Hardware Security

Trusted Execution Environments (TEEs) & Trusted Computing

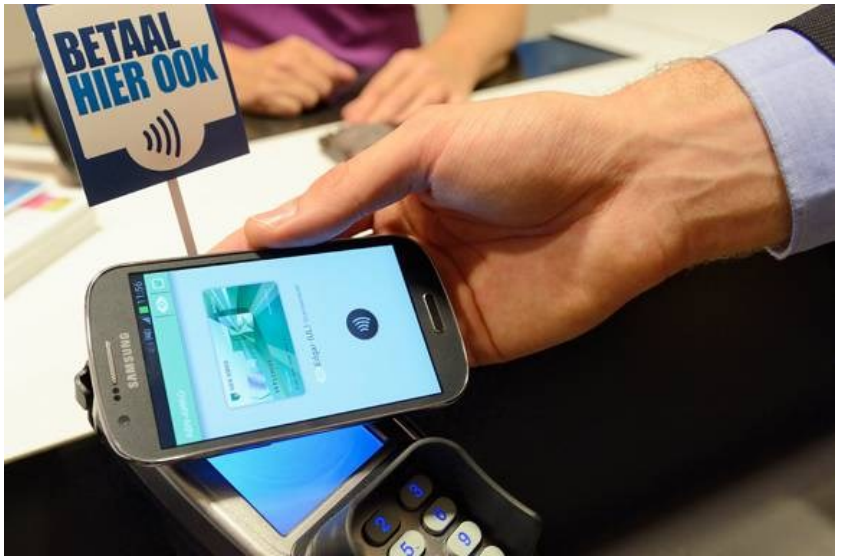
Erik Poll

Digital Security

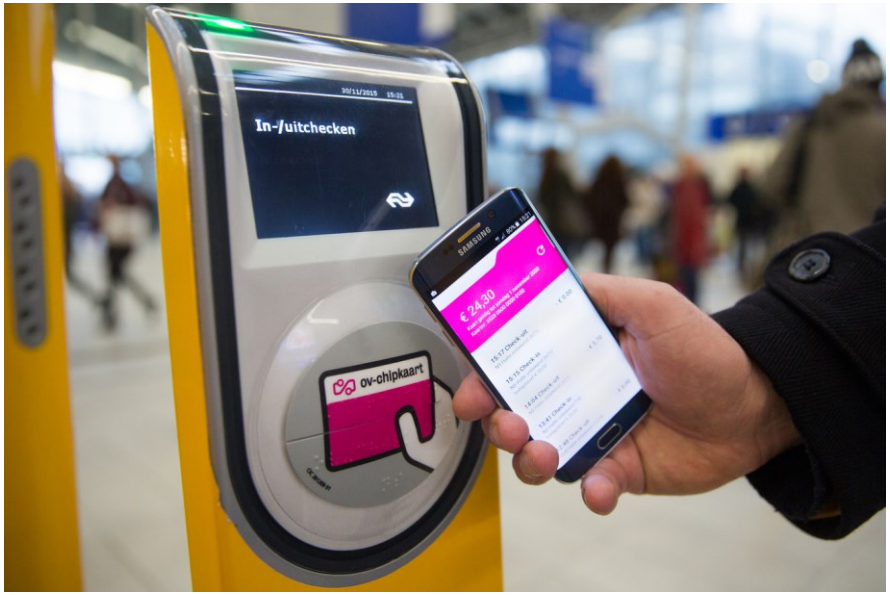
Radboud University Nijmegen

TU/e Master in Cyber Security

Exit smartcards?



Exit smartcards?



Exit smartcards?

Beware the dreaded T-word...

TEE = *Trusted* Execution Environment

Statements involving the word 'trust' are

- **likely to be bullshit**
- **often wrong or vacuously true**

Do you think that "trusted execution environments" exist?

Can you give examples?

example *trusted* execution environments

Beware the dreaded T-word…

- All execution enviroments are trusted (if they are used)

  by some party, for some purpose

- By TEEs people mean a bunch of technologies to provide a highly trustworthy, secure computing environment inside a 'normal' processor/device

  ie. providing some security guarantees, for some attacker model

  Best-known examples: *TPMs, ARM TrustZone, Intel SGX*

  You can argue that SIM cards, Apple Secure Enclave and Android Strongbox Keymaster are also TEEs, even though these involve separate processor

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

How to resolve this dilemma?

1. use two devices, eg
   a) mobile phone + SIM card 
   b) laptop + bankcard & EMV-CAP smartcard reader 
   c) web server + HSM 

2. have simple, secure computers for specific purposes




3. TEEs: integrate small & trustworthy execution environment inside the richer powerful device

Goals of this lecture

Understanding TEEs:

- **Security objectives**
- **Technologies / architectures to achieve these**

In other words:

- ***What are the security properties ?***
- ***How are these properties guaranteed ?***
- ***How good are these guarantees ?***
  - ***incl. what is the attacker model and the Trusted Computing Base (TCB)?***

Motivating example: the SIM card

What are the security advantages for the telco?

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

Overview of rest of this lecture

1. Attacker Models for TEE
2. Security Goals of TEEs
3. Technologies to build TEEs

Attacker models: for crypto, protocols, software

1. **Crypto attacker**

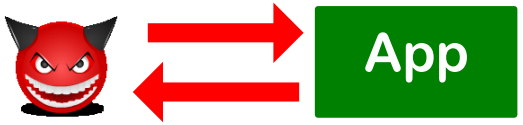
   wants to extract keys

2. **Dolev-Yao attacker**

   wants to break security guarantees of a protocol

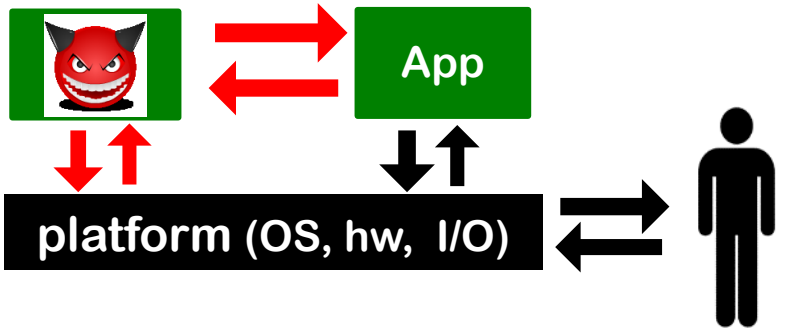
3. **I/O attacker**

   wants to create *any* unwanted behaviour with malicious input



More powerful I/O attacker models

- **Malware attacker**
  access to the same platform
  incl. common I/O channels
  and persistent storage
  (eg spoofing/phishing)



- **Code attacker**
  controls (some) code of by victim app
  (eg code injection via buffer overflow, XSS, or malicious library)

- **Platform attacker**
  eg malware attacker that
  managed to compromise the OS

TEEs also aim to protect against code & platform attacks!

Defending against code/platform attacker?

You might think that defending against code or platform attacker is hopeless.

But all is not lost! Possible defences:

- Runtime integrity checks on code, eg
  - stack canaries, CFI (Control Flow Integrity), … against buffer overflows
  - TPM reporting a hash of the software stack
- Sandboxing aka compartementalisation, eg
  - hardware-based Flicker sessions & Intel SGX enclaves

Of course, such mechanisms do come with a TCB.

