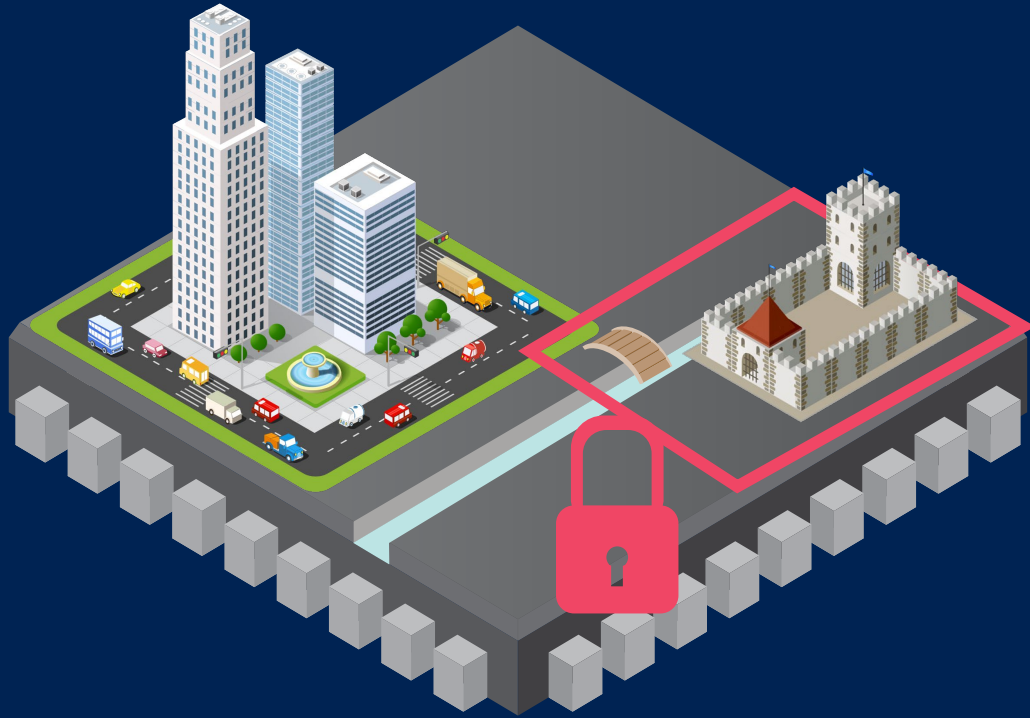




Standardizing Security Services

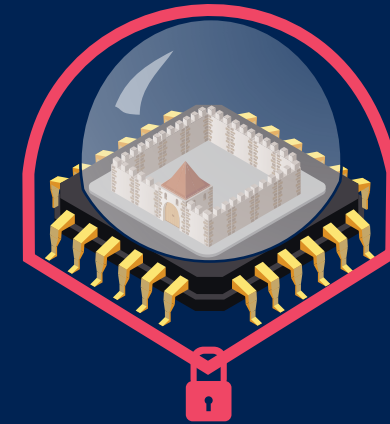
Richard Hayton

Chair TES Committee
Co-Chair Automotive Task Force



TEE (Trusted Execution Environment)

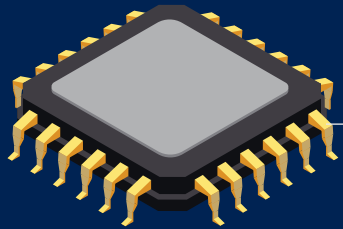
Secure OS on a large chipset



SE (Secure Element)

Javacard Secure OS on a hardened microcontroller

Automotive History



Apps run on
MicroController
(often AutoSAR)



Hardware Element
Provides Crypto
(SHE, SHE+, EVITA...)

Safety is the priority. Security comes second.

SHE provides a standard, but it is very low level

- Focus is on 'key slots' which can hold different types of keys / key material

Lots of variation on how it is used (per project)

- CAN limitations lead to a lot of variation
- Different approaches for establishing secure boot / trusted ID
 - SHE stores expected MAC for boot image + secret material to generate it
 - Provisioning MAC/Secret is "hard"
- SHE does not protect the broader application
- Risk of confused deputy attacks if Microcontroller is compromised.

Automotive Trajectory

Desire to centralize more functions (SDVs)

- Traditional TEEs

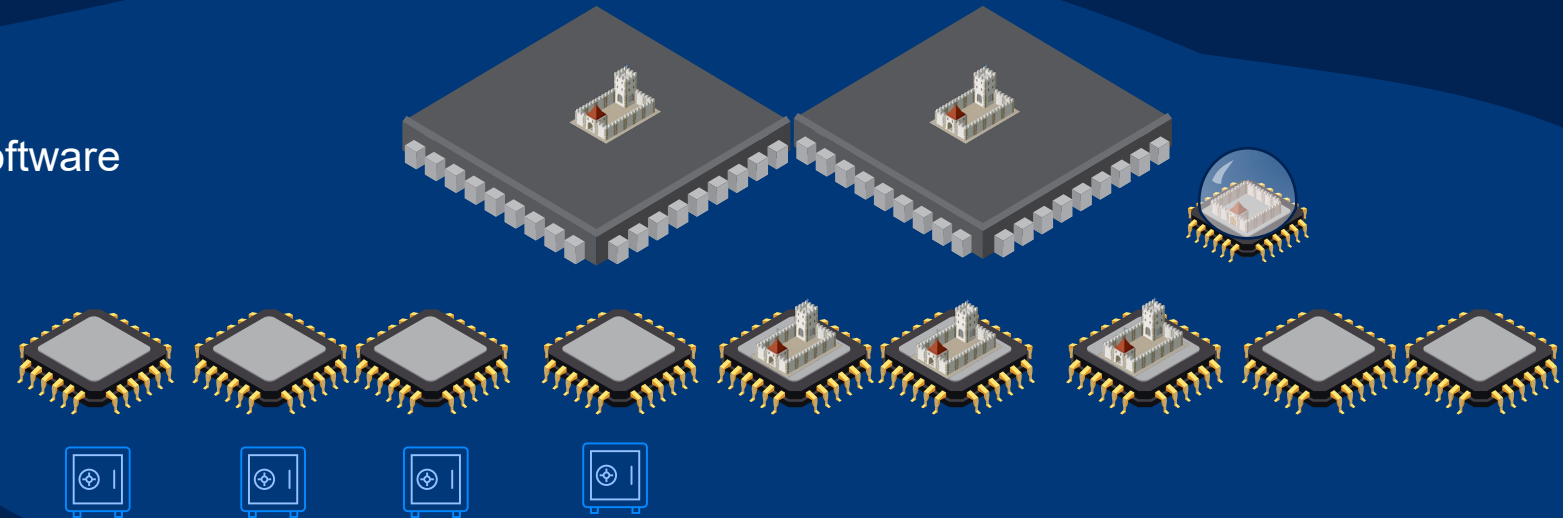
New small ECUs

- Opportunities for “Micro TEE”
- Security services within MCU as software

Legacy ECUs

- Traditional SHE...
- ECUs with no Security

Opportunities to leverage existing SEs and/or add new ones



Challenge #1

We need to consider safety (new for GP)

- Many functions are hard to move from discrete MPUs to CPU due to safety concerns
 - Timing guarantees are not part of GP specs or common implementations
 - Boot times on larger processors can be long(!)
 - [Freedom from] Interference is a relatively new topic for A-Class processors
- TEEs are primarily used for non-safety critical features today
 - There is a demand for more guidance / standardization on using TEEs in this context

Opportunity

The TEE Architecture supports great flexibility in system design.

Should we be more prescriptive?

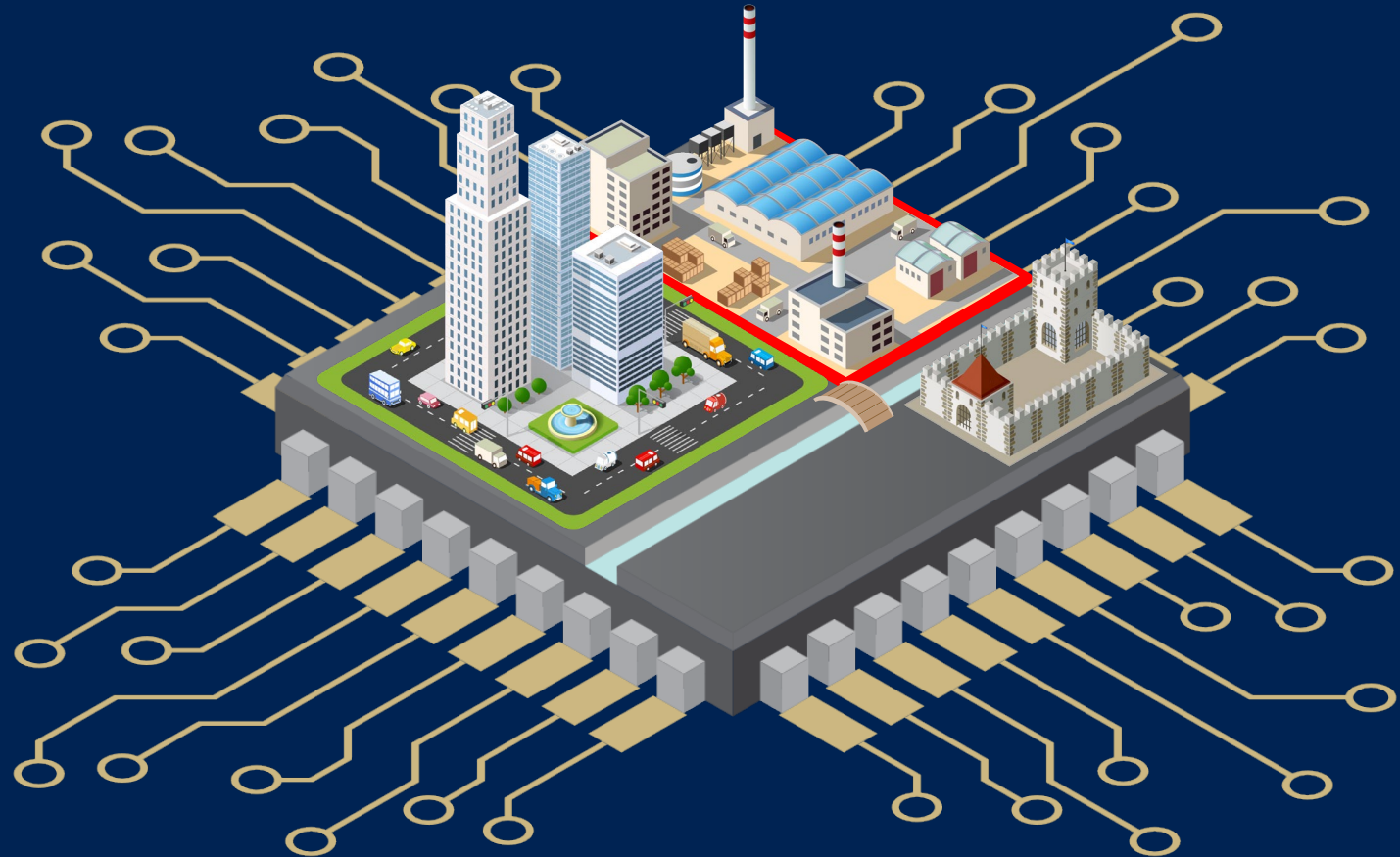
Supporting Safety Critical Applications

A modern “City-like” operating system enables lots of different applications to run at once, but is generally not suitable for safety critical or real time functions

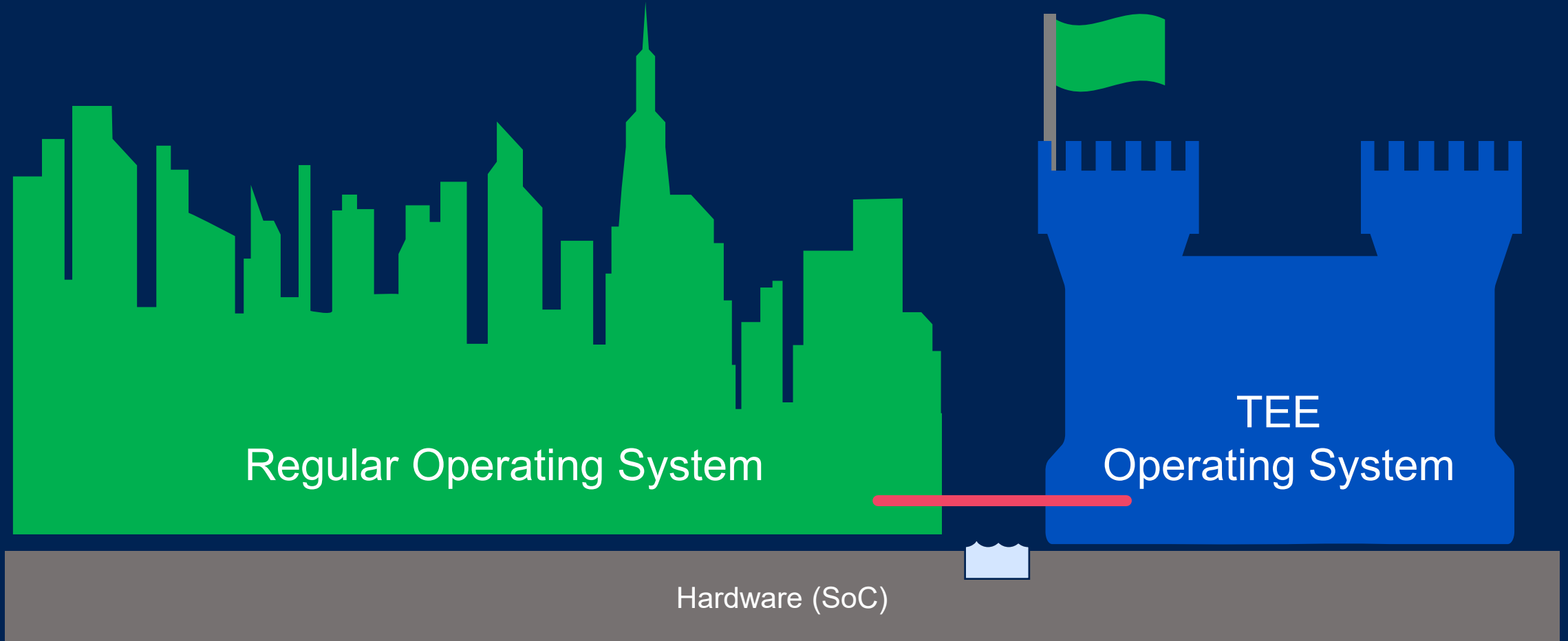
Supporting safety critical functionality in Software Defined Vehicles remains an active area of discussion

One common approach in SDVs is to add a second or third operating system which are more focused on safety critical applications

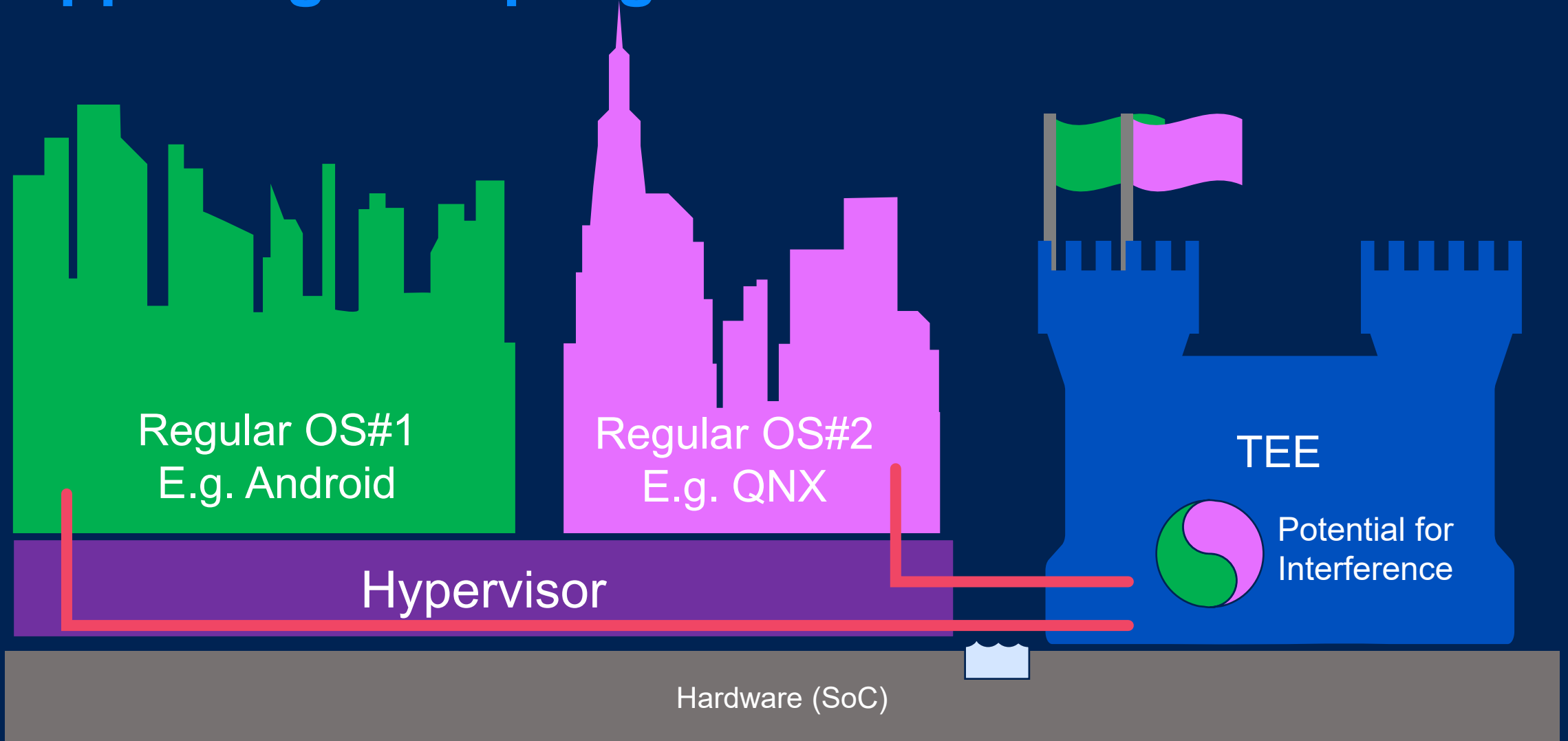
An open discussion is how (or if) the TEE should support mixed-criticality clients



