Design of secure embedded systems

HSM (Hardware Security Module)

Part A

Prof. Sergio Saponara,
Dip. Ingegneria della Informazione, Università di Pisa
sergio.saponara@unipi.it
+39 3468790937

Outline Part A

- Safety&Security correlation and needs of embedded security
- Analysis of HW secure specifications for ECUs:

SHE (Secure HW extension),

EVITA (E-safety vehicle intrusion protected applications) and IS0 21434

- Architectures and components of HSM:

Example analysis of HSM in commercial automotive MCUs (TPM, ARM, Infineon)

Outline Part B

- Architectures and components of HSM:

Analysis of HSM in commercial MCUs (NXP, ST, Renesas)

Analysis of commercial IPs for HSM (by Rambus, Synopsys, …..)

- Secure key storage/management and advanced secure features in EPI HSM
- HSM evolution roadmap
    - Intrusion Detection System (IDS) support
    - Post-quantum cryptography (PQC) and tracking of NIST standardization
- Q&A session

Safety & security in autonomous & connected intelligent machines (robots, vehicles, industry 4.0, defense,…)

- Machines (e.g. Vehicles) are connected Inside to create a distributed control system

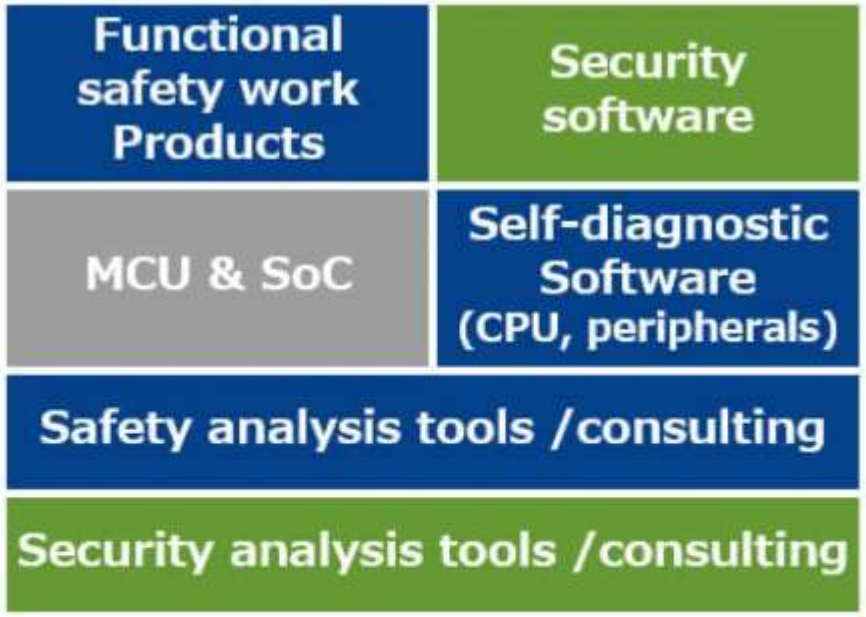
(ECUs –Electronic Control Unit- using controller area networks mixing multiple standards such as LIN-Local Interconnect Network, CAN/CAN FD – Controller Area Network Flexible Data-rate and in advanced vehicles Flexray and/or Automotive Ethernet)

- Machines (e.g. Vehicles) are connected Outside to enable V2X (Vehicle to Everything) services using Bluetooth and/or 802.11-like DRSC (Direct Short Range Connectivity) and/or Cellular-V2X (4GLTE and 5G) and/or GNSS (Global Navigation Satellite Systems)

Security is important and sometimes it is important to sell it for safety CS is still seen as an added feature unlike safety. But in general, they are correlated. Think of autonomous vehicles,

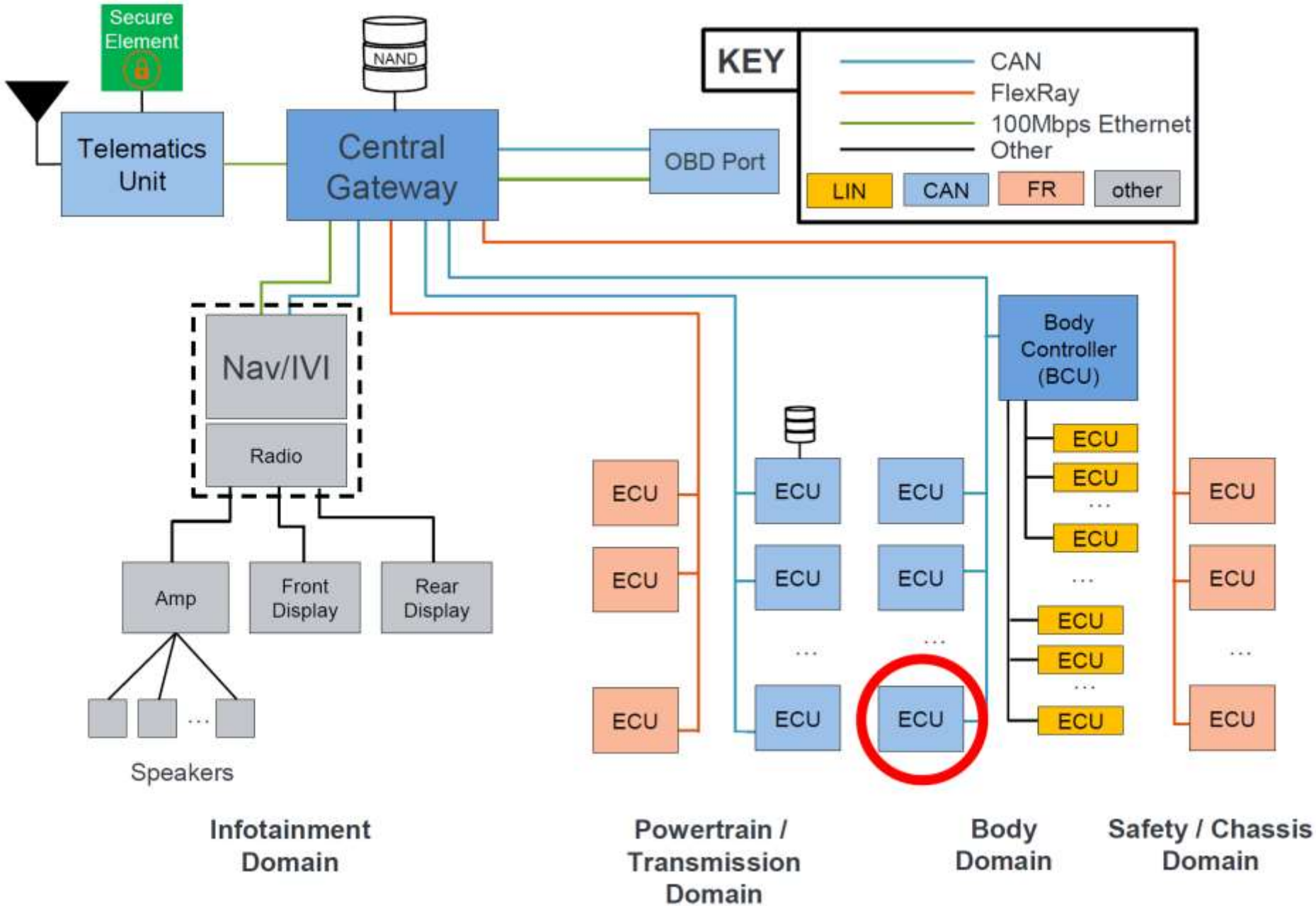
Safety & security in autonomous & connected vehicles

- Security is strictly linked with safety since security vulnerability can be used in attacks to maximize severity consequence of faults/denial of services
- The increased connectivity of vehicles (i.e. automotive, trucks, ships, drones, …) in the Internet-of-Things (IoT)/ Internet-of-Vehicles (IoV) era increases the attack surface and vulnerability (→ increased **risk probability**)



A security fault can turn into a safety fault.
A lot of push for CS is coming from the world of vehicles, that have the safety and security linked together.

Safety & security in autonomous & connected vehicles

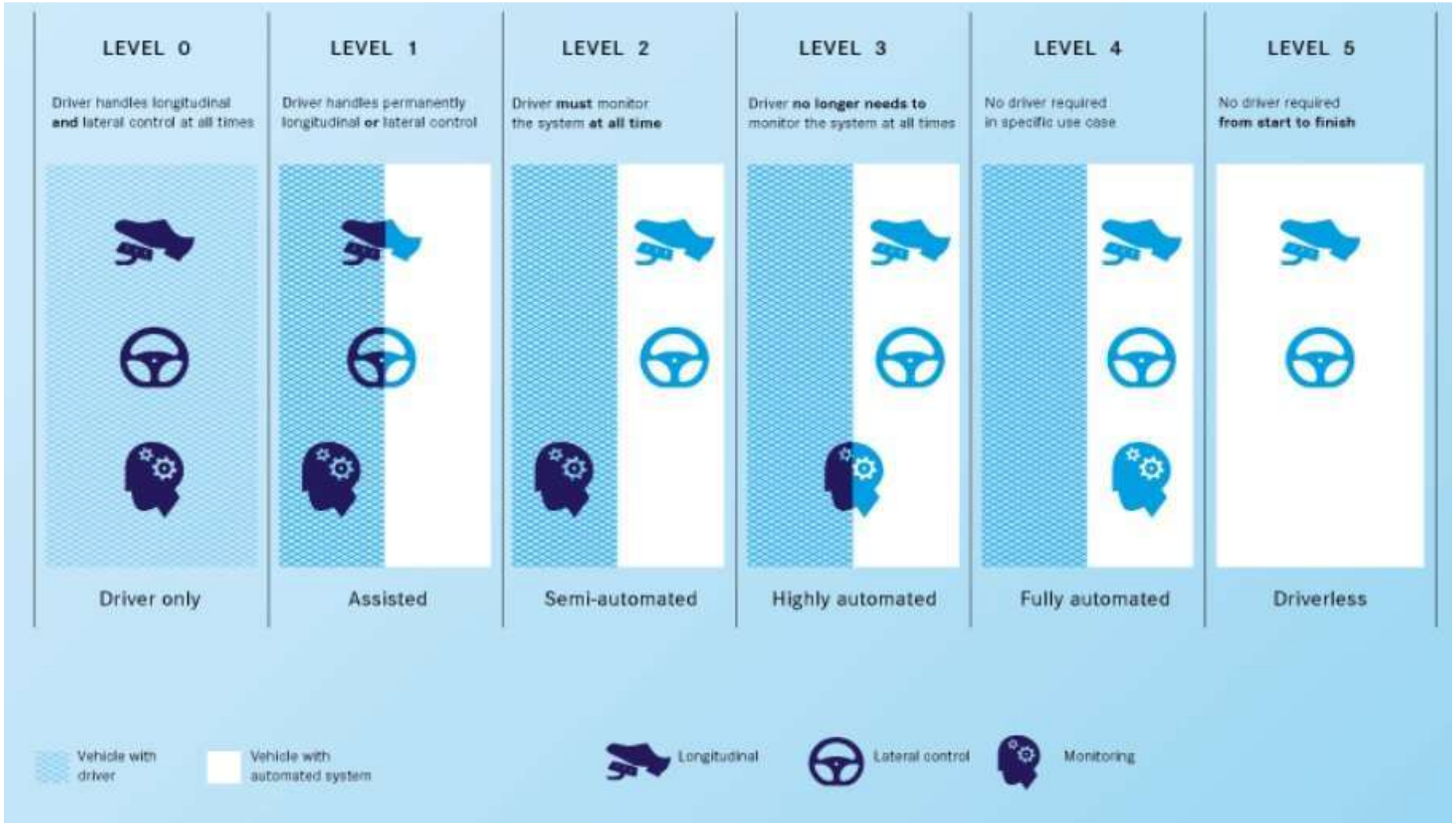


Safety & security in autonomous & connected vehicles

In SW-defined autonomous machines such as car Advanced Driver Assistance Systems (ADAS), Unmanned Ground Vehicles (UGV) a cyber attack leads to severe safety issues

→

increased **severity effect**



Safety & security in autonomous & connected vehicles

Vehicle security needs are related to:

- **Confidentiality** of the users (driver, passengers) data:

e.g. travel habits of users, driving style, biometric data of people acquired to test e.g. driver attention or health status, people preferences in terms of memorized car radio channels, seat configuration, climate configuration

- **Confidentiality** of the vehicle data:

e.g. maintenance status, Software (SW) and Firmware (FW) update, vehicle identification, vehicle aging, vehicle history in terms of average/maximum speed, traveled kms, crashes,…

Safety & security in autonomous & connected vehicles

Vehicle security needs are related to:

- **Authentication** (Fingerprinting)③

(key: authentication of driver) (auth of SW update)

Authentication of driver accessing/starting the vehicle, of the SW/FW provider and SW/FW version in case of SW update, of the sensors data①, of the navigation maps and satellite signals②, of the wireless and/or wired ECU sources in Inside and 0utside vehicle networking

- **Integrity**

of the SW in case of SW update,

of the data acquired by sensors,

of the data exchanged between ECUs,

of the data exchanged with other vehicles, with the infrastructure

Auth. related to driver, to SW provider and to SW itself.

① Imagine fake sensor data

② More related to protection for important people, but for normal ones may be too much.

Auth: I fake the source. Integrity: I change the value of actual authentic data.

The difference between **authentication** and **integrity** of sensor data lies in what each concept ensures:

- **Authentication of sensor data** verifies **who** or **what** generated the data. It ensures that the data comes from a legitimate source, preventing attackers from injecting false sensor readings by impersonating a trusted device. This typically involves cryptographic techniques like digital signatures, message authentication codes (MACs), or public-key cryptography.
- **Integrity of sensor data** ensures that the data has not been **tampered with** or altered in transit or storage. Even if the data comes from an authenticated source, an attacker could still modify it. Integrity mechanisms (like cryptographic hashes, checksums, or MACs) detect any unauthorized changes.

In short, **authentication** confirms the source of the data, while **integrity** ensures that the data remains unchanged from source to destination.

③ Why fingerprinting? Not only question of pulling a pin from phone etc. but also biometry. Same for HW with PUF. Again, this might be too extreme sometimes.

Safety & security in autonomous & connected vehicles

Security must be guaranteed as well as

Reliability (linked to safety)

Availability

Traceability (non repudiability) Check and log what happens to ensure that you know the root of an attack

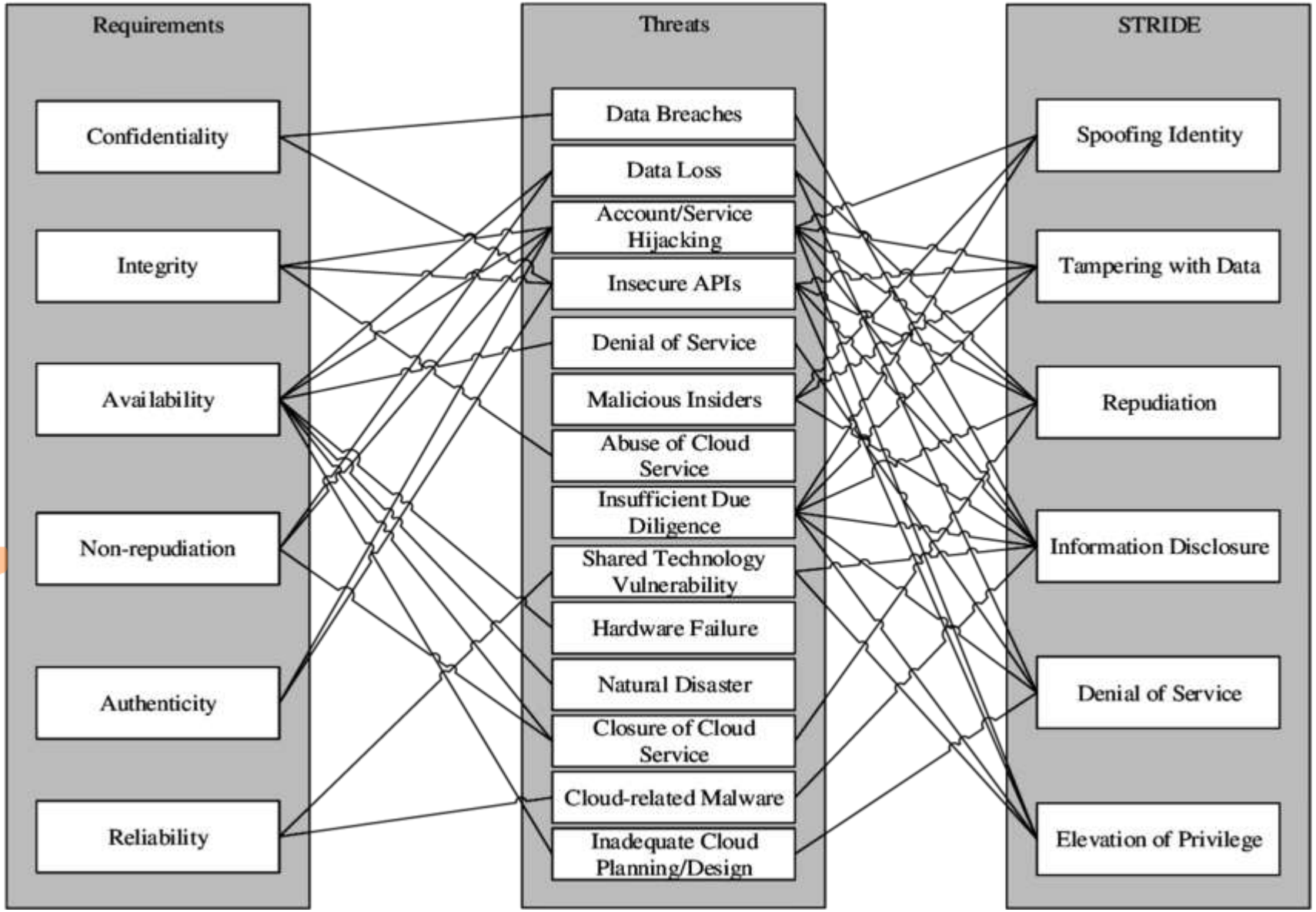
Low-cost of the solution for the automotive market

Energy-efficiency of the solution for the automotive market

Problem is: I need to understand which are the vulnerabilities and attack avenues to determine countermeasures. For this, we need a model.

STRIDE threat model: is a good compromise, you take 6 threats.

Define threats from STRIDE and find requirements. You can still have risks: If it is known that you used stride model, then you know where most of the focus went.



STRIDE threat model

You achieve non repudiation through logging, hacking. This reduces also probability of insider attacks, but also attacks in general

Often the baseline for other attacks

SPOOFING

In the context of information security, and especially network security, a spoofing attack is a situation in which a person or program successfully identifies as another by falsifying data, to gain an illegitimate advantage.

TAMPERING

Tampering can refer to many forms of sabotage but the term is often used to mean intentional modification of products in a way that would make them harmful to the consumer.

REPUDIATION

In digital security, non-repudiation means a service that provides proof of the integrity and origin of data, or an authentication that can be said to be genuine with high confidence.

INFO DISCLOSURE

↳ Am I subject to data privacy? If yes, put features to avoid that

Information disclosure is the unwanted dissemination of data, technology, or privacy. legal and political issues surrounding them. It is a violation of data privacy[2] or data protection. The challenge of data privacy is to use data

DENIAL OF SERVICE

A denial-of-service attack (DoS attack) is a cyber-attack in which the perpetrator seeks to make a machine or network resource unavailable to its intended users by temporarily or indefinitely disrupting services of a host connected to the

↳ Consider probability and countermeasures

ELEVATION OF PRIVLEGE

Privilege escalation is the act of exploiting a bug, design flaw or configuration oversight in an operating system or software application to gain elevated access to resources that are normally protected from an application or user.

Risk comes especially if you're connected on network. Countermeasures: IDS.

The stride model is good because it's what a model. People will know and it is more easily seen as an accepted solution.

Needs of embedded security for edge devices

→ With embedding security in HW we are talking about edge devices in which you have scarce area, not a lot of power etc ①

A
- Edge devices (e.g. vehicles) must be protected by ensuring encryption (at TX side, and hence decryption at RX side) of communication, secure device configuration, boot management, power management, secure SW/FW update, to allow for confidentiality, authentication and integrity of data and programs

- Edge devices (e.g. vehicles) must be protected by attacks from cloud data centers but still ensuring low-power of the solution
- Missing of a single unified standard for embedded security, but many HW/SW solutions at State-of-art

① You cannot use that much power like server size.
A: No guarantee STRIDE models you should consider:

Needs of embedded security for edge devices

- For vehicles need of Hardware(HW)-based security module since pure SW security (i.e. secure SW apps running on a general purpose HW missing security dedicated features) is energy not-efficient, is critical in terms of computing latency (missing time deadlines and not deterministic), is easier to attack.
- Best solution for vehicle security is a HW/SW one where secure SW apps ensure flexibility by running on HW including dedicated resources for a real-time and cost-effective implementation of computing intensive primitives for:
  - Accelerating symmetric and asymmetric encryption/decryption,
  - Accelerating digital signatures and verification and hash-based data integrity checks
  - HW and on-chip generation and storage of secrets (e.g. secure keys)
  - HW restricted access rights to memories/peripherals

Those are the usual solutions you can expect to find, and mostly at hardware level.

Attacks to vehicle already happened

https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

Man in the middle» attack with spoofing and denial of service (Attackers: C. Miller and C. Valasek on a Jeep Cherokee)

Attacks to vehicle already happened

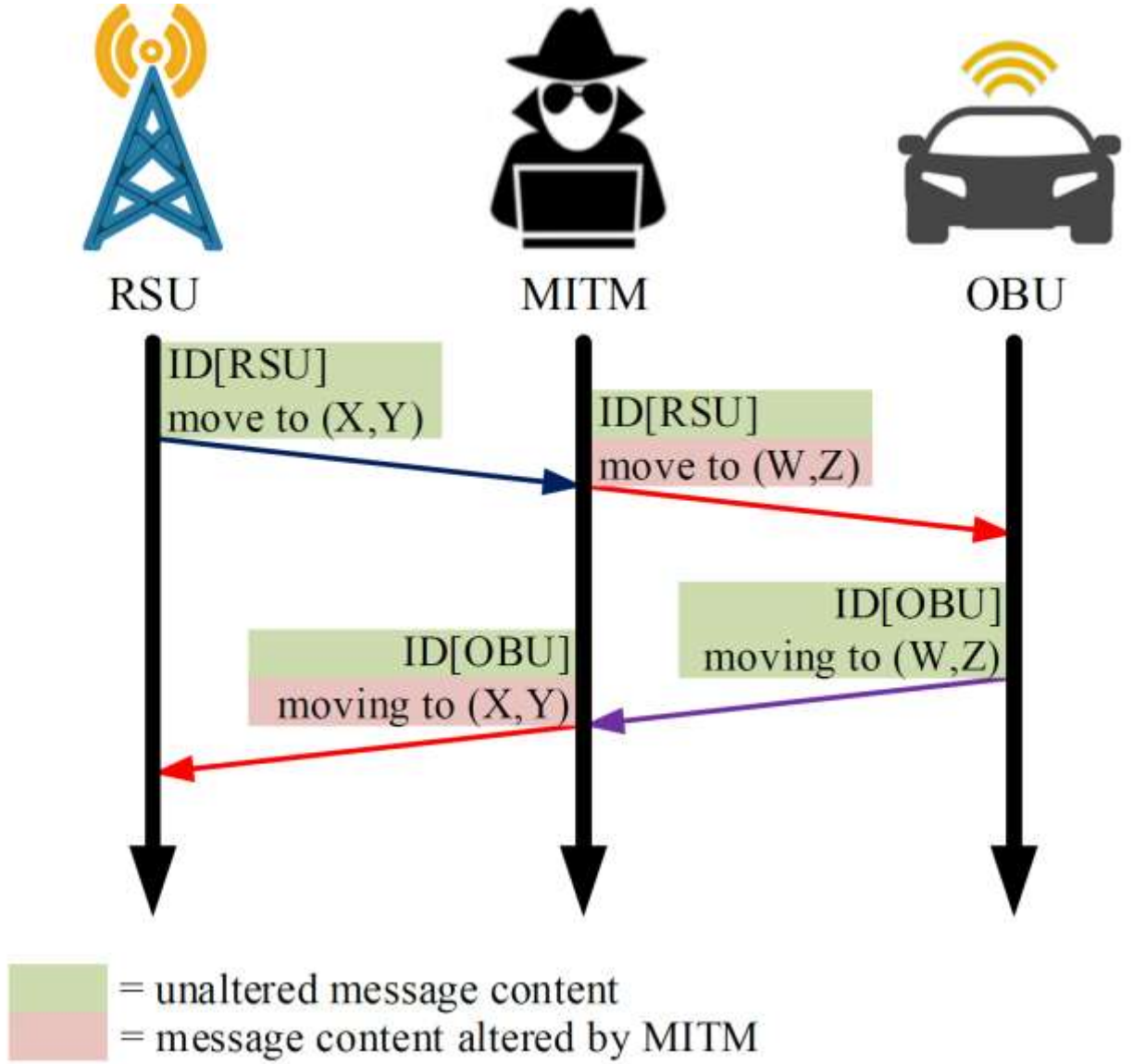
https://www.youtube.com/watch?v=krSj81thN0w

In 2021 R. P. Weinmann and B. Schmotzle (**Comsecuris)**
using a drone and a wi-fi card attacked remotely a Tesla Model 3 by targeting with an attack the Vcsec unit of Tesla (connected to the wi-fi car gateway, but also to the internal CAN).
They were able to disable the car firewall and by using the car gateway and then using the CAN bus they were able to send messages internally in the car thus opening the car door.

CANSECWEST 2021: Tbone Drone vs Tesla - Ralf-Philippe Weinmann & Benedikt Schmotzle, Comsecuris

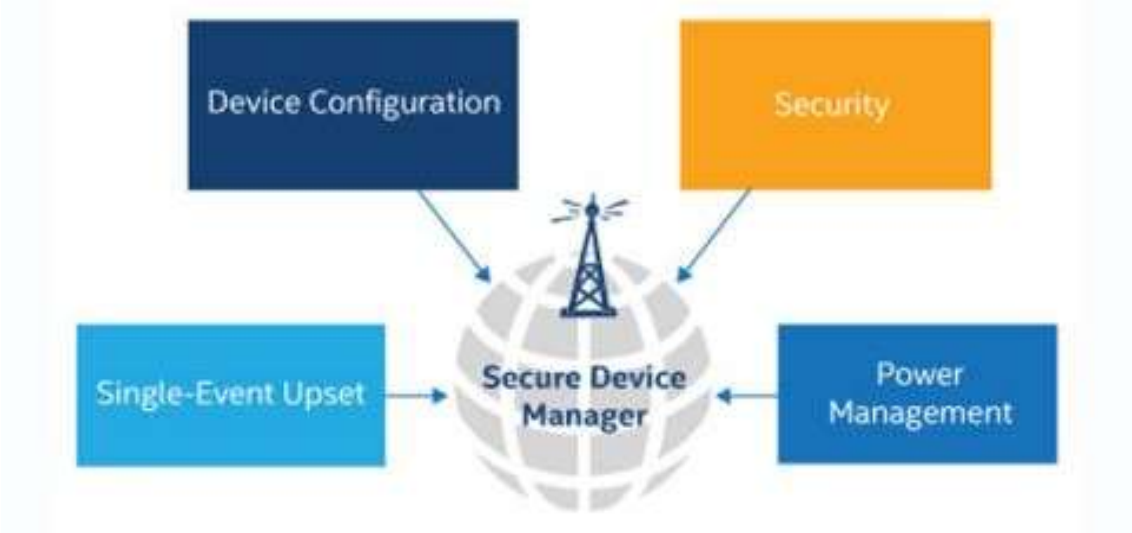
Attacks to vehicle already happened

Man in the Middle attack to exploit the V2X communication between a RSU (Road Side Unit) and the 0BU (0n-Board Unit) to move the car to another position vs. the desired one and e.g. stole the car



R&D open challenge

- Integrating in the device manager features for secure device configuration, secure device update, secure power management, **anomaly detection** (fault and/or intrusion detection), encrypted data exchange
- Similar to faults (e.g. due to ageing, Single Event Effects, harsh conditions) cyberattacks creates anomalies, that however **are more difficult to detect & correct**
- Due to increased computational capabilities of cloud centers and the emerging of quantum computers, many current crypto algorithms loose their security and new **quantum-attack resistant techniques** must be found



We are still in an R&D phase (research and development)

Of course we are discussing technical part, but there are different shields to consider: awareness, training, supply chain security...

Needs of embedded security for edge devices

- Security by design in automotive and IoT markets needs:

Advanced cryptographic techniques for end-to-end security

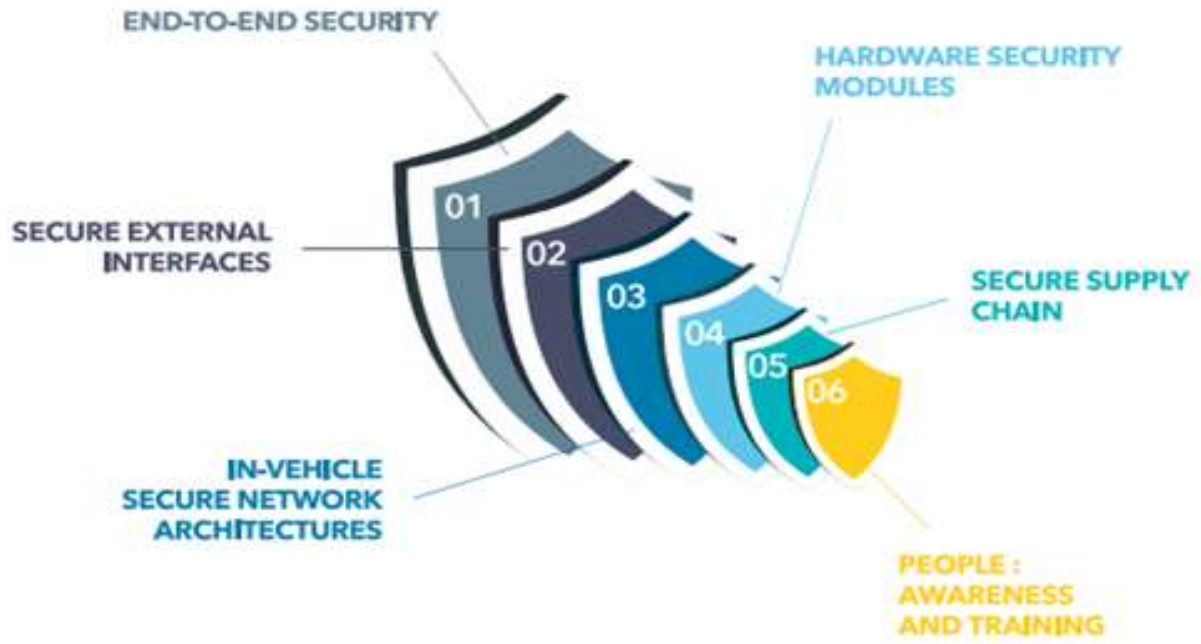
(data encryption/decryption, signature&verification, secure key generation and distribution)

Secure external interfaces and secure on-board networks

Availability of Hardware Security Module (HSM) for high energy-efficient cryptography

Application of security concept to the supply chain

Awareness and training of people/users



SECURITY BY DESIGN TOPICS IN AUTOMOTIVE

Outline

- Safety&Security correlation and needs of embedded security in vehicles
- Analysis of HW secure specifications for ECUs:

SHE (Secure HW extension),

EVITA (E-safety vehicle intrusion protected applications) and IS0 21434

- Architectures and components of HSM in automotive:

Analysis of HSM in commercial automotive MCUs (TPM, ARM, Infineon)

Automotive security specifications

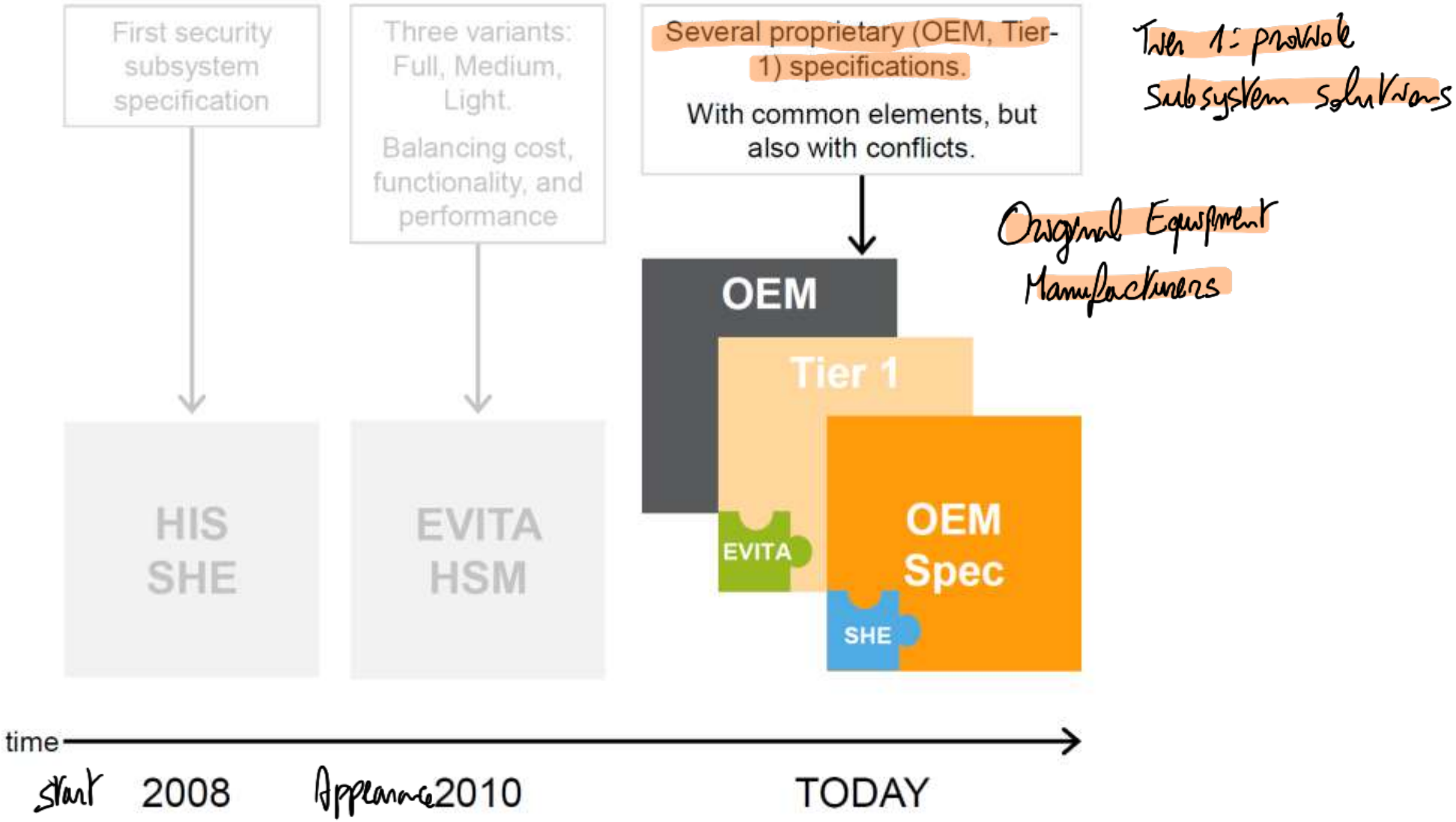
- The SHE (Secure Hardware Extensions) specification set the foundation, introducing the concept of a configurable (automotive) security subsystem

https://www.autosar.org/fileadmin/user_upload/standards/foundation/19-11/AUTOSAR_TR_SecureHardwareExtensions.pdf

- EVITA's HSM specification extended this concept into a programmable subsystem, in three flavors (Full, Medium, and Light), addressing a broader range of use cases
- Nowadays, OEMs are creating their own technical specifications, including selected aspects of SHE, EVITA, and FIPS (Federal Information Processing Standards, US) 140-2 to assess new security regulations

There is not a common standard yet.

Automotive security specifications



Automotive security specifications coverage in Commercial Off The Shelf products (example NXP)

	SHE	EVITA (Light / Medium / Full)	More recent needs
ARCHITECTURE	• Configurable, fixed function	• Programmable (except EVITA Light)	• Acceleration close to the interfaces (CAN and ETH MAC/PHYs) • Support for Flash-less technologies
FUNCTIONALITY	• Secure boot • Memory update protocol • AES-128 (ECB, CBC) • CMAC, AES-MP • TRNG, PRNG • Key derivation (fixed algorithm) • 10+4 keys, key-usage flags	Same as SHE, plus: • AES-PRNG • monotonic counters (16x, 64bit) Plus, for EVITA Medium and Full: • WHIRLPOOL, HMAC-SHA1, ECDH and ECDSA (P256)	• Further crypto algorithms (e.g. RSA, SHA1-3, Curve25519, …) • Rollback protection • Key negotiation protocols • Communication protocol offloading (e.g. TLS, IPsec, MACsec, …) • Context separation / multi-application scenarios • Increased attack resistance (e.g. SCA, Fault Injection, …)
OTHER			
Covered by:	NXP CSE family (since 2010)		
	NXP HSM family (since 2015)		
	NXP HSE family (since 2019)		

Security at the edge of communication

Programmable Processor

Move them as near for communic.

(for message auth)

(on chip)

specific curves

hash

elliptic curve cryptography

(rollback to old versions*)

support for TLS in HW

Consideration against side channel attacks

* Rollback when we recognise someone can't speak our language (TLS 2.0)

Automotive security specifications coverage in Commercial Off The Shelf products (Notes)

CSE (Crypto Security Engine), HSM (HW Security Module), HSE (Hardware Security Extension) are NXP commercial names for different types of HSM Hardware security module

Protocol offloading in new secure automotive MCUs thanks to HSM technology Entire protocols on hardware

Transport Layer Security (TLS), the successor of the now-deprecated Secure Sockets Layer (SSL), is a cryptographic protocol designed to provide communications security over a computer network. The protocol is widely used in applications such as email, instant messaging, and voice over IP, securing HTTPS remains the most publicly visible.

IEEE 802.1AE (MACsec) is a network security standard that operates at the medium access control layer (between network and data link) and defines connectionless data confidentiality and integrity for media access independent protocols. MACsec frame format is similar to the Ethernet frame, but includes additional security fields (e.g. Security Tag, Message authentication code,…)

Internet Protocol Security (IPsec) is a secure network protocol suite that authenticates and encrypts the packets of data to provide secure encrypted communication between two computers over an Internet Protocol network. It is used in virtual private networks (VPNs).

You find one of those usually.

Automotive security specifications coverage in Commercial 0ff The Shelf products (Notes)

Increase complexity and co-existence of many standards and both new and legacy applications (if a car model is in production for 10 years and the owner will use it up to 10 years the technology cycle is 20 years) may be a source of threats

Rollback attack, is a cryptographic attack (downgrade attack) on a computing system or communication protocol that makes it abandon a high-quality mode of operation (e.g. an encrypted connection) in favor of an older, lower-quality mode of operation (e.g. cleartext) that is provided for backward compatibility with older systems.

An example of such a flaw was found in OpenSSL (Open SW version of the Secure Sockets Layer) that allowed the attacker to negotiate the use of a lower version of **TLS (Transport Layer Security)** between the client and server.

Opportunistic encryption protocols such as STARTTLS are vulnerable to downgrade attacks, as they, by design, fall back to unencrypted communication. Websites which rely on redirects from unencrypted HTTP to encrypted HTTPS can also be vulnerable to downgrade attacks (e.g., sslstrip), as the initial redirect is not protected by encryption.

Secure Hardware Extension (SHE)

- The Secure Hardware Extension (SHE) is an on-chip extension to any given microcontroller.
- SHE is intended to move the control over cryptographic keys from the software domain into the hardware domain and therefore protect those keys from software attacks.
- SHE it is not meant to replace highly secure solutions like TPM chips (see later slide) or smart cards, i.e. no tamper resistance is required by the specification.

The main goals for the SHE design are:

- Protect cryptographic keys from software attacks
- Provide an authentic software environment
- Let the security depend on the strength of the underlying algorithm and the confidentiality of the keys
- Keep the flexibility high and the costs low
- Basically SHE consists of three building blocks, a storage area to keep the cryptographic keys and additional corresponding information, a implementation of a block cipher (AES) and a control logic connecting the parts to the CPU of the microcontroller.
- SHE can be implemented in several ways, e.g. a finite state machine or a small dedicated CPU core.

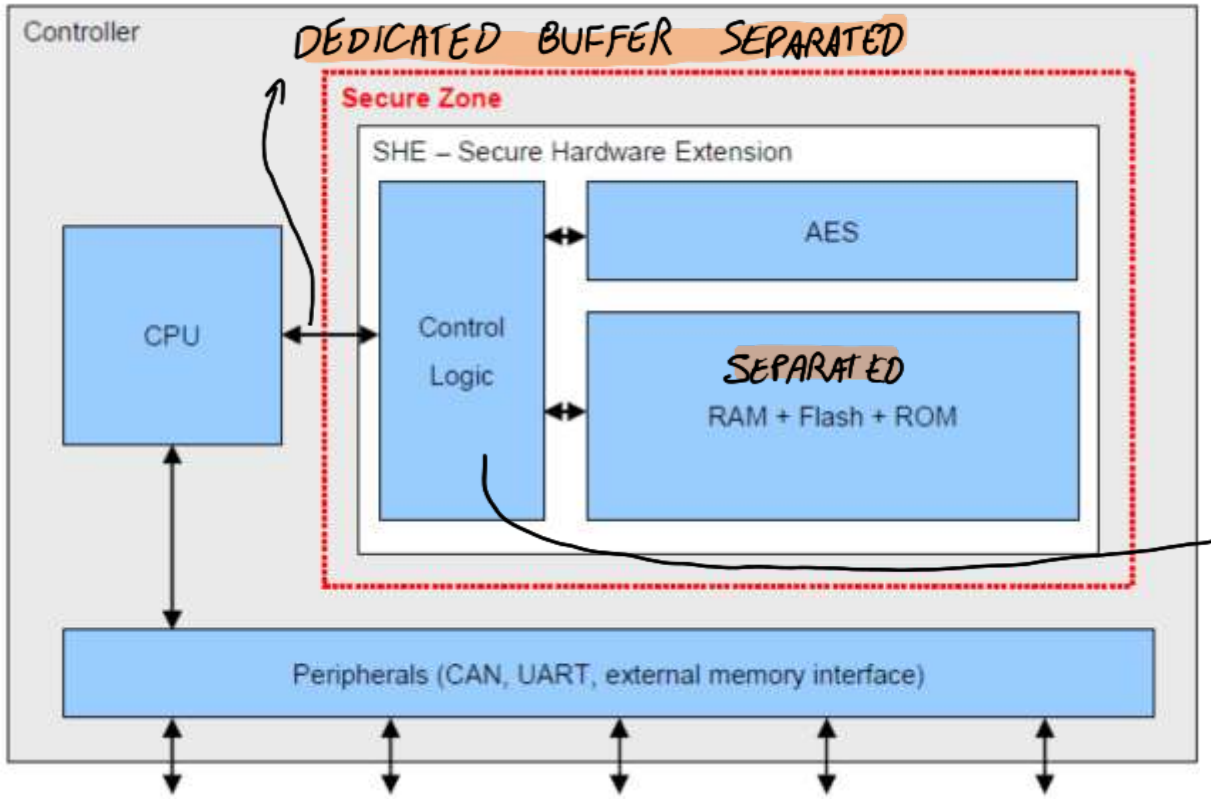
↳ separate

You can see the SHE as another microcontroller peripheral

SHE is an on chip extension of a normal microcontroller. Goal was moving control over cryptographic keys from SW domain to HW domain, protecting them from SW attackers. SHE was not designed to replace TPM. Inside the microcontroller you should add features to ensure security. You still use TPM external chip for protection.

Secure Hardware Extension (SHE)

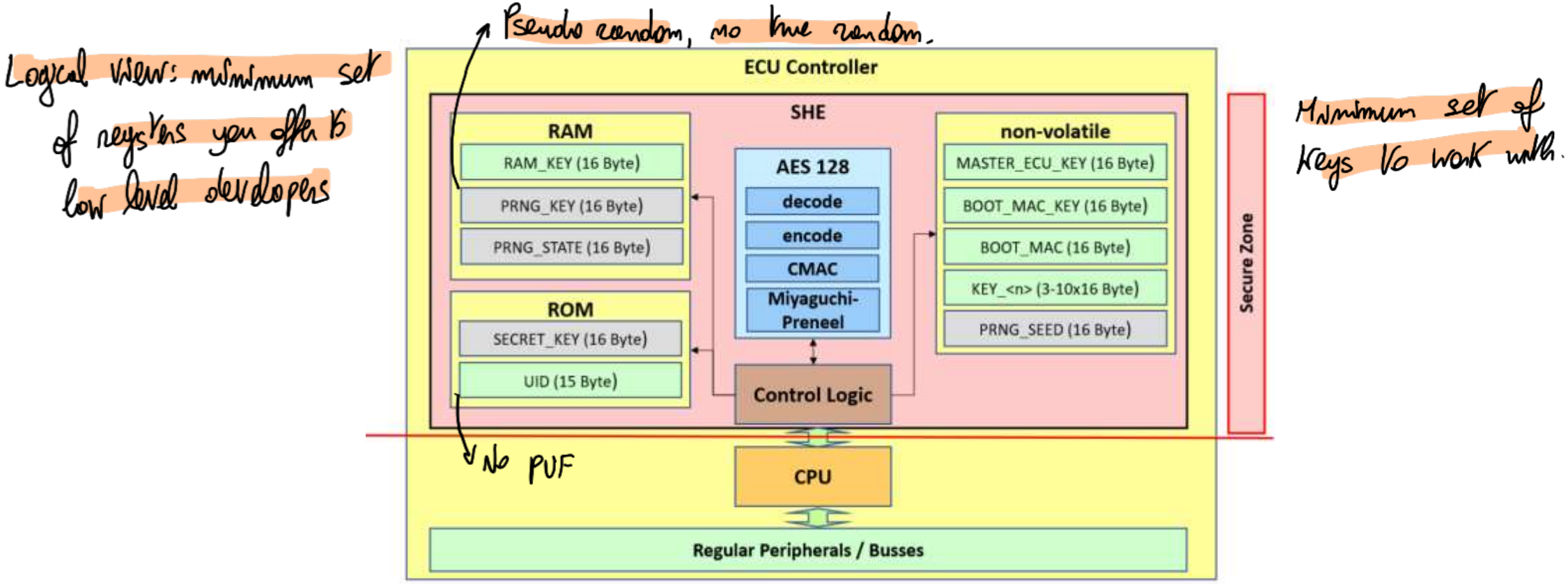
SHE implemented the concept of SECURE ZONE for the first time (inside microcontroller)



To guarantee security of this connection

SHE can be connected to the CPU in several ways, e.g. through a dedicated interface or an internal peripheral bus. The interconnection must be implemented in a way that no other peripheral or an external entity can modify the data transferred between the CPU and SHE.

Secure Hardware Extension (SHE)



Logic view of a SHE-compliant HSM in Autosar

Secure Hardware Extension (SHE)

All cryptographic operations of SHE are processed by an AES-128.
The latency of the AES must remain <2 us per encryption/decryption of a single block, including the key schedule
↳ specification to be SHE compliant. 2µs max for 1 block
Minimum ECB and CBC modes of AES supported
CMAC supported for Message Authentication Code (verification/generation)

The performance of the AES must be high enough to allow for a secure boot of 5% of the flash memory, but 32kByte at minimum and 128kByte at maximum, of the microcontroller in <10ms.
↳ Allow verification of memory within 10ms
↳ If you need more, SHE is not enough

SHE needs memory to store secure keys and MACs.
A non-volatile memory is required to store information that needs to be available after power cycles and resets of the microcontroller.
A volatile memory is required to store temporary information. The volatile memory may loose its contents on reset or power cycles. The memory of SHE should only be accessible by the SHE control logic.

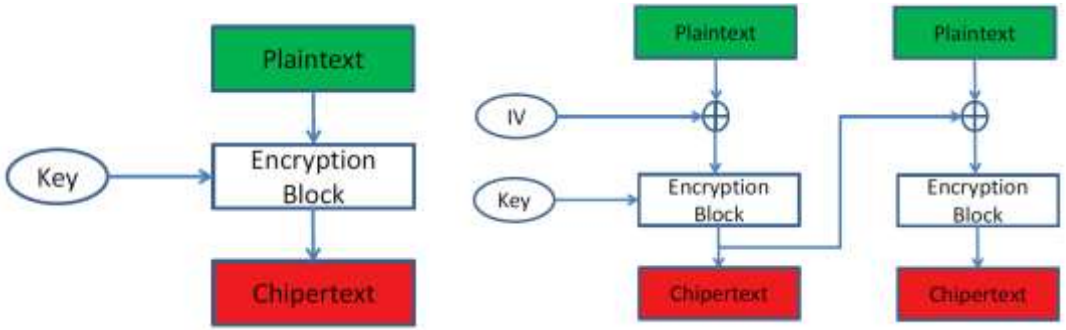
At least 100 successful write-cycles to the non-volatile memory must be guaranteed per memory slot by the implementation, more write cycles must be possible
↳ Non volatile memory has problem of endurance : so no OTP memory

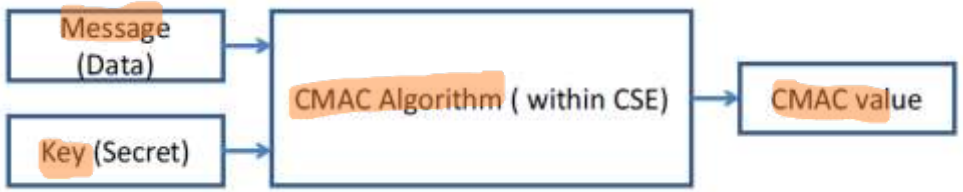
Why multiple modes to be supported in HW?

Example: working principle of AES modes

ECB (Electronic Code Book) and **CBC** (Cipher Block Chaining) for encryption, both for confidentiality but with different trade-off between security level and computing time/power consumption

CMAC (Cipher-based Message Authentication Code) for authentication/integrity rather than confidentiality

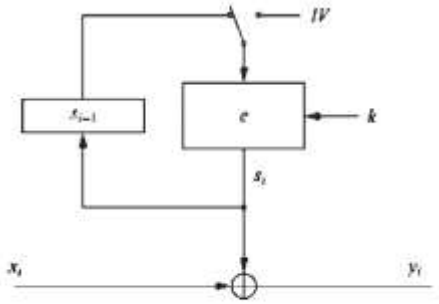




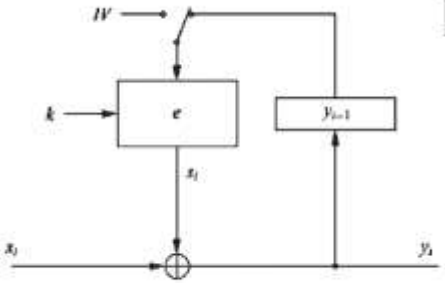
Why multiple modes?

Example: working principle of AES modes

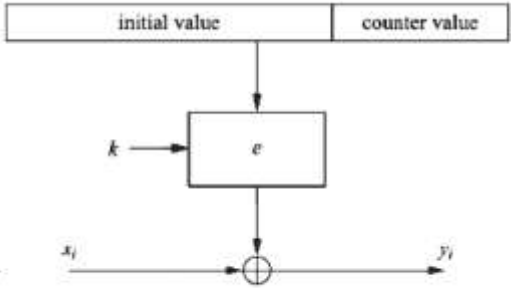
OFB (Output FeedBack)
CFB (Cipher FeedBack)
CTR (CounTeR)
GCM (Galois Counter mode)



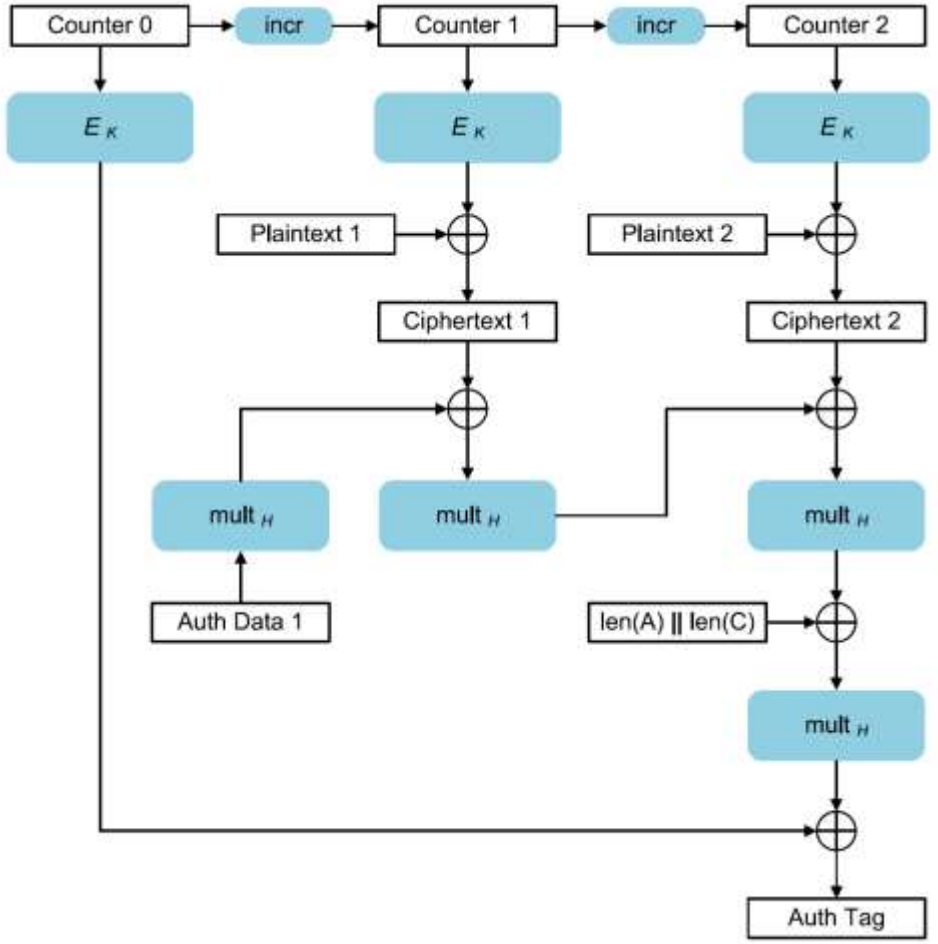
OFB Mode



CFB Mode



CTR Mode



Why multiple modes?

Security services in AES SHA and ECC modes

Cryptographic primitive	Cybersecurity service		
	Confidentiality	Integrity	Authentication of integrity (MAC)
AES-ECB	✓	×	×
AES-CBC	✓	×	×
AES-CFB	✓	×	×
AES-OFB	✓	×	×
AES-CTR	✓	×	×
AES-CMAC	×	✓	✓
AES-CCM	✓	✓	✓
AES-GCM	✓	✓	✓
AES-XTS	✓	×	×

Cryptographic primitive	Cybersecurity services	
	Integrity	Authentication of integrity (MAC)
SHA2-224	✓	×
SHA2-256	✓	×
SHA2-384	✓	×
SHA2-512	✓	×
SHA-3-224	✓	×
SHA-3-256	✓	×
SHA-3-384	✓	×
SHA-3-512	✓	×
Hardware assistance for HMAC	✓	✓

Cryptographic primitive		Cybersecurity service					
		Confidentiality	Integrity	Authentication of integrity (MAC)	Authentication of source (digital signature)	Non-repudiation	Support services
Key pair generation		×	×	×	×	×	Key generation
Public key generation		×	×	×	×	×	
ECDSA		×	✓	✓	✓	✓	×
Hardware assistance	ECIES	✓	✓	×	×	×	×
	ECDH	✓	✓	×	✓	✓	Key exchange
	ECMQV	✓	✓	×	✓	✓	

↳ Policy implemented in HW: VERY INTERESTING

Key policy in the Secure Hardware Extension (SHE)

↳ Concept of ownership

The **MASTER_ECU_KEY** is for the “owner" of the component using SHE and it can be used to reset SHE or change any of the other keys. The MASTER_ECU_KEY is only used for updating other memory slots inside of SHE [Like lord of the rings]

The **BOOT_MAC_KEY** is used by the secure booting mechanism to verify the authenticity of the software

The **BOOT_MAC** is used to store the MAC of the Bootloader of the secure booting mechanism and may only be accessible to the booting mechanism of SHE. ↳ 2 diff auth: SW for the boot but also auth. of the unit that loads boot program Double validation

KEY_ n can be used for arbitrary functions. n is a number 3..10, i.e. SHE must at least implement three and at maximum ten keys for arbitrary use.

PRNG_SEED is used to store the seed for pseudo random number generator as described. in worst-case scenarios it is written on every power cycle/reset. Seed should be provided externally at least each power on. SHE does not have a TRNG

The **RAM_KEY** can be used for arbitrary operations. The PRNG_KEY and PRNG_state are used by the pseudo random number generator

All 16 bytes (except UID)

ROM slots may be writable during production but not after leaving the fabrication.

ROM IS OTP

Key policy in Secure Hardware Extension (SHE)

ROM slots may be writable during production but not after leaving the fabrication.

SHE must contain a unique secret key SECRET_KEY that shall not only be derived from the serial number or any other publicly available information.

NOT something derived just from public info. SECRET_KEY and UID are two chip secrets.

The SECRET_KEY has to be inserted during chip fabrication by the semiconductor manufacturer and should not be stored outside of SHE. It can be generated by a certified physical random number generator, e.g. a Hardware Security Module (HSM).

Master key maybe is released to buyer (it can be changed)
With this you can control master key for 1st time

The SECRET_KEY may only be used to import/export keys.

The UID is specified to 120 bit because it is always used in conjunction with two key ids or the status register to form a 128 bit block.

↳ 8 bits from something else.

If the identification item is smaller than 120 bits it has to be padded with zero bits on the MSB side before feeding it into SHE.

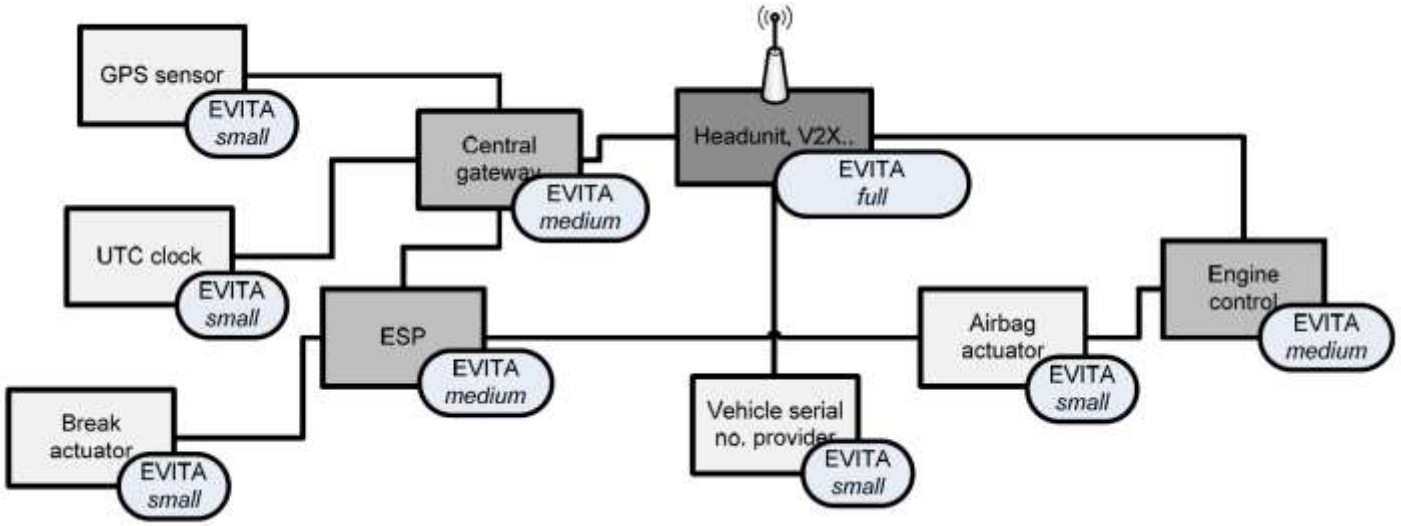
The UID has to be inserted during chip fabrication by the semiconductor manufacturer

If identification is done by combining UID with content of registers etc., this makes cloning harder.
15 bytes are written by manufacturer, but you have the additional byte secure.

Today in HW design, SHE key policy is considered to be the minimum stanking power

EVITA

Depending on the application minimal or full features can be deployed
(or disabled in the full version to save power)
→example to automotive from EVITA (E-safety Vehicle Intrusion proTected Applications) involving industrial partners like Infineon, Bosch, Continental, BMW, Escrypt, Fujitsu,… https://www.evita-project.org/



the "EVITA HSM Full Version" as hardware extension to the ECU specifically responsible for V2X applications,

the "EVITA HSM Medium Version" as hardware extension to the ECU connected to the in-vehicle domain controls (e.g., power train control) and,

the "EVITA HSM Light Version" for security-critical sensors and actuators.

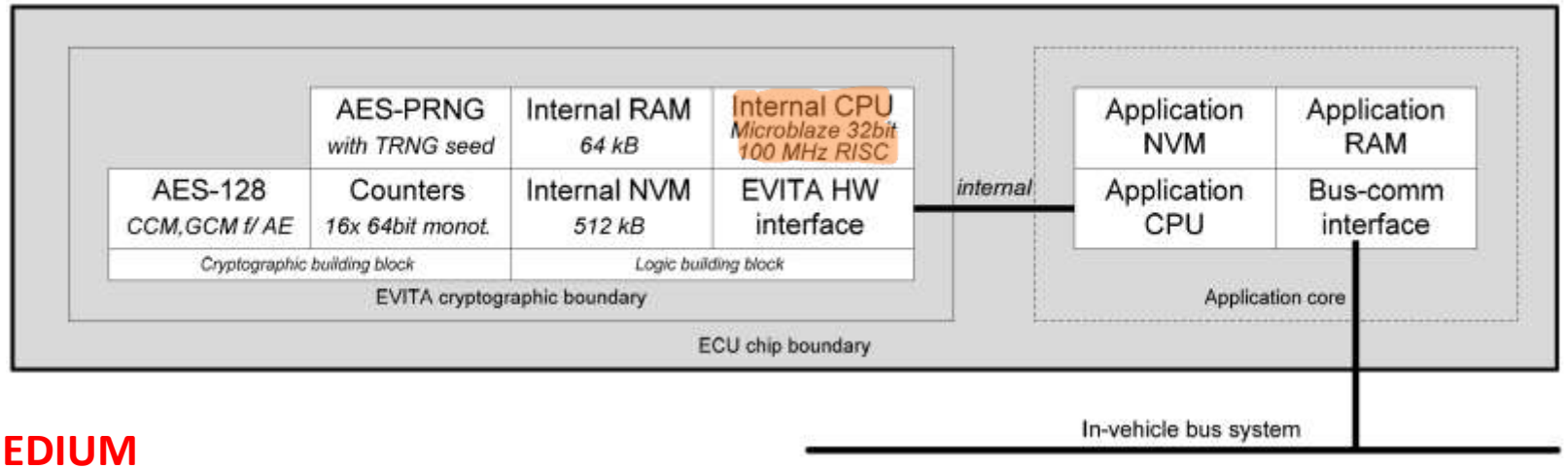
EVITA is a consortium. It lacks standard (not standardised), but a group of partners that developed it and became adopted by a ton of providers.

What are the concepts? EVITA has a heterogeneous level of security.

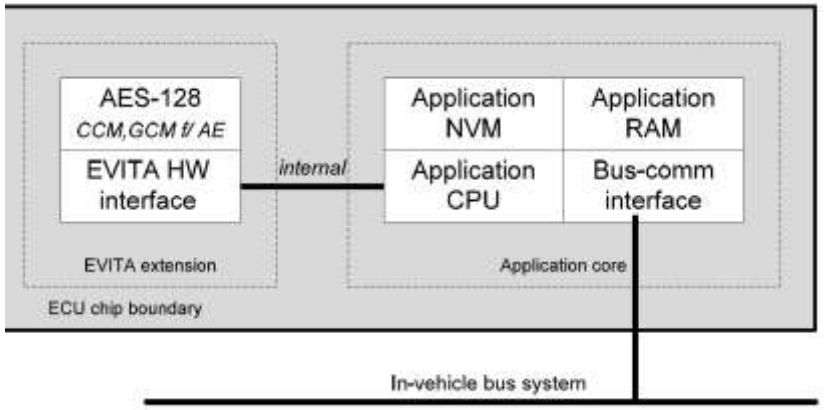
- Different components in our vehicles need different level of security.
  Okay, some are more problematic, but we should consider general security for other components too.
  Evita defined different levels of security requirements in which components can fall into. Component element A should fall under level 1. B under level 2. Etc.
- One single level of security is not enough: we have different probabilities for attack.

Δ Behind mapping where is vulnerability and threat analysis.

EVITA Medium & Small



EVITA MEDIUM



EVITA SMALL

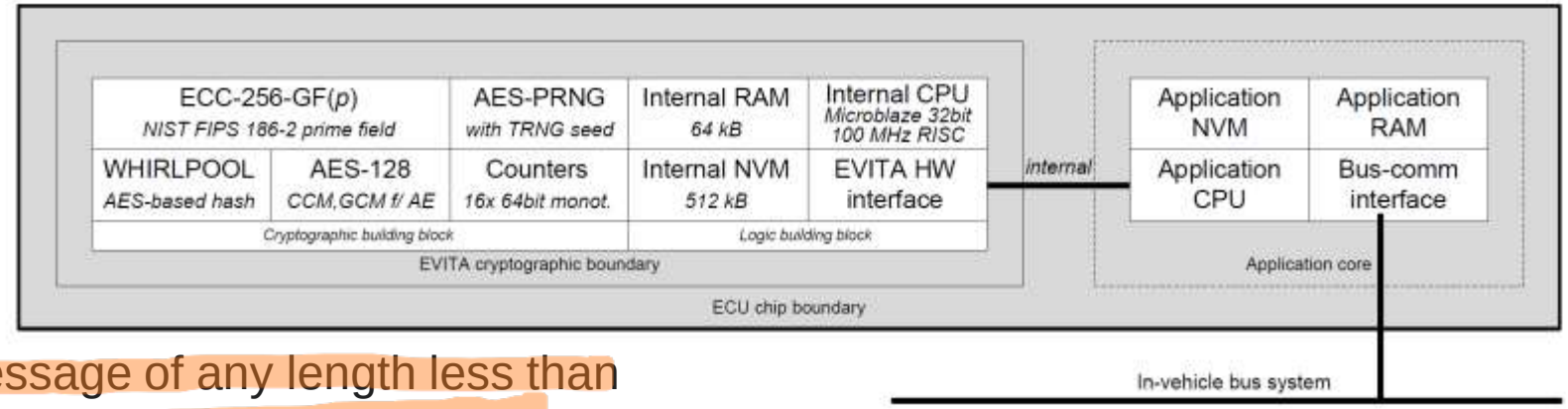
(AES like in SHE but with modes ensuring confidentiality, authentication and integrity check)

Small vs like SHE: AES supports more advanced modes, more powerful accelerating engines.

Medium: the evita cryptographic environment becomes another processor, plus 64 KB of RAM and 512 KB of non volatile memory; dedicated counters (AES), AES module and a TRNG (seed) and a PRNG based on AES.

EVITA Full

EVITA FULL



Whirlpool takes a message of any length less than 2^{256} bits and returns a 512-bit message digest

Building block	FPGA size (slices) estimation	Performance estimation
ECC-256-GF(p)	2,000	200 sig/s
WHIRLPOOL	3,000	1 Gbit/s
AES-128	1,000	1 Gbit/s
PRNG	200*)	1 Gbit/s
COUNTER	100	16x 64bit
CPU	2,000	32bit-RISC, 100MHz, 100 DMIPS
RAM	N/A (use block RAM)	64 kB
NVM	N/A (external)	512 kB
CONTROL / IF	1,000 – 2,000	N/A
Total	~ 10,000	

Faster performances

EVITA FULL COMPLEXITY

Full: same memory, universal processor, but also asymmetric cryptography supported (ECC) plus Whirlpool (AES based hash).

Why is it public? So using this, a company knows that BMW wants it, so they make solution for them.

NOTE: these are low level standards and bottom up approaches starting from manufacturer.

What's Next: UN R155 and ISO/IEC 21434

- From July 2022 onward, vehicle manufacturers must comply with the **R155 automotive cybersecurity** regulation (respect of security best practices to have a "secure by design" vehicle) for new vehicle type launches in Europe, Japan and Korea. They adopt these regulations that impose security by design practices for design and
- UN R155 covers aspects from the design to the production to the operation to the maintenance and throughout assistance to the decomissioning of vehicles. life cycle
- UNR155 secure by design is linked to process-related aspects (Cyber Security Management System or CSMS) and product-related aspects
- The standard **ISO/SAE 21434** (cybersecurity engineering for road vehicles) released in 2021 is deemed very supportive in implementing the requirements on the Cyber Security Management System (CSMS), as specified in UN R155, in organizations along the supply chain.
- ISO/SAE 21434 complete on the secure side what IS0 26262 does on the functional safety side.
- ISO/SAE 21434 covers security aspects from the design to the decomissioning of components.
- The relation between customers and technology (Tier-1 vs Tier2, 0EM vs Tier-1) providers is covered in ISO/SAE 21434

Companies should have a CSMS. Those are very high level regulations.
A company should follow a procedure, they should have a CS manager and a response team etc. You decide how to implement this.
This is a regulation so companies are forced to comply.

Problem: there is now a plethora of standards that can apply
ISO 21434 can be adopted for the CSMS to reach compliancy.

ISO 26262 is a standard for fault management of vehicles. The equivalent for security is 21434.
This is optimised for vehicle industry.
The standard also shows how CS responsibilities are distributed among the supply chain.

What's Next: UN R155 and ISO/IEC 21434

- ISO/SAE 21434 like the ISO/IEC 18045:2008 includes the definition of rating of fesibility of an attack and the potentiality of attack
- ISO/SAE 21434 like SAE J3061l includes the continuum process for cyber security management and the support of a cybersecuity culture
- ISO/SAE 21434 like ISO 26262 define rules for sharing information collected from vehicles and the need for an assessment of the quality of the protection processes used.
- ISO/SAE 21434 and UN R155 do not provide technologies, methods or solutions to be implemented to obtain safe components and compliance with the standard.
- This situation wants to keep the standards at a generic level and leave the manufacturer free to adopt the best solutions for each system. On the other hand, this lack of defined technologies and methods can create situations in which each company uses its own proprietary solution, creating possible conflicts in a highly connected environment.
- For example, the lack of defined limits for risk analyzes can allow having similar components from different manufacturers with different levels of cyber security, without however the customer having perception, because both are ISO / SAE 21434 and UNR155 compliant.

What's Next: NIS (Network& Information Security) 2

- European Directive NIS 2, defined in October 2023, imposes to the EU membering states to adopt within October 2024 laws about cybersecurity with at least should ensure:

- (a) policies on risk analysis and information system security;
- (b) incident handling;
- (c) business continuity, such as backup management and disaster recovery, and crisis management;
- (d) supply chain security, including security-related aspects concerning the relationships between each entity and its direct suppliers or service providers;
- (e) security in network and information systems acquisition, development and maintenance, including vulnerability handling and disclosure;
- (f) policies and procedures to assess the effectiveness of cybersecurity risk-management measures;
- (g) basic cyber hygiene practices and cybersecurity training;
- (h) policies and procedures regarding the use of cryptography and, where appropriate, encryption;
- (i) human resources security, access control policies and asset management;
- (j) the use of multi-factor authentication or continuous authentication solutions, secured voice, video and text communications and secured emergency communication systems within the entity, where appropriate.

NIS 2 is an European directive that asks member states within Oct. 2024 to introduce national laws based on security guidelines for org. in specific critical sectors. Law imposes the fact that you should take care of things, not how to do that.

Remember: SHE is kind of bottom up, while NIS 2 is very high level top down. Business continuity, yes, but no specification on how to do that.

Idea is before NIS 2, security was a concern that needed to start from the org. itself. Now, it is coming from outside.
Those are procedural standards.

NIS is going top down. EVITA bottom up. You can solve high level requirements with low level solutions offered by EVITA, SHE etc.

What's Next: NIS (Network& Information Security) 2

- NIS2 applies to:
- OES (operatore di servizi essenziali) such as water, energy, digital infrastructures; banking; Health; transports

DSP

- FSD (fornitore di servizi digitali) such as web browsing engines, Cloud computing, e-commerce

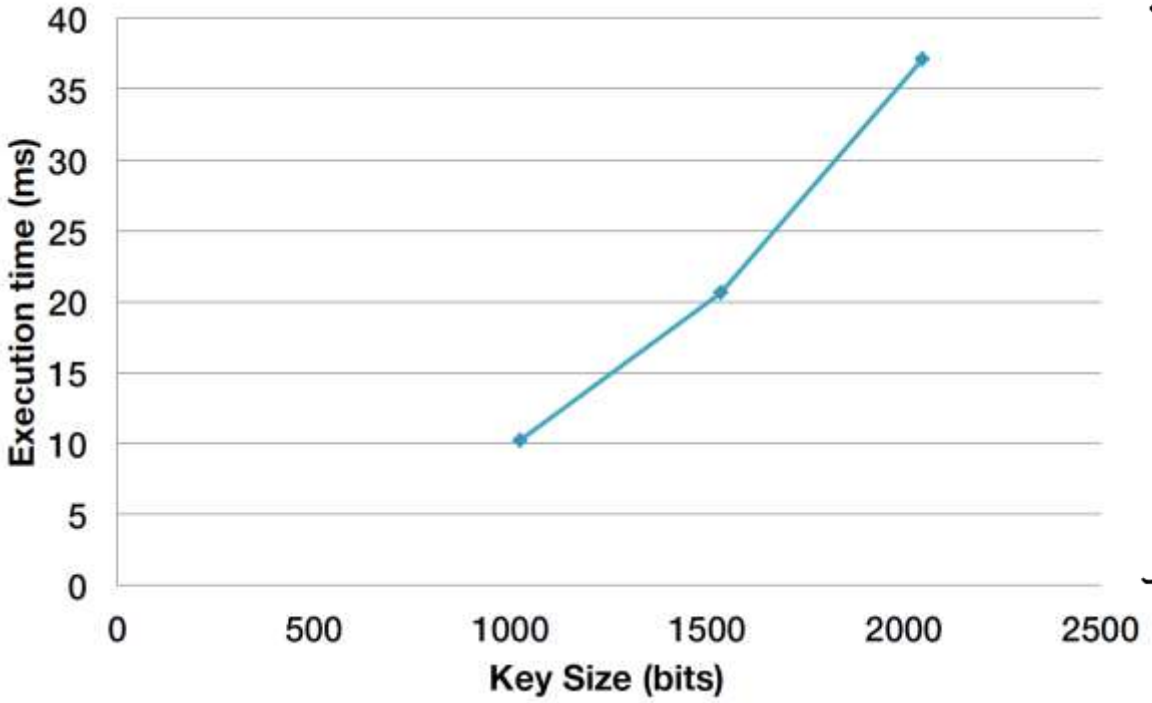
Outline

- Brief module & speaker intro
- Safety&Security correlation and needs of embedded security in vehicles
- Analysis of HW secure specifications for ECUs: SHE (Secure HW extension), EVITA (E-safety vehicle intrusion protected applications) and IS0 21434
- Architectures and components of HSM in automotive:

Analysis of HSM in commercial automotive MCUs (TPM, ARM, Infineon)

Is there an added value?

HW acceleration really needed? (1/3)



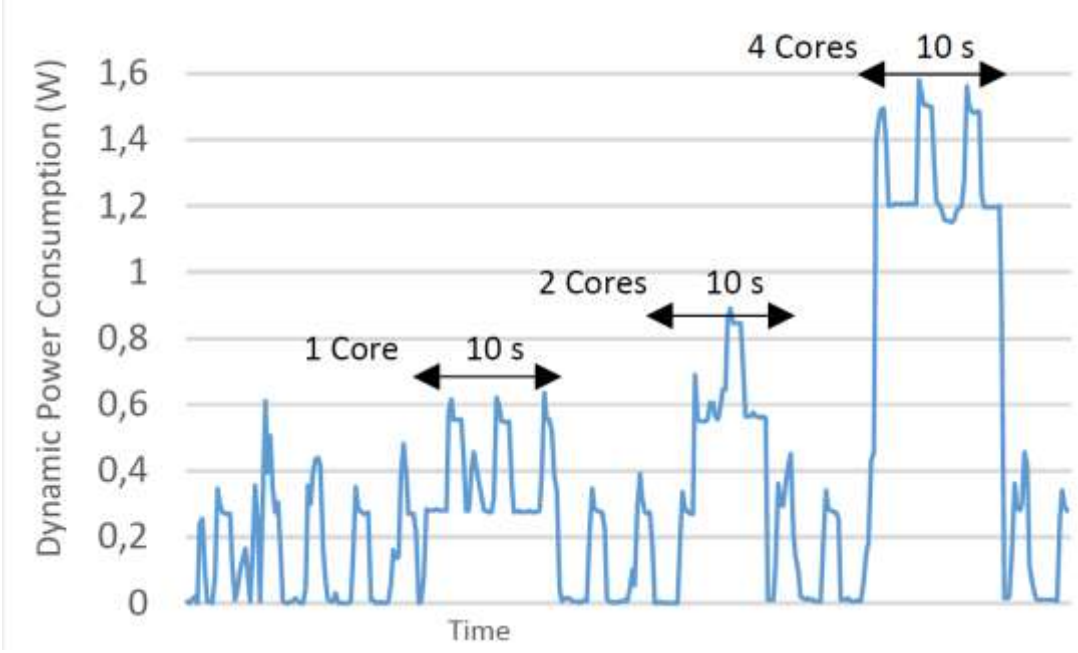
Nowadays minimum requirement is 2048 bits, and via software it takes 40 ms!

- Think of autonomous vehicles in which checking for a signature cannot take this long
- 50 ms might disable security in latency critical application

via software

Execution time of a RSA Public Key algorithm vs. key size on NXP processors without HW support to security.
Thanks to CSE the HW accelerated version takes max 100 us
(x2 order of magnitude gain in speed thanks to HW acceleration), source: NXP security white paper

HW acceleration really needed? (2/3)



Power consumption when executing via SW a basic crypto function on 4-core Cortex A53 processor (the one on Raspberry PI3)

↳ it takes a certain power. For a raspberry PI or limited systems might be too much

HW acceleration leads to an improvement of several order of magnitude in speed and energy→ key asset for connected vehicles

In addition, hw implementation makes writing SW easier. You don't need 100 basic operations, just one

HW acceleration really needed? (3/3)

		Figure of Merit (FoM)			
		Software implementation			Hardware accelerators
		1 core	2 cores	4 cores	
Security algorithm	AES-256	312.04	610.37	1130.21	204119.60
	SHA2-256	41.24	75.64	128.86	408293.46
	ECDSA-521	2.57	5.06	8.85	20131.61

The more complex the algorithm, the higher the gain

FOM (Figure of Merit)= Throughput/energy (the higher the better)

Throughput is encrypted bit/s or signature&verification-operations/s

Energy is energy-per-bit o energy-per-operation

FoM of OPEN SSL SW solution on 4-core Cortex A53 processor vs. HW acceleration on 45 nm standard-cells (source: test done at UNIPI, published in IEEE ICECS 2019)

HW acceleration really needed? (3/3)

Remember that in digital circuits Power (Watt)= Pstatic + Pdynamic

Pstatic is Ileak*Vdd where Ileak is proportional to the chip size (theoretically is 0, but static power is increasing)

Pdynamic is proportional to Fclk*Vdd2 (Transistors have thresholds for activations, so minimum voltage is not less than 1V) ①

By increasing the number of cores working in parallel the same throughput can be achieved with reduced clock frequency and moreover reduced Vdd*

↳ same throughput with less clock frequency.

This is why in previous slide the best solution is with 4 cores and not 1 or 2 cores.

* Same computation can distribute over multiple cores. So with multi core same throughput is divided for different cores, so you can achieve that with lower Vdd ① of course you cannot reduce too much because of this limit.

What can you enable with acceleration?

Need of over the air SW/FW update

- Premium vehicles have over 100M lines of code! (Windows 10 has 50M)
- 15% of vehicle recalls and 60% of warranty costs are firmware related
- Firmware updates require vehicle to be returned to the garage and this is Time-consuming and costly
- No guarantee customer will return it for recall
- Difficult to deliver new features to vehicle owners
- OEMs are missing post-purchase, revenue-generation opportunities
- Remote SW/FW update is a status symbol and part of the digital lfe style (premium cars)

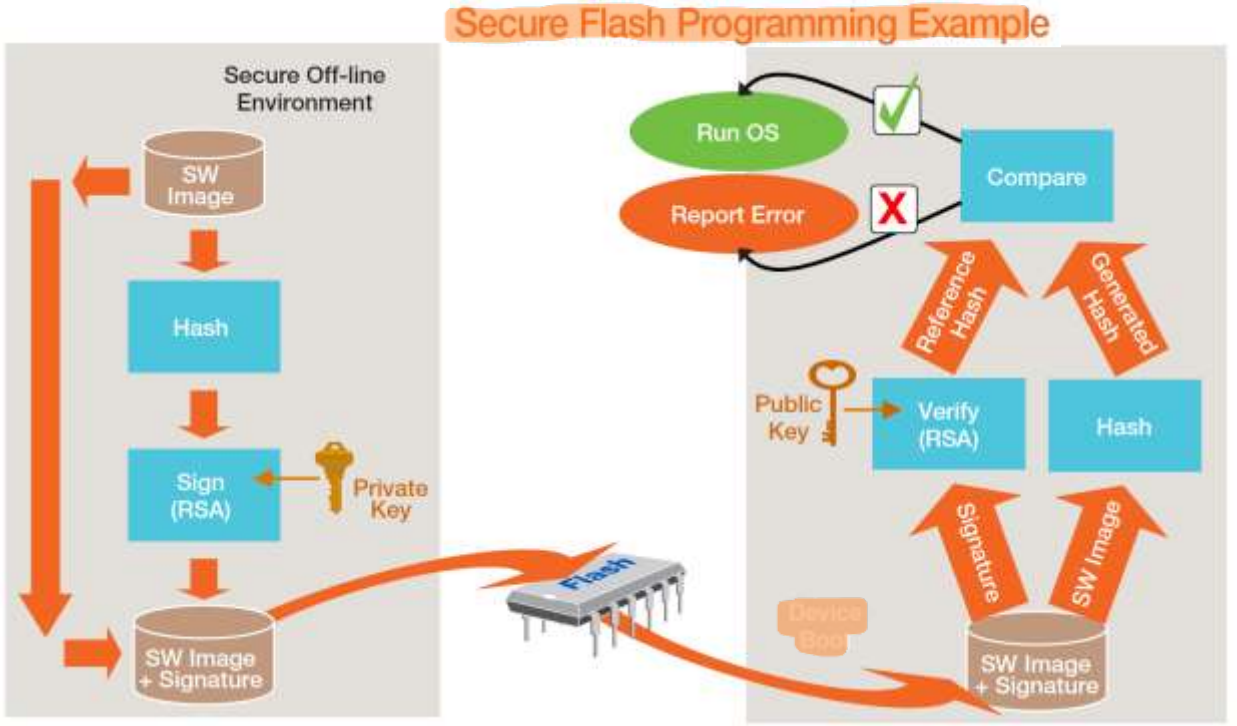
Example application to SOTA

SOTA/FOTA is the SW/FW over-the-air update
Already supported by TESLA
Key element in the Internet of Vehicle era (ADAS, V2X)

Algorithm 1 Software over-the-air update

1: Backend notifies the vehicle about available SW update.
2: A dedicated ECU inside the vehicle *verifies* the *authenticity* of the notification from the backend.
3: **if** verification is successful **then**
4: SW package is downloaded by the dedicated ECU.
5: **end if**
6: **if** SW packages is *authentic* and *integrity* is ensured **then**
7: Start flashing process via a *secure diagnosis* access protocol.
8: Target ECU checks the *integrity* of the flashed SW.
9: **if** flashed SW is correct **then**
10: Target ECU updates its *secure boot* reference value.
11: Reboot target ECU.
12: **end if**
13: **end if**

Secure SW/FW flashing of IoT/IoV MCUs (1/2)

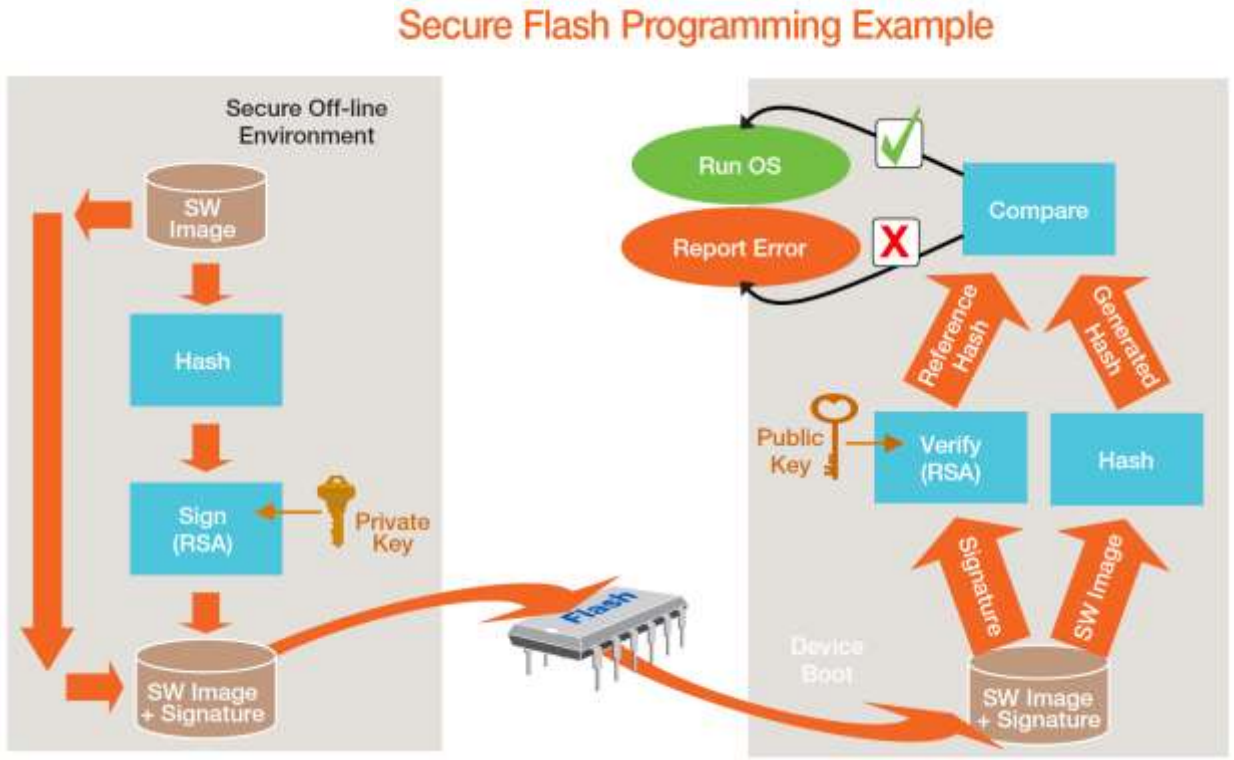


This is flashing on the memory system. In ECU you need implementation of services for signature verification, and this will be done also for over the air update.

The example uses an **RSA (ECC** may be an alternative) **private/public key pair** to sign and verify that the code is authentic and has not been modified by an attack during the programming process.
In the secure offline environment, a **hash** is made of the SW image, e.g. using **SHA2-256**. The hash value, which uniquely represents the SW image, is then signed with a private key that uniquely identifies the owner of the SW. The resulting signature plus SW image is then transmitted to the embedded memory system, which performs its own hash on the SW image.
The embedded system also authenticates the signature received from the offline environment using the public key associated with the private key that produced the signature.
The authentication procedure results in a hash value that must match the value from hashing the SW image.

You have to sign the SW image. So you can hash + RSA for Digital Signature. You send

Secure SW/FW flashing of IoT MCUs (2/2)



If they match, it means the SW image has not been modified, and it was received from the entity that used the corresponding private key. This confirms both that the SW is good and where it came from.

The signature could equally well be created and verified using **symmetric cryptography (e.g. AES128/256)** in which case the same key is used to create and verify the signature.

Unlike asymmetric keys, **the symmetric protocol relies on the embedded system and external entity both being able to secretly store the same key**. A successful attack on either the microcontroller or external entity will compromise the security of the system (i.e. **symmetric crypto has the problem vs. public-key cryptography of secure distribution of the single key**)

Threat in over the air SW/FW update

↑ Not only automotive OFC

- Allowing Over The Air updates on a Automotive ECU opens new ways of hacking the device
  - Protect communications and authenticate new data
  - Each step of the process must be secured and verified

- Establish a Chain of Trust

- To keep up against malicious attacks, Security must remain up to date
- Verify that the user has completed the update process check to defend people from themselves
- Verify that the new configurations of ECUs are fully coherent

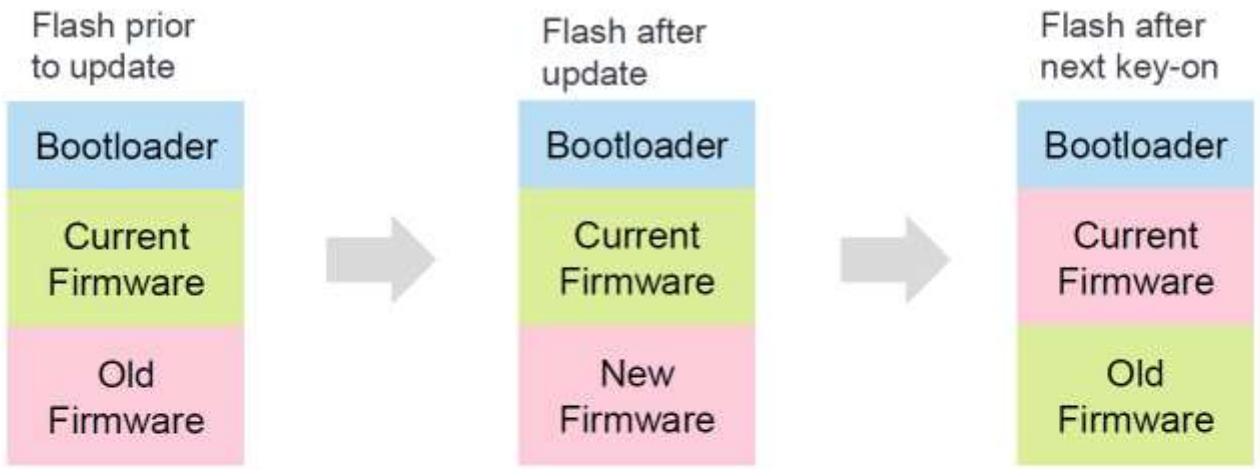
over the air

- SOTA/FOTA update is a security-dependent service that needs the support of an hardware security module.
- Hence the automotive HSM needs to support Digital signature, hash functions and (at least symmetric) encryption
- Low-complex/low-power HSM does not support dedicated public key encryption or key encapsulation (KEM) at ECU side→ New generations of HSM need the support of KEM schemes, i.e. to support key exchanges between ECUs during runtime

Procedure for over the air SW/FW update

Security: Supports CMAC authentication and AES-128 decryption

2 versions of firmware exist in internal flash.



Advantages

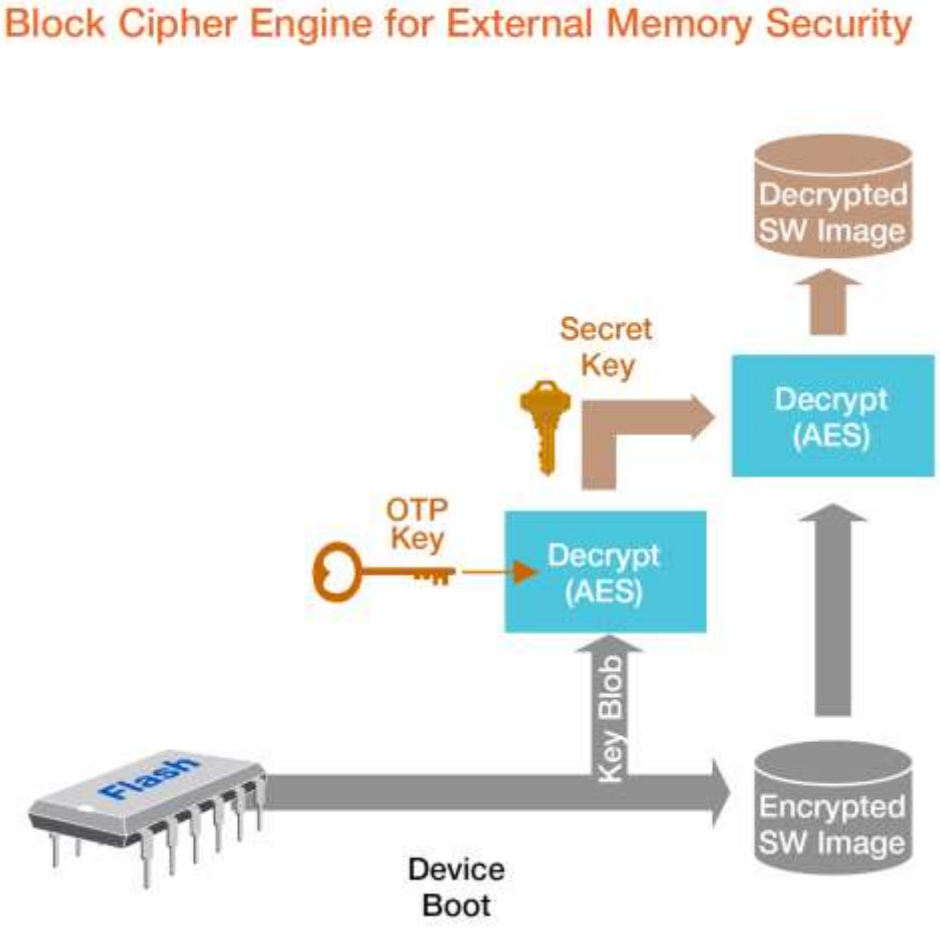
- Update can be carried out whilst application is actively running from flash
- Always have original firmware to roll back to in case of issue
- Vehicle always available – guaranteed no vehicle downtime regardless of update errors

Cost

- Requires 2x flash application storage
- Higher max current (run current in block A + erase/program current in block B)

EX: you double memory dedicated (especially in critical apps). You always want back up; you update, verify, if it is okay, then replace it no problem.

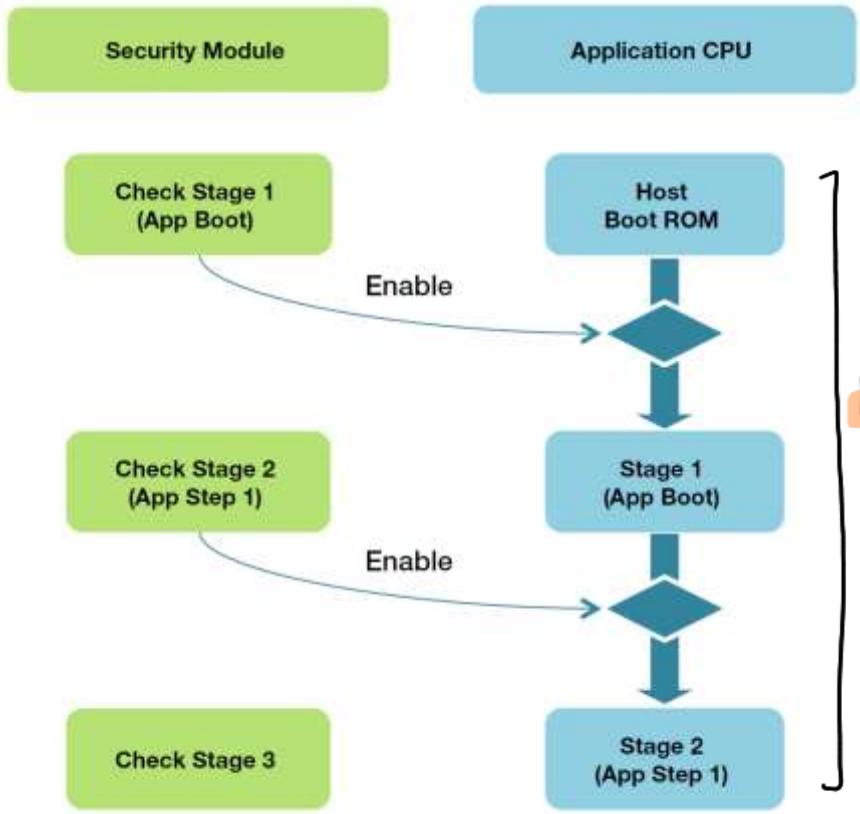
Secure data transfer from external memories



Some method of robust security must be provided for applications that expose the transfer of data and instructions on an external memory bus.
One way is to encrypt all content that is stored in external memory, and implement a cryptographic engine on the microprocessor, between the external bus interface and the CPU.
Code and data fetched from external memory must now be decrypted before being processed by the CPU.
Likewise, data stored back to external memory must first be encrypted to prevent successful snooping attacks on the external bus.
The main challenges in this architecture are how to prevent reduction in CPU performance and how to support random access fetches from external memory.
To this aim an AES block cipher operating in counter mode make it suitable for on-the-fly decryption without impacting performance.
Figure shows the general architecture of such a secure system.

Same concept applied to every access from an external memory and data transfer.

Chain of Trust for secure boot



Starting from a program burned inside your chip (so you can trust it) that is the root of trust.

FIRST PART OF CODE WAKES UP PROCESSOR AND ACCELERATORS AND SHORTLY YOU MOVE TO SECURE STEPS

Secure boot time could be shortened by using a chain of trust strategy:
Split up the memory space into multiple partitions, each of which is checked in sequence as the application SW progresses through its flow.
Figure shows an example of this strategy

Of course for secure boot you cannot work at SW level. You need acceleration that is mandatory.

Lot of systems in which Hardware Security and dependability are related

Soft errors (reliability) and Hardware Security

Hardware Secure Modules (HSM) have been also proposed as mitigation/countermeasure techniques for soft errors (i.e. Single Event Transients, Single Event Upset,..)

Due to the diffusion property of ciphers, there is an increase of the error effect of one bit at the input of a crypto function over many bits in the output.

For example, a single error in AES causing one bit flip causes over 50% of the ciphertext bits to be in error.

Moreover, since hash functions are used to identify the integrity of files, even a single bit flip in the data saved in a memory will cause a change of the hash function.

Hence, techniques like SHA-2 or AES-CMAC can be used for soft errors detection

In literature also combined encryption and error correction schemes have been proposed

C.H. Gebotys, Reliable Testable Secure Systems, Chapter 10 in Springer Security in Embedded Devices, pp 263-289, 2009

H. Cam et al., A COMBINED ENCRYPTION AND ERROR CORRECTION SCHEME:AES-TURBO, JOURNAL OF ELECTRICAL & ELECTRONICS ENGINEERING 2009

J- M. Mogollon et al., Real Time SEU Detection and Diagnosis for Safety or Mission-Critical ICs Using HASH Library-Based Fault Dictionaries, IEEE RADECS 2011

Hardware security in automotive ECUs

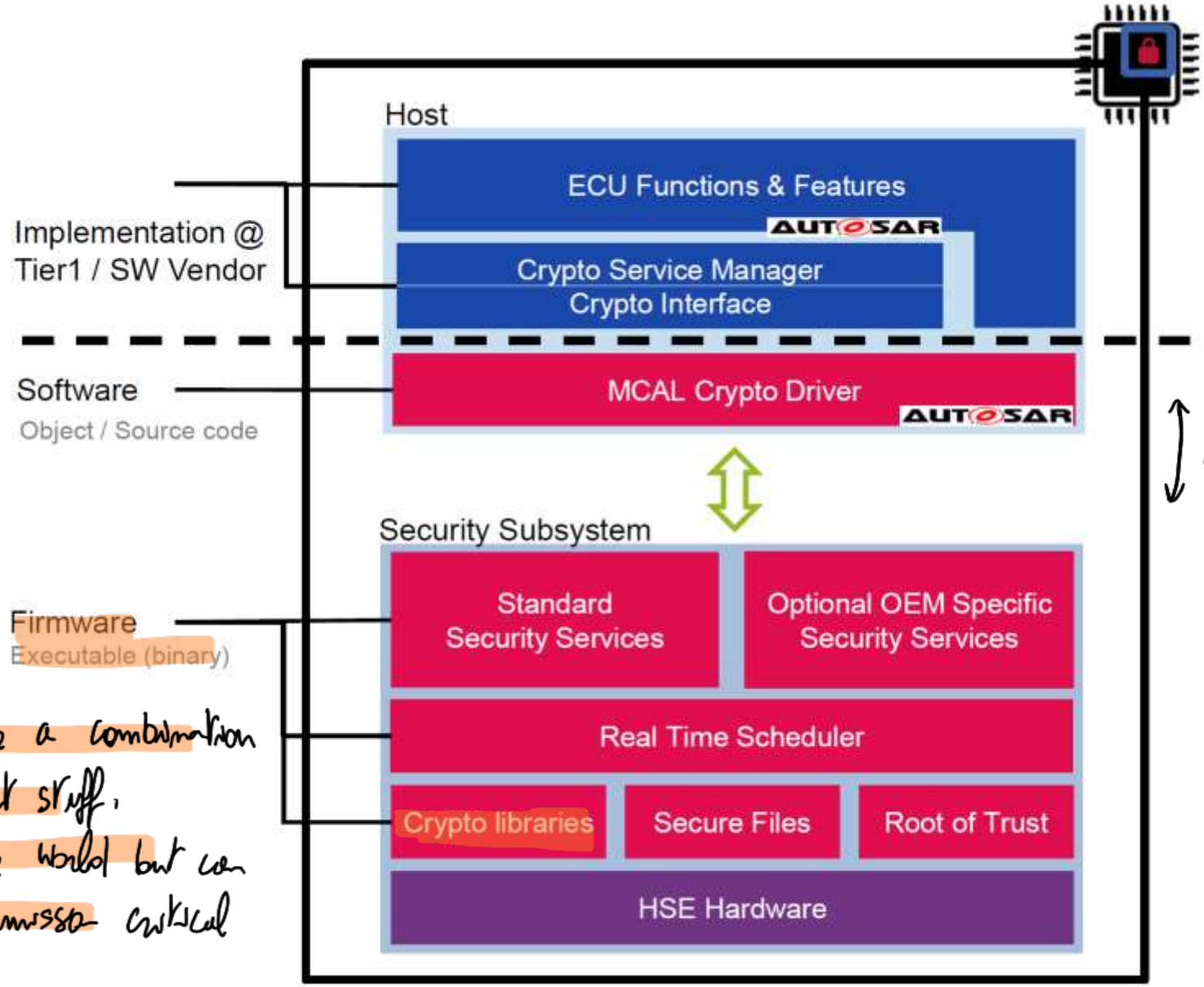
SW

You need hw module at your base (can be TPM or not) + SW that exploits those tools and of course security at SW level

FW → Crypto drivers make communication and usage of HSE possible by SW.

HW + related FW

Hardware security in automotive ECUs



interface at SW with abstraction layer of what we have at HW

FW can be a combination of different stuff,
Comes from automotive world but can be generalised to all mission critical applications

Alright Giovanni, in the world of automotive ECUs (Electronic Control Units), a **hardware security engine** is like a built-in security guard inside the chip—dedicated and fast.

It's a **specialized hardware block** within the ECU's microcontroller that handles cryptographic operations (like encryption, decryption, hashing, digital signatures) and secure key storage, all without having to rely on slower, more vulnerable software processes. Think of it like a mini fortress baked into the chip that makes sure stuff like firmware updates, secure communication between ECUs, or access controls happen in a safe, tamper-resistant way.

In automotive, where safety and reliability are everything, this engine helps protect against threats like code injection, spoofing, ECU cloning, or even someone trying to reverse-engineer how your car works. It's essential in systems that support things like **secure boot**, **firmware over-the-air (FOTA)** updates, and **vehicle-to-everything (V2X)** communication.

Ah, *HSE firmware*—that's the soul of the **Hardware Security Engine**.

While the HSE is the actual **hardware block** inside the microcontroller, the **HSE firmware** is the software that **runs on that block** to coordinate and execute all the security functions. It's like the brain telling the security muscle what to do.

In automotive microcontrollers (especially ones like NXP's S32 family), the HSE firmware handles stuff like:

- **Secure boot**
- **Key management** (generation, storage, distribution)
- **Cryptographic operations** (AES, RSA, ECC, etc.)
- **Secure lifecycle management** (changing ECU states like from development to production)
- **Flash encryption** and **secure debug access**

Trusted Platform Module (TPM)

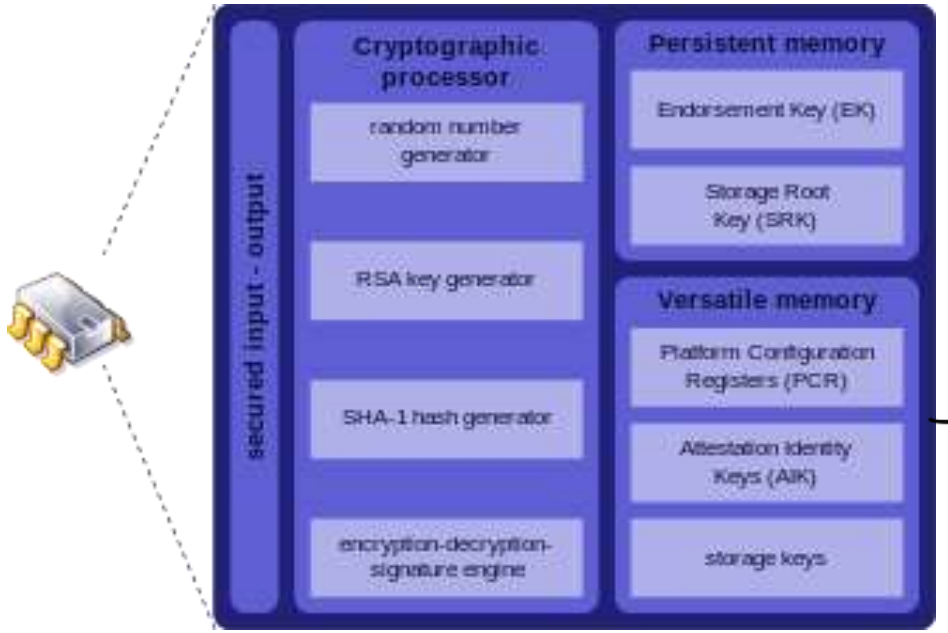
Set of basic HSM foreseen in **Trusted Platform Module** (TPM)

Published as ISO/IEC 11889 Parts 1-4 yr is now a standard

TPM supports secure keys for **authentication** and **encryption functions**

TPM implemented as an external peripheral with a communication bus to another microcontroller in the system (**Co-processor**) TPM is external chip with a bus to give services to processor, HSM is internal to the microcontroller (integrated IP)

One solution at low level to implement root of trust you can work with TPM.



→ SHA, RSA support, dedicated memory for key storage + other memory for certificates etc.

Trusted Platform Module (TPM)

There is something similar here to what we saw for SHE

TPM1.2

Specifies non-volatile memory, secret key storage, a random number generator (RNG)

RSA (RSA-2048 key): RSA is the Rivest, Shamir, Adleman algorithm for public key cryptography

SHA-1 (Secure Hash Algorithm) version1 is the basic for hashing (160 bit output tag) now considered breakbale in cryptoanalysis

HMAC (keyed-hash message authentication code)

alternative to AES

Vernam (based on polyalphabetic Vigenère substitution cipher) one-time pad algorithm is a basic for encryption

optional

In TPM1.2 use of AES (Advanced Encryption Standard) for symmetric cryptography is optional

Trusted Platform Module (TPM)

TPM1.2 used in personal computers, e.g. INTEL TEE (Trusted Execution Technology) chips and in Windows (server2016, server2019, Windows 10), and in industry 4.0/vehicles

https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/trusted-execution-technology-security-paper.pdf

https://www.infineon.com/dgdl/Infineon-data-sheet-SLB9670_1.2_Rev1.3-DS-v01_03-EN.pdf?fileId=5546d462689a790c016929e445ea4ff7

Versione TPM	Windows11	Windows10	Windows Server 2016	Windows Server2019
TPM 1.2		>= ver 1607	>= ver 1607	Sì
TPM 2.0	Sì	Sì	Sì	Sì

TPM2.0

Advanced secure features: it specifies also SHA-256 for hash, 128-b AES symmetric, ECC (Elliptic Curve Cryptography) using the Barreto-Naehrig 256-bit curve and NIST P-256 curve for public-key cryptography and asymmetric digital signature generation and verification

https://www.st.com/en/secure-mcus/st33gtpmai2c.html

Post quantum era TPM 2.0 is not enough (AES 256, No ECC...)

Physical specifications (vibration, temperature etc.)

TPM2.0 example ST33GTPMAI2C

AEC-Q100 "Failure Mechanism Based Stress Test Qualification For Integrated Circuits“ qualified (http://www.aecouncil.com/)

Compliant with Trusted Computing Group (TCG) Trusted Platform Module (TPM) Library speci. 2.0, Level 0, Revision 138

Fault-tolerant firmware loader that keeps the TPM fully functional when the loading process is interrupted (self-recovery)

Arm® SecurCore® SC300™ 32-bit RISC (Reduced Instruction Set Computer) core

protection for safe discharge

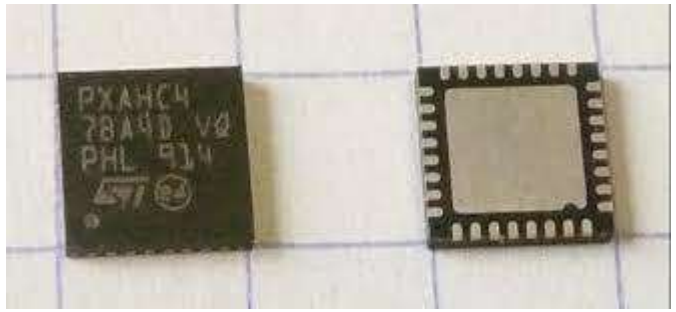
Automotive grade 2: −40 °C to 105 °C, ESD (Electro Static Discharge protection) up to 4 kV (Human Body Model), 1.8 V, 3.3 V or 5 V supply voltage range

based on human body

Hardware and software protection against fault injection

FIPS compliant RNG built on SP800-90A compliant SHA256 DRBG (Deterministic Random Bit Generator) and an AIS-31 Class PTG2 compliant true random number generator (TRNG)

RSA key generation (1024, 2048 bits), RSA signature, RSA encryption



SHA-1, SHA-2 (256 and 384 bits), SHA-3 (256 and 384 bits)

HMAC SHA-1, SHA-2 and SHA-3

AES-128, 192 and 256 bits

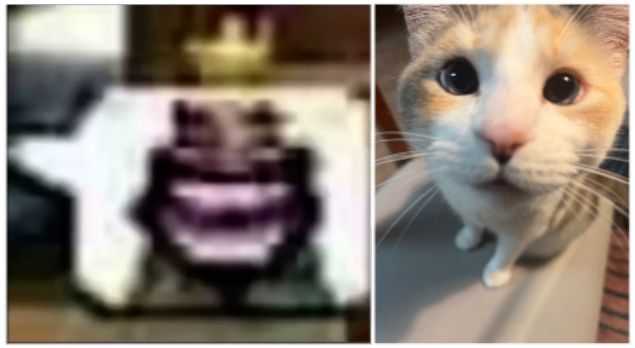
Triple DES (Data Encryption Algorithm) 192 bits (old applications)

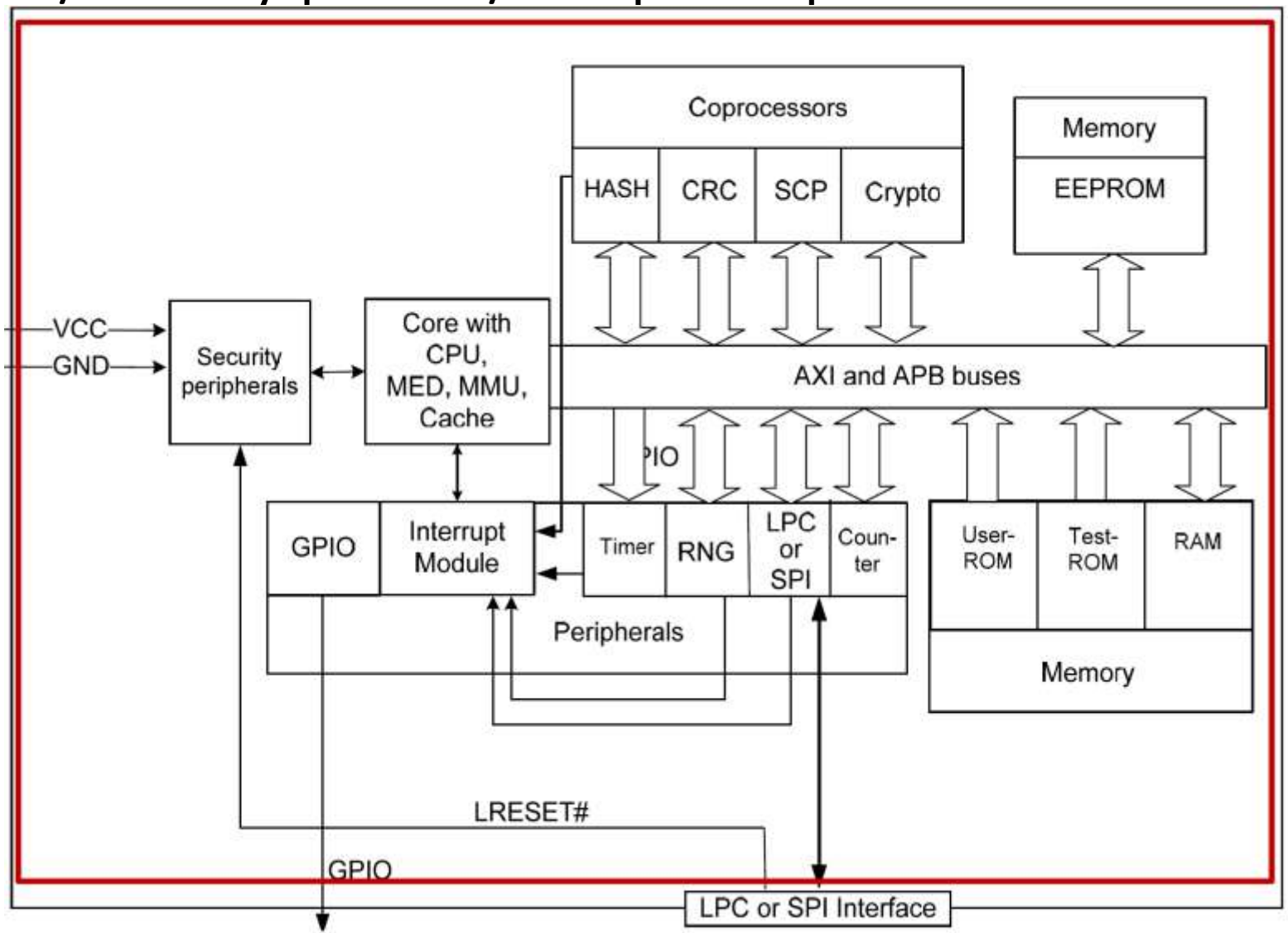
ECC (NIST P-256/384): Key generation, ECDH (Elliptic-curve Diffie–Hellman), ECDSA (Elliptic-curve Digital Signature Algorithm)

Device provided with 3 EK (Endorsement Key. This is an asymmetric key contained inside the TPM and injected at manufacturing time). The EK identifies the TPM. The EK cannot be changed or removed) and 3 EK certificates (RSA2048, ECC NIST P_256 and ECC NIST P_384): A TPM manufacturer-issued certificate for EKPub.

TPM2.0 example SLB 9660/9665/9670

- https://csrc.nist.gov/csrc/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp2959.pdf





TPM2.0 example SLB 9660/9665/9670

MMU (memory management with privilege levels)
MED (Memory Encrypt/Decrypt)
SCP (symmetric co-processor) for AES hardware acceleration
Asymmetric Crypto Co-processor (labeled Crypto) for modular math (e.g. RSA 2048-bit, ECC) acceleration
The checksum module (labeled CRC) allows simple calculation of 16-bit CRC checksums for Error Detection And Correction

Role ID	Role Description
CO	Cryptographic Officer, also known as the TPM Administrator or Admin Role. Controls certification of objects and changes Authentication Data of objects.
User	User, also known as the object owner. Uses the TPM to create cryptographic objects and to obtain cryptographic services for cryptographic objects.
DUP	Duplication Officer. Uses the TPM to duplicate TPM objects.

TPM Identification and Authentication Methods

Password Verification: Operators in the CO or User roles are authenticated by a demonstration of knowledge of a Password as authentication data.

↑ As TPM requirement

When using the password verification mechanism, a password consisting of at least 12 alphanumeric characters shall be used. Assuming as a worst case that the operator uses 12 decimal digits only, but still randomly chosen, the probability that a single random authentication attempt (by guessing the password value) will succeed is 10^{-12}.

A very conservative estimate of the maximum authentication rate is 10^6/minute (60 µs per attempt). Under this assumption the probability that random authentication attempts will succeed within a oneminute interval is $10^6 * 10^{-12} = 10^{-6}$

HMAC Challenge-Response Authentication

This Challenge-Response Authentication is described as HMAC Authorization Session within TCG Specifications. Operators in the CO or User roles are authenticated by a challenge and response demonstration of knowledge of a shared secret. The shared secrets are HMAC-SHA1 and HMAC-SHA256 cryptographic keys. The TPM HMAC authorization session mechanism includes nonce values (pseudo random values used once) to prevent replay attacks.

As a worst case it is assumed that for challenge-response authentication HMAC-SHA1 is used, which has smaller key length and smaller tag length (160 bit each) than HMAC-SHA256. Under this assumption the probability that a random authentication attempt (by guessing key value or tag value) will succeed is: $2^{-160} = 6.8 * 10^{-49}$

With the same assumed maximum authentication rate of 106/minute as above, the probability that random authentication attempts will succeed within a one-minute interval is $6.8 * 10^6 * 10^{-49} = 6.8 * 10^{-43}$

TPM Identification and Authentication Methods

Enhanced Authorization for Authentication

Operators in the CO or User roles can be authenticated via a policy digest, which requires as action the use of an authentication mechanism. Password Verification or HMAC Challenge-Response Authentication as described above, or Challenge-Response Authentication based on a Public Key Digital Signature Algorithm (RSA 2048-bit or ECDSA 256-bit) can be used as authentication mechanism. The TPM policy authorization session mechanism includes nonce values to prevent replay attacks.

For challenge-response authentication using a Public Key Digital Signature Algorithm as required authentication mechanism it is assumed as worst case that RSA 2048-bit is used, which provides 112-bit security strength (ECDSA 256-bit provides 128-bit security strength). Therefore the probability that a random authentication attempt will succeed using this authentication mechanism is:

$2^{-112} = 1.9\times10^{-34}$

With the same assumed maximum authentication rate of 10^6/minute as above, the probability that random authentication attempts will succeed within a one-minute interval is:

$10^6 \times 1.9\times10^{-34} = 1.9\times10^{-28}$

Another approach we have is trusted zone, other than having external chip for TPM.

Trusted Zone

CONCEPT INTRODUCED BY ARM

Key principle of trusted zone

Separation

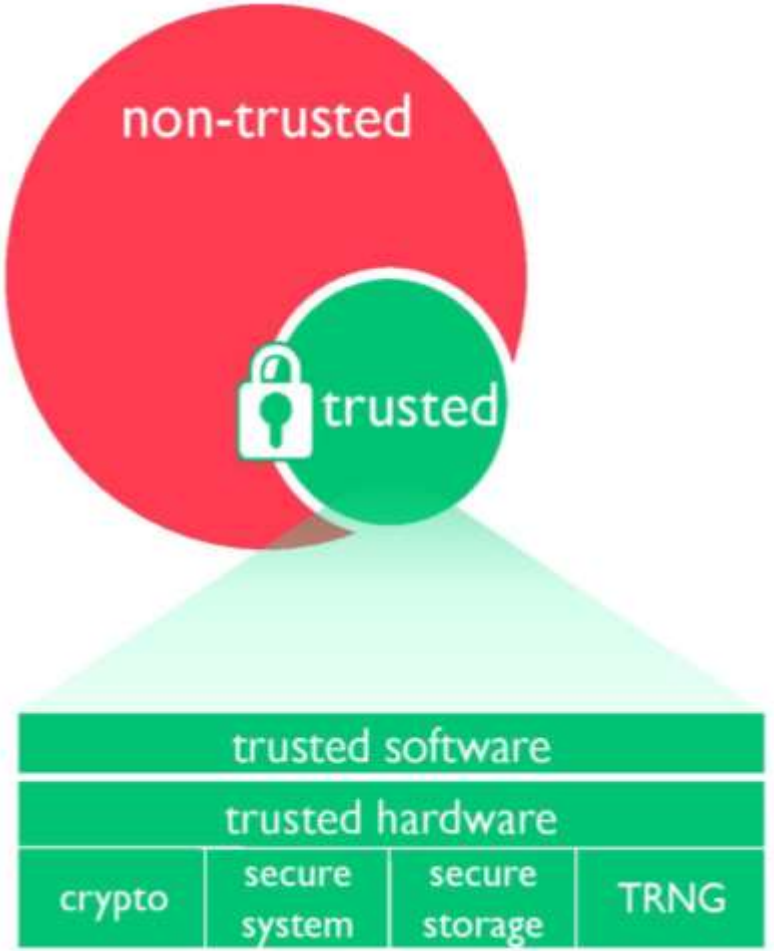
- Isolation of trusted resources (processors, memories, sensors, peripheral) from non-trusted ones
- Isolation of non-trusted SW
- Reduce propagation of attacks to key components
  (e.g. Memory partitioned into secure & non-secure regions;
  the non-secure world is unable to access memory tagged as secure)

Trusted Software

- Privilege rules supported
- Trusted boot & Root of Trust
- Short and commented and tested code
- Encrypted and authenticated SW/FW update

Trusted Hardware

- Hardware Security Modules for secure storage
- Hardware Security Modules for core cryptofunctions acceleration
- Access verification
- On-chip generation of random seeds



Is a trusted zone usually created with hardware security modules?

Yeah, Giovanni, you're on the right track. A **trusted zone**—sometimes called a **trusted execution environment (TEE)**—can definitely be created using **Hardware Security Modules (HSMs)** or **security-dedicated cores** like the HSE we talked about.

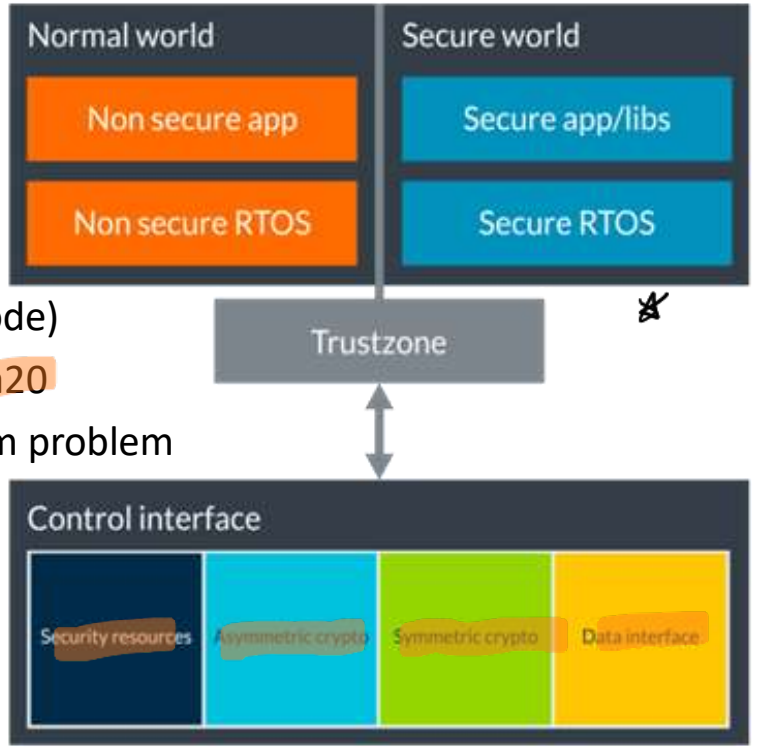
In automotive systems or other embedded setups, the idea is to carve out a **secure, isolated environment** within the system where sensitive operations can happen safely—away from the potentially compromised or general-purpose parts of the system.

Example: Trusted Zone in ARMv8

Trusted Zone implemented only on 64bits processors: it is related to multicore: you can have bad guys working together with safe guys.

Feature/Architecture	TrustZone for Armv8-A	TrustZone for Armv8-M
Additional security states	SEL0 - Trusted Apps SEL1 - Trusted OS EL3 - Trusted Boot and Firmware (Armv8-A)	Secure thread - Trusted code/data Secure handler - Trusted device drivers, RTOS, Library managers...
Secure interrupts	Yes	Yes (Fast)
State transition (Boundary crossing)	Software transition	Hardware transition (Fast)
Memory management	Virtual memory MMU with secure attributes	Secure Attribution Unit (SAU) and MPU memory partitions
System interconnect security	Yes	Yes
Secure code, data and memory	Yes	Yes
Trusted boot	Yes	Yes
Software	Arm trusted firmware (and third-party TEEs)	Keil CMSIS, Arm mbed OS, mbed uVisor and third-party software

Crypto-cells in ARM



Crypto-cell 312 for for Cortex-M or Cortex-R in IoT applications

200MHz in 40 nm

HASH: SHA (Secure Hash Algroithm) SHA1, SHA2-256; HMAC (Hash-based message authentication code)

Symmetric cryptography: Engine for AES (Advanced Encryption Standard) with 128b keys and Chacha20

Public-key cryptography: based on RSA (Rivest-Shamir-Adleman) and Elliptic Curve Discrete Logarithm problem

Crypto-cell 700 for for Cortex-A

400 MHz/500 MHz in 28nm/16nm

Symmetric encryption: AES

- Confidentiality modes: ECB, CBC, CBC-CTS, OFB and CTR
- Storage modes: ESSIV, BitLocker and XTS
- Message Authentication Codes (MAC): CBC-MAC, CMAC and XCBC-MAC
- Authenticated Encryption with Associated Data (AEAD) modes: CCM and GCM key sizes of 128, 192 and 256 bits
- Software and Hardware introduced keys

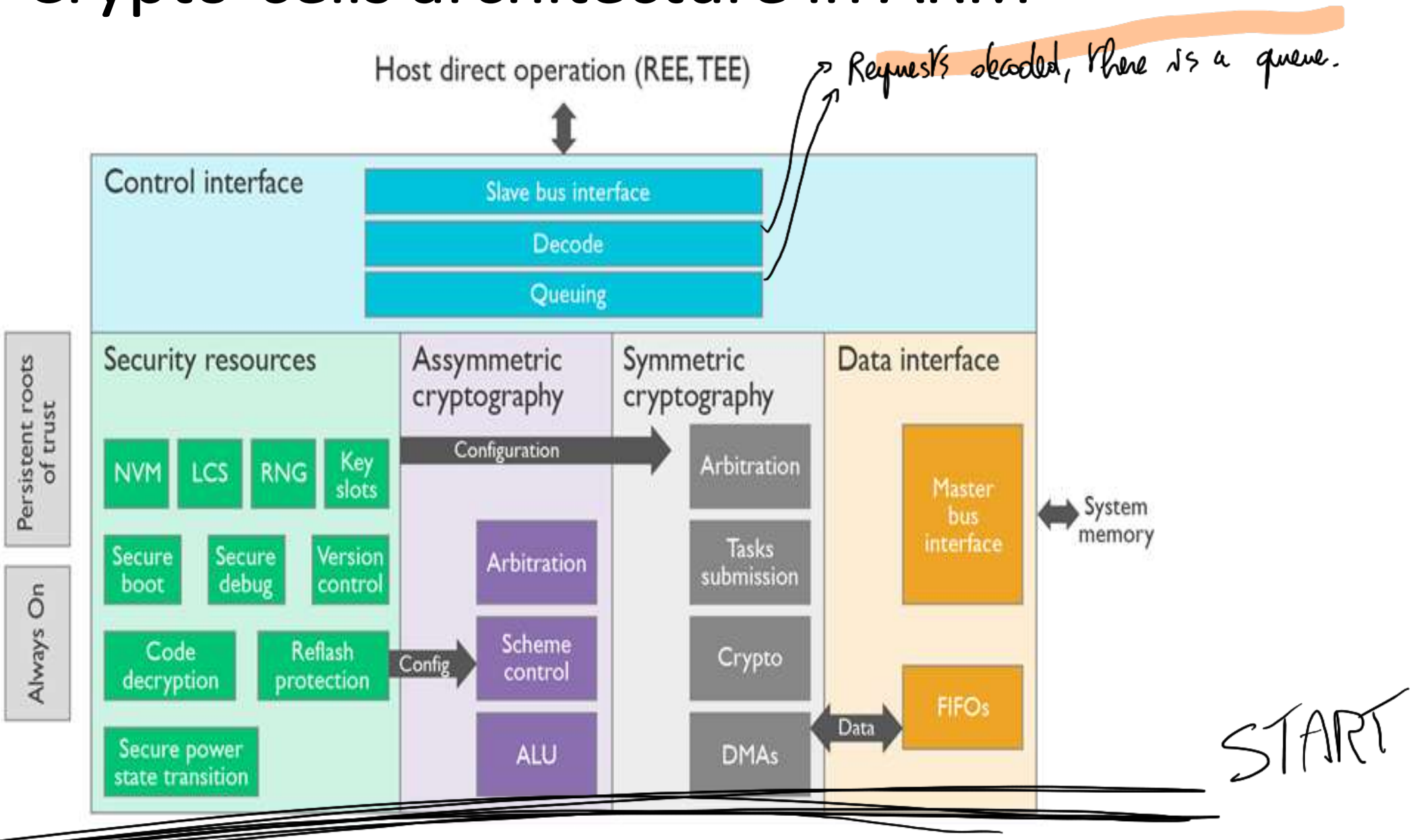
HASH: SHA1, SHA2-256/384/512 and MD5 modes, as well as HMAC

Public-key cryptography (128 bits and 4K bits in size): based on the Discrete Logarithm problem,
the Elliptic Curve Discrete Logarithm problem (ECDH, ECDSA), and the Integer Factorization problem

On-chip RNG: True Random Number Generator (TRNG) + Deterministic Random Bit Generator (DRBG)

* A chip with cores divided in secure world w/ secure system and apps and normal world. We then have control interface to manage secure world and offer security services. Making sure that everything is up to the defined rules.

Crypto-cells architecture in ARM



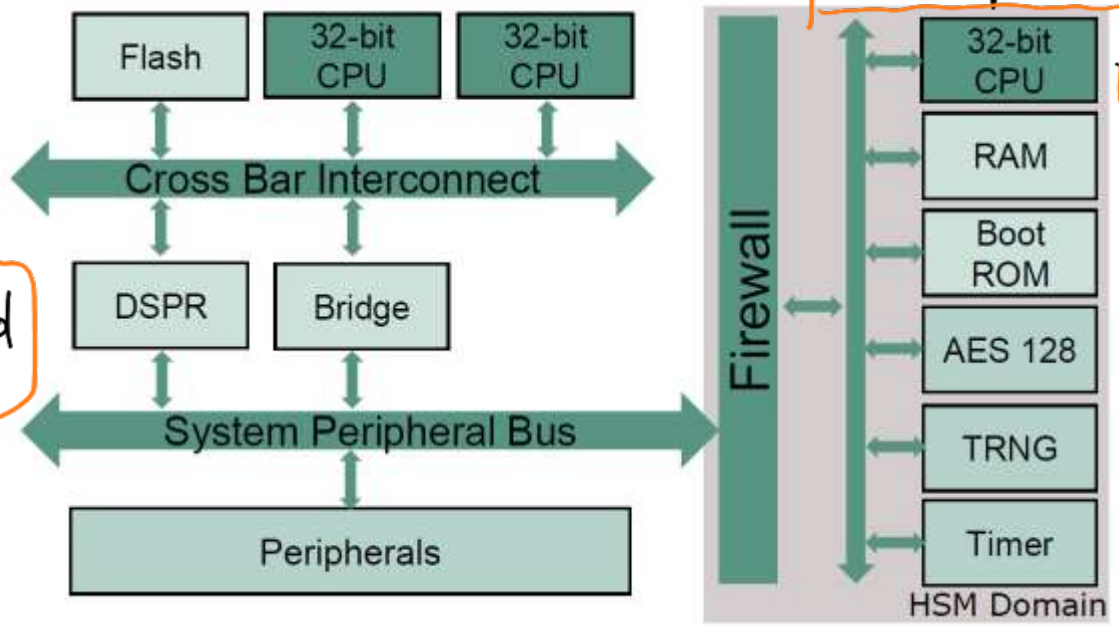
Hardware security in Infineon Aurix TC2XX

https://www.infineon.com/dgdl/Infineon-AURIX_Hardware_Security_Module-Training-v01_01-EN.pdf?fileId=5546d46269bda8df0169ca6e34c62549

- On-chip True RNG (128-bit random numbers AIS-20/31 class P2 compliant)
- Hardware acceleration for AES-128 (Cipher modes: ECB, CBC, CTR, CFB, 0FB, GCM, XTS)
- AES CMAC with minimum rate 25 MBytes/s
- Dedicated storage for secure keys (in separate Flash portion 64KB in TC29x and TC27x)
- 32 bit ARM Cortex M3 processor with up to 100 MHz CPU speed.
- MPU (Memory Protection Unit)

One of the most used chips for ECU in automotive field.

- Tw-core. We have two 32 bits CPU for safety (& add redundancy), so usually not for increased capability.



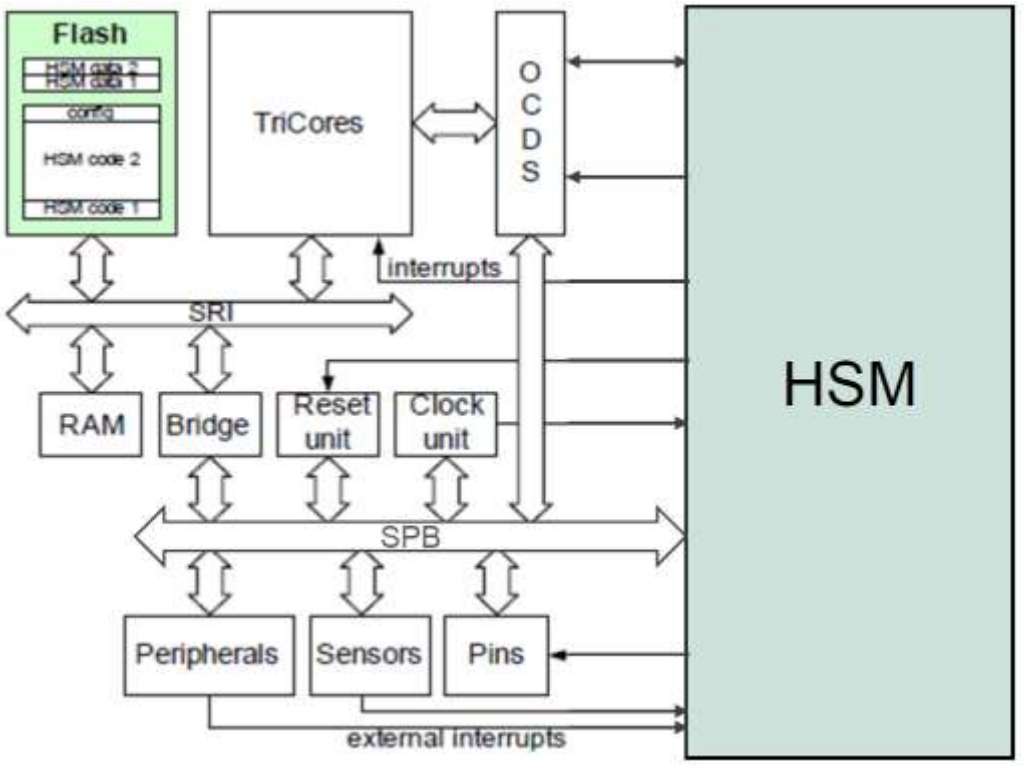
Separated by a firewall, there's the other HSM domain with dedicated timer (for CTR mode, for example) &

* Of course some PRBG is there too, hardware acceleration based on AES. Plus, the dedicated CPU inside HSM is programmable, so you can implement algorythms you might need. Then you have a MPU.

These specifications are not TPM/EVITA compliant; we are missing HW solutions for PKE for instance.

Hardware security in Infineon Aurix TC2XX

HSM is connected with the device via the SPB (System Peripheral bus)
The Bridge module acts as a „firewall“ so the HSM internal resources are protected from accesses by other masters
HSM, as a system on chip, is a bus master on the SPB
Supports challenge-response authentication



This component exists in several version (TC27S etc) with slight performance differences (ex. memory size). We can analyze how HSM changed in different versions. Before we analyzed TPM chips (only for security services cannot be used for general purpose ops.), here, you have HSM integrated inside of processor. This is a MICROCONTROLLER

Interesting: This HSM supports challenge and response authentication for working with the rest of the system. This is kind of similar to the idea of trusted zone in ARM.

Hardware security in Infineon Aurix in TC3XX

https://www.infineon.com/dgdl/Infineon-AURIX_TC3xx_Hardware_Security_Module_Quick-Training-v01_00-EN.pdf?fileId=5546d46274cf54d50174da4ebc3f2265

- On-chip True RNG (128-bit random numbers AIS-20/31 class P2 compliant, throughput typ. 360kb/s)
- Hardware acceleration for AES-128 (Cipher modes: ECB, CBC, CTR, CFB, 0FB, GCM, XTS)
- Dedicated storage for secure keys (in separate Flash portion 128KB)
- 32 bit ARM Cortex M3 processor with up to 100 MHz CPU speed.
- MPU (Memory Protection Unit)
- new vs. TC2XX: Hardware acceleration for for public key cryptograhy ECC 256 (100 to 200 ver./s) → supported ECC (Elliptic Curve Cryptography) curves: NIST, Brainpool, ED25519
- new vs. TC2XX: SHA224/256 Hardware Accelerators (98 Mbyte/s SHA256 module performance)

Evolution of TC2: concepts were the same

- In TPM we don't discuss much throughput: there you talk through an interface which is slower and has the idea that security checks happen mostly at beginning of sessions. Here we go for the idea that security is a continuous concern so we care about performances.
- Why does the TRNG only has throughput of 360KB/S? Very low! But a TRNG relies on physical changes, which happen slowly.

Hardware security in Infineon Aurix

https://www.infineon.com/cms/en/about-infineon/company/cybersecurity/

Same concept, but extended

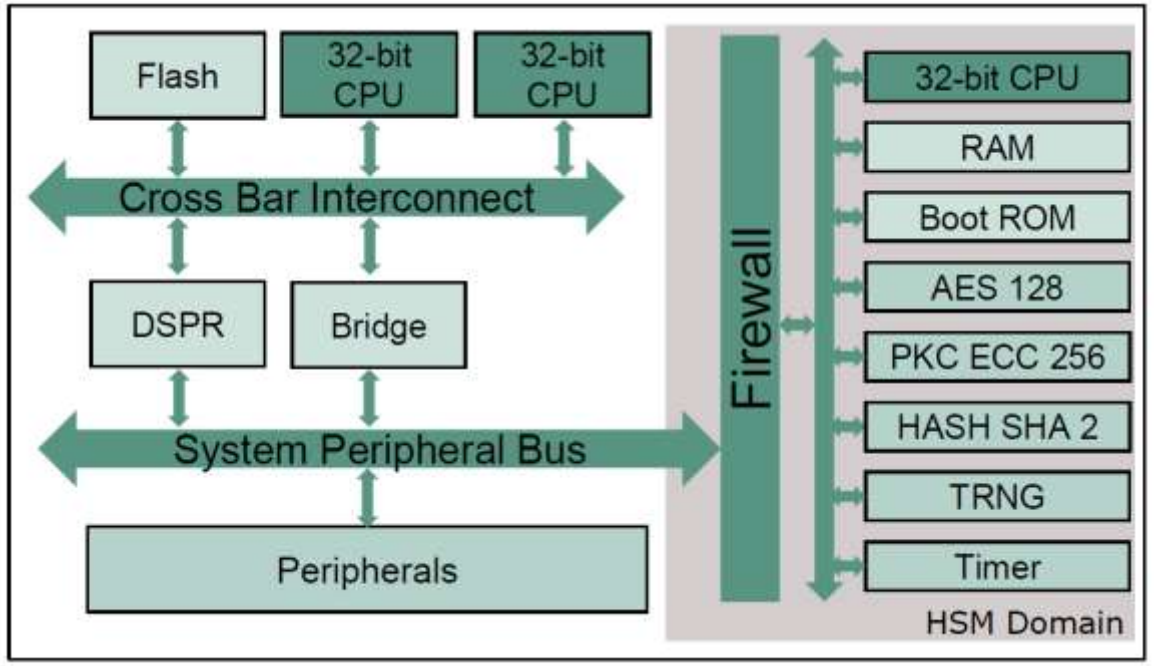
Secure Hardware Extension (SHE+) in TC2xx, TC3xx

The **S**ecure **H**ardware **E**xtension (SHE+) drivers control the hardware security peripheral in the HSM domain.
SHE+ offers an interface to the AUTOSAR CRY (**Cry**pto library module) to integrate the HSM security features into an automotive application. This includes the interface to AUTOSAR and communication from HSM and vice versa, key storage, management functionality and security peripheral drivers.

SHE+ can be added to existing MCAL (Microcontroller Abstraction Layer) Driver packages and is configured with the Tresos Studio. This is an Eclipse-based tool for the configuration of AUTOSAR software modules.

SHE+ can offer the following functionalities:
Symmetric Data Encryption/Decryption
MAC Generation/Verification + Safe MAC Verification
Random Number Management
Secure Boot
Debug Access for development
Asymmetric Encryption / Decryption

Infineon has Cyber Defense Center (CSIRT) and PSIRT (Product Security Incident Response Team)



- You don't need only the HW, but an enriched set of libraries that help your OS to exploit the functionalities you offer. Infineon is providing those Secure Hw Extension + drivers, exactly for this purpose (SHE+)

This library is compatible with AUTOSAR*, common environment specification for providers of HWs, SWs for automotive systems. This is a standard.

So everything compliant with AUTOSAR can use SHE+.

(1)

* Drivers are compatible with AUTOSTAR STANDARD for ECUs in vehicles.

AUTOSAR (short for **AUTomotive Open System ARchitecture**) is a worldwide development partnership of vehicle manufacturers, suppliers, and other companies in the automotive and software industries. It was created to **standardize the software architecture of Electronic Control Units (ECUs)** in vehicles.

Think of it like this: modern cars are packed with ECUs that control everything from airbags to engine timing to infotainment. Without a common framework, every manufacturer (and sometimes every model) might have its own way of doing things, making updates, compatibility, and collaboration really messy. AUTOSAR comes in to clean all that up, by setting a modular and layered architecture where the software is portable and reusable across different systems and vehicles.

It's kind of like giving everyone the same set of LEGO rules and blocks, so they can build cool car functions without reinventing the wheel every time.

Imagine you're writing software for a car's ECU. Without MCAL, every time you switch microcontroller vendors (like moving from an Infineon chip to an NXP one), you'd have to rewrite all the low-level code that talks to the hardware. Total headache.

MCAL steps in to save the day by acting as a **translator between your software and the microcontroller hardware**. It provides standardized interfaces to control peripherals like GPIOs, ADCs, PWMs, CAN, SPI, etc., no matter which microcontroller you're using.

So, if AUTOSAR is like a big, layered cake, MCAL is the bottom layer that makes sure everything above can stay the same—even if the hardware changes underneath.

Basically, it lets developers focus on the *what* their code should do, not the *how* it talks to hardware. Makes life way easier, especially in an industry where hardware changes often but safety and reliability are non-negotiable.

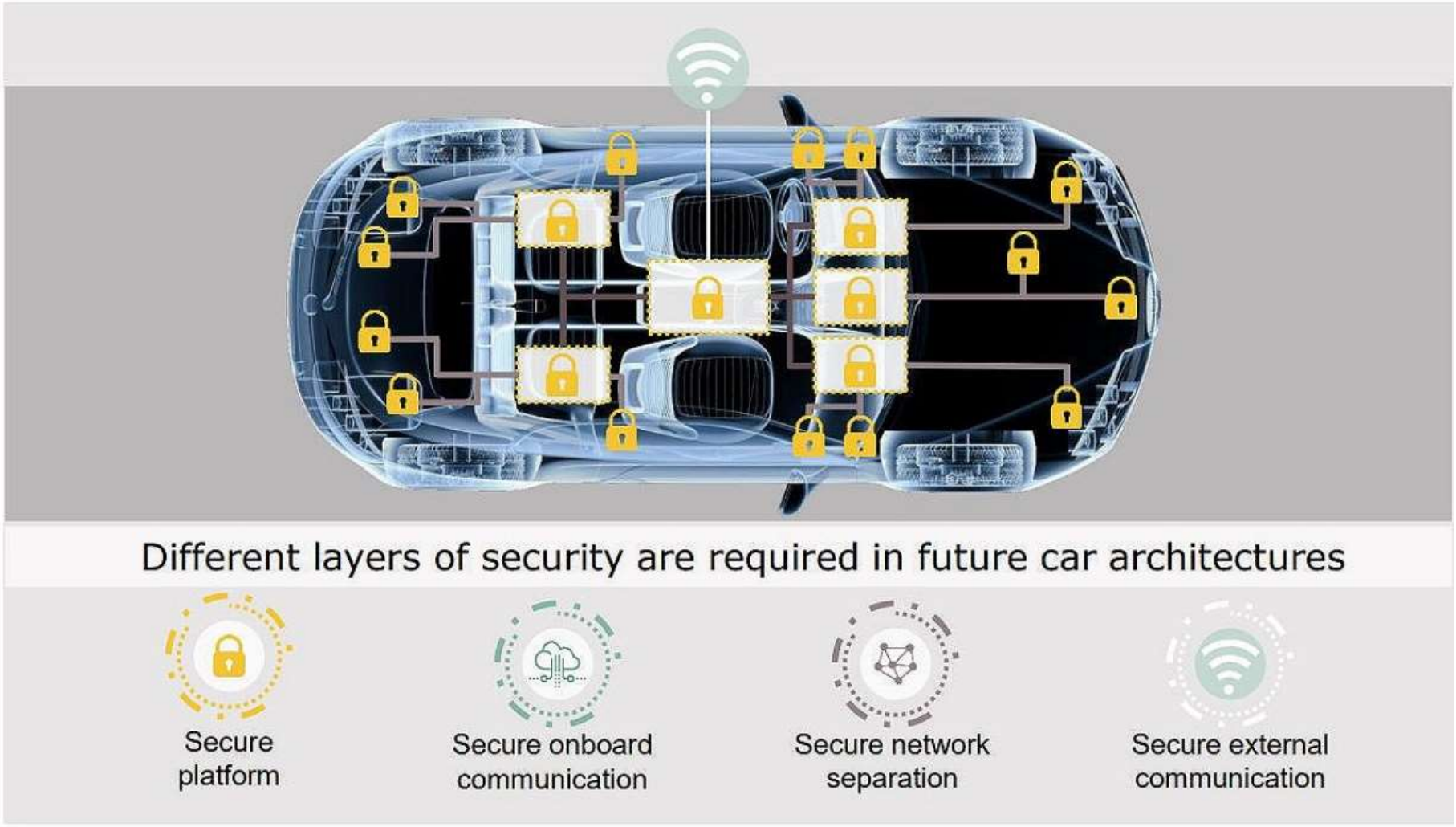
AUTOSAR *is* a **standard**—or more precisely, a **set of standards**—developed and maintained by the AUTOSAR partnership. It defines everything from software architecture and interfaces to communication protocols and more. It's like the rulebook that says, "If you want to make software for cars that plays nice with other components and companies, follow this."

Now, **MCAL** is **part of the AUTOSAR standard**, specifically in the **Classic Platform**. So yes, it's standardized *within* AUTOSAR. It defines a bunch of modules and APIs that hardware vendors are supposed to implement for their specific microcontrollers. That way, no matter which chip you're using, the software built on top doesn't have to change much, because the interface stays the same.

Hardware security in Infineon Aurix TC4XX

Hardware security features in AURIXTC4xx

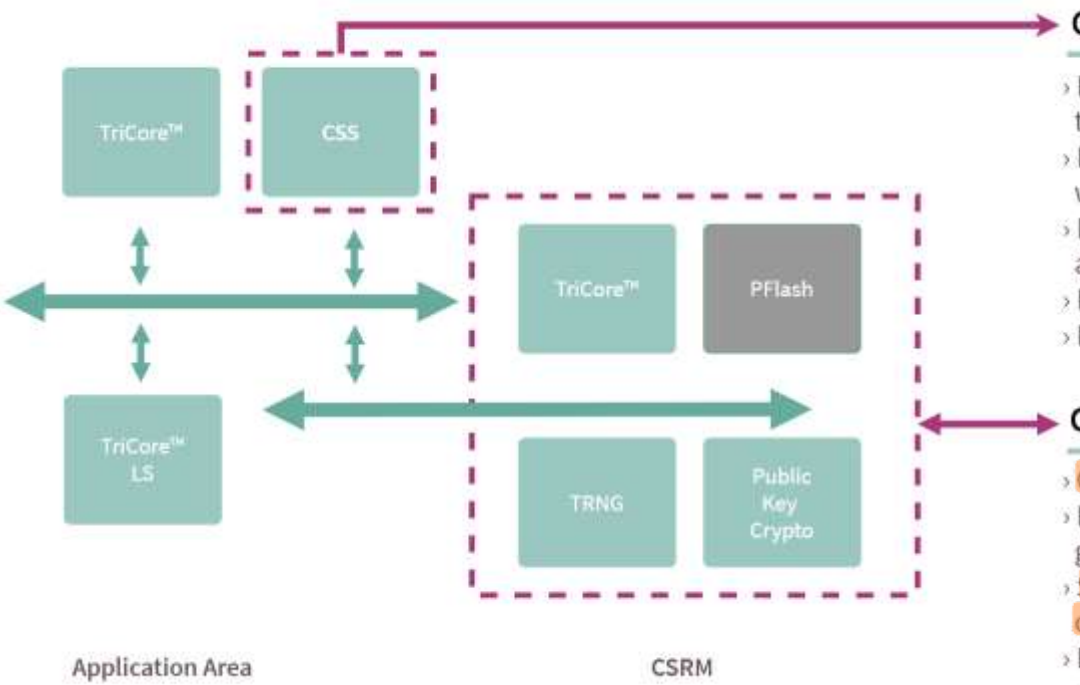
https://www.infineon.com/cms/en/product/promopages/safety-security-and-connectivity/



In TC4 there's new support for high level technology, fast network communication for safety and management of high volume of data from sensors, network separation for security, secure internal and external communication.

Hardware security in Infineon Aurix in TC4XX

AURIXTC4xx has a cybersecurity cluster with 2 modules: cybersecurity real time module (CSRM) + cybersecurity satellite (CSS)



Cyber Security Satellite (CSS)

- Parallelization of HW accelerators is service provider to application area
- Multiple channels to be used by independent applications with different trust levels
- Multiple HW accelerators to increase through put avoid performance bottlenecks
- Providing freedom of interference
- Providing support to ASIL-D use-case

Cyber Security Real-time Module(CSRM)

- CSRM as trusted secure HW environment → So it is a trusted zone
- Providing~5-15x more performance vs.previous generation.
- Support individual security SW updates independent of application core
- Enable realization of multiple security use cases for wide ranging applications

CSS is the novel addition to the cluster.
It enables the parallelization of hardware accelerators as a service provider to the application area, allowing for multiple channels to be used by independent applications with different trust levels.
Multiple hardware accelerators increase throughput, avoid performance bottlenecks and provide freedom of interference for safety-related applications.

We work with a CS cluster with two modules:

* Not anymore a simple 32 bits CPU, but another TriCore! It's like a brand new TC3 inside.
  - On chip we have a complete processor for security, while another works like TPM (CSS).
- Tricore:

Hardware security in Infineon Aurix in TC4XX

One of the key features of the CSx is that it supports a variety of use cases with a specific focus on communication requirements, which are increasing in the evolving vehicle E/E architecture.
Nevertheless, CSx also provides special attention to in-vehicle network as well as to vehicle-to-infrastructure (V2X)use cases:
- Intrusion Detection System
- Intrusion Detection Prevention System
- Firewall: feasible with hardware filters in MAC and software
- Authenticated Encryption with Associated Data
- Authentication with Associated Data
- Combined modes

All this allows:
Minimizing latency and maximizing throughput, as an increasing number of security use cases are expected in the future
compliance to new security standards, namely **ISO 21434 and UNECE WP.29**
Enabling software-over-the-air use cases, which require secure and safe distribution of software updates from the cloud or within the vehicle network
↳ They guarantee over-the-air use case
Serving Authenticated Encryption with Associated Data and Authentication with Associated Data solutions, which are expected to gain importance in the future

Features that might be needed

Part A: Q&A session

HSM (Hardware Security Module)

Prof. Sergio Saponara,
Dip. Ingegneria della Informazione, Università di Pisa
sergio.saponara@unipi.it
+39 3468790937