SECURITY FOR THE AUTOMOTIVE EDGE NODES

AMF-AUT-T2354

OSVALDO ROMERO

APPLICATIONS ENGINEER

SECURE CONNECTIONS FOR A SMARTER WORLD

AGENDA

- Why do we need security?
- NXP Layered security model
- S32K Overview
- SHE Specification
- S32K144 CSEc

Objectives

- Understand the need of security in the car
- Get familiar with the SHE specification
- Learn on the CSEc module and how is complies wit SHE
- Understand how the CSEc could help in you security application.

THE NEED FOR SECURITY

Today: 90% of Auto Innovation via Electronics

NXP: THE GLOBAL MARKET LEADER IN AUTOMOTIVE SEMICONDUCTOR SOLUTIONS

#1 INFOTAINMENT
- TUNERS
- SOFTWARE-DEFINED DIGITAL RADIO
- MULTIMEDIA PROCESSORS
- SOUND SYSTEM DSPs & AMPLIFIERS
- NFC BT PAIRING
- WIRELESS POWER CHARGING
- POWER MANAGEMENT

STANDARD PRODUCTS
- LOGIC
- POWER
- DISCRETES

#1 VEHICLE NETWORKING
- CAN/LIN/ FLEXRAY
- ETHERNET
- CENTRAL GATEWAY CONTROLLER
- SECURITY
- RF

#1 BODY
- MICROCONTROLLERS
- POSITION/ ANGLE SENSORS
- SYSTEM BASIS CHIPS

ADAS & SECURITY

#1 SAFETY
- MICROCONTROLLERS AIRBAG
- ANALOG AIRBAG
- MICROCONTROLLERS BRAKING
- ANALOG BRAKING
- SENSORS BRAKING
- TIRE PRESSURE MONITORING

POWERTRAIN & CHASSIS
- MICROCONTOLLERS
- PRESSURE/ MOTION SENSORS
- BATTERY MANAGEMENT
- DRIVERS

#1 SECURE CAR ACCESS
- IMMOBILIZER/ SECURITY
- REMOTE KEYLESS ENTRY
- PASSIVE KEYLESS ENTRY/ GO
- BI-DIRECTIONAL KEYS
- NFC
- ULTRA WIDE BAND

#1 Auto Analog/ RF | **#1 Auto MCU (ex JPN)** | **#1 Auto Merchant MEMS Sensors**

Increasing Connectivity = Increasing Risks

FBI: Estimated 3 Trillion USD Annual Damage from Hacking

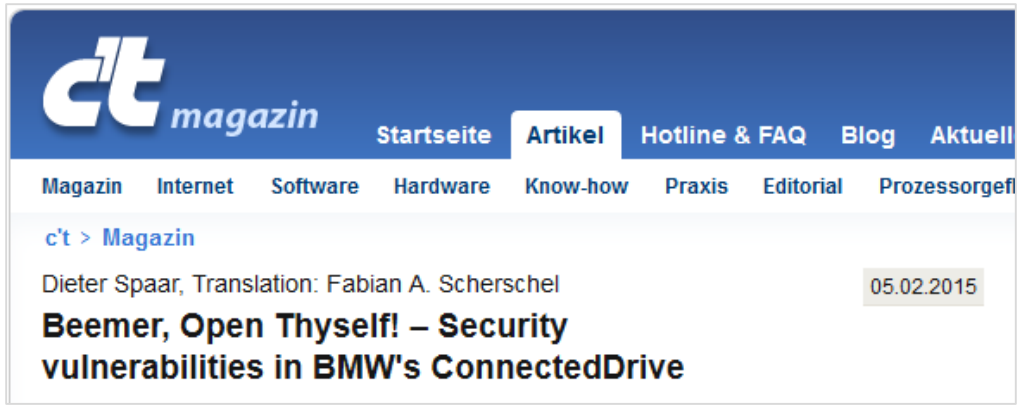
Requiring maximum protection of . . .

