woven by TOYOTA

Global Platform Use Cases

October 24th, 2024

Vincent Mailhol

Senior Product Security Engineer
vincent.mailhol@woven.toyota

Meeting Agenda

About me

- Joined Woven by Toyota in October 2020
- [Maintainer of the CAN subsystem of the Linux kernel (a.k.a Socket CAN)]

01

Software define vehicle

A story of reusability

Reusable Platform	**TNGA: Toyota New Global Architecture**
History	Physical platform that is used to build Toyota vehicles ● Accounts for 80%+ of all vehicles ● Defined variants ● Scales and is reusable

Reusable Platform	**ePF: Toyota Electronic Platform**
Software	Software platform that is used to build Toyota vehicles ● Defined variants ● Scales and is reusable ● Is certified; no bespoke software

Reusable Platform

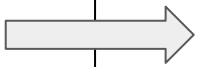
Common hardware components

ARM based chipset

Ideally Cortex-M or Cortex-A

Standardized APIs

Standardized security controls



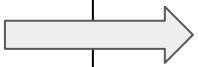
Supplier agnostic builds

Known technology

Known supported features

Reusable software

Testable functionality and features



Provide reusable components for engineers

Provide capability for platform to scale and be independent (loosely coupled) with the hardware

Provide a known secure and safe foundation for developing functionality

Capability to separate out the configuration of the software from the operation of said software

Automotive Specific Items

01
Functional Safety

Our software **must not** have any failure that impacts the safety of the road user, or any person that could be impacted by the road user.

02
Long Lifespan and Quality

It is possible to fix an issue via OTA in modern automobiles, but the cost is high and some items require a service visit. Toyota aims to support its vehicles in the field for **15-20** years.

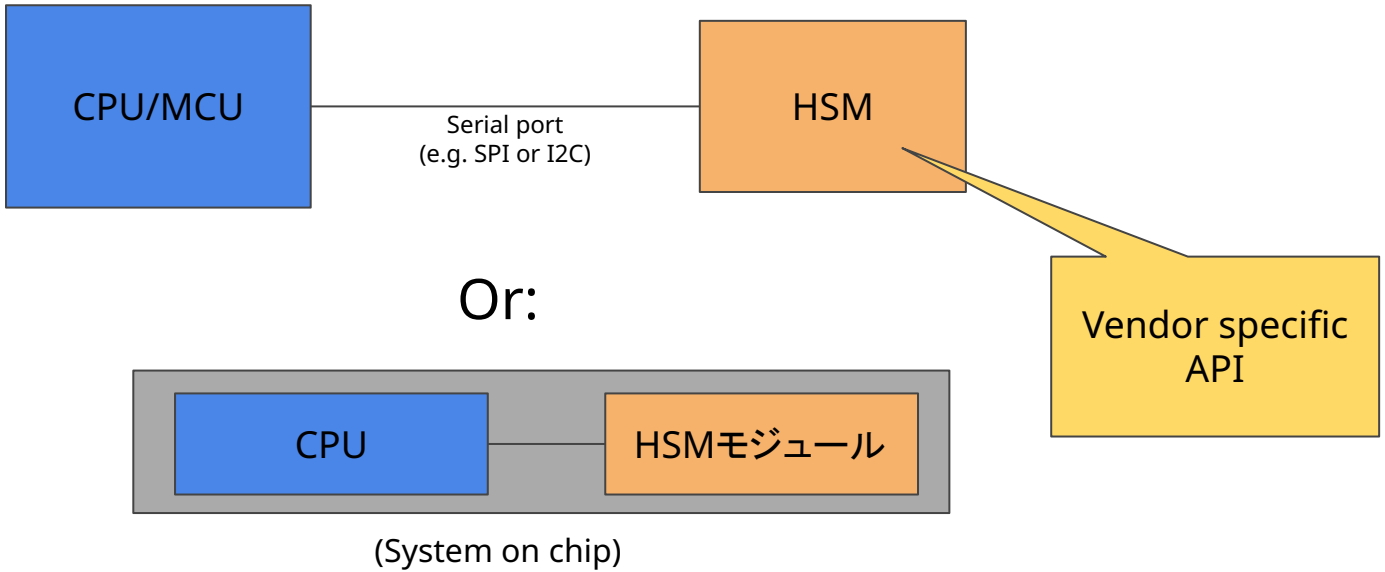
03
Performance

There are some scenarios, required for safety, security, or legislation that require specific actions to happen within a **defined amount** of time.

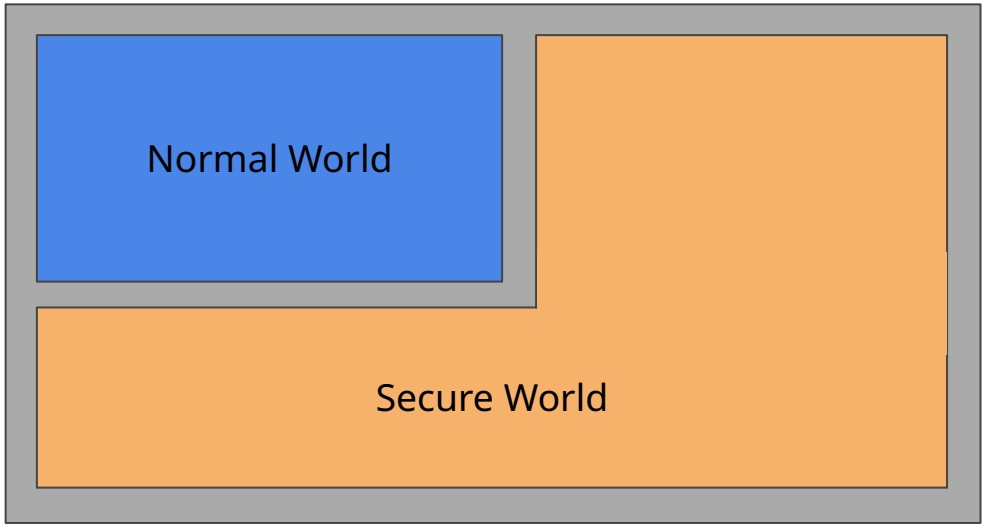
02

Global Platform Standard API

Classic automotive hardware security

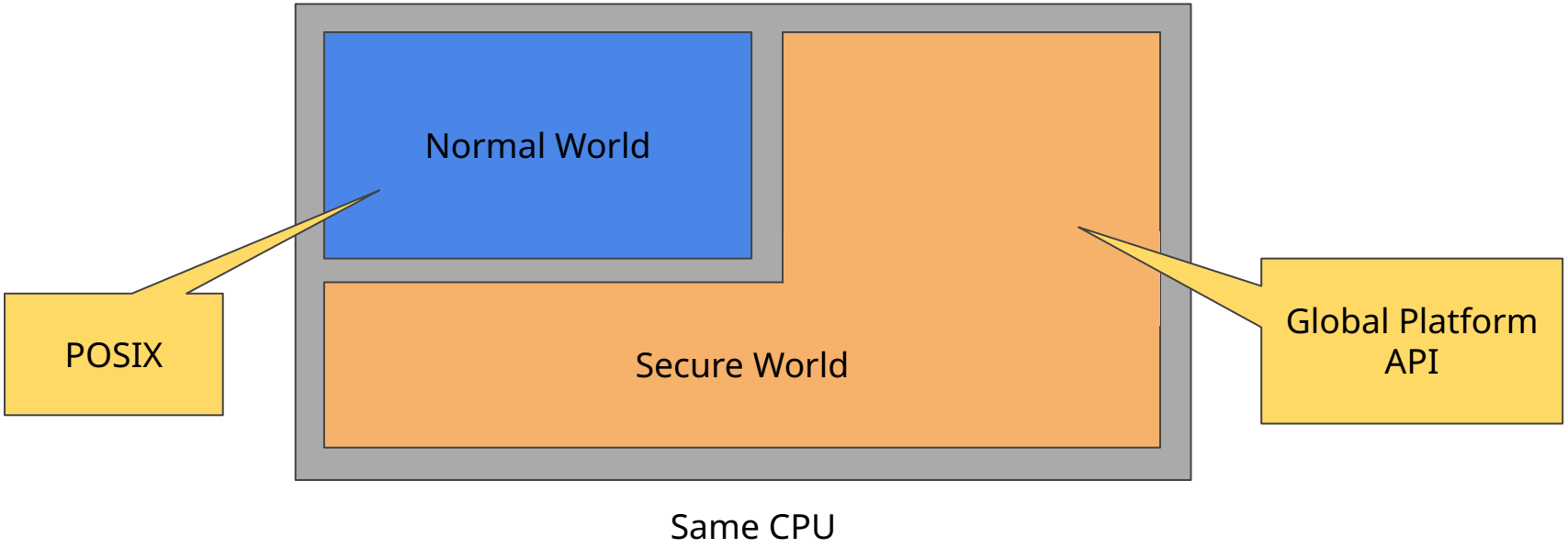


Trusted Execution Environment



Same CPU

Trusted Execution Environment + Use of standard API



Benefits of TEE with GP API

01
Cost

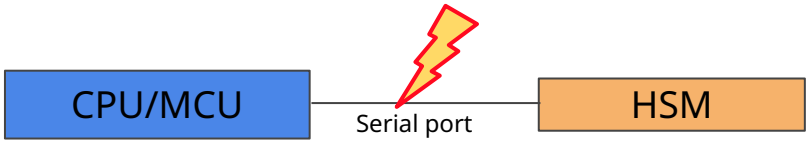
- Available by default on Armv8-A architectures.
- No additional module are needed.
- Code reusable

02
Speed

- Secure and non secure operation runs on the same CPU: less overhead communication cost.
- CPU is usually faster than HSM.

03
Security

- No serial port: more robust against hardware attacks.



03

How could reusability go wrong?

Study case on Ariane 5

Failure in the Inertial Reference System (SRI)

Overflow on 16 bit integer

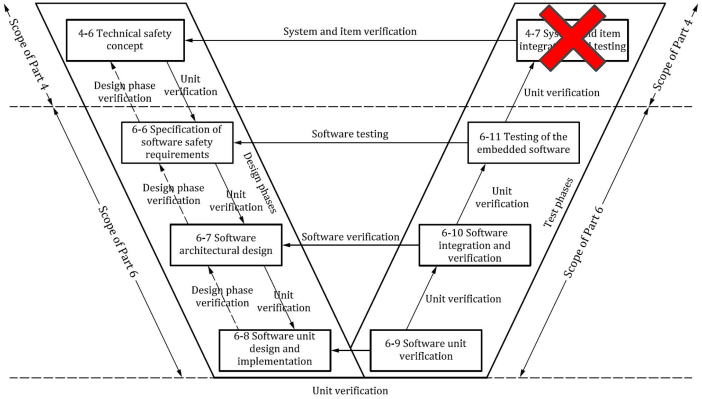
Consequences: $370M loss

Ariane 5 launch (June 1996)

SRI developed for Ariane 4

Ariane 4

No integration tests



SRI reused in Ariane 5

Ariane 5

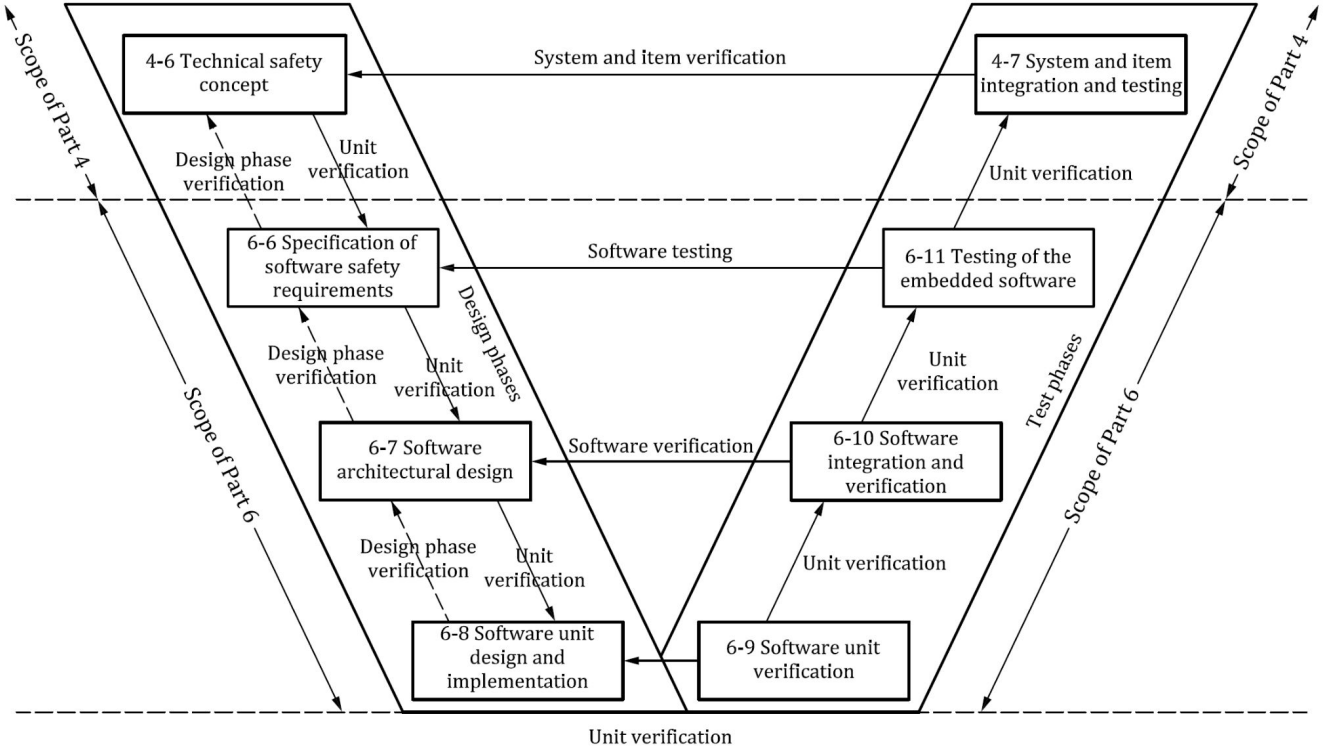
04

How to prevent failure

Processes and testing

Processes

- ISO 26262
- ISO 21434
- MISRA
- …

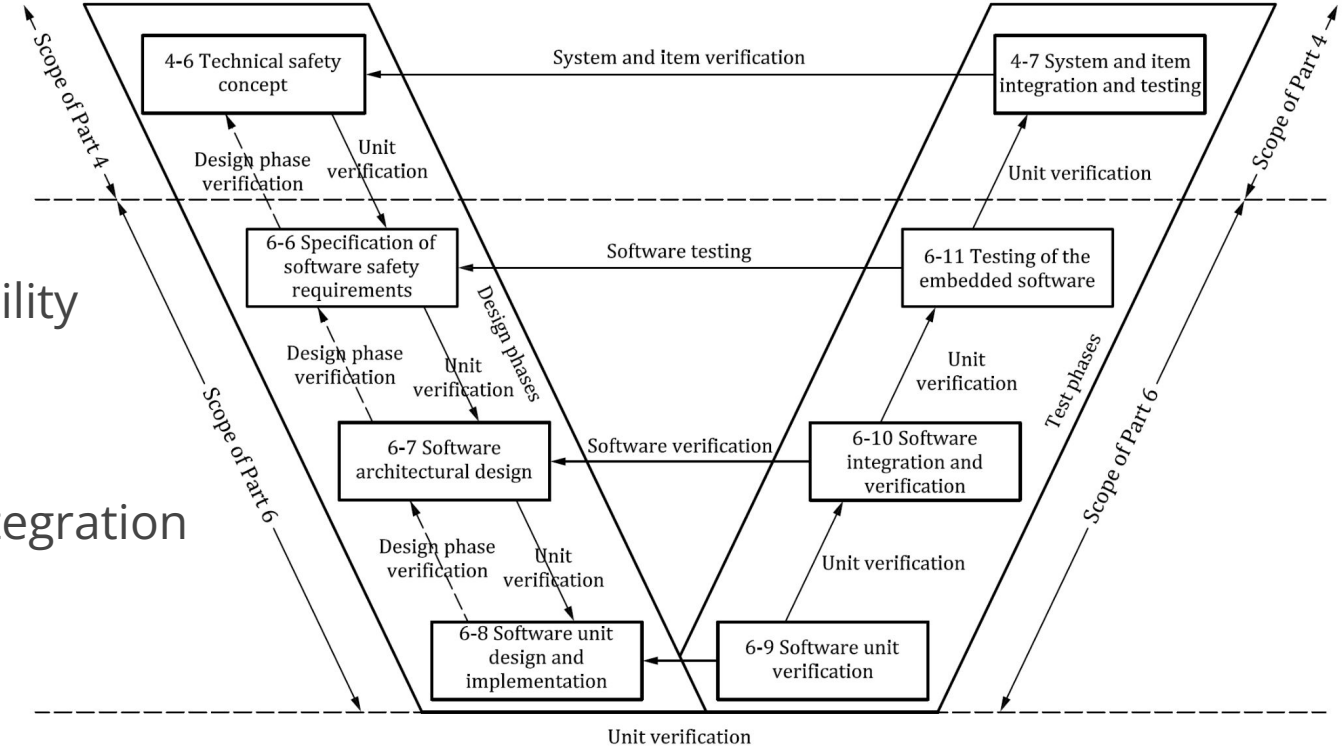


Automation:

- Reduce cost
- Increase reliability

Example:

- Continuous integration
- SIL
- HIL



05-1

Global Platform Properties

Global platform allow to query security properties

Example with time:

Table 7-1: Values of the `gpd.tee.systemTime.protectionLevel` Property

Value	Meaning
100	System time based on REE-controlled timers. Can be tampered by the REE. The implementation SHALL still guarantee that the system time is monotonic, i.e. successive calls to `TEE_GetSystemTime` SHALL return increasing values of the system time.
1000	System time based on a TEE-controlled secure timer. The REE cannot interfere with the system time. It may still interfere with the scheduling of TEE tasks, but is not able to hide delays from a TA calling `TEE_GetSystemTime`.

Global platform allow to query security properties

Code:

```
uint32_t system_time_procection_level = 0;

TEE_GetPropertyAsU32(TEE_PROSPSET_TEE_IMPLEMENTATION,
                     "gpd.tee.systemTime.protectionLevel",
                     &system_time_procection_level);
switch (system_time_procection_level) {
case 100:
        ERROR("Warning: REE-controlled timer");
        break;
case 1000:
        /* TEE-Controller timer: OK */
        break;
default:
        ERROR("Unknown system time protection level?!");
        break;
}
```

Global platform allow to query security properties

Other properties:

- `gpd.tee.cryptography.*`: check which cryptography algorithms are supported. Allow for crypto agility
- `gpd.tee.trustedStorage.*`: check the protection level of the secure storage

Global platform allow to query security properties

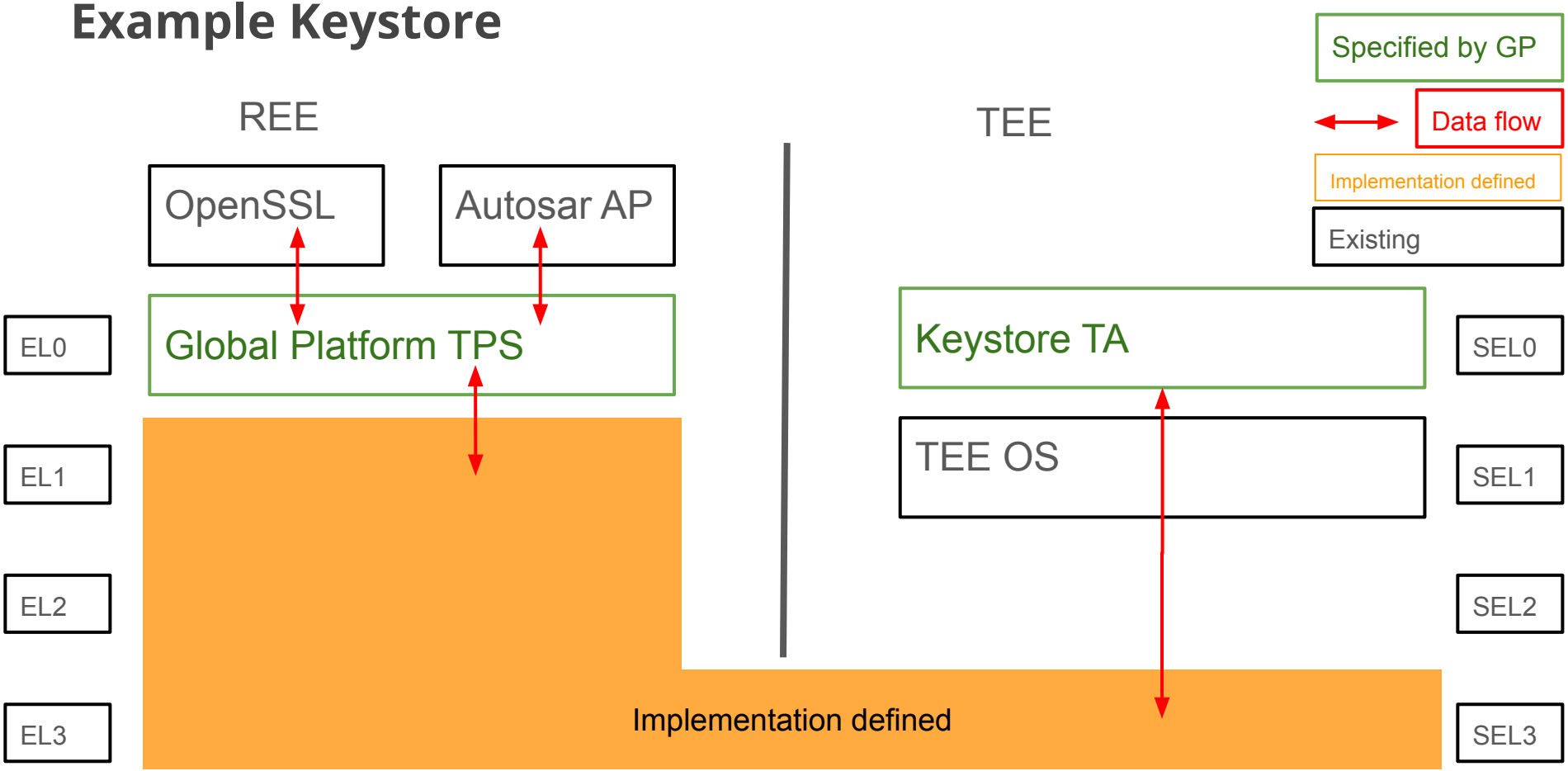
Idea: introduce new properties for the random generator:

- `gdp.tee.rng.prng`: pseudo random generator
- `gdp.tee.rng.trng`: true random generator (unspecified)
- `gpd.tee.rng.nist`: compliance to NIST SP 800-90*
- `gpd.tee.rng.bsi`: compliance to AIS 20 and AIS 31
- …

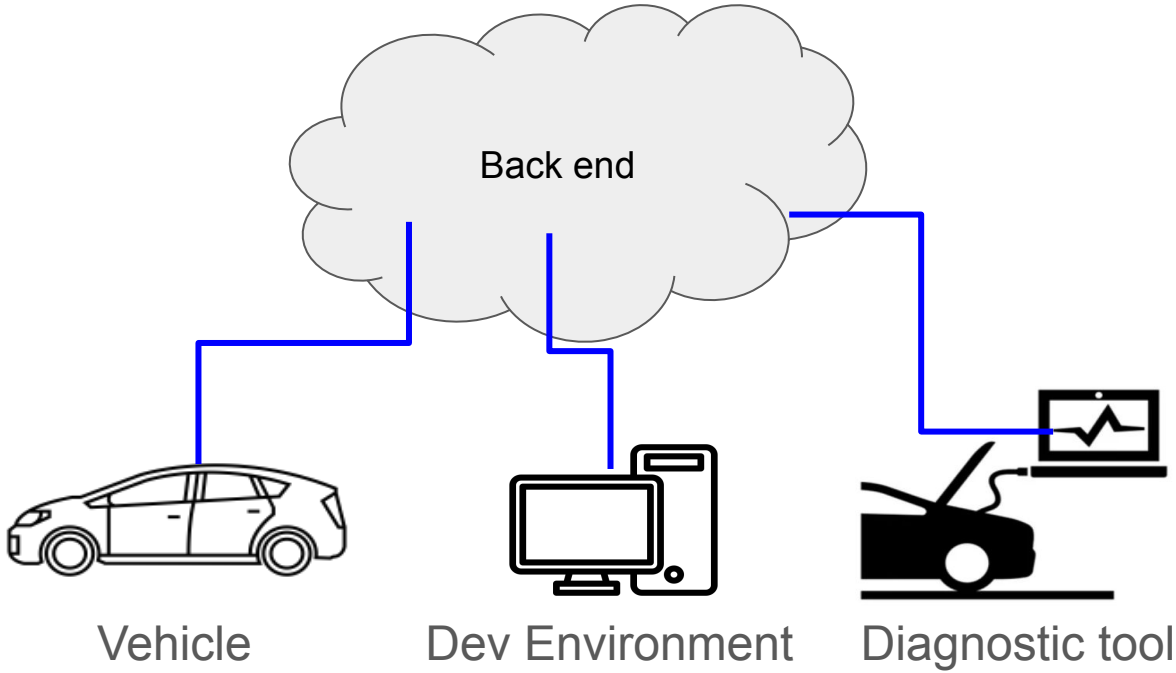
05-2

Trusted Platform Services (TPS)

Example Keystore



Example Keystore



Trusted Platform Service benefits

01
Standardised services

Open standard: less internal effort
Competition between vendor

02
Maximise portability

The same use application could run regardless if the device has a TEE, a secure element or nothing (example during development).

03
Service discovery

Flexibility: can query which services are available.

Thank you