Trusted Environments for Future Consumer Devices

Dr. Jan-Erik Ekberg

Assure Workshop

25.4 2023

- What is a TEE? Why is important?
- Hardware Evolution
- Attestation
- Runtime(s) and Code Development
- Opportunities and Research

Secure Environments in Mobile Phones – since 2008 or so

- Hardware-assisted isolated computing (Processor Secure Environments) came to be via a few detours through a technology called **Arm TrustZone**. This was a time where nothing similar had been deployed yet. Argument: **Costs less than 1 cent**.

- As smart card (provisioning) and smart card terminals was standardized by the **GlobalPlatform** consortium, the same organization undertook the standardization of these new secure environments – named Trusted Execution Environments (TEEs)

Properties of the GP TEE architecture (with ARM TZ)

- Trusted applications are primarily **authorized by origin**. I.e. Only applications from trusted / approved source (signed by an approved party) are allowed to run.

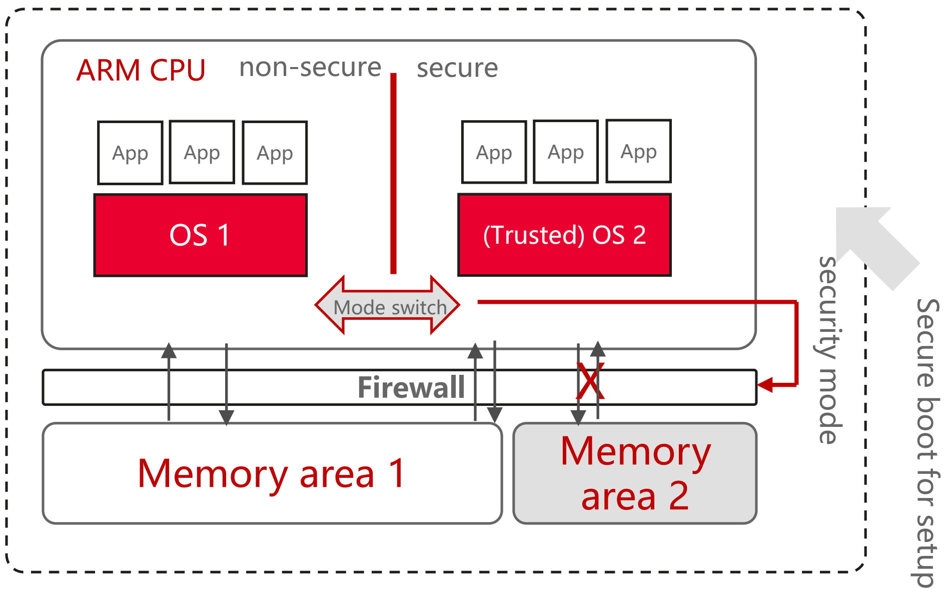
- The **TEE kernel is a singleton**, i.e. there is a secure mode that at large mirrors the standard OS. All trusted apps run on the same kernel, and e.g. secure drivers and interfaces are managed by the TEE kernel

- With few exceptions, **TEE implementations are passive**, i.e. they are invoked by a control thread from the rich enviroment, and run only as long as needed. In theory the architectur does not inhibit active operation.

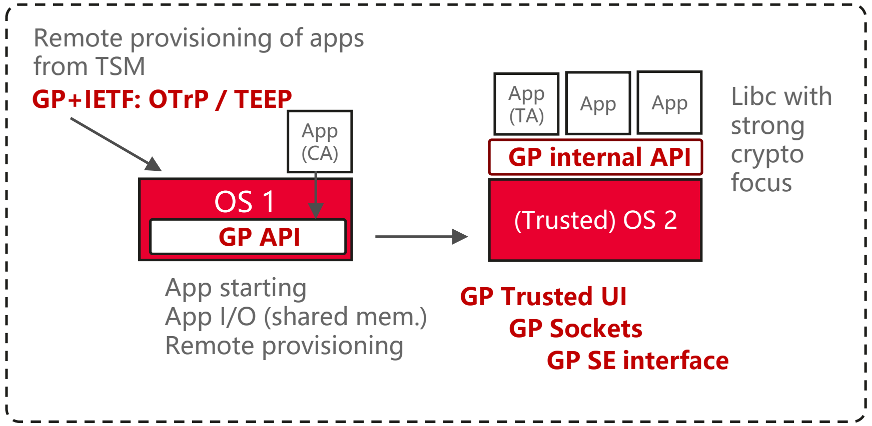
- The architecture (e.g. for I/O) assumes that **secure mode has non-secure memory access**. Also no memory encryption was considered when this standard evolved.

- All code and interfaces are C. **No binary compatibility** (across manufacturers)

Processor Secure Environment



GP TEE Architecture

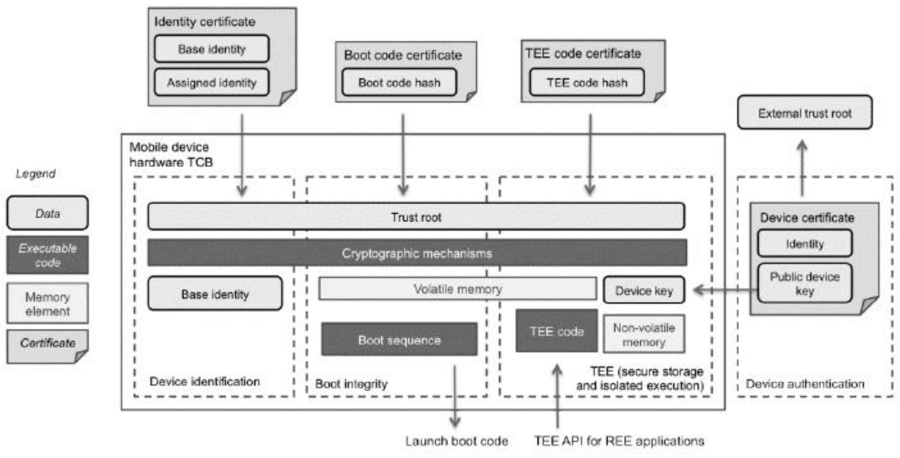


… and the use of TEE can be considered a success in phones

With **Billions of devices** deployed and in use, TEE is one of the most used secure environments in the world, overshadowed by smart cards only. However, the ecosystem aspect – ability for 3rd parties to **develop and deploy their own** secure applications has never materialized, and likely will not any more.

Services run from / with the TEE

- Operator-related services like SIMLock, IMEI, OMA DRM
- Android hardware-backed keystore and keymaster (3rd party support)
- TenCent / WeChat Pay -- 500 million active TEE customers
- Most manufacturers rely partially on TEE for system protection like run-time protection, attestation, KIP / HKIP / EIMA ..



Ekberg, Jan-Erik, Kari Kostiainen, and N. Asokan.
"The untapped potential of trusted execution environments on mobile devices." *IEEE Security & Privacy* 12.4 (2014): 29-37.

GlobalPlatform conservatively estimates that there were 5 billion TEE-enabled processors worldwide within devices at the end of 2017.

https://globalplatform.org/wp-content/uploads/2018/05/Introduction-to-Trusted-Execution-Environment-15May2018.pdf

Riscure and Trustonic achieve the first EAL 5+ TEE certificate compliant with GP TEE PP

https://www.trustonic.com/news/riscure-and-trustonic-achieve-the-first-eal-5-tee-certificate/

TEE + SE

Since 2020, a combination of a TEE and an eSE / hardware trust root has become the norm in upscale phones – TEEs are not very side-channel resistant, and some services with keys (credit cards, secure boot, attestation) is better done with this combination (Apple SEP, Samsung eSE, Huawei iSE, MSP)

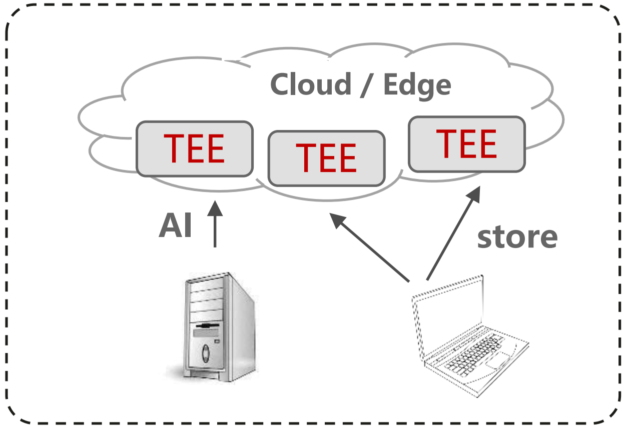
HUAWEI

TEE terminology

- **Trusted Execution Environments**, i.e. hardware-assisted isolated computing in the CPU cores, were first deployed in mobile phones, using Arm TrustZone. These environments in consumer devices (since around 2010) have always been called TEEs

- When Intel around 2015 launched SGX, for PCs and later in servers, they introduced the term enclave, or **Secure Enclaves**, for the same structure, i.e. where security-critical parts of a workload is run in hardware-assisted isolation. In servers, also the term **Secure Virtual Machines (VM)** has been used. Recent hardware technologies for TEEs include iTDX, AMD-SEV and ARM-CCA

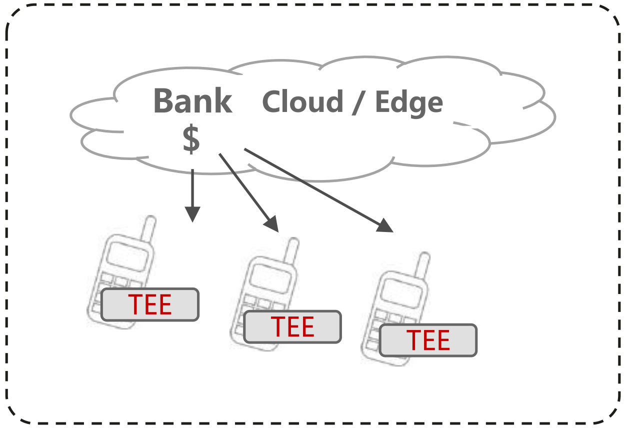
Confidential Computing



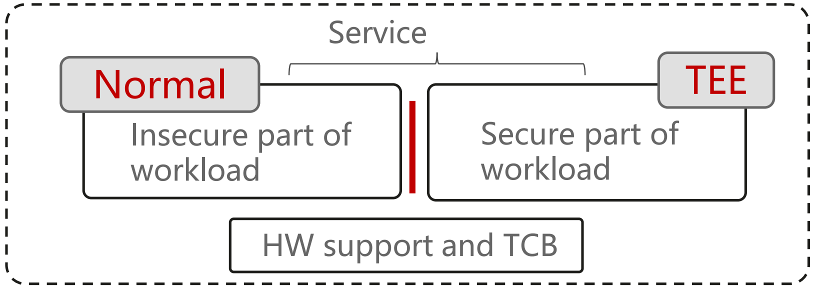
When TEE work is outsourced to the cloud, in a way where the cloud operator should not learn about the secrets used or the results produced – we call this confidential computing

Trusted Execution

When TEE work is outsourced to a consumer device (e.g. payment algorithm or eID) where the user should not learn about the secrets used or the results produced – we call this trusted execution



Fundamentally:
A cat is a cat is a cat



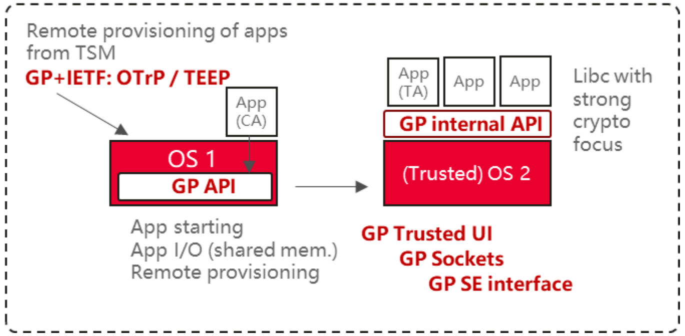
... but there is an argument for mobile phones to **leverage Confidential Computing TEE design patterns**

HUAWEI

So – why change a working setup ?

The GP TEE hardware and software stack **is showing its age** compared to alternative solutions:

GP TEE Architecture



- Intel SGX (user-space enclaves) showed **that 3rd-party ecosystem building is possible**. Yes, it had its problems, but spawned much more recognition, research contribution and eventually also attack contribution.
- After GP/A-TZ and iSGX, most enclave solutions choose to assert **code integrity based on code attestation** rather than on origin validation.
- SGX and more modern VM-based enclaves show that the **Trusted Computing Base (TCB) of a secure workload** can be almost **completely done in hardware**. In GP setups, the OS kernel code can easily be **several 10s of MB,** and this is a huge attack surface (almost all published TEE attacks rely on bugs in the TEE software)
- The **memory allocations of ARM-TZ and GP-TEE are static**, since firewalls, memory assignments etc. are fixed at boot. Ideally, memory is consumed by secure workloads only when they are running / used. With more flexibility, also more complex secure workloads can be implemented (machine learning, face recognition)

GP+ ARM TZ TEE

- Gigantic **TCB** (10+ MB)
- No memory encryption
- Context switch latency high
- Memory statically reserved
- TA resources limited

(Working HW driver support – NS bit on bus)

- Minimal **TCB**
- Isolation with memory encryption
- Low latency context switch
- Memory reserved as needed
- TA resources almost unlimited

(how to do drivers? Trust mechanism? Do we need to revisit sofware stack?)

Ideal end state ?

What can we learn from CC ?

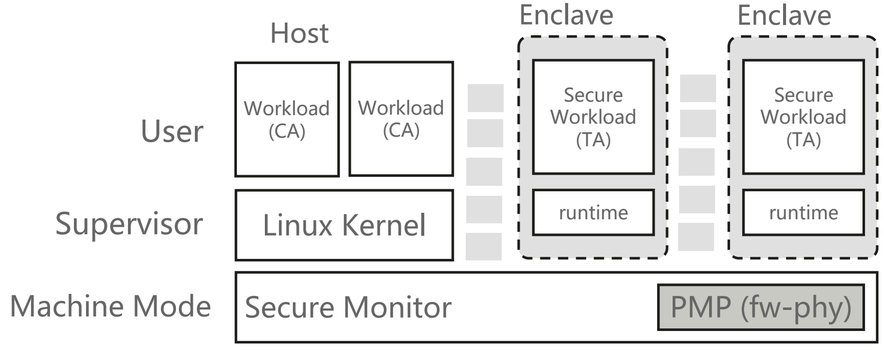
In this talk I argue for a new TEE model also for consumer devices!

Hardware

Hardware support "today"

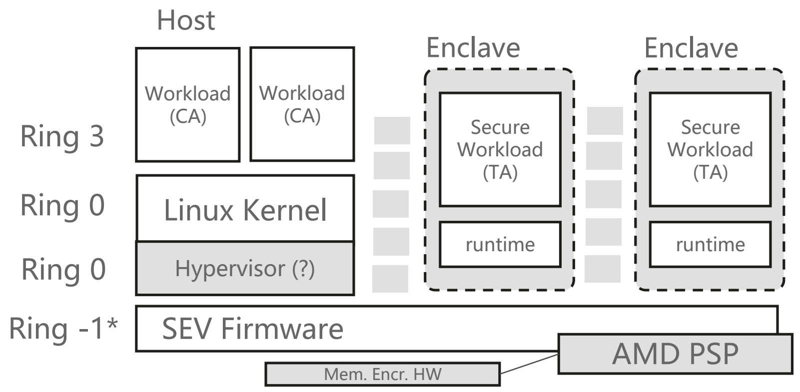
ARM & RISC-V most likely consumer device cores

RISC-V



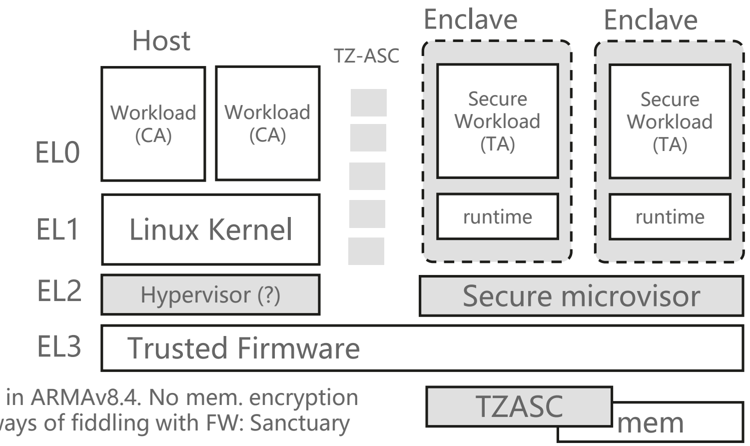
The **Keystone** project (MIT, Berkeley)
No memory encryption
Virtualization Extensions (Hypervisor) coming
Drivers with Composite Enclaves *)
Or more full-scale enclave overhaul with CSEE **)

AMD-SEV



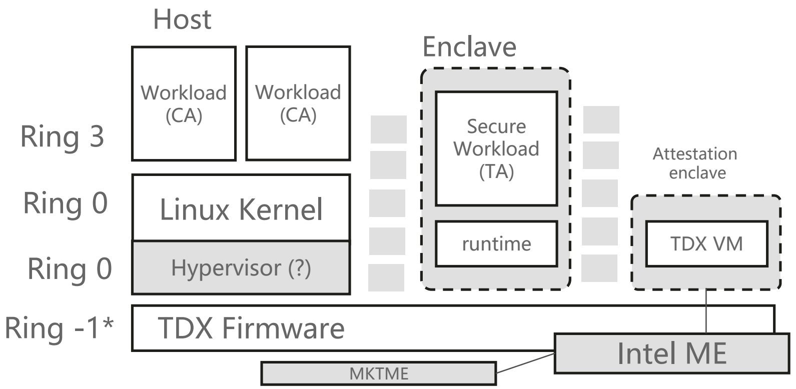
Trust roots / mem encryption tweaks. Max 16/32 tweaks
PSP originated device / enclave attestation

ARM 8.4



SEL2 added in ARMAv8.4. No mem. encryption
***) Other ways of fiddling with FW: Sanctuary

Intel TDX



Attestation moved to a BIOS-setup enclave
Infinite number of enclave keys

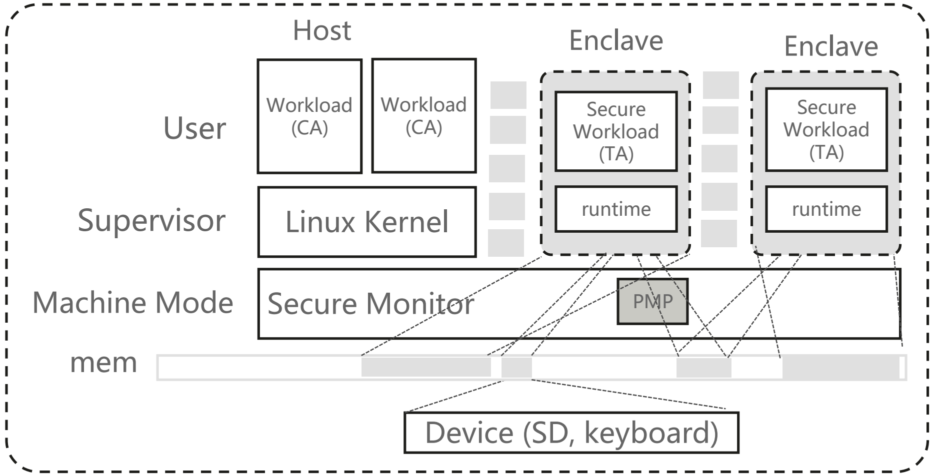
These solutions come to the table from cloud / Linux CC perspective

*) ETH Zurich: Composite Enclaves (IACR Trans on CH&ES) https://arxiv.org/pdf/2010.10416.pdf
**) TUDarmstadt (Usenix 21): CURE https://www.usenix.org/system/files/sec21-bahmani.pdf
***) TUDarmstadt (NDSS 19): Sanctuary https://www.ndss-symposium.org/wp-content/uploads/2019/02/ndss2019_01A-1_Brasser_paper.pdf

Learnings from (RISC-V) Enclave Research

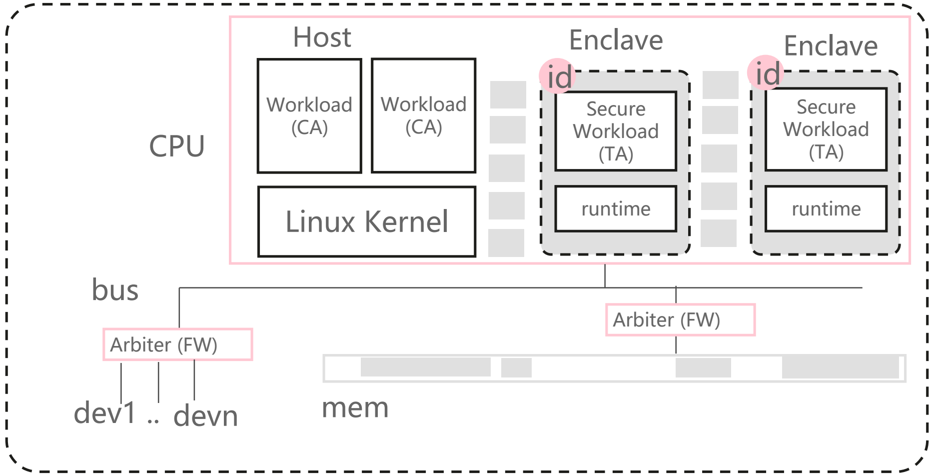
Composite Enclaves *)

- We need some ways to do drivers in VM enclave settings
- Composite enclaves focuses on exactly this – how do we segment physical memory in a secure way for communicating enclaves and enclaves communicating with external hardware
- The ´composition´ comes from several enclaves serving as one



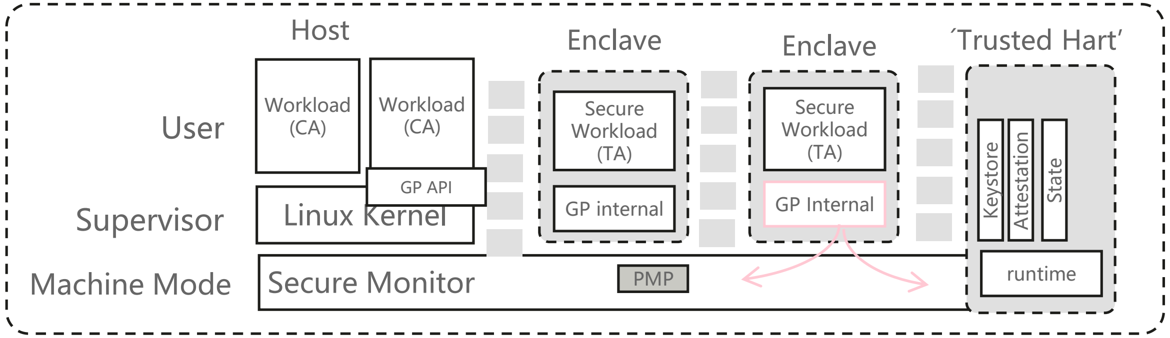
CURE **)

- Holistic, hardware-based enclave design
- Based on enclave idenfication in CPU, caches and firewalls (arbiters) cooperate
- Allows different kinds of enclaves to be constructed (user-space, VM, kernel, ..)
- Side-channel protection (cache-way separation) + mem. encryption integrated



Trusted Hart *)**

- Software forward-port of the GP-TEE ecosystem onto an enclave scenario
- Define the role of the trusted component (RoT) as a provider of attestation, keystore and access control arbiter
- The HW RoT approach has been prevalent in mobile devices for several years (Apple SEP, Huawei MSP, ..), and is visible also in CC (ARM HES, AMD PSP., ..)



All of these works put high emphasis on the role and function of attestation. We will come to that later

*) ETH Zurich: Composite Enclaves (IACR Trans on CH&ES) https://arxiv.org/pdf/2010.10416.pdf
**) TUDarmstadt (Usenix 21): CURE https://www.usenix.org/system/files/sec21-bahmani.pdf
***) Huawei (TrustCom 2022): Trusted Hart https://arxiv.org/pdf/2211.10299.pdf

HUAWEI

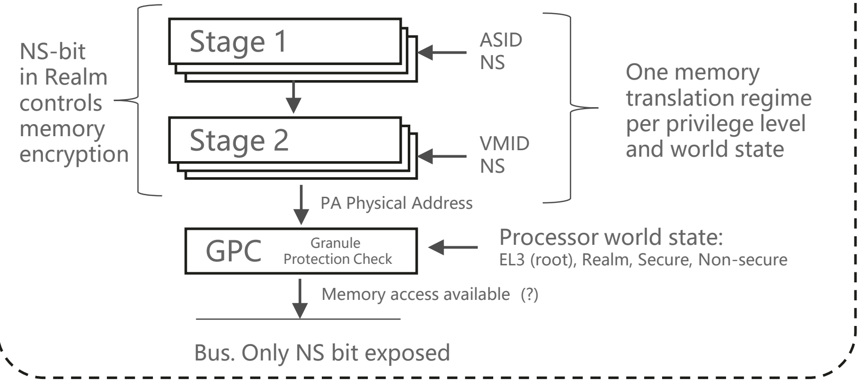
And for Mobile Phones – the N.G. architecture, ARM CCA

- An architecture that to large extent follows in the footsteps of AMD-SEV and Intel TDX. I.e. to provide isolation + VM memory encryption to isolate from access. Logically, this extends dual-world TZ to a three(+one) world arrangement
- Contains a *root mode*, i.e. EL3. Since translations and GPC are memory based, nothing else makes really sense. Rest of the required isolation can be made by translation rules and protection of the translation tables.
- Not yet clear how this extends to the device regime. NS bit is already on the bus, virtual drivers might be achievable via secure world, but DMA protection is not part of CCA for now.
- Trust roots inherited from ARM PSA architecture / HES is an abstract hardware trust root

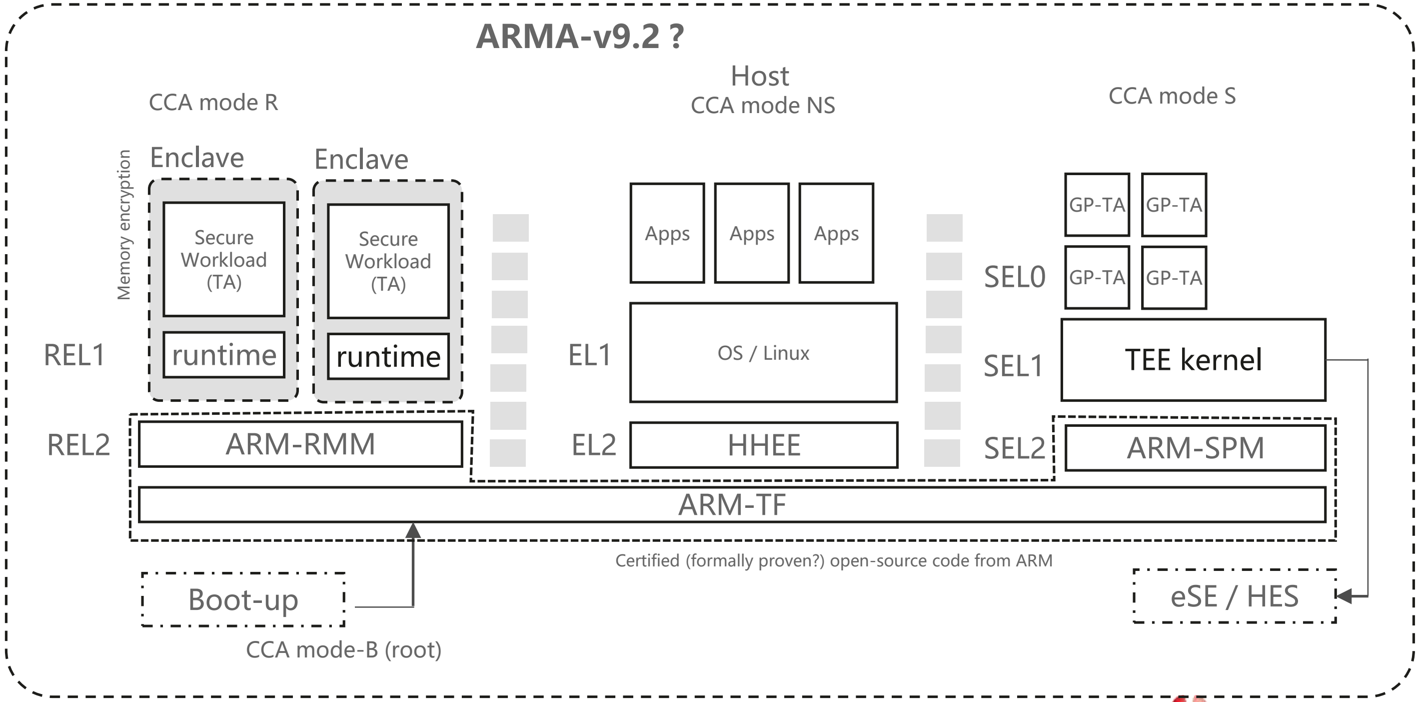
Hardware Architecture

GPC also done at DMA controller. Adress domain specific encryption done in memory controller

Security State/PA space	Non-Secure PA	Secure PA	Realm PA	Root PA
Non-secure	Allow	Block	Block	Block
Secure	Allow	Allow	Block	Block
Realm	Allow	Block	Allow	Block
Root	Allow	Allow	Allow	Allow



Functional Architecture



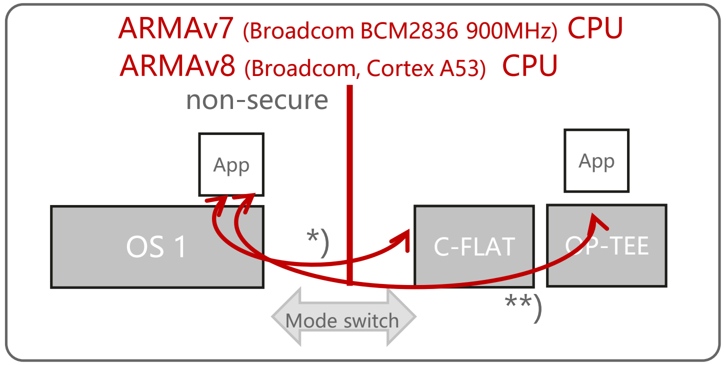
HUAWEI

Linaro ARM talk (21): https://static.linaro.org/connect/armcca/presentations/CCATechEvent-210623-CGT-2.pdf
J Weidner, CCA overview: https://sys.cs.fau.de/extern/lehre/ws22/akss/material/arm-cca.pdf

About Performance

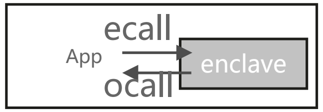
ARM-TZ

- C-FLAT *); Bare-metal measurements for Arm TrustZone only including context switch, related register stores, TLB flushes and minimal logic: 237 µs per transaction. Measured at 900 MHz. I.e., no TEE OS used
- Secure Edge Computing **): OP-TEE NOP operation (GP-TEE interface, ping-pong, 82 µs per transaction. Measured at 1.2 GHz



Intel SGX

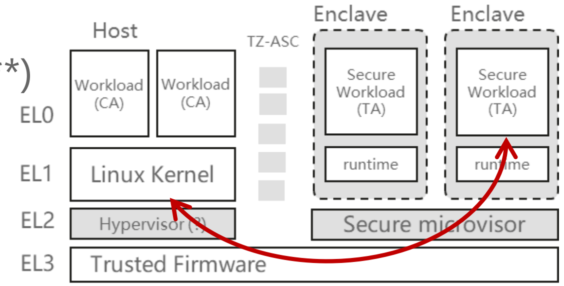
- On SGX perf ***); Ecall + Ocall every 16000 instructions → overhead 120% (7000 cycles/ecall) Baseline transactional measurements with no memory use (since SGX encrypts memory, increased bandwidth slows down operation). Similar measurements ****) puts an empty Ecall at the same ballpark: 9.3 µs



Intel Core i5-6200U SGX, 2.4GHZ

VM-based

- AMD-SEV measurements on am 2 GHz AMD EPYC 7251 for emulating SGX on SEV ****) (which is not an ideal comparison) puts enclave entry latency at around 100-200 µs
- Huawei measurements with Google Hafnium microvisor + FreeRTOS runtime + WASM interpreter (Kunpeng 920 / ARMAv8, 2GHz): 6.4 µs, stddev 1.7 µs



*) TUDarmstadt (SigSac 19) https://www.usenix.org/system/files/sec21-bahmani.pdf
**) Tromsö Uni (IoTBDS 21) https://pdfs.semanticscholar.org/fa08/a6499998b83d548f8dc079343dbcdd767474.pdf
***) Tsinghua Uni (WISA 16) https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7878255
****) Ohio State (IEEE S&P 22) https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9833694

HUAWEI

Code Provisioning / Attestation

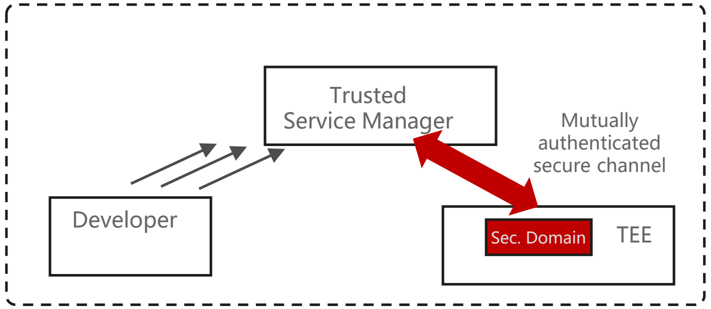
Secrets Provisioning (in GP)

The GP standards (TEE and smart cards) have successfully followed the following provisioning paradigm:

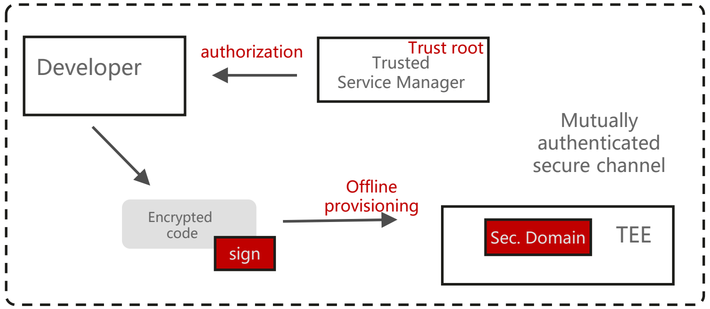
- Endpoint devices are not necessarily uniquely identifiable
- Endpoint devices come pre-provisioned with a security domain, i.e. secret key material to which code / data can be encrypted. Sometimes identification is bound to key injection with the domain
- Endpoint devices come pre-provisioned with trust roots, and TEE execution and data injection is conditioned to trust root signatures. For symmetric-key security domains (sec.dom==trust root)
- Applications have a UUID, that e.g. match them to their respective storage. GP UUIDv5 is code-hash based

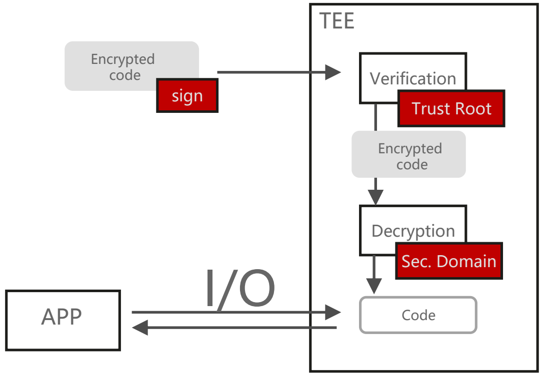
Business logic: **Trusted applications (TAs) are vetted in source (certified), and only certified binaries and secrets (signed by trust root) are allowed to run in TEE**

OTrP



TMF – ASN.1 (PKI-style)





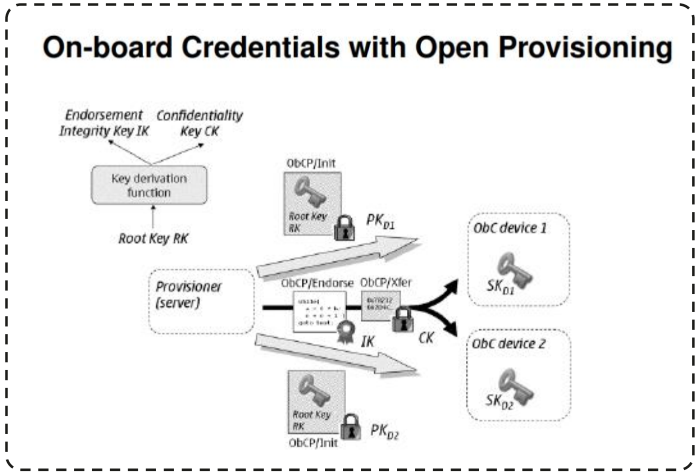
TEE Management Framework: https://globalplatform.org/wp-content/uploads/2018/06/GPD_TEE_MgmtFramework_v1.0_PublicRelease.pdf
Open Trust Protocol: https://globalplatform.org/specs-library/tee-management-framework-open-trust-protocol/

Secrets Provisioning Going Forward

Neither GP-TEE 3rd-party development OR SGX deployment models have really taken off, one can potentially attribute this to the business model, which essentially is closed. Developers need authority interaction to get their code deployed, and the authorities are fragmented across ecosystems

Confidential Computing solutions predominantly stock an open provisioning model, in the following spirit:

- Endpoint devices are able to provide platform attestation up to and including the TEE workload – to anyone who asks. OEM/ODM is the trust root
- Endpoints provide storage or TEE-workload-local keys for all workloads
- Workload identity is in practice the TEE workload hash / imprint

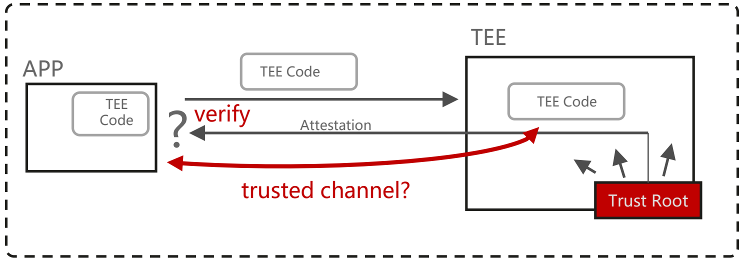


The open provisioning model dates back to around 2010

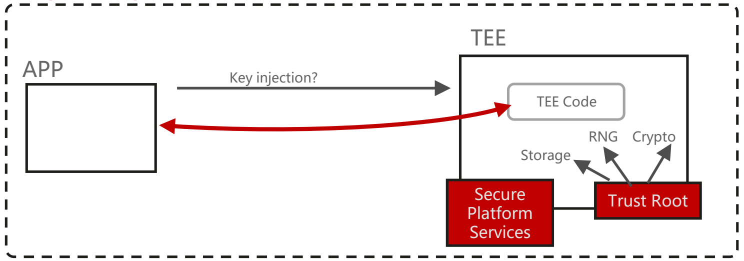
ObC Nokia, AsiaCCS 2009: https://dl.acm.org/doi/pdf/10.1145/1533057.1533074

Business logic: **Anyone can run a TA in a TEE. TEE sandbox is strong enough to curtail attacks from TA. The source of the TA is responsible for attesting viability of TEE platform. TEE trust roots are provided by platform and implicitly trusted**

Step 1: Establish Context



Step 2: Operate

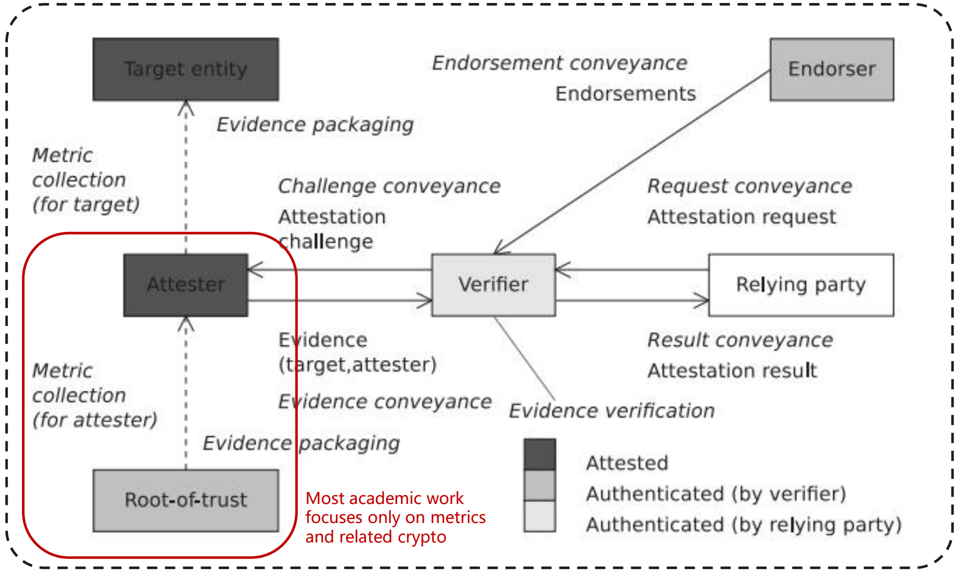


Trusted Sockets Layer (NordSec 21) https://events.tuni.fi/uploads/2021/12/fc3718db-nordsec21-paper7-slides.pdf

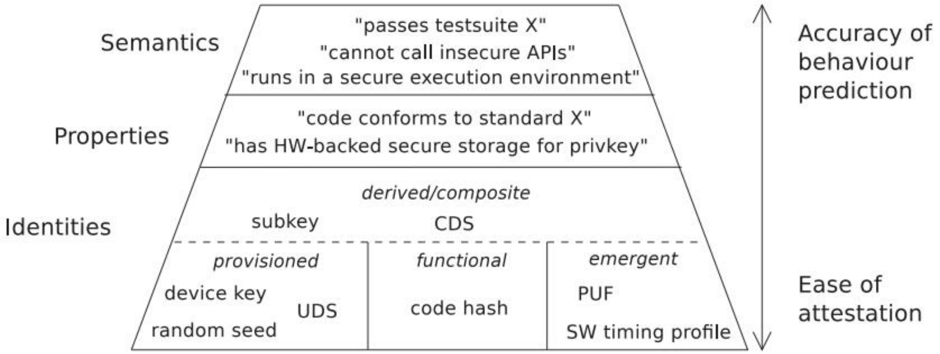
The Problem(s) of Device / TEE Attestation

For closed system attestation, decades of academic work is available to find out the optimal attestation metric, method and security level. However, what is missing for TEE is a unified method that can work across devices, device types and trust roots

The IETF RATS (+TCG) attestation framework

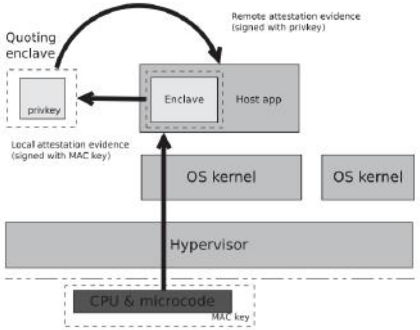


Possible metrics

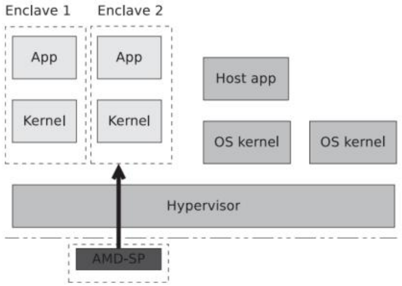


Architectural Differences

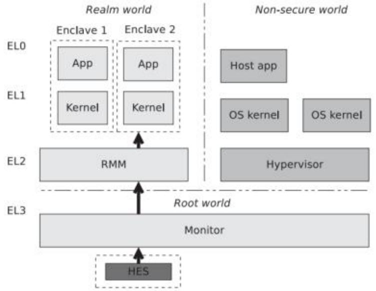
SGX: Enclaves locally attested by CPU + microcode. Intel-provided Quoting Enclave signs attestation evidence. (attestation as a business)



SEV: VM enclaves directly attested by discrete security processor (AMD-SP).



CCA: Hierarchically layered attestation. Three CPU-isolated worlds (realm, non-secure and root), each with multiple privilege levels. Discrete HW (HES) at root



Towards Interoperable Enclave Attestation (Fruct 22) https://ieeexplore.ieee.org/abstract/document/9770907
Platform Attestation in Consumer Devices (in submission)

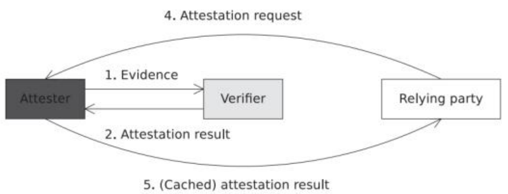
HUAWEI

Attestation and RoT Alternatives

- A Trusted Execution Environments (TEE). is commonly defined as an environment that provides a level of assurance of data integrity, data confidentiality, and code integrity. For this assurance to take place, we need to consider attestation as a service
- The Linux CC consortium (and others) work to harmonize and standardize all interfaces and functionalities for confidential computing
- The TEE environment will need a root-of-trust to host system keys and to provide attestation services. But not only that, there needs to be protocol support in the OS, and some service to verify the attestation claims

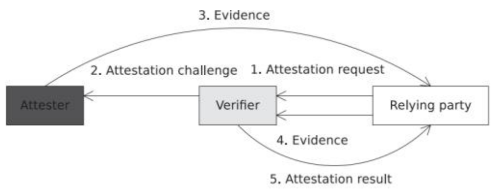
Some attestation models

Passport model



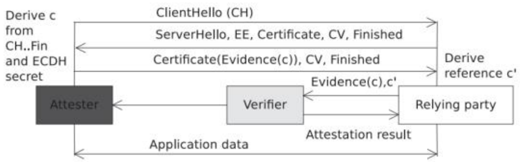
(Signed) verification result is presented by attester to relying party after the fact

Background check



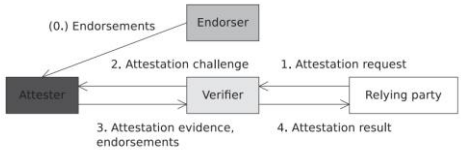
Relying party collects evidence checks with verifier later

Trusted channel w. attestation



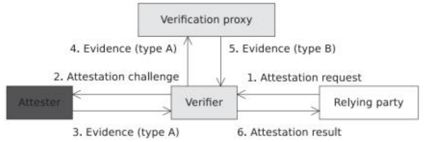
Protocol binding, resulting in shared secret AND attestation

Bundled model



(Signed) endorsements provided in-line with attestation (peer-peer ?)

Proxy model



Verification happens / is translated by background service

Attestation / RoT alternatives

Cloud services: CA | SafetyNet | Veraison | SecGuard | Privacy CA | ActiveDir (MS)

Cloud

- ✓ Run-time attestation
- ✓ Key attestation
- ✓ Certificate Enrolment
- ✓ OTA mgmt (enterprise)

Run-time attestation and system services: HUKS | TPM audit logs | SafetyNet (Android) | PSA (Arm) | SecGuard (HOS) | PKCS#11 / 15 | (E)IMA | MARS DICE

OS (Kernel)

- ✓ Static device attestation
- ✓ Secure Boot
- ✓ Key-store

TEE

RoT: Titan (Google) | SEP (Apple) | Caliptra | MARS | SGX | TPM | eSE | MSP (Huawei) | HES (Arm) | TZ / TZ-M | PSP (AMD)

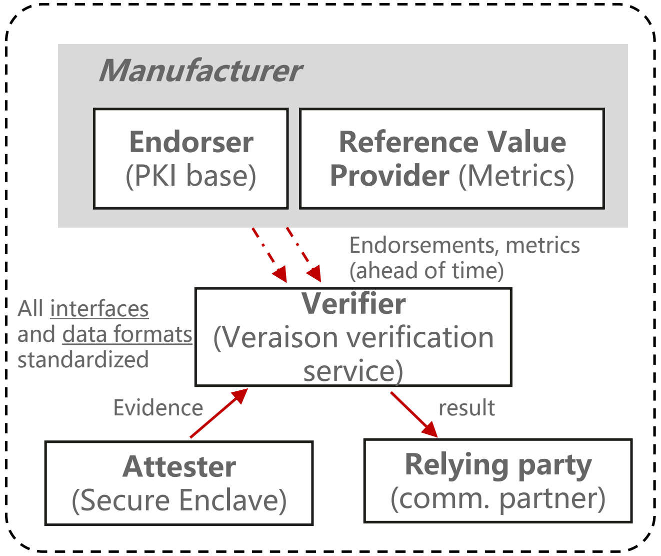
Veraison (ARM-CCA) Attestation

To move towards a common baseline for TEE attestation, our team in Huawei has worked in project Veraison, Linux CC, led by ARM Ltd.
" If each deployment needs a custom [attestation service], there is a significant software barrier and hence cost of entry to establishing a system that can be used in a secure manner. Veraison aims to provide consistency and convenience to solving this problem by building software components that can be used to build Attestation Verification Services. The components encompass a core structure of verification and provisioning pipelines" *)

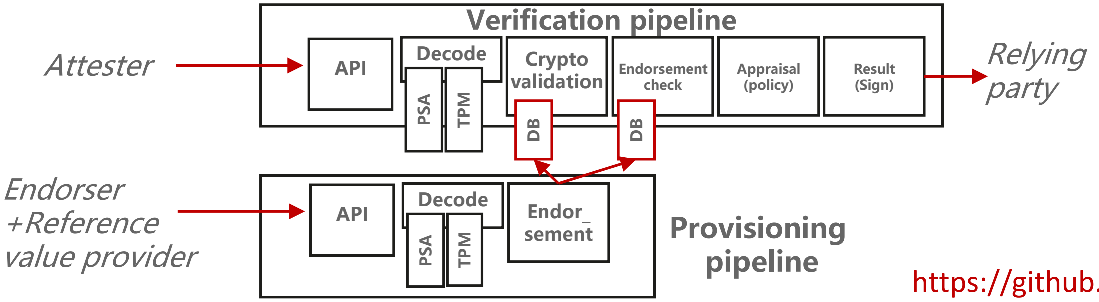
Veraison work builds on existing, standardized attestation formats:

1. CWT (IETF): **RFC8392** 2. JWT (IETF): **RFC7519**	Standard used for representing claims to be transferred between attester & verifier.
CDDL : Concise Data Definition Language : RFC8610	Language to define claims independent of encoding language.
CBOR Object Signing and Encryption RFC8152	Standard that defines how to encode signatures & keys in CBOR.
RATS-PSA : IETF-RATS-PSA-Attestation RATS-EAT: IETF-EAT	Describes the exact syntax and semantics of the attestation claims.
Attestation results for secure interactions : Attestation Appraisal	Standard on how verification result (appraisal) can be represented.
Concise representations of reference values for supply chain 1. CORIM : Concise Reference Integrity Manifest 2. COSWID: Concise Software Identification Tags 3. COMID: Concise Module Identifiers	**CORIM**: Trustworthy bundles of CoMID and CoSWID, signed by supply chain manufactures. **COMID:** Are the "hardware component" complement. **COSWID**: Structure to identify and describe individual software components, patches etc..

In IETF RATS terms, Veraison attempts to become the unified interface for endorsement collection and evidence validation



Veraison Software Architecture



https://github.com/veraison

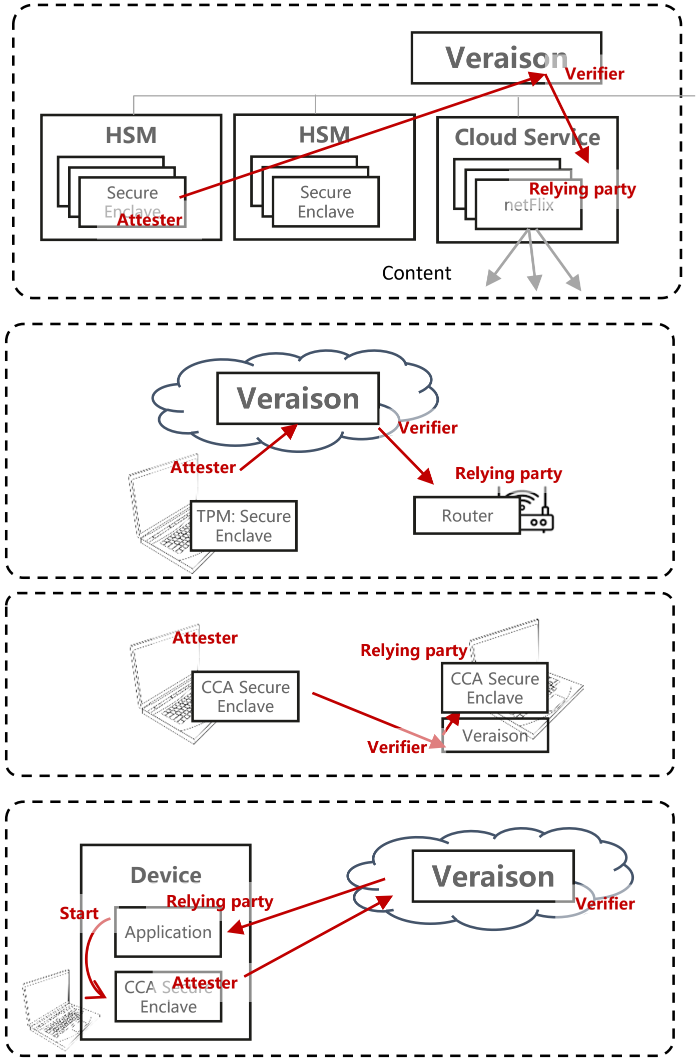
Veraison verification in 3+1 ways

1) Securing cloud workloads with enclaves

2) Securing terminal devices enclaves

3) Securing enclaves peer-peer (migration)

+1) Authenticated enclave execution



Runtimes and TEE Code Development

New Hardware-new Runtime ?

- In GlobalPlatform TEE, the singular TEE kernel is running at a higher privilege level, and just like a normal OS, it provides TA-TA and TA-kernel isolation using memory management, access control to OS resources like drivers, static and run-time attestation of TAs as well as TEE-specific services secure storage, key-store access. At a high TCB cost, with a single SDK.

- For VM-based enclaves, the setting is different. The hardware isolation is done at hypervisor-level, but we want minimize TCB and let hypervisor/microvisor only handle VM-VM isolation. By default this is suitable for full cloud VM Confidential Compute (say a Linux VM running in isolation), but to transition the consumer-device TA to this architecture we must re-architect:

  - Having one full OS kernel in each enclave is wasting memory, if individual TA workloads are small

  - The run-time must handle at least part of providing service and attestation / access control to drivers – therefore TA-runtime isolation is important

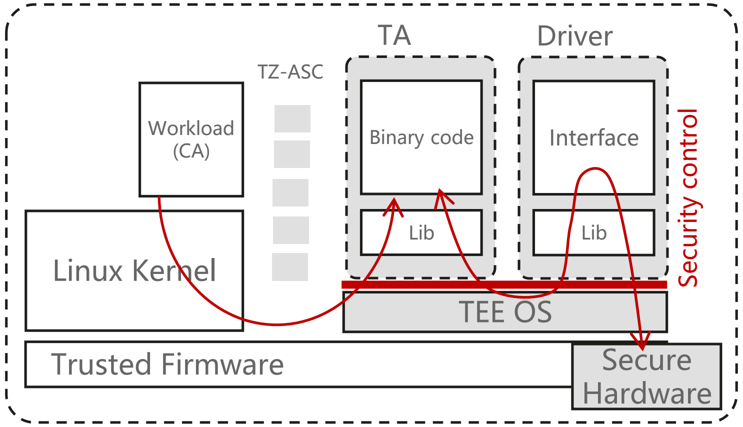
  - At the same time, allowing for TA workloads written in different languages and regimes (SDKs) might make sense, especially since strict memory usage limits can be lifted

  - Driver (secure hardware) support is a feature required in consumer devices - more than in CC

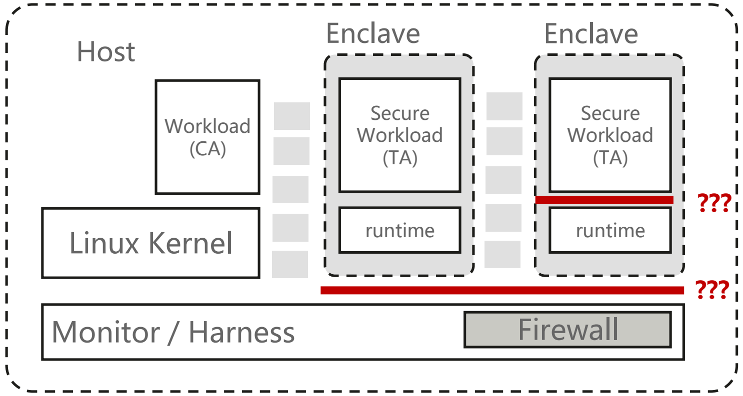
  - GP (5-7 Billion) device (and TA) legacy must be accounted for as part of the software stack. What is the migration path ?

  - Memory protection of run-time code - in software (Rust ?) or in hardware (ARM PA/MTE/BTI/..) may be a wise choice in a re-write

TrustZone

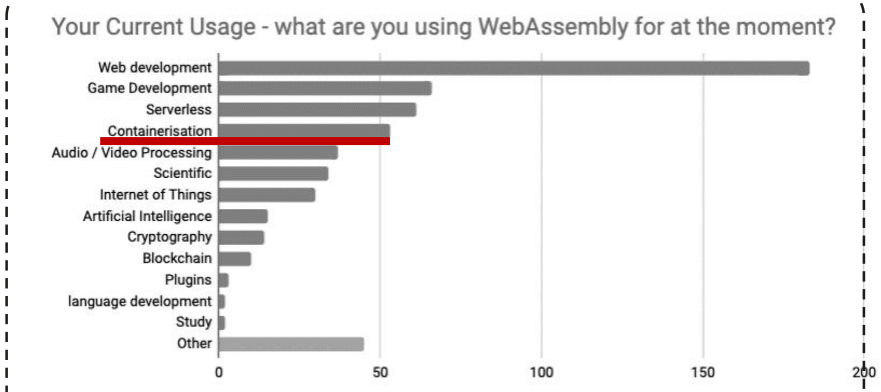


VM-based enclaves



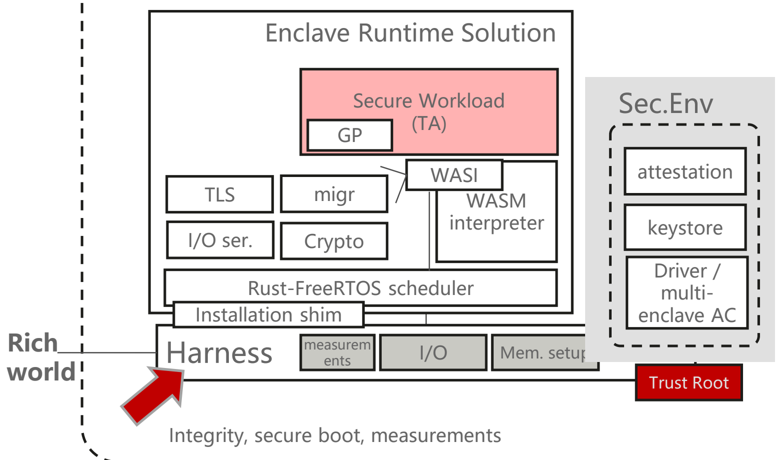
WebAssembly Interpreter as a TEE Runtime Component

- The WebAssembly (WASM) virtual hardware description, provides *) a memory-safe, sandboxed execution environment e.g. for browsers, but increasingly available also for embedded and stand-alone use.
- Using WASM in TEE has been proposed in recent years **) - ****) , primarily as a sandbox, to isolate the TEE workload from being a danger to its environment (the TEE). Additional advantages is a coherent interface specification (WASI) to the platform, which allows for easy integration of e.g. access control or service APIs.
- The isolation aspect of WASM is however overrated, i.e. there is little protection against in-workload memory safety, which badly reflects on security if JIT compilation is used. Further hardening of the WASM runtime is needed for TEE use.
- WASM LLVM backend immediately allows TA writing in C, Rust, Java, Pascal, … any language for which a front-end exists



https://blog.scottlogic.com/2021/06/21/state-of-wasm.html

HSSL implemented software architecture



M.Sc https://www.utupub.fi/bitstream/handle/10024/152762/Pop_VasileAdrianBogdan_Thesis.pdf?sequence=1

HSSL tests – language independence works

Java

Measurements (for SPECjvm crypto test in PoC)

Test	Speed	CPU load	Size
Reference (java jvm)	1.59s	324%	2.9MB (jar)
TZ w. WASM interpreter	1.41s	99% (single core)	1.2MB
TZ + native	0.17s	100% (single core)	2.2 MB

Micropython benchmarks

Test	Native	TA
Integer (add/mul/div)	0.04s 0.05s 0.05s	0.04s 0.05s 0.05s
PI to 1000 digits	0.05s	0.06s

With Pyodide and 200% overhead, python dependencies can be (live) attested and loaded from host (driver)

Huawei public competition
https://ww.chaspark.net/#/questions/790084197791088640?sub=790087690866180098

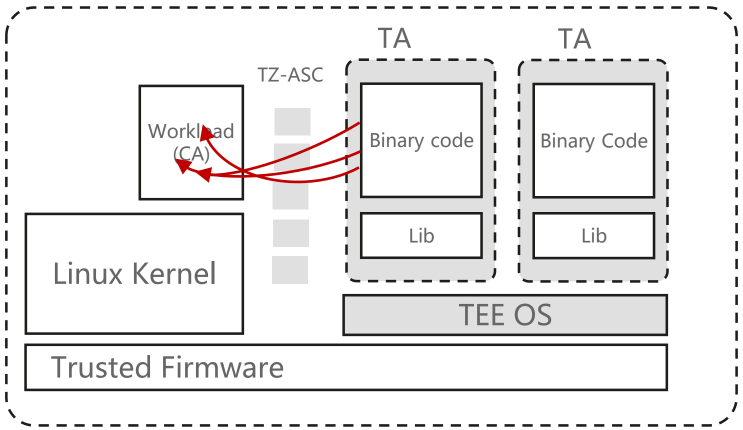
*) WASM: https://webassembly.org
**) WaTZ, Uni. Of Neuchatel (IEEE ICDCS 22): https://arxiv.org/pdf/2206.08722.pdf
***) TWINE, Uni. Of Neuchatel (ICDE 21): https://www.researchgate.net/profile/Valerio-Schiavoni/publication/350512786_Twine_An_Embedded_Trusted_Runtime_for_WebAssembly/links/60743fbaa6fdcc5f779d0c1a/Twine-An-Embedded-Trusted-Runtime-for-WebAssembly.pdf
****) AccTEE, TU Braunschweig (int. Middleware Conf. 20): https://dl.acm.org/doi/pdf/10.1145/3361525.3361541

HUAWEI

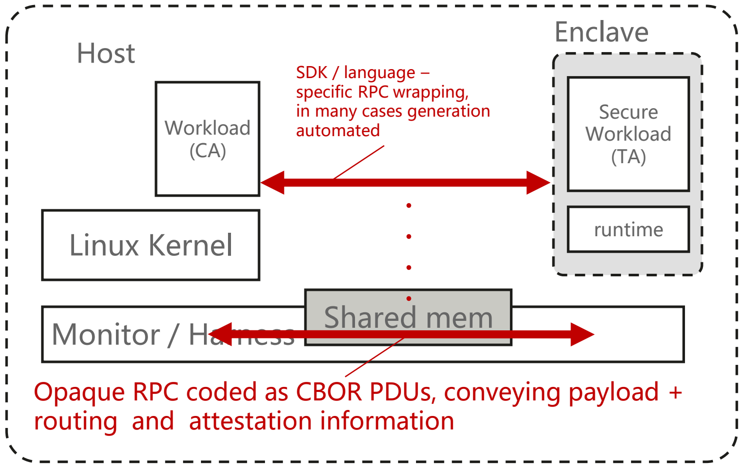
RPC and SDK for Enclaves

- In GP TEE, RPC is arranged via direct memory addressing from TA to CA, i.e. the APIs for communication between host and TA is arranged by conveying pointers. This is not only bad for CA security --- safe and unsafe memory references are in no way distinct in GP TA SDKs *). This is a big risk for developers.
- We propose PDU-based interaction as is used e.g. in Google Project Oak (Linux CC), where CA-TA interaction is done via messaging. Different to Oak, we introduce a language-agnostic, but routing aware lower RPC, complemented by a language-specific but varying payload format.

TrustZone

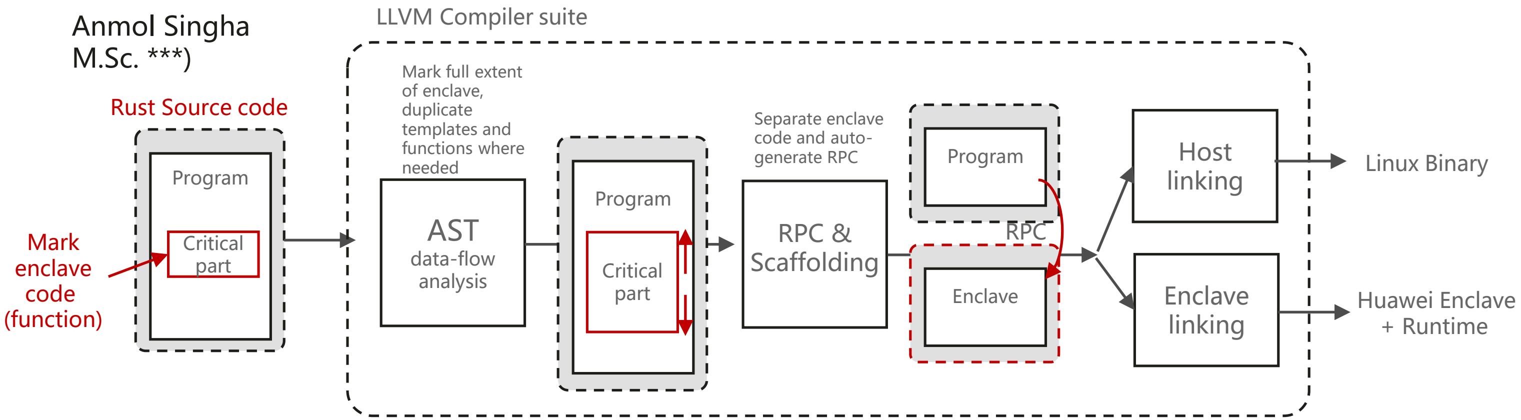


VM-based enclaves



Code Development

- With few exceptions, Host and TA applications come in pairs: Banking, eID, Video player/DRM. Therefore, it is not conducive to program the pair in separation:



*) Nokia, (Trust 2012): https://link.springer.com/chapter/10.1007/978-3-642-30921-2_1
**) Google Project Oak: https://github.com/project-oak/oak-enclave
***) Enclave Host Interface (Aalto M.Sc 22): https://aaltodoc.aalto.fi/bitstream/handle/123456789/116438/master_Sinha_Anmol_2022.pdf?sequence=1

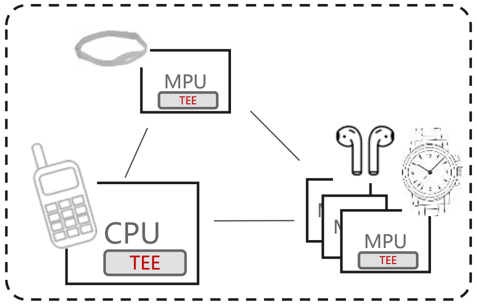
Where to Next?
– if trusted execution with enclaves materializes

Remaining Research Challenges

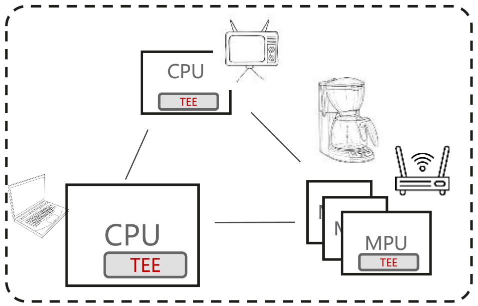
Computing going Heterogeneous (CPU→ Set of CPUs in a connected network)

Moore´s law allowed industry for 40 years to make ever faster, general-purpose CPUs. Now this is not possible any more, so we are faced with architectures where many, special-purpose computers work together with the same memory (server), in the same machine (car) or in the same context (home automation, personal-area network)

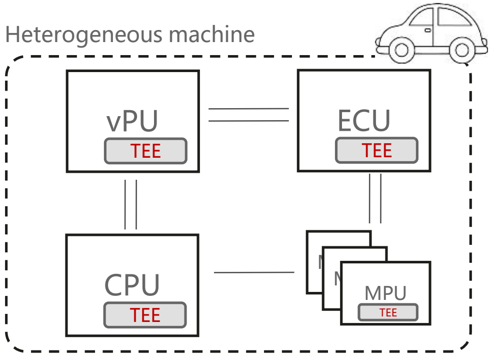
Heterogeneous use case



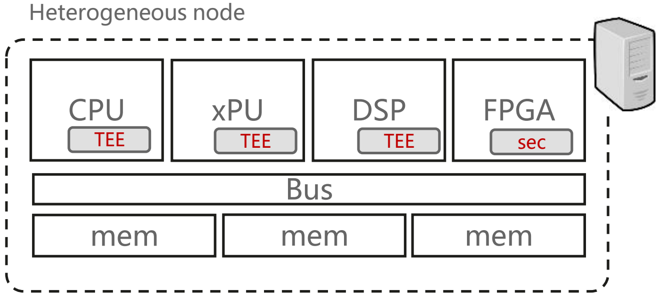
Heterogeneous network



Heterogeneous machine



Heterogeneous node



Impact of Heterogeneity to TEE use-cases:

1) Provisioning, key injection, code installation isolated execution, TEE security level assessment (attestation)

2) Multi-TEE: Many TEE working together to fulfill one use-case (e.g. CPU + NPU → secure AI models)

3) Mobile Code: TEE code (service) migrates between nodes for performance, backup, OR as a part of expected behavior (different devices have different local state or accessible features)

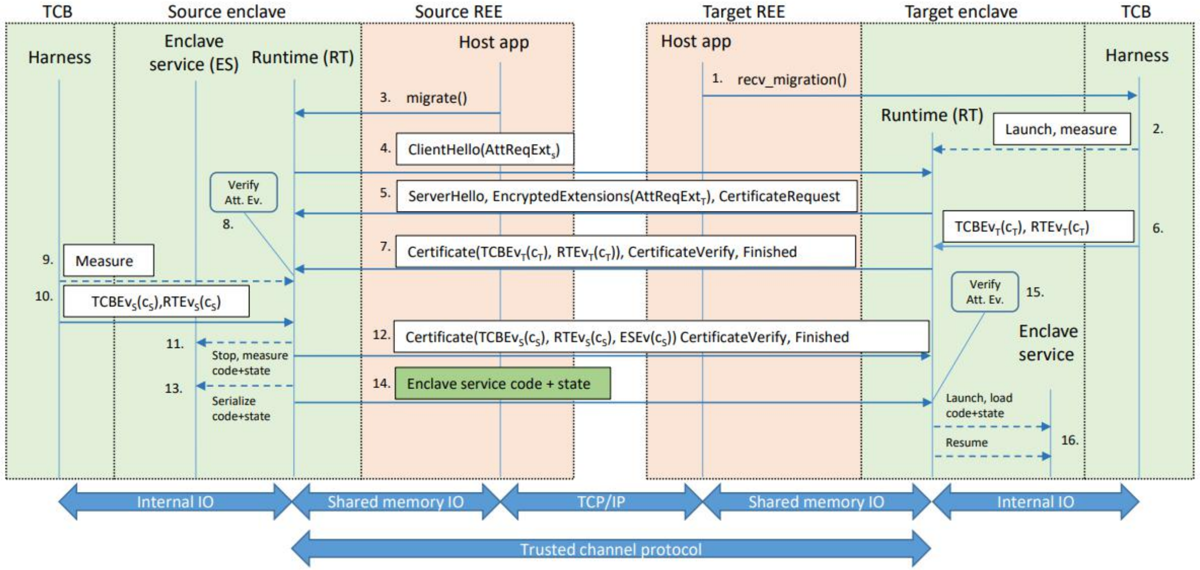
In both consumer and cloud devices, these research directions represent a combination of platform isolation, attestation and secure state-fulness with secure protocols, key sharing, or features like oblivious storage, multi-party computation and the like

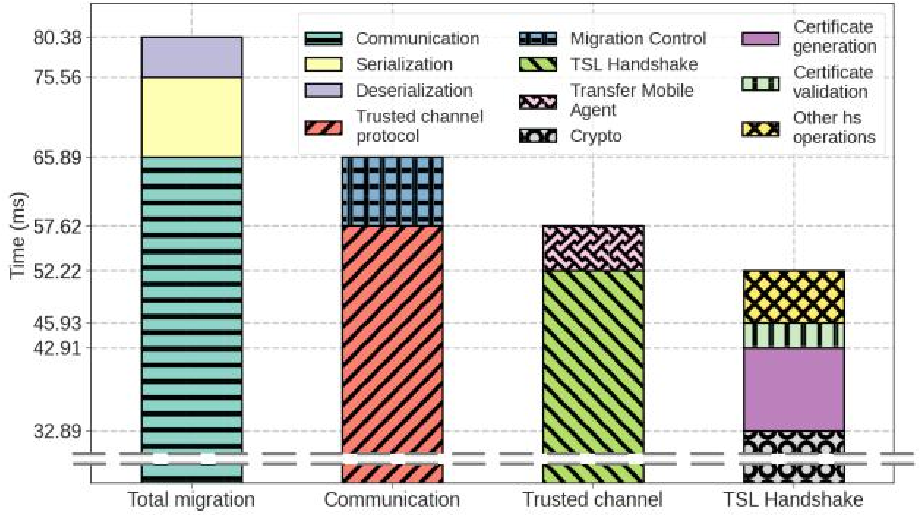
Migration

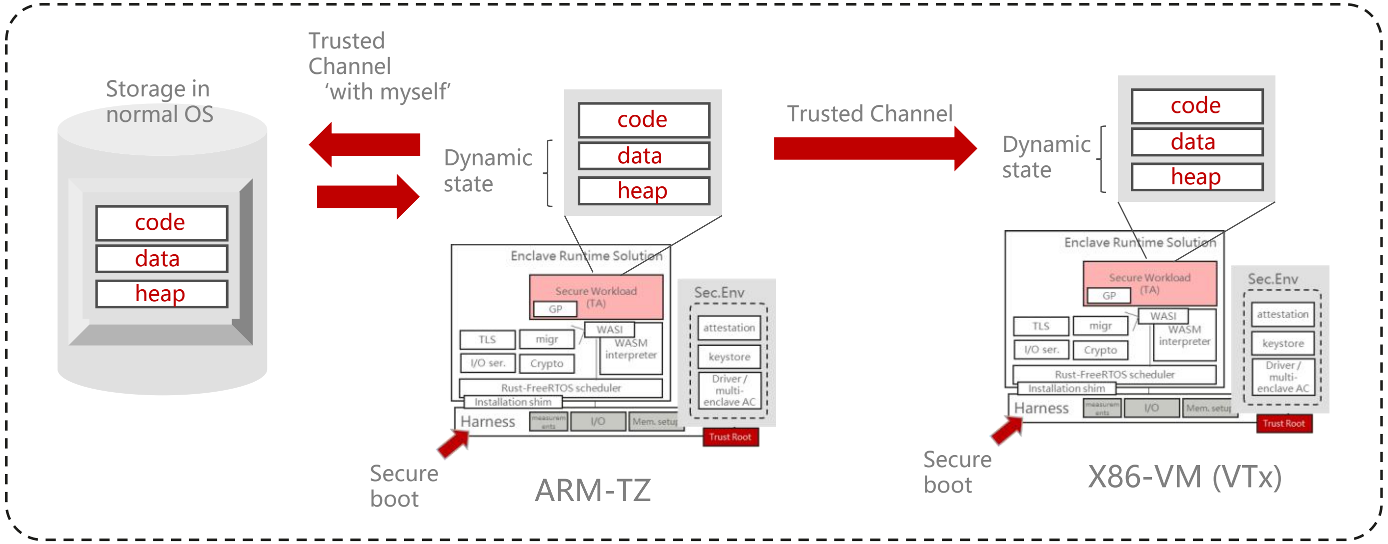
Having a virtual ISA allows live migration of enclaves, and live migration can also be used for secure local storage

A complete handover between two devices with different hardware and enclave setups, including attesation (trust-root excluded) could be done in <100 ms. For enclaves with 10000s bytes of state and code, we were talking seconds.

This is a given use-case in cloud. In consumer, maybe less business need for the moment.







15th European Workshop on Systems Security (22): https://dl.acm.org/doi/pdf/10.1145/3517208.3523755
Multi-TEE for eID – soon in submission

HUAWEI

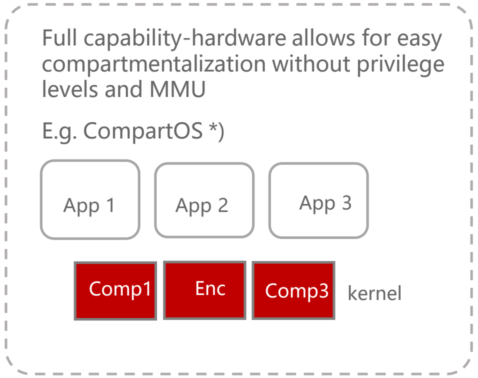
And hardware may still surprise us – In more than one way

Although we can be quite confident that VM-based enclave technologies will allow consumer device architecture to make a leap in how trusted execution is arranged, there might be more competent (and faster) hardware out there to run enclaves in isolation.

Also, these new chip isolation (and trust root) designs are still young, and will have unforeseen shortcomings w.r.t. to security such as malfunction, side-channels etc. Can these be mitigated in the field, e.g. by hardware reconfiguration? And if so, should not platform attestation include self-healing / reconfigurable hardware state?

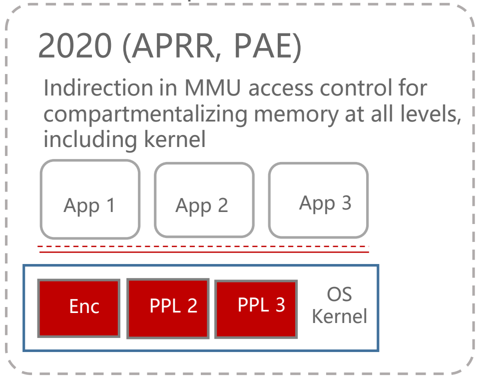
And will computing architecture overall change in consumer devices? What about in-memory computing? Non-volatile system memory? Most enclave (isolation) solution rely on a dominant CPU that manages all system state – is that even a realistic assumption?

Cambridge CHERI / Morello



*) https://arxiv.org/pdf/2206.02852.pdf

In-kernel compartments



Thank You!