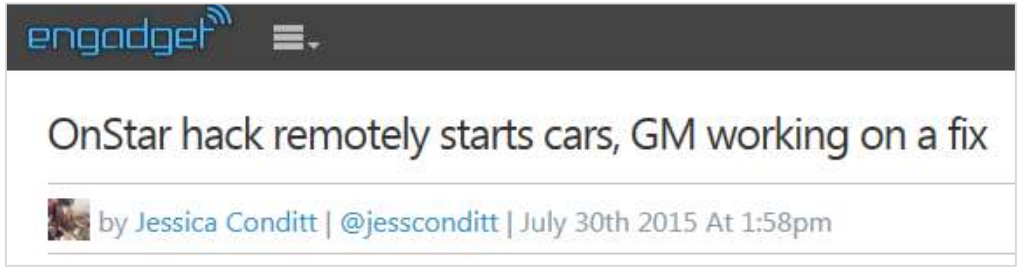
Privacy

Personal Assets

Lives

Car Hacking is ‘Hot’

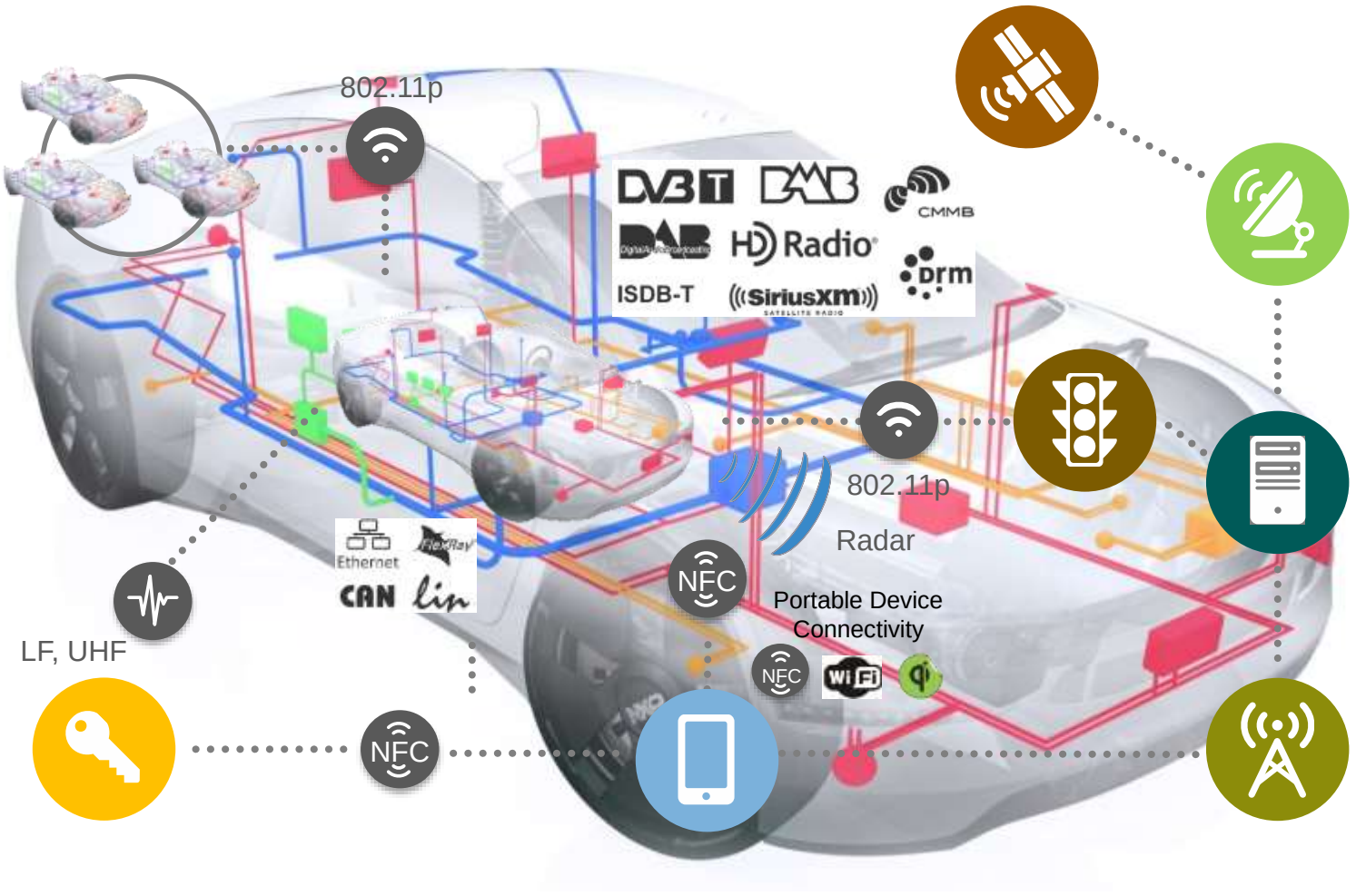




The Connected Car…

A Cloud-connected Computer Network on Wheels

- **A networked computer**
  - up to 100 ECUs per car
  - and many sensors
  - inter-connected by wires
  - more and more software
- **Increasingly connected to its environment**
