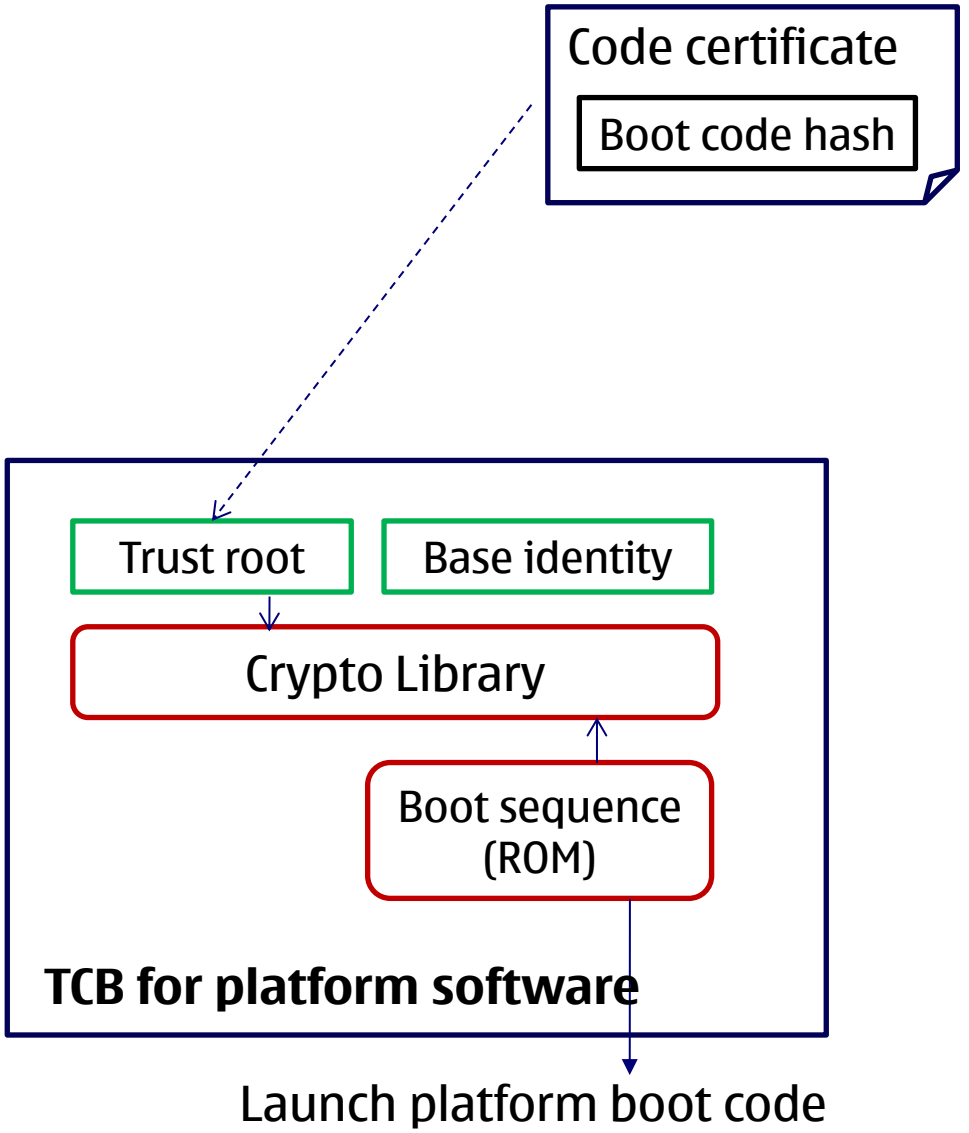
Hardware security enablers

Hardware support for platform security



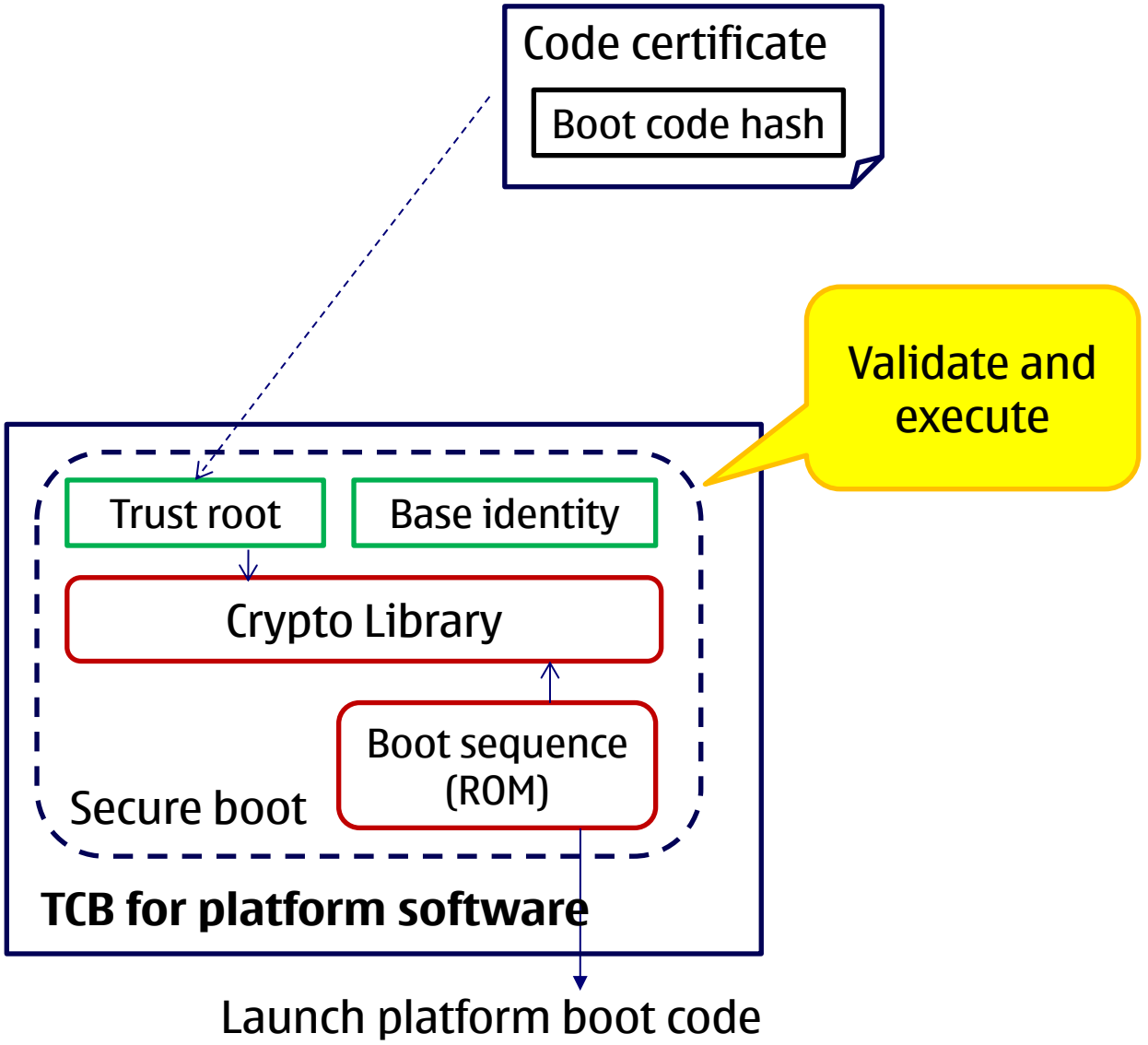
Basic elements in immutable storage

Secure bootstrapping



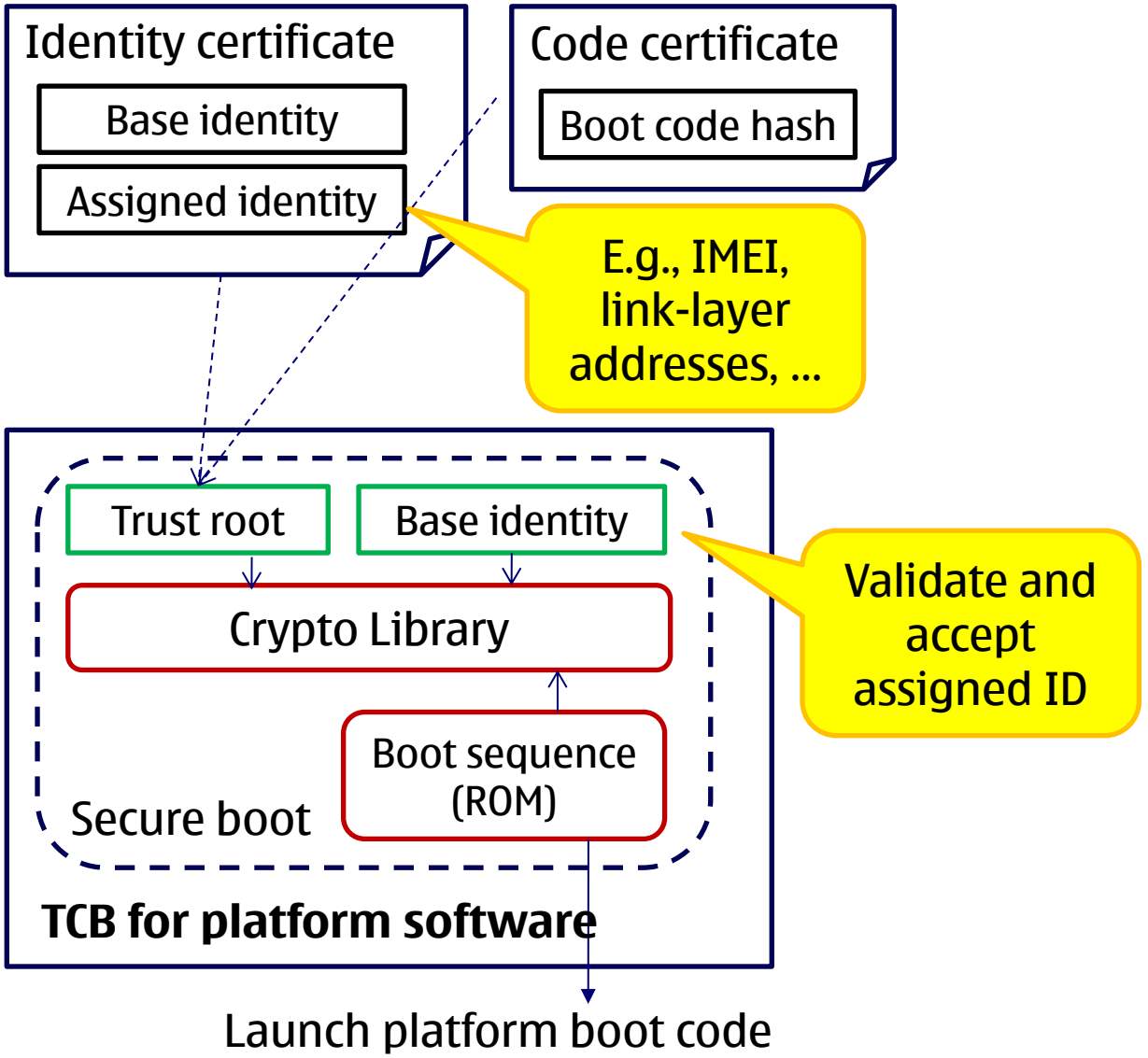
Ensure only authorized boot image can be loaded

Secure bootstrapping



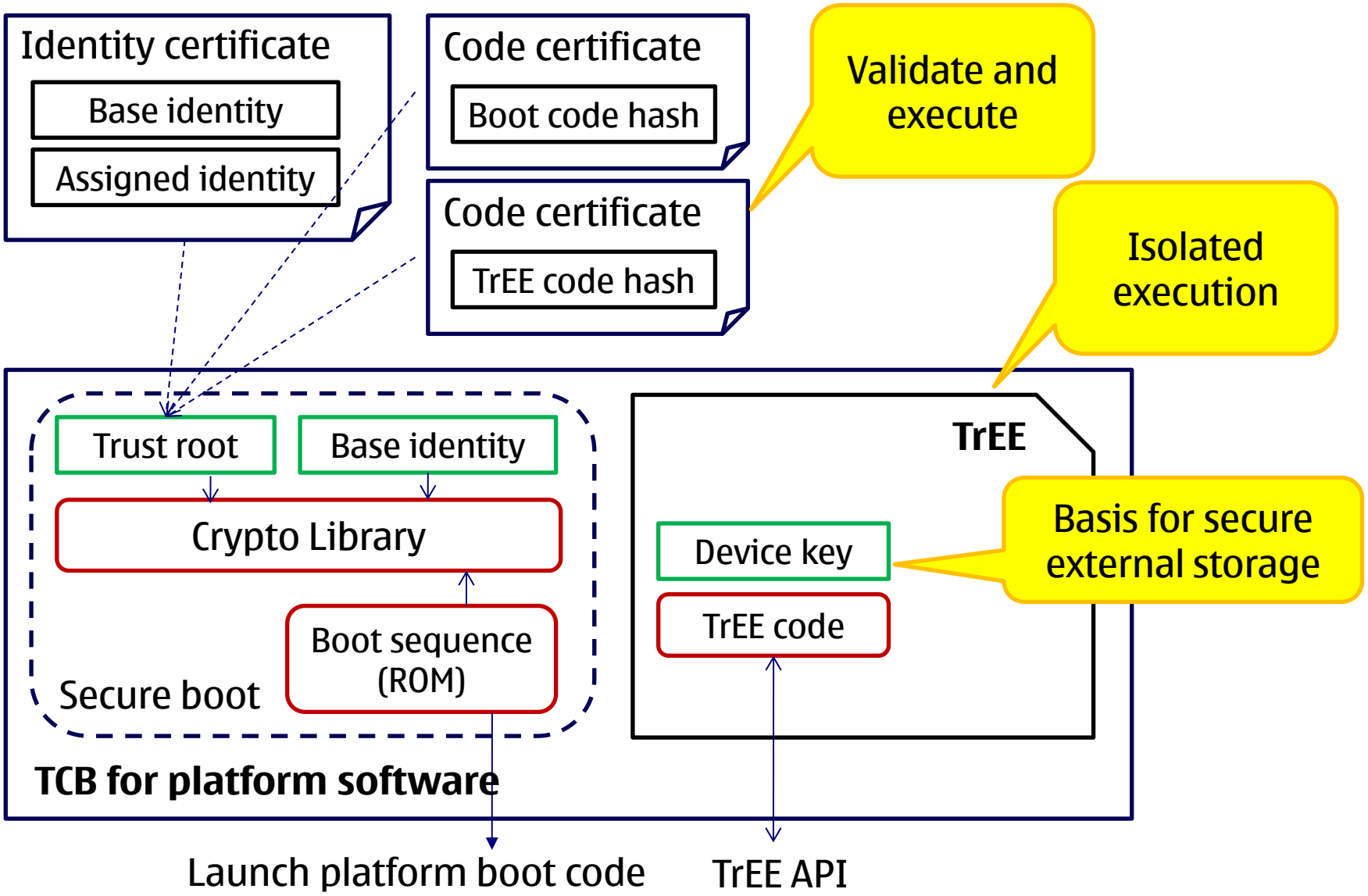
Ensure only authorized boot image can be loaded

Identity binding



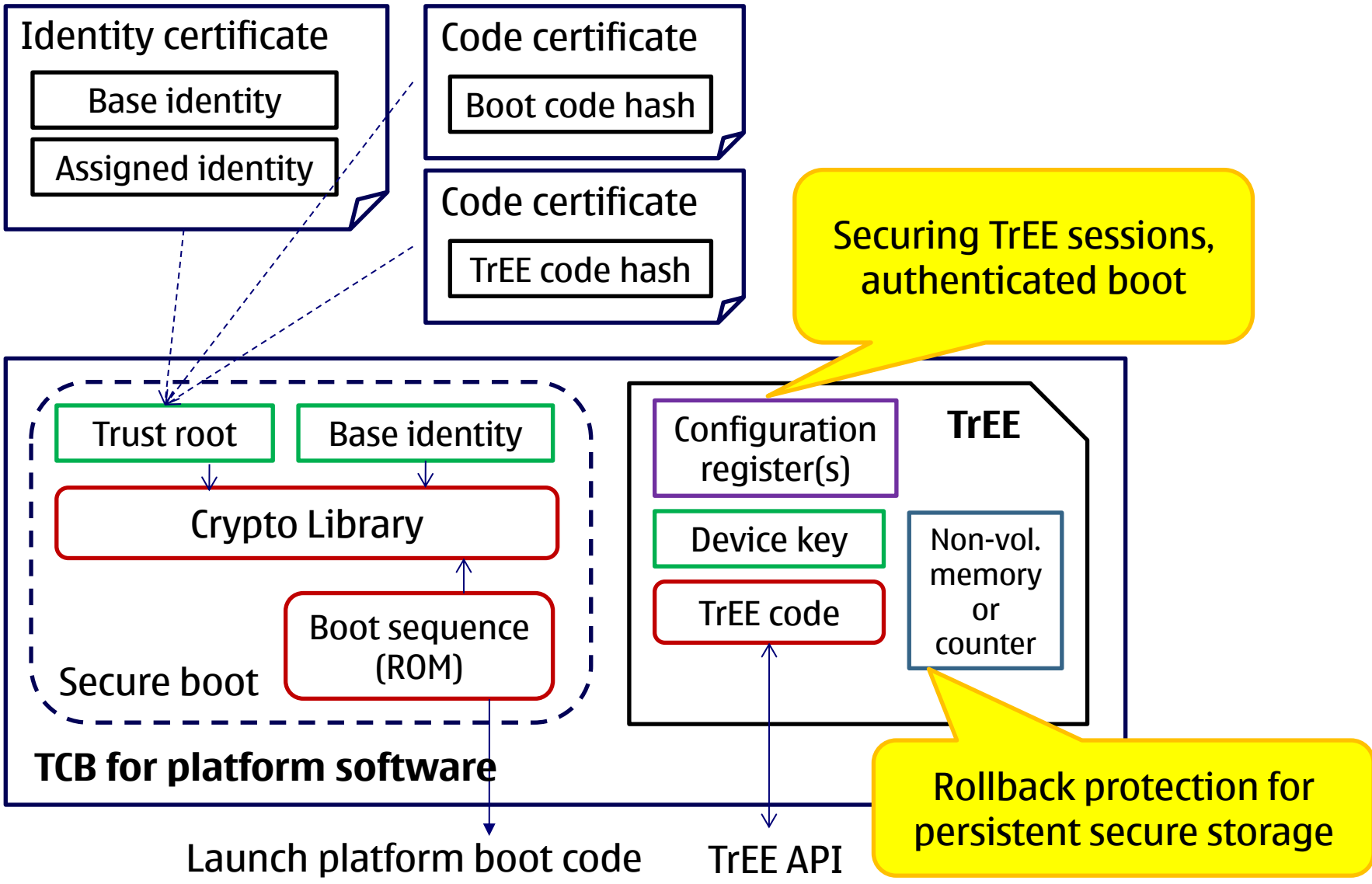
Securely assign different identities to the device

Trusted Execution Environment (TrEE): isolation



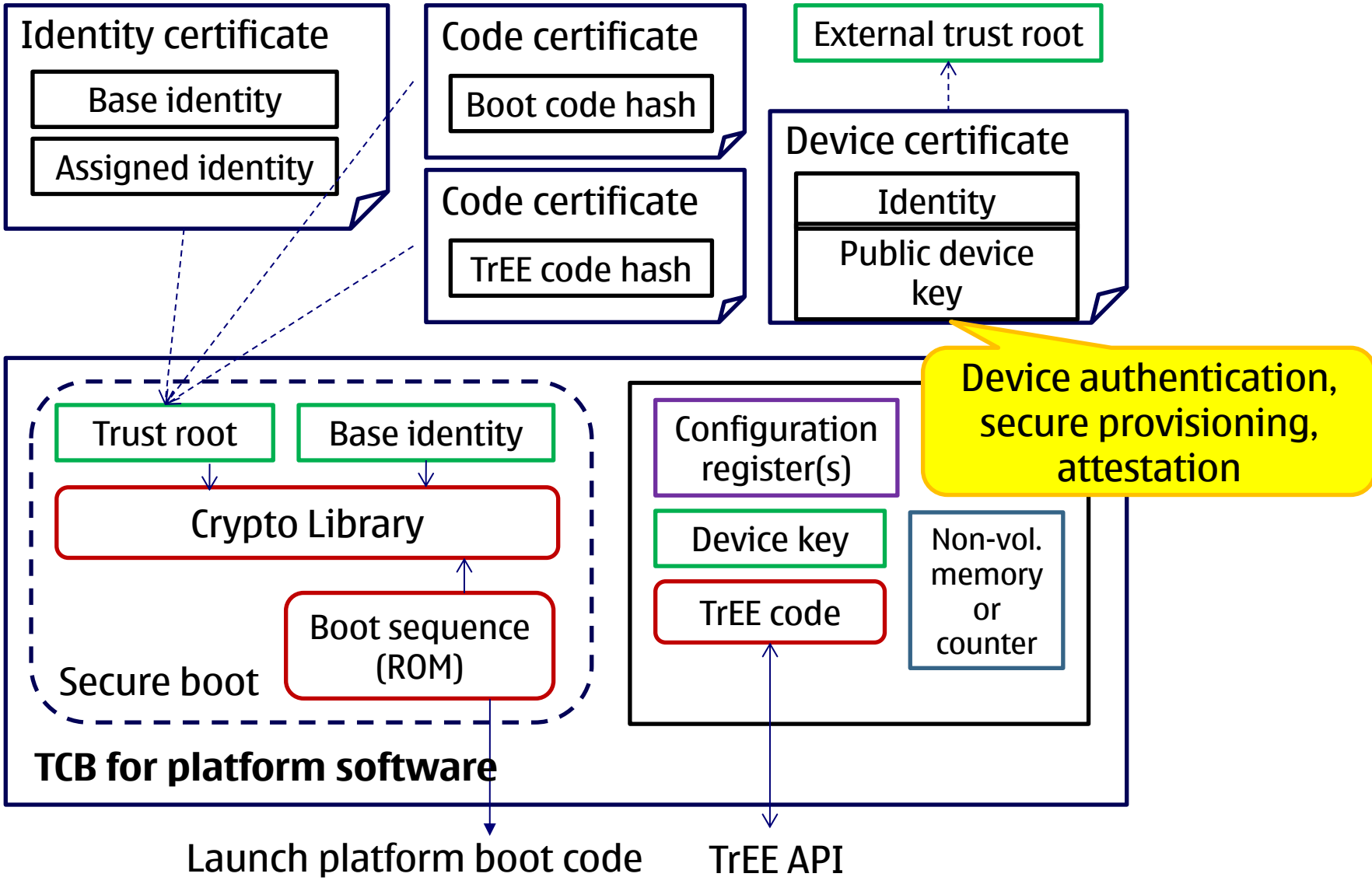
Authorized code execution, isolated from the OS

Trusted Execution Environment (TrEE): integrity



Integrity-protected state information within the TrEE

Device authentication



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

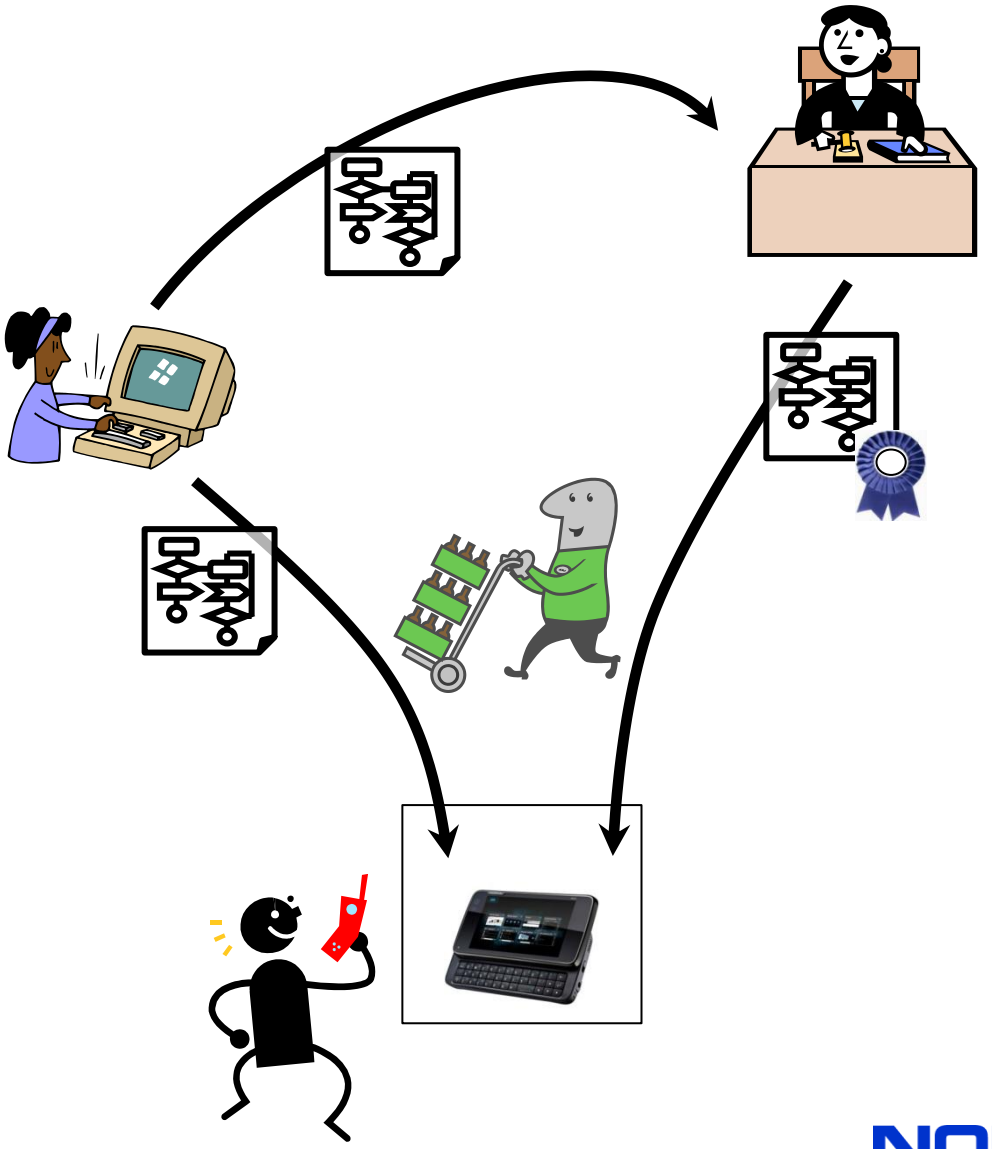
- Distribution
- Installation
- Run-time enforcement

Common techniques

- Permission-based access control architecture
- Code signing

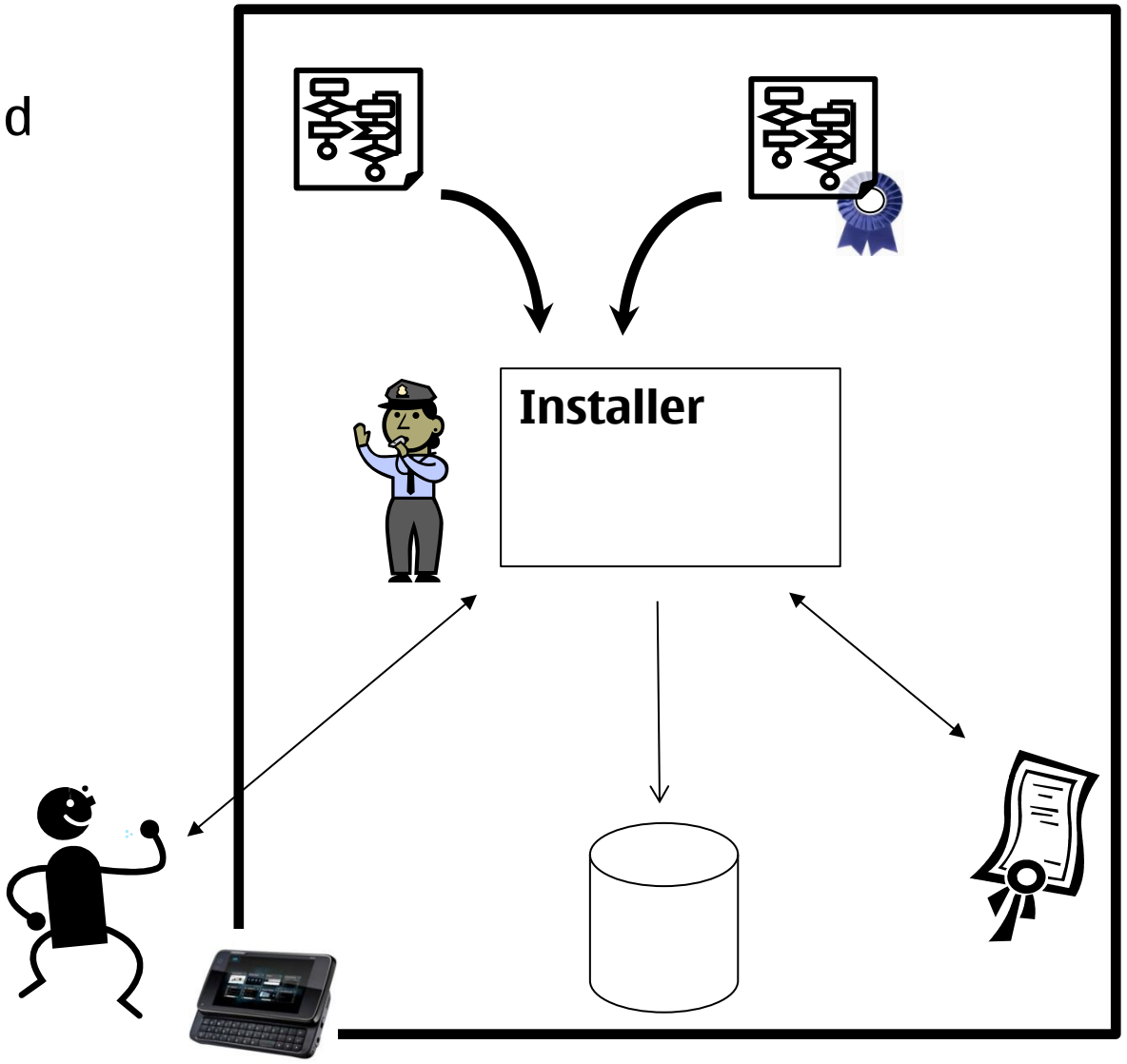
Distribution

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



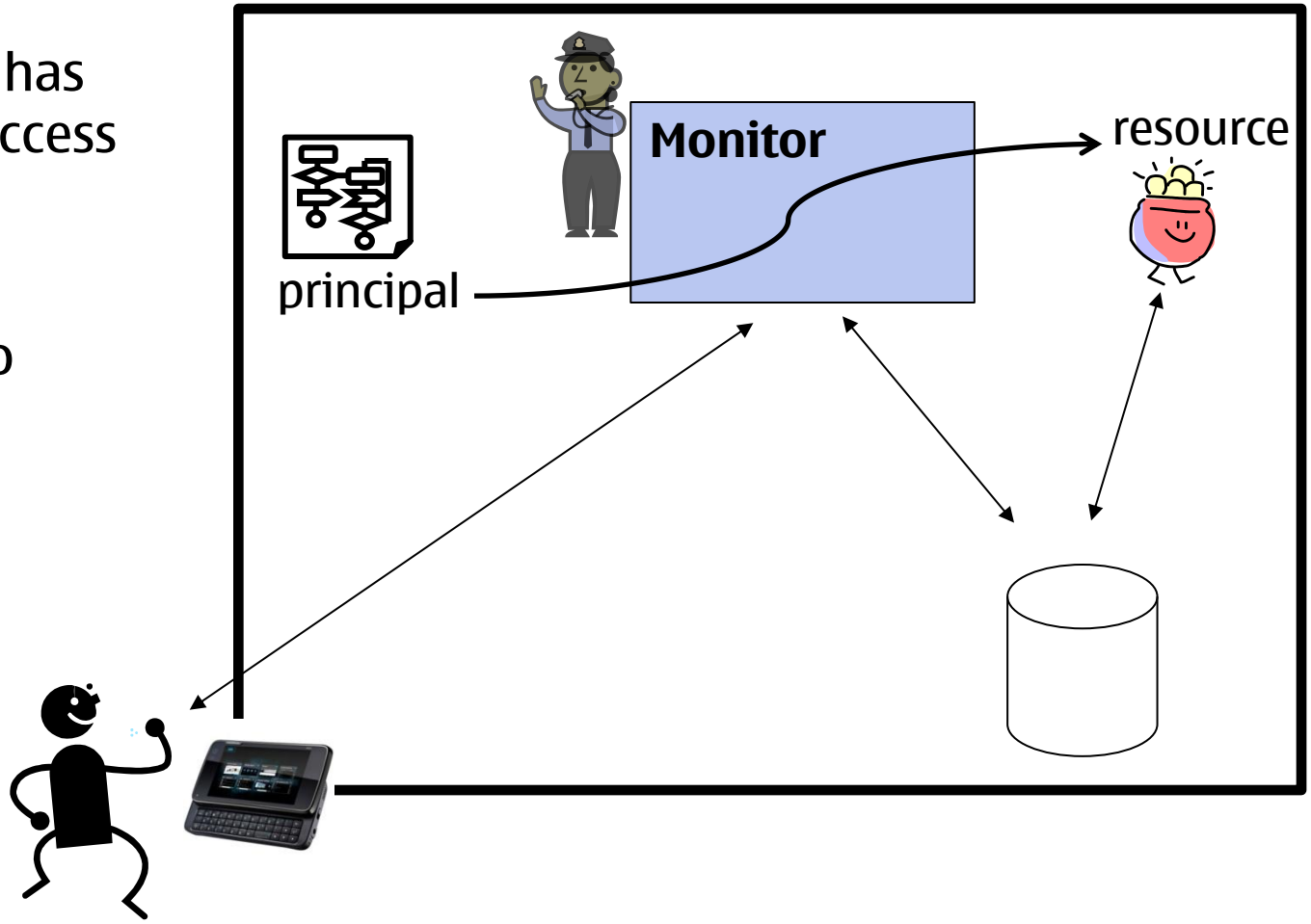
Installation

- 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 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?

Symbian	Java ME	Android	MSSF
Secure boot	Not applicable	No?	Authenticated boot: “Normal mode” vs “Developer mode”

Application identification

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

Symbian	Java ME	Android	MSSF
Install- and run-time • Protected range SID, VID (managed) • UID (unmanaged)	Install-time • Signing key and midlet attributes	Install-time • Signing key Run-time • Unix UID and package name (locally unique)	Install-time Software source (signing key), package name Run-time • Software source, package name, Application ID

Application update

How is a new version of an existing application verified?

Symbian	Java ME	Android	MSSF
• 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

Permission granularity

How finely is access control defined?

Symbian	Java ME	Android	MSSF
Fixed set of "capabilities" (21)	Fine-grained permissions (many)	• Fine-grained permissions (112) • Linux access control	• Fine-grained resource-tokens • Linux access control

Android and MSSF: Each application is installed under a separate Linux UID

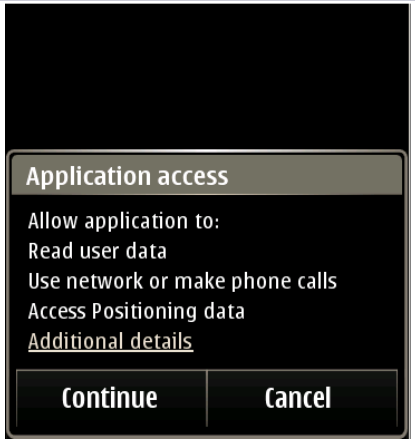
Permission assignment: prompting

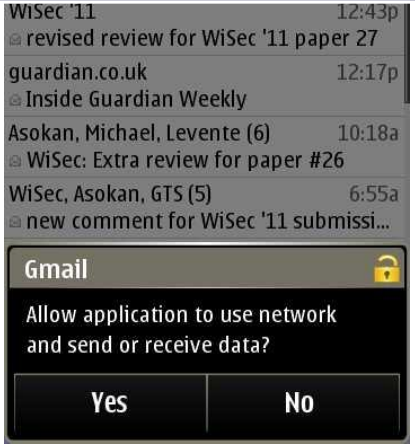
What is shown to the user?

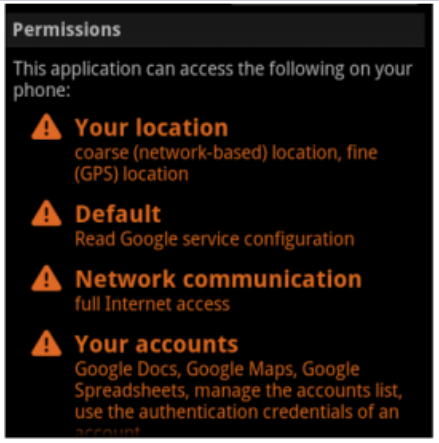
E.g., NetAccess PhoneCall Location, ...

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

Symbian	Java ME	Android	MSSF
• Usually, nothing, • Capability descr. • 21 capabilities	• Function group description • 15 groups	• Permission group description • 11 groups	







Permission assignment: timing

When are permissions assigned to a principal?

Symbian	Java ME	Android	MSSF
install-time assignment	run-time prompts	install-time assignment	install-time assignment, run-time privilege shedding possible

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

Application integrity

How is the integrity of installed applications protected?

Symbian	Java ME	Android	MSSF
dedicated directory	java sandboxing	java sandboxing, Linux access control	• IMA, Smack • Offline protection with EVM and TrEE

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

open

An credential platform that leverages on-board trusted execution environments

Secure yet inexpensive

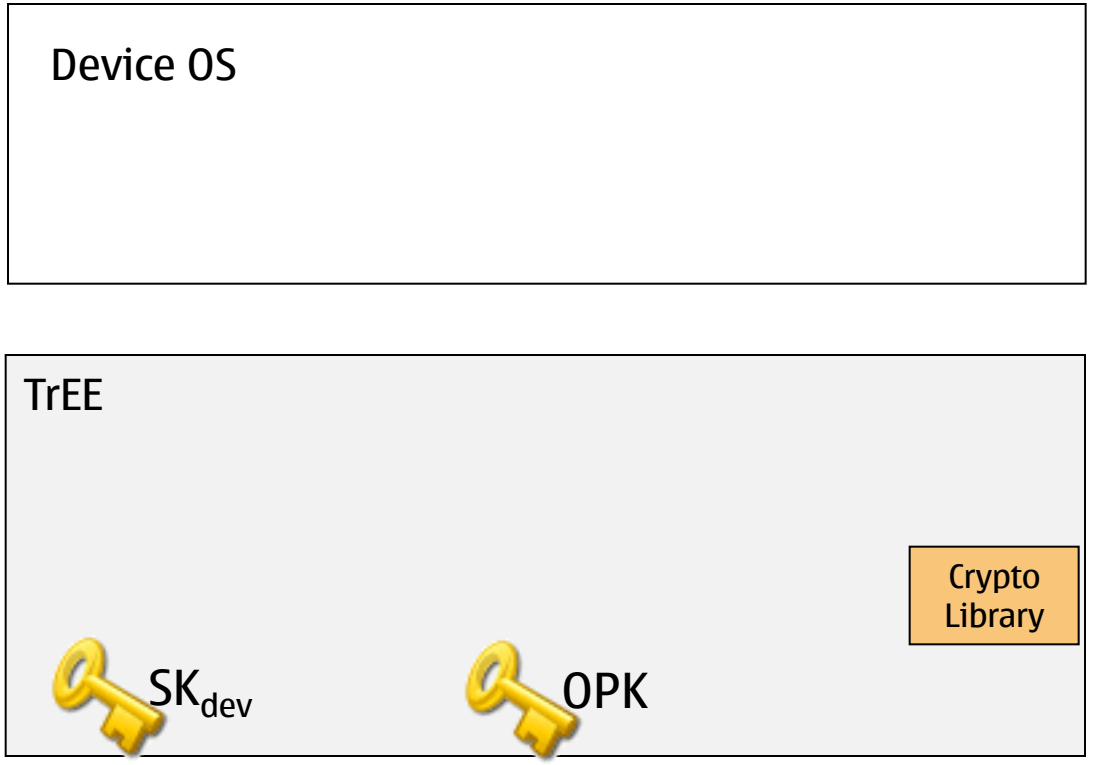
Open: dynamic protection domains

Skip to ObC summary

ObC Architecture

On Trusted Execution Environments (TrEEs) with

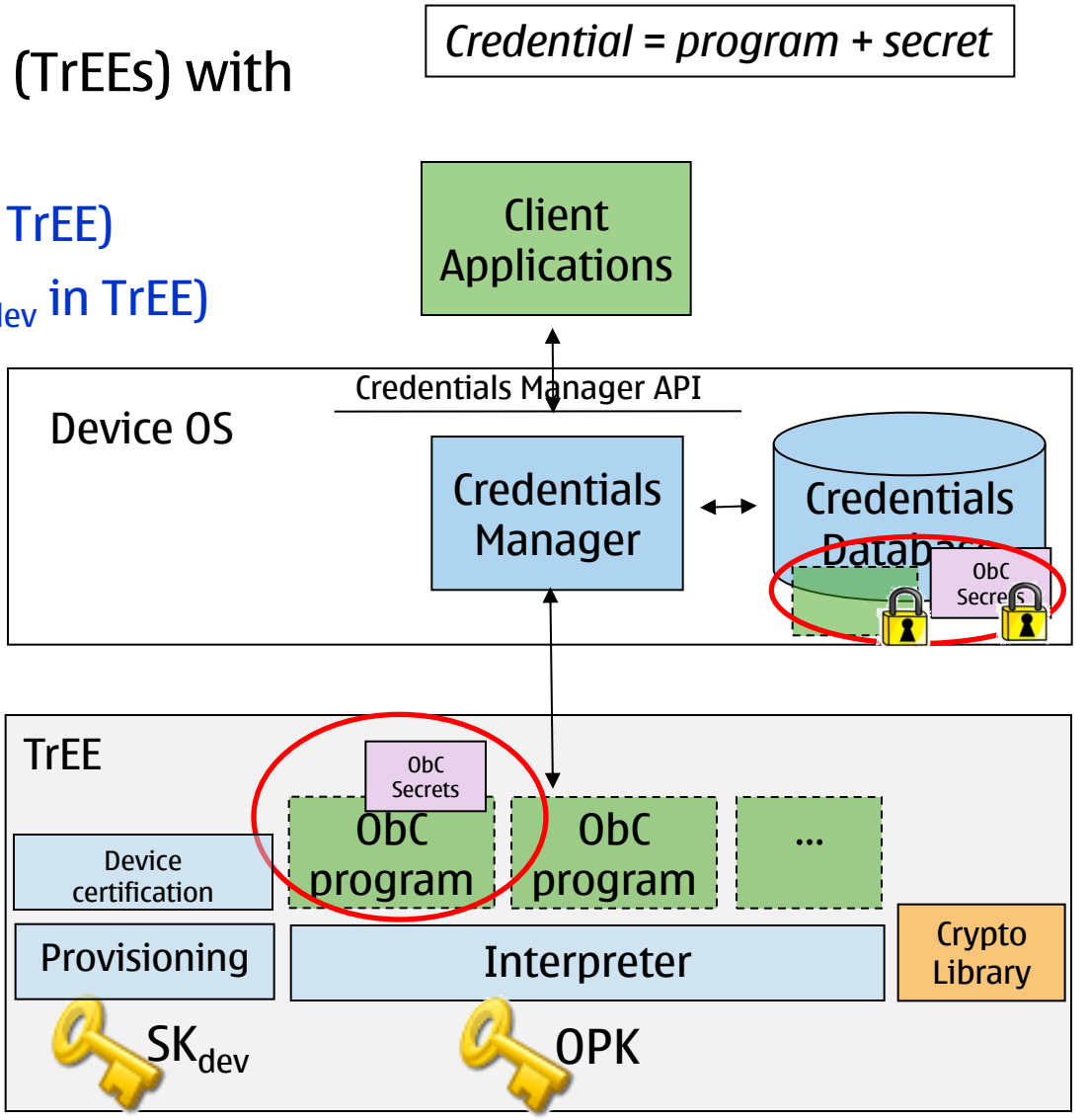
- Secure execution (within TrEE)
- Secure storage (secret key OPK in TrEE)
- Certified device keypair (PK_{dev}/SK_{dev} in TrEE)