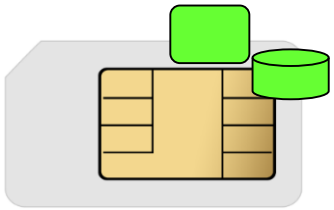
Overview of rest of this lecture

1. Attacker Models for TEE
2. Security Goals of TEEs
3. 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

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

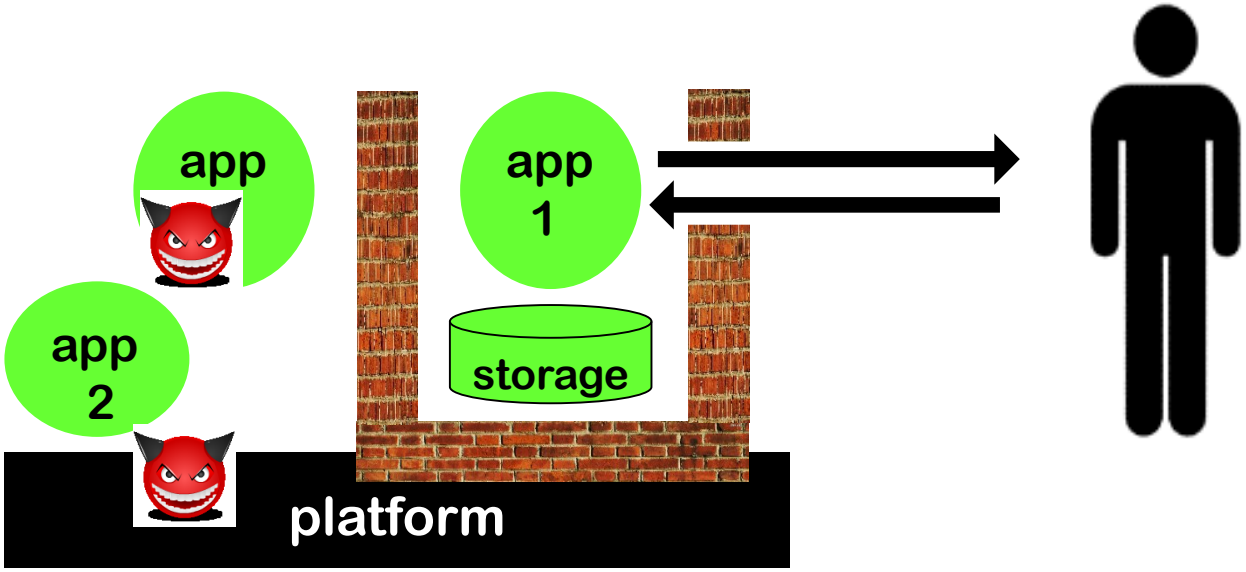
  Secure attention sequence, eg.

  - Ctrl-Alt-Delete on Windows,
  - 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?**

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**

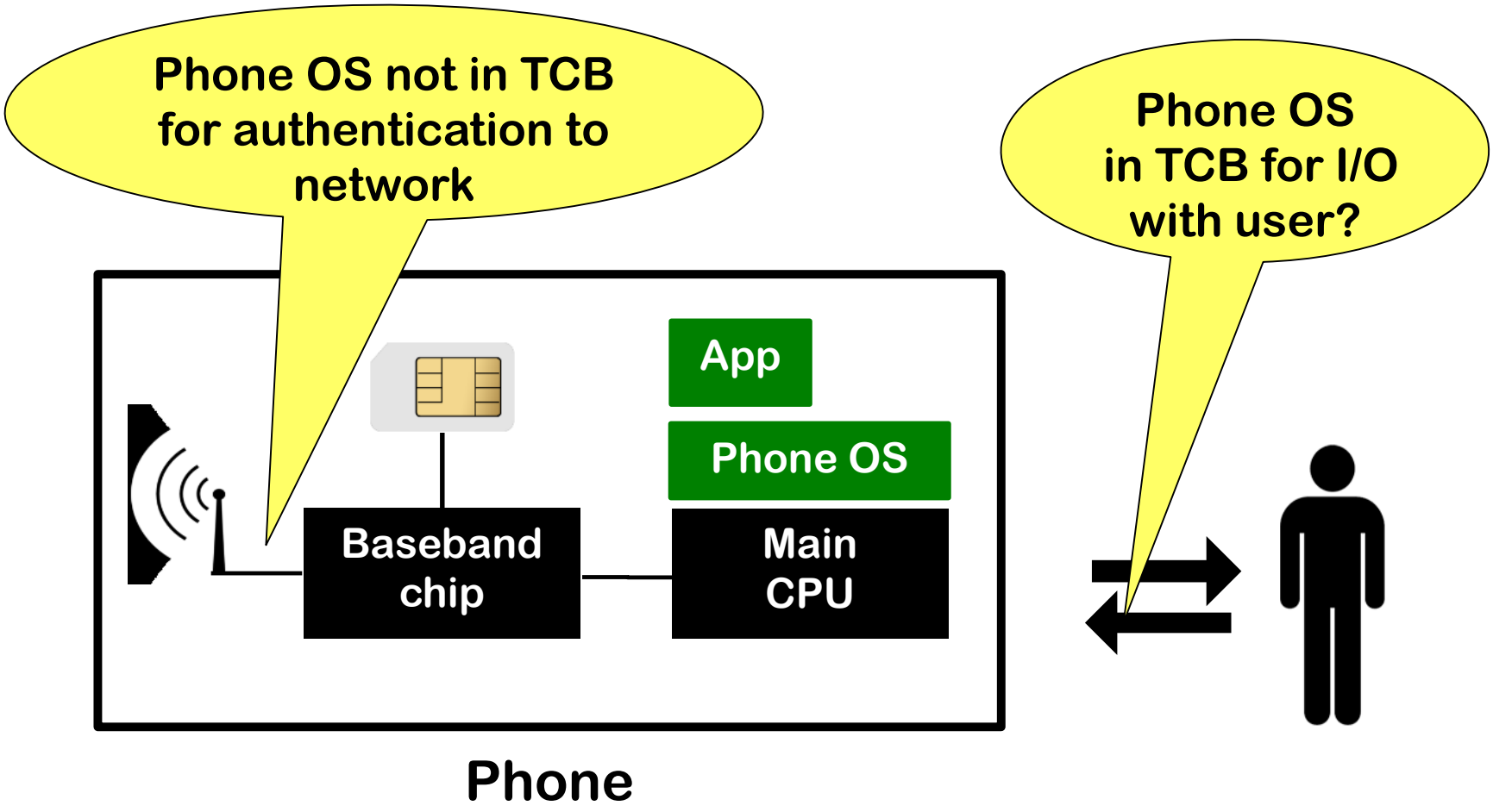
How to provide 'isolation'?

- use different physical devices
  - very secure ☺ and provides Trusted Path ☺
  - but costly & cumbersome ☹

How to provide 'isolation'?