  - to vehicles & infrastructure
  - to user devices
  - to cloud services



… is an Attractive Target for Hackers!

Valuable Data

- Collection of data/info
- Storage of data
- Diagnostic functions

Protect Privacy

High Vulnerability

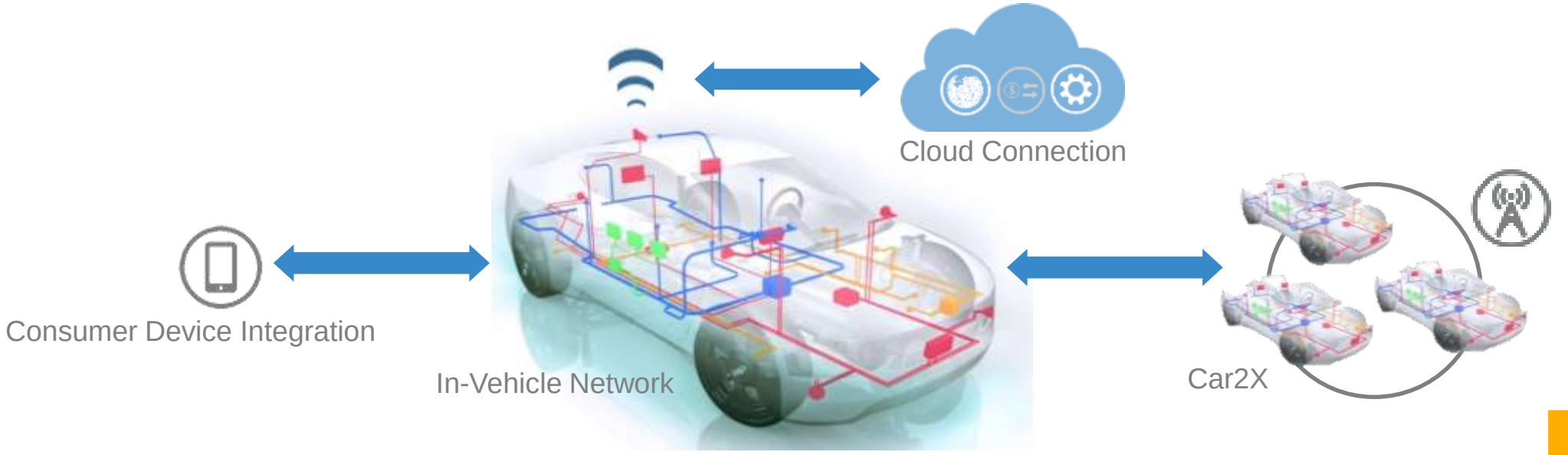
- Increasing number of nodes
- More advanced features
- X-by-Wire