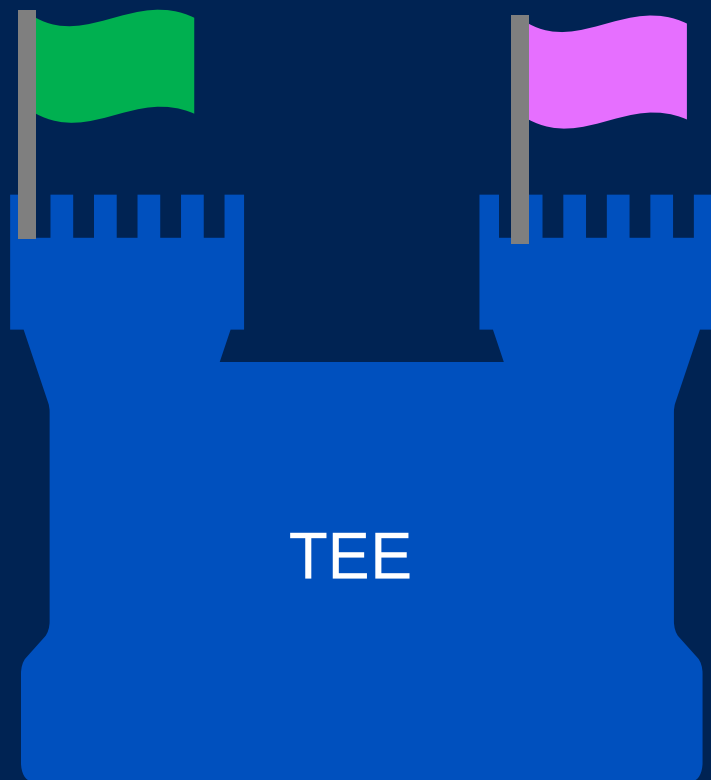
Simple NWD-SWD Architecture



Supporting multiple guests



TEE Guarantees



Trusted apps are strongly isolated from a security perspective (GlobalPlatform Protection Profile)

- Pink cannot see Green's data
- Pink cannot see if Green is running



The TEE may rely on an external system for scheduling

- If Pink's client OS dies it may prevent green from running



The TEE OS may rely on an external system for peripherals (e.g. Flash Storage)