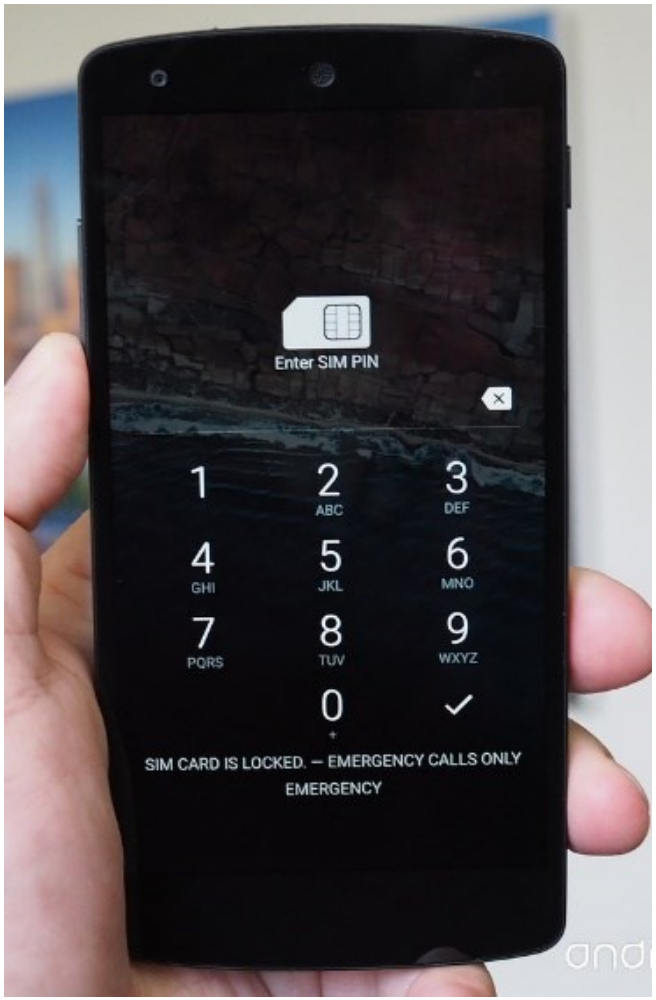
- use different physical devices
  - very secure ☺ and provides Trusted Path ☺
  - but costly & cumbersome ☹
- classic OS
  - huge TCB ☹
- hypervisor
  - smaller TCB 😐
- smartcard
  - even smaller TCB ☺
  - also some protection against physical attacks ☺
  - limited resources ☹,
    no I/O to user ☹, let alone Trusted Path ☹
- TPM & TEE hardware-based solutions ☺ ☺ ☺ ??

SIM card regarded 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



Overview of rest of this lecture

1. Attacker Models for TEE
2. Security Goals of TEEs
3. **Technologies to build TEEs**

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

Trusted Computing & TPM

Trusted Computing

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

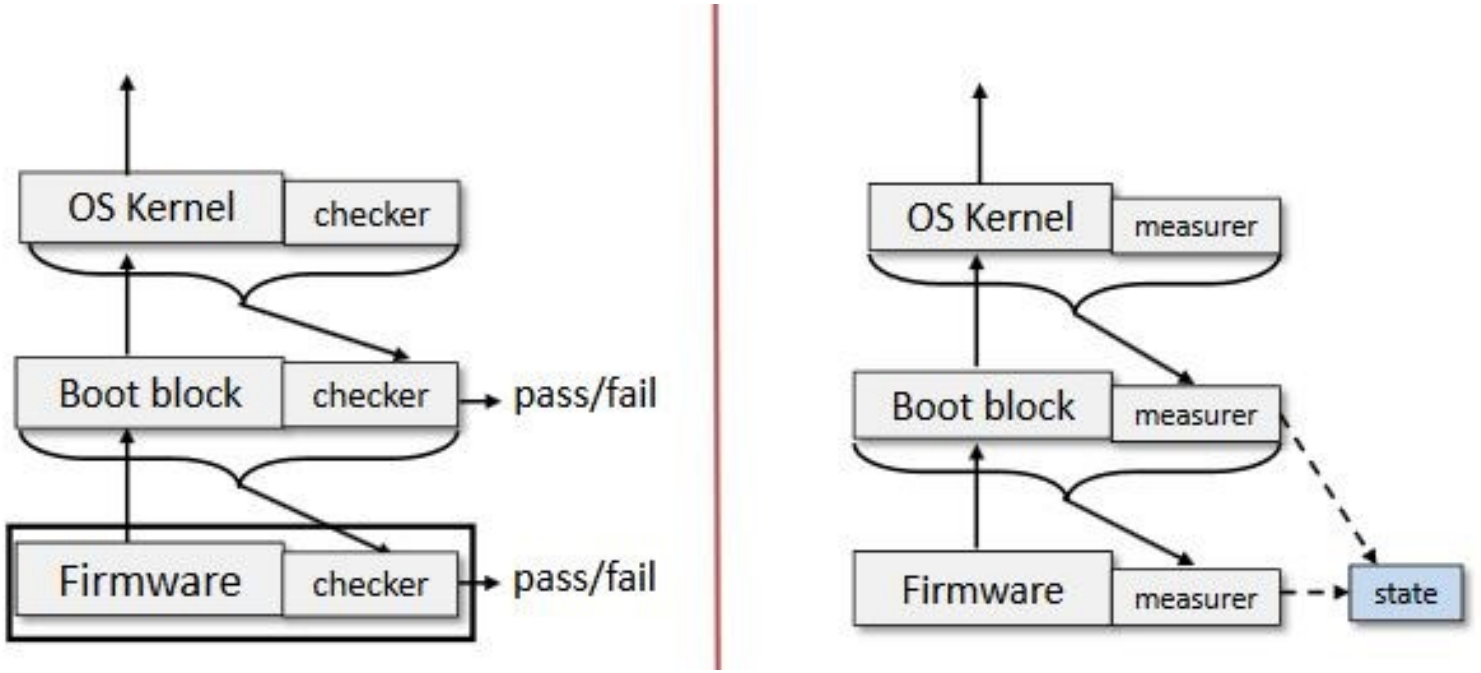
  including Microsoft, AMD, Intel, IBM, HP,....

- 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 information

- 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

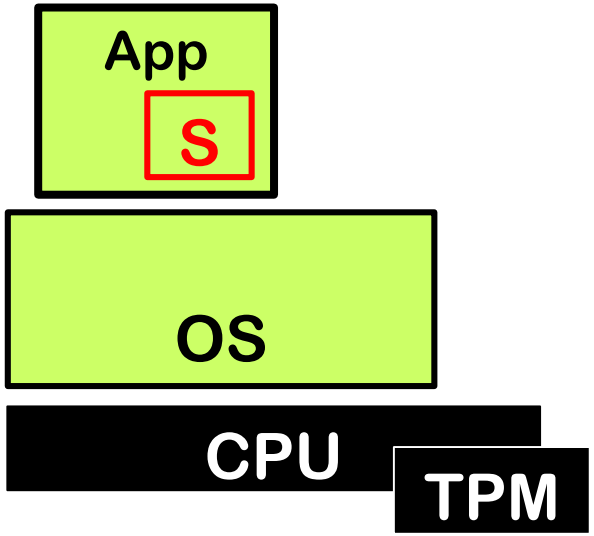
- Microsoft stopped development of NGSCB aka Palladium, their intended 'trusted OS' that would use the TPM, in 2004.
- TPM is still used for Bitlocker

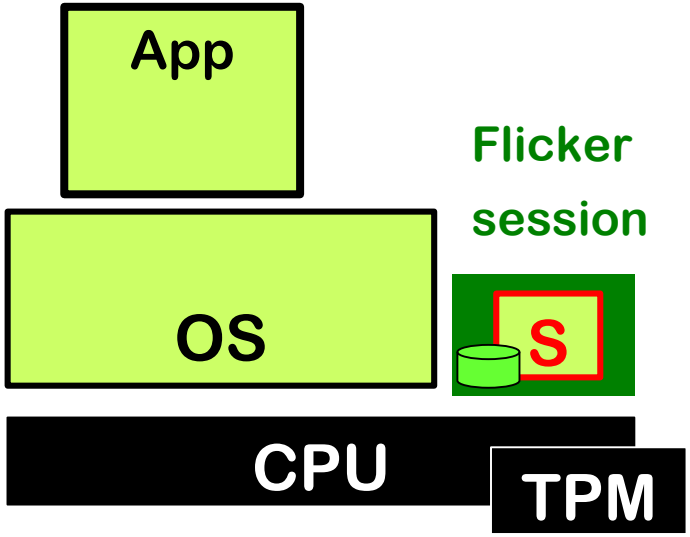
Flicker & SGX

-

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.

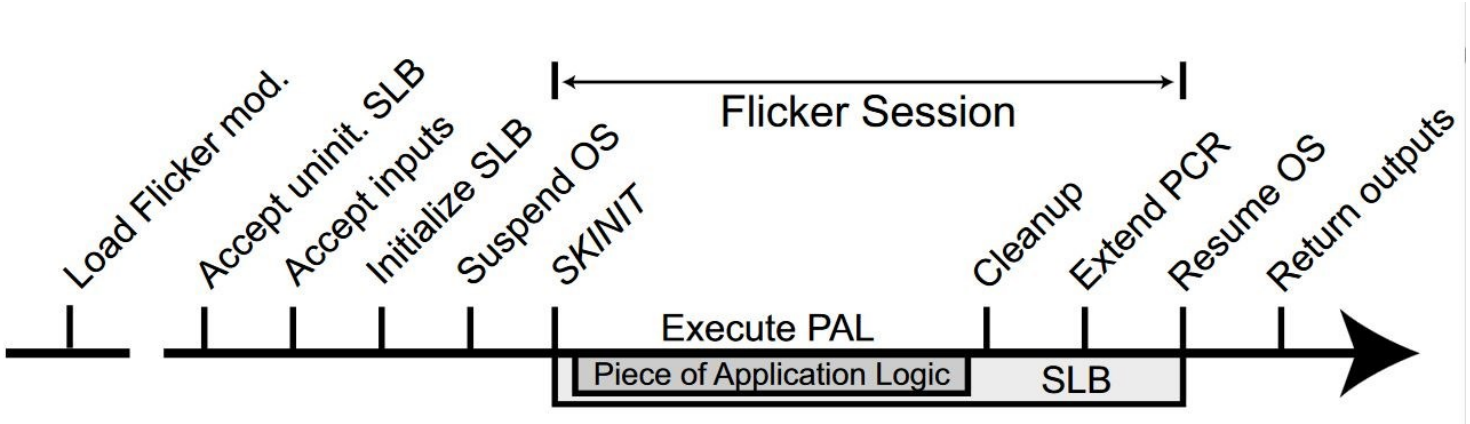
To switch to secure mode

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:

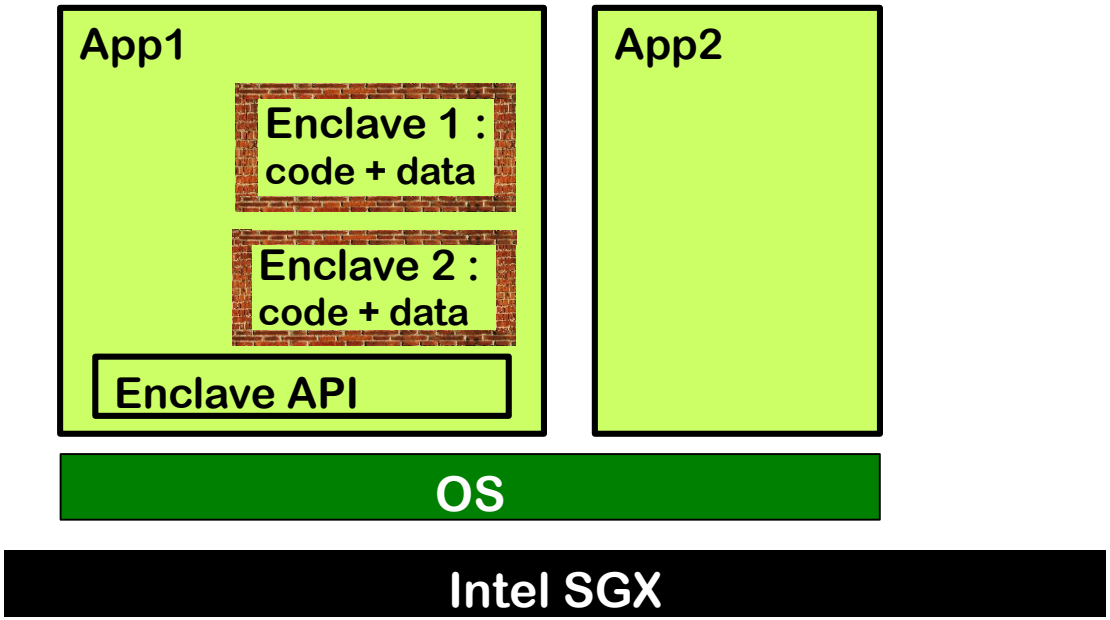
- running some crypto code with access to key material in sealed memory
- a password check with access to password

[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
- *Can provide Trusted Path in combination with Intel IPT?*

- 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

Intel ITP

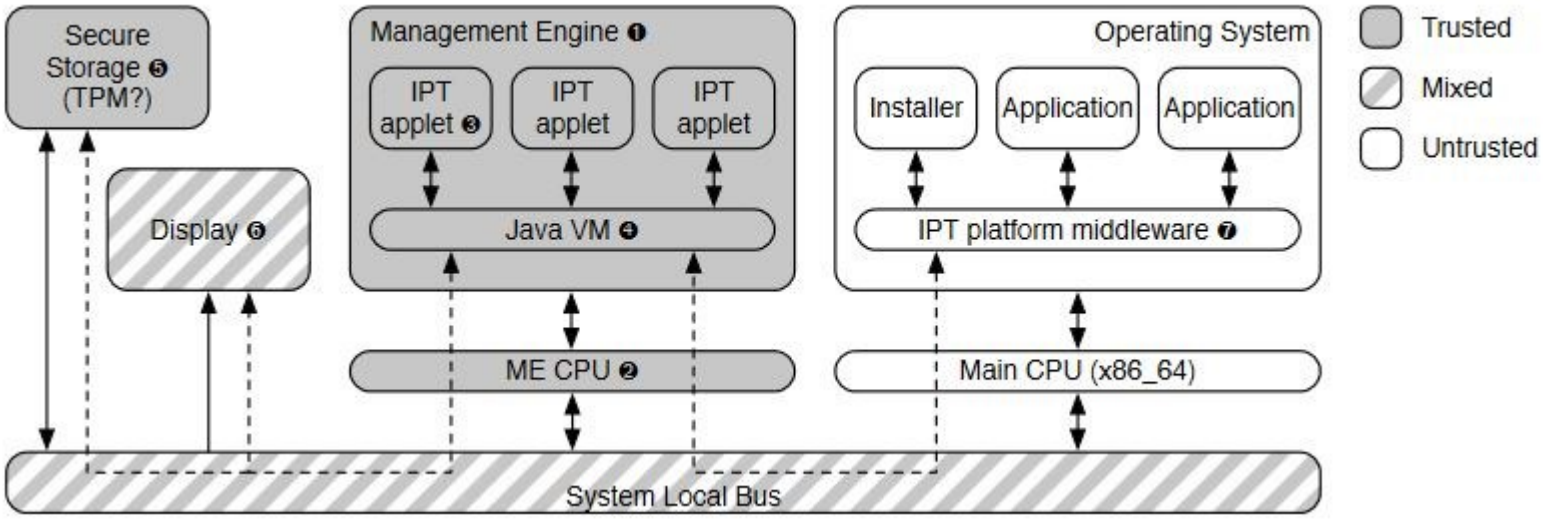
-

using separate processor for trusted execution

Intel IPT (Identity Protection Technology)

Separate processor providing a Java VM

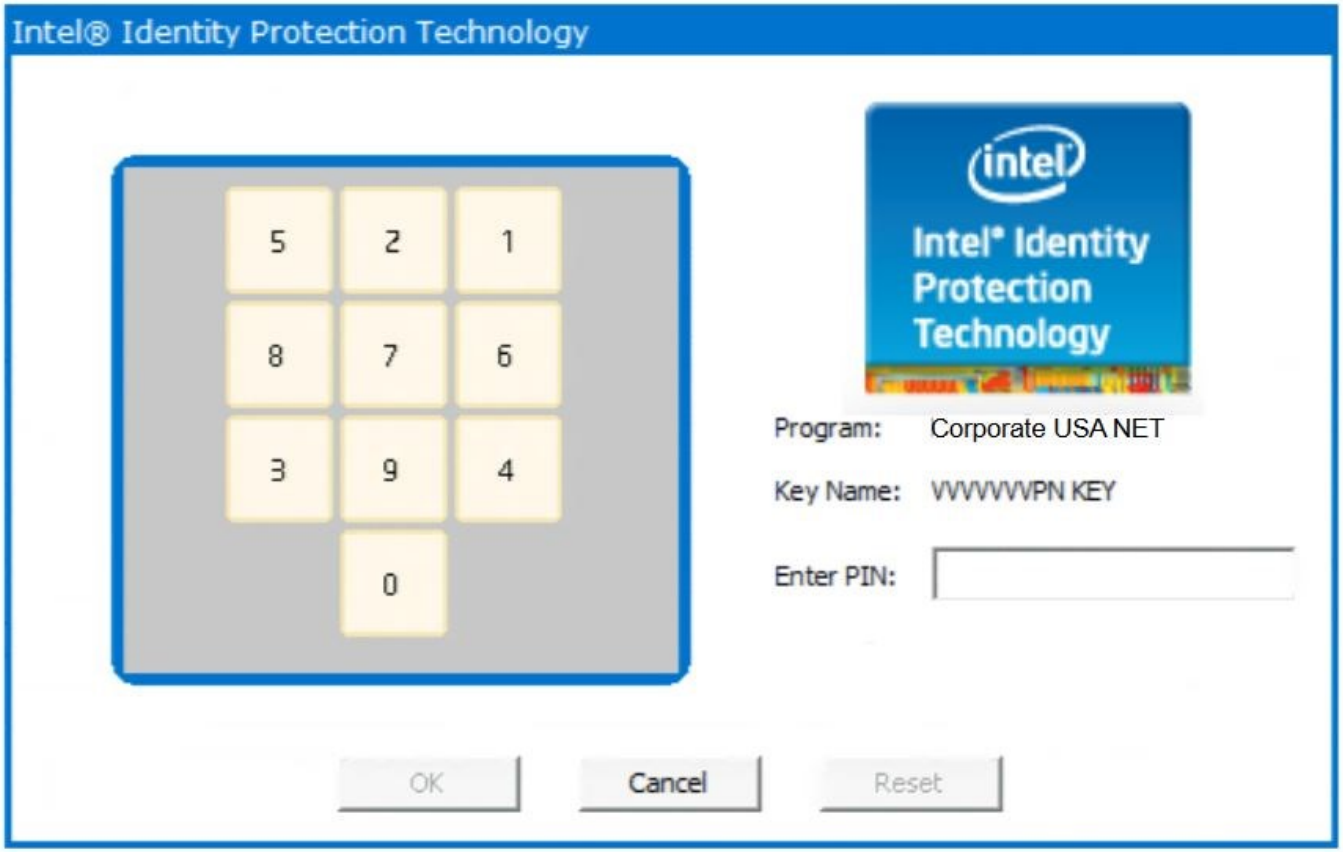
- can be used for crypto, incl. key storage & RNG
  - integrated with Windows Crypto API
  - example use case: One-Time-Password (OTP generation)
- also controls the display, so can provide Trusted Path



[source: van Rijswijk-Deij et al., Using Trusted Execution Environments in Two-Factor Authentication, Open Identity 2013]

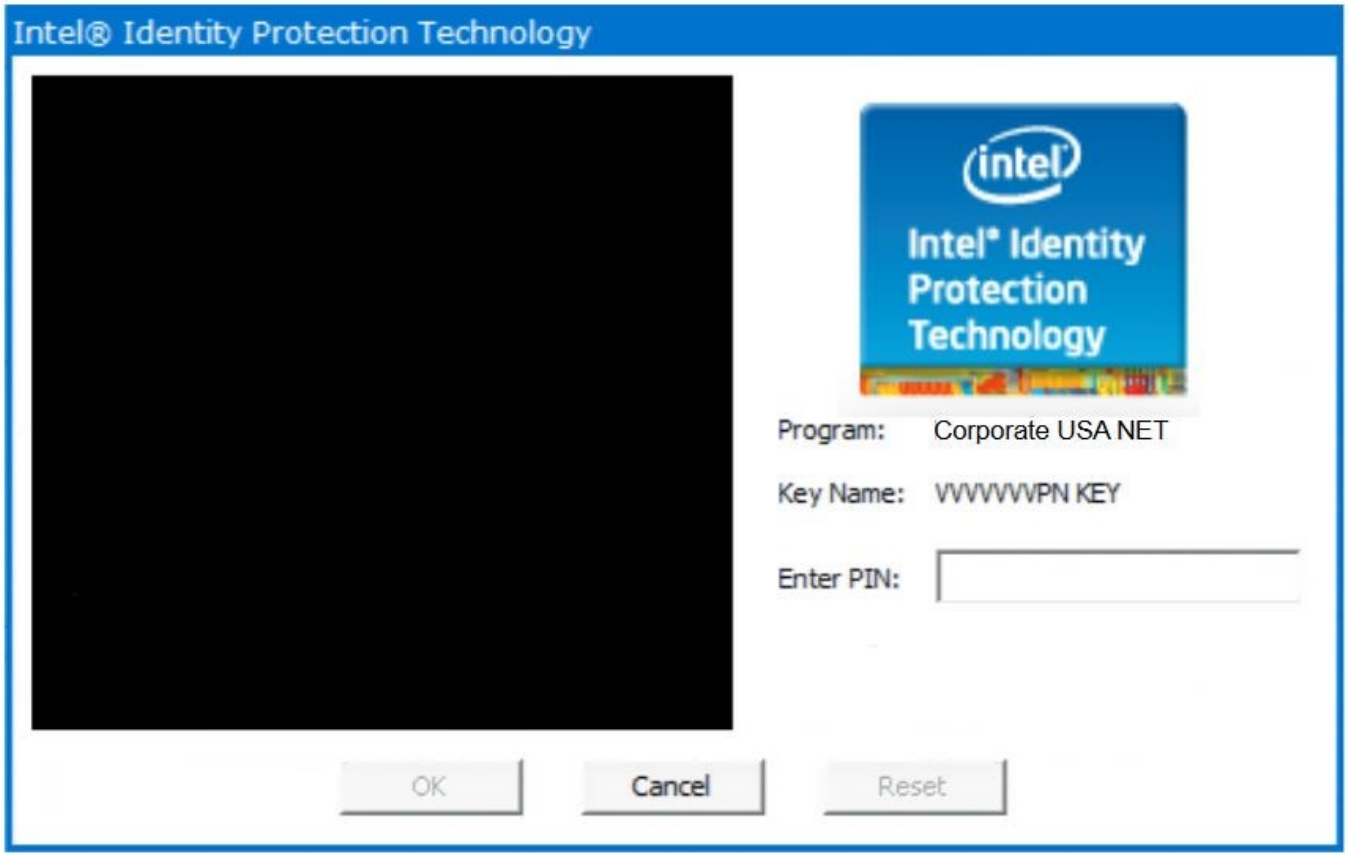
IPT for secure PIN entry

What the user sees



IPT for secure PIN entry

What malware on the main CPU would see



http://www.intel.com/content/www/us/en/architecture-and-technology/identity-protection/protected-transaction-display.html

ARM Trustzone

-

providing a secure & an insecure world

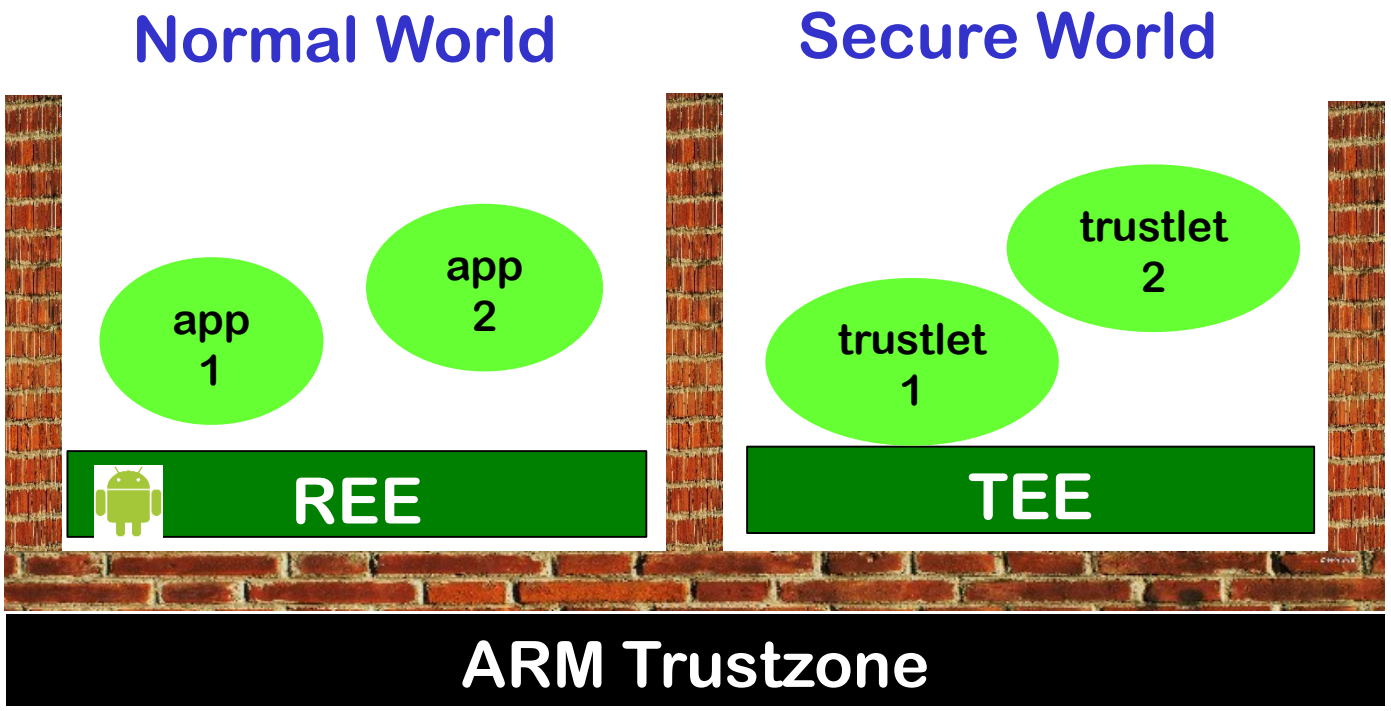
ARM TrustZone

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

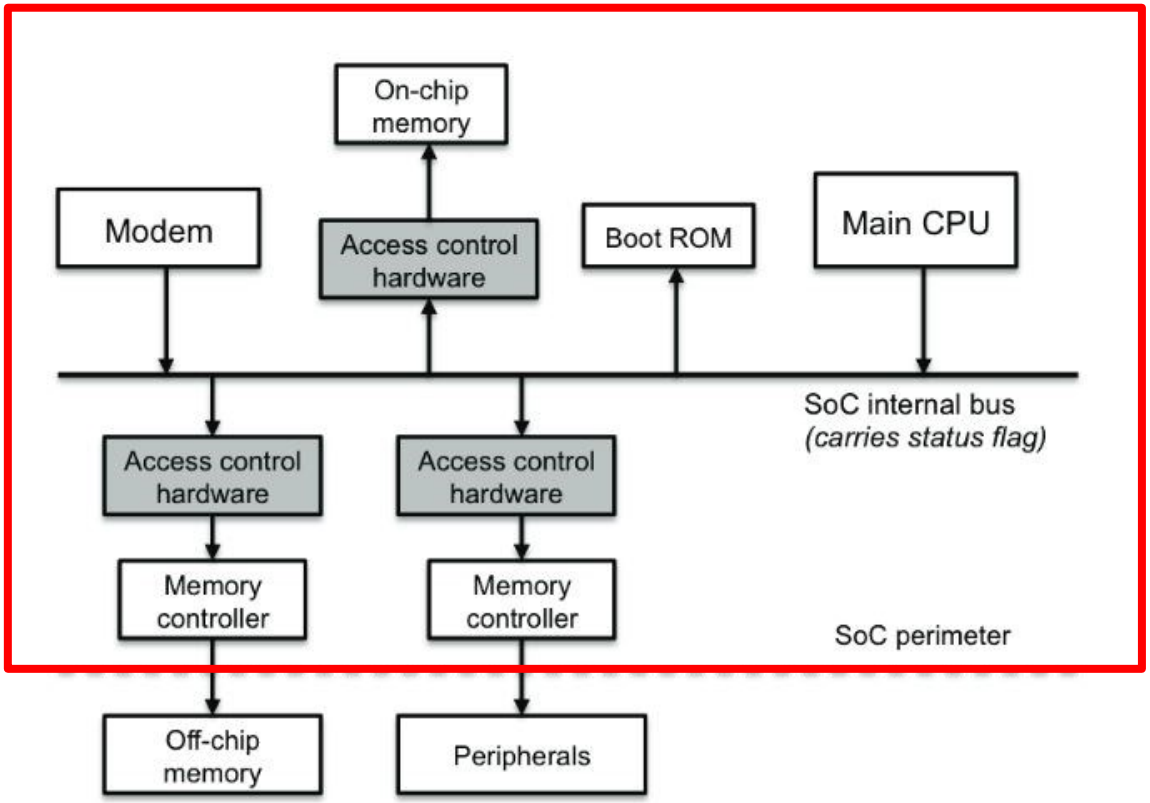
- 'normal world' and 'secure world'
  - Extra 33rd bit on the bus, to indicate the mode
  - Device could have an indicator (eg LED) for the mode
  - Separation of memory, peripherals, DMA, and interrupts
  - Context switch between worlds is slow

- 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

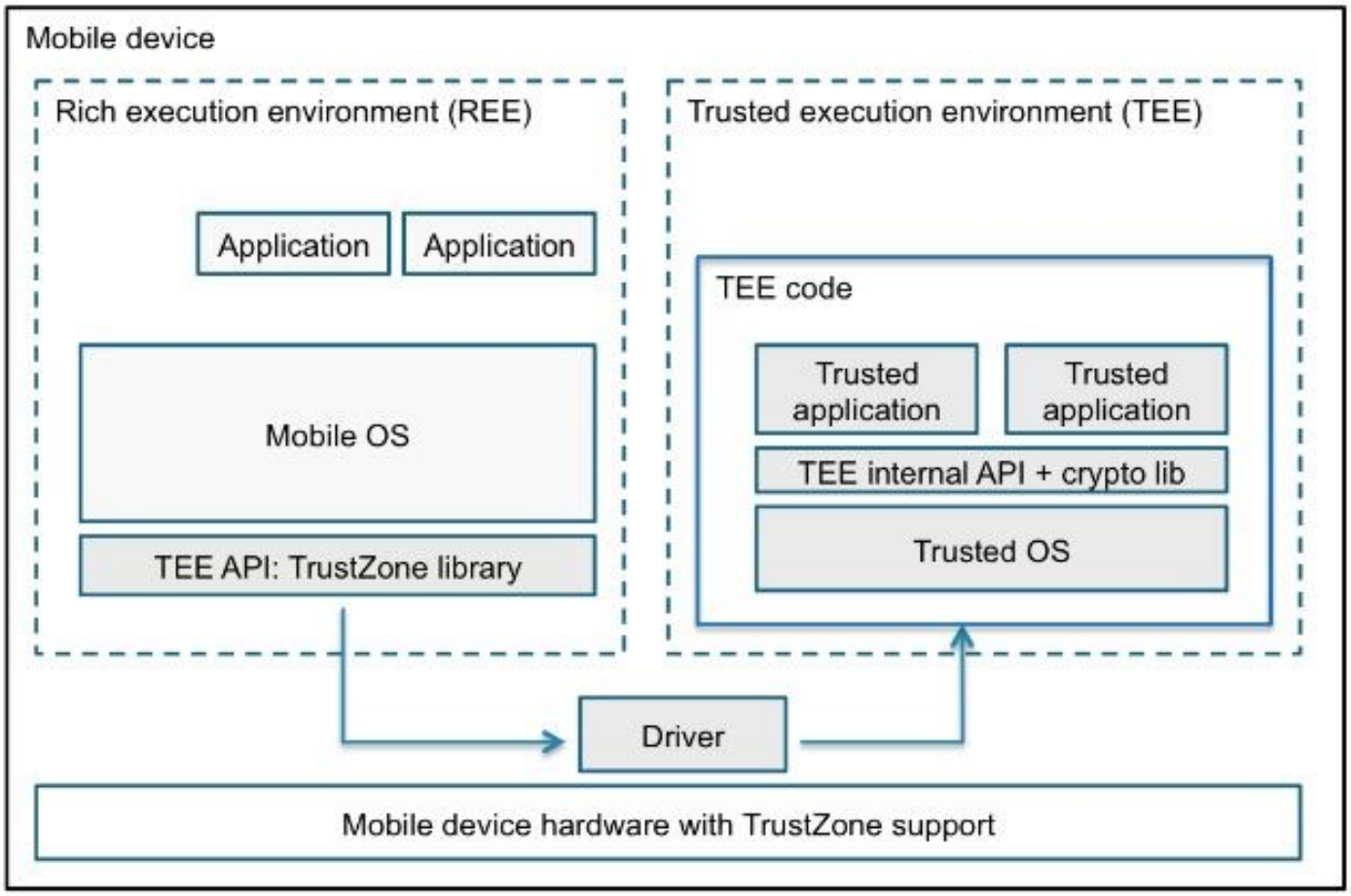


TrustZone SoC hardware architecture



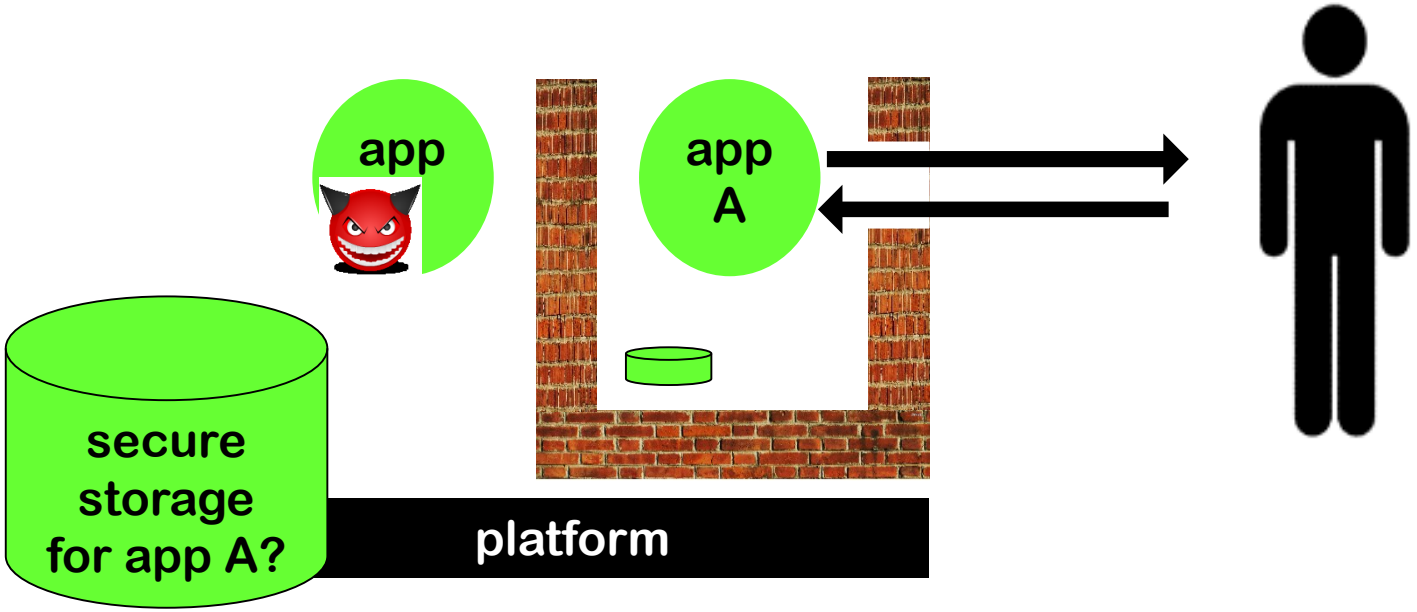
[source: Ekberg et al., The Untapped Potential of Trusted Execution Environments on Mobile Devices, IEEE Security & Privacy 2014]

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?



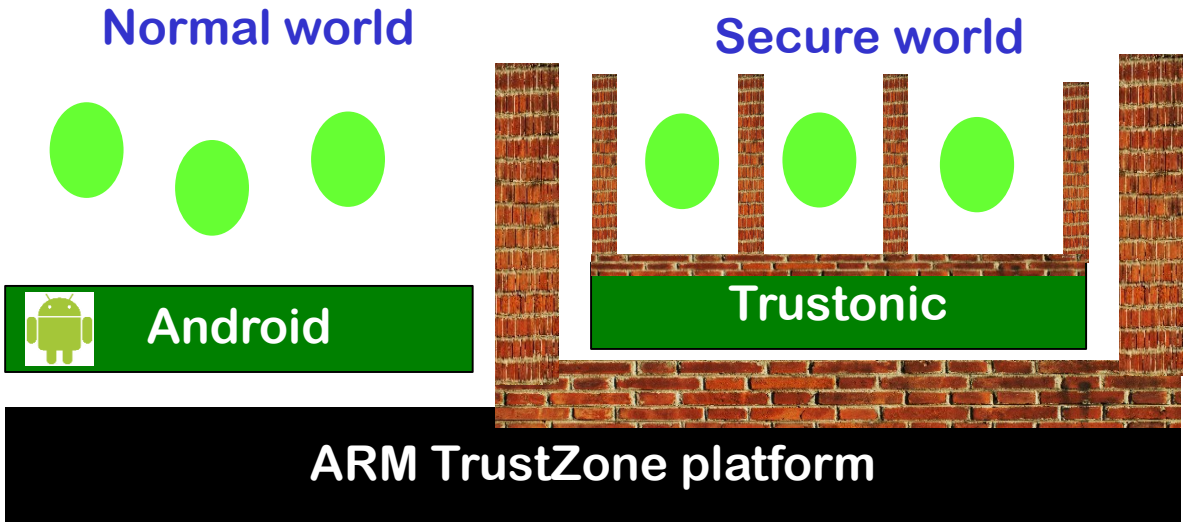
Persistent storage can be done in untrusted world, if we use encryption plus integrity & freshness checks.

Trusted app still needs some secure storage in trusted world

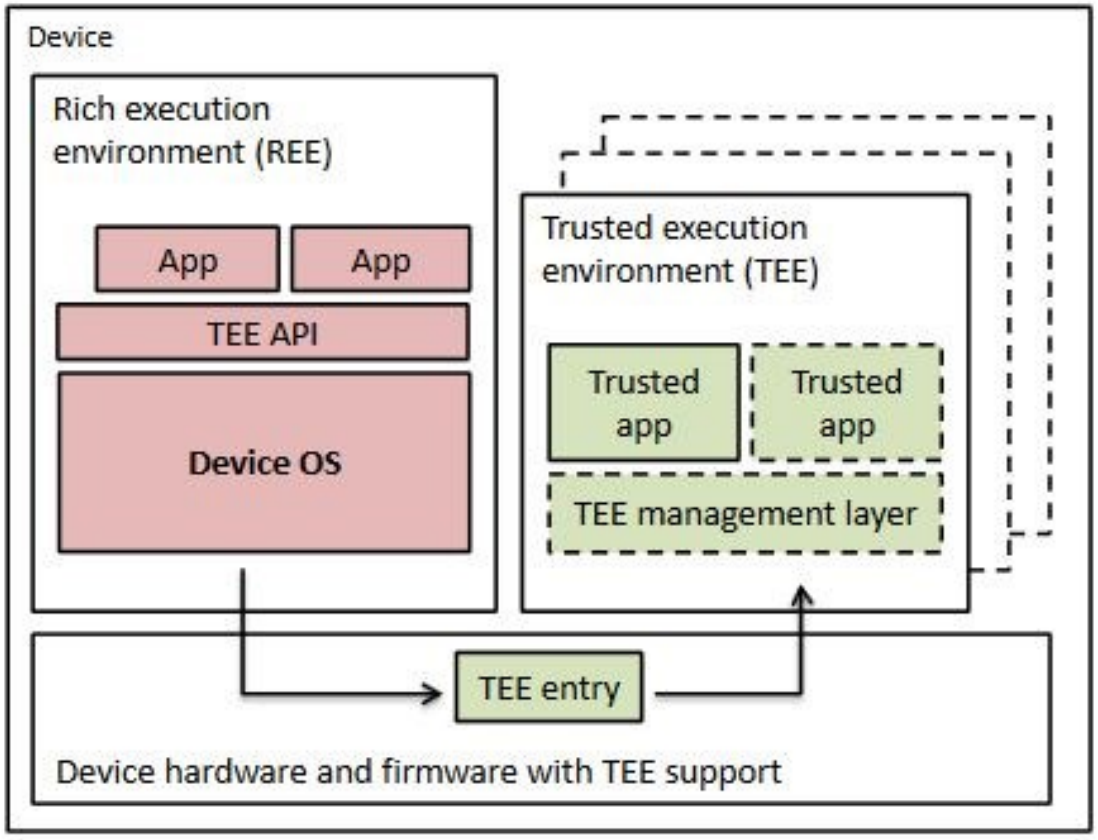
- for crypto keys for confidentiality & integrity
- for sequence numbers to ensure freshness (Data Rollback Protection)

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

Comparison & Conclusions

Performance & functionality

- TPM
  - slow
  - fixed & limited functionality, not programmable
- Flicker
  - trusted code runs on the main CPU
  - no access to OS or drivers, so no/very limited I/O
- IPT
  - more powerful than TPM & programmable, but less powerful than main CPU
- Trustzone and SGX
  - TEE code runs on main CPU
    - for Trustzone RAM may be limited
  - Trustzone provides 1 secure & 1 insecure world, SGX provides *many* secure enclaves

no trusted path!

with *some* trusted path, in *some* configurations

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 ☹

Openness, control, privacy

- TPM specs are open & standardised.

  Solutions like SGX, Trustzone, IPT are vendor-specific
- IPT, Trustonic, KNOX are 'closed'

  any code running in TEE requires permission from Intel, Trustonic or Samsung
- Who is in control?

  the chip manufacturer (eg Intel),
  the handset manufacturer (aka OEM, eg Samsung)
  the OS vendor (eg Google or Apple),
  the telco (eg Vodaphone),
  the user ???

  Even if solution is technically interesting, conflicting interests may prevent take-up & limit use.
- Unique device IDs and certificates held in TEE are privacy concerns

Conclusions

- TEEs offer interesting possibilities
  - for Isolated Execution, Secure Storage, Trusted Path and Remote Attestation
  - smartcard-like levels of security inside a normal processor
- 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 not use any special hardware features?
    - eg mobile payment solutions on Android do not use any hardware security measures, but use rapid key turnover instead (aka EMV Tokenization)
- Economics rather than security may (will?) determine success or failure of solutions

References

- Murdoch, Introduction to Trusted Execution Environments, slides
- Felten, Understanding Trusted Computing, IEEE Security & Privacy 2003
- Ross Anderson's Trusted Computing FAQ http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html
- McCune et al., Flicker: An Execution Infrastructure for TCB Minimization, EuroSys 2008
- Vasudevan et al., Trustworthy Execution on Mobile Devices: What Security Properties Can My Mobile Platform Give *Me*?, Trust 2012
- Ekberg et al., The Untapped Potential of Trusted Execution Environments on Mobile Devices, IEEE Security & Privacy 2014
- Malware Guard Extension: Using SGX to Conceal Cache Attacks, https://arxiv.org/abs/1702.08719
- Cerdeira et al, SoK: Understanding the Prevailing Security Vulnerabilities if TrustZone-assisted TEEs, IEEE S&P 2020.
- van Rijswijk-Deij and Poll, Using Trusted Execution Environments in Two-Factor Authentication, Open Identity 2013
- Android 11 Compatibility Definition, Section 9.11.2 StrongBox https://source.android.com/compatibility/11/android-11-cdd.pdf