Increase Safety

Easy (Remote) Access

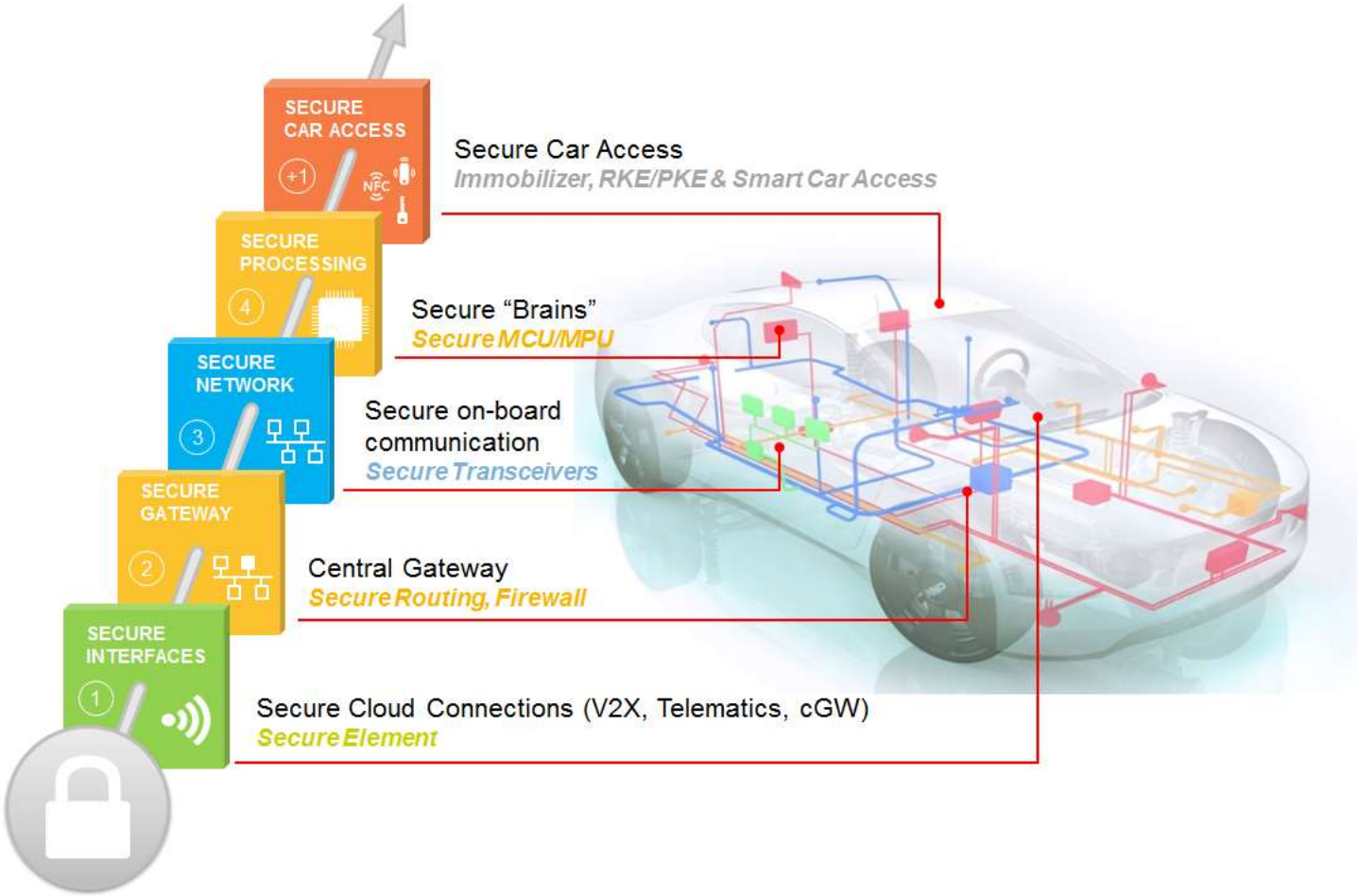
- Fully Connected Car
- External & internal interfaces
- Wired & wireless interfaces