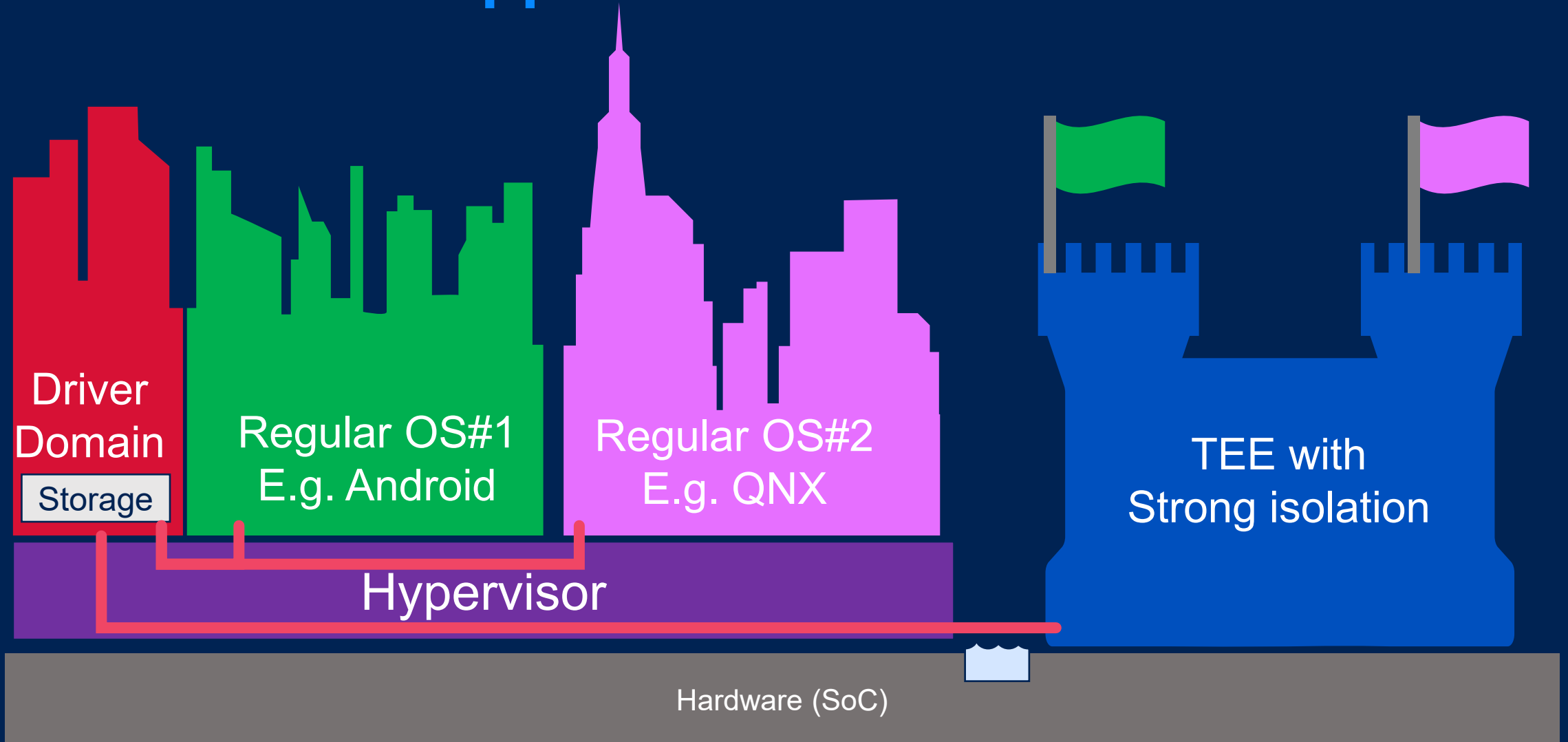
- If Pink's client OS dies it may prevent green from running



The TEE may have shared resources

- Shared heap (Pink may consume Green's resources)
- Need for 'background' threads/activity

Driver Domain approach



Driver Domain approach



Trusted apps are strongly isolated from a security perspective

- Pink cannot see Green's data
- Pink cannot see if Green is running



The TEE may rely on an external system for scheduling

- Driver Domain handles scheduling



The TEE OS may rely on an external system for peripherals (e.g. Flash Storage)

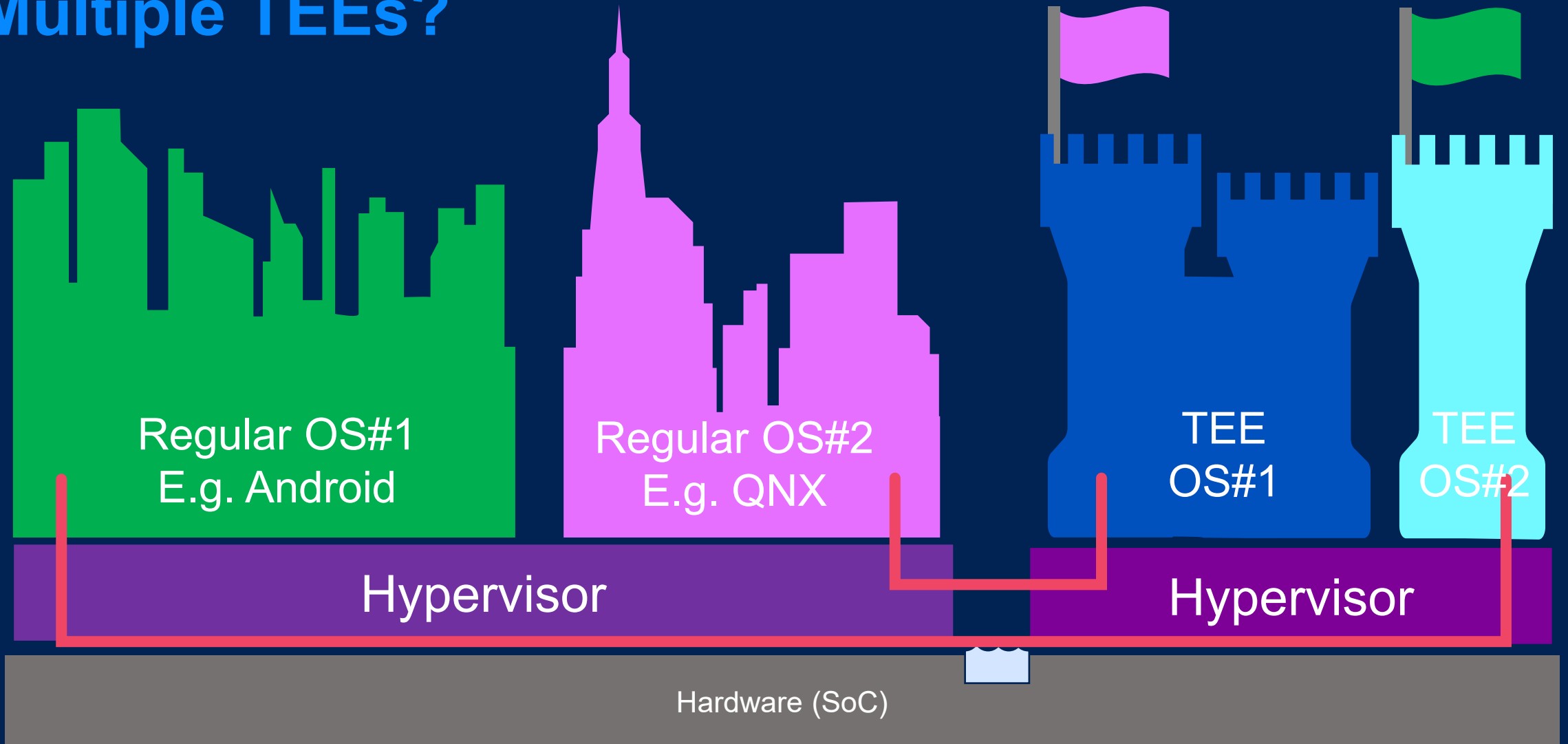
- Driver domain handles storage



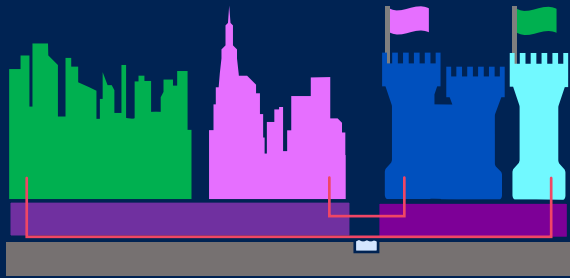
The TEE may have shared resources

- Shared heap (Pink may consume Green's resources)
- Need for 'background' threads/activity

Multiple TEEs?



Multi-TEE approach



Trusted apps are strongly isolated from a security perspective

- Pink cannot see Green's data
- Pink cannot see if Green is running



The TEE may rely on an external system for scheduling

- Each TEE relies on its client OS

The TEE OS may rely on an external system for peripherals (e.g. Flash Storage)



- Each TEE can rely on its client OS
- But NWD hypervisor must still solve flash sharing



The TEE may have shared resources

- Each TEE is independent
- (but overall uses more resources)

Current thinking... [too soon for conclusions]

SWD hypervisors are a useful tool in achieving freedom from interference

But they rely on hypervisor features that are not universal (or standard)

GlobalPlatform needs to look at standardizing (or providing guidance) on features it has traditionally avoided

- Performance (e.g. boot time for TEE and broader system)
- Denial of Service
 - TA-TA resource consumption (processor, heap, storage)
 - NWD-TEE failure modes if >1 guest
 - Share use of cores between guests

Other isolation technologies....

