THALES
Building a future we can all trust

GP CVF Yokohama
22nd of May 2025

How Secure Elements enhance future Smart Mobility
eSE as HSM extension
µController with AUTOSAR Classic

www.thalesgroup.com

Many cyber challenges throughout the vehicle life cycle

Secured by design in the automotive life cycle

Architecture & Development → Integration & Tests → Type Approval then Mass Production → Update & Repair → Circular Economy

Progressive Activation of Cyber mechanisms

No → Partly activated → Fully activated

Impacts to Key Management (Crypto)

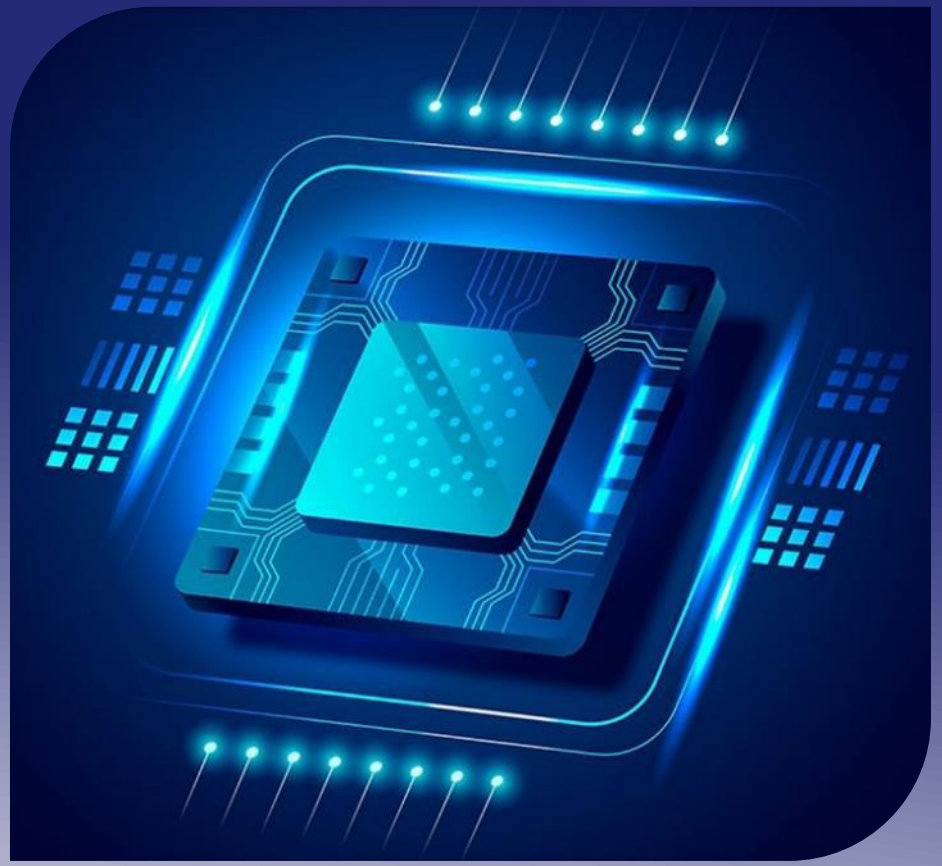
Development keys → Pre-production keys → Production keys → Decommissioning

Key renewal

Crypto update

Re-use

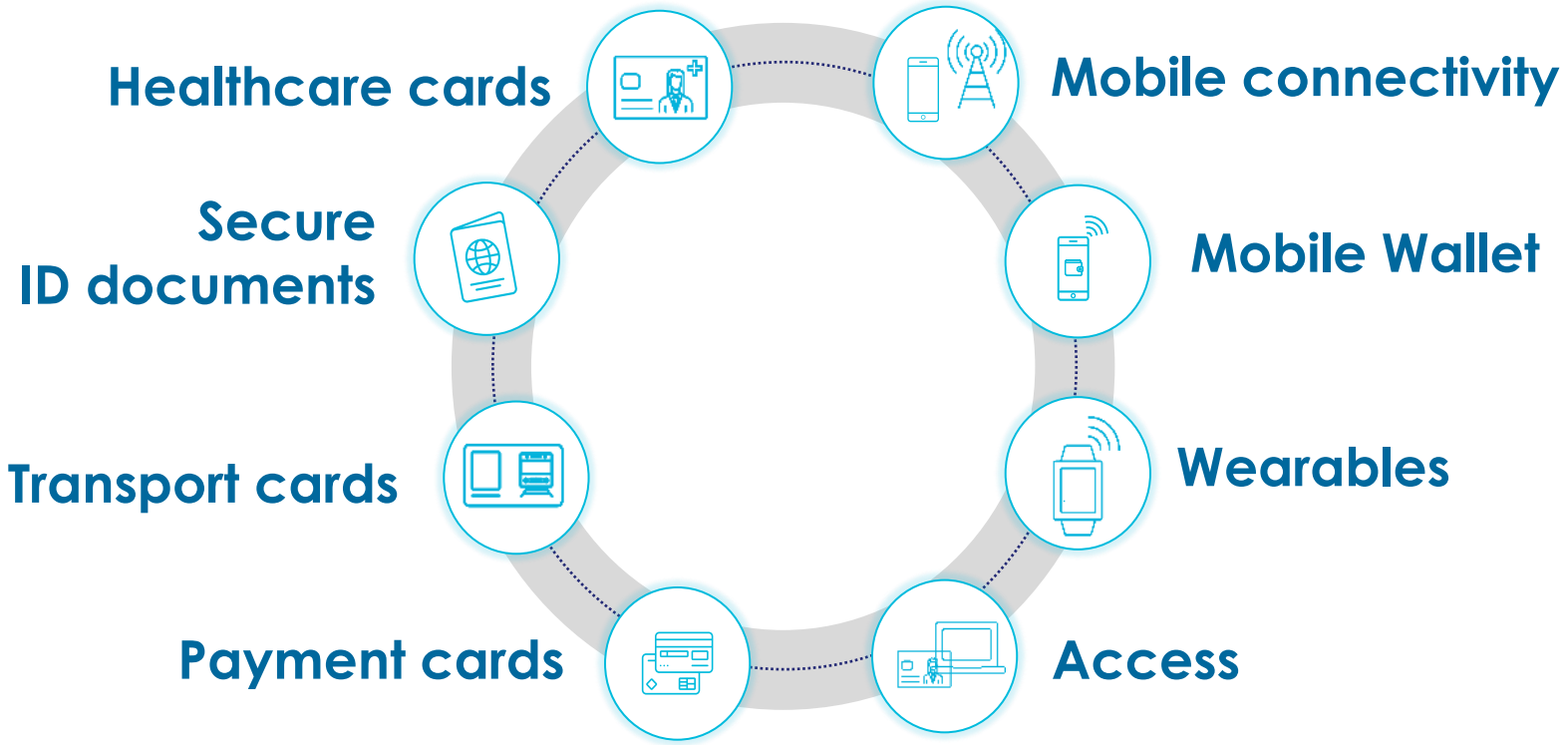
Secure Element is ideal to support and answer these challenges



What is a Secure Element?

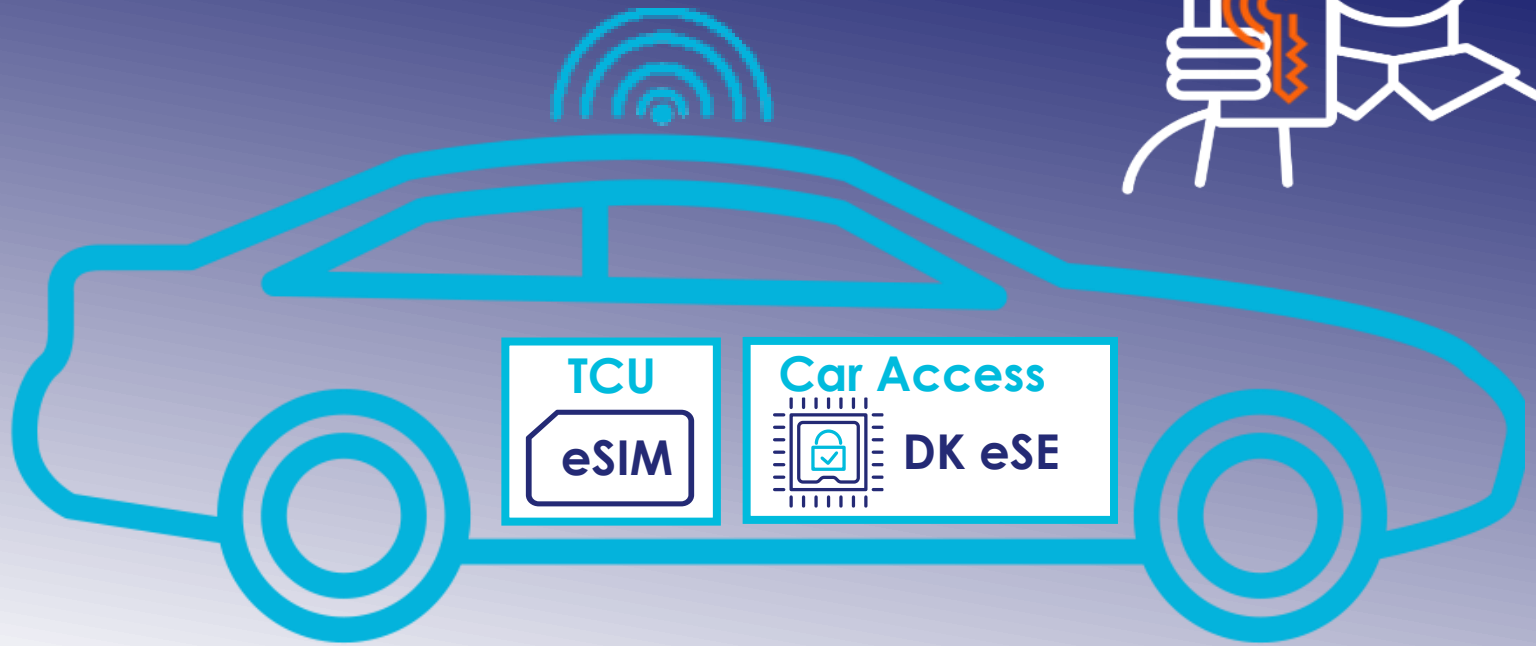
Present in your daily life for decades

Source GlobalPlatfom



Deployed in all connected vehicles

Car Connectivity Consortium
Digital Key



Benefits of embedded Secure Elements in Automotive

a **secure execution environment**

- **Tamper resistant**[1]
- **Execution of crypto service** and **business logic**
- **Separated resources**

standardized, proven, mass-produced

- Standardized **protocols** & **mechanisms**
- **interoperable**[2] & **upgradable** applications
- Well-defined **certification schemes with high assurance** (EAL4+)

with interesting **complementary properties**

- **Agile**
- **Low consumption**
- **More than 1Mbytes available**
- **Good performances** boot time / crypto operations

[1] Resistant to physical attacks, AVA.VAN.5

[2] Interoperability of the binary level

Use cases with embedded Secure Elements in Automotive

> Key management life cycle

- Personalize eSE during its production
- Ease transition phases from development to production
- Allow secure key provisioning at Tier1 manufacturing and OEM assembly line

> Business logic control

- Business logic implemented eSE
- Enforce control of key and crypto engine usage

> Crypto agility

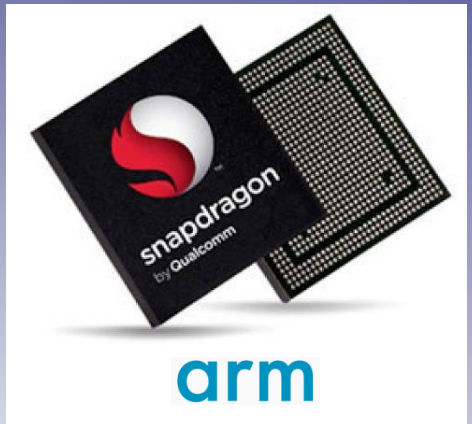
- Provide secure key provisioning on-field, at repair
- Tackle circular economy
- Support OS and Applet upgrade
- Ensure PQC readiness

OUR FOCUS TODAY

µProcessor A-Class
(TEE, TZ)

Examples:

- Telematic
- Central HPC
- Infotainment
- ADAS Supervisor
- …



µController
(HSM)

Examples:

- VHL Access
- VHL Health
- EV Charging
- Anti Chip Tuning
- Zonal Controller
- …

EVITA project – HSM Version

> HSM Light

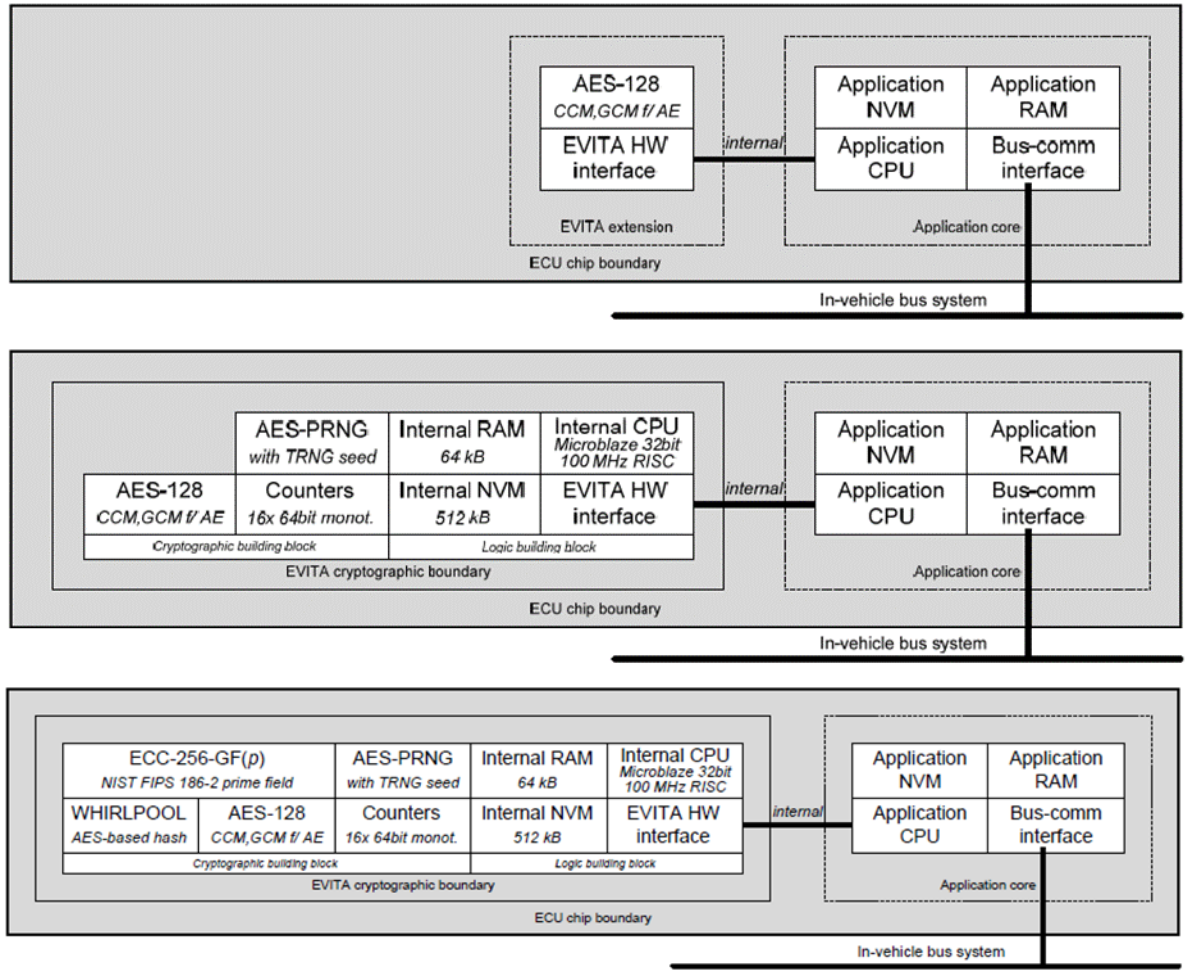
- For security-critical sensors and actuators

> HSM Medium

- As hardware extension to the ECU connected to the in-vehicle domain controls

> HSM Full

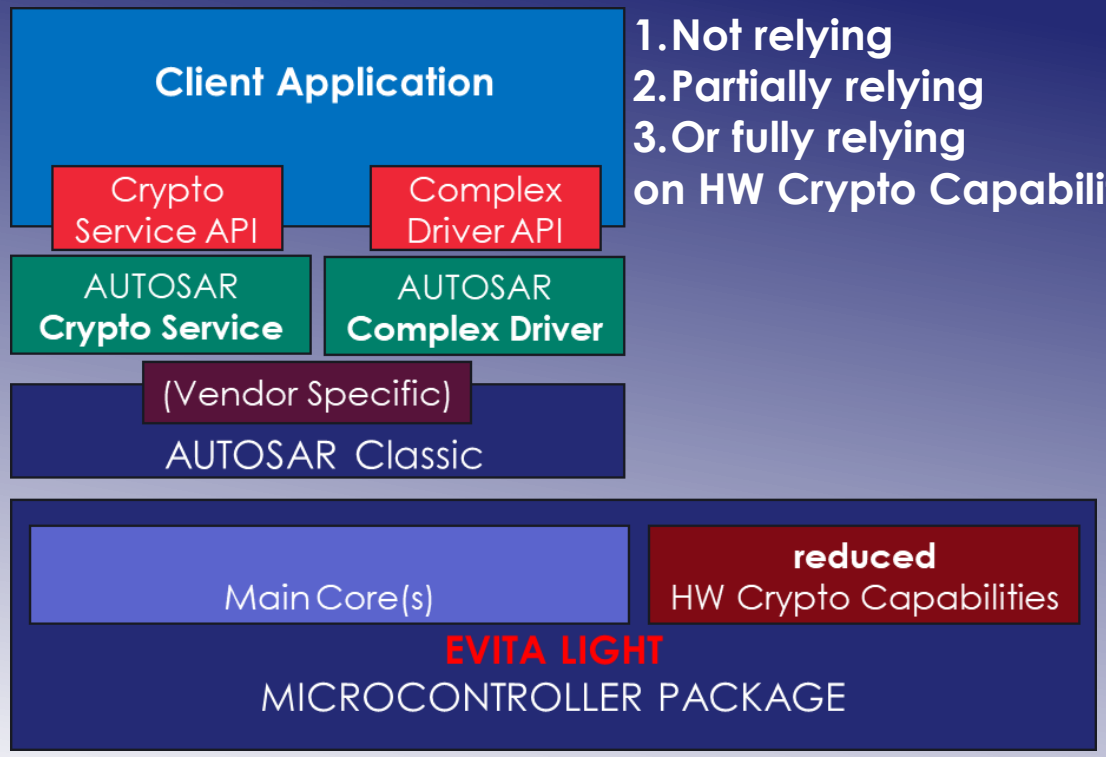
- As hardware extension to the ECU specifically responsible for V2X applications



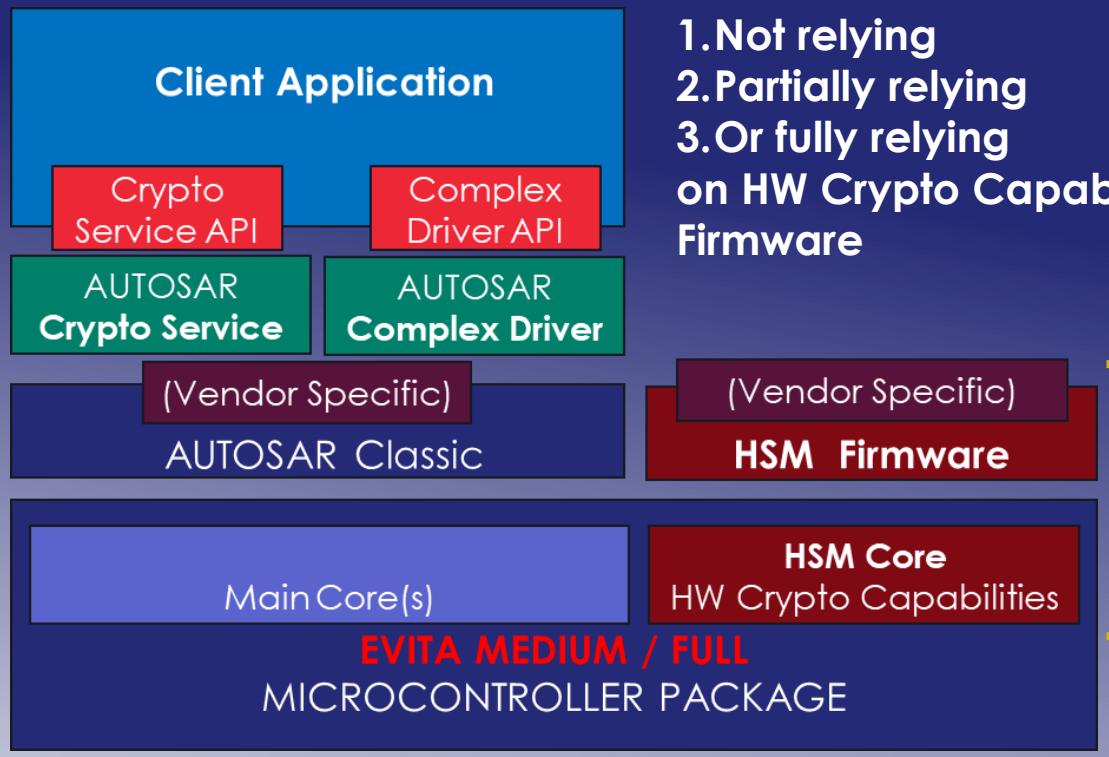
- Symmetric crypto engines
- + CPU to execute HSM Firmware with privileged access to Flash / RAM area
- + Asymmetric crypto engines

Implementation variants with AUTOSAR + Evita HSM

AUTOSAR Crypto Service/ Complex Driver



1. Not relying
2. Partially relying
3. Or fully relying on HW Crypto Capability



1. Not relying
2. Partially relying
3. Or fully relying on HW Crypto Capability + HSM Firmware

Split FW / HW?

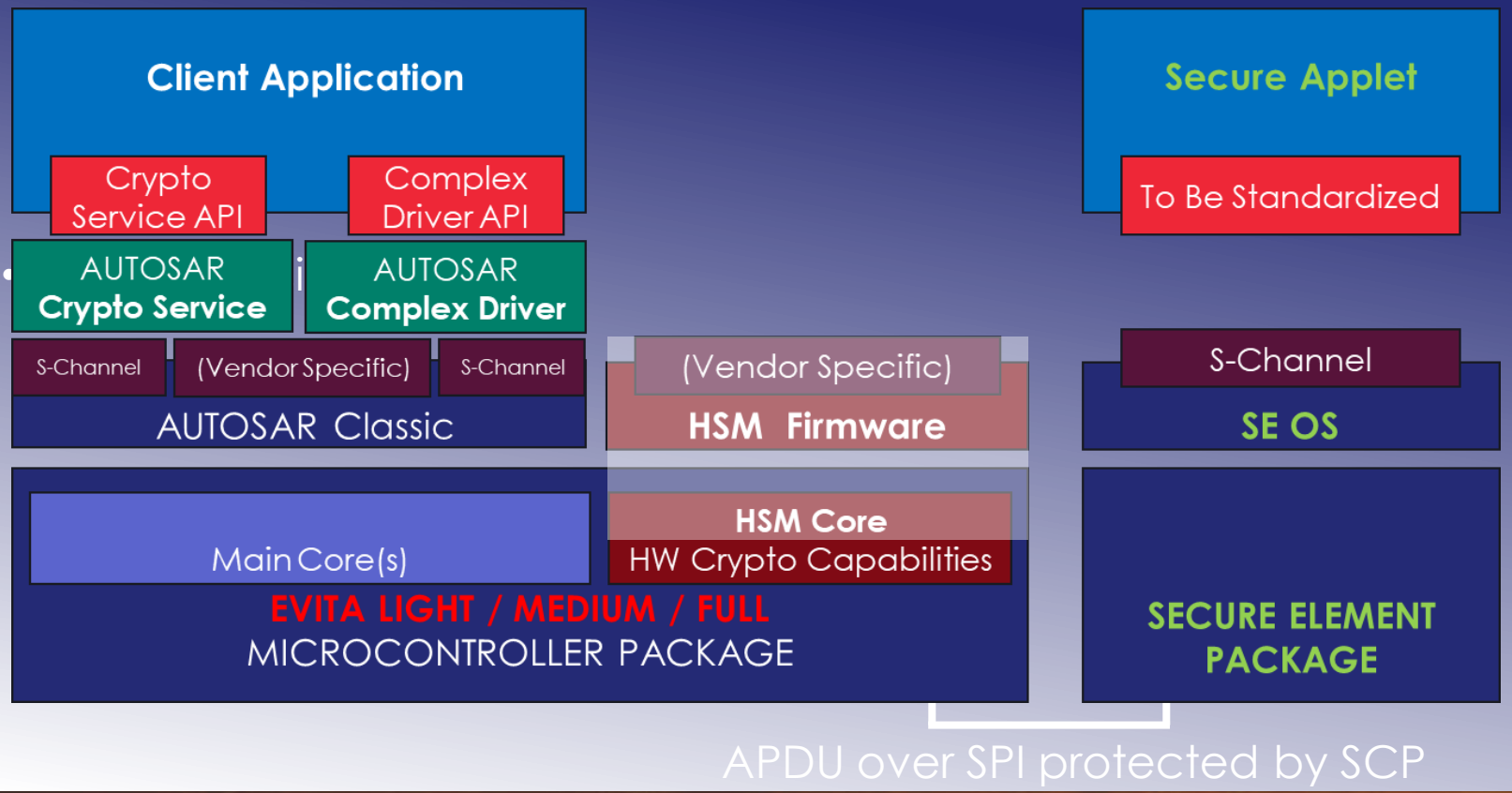
A nightmare to understand how things are really working

Difficult to demonstrate security objectives are met and evaluate resistance level

Obscure by design

- > Lack of clarity on how/where crypto services are implemented
- > As many implementations as actors to fit given security goal.
- > Supported features are vendor (HW and FW) specific
- > No resistance to hardware attacks
- > Maturity is difficult to evaluate
- > Frozen capabilities, no agility
- > Huge costs and planning impacts each time a change is required
- > Limited cryptographic algorithms
- > No or low capability to fix vulnerability after deployment

Extension of HSM capabilities with Secure Elements



HSM

- Legacy implementation
- Access to internal resources

eSE

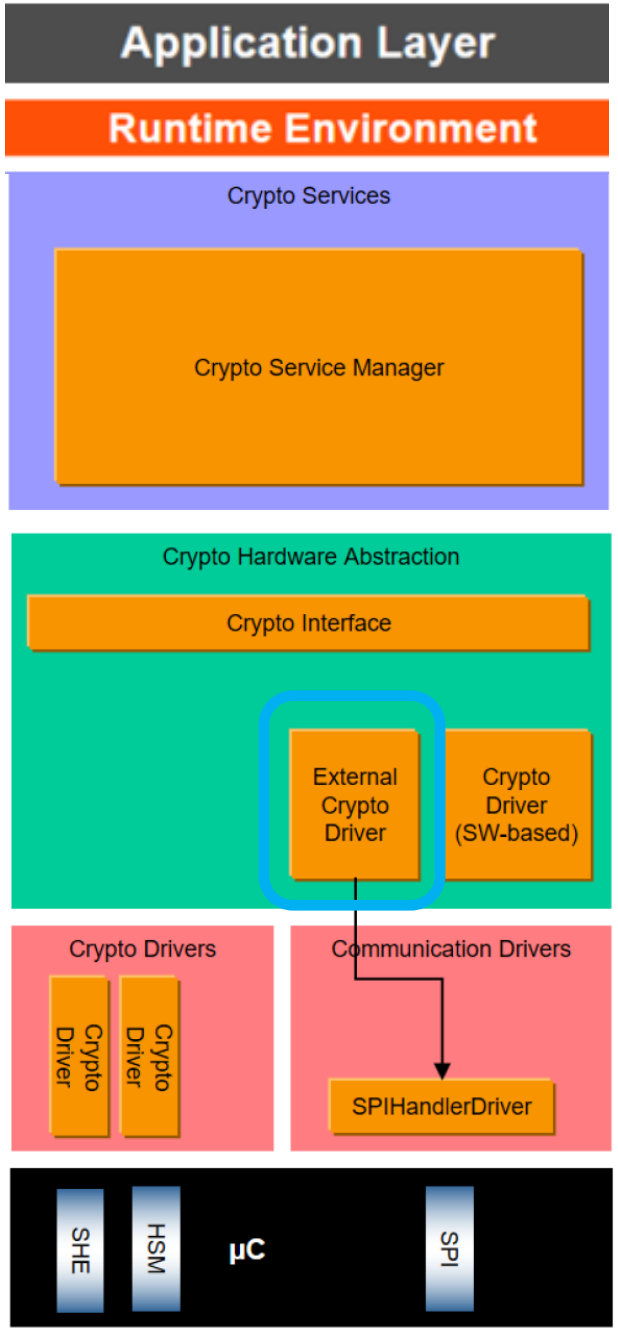
- Tamper resistance
- Certification
- Advanced crypto algorithms Diffie Hellman, miscellaneous ECC curves, etc.
- Crypto agility. Upgradable, PQC readiness
- Key Management Life Cycle
- Business logic

Take benefit of the both HSM and Secure Element.
Crypto services always running in secure environment (HSM or SE)

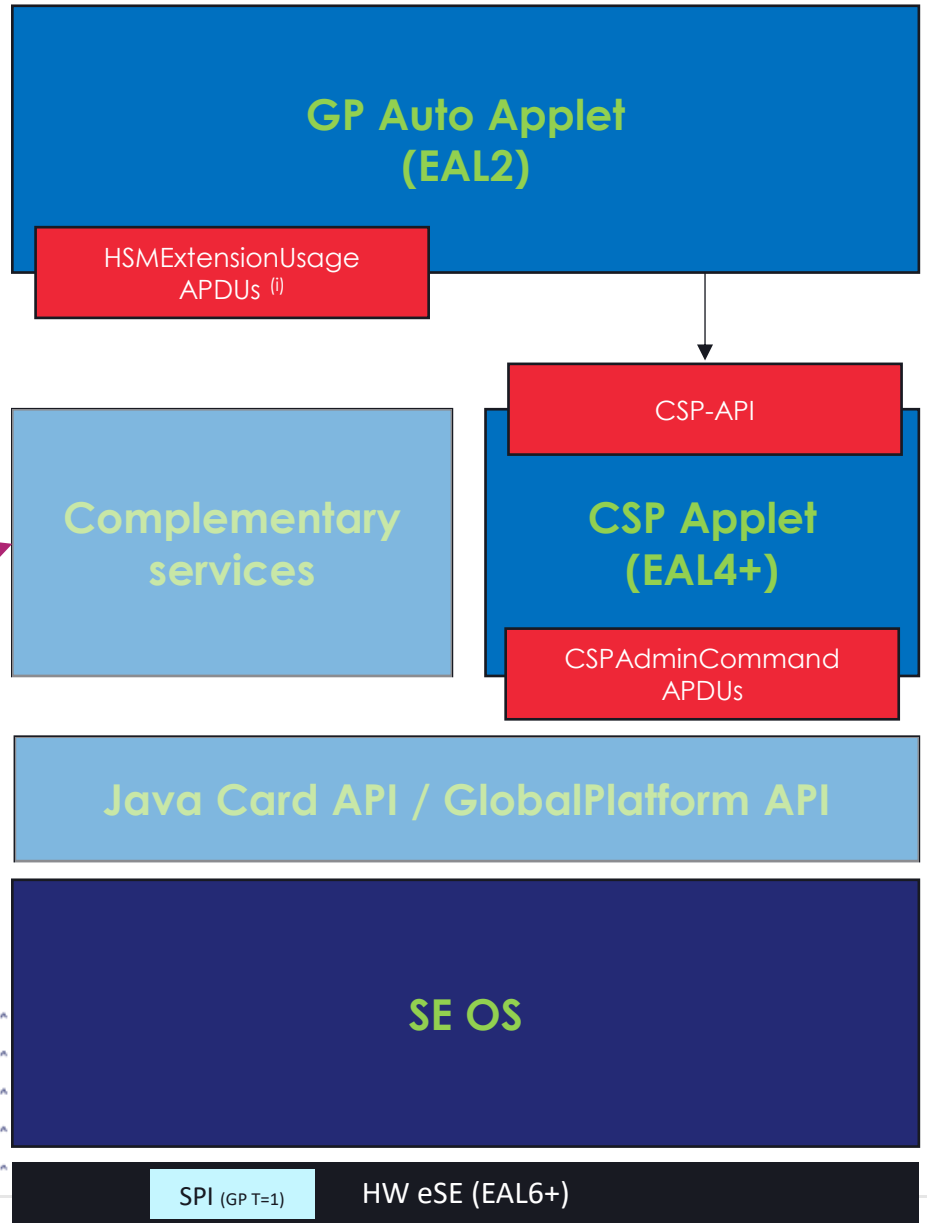
USECASE	HSM ROLE	eSE ROLE
Secure binding between MCU and eSE	• Secure storage of SCP[1] Key / MCU side [1] Secure Channel Protocol (e.g. SCP03)	• Secure storage of SCP[1] Key / eSE side • Secure Channel Protocol implementation
Secure Boot of MCU	• Before releasing from reset, CMAC signature verification of immutable boot area • Hash computation	• Asymmetric signature verification of updatable area(s) against pre-defined Root Of Trust
MACSec between 2 ECUs	• GMAC computation/verification using Secure Association Key	• CAK[1] provisioning/learning • MACSec key agreement and SAK[2] creation [1] Connectivity Association Key [2] Secure Association Key
Vehicle to Cloud mTLS	• Not supported	• Manage critical steps during mTLS handshake
Digital Key (DK)	• Not relevant in DK protocol • Secure transfer of UWB keys to UWB sub-system	• Digital Key storage • Implementation of the CCC protocol between vehicle and device

AUTOSAR Layered View with CSM

> Use external crypto driver to handle APDU towards eSE



- HSMExtensionUsage APDUs handling functions of CSM AUTOSAR interfaces:
  - Hash
  - Mac
  - Cipher
  - Authenticated Encryption with Associated Data (AEAD)
  - Signature
  - …



- GP Auto Applet itself relying on CSP-API

- CSPAdminCommand APDUs handling key management functions of CSM AUTOSAR interfaces:
  - Key Generation
  - Key Derive
  - …

- Despite a very rich set of functions defined within CSP, some complementary services can be defined to propose a GP Auto Applet that fits most of needs:
  - Secure vault to be able to write sensitive data inside eSE, cypher and extract later on demand…
  - Complementary missing algorithm

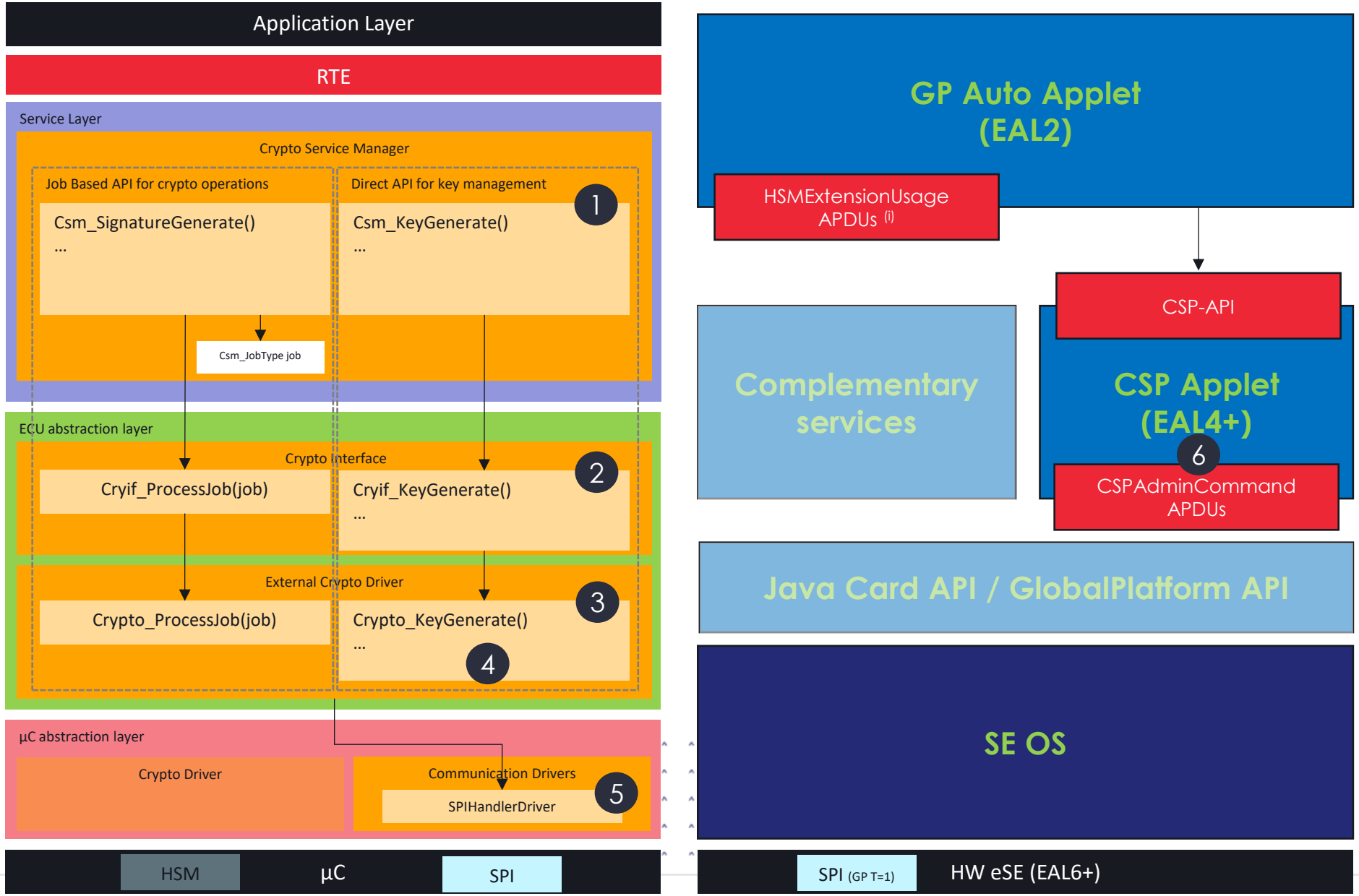
(i) To be standardized

Going deeper into mapping to AUTOSAR APIs

> Two examples

- Csm_KeyGenerate
  - Direct API involving CSPAdminCommand APDU to CSP Applet
- Csm_SignatureGenerate
  - Job based API involving HSMExtensionUsage APDU to GP Auto Applet

Example 1 Csm_KeyGenerate: Direct API involving CSPAdminCommand APDU to CSP Applet



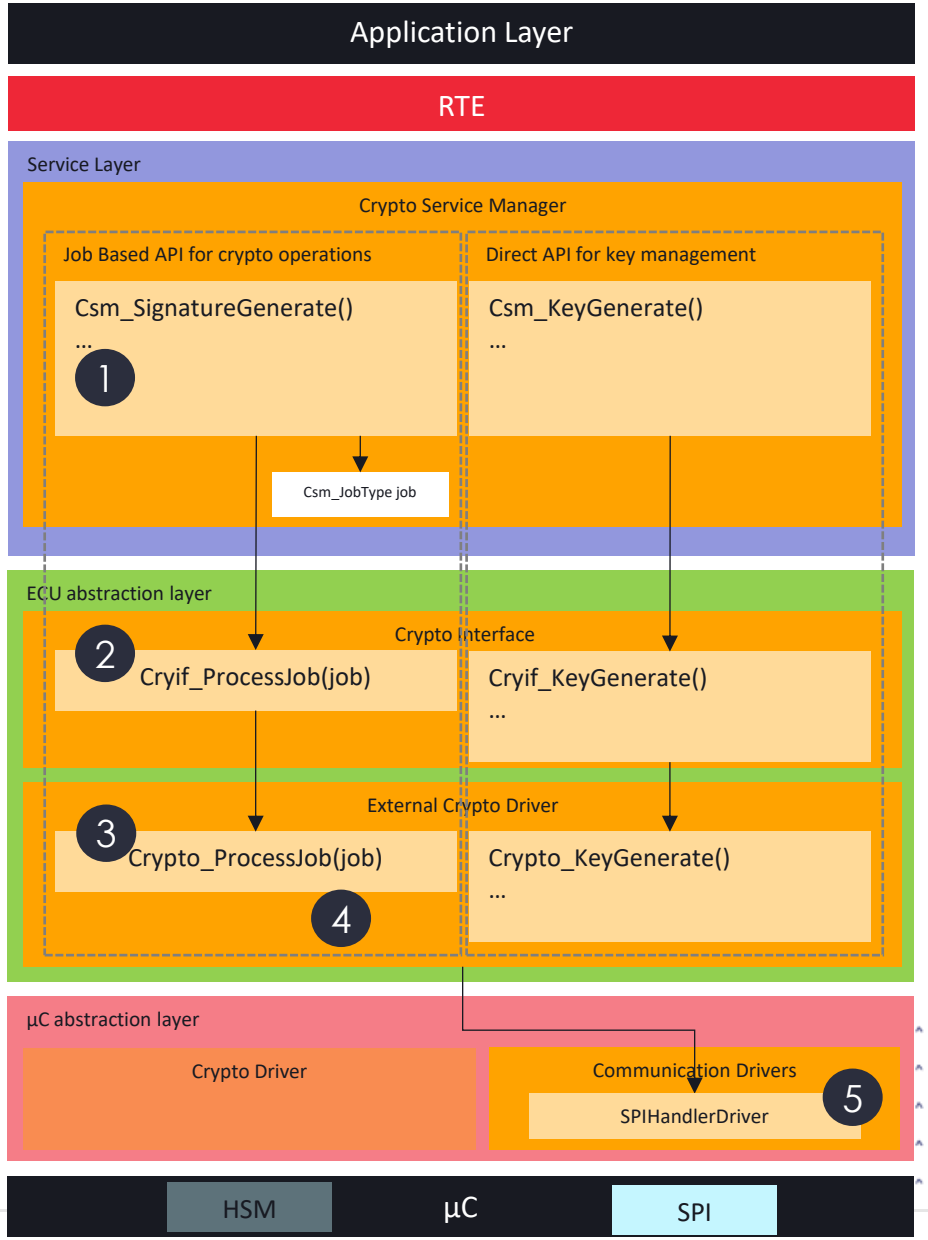
Pre-requisite: SCP channel opened with CSP Applet upon Csm_Init()

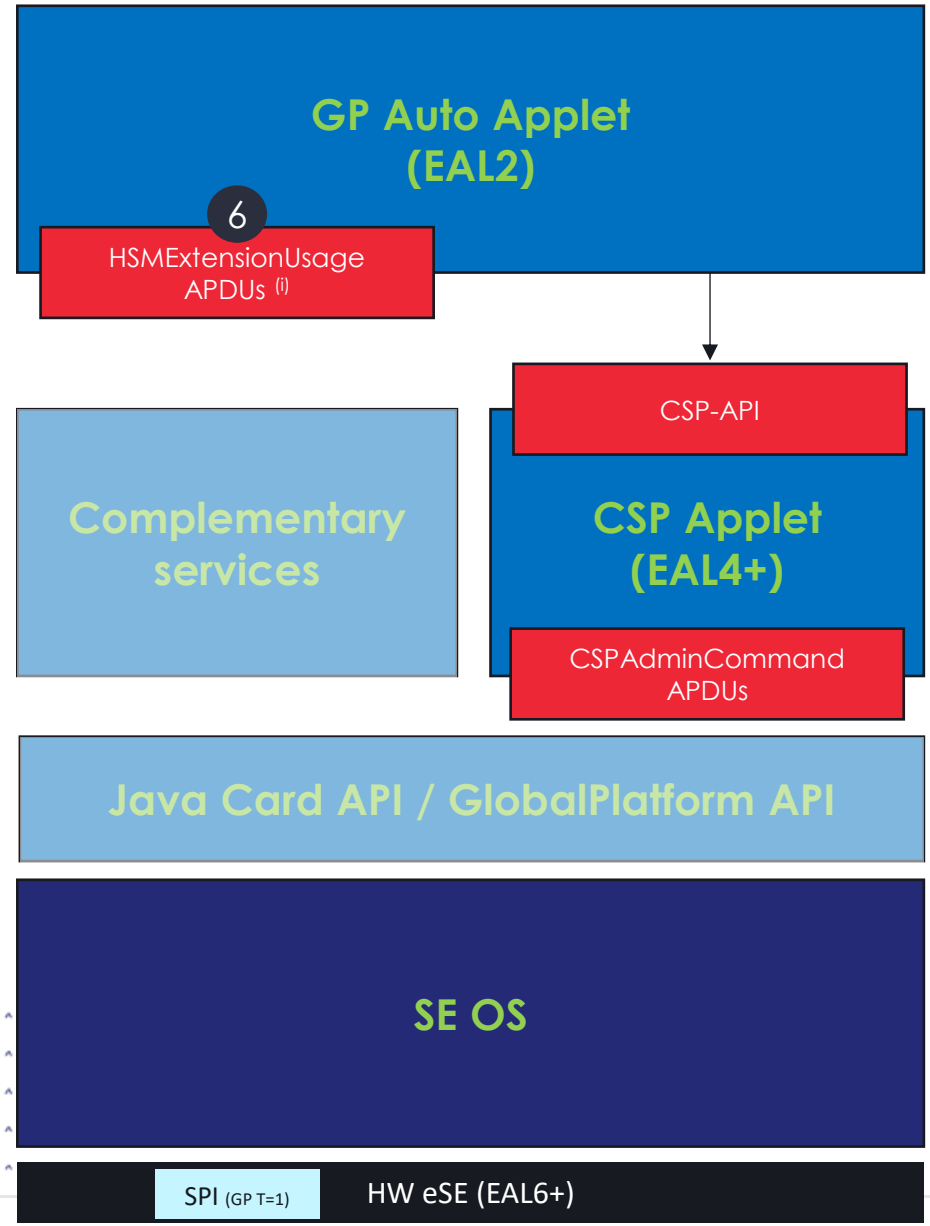
1) Application calls Csm_KeyGenerate (service layer)
2) Which calls Cryif_KeyGenerate (crypto interface layer)
3) Which calls Crypto_KeyGenerate (external crypto driver layer)
4) External crypto driver prepares CSPAdminCommand APDU for CSPGenerateKey
5) Communication driver sends CSPAdminCommand APDU for CSPGenerateKey over SPI to CSP Applet
6) CSP Applet computes CSPAdminCommand APDU for CSPGenerateKey and sends back APDU(s) response to µController

Various Host layers process response up to Application

(i) To be standardized

Example 2 Csm_SignatureGenerate: Job based API involving HSMExtensionUsage APDU to GP Auto Applet





Pre-requisite: SCP channel opened with GP Auto Applet upon Csm_Init()

1) Application calls Csm_SignatureGenerate (service layer)
2) Which calls Cryif_ProcessJob (crypto interface layer)
3) Which calls Crypto_ProcessJob (external crypto driver layer)
4) External crypto driver prepares HSMExtensionUsage APDUs for ComputeSignatureInit / ComputeSignatureUpdate
5) Communication driver sends HSMExtensionUsage APDUs for ComputeSignatureInit / ComputeSignatureUpdate over SPI to GP Auto Applet
6) GP Auto Applet computes HSMExtensionUsage APDUs for ComputeSignatureInit / ComputeSignatureUpdate relying on CSP-API and sends back APDUs response to µController

Various host layers process response up to Application

(i) To be standardized

Opportunity for Standardized APIs, interoperability testing and security certification?

- Today eSE are mainly used with **proprietary solutions** that is a **mainstream adoption drawback**
- So even if some eSE are used for a specific use cases (Digital Key, Qi …) It is not easy to extend it for generic services, especially for AUTOSAR.
- GP could be the way to develop such “Applet” offering a generic a set of standardized APIs to be run on top of an “eSE with JVC OS”

> On-going discussions with AUTOSAR Security Working Group – Classic Platform

THALES
Building a future we can all trust

Thank you

www.thalesgroup.com