Prevent Unauthorized Access



LAYERED SECURITY MODEL

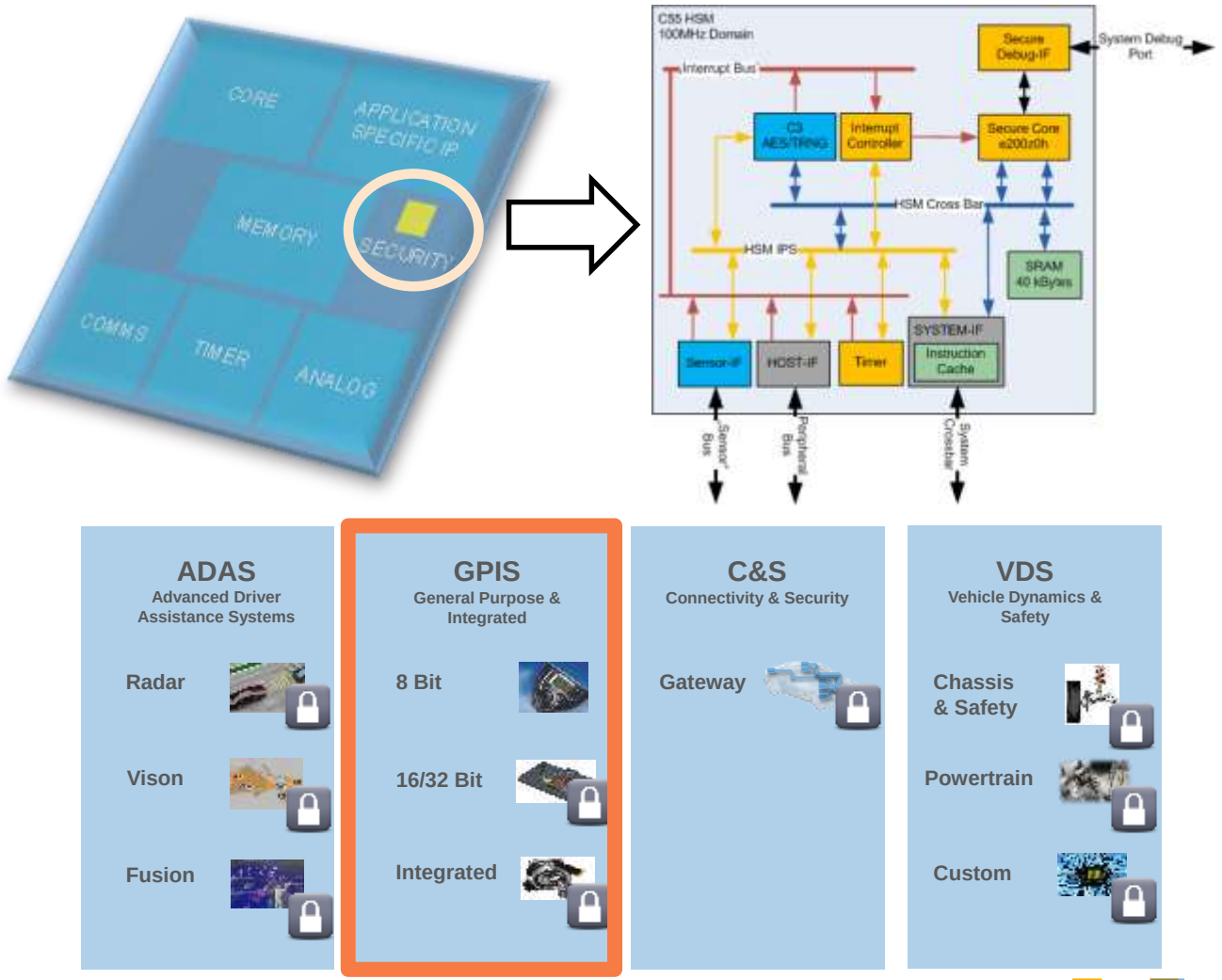
NXP Automotive Vehicle Security Architecture (4 +1 Solution)



- NXP #1 in Auto HW Security
- 4-Layer Cyber Security Solution
- Plus 'Best In Class' Car Access Systems
- Recognized Thought & Innovation Leader
- Partner of Choice for OEMS, T1s & Industry Alliances

Layer 4 – Secure Processing: What is It?