Challenge #2

We want solutions that will scale across vehicle to allow functions to move from MCU to CPU over solution lifetime

- Low end vehicles in a model range may make different choices to high end vehicles
- Decisions may change over time

Opportunity

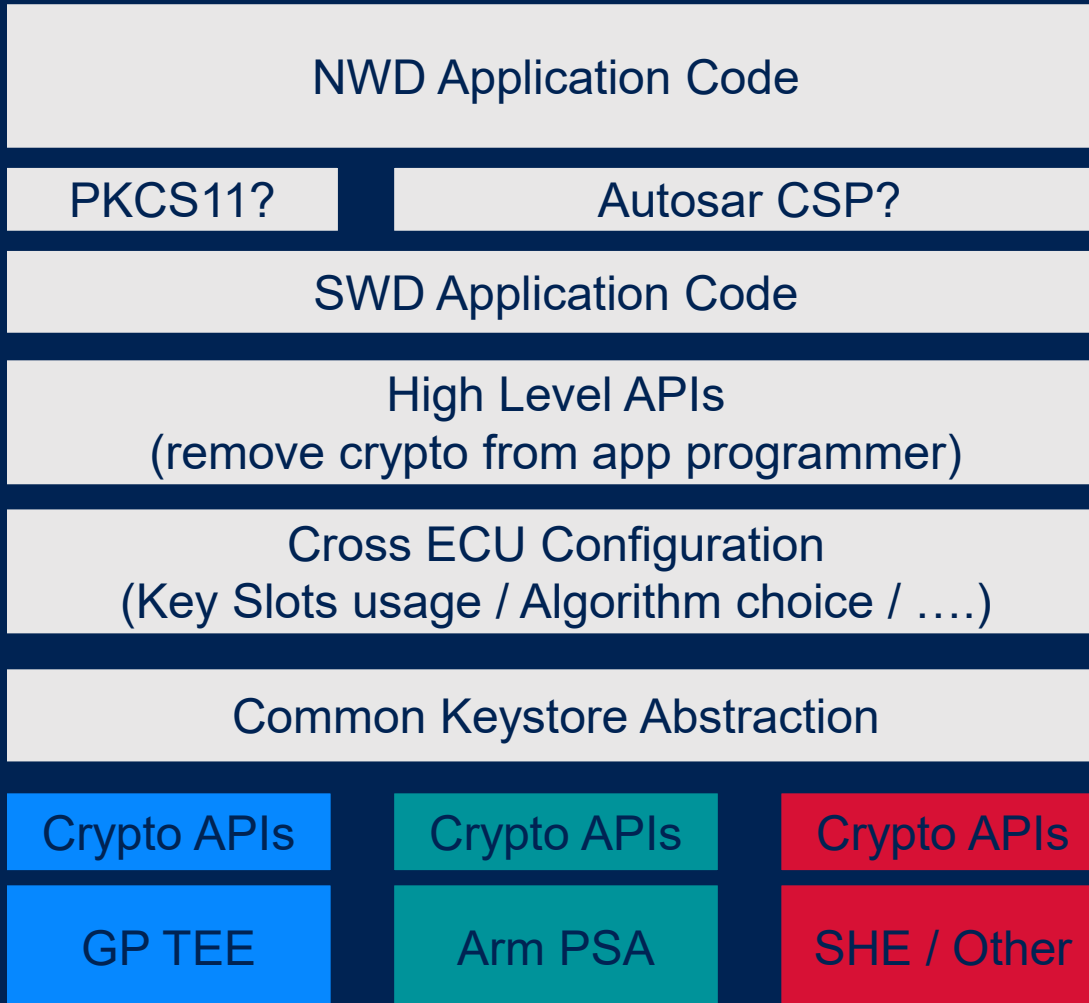
Software / APIs offer an opportunity to provide “useful abstractions” that can be implemented in different ways in different ECUs

GlobalPlatform TEE APIs have proved very stable over the long term

ARM PSA APIs are more common in Micro-Controllers and are another good starting point.

We can imagine a new “key store ++” set of APIs that may run over either of these platforms.

New “Malaga” Stack – maybe????



← Would help adoption [but a lot of work]

← Under debate if this is required/optional/unnecessary

← C.f. SSL (hide algorithm choice)

← New “interesting problem” solved by all OEMs/projects today

← Existing TPS KeyStore APIs ??

← Must support legacy systems if we hope for adoption



**Global
Platform[®]**

Securing the digital future

→ globalplatform.org