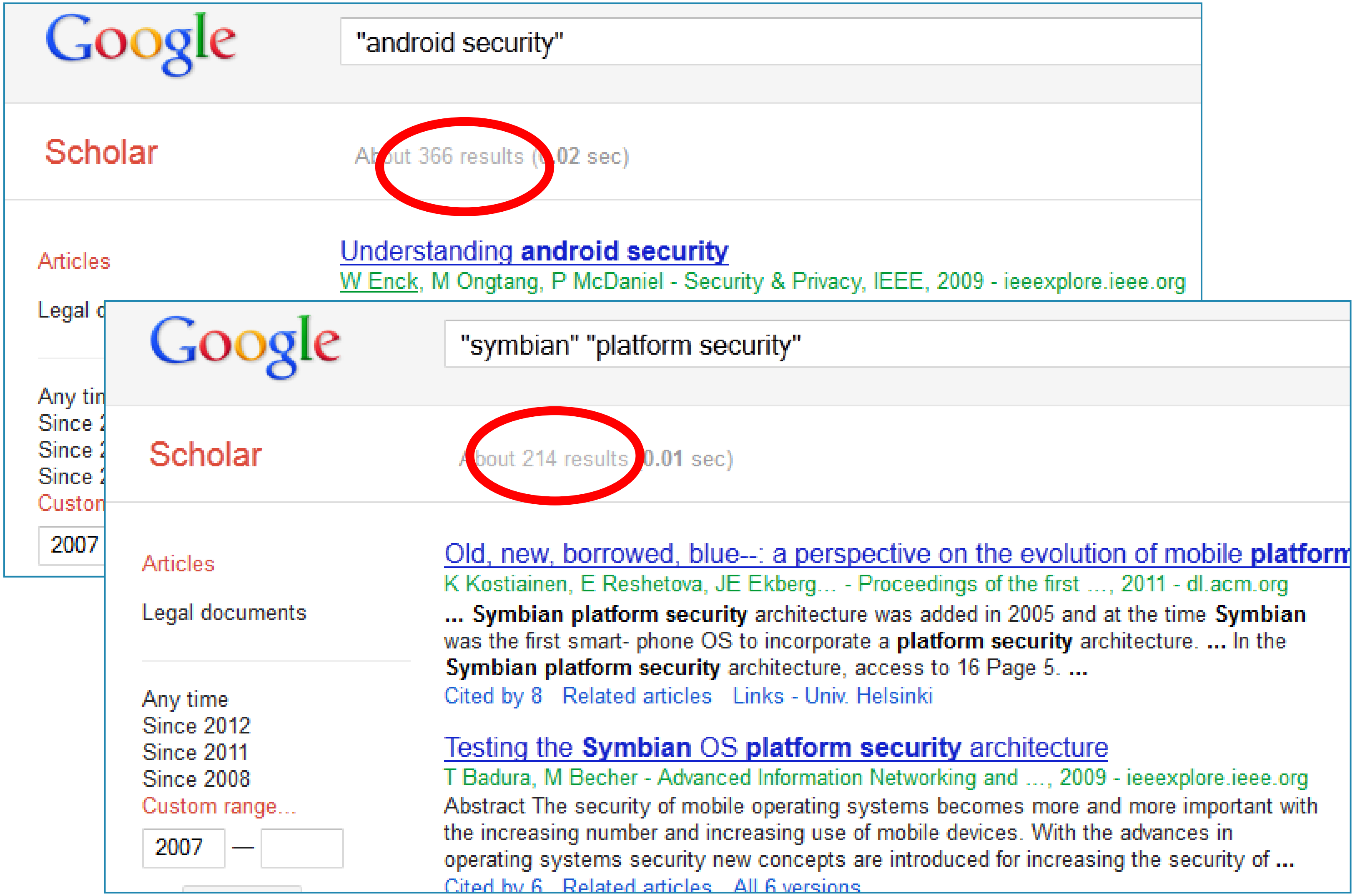
Mobile Platform Security Architectures

A perspective on their evolution

N. Asokan
Kari Kostiainen

Recent interest in smartphone security

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;

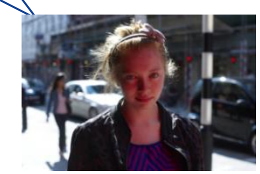
1. Subsidy locks → immutable ID
2. Copy protection → device authentication, app. separation
3. …

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 als
(where applicable) rem

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:

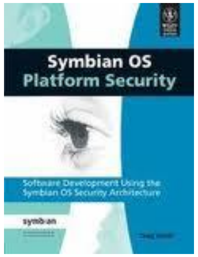
widespread use of hardware and software platform security

~2001 J2ME

~2002

TrustZone® Security Foundation by ARM®

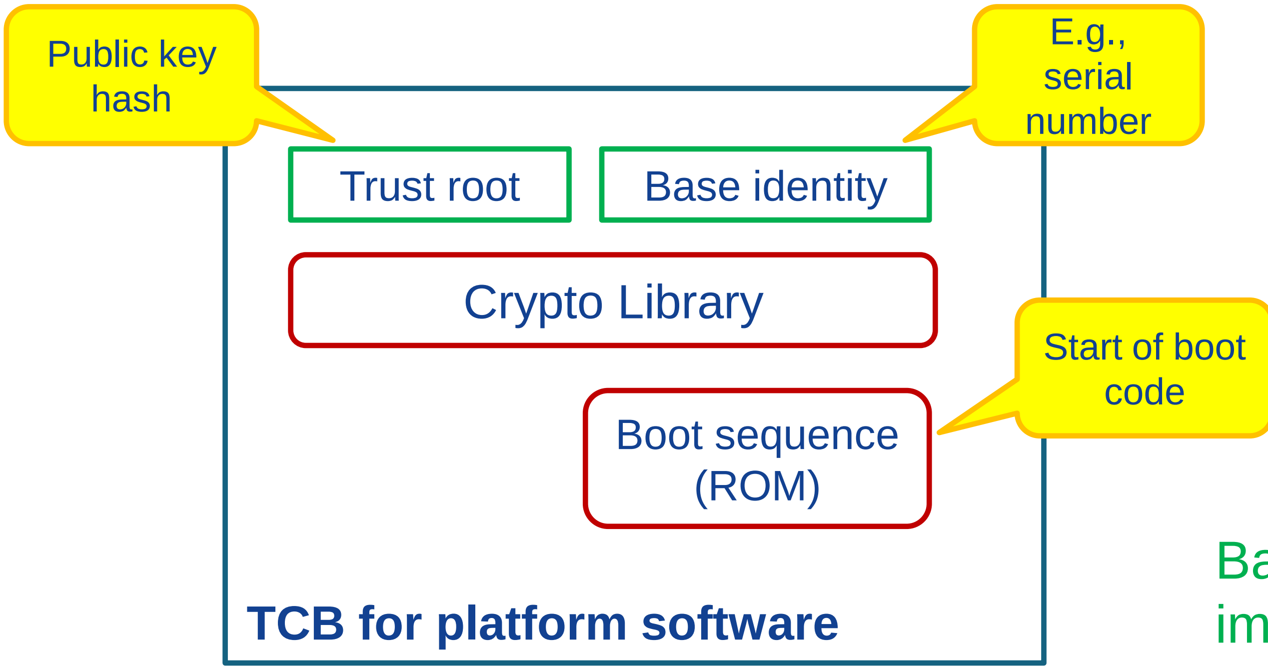
~2005



~2008

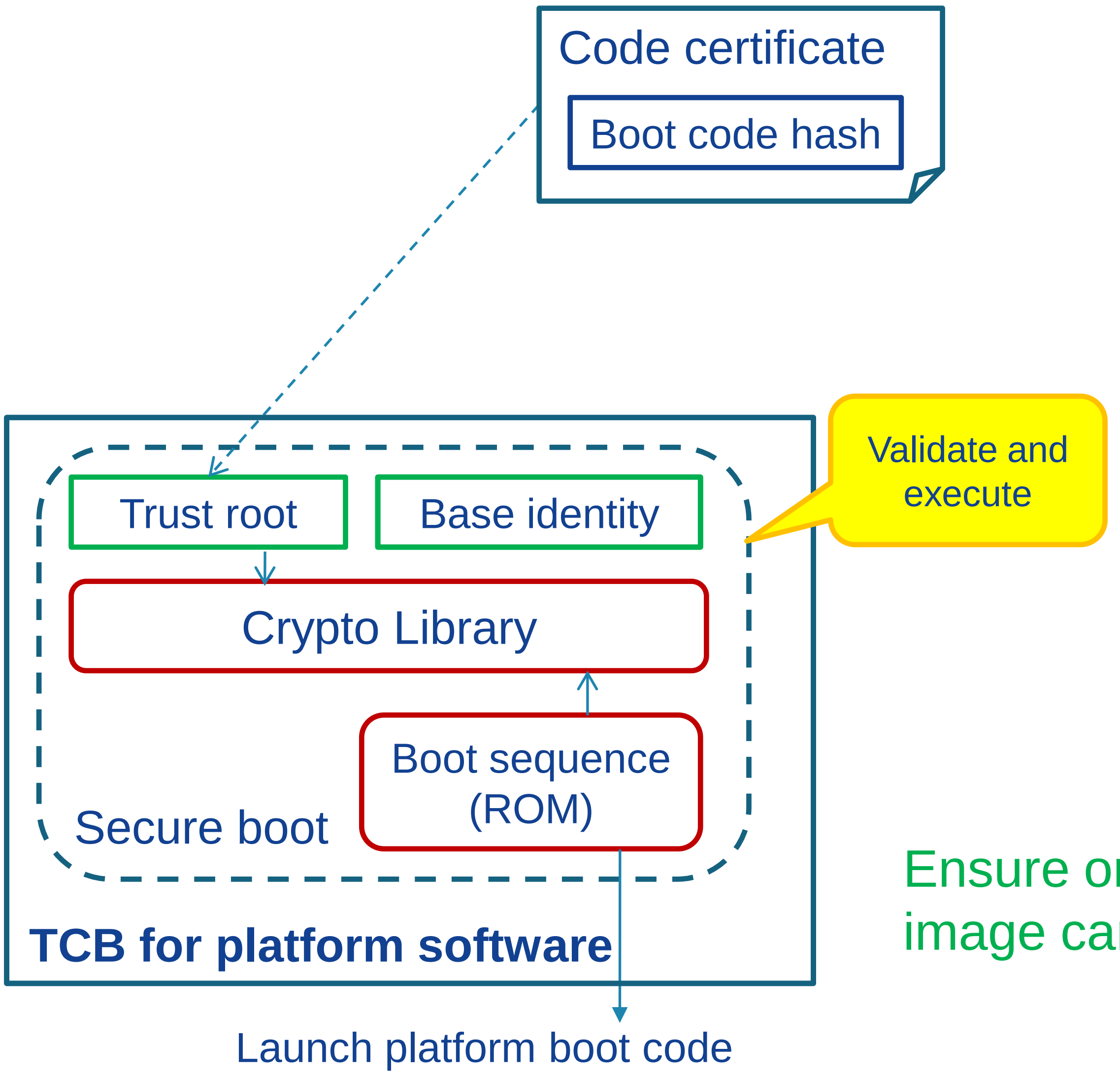
Hardware security enablers

Hardware support for platform security



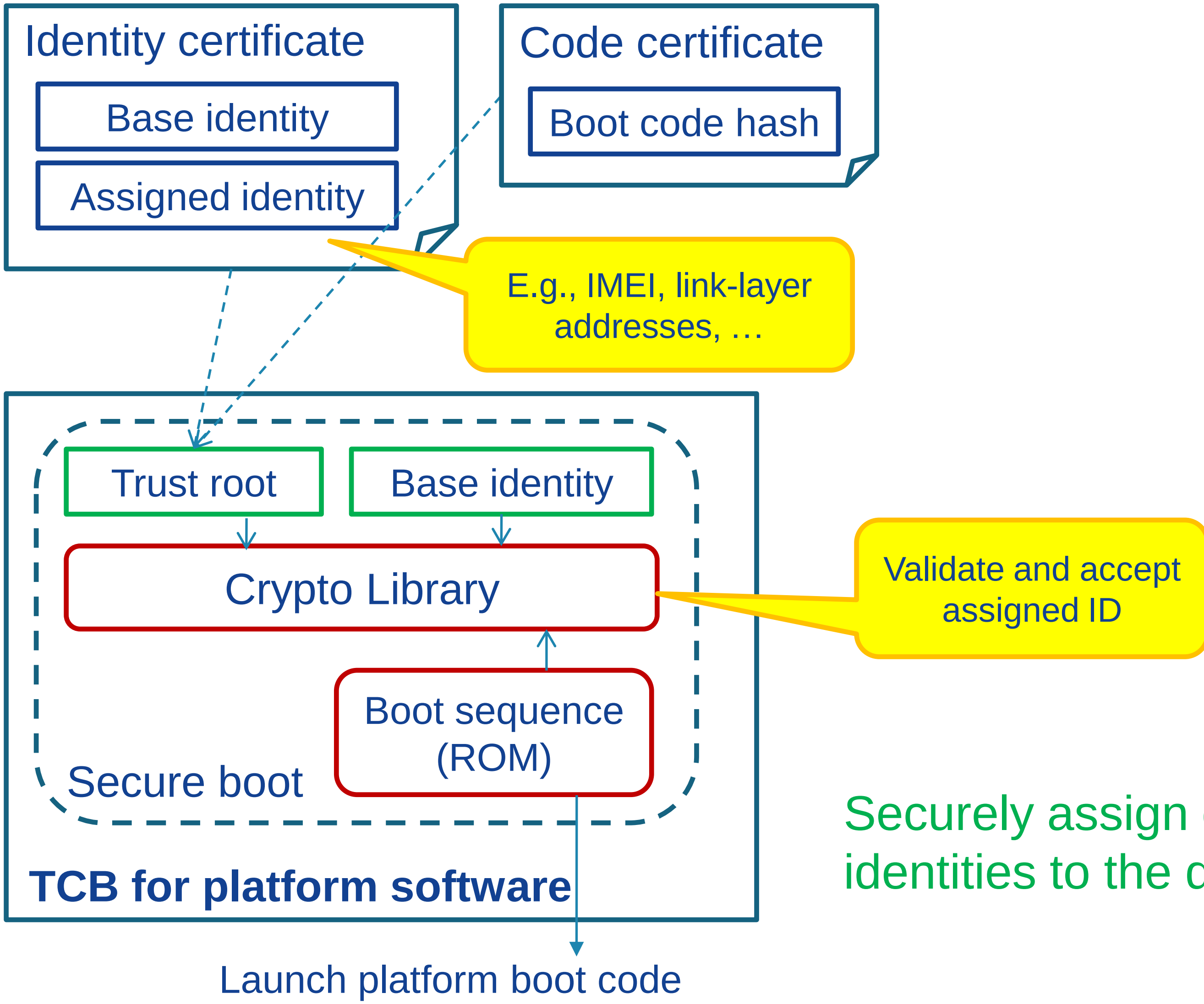
Basic elements in immutable storage

Secure bootstrapping



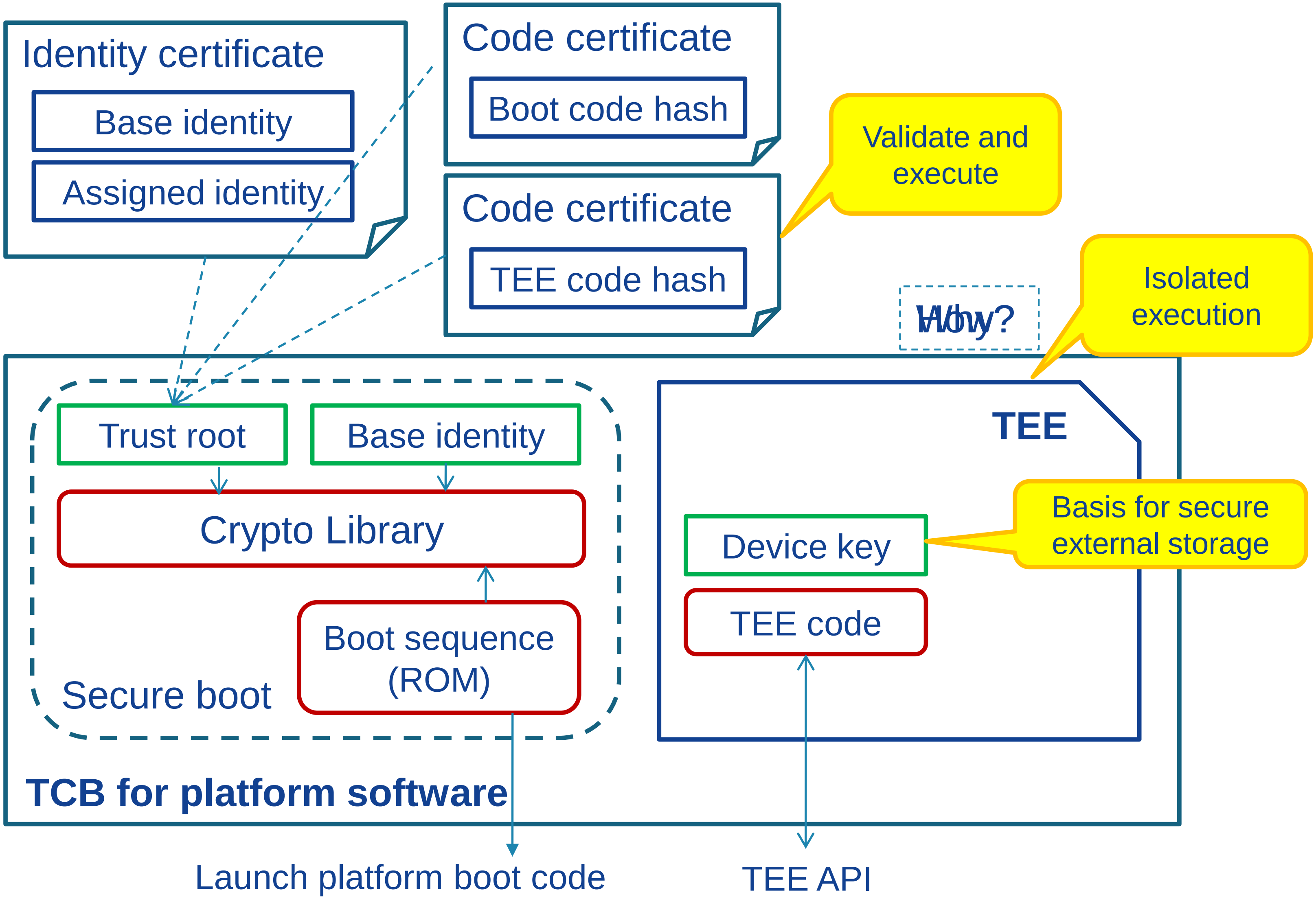
Ensure only authorized boot image can be loaded

Identity binding



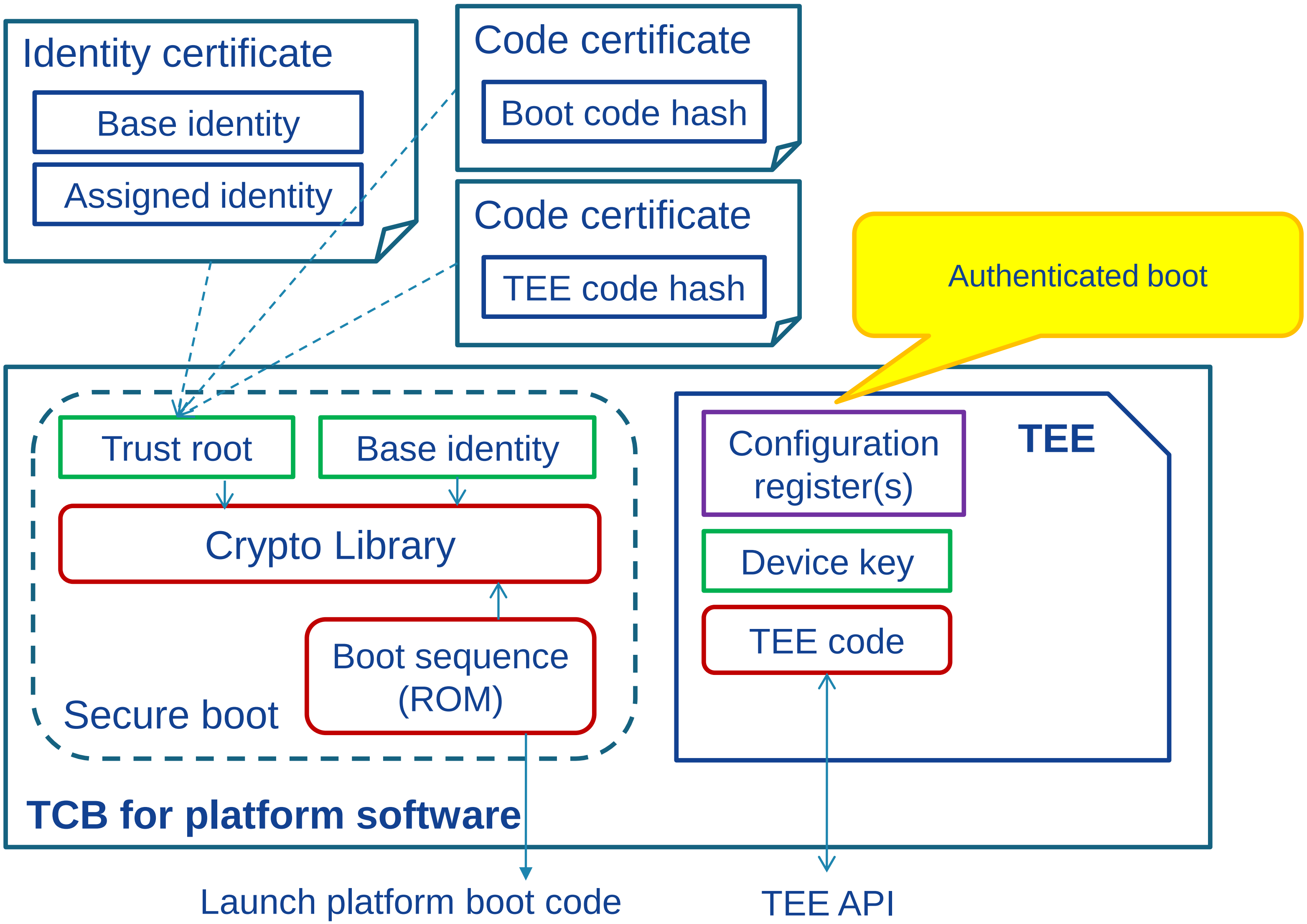
Securely assign different identities to the device

Trusted execution environment (TEE)

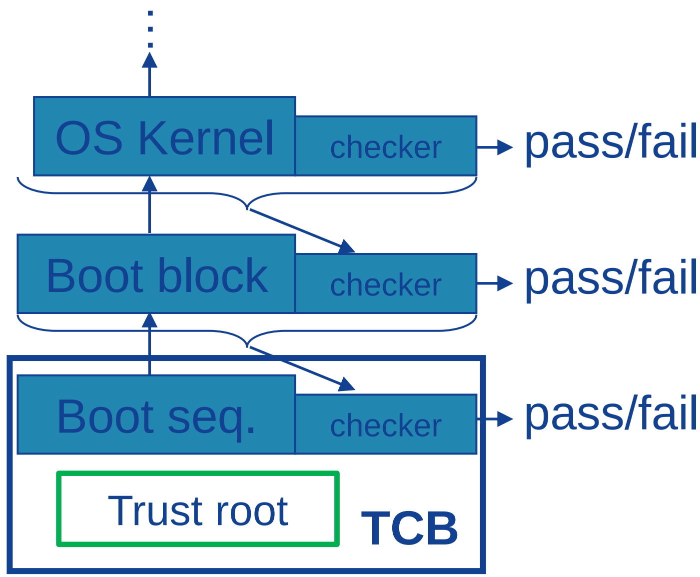


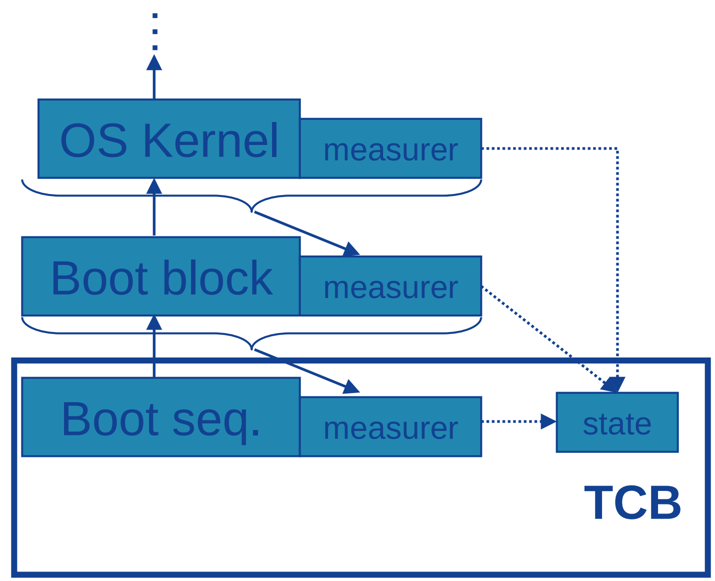
Authorized execution of arbitrary code, isolated from the OS; access to device key

Secure state



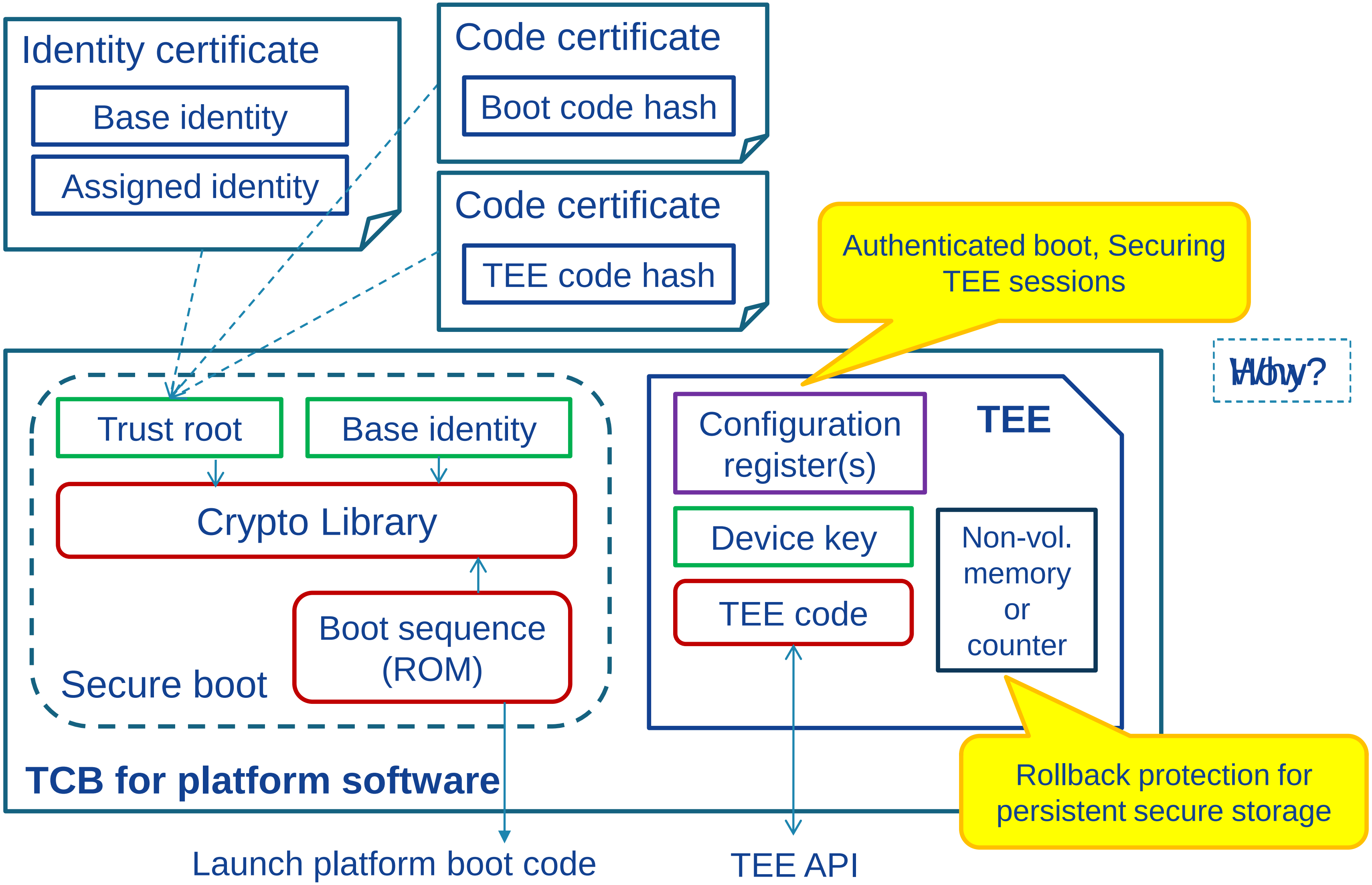
Secure boot vs Authenticated boot





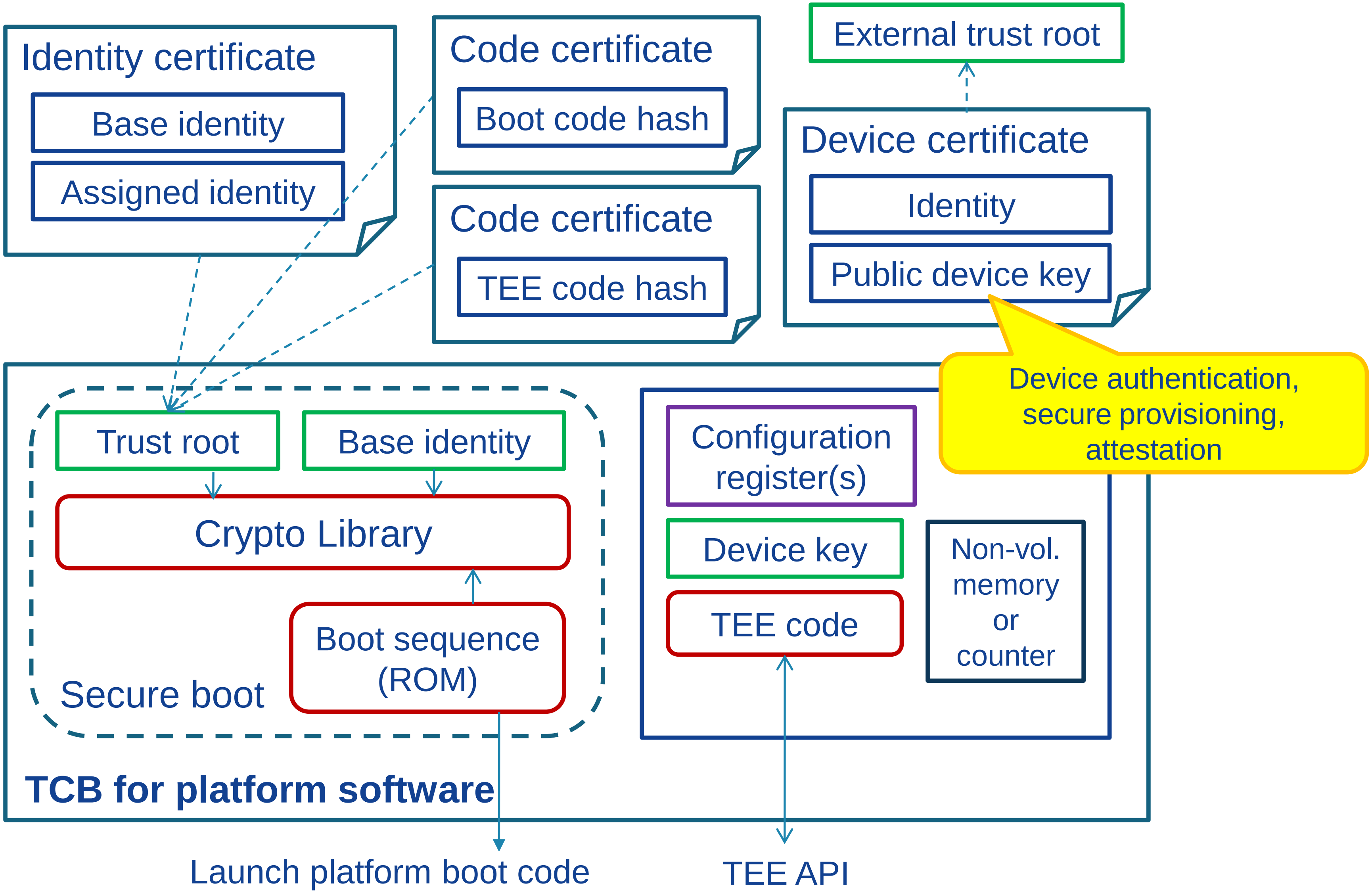
Root of Trust for measurement

Secure state



Integrity-protected state within the TEE

Device authentication

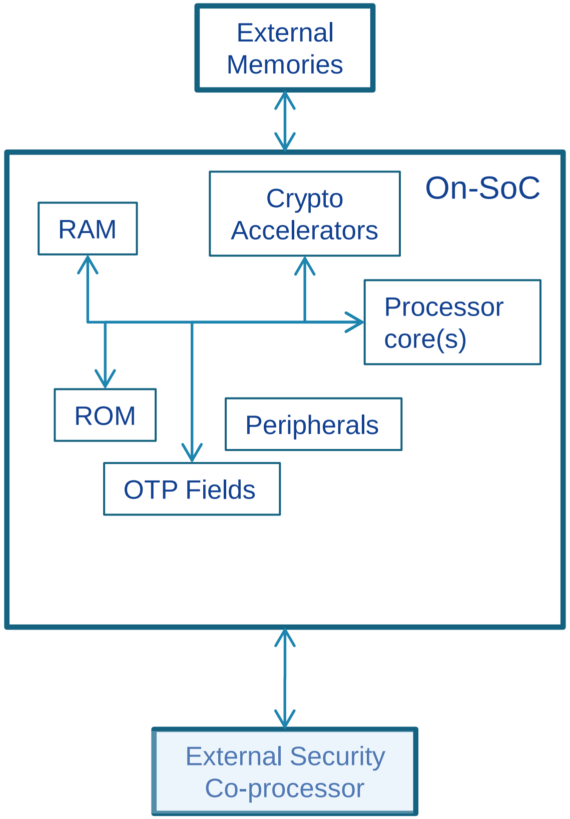


Prove device identity or properties to external verifier

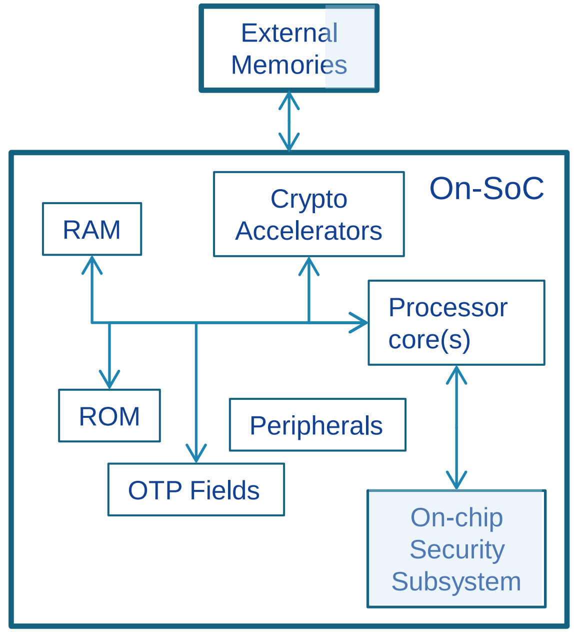
Hardware platform security features: summary

- **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

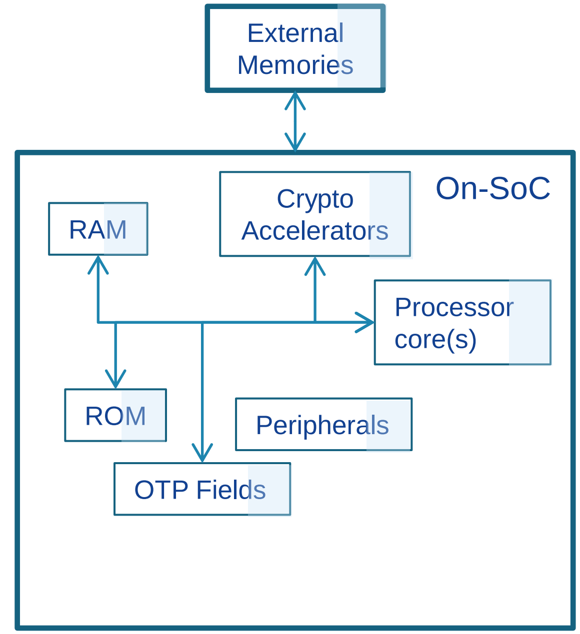
Architectural options for realizing TEEs



External Secure Element



Embedded Secure Element



Processor Secure Environment

TEE component

Figures taken from "GlobalPlatform Device Technology, TEE System Architecture", Version 1.0, December 2011

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

Processor Secure Environment

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

External Secure Element

- 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?**
  - an example in the second part of this seminar

Software platform security

Open mobile platforms

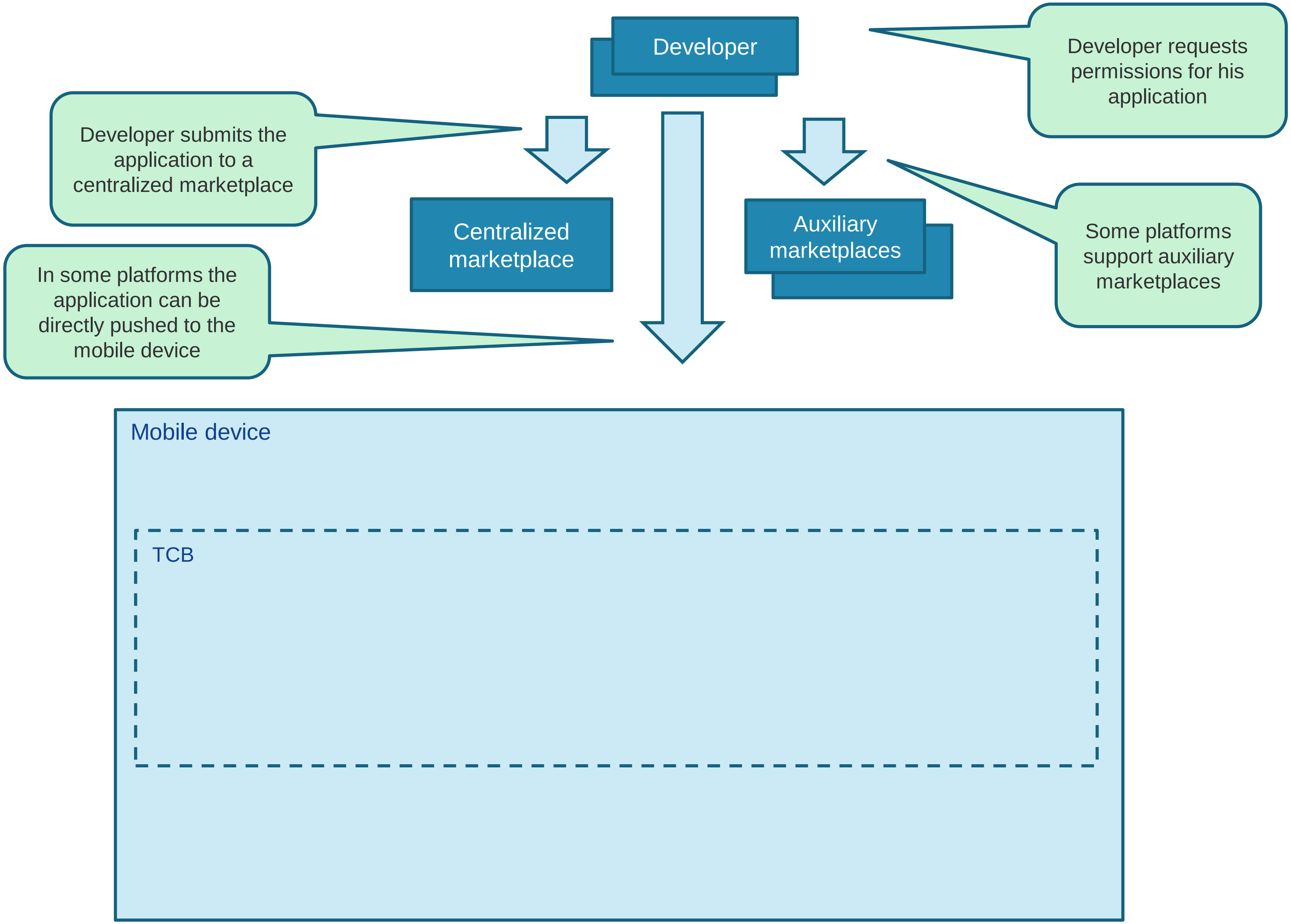
- Java ME ~2001
  - For "feature phones"
  - 3 billion devices!
  - Not supported by most smartphone platforms
- Symbian ~2004
  - First "smartphone" OS
  - App development in C++ (Qt)
- Android ~2007
  - Linux-based OS
  - App development in Java
- MeeGo ~2010
  - Linux-based OS
  - App development in C++ (Qt)
  - MSSF (Intel Tizen)
- Windows Phone ~2010
  - App development in .NET

Mobile platform security model

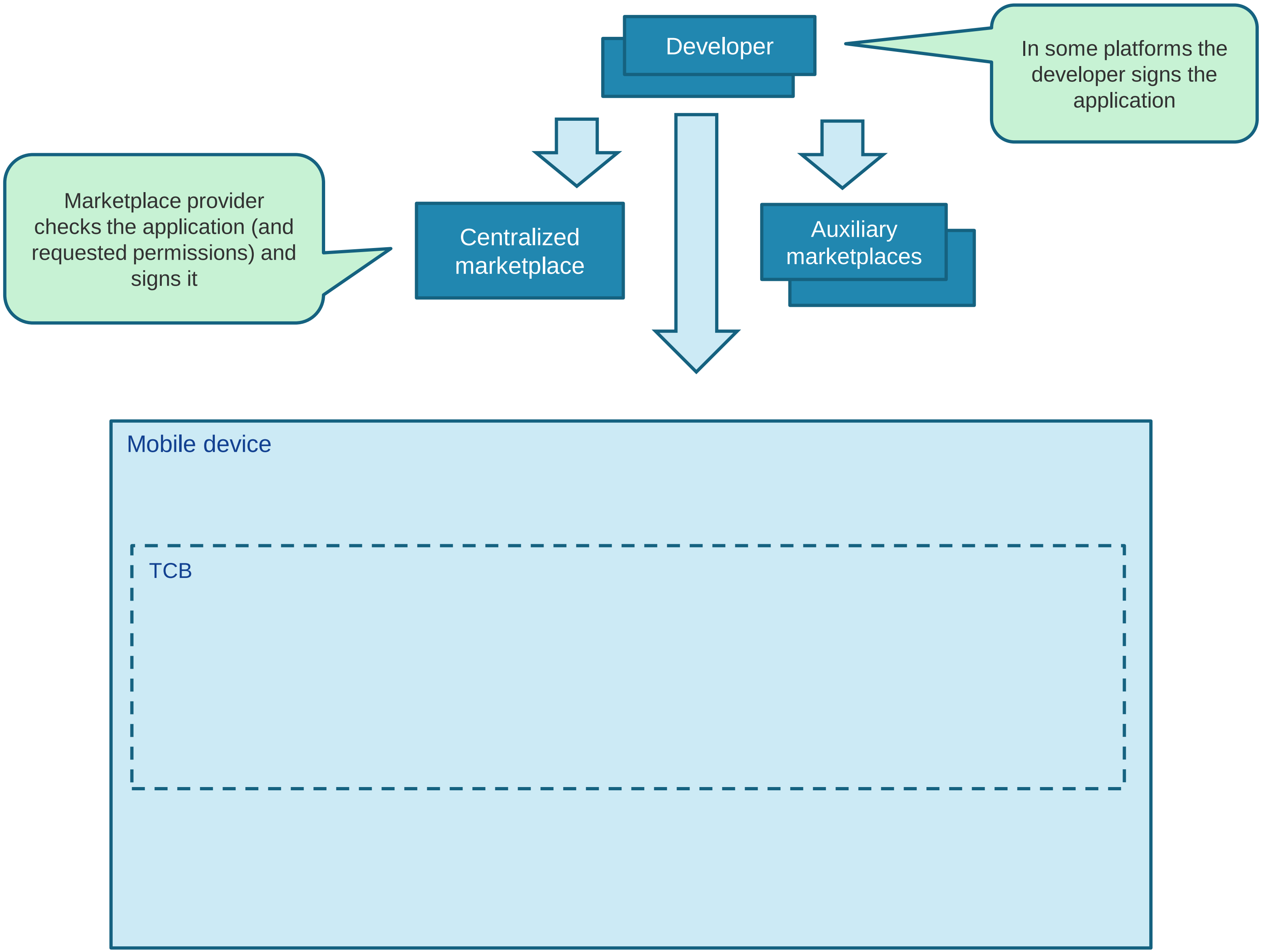
- Common techniques
  - Application signing
  - Permission-based access control architecture
  - Application isolation

- Common operations
  1. Permission request
  2. Application signing
  3. Application installation
  4. Application loading
  5. Run-time access control enforcement

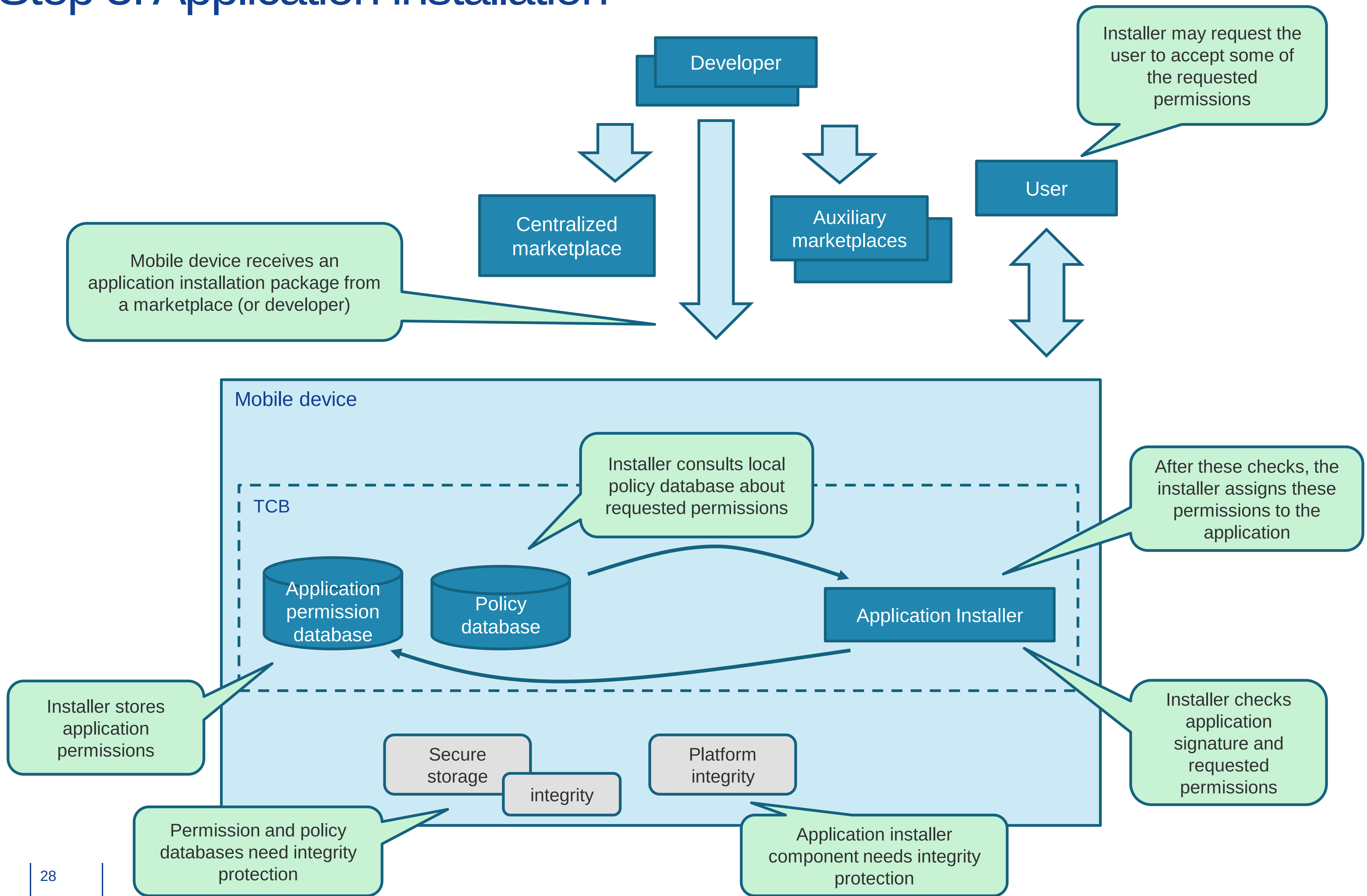
Step 1: Developer publishes an application



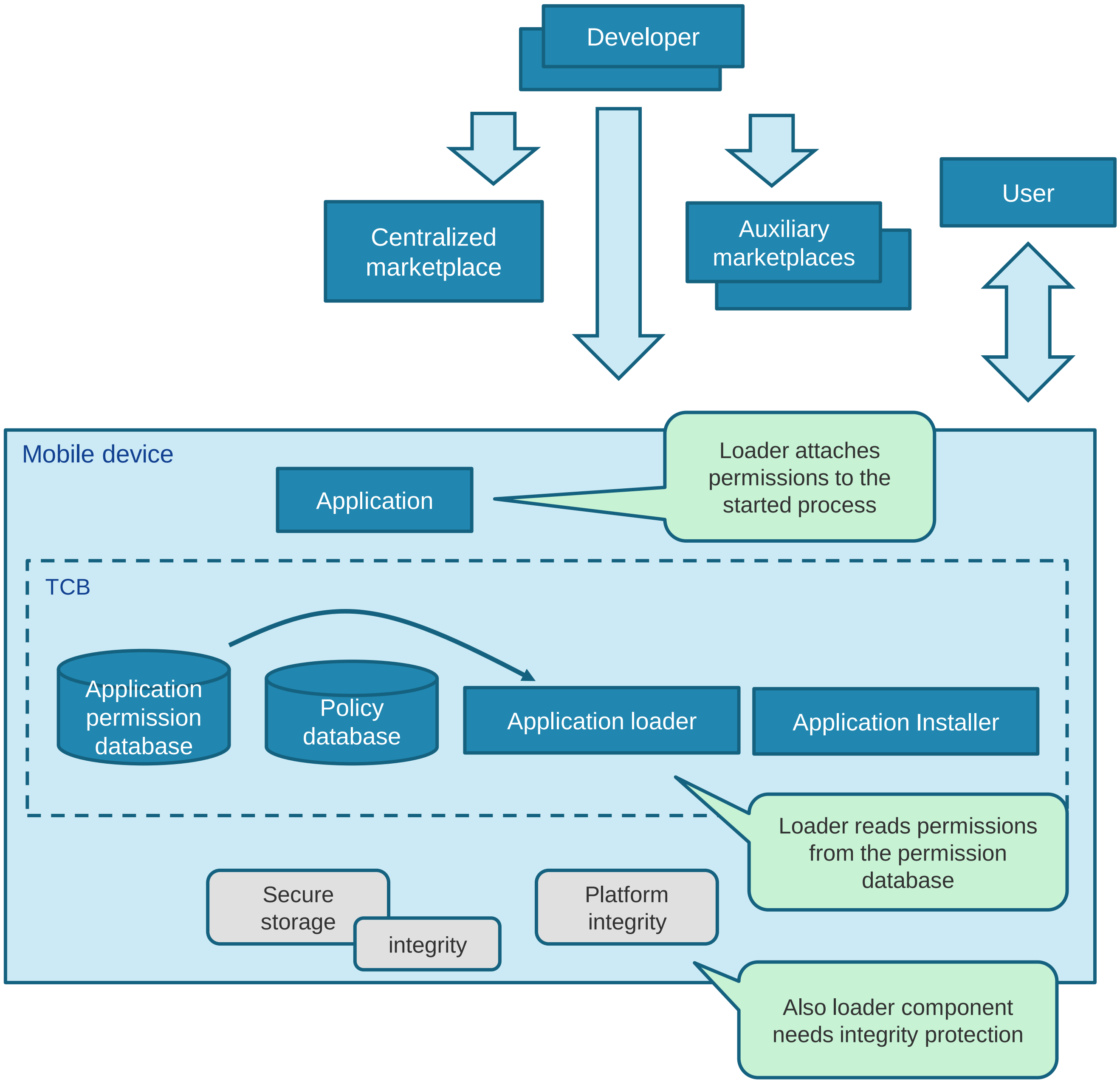
Step 2: Marketplace signs the application



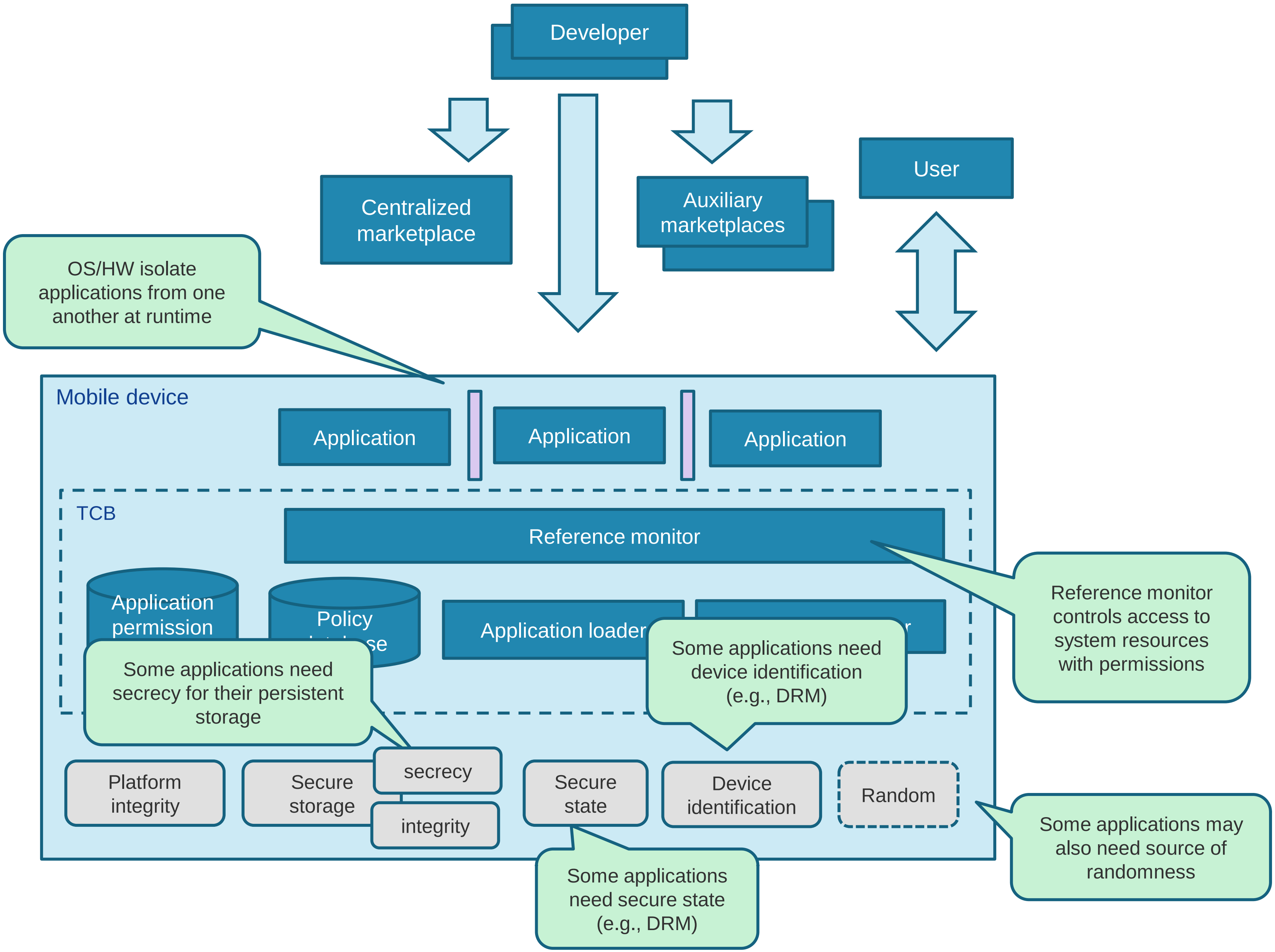
Step 3: Application installation



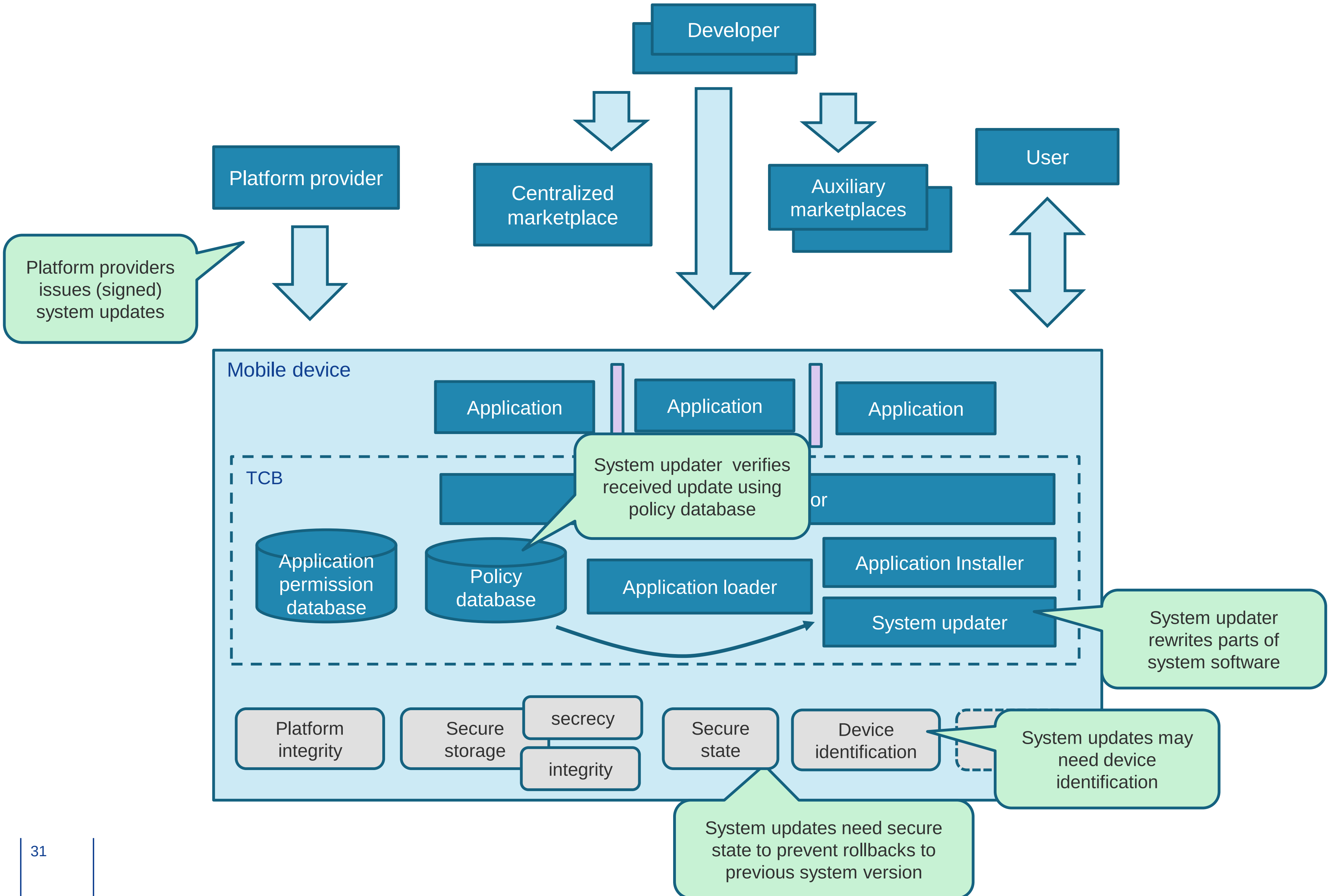
Step 4: Application loading



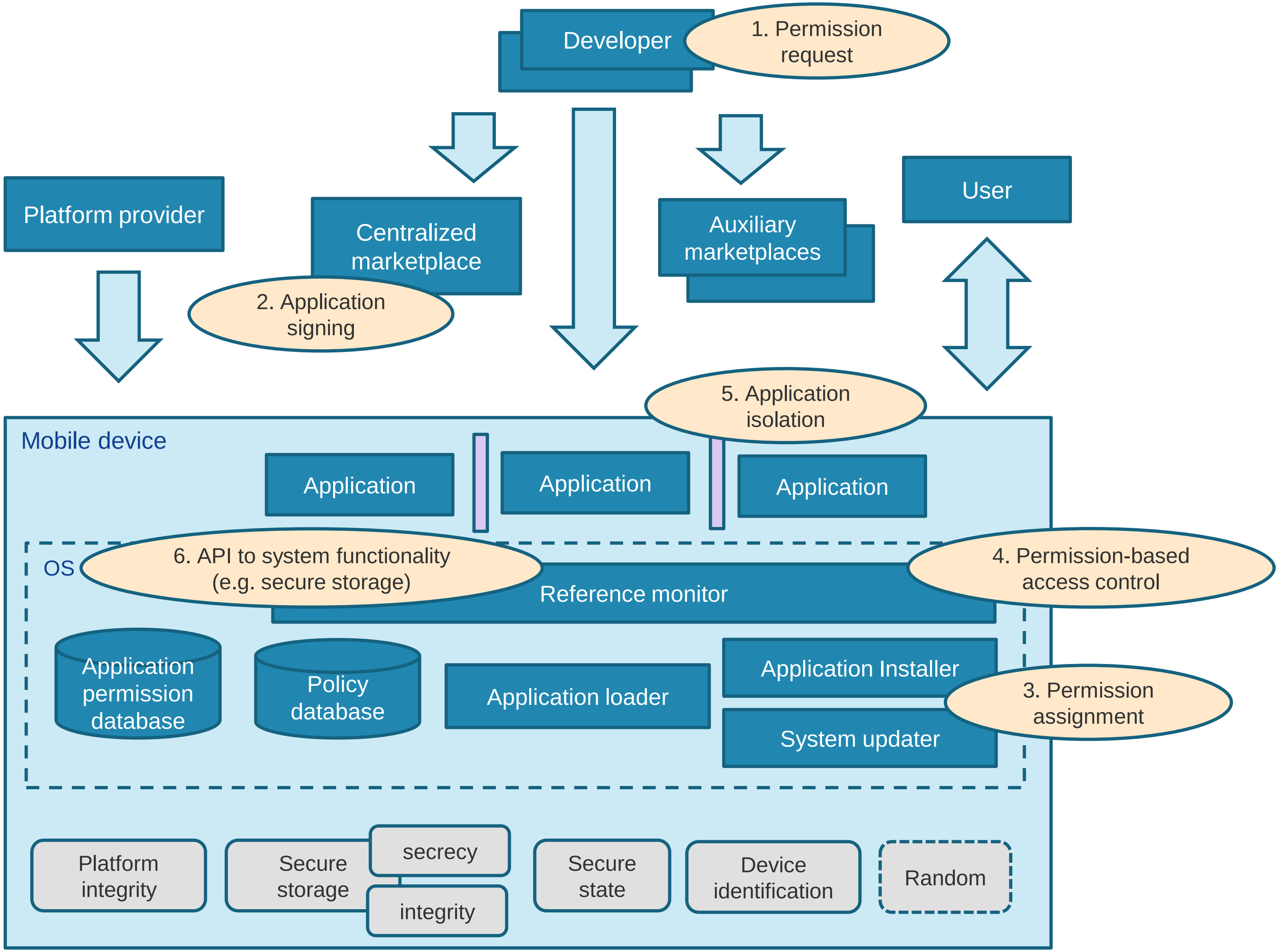
Step 5: Application execution



Step 6: System updates



Recap – main techniques



Software platform security design choices

Device boot
- How is platform integrity verified?

Application development and installation
- How finely are access control policies defined?
- What is the basis for granting permissions?

Application installation
- What is shown to the user?

Application runtime
- How is the integrity of installed applications protected?
- How can applications protect the confidentiality and integrity of their data?

Application updates
- How is a new version of an existing application verified?

OS bootstrapping

Is hardware security used to secure OS bootstrapping?

Symbian	Java ME	Android	MSSF	Windows
Secure boot	Not applicable	[illegible]	Authenticated boot: “Normal mode” vs “Developer mode”	

Permission granularity

How finely is access control defined?

Symbian	Java ME	Android	MSSF	Windows Phone
Fixed set of "capabilities" (21)	Fine-grained permissions (many)		Fine-grained resource-tokens Linux access control	Fixed set of "capabilities" (16)

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

Permission assignment (basis)

What is the basis for granting permissions?

Symbian	Java ME	Android	MSSF	Windows Phone
4 categories Trusted signature (also user prompts)	Trusted signatures for protection domains 4 permission modes	4 protection levels	Trusted signatures Local policy file	Trusted signatures (user prompt for location)

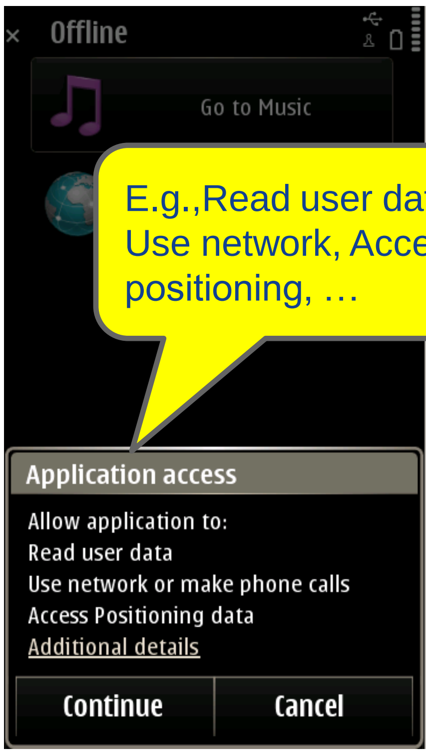
User
System,
Restricted,
Manufacturer

Blanket,
Session,
One-shot,
No

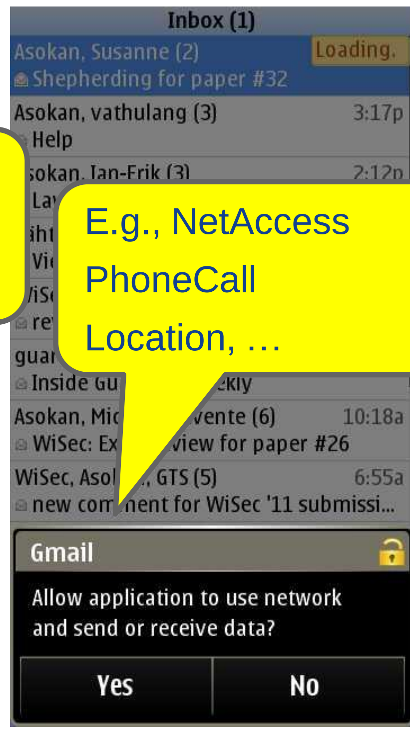
Normal (automatic)
Dangerous (user-granted)
Signature (developer-controlled)
SystemOrSignature (Google-controlled)

Permission assignment (user prompting)

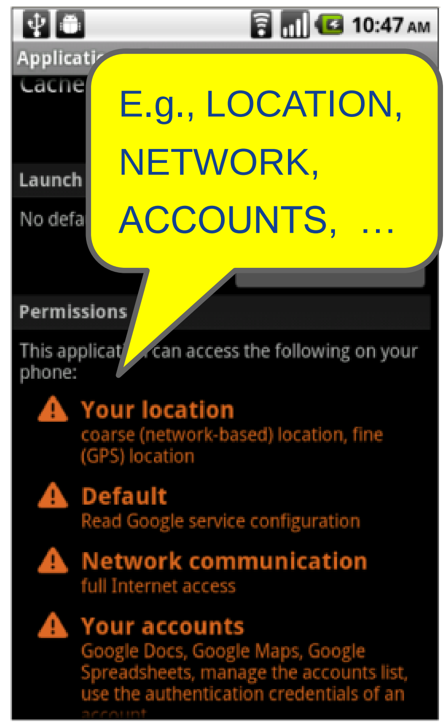
Symbian	Java ME	Android	Windows Phone
Capability description • 21 capabilities	Function group description • 15 groups	Permission group description • 11 groups	User prompted only for location capability



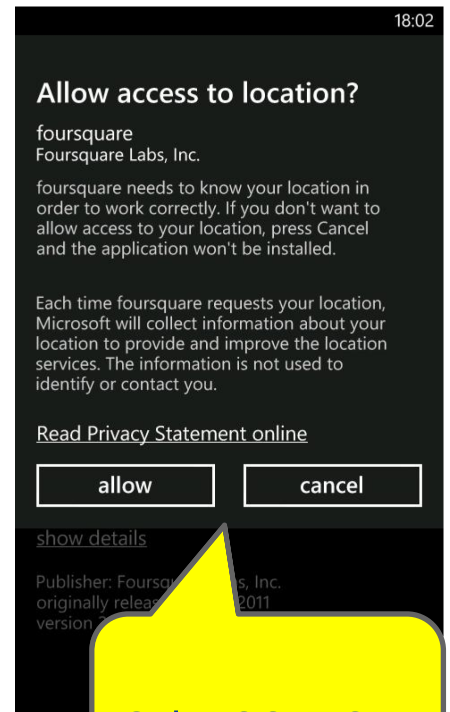
E.g.,Read user data, Use network, Access positioning, …



E.g., NetAccess
PhoneCall
Location, …



E.g., LOCATION, NETWORK, ACCOUNTS, …



Only LOCATION

What is shown to the user?

Skip to “Application Updates”

Permission assignment (timing)

When are permissions assigned to a principal?

Symbian	Java ME	Android	MSSF	Windows
Install-time assignment	Run-time prompts	assignment	Install-time assignment Run-time privilege shedding possible	assignment

Symbian and MSSF: Permissions of app loading a DLL is a subset of permissions of DLL

Access control policy

How does a resource declare the policy for accessing it?

How is it enforced?

Symbian	Java ME	Android	MSSF	Windows Phone
Declare in code Enforced by IPC framework or code	[System resources] Enforced by VM	Declare in manifest Enforced by VM	Declare in manifest Enforced by Smack or via libcreds	[System resources] Enforced by VM

Application identification

How are applications identified at install and runtime?

Symbian	Java ME	Android	MSSF	Windows Phone
Install and run-time: • Protected range SID and VID (managed) • UID (unmanaged)	Install: • Signing key • Midlet attributes	Install: • Signing key Runtime: • Unix UID • Package name (locally unique)	Install: • Software source (signing key) • Package name Runtime: • Software source • Package name • Application ID	Install and run-time: • Unique ID (assigned by marketplace)

Application integrity

How is the integrity of installed applications protected?

Symbian	Java ME	Android	MSSF	Windows Phone
Dedicated directory	Java sandboxing	Java sandboxing Linux access control	IMA, Smack Offline protection with EVM and TEE	.NET sandboxing

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

Application update

How is a new version of an existing application verified?

Symbian	Java ME	Android	MSSF	Windows Phone
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	Trusted signature

Application data protection

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

Symbian	Java ME	Android	MSSF	Windows Phone
Runtime: • private directory Off-line: • private secure storage	Runtime: • private record stores	Runtime: • dedicated UID • file system	Runtime: • fine-grained data caging Off-line: • private secure storage	Runtime: • private directory

Discussion

Discussion

Recurring themes (hardware enablers)

- Hardware-support for platform security
  - Cambridge CAP etc. (~1970's)
  - → Extended to Processor Secure 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/process isolation

Open issues

- Permission granularity
  - Coarse-grained permissions vs. principle of least privilege
  - Fine-grained permissions vs. user/developer confusion [Felt et al, CCS '12]
- Permission assignment
  - Is it sensible to let end users make policy assignment decisions? [Chia et al, WWW '12] [Felt et al, SOUPS '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] [Bugiel et al, NDSS '12]

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.