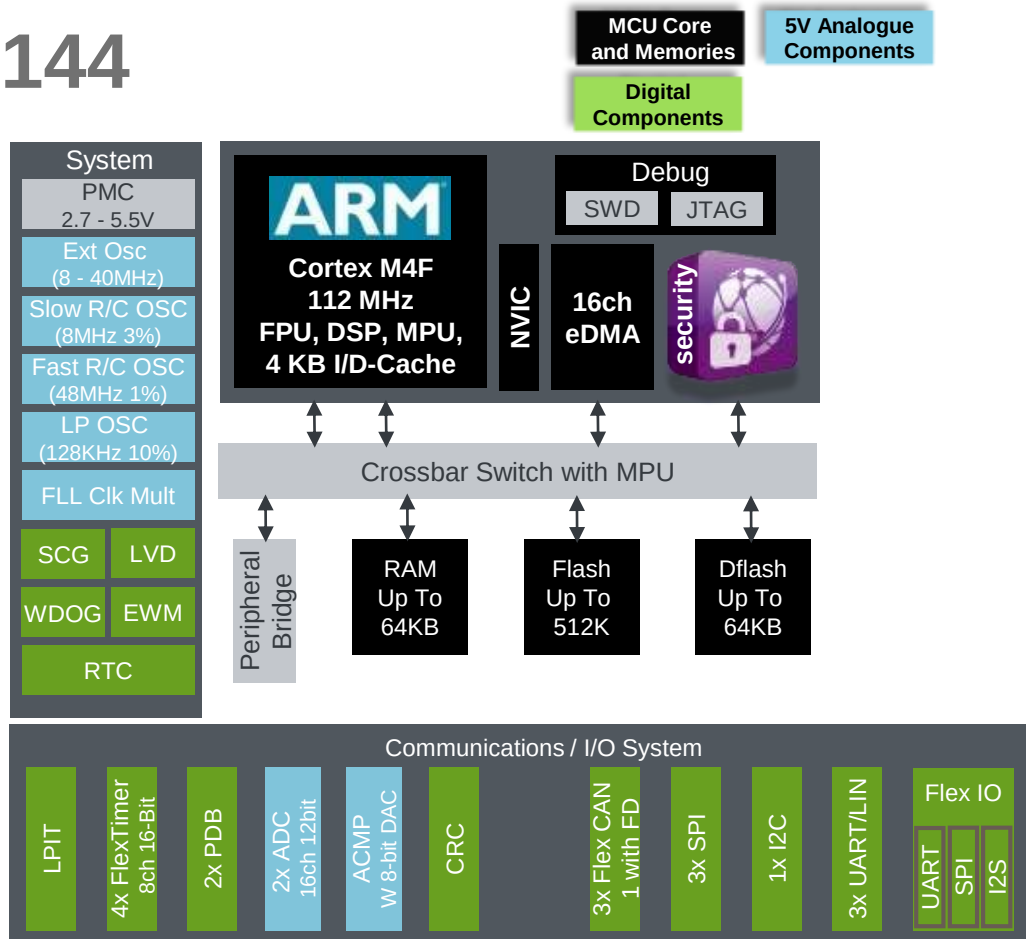
- Secure MCU - Defined by hardware accelerated Crypto capability
- IP can be applied to any MCU/Processor
- Use cases:
  - CAN Message authentication
  - Secure boot – FW auth.
  - Key storage
  - Encryption
  - OTA software updates in the field



S32K PORTFOLIO

S32K PORTFOLIO - S32K144

- **High performance**
  - ARM Cortex M4F up to 112MHz w FPU
  - eDMA from 57xxx family
- **Software Friendly Architecture**
  - High RAM to Flash ratio
  - Independent CPU and peripheral clocking
  - 48MHz 1% IRC – no PLL init required in LP
  - Registers maintained in all modes
  - Programmable triggers for ADC no SW delay counters or extra interrupts
- **Functional safety**
  - ISO26262 support for ASIL B or higher
  - Memory Protection Unit
  - ECC on 512K Flash / 64K Dataflash and RAM
  - Independent internal OSC for Watchdog
  - Diversity between ADC and ACMP
  - Diversity between SPI/SCI and FlexIO
  - Core self test libraries
  - Scalable LVD protection
  - CRC
- **Low power**
  - Low leakage technology
  - Multiple VLP modes and IRC combos
  - Wake-up on analog thresholds
- **Security**
  - CSEc (SHE-spec)



Packages & IO

- Open-drain for 3.3 V and hi-drive pins
- Powered ESD protection
- Packages: 100 BGA, 64 LQFP, 100 LQFP

