Open-TEE Tutorial

An open virtual Trusted Execution Environment

Brian McGillion, Tanel Dettenborn
Intel
N. Asokan
Aalto University and University of Helsinki

What is a TEE?

Execution Environment

What is a TEE?

Processor, memory, storage, peripherals

Trusted Execution Environment

Isolated and integrity-protected

From the "normal" execution environment (Rich Execution Environment)

Chances are that:

You have devices with hardware-based TEEs in them!

But you probably don't have (m)any apps using them

Mobile Trusted Computing. Proceedings of the IEEE 102(8): 1189-1206 (2014)
The Untapped Potential of Trusted Execution Environments on Mobile Devices. IEEE Security & Privacy 12(4): 29-37 (2014)

Outline

- Introduction - Asokan
  - Why do mobile devices have TEEs?
  - What constitutes a TEE?
  - Mobile hardware security APIs

- Nuts and Bolts of Open-TEE - Brian

Why do most mobile devices today have TEEs?

A LOOK BACK

Platform security for mobile devices

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 PCs

Early adoption of platform security

Both IMSI and IMEI require physical protection.

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 ens
(where applicable) remains

GSM 02.09, 1993

The IMSI is stored securely within the SIM.

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.

3GPP TS 42.009, 2001

Different starting points compared to PCs:
Widespread use of hardware and software platform security