ObC Architecture

On Trusted Execution Environments (TrEEs) with

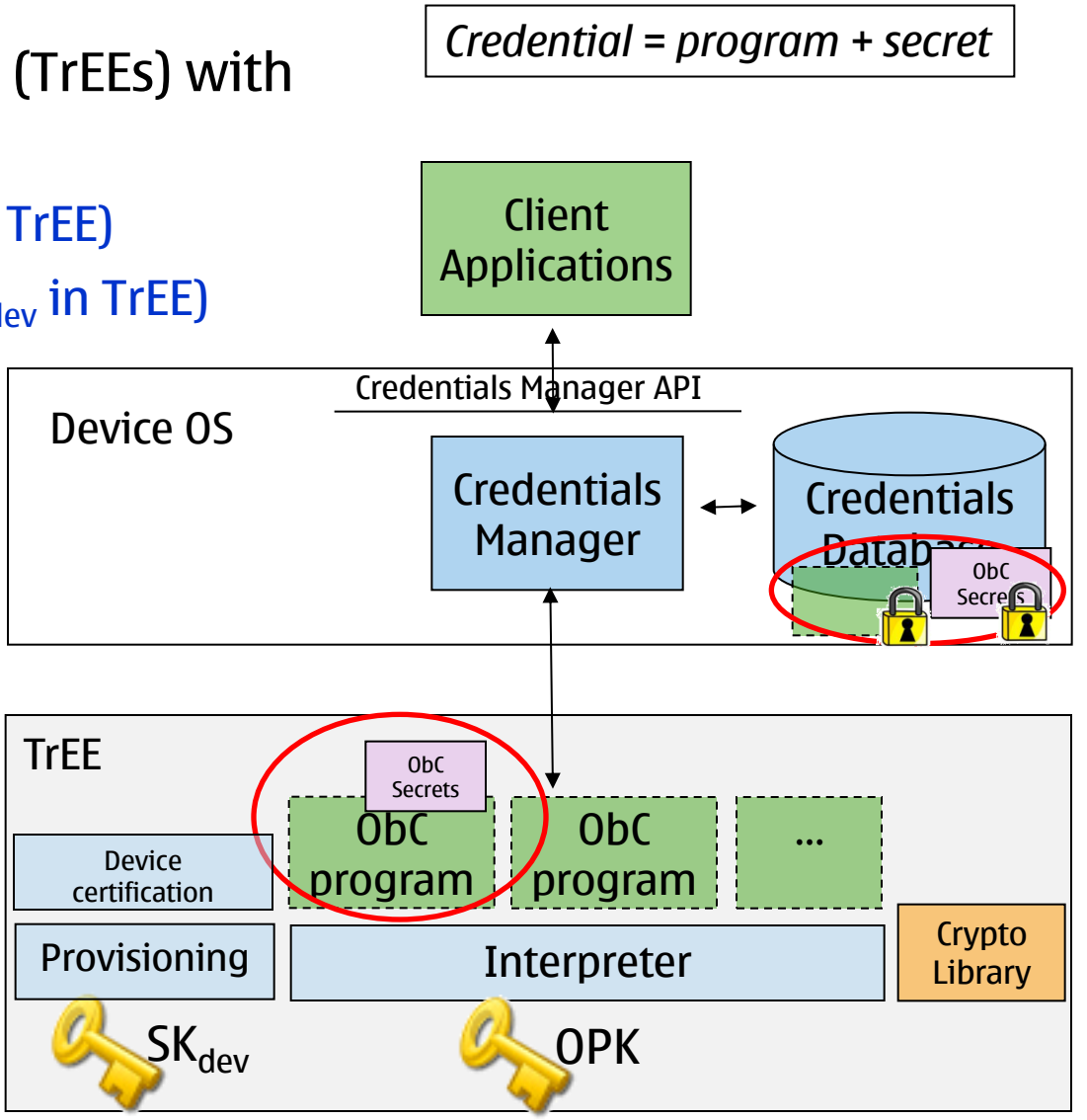
- Secure execution (within TrEE)
- Secure storage (secret key OPK in TrEE)
- Certified device keypair (PK_{dev}/Sk_{dev} in TrEE)



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)



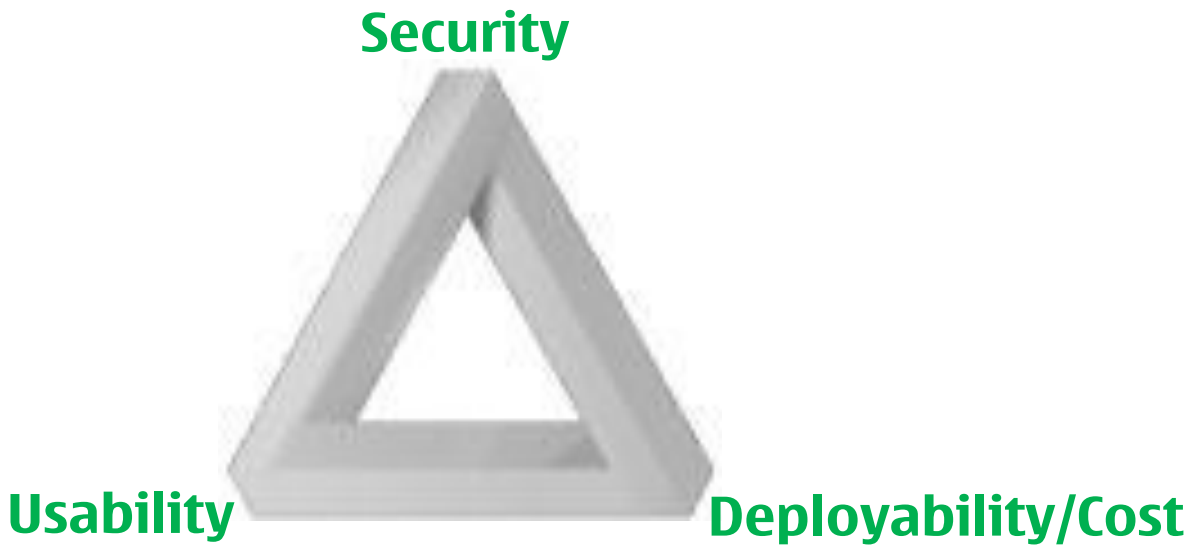
More in ACM ASIACCS '09 paper

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