Operating Characteristics

- Voltage range: 2.7V to 5.5V
- Temperature (ambient): -40°C to +125°C

S32K Portfolio: Targeting General Purpose Applications

Product Longevity 10 15


Body control module


Human machine interface

Wireless charging

Battery Management


Tire pressure receiver


Climate control

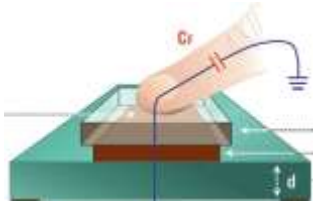

Door/Window/sunroof

Near Field Communication


Lighting

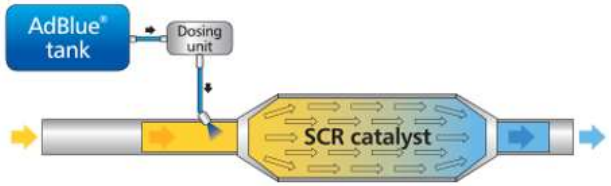
Secure transmission / encryption in cars


PMSM/BLDC motorcontrol



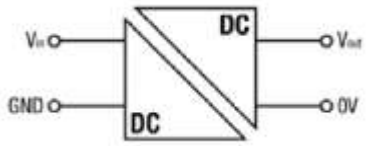
Touch sensing


Park assist



Nox reduction systems


Motorbike ECU/ABS



DC/DC converters


E-shifter


Rear view camera tilt


Steering wheel electronics

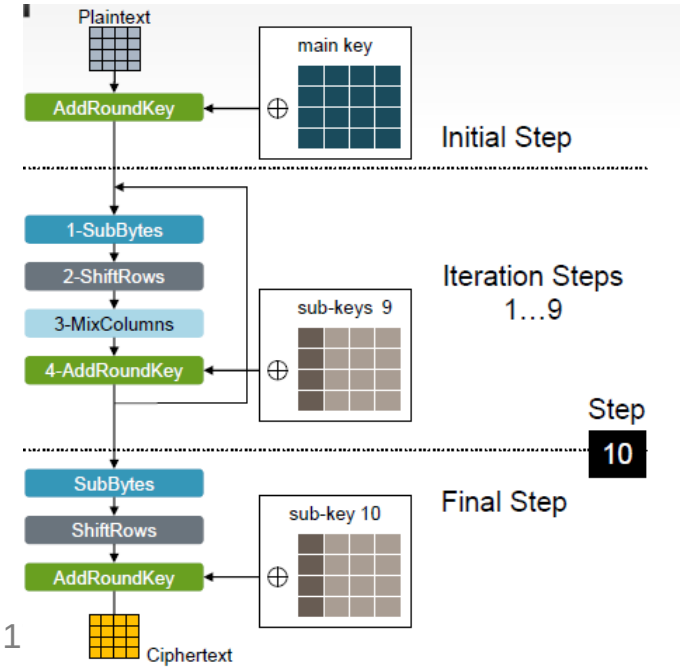
THE HIS-SHE SPECIFICATION OVERVIEW

SHE Specification- Introduction

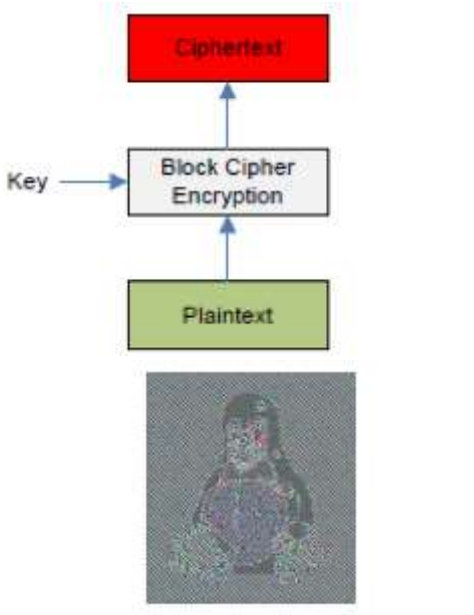
- Created by Audi (main driver), BMW and Escrypt
- Published as a official HIS standard
  (HIS => Herstellerinitiative Software, German for 'OEM software initiative')
- Re-view of the Spec. by Freescale/NXP in an early phase
- Key features of the SHE specification:
  - A secure storage for crypto keys
  - Crypto algorithm acceleration (AES-128)
  - Secure Boot mechanism to verify custom firmware after reset
  - Offers 19 security specific functions
  - Up to 10 general and 5 special purpose crypto keys

SHE Specification – