Hardware Security

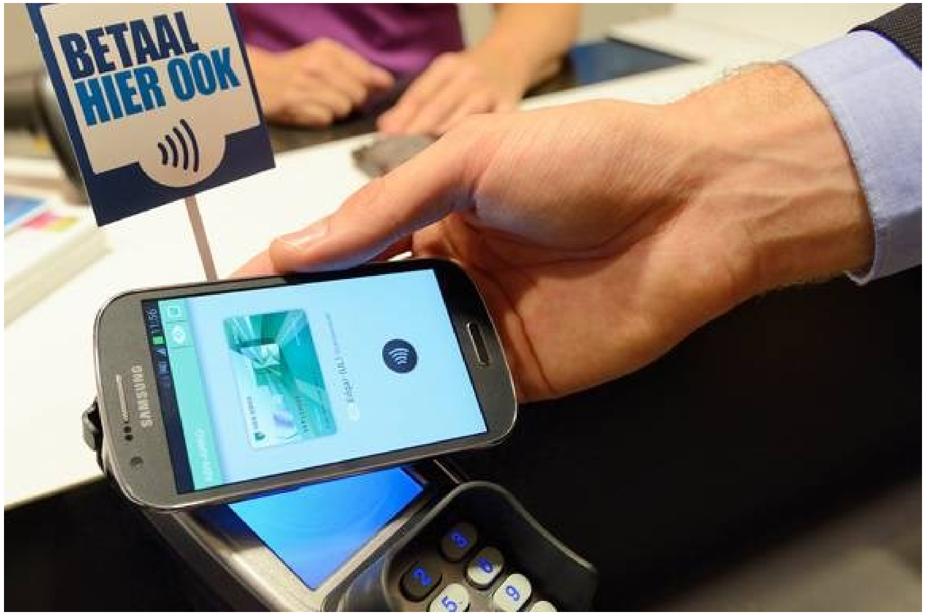
Trusted Execution Environments (TEEs) & Trusted Computing

Erik Poll

Digital Security

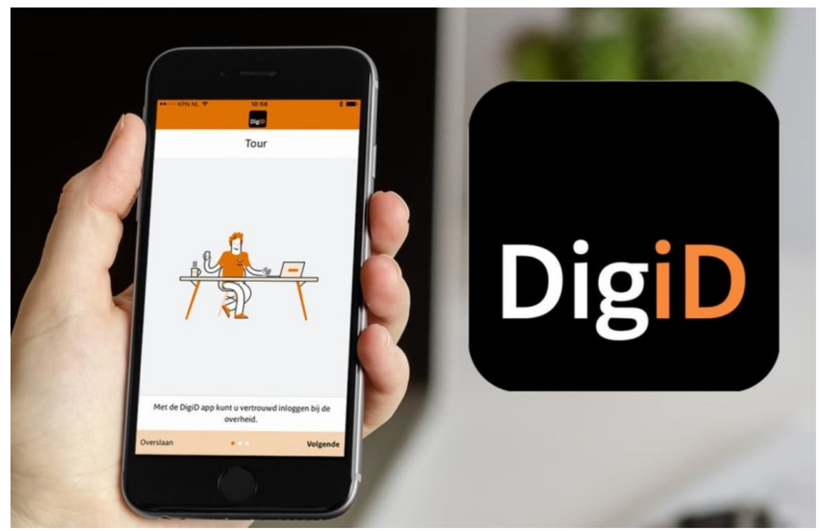
Radboud University Nijmegen

Exit smartcards, enter apps

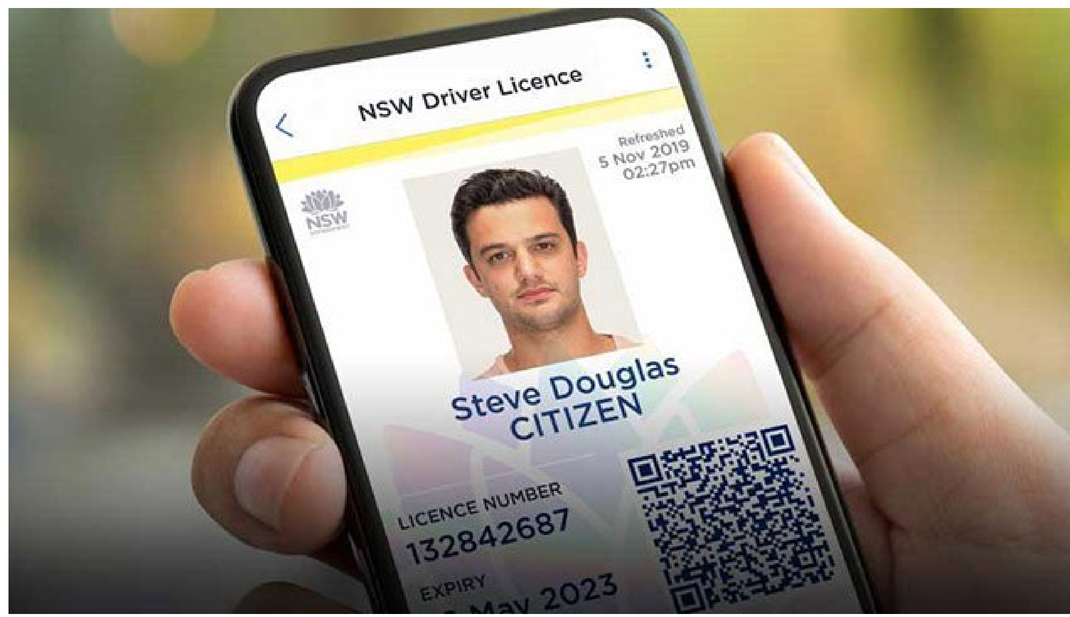


Exit smartcards

Exit smartcards

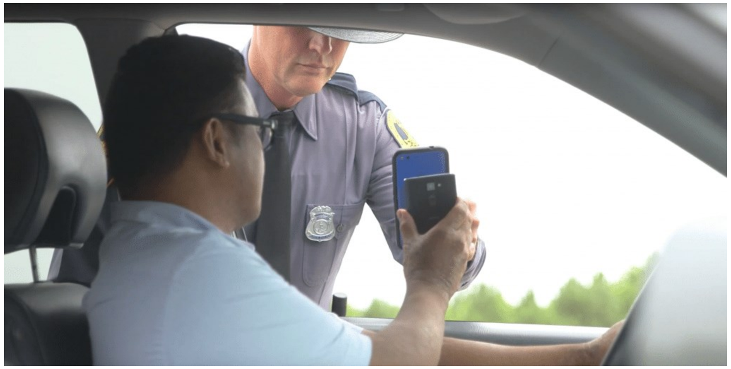


Exit smartcards



Another trend: Offline ⟶ Online

- Offline use in the physical world
- Online use in the cyberspace
- Combinations
  - incl. digital onboarding



Very different risks! Eg attacks in physical world usually

- do not scale
- come with risk of getting caught

Why TEEs?

Recurring security dilemma

- We want a powerful, fast, device, with lots of features, a nice GUI, and rich platform APIs that is easy to program
- We want a simple device, with a minimal TCB, for small & simple applications, that we can trust

ie. the eternal dilemma between functionality & security

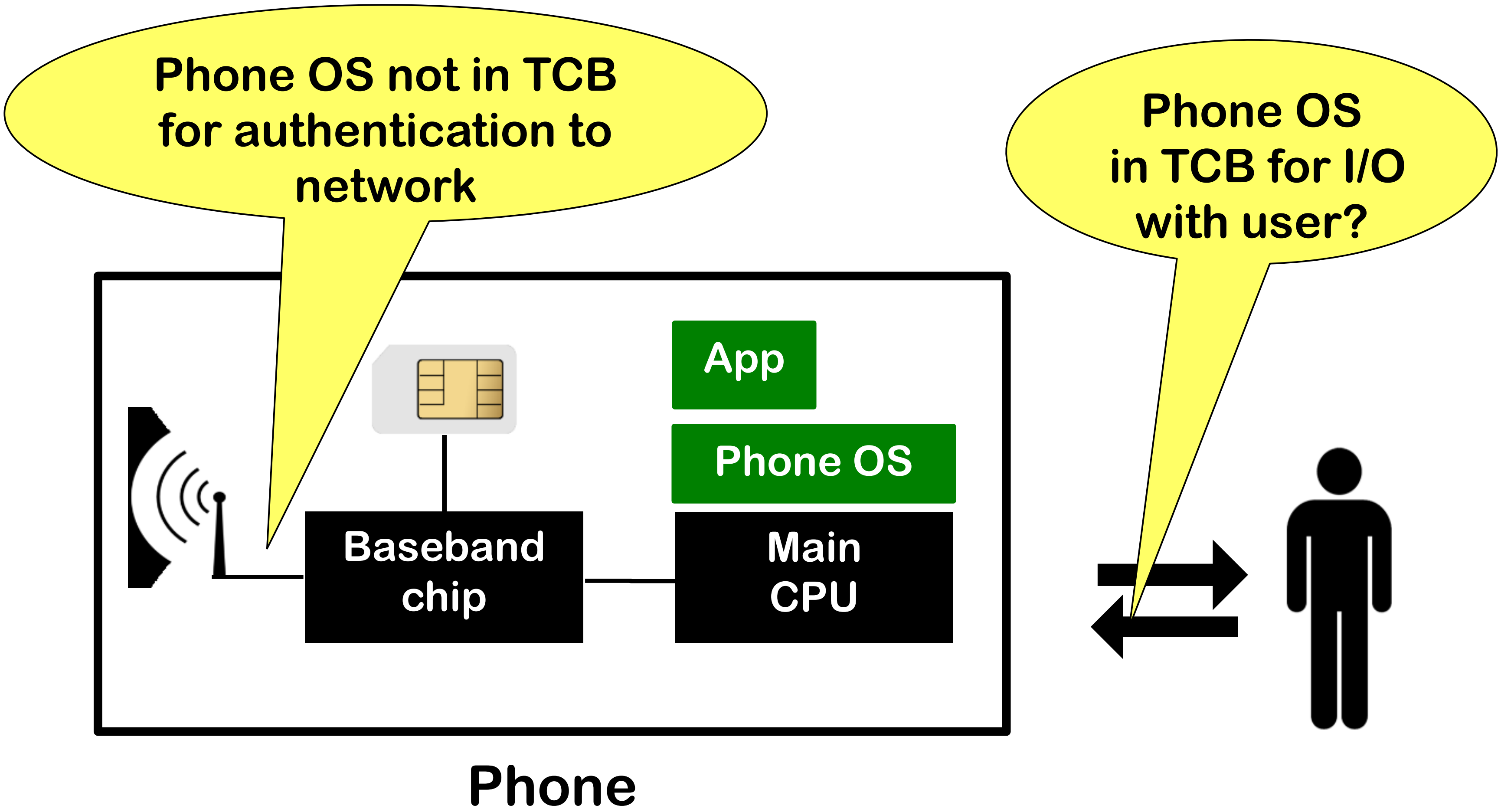
Motivating example: the SIM card

What are the security advantages for the telco?

- The phone hardware & software are not in the TCB for authentication
- ie. the telco does not have to trust the phone to keep crypto keys for authentication confidential
- ie. the telco only has to trust the SIM for confidentiality of keys and integrity of code

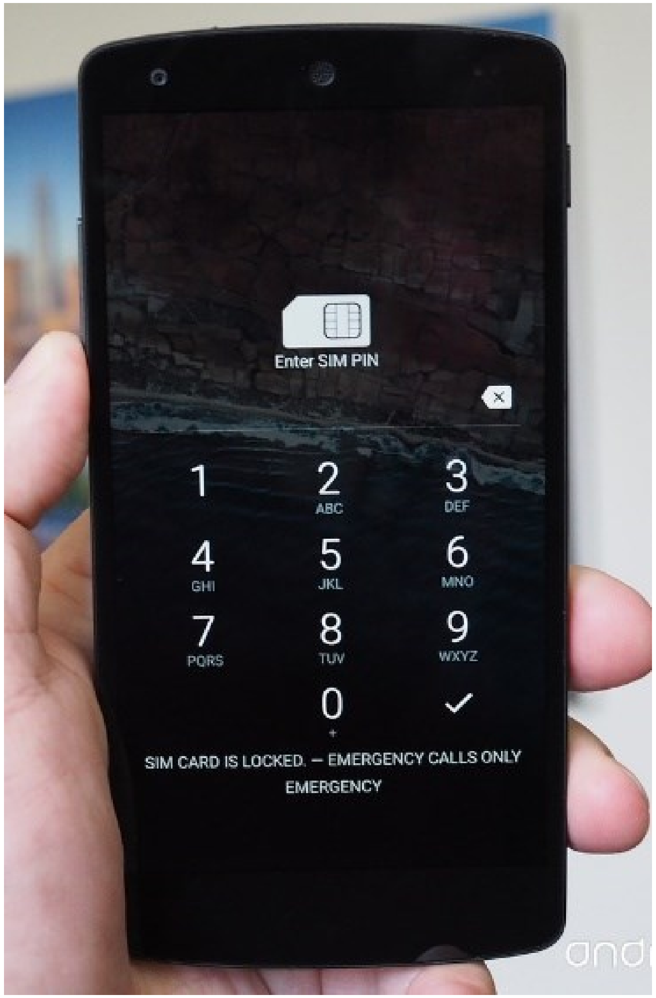
Limitation: user has to type in the PIN code to unlock the SIM, so some phone hw & sw in TCB for confidentiality of the PIN

SIM card as TEE

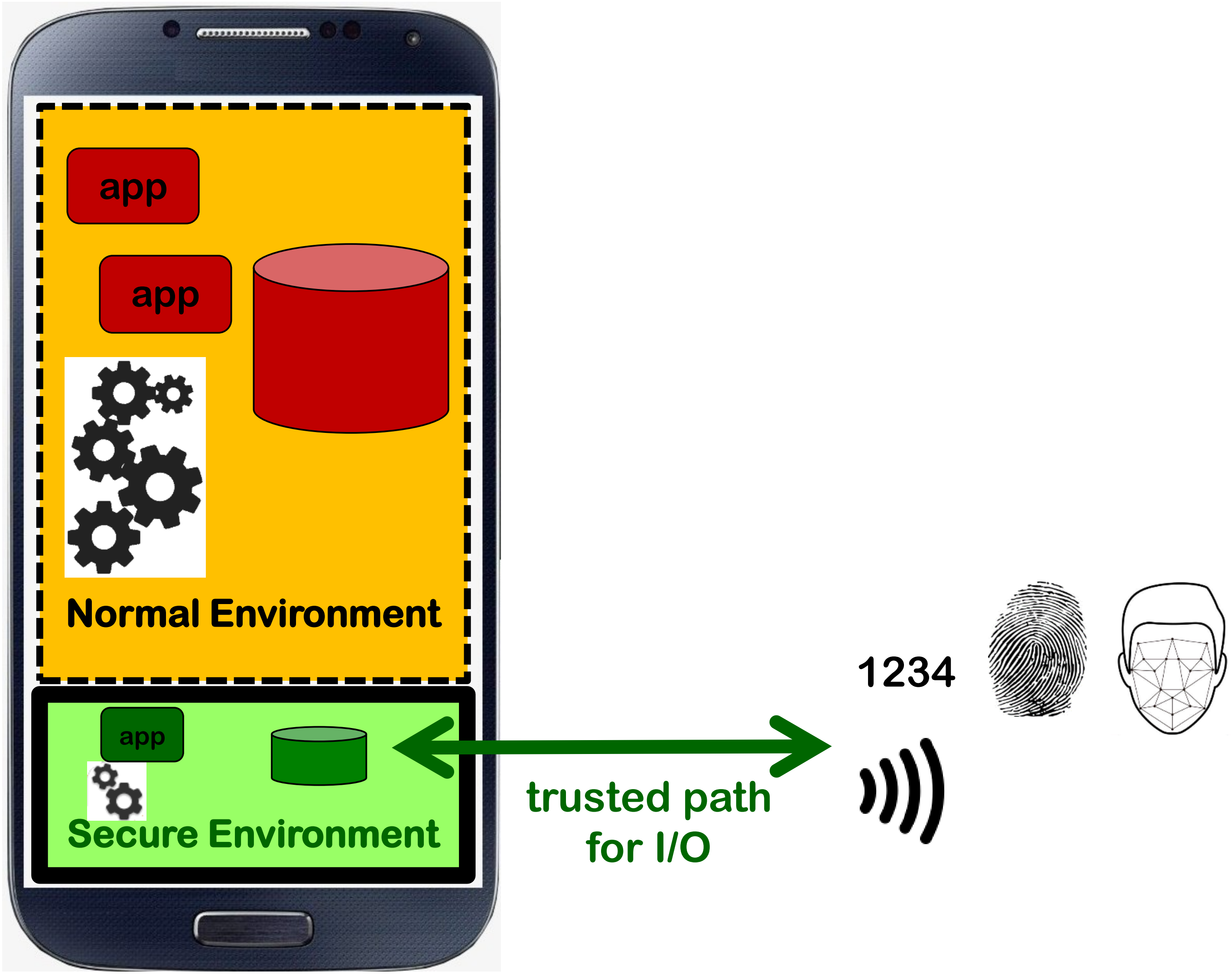


Trusted path?

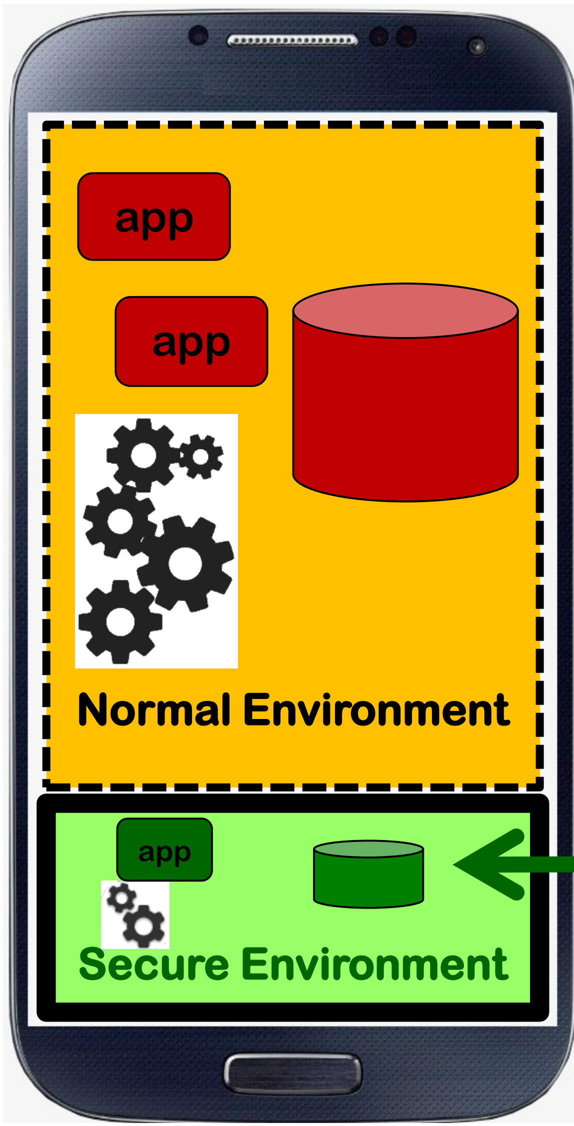
- What is in the TCB when you unlock you SIM card?
- Even if main OS is not in the TCB, malware on the phone could phish this!
  - by faking this display



Secure Environments *in* mobile phones

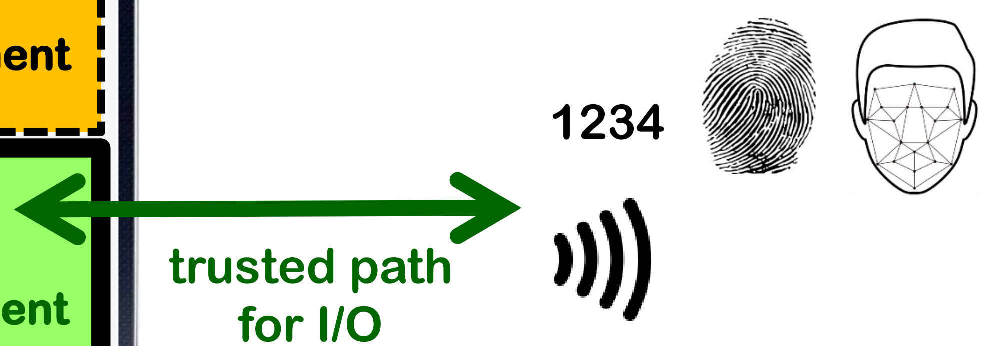


Secure Environments *in* mobile phones

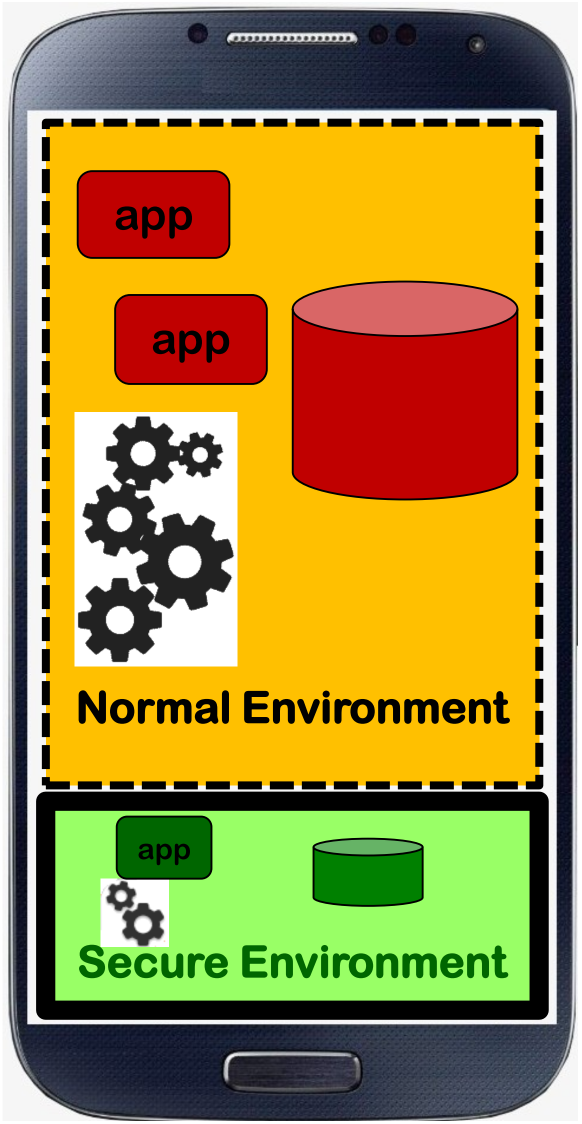


What can SE do?

- **processing** for crypto and access control checks
- **RNG**
- **data storage** for keys, PINs, biometrics
- **Fixed functionality provided by OEM**, or **extensible with trustlets**



Secure Environments *in* mobile phones



How?

1. ***physically*** separate
   a) SIM card
   b) Secure Element (RIP?)
   c) Apple Secure Enclave &
      Android Strongbox Keymaster
2. ***virtually*** separate
   a) ARM TrustZone TEE
      (getting less fashionable?)
   b) Whitebox crypto (😂)

TEE technologies

1. Having a separate chip
    a) SIM card
    b) Apple Secure Enclave
    c) Android StrongBox Keymaster (since Android 9)

2. TPM: a separate chip that can monitor the main processor
3. Flicker: which uses TPM
4. Intel IPT (Identity Protection Technology)
5. ARM TrustZone
6. Intel SGX

When people talk about TEE, they usually mean 2-6

Security of *mobile phone with SE* vs *smartcard*

– More complex and (hence) less secure

\+ Mobile phone can do I/O

\+ Mobile phone can do biometrics

\+ Loss of control: dependency on 3rd party device, OS, app store

- New and more powerful attacker models, in addition to usual attacks on SEs/smartcards
  1) Compromised OS
  2) Compromised SE
  3) Compromised app
  4) Malicious app

- Nearly always online
  This is both good (eg. for monitoring & response and for updating)
  and bad (as attack vector & for phishing)

– One SE can hold many credentials
  Like a multi-application smartcard. Bad for phishing.

- Enrolment & revocation are totally different:
  – complex, but + cheaper & more flexible

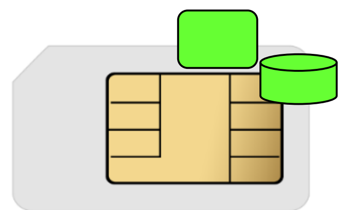
Rest of this lecture

- Security Goals of TEEs
- Technologies to build TEEs

Goals of TEE - conceptually

isolated execution
and
secure storage

+



+

trusted path
for trusted I/O

integrity & confidentiality of code & data even if OS is compromised

integrity & confidentiality of user I/O even if OS is compromised

First attempt at defining TEE

Platform that provides applications with the security guarantee of *isolation*

- integrity of behaviour
- integrity & confidentiality of data, at rest & during execution

against very powerful attacker

- malware on the same platform
- and even (partial) compromise of the application or platform

with a high level of trustworthiness

- minimal TCB
- ultimately relying on hardware

and mechanisms *to attest to the integrity* of the system

- as basis for others to trust it

TEE security goals (1) – ‘isolation’

- **Isolated Execution**

  Execution of an application cannot be compromised.

  Integrity & confidentiality of code and of data in use.

- **Secure Storage**

  Integrity, confidentiality and *freshness* of data at rest.

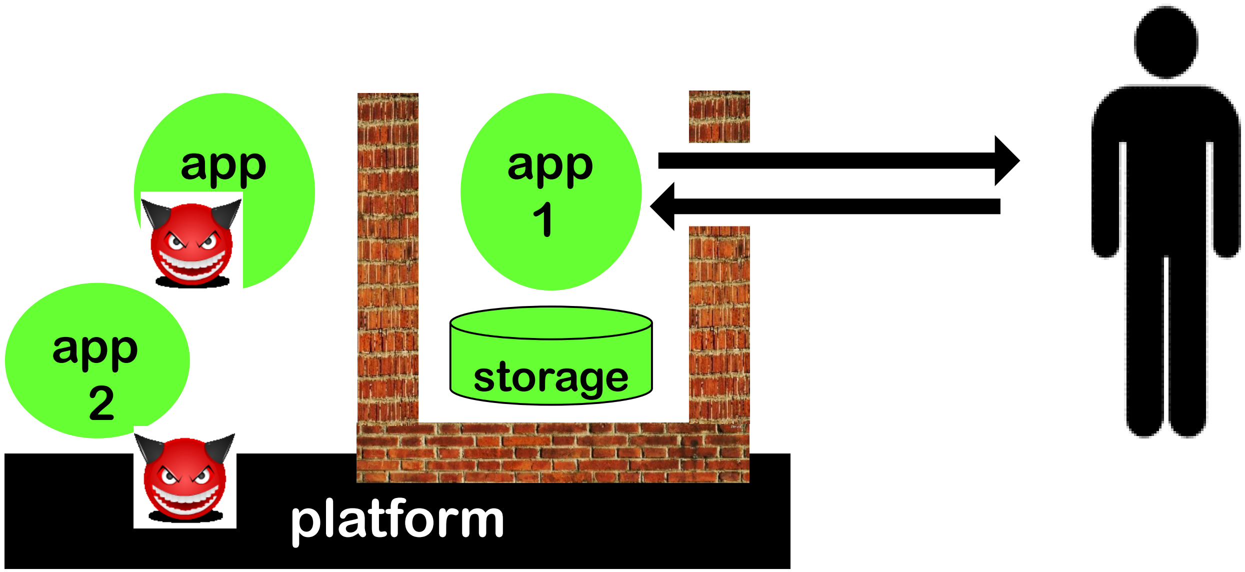
- **Trusted Path: a secure path to and from the user**

  Integrity & confidentiality of communication

  - **secure attention sequence, eg. Ctrl-Alt-Delete on Windows, or Home button on iOs & Android, is a special case of Trusted Path**

This is nothing new!
Any OS aims to provide these properties.

These security goals (1) – 'isolation'



Spoofing remains a tricky concern

- an app can know it has exclusive use of display or keyboard, but how can the human user know who it is talking to?

TEE security goals (2) – 'assurance'

Who & what are we dealing with? Can we trust this?

from perspective of an app, remote party, or local human user

- Platform Integrity
  - Can we trust or verify platform integrity?
- (Remote) Attestation
  - Can a (remote) party verify integrity of platform or app?
- Identification & Authentication
  - Can we authenticate the identity of a platform or app?
  - Ultimately, this requires some device identity
- Secure Provisioning
  - Mechanism to send data to specific software module on a specific device
    - eg for DRM, updating, or sync-ing apps across devices

- Security Goals of TEEs
- Technologies to build TEEs
  - TPM
  - Flicker & SGX
  - ARM TrustZone

Trusted Computing & TPM

Trusted Computing

- Initiative by industry consoritium
  - initially TCPA (Trusted Computing Platform Alliance), succeeded by TCG (Trusted Computing Group)

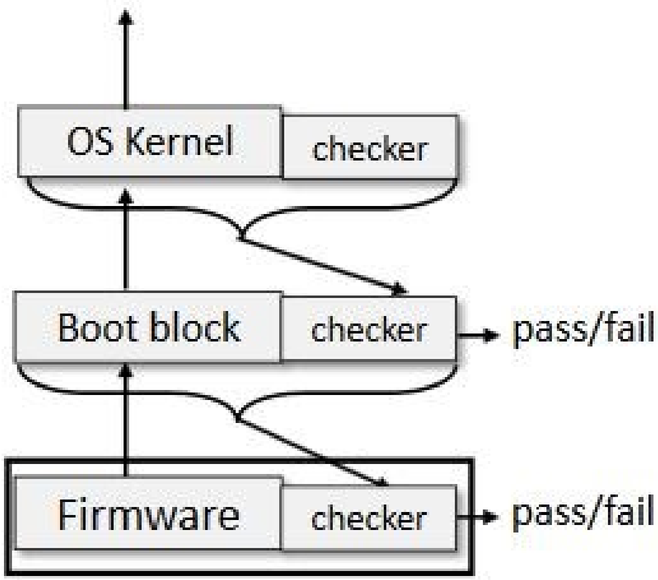
  including Microsoft, AMD, Intel, IBM, HP,....

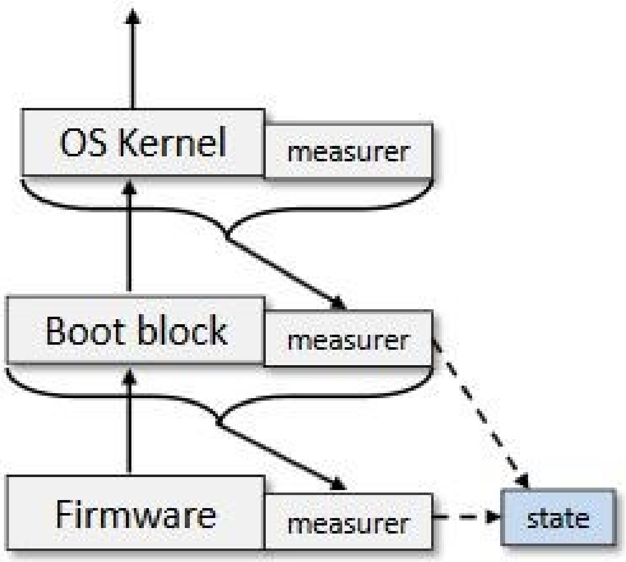
- Goal: common open spec of TPM (Trusted Platform Module)

- TPM is separate chip on the motherboard
  - that *monitors* the CPU & *offers services* to the CPU, aka protected capabilities that use shielded locations, incl. authenticated boot

  NB the main CPU remains in control!

Platform Integrity: Secure vs Authenticated Boot





Secure boot

Authenticated boot

Secure vs Authenticated Boot

- Secure Boot: ensuring that the right system is booted
  - At each step of the boot process,
    before code is loaded & executed, the integrity is checked,
    eg using code signing
  - The boot process is halted if integrity checks fails
  - The integrity checks have to be trusted, of course
- Authenticated Boot: checking which system has booted
  - At each step of the boot process,
    a cryptographic hash of the code is computed (a integrity measurement), and chained with earlier hash
  - The boot process is never halted, but integrity measurement can be checked later
  - *The computation, storage & reporting of integrity measurements has to be trusted, of course*
    - *hence.... the TPM*

Protected Capabilities of TPM

- Crypto, incl. secure key storage & random number generation
- Integrity metric reporting
  - chip can compute & report integrity measurements
    - stored in PCRs (Platform Configuration Registers)
  - for attesting to the state of device, incl. authenticated boot
- Special kind of secure storage: sealing of data
  - access to data conditional to device being in a particular state
    - ie you can only access the data if the integrity measure of the code is a certain value
    - Typical use case: DRM

Using TPM for TEE?

Basic idea:

- TPM measures hash of all software loaded since BIOS boot, incl. OS, and even application code, and attests to the integrity

so that

- software running on the machine and external parties can verify system state (remote attestation)
- access to remote services or local data can be conditional on system state
  - by using sealed storage of data
    - eg this file can only be opened for a given software stack
  - by using remote attestation for remote services
    - eg attesting that this is a genuine Intel processor running a correct version of Windows

Trusted Computing controversy (early 2000s)

Lots of debate about: openness, privacy, and control

- **TPM *cannot* prevent user running Linux on Intel hardware, but *can* prevent LibreOffice on Linux from opening .doc files**
  - **by using sealed storage**
- **TPM is ‘a way for Bill Gates to make the Chinese pay for software’?**
- **Privacy concern: TPM has a unique serial number**
  - **But DAA for anonymous remote attestation to reduce privacy impact**
    - **attesting that eg. 'this is *some* legitimate copy of Windows running on *some* AMD machine'**

More info:

- **Ross Anderson’s FAQ**

  http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html
- **[Felten, Understanding Trusted Computing, IEEE Security & Privacy 2003]**

Trusted Computing  ?

Big practical problems built-in from start

- **Software stack is far too dynamic**
  - **with continuous patching of OS, variety in device drivers, etc., the chance that 'identical' computers produce identical integrity measurement is small**
- **OS is far too big to be trusted as TCB**
  - **the idea that checking the integrity of boot sequence incl. the entire OS will ensure absence of malware is silly**

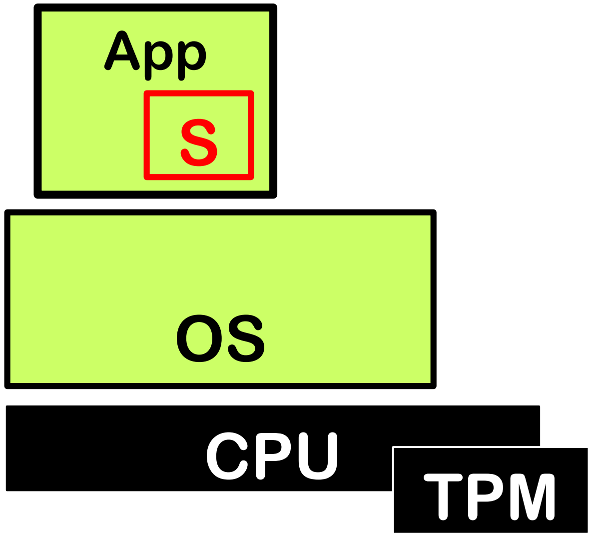
- **Microsoft stopped development of NGSCB aka Palladium, their intended 'trusted OS' that would use the TPM, in 2004.**
- **TPM is still used for Bitlocker**

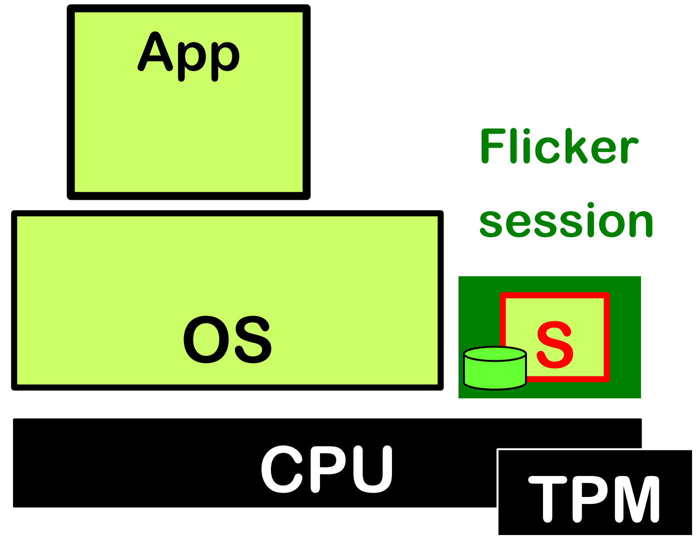
Flicker & SGX

-

providing secure sessions/enclaves on main CPU

Normal Execution vs Execution using Flicker





OS is in the TCB for entire App

Part of the App, S, executed in Flicker session

- OS no longer in TCB for S

Dynamic Root of Trust in TPM v1.2

- TPM v1.2 added for *dynamic* PCRs
  - not for integrity measurement *starting at boot*, but for integrity measurement starting *from later point in time*
  - set to -1 on boot; can be set to 0 by CPU, to record integrity measurement from that point on

- Special register PCR 17 :
  - can only be reset by one special instruction of CPU
    - `SKINIT` on AMD SVM, `SENTER` on Intel TXT/LaGrande
  - resets the CPU, disables interrupts and DMA
  - measures & executes Secure Loader Block

Flicker TEE

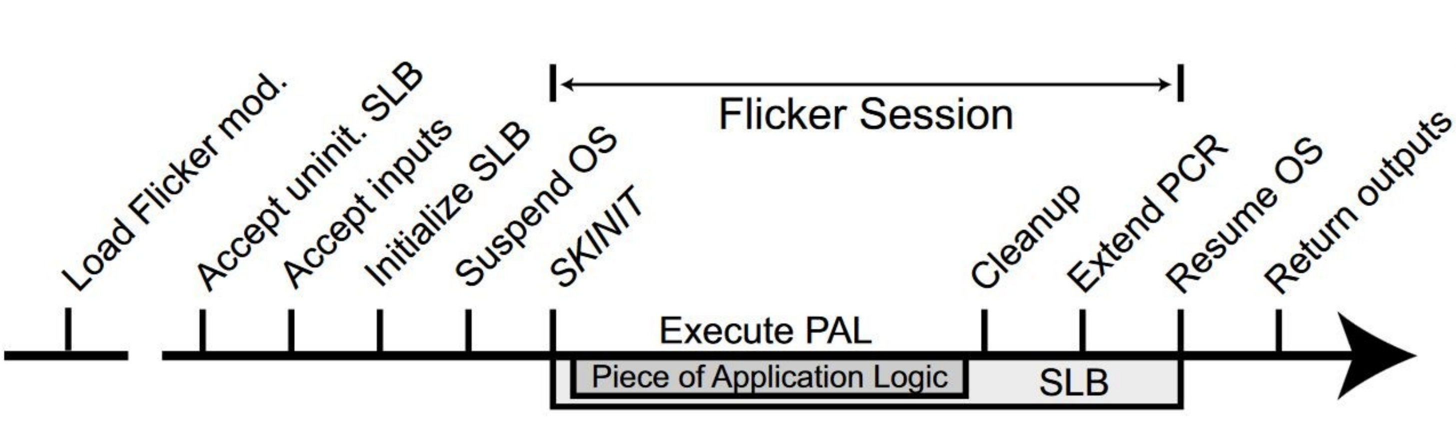
Flicker uses TPM with dynamic PCRs for trusted execution, briefly switching to secure mode & back to normal, with the following steps

1. all normal execution (incl. OS) is suspended
2. Flicker session: small piece of code executed using `SKINIT`
   - with code integrity measurement in PCR 17
   - possible accessing & updating sealed memory
3. normal execution (incl. OS) resumes

Code executed in Flicker Session isolated from all other execution:

- No code executed before or after can influence or observe it
- Only 250 lines of software in TCB
- Downside: the code cannot use any OS services

Flicker TEE



- sensitive code fragment called PAL (Piece of Application Logic)
- PAL is included in the SLB (Secure Loader Block) that is passed to the `SKINIT` instruction

Example uses:

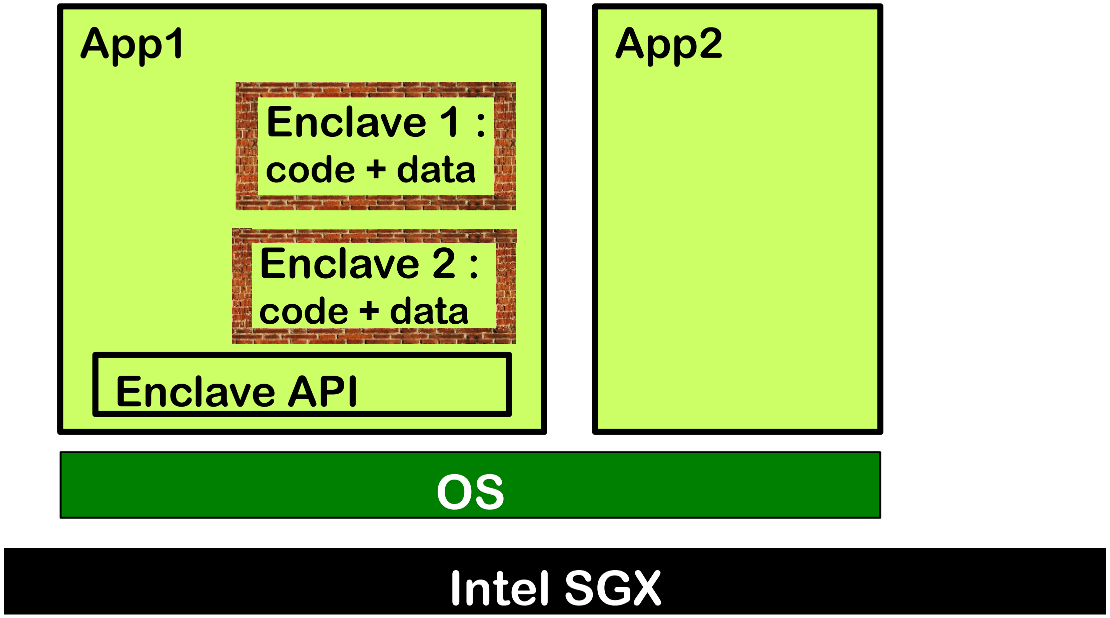
- running some crypto code with access to key material in sealed memory
- a password check with access to password

[McCune et al., Flicker: An Execution Infrastructure for TCB Minimization, EuroSys 2008]

Intel SGX

Parts of app can be done in secure enclaves

- Similar to Flicker session, so main OS no longer in TCB
- Each enclave has its own code & data, but can access all memory of the app
  - Confidentiality & integrity of code & data protected
  - Entry points into enclave's code are secured
    - to stop ROP (Return-Oriented Programming) style attacks



Intel SGX – capabilities & limitations

- HW provides Isolation, Attestation, Sealed Storage
- Context switch to enclave is fast

- But: side-channel attacks on SGX exist
  - Malicious enclave can eg extract RSA private key used by other enclave on same machine
  - Malicious enclave code is impossible to detect or analyse, as it is protected by the enclave mechanism

ARM Trustzone
-
providing a secure & an insecure world

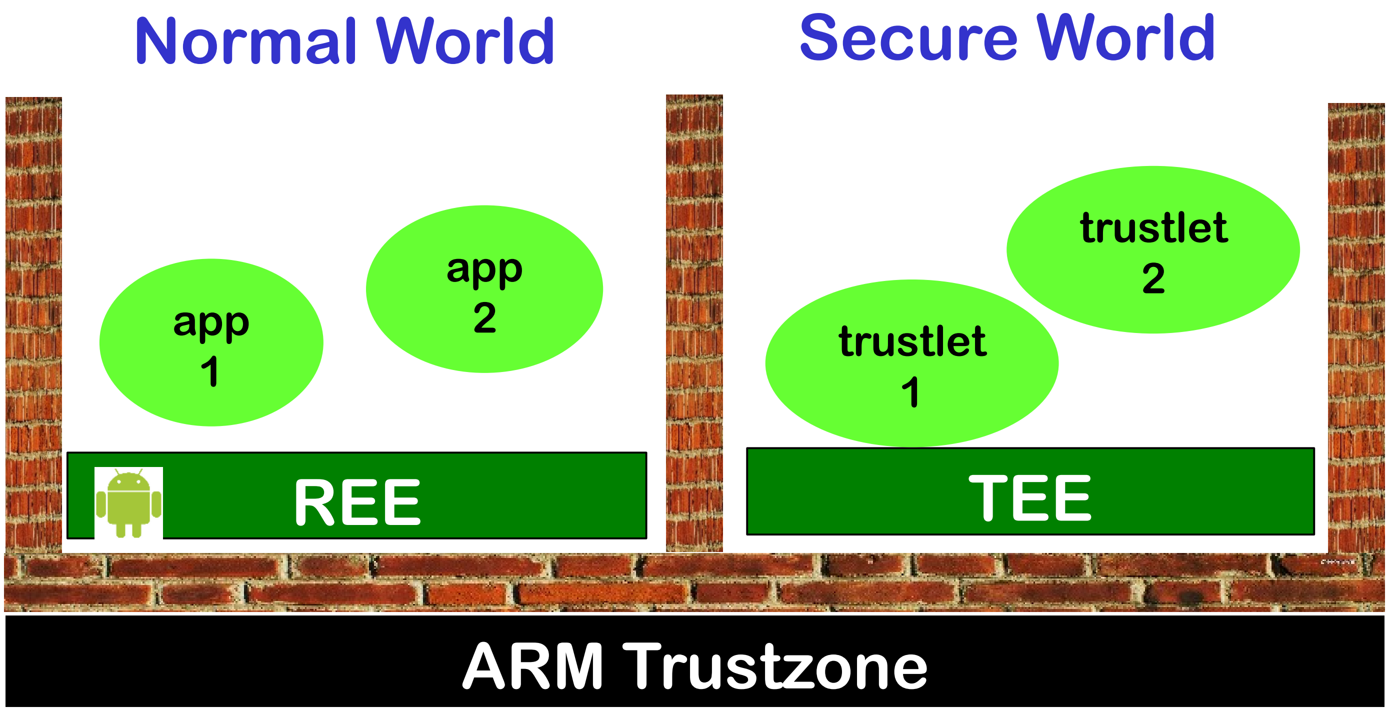
ARM TrustZone

ARM TrustZone is a single processor (SoC) offering 2 modes:

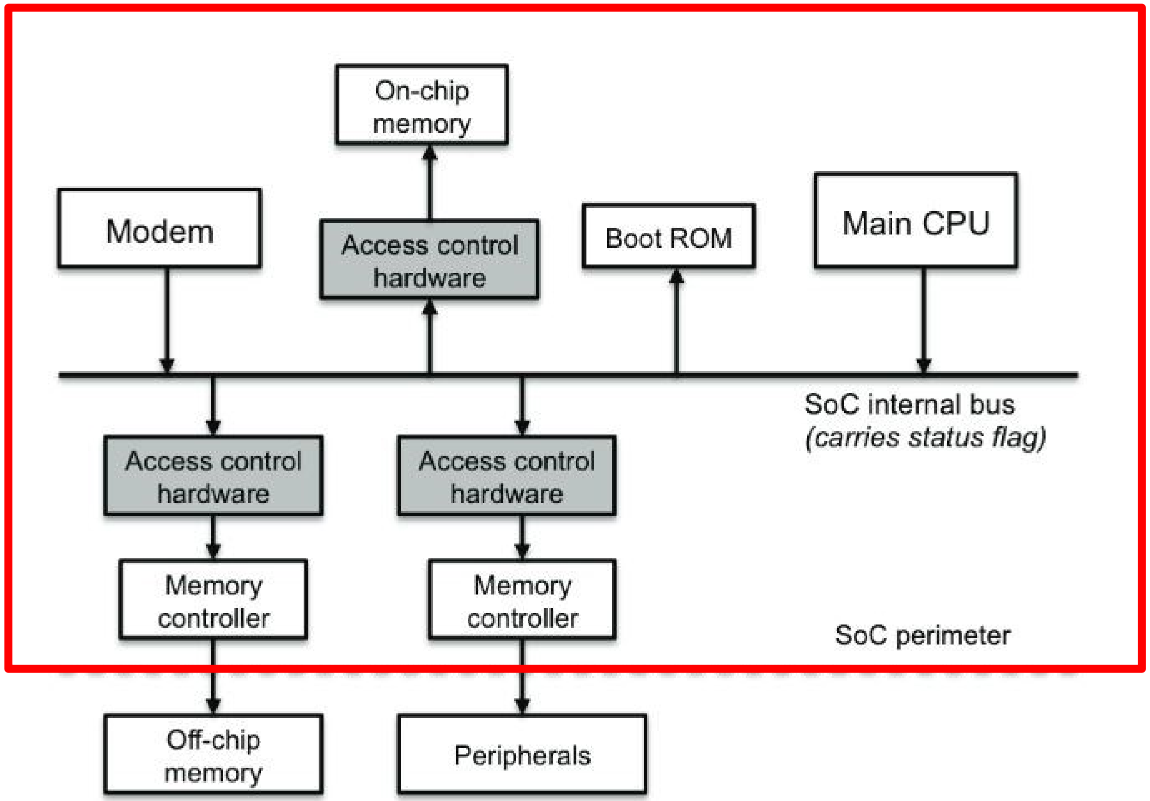
- ‘normal world’ and ‘secure world’
  - Extra 33rd bit on the bus, to indicate the mode
  - Device could have an indicator (eg LED) for the mode
  - Separation of memory, peripherals, DMA, and interrupts
  - Context switch between worlds is slow

- Intended use
  - Untrusted OS, eg Android, runs in the normal world, providing REE (Rich Execution Enviroment) for normal apps
  - Secure world provides TEE for sensitive applications & services (aka trustlets)
- TrustZone available on many Android smartphones/tablet, but use of secure world for for manufacturer-internal purposes

ARM TrustZone

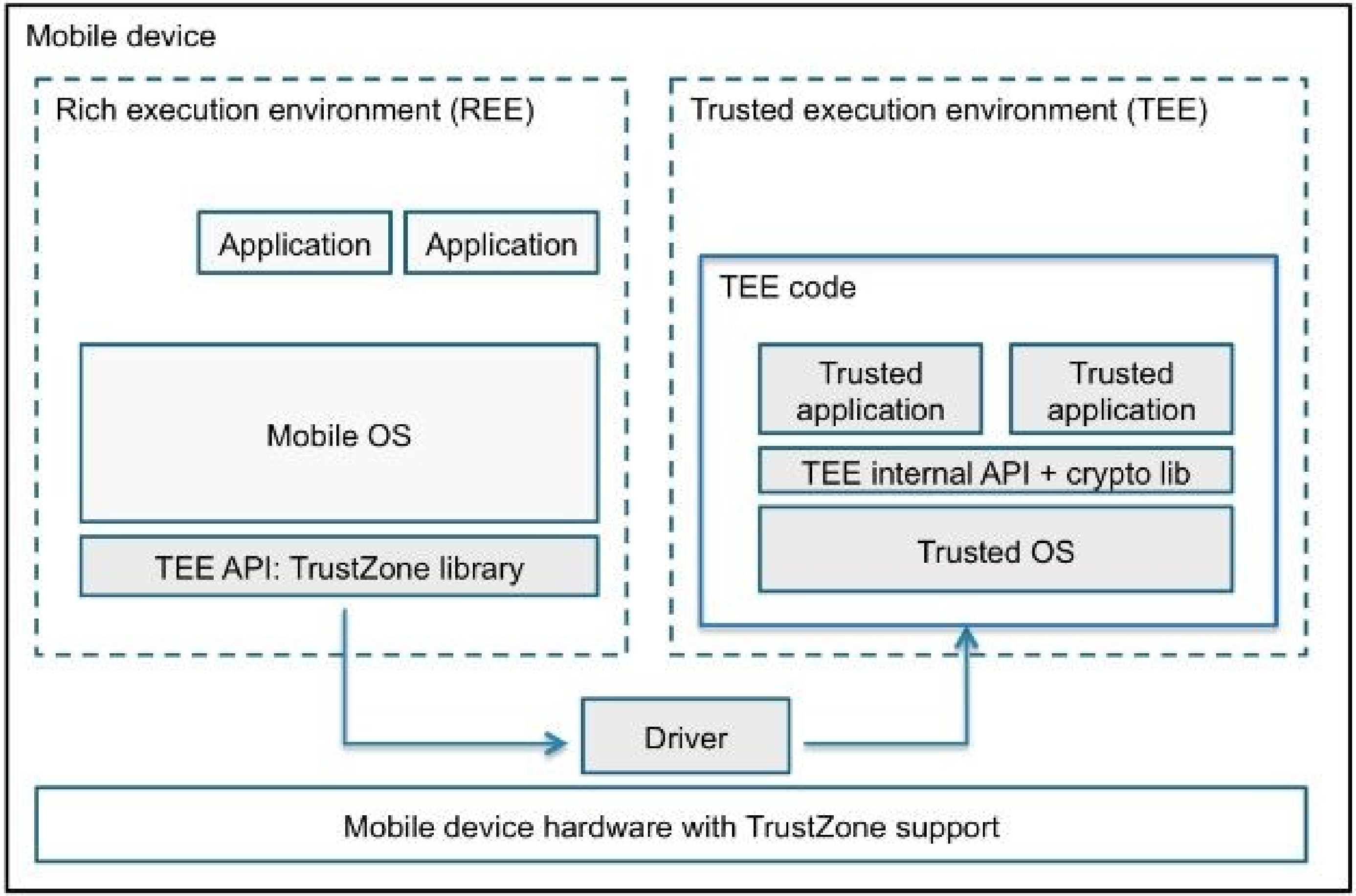


TrustZone SoC hardware architecture



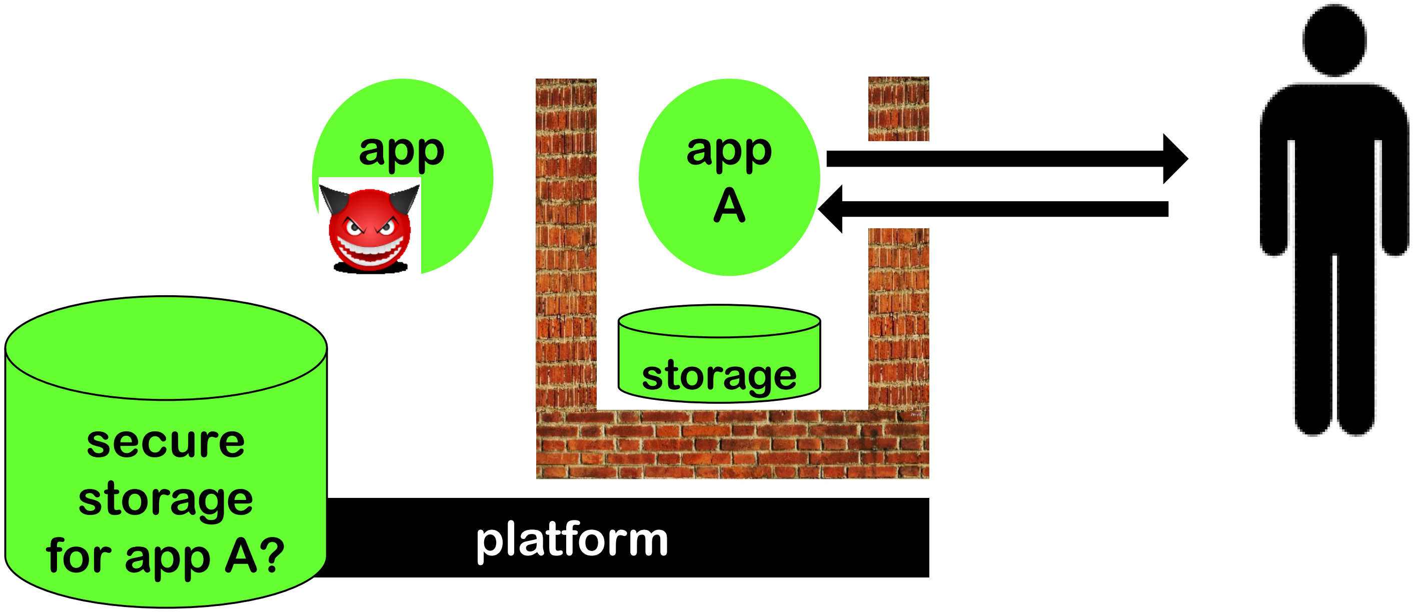
[source: Ekberg et al., The Untapped Potential of Trusted Execution Environments on Mobile Devices, IEEE Security & Privacy 2014]

TrustZone software architecture



[source: Ekberg et al., The Untapped Potential of Trusted Execution Environments on Mobile Devices, IEEE Security & Privacy 2014]

Secure storage in untrusted world?



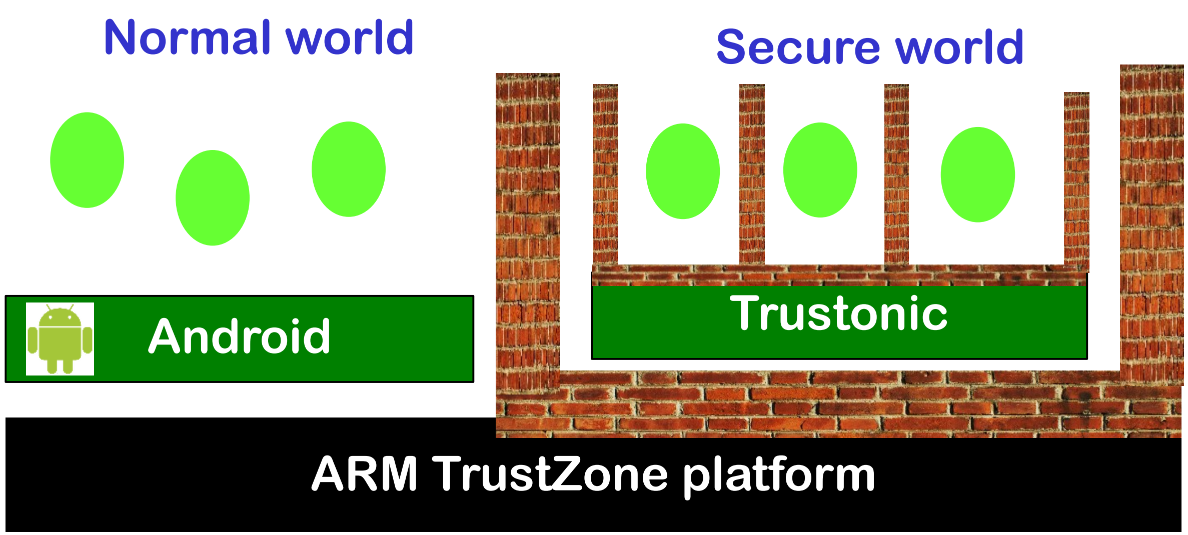
Persistent storage can be done in untrusted world, if we use encryption plus integrity & freshness checks.

Trusted app still needs some secure storage in trusted world

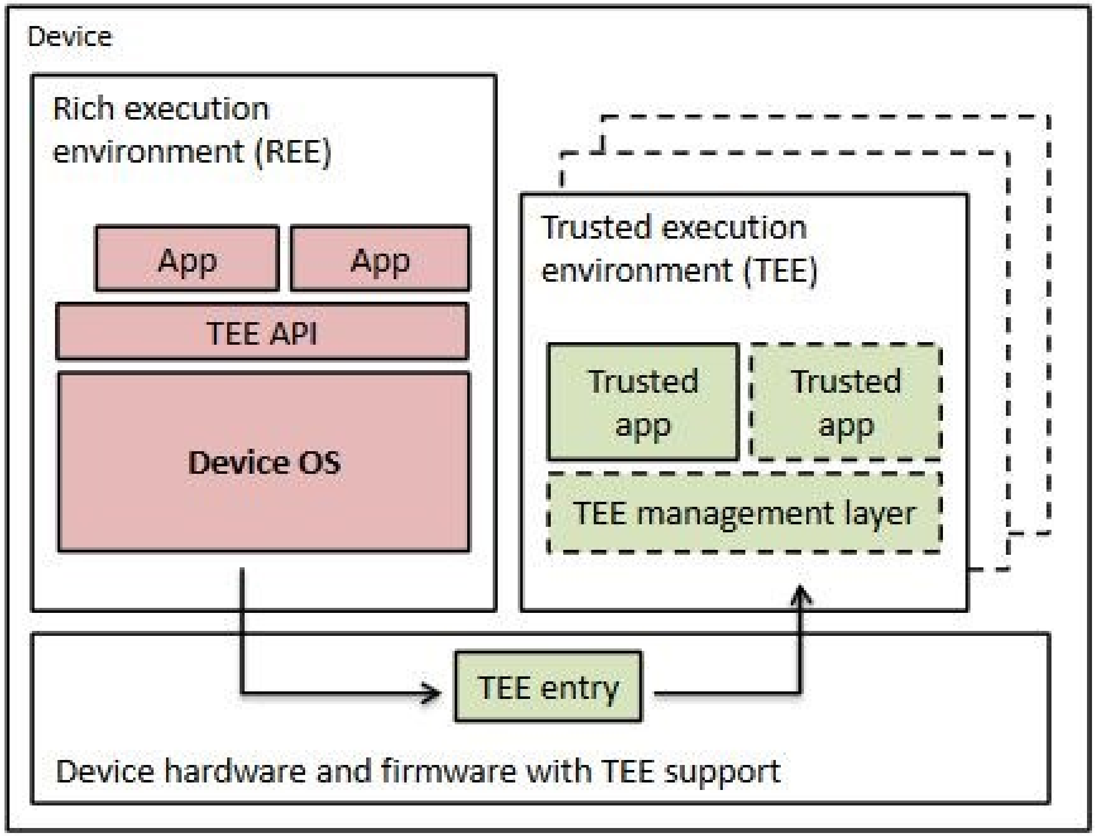
- for crypto keys for confidentiality & integrity
- for sequence numbers to ensure freshness (Data Rollback Protection)

Trustonic

- TrustZone only provides two worlds
  – protection one way: trusted protected from untrusted, not vv
- Trustonic provides multiple isolated enviroments within the secure world
  – like Global Platform isolates applets on JavaCard smart card
- Samsung KNOX does something similar



Trustonic/KNOX software architecture



[source: Ekberg et al., The Untapped Potential of Trusted Execution Environments on Mobile Devices, IEEE Security & Privacy 2014]

Analysis of TrustZone security failures

Cerdeira et al, SoK: Understanding the Prevailing Security Vulnerabilities if TrustZone-assisted TEEs, IEEE S&P 2020

- SoK = Systemisation of Knowledge

Security problems due to

- software bugs in trusted OS and trusted apps
- architectural deficiencies
  - large attack surface, dangerous API calls,
    no ASLR, no stack canaries, …
- hardware attacks
  - voltage & clock manipulations (CLKSCREW)
  - micro-architectural side-channels via caches, branch prediction, or RowHammering

last month

Trust Dies in Darkness:
Shedding Light on Samsung's TrustZone Keymaster Design

Alon Shakevsky
shakevsky@mail.tau.ac.il

Eyal Ronen
eyal.ronen@cs.tau.ac.il

Avishai Wool
yash@eng.tau.ac.il

Tel-Aviv University

https://eprint.iacr.org/2022/208

Feb 20, 2022

Comparison & Conclusions

Separate processors or not?

- TrustZone and SGX use the same processor for both trusted and untrusted code

- TPM involves a separate processor
- Apple Secure Enclave and Android Strongbox Keymaster also involve a separate execution environment
  - processor + RNG + (limited) storage,
    but without TPM's functionality to monitor the main processor
  - beware: not all implementations of Android KeyStore API are hardware-backed!

- Advantage of using the same processor:
  lots of CPU power, lots of memory ☺
- Disadvantage: more security risk of side channels ☹

Open questions

- Will smartcards disappear and will we use our smartphones for everything?
  - If so, will we use TEEs like ARM Trustzone & SGX or separate processors like Apple Secure Enclave & Android Strongbox Keymaster?
  - Or will some security-sensitive apps choose not use any special hardware features?

- How can we compare the security of app-based solutions to smartcard-based solution?

- How do we evaluate the security of app-based solutions?