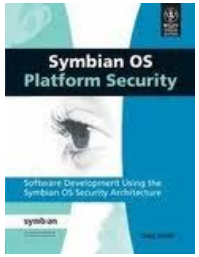
~2001

J2ME

~2002

TrustZone®
Security Foundation by ARM®

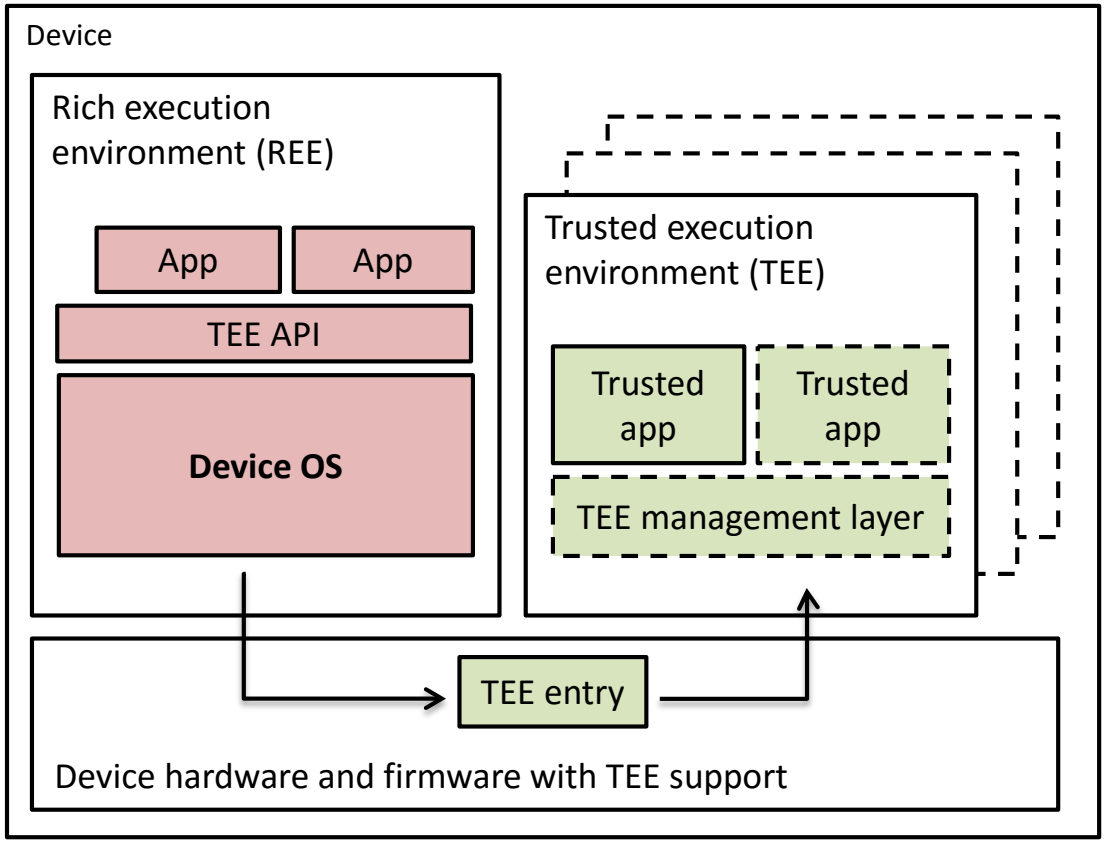
~2005



~2008

Old, new, borrowed, blue --: a perspective on the evolution of mobile platform security architectures. CODASPY 2011: 13-24

TEE system architecture



Architectures with single TEE

- ARM TrustZone
- TI M-Shield
- Smart card
- Crypto co-processor
- Trusted Platform Module (TPM)

Architectures with multiple TEEs

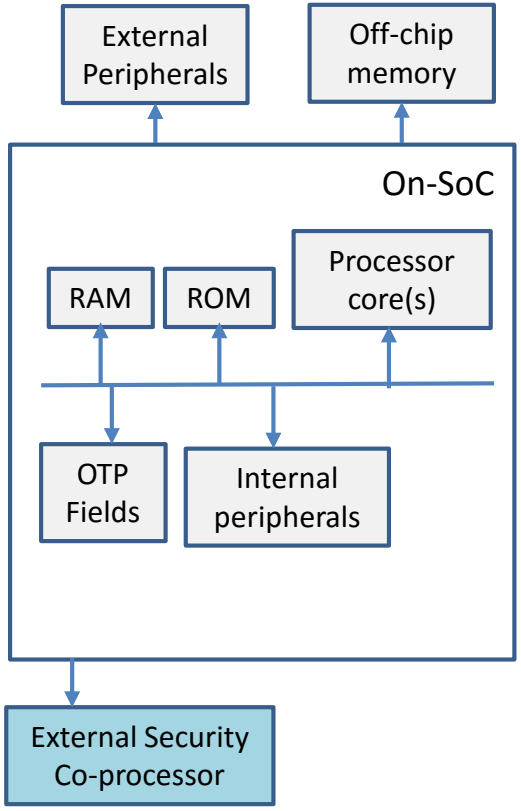
- Intel SGX
- TPM (and "Late Launch")
- Hypervisor

Figure adapted from: Global Platform. TEE system architecture. 2011.

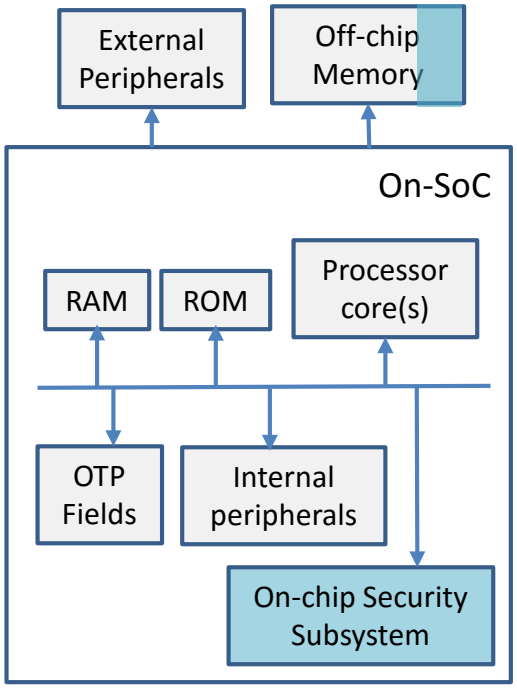
Legend:
SoC : system-on-chip
OTP: one-time programmable

TEE hardware realization alternatives

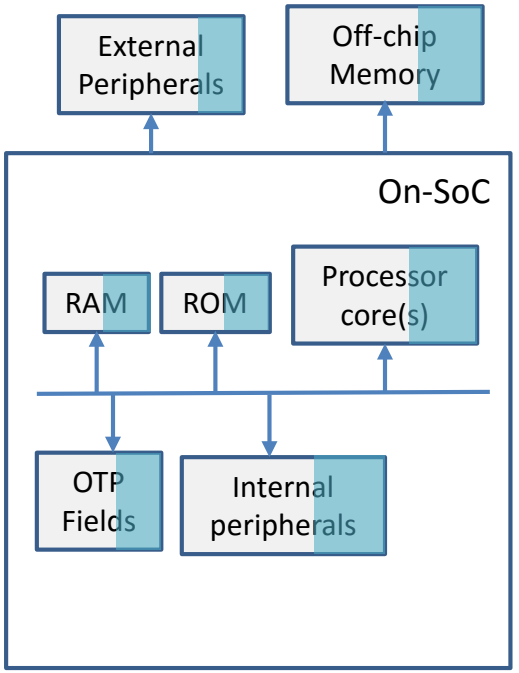
TEE component



External Secure Element
(TPM, smart card)



Embedded Secure Element
(smart card)

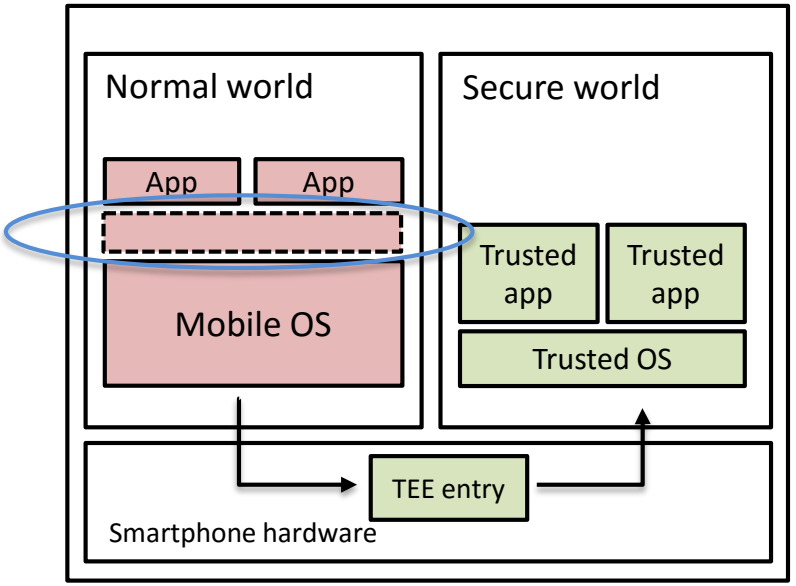


Processor Secure Environment
(TrustZone, M-Shield)

Figure adapted from: Global Platform. [TEE system architecture](). 2011.

Mobile TEE deployment

- TrustZone support available in majority of current smartphones
- Mainly used for manufacturer internal purposes
  - Digital rights management, Subsidy lock…
- *APIs for developers?*



Android Key Store API

Android Key Store example

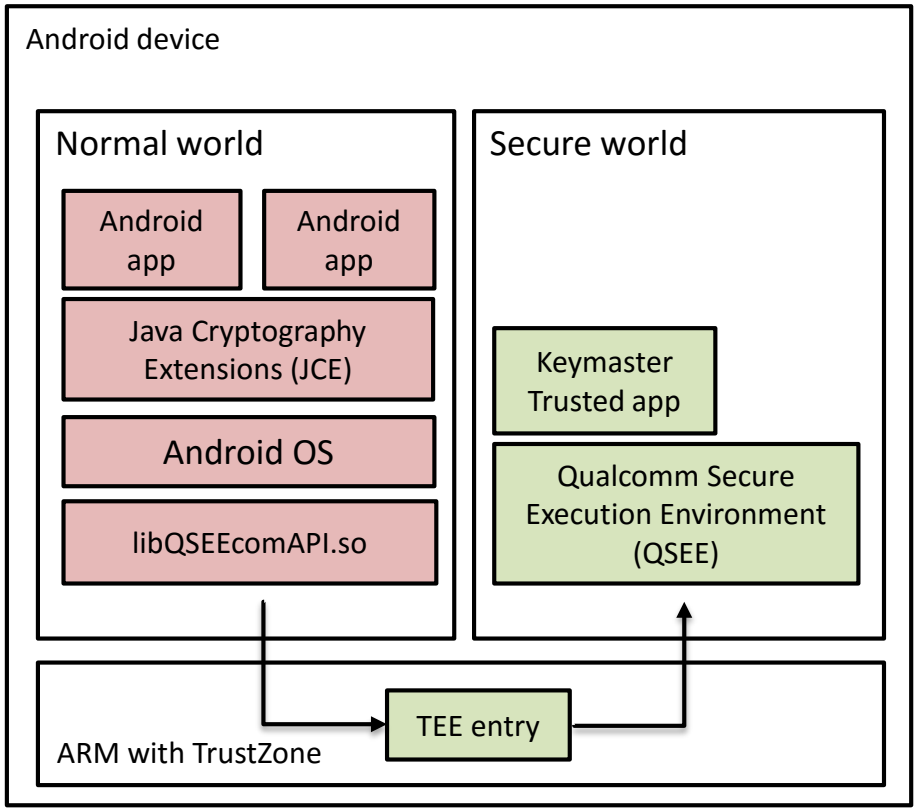
```
// create RSA key pair
Context ctx;
KeyPairGeneratorSpec spec = new KeyPairGeneratorSpec.Builder(ctx);
spec.setAlias(”key1")
…
spec.build();

KeyPairGenerator gen = KeyPairGenerator.getInstance("RSA", "AndroidKeyStore");
gen.initialize(spec);
KeyPair kp = gen.generateKeyPair();


// use private key for signing
AndroidRsaEngine rsa = new AndroidRsaEngine("key1", true);
PSSSigner signer = new PSSSigner(rsa, …);
signer.init(true, …);
signer.update(signedData, 0, signedData.length);
byte[] signature = signer.generateSignature();
```

Elenkov. Credential storage enhancements in Android 4.3. 2013.
Elenkov, Keystore redesign in Android M, 2015.

Key Store implementation: example



Keymaster operations

- Public key algorithms
- Symmetric key algorithms (AES, HMAC) from v1.0
- Access control, key usage restrictions

Persistent storage on Normal World

Elenkov. Credential storage enhancements in Android 4.3. 2013.
Android, Hardware-backed Keystore, 2015

Android Key Store

- Available operations
  - Signatures
  - Encryption/decryption
- Developers cannot utilize programmability of mobile TEEs
  - Not possible to run arbitrary trusted applications

- Global Platform is standardizing TEE APIs

- Different API abstraction and architecture needed…
  - Example: On-board Credentials

Open-TEE

- Specifications provide sufficient basis for TA development
- Issues
  - Application installation (provisioning) model not yet defined
  - Access to TEE typically controlled by the manufacturer
  - User interaction
- Open TEE
  - Virtual TEE platform for prototyping and testing
  - Implements GP TEE interfaces
  - https://github.com/Open-TEE

Open-TEE - An Open Virtual Trusted Execution Environment. TrustCom/BigDataSE/ISPA (1) 2015: 400-407


The Register
Biting the hand that feeds IT

DATA CENTRE SOFTWARE NETWORKS SECURITY INFRASTRUCTURE BUSINESS HARDWARE SCIENC

Intel infosec folk TEE off open source app dev framework

World+dog can TEE off too, without spending megabucks

30 Jun 2015 at 11:10, Darren Pauli 155 18 2

A trio of Intel boffins have broken a vendor lock-down on trusted execution environments (TEEs) with the release of an open source framework that could help developers to build more secure apps.

Intel wonks Brian McGillion, Tanel Dettenborn, and Thomas Nyman (plus N. Asokan of Aalto University and University of Helsinki) released the OpenTEE software framework for developers as an alternative to expensive or non-existent TEE tools.

http://www.theregister.co.uk/2015/06/30/opentee_an_open_virtual_trusted_execution_environment/

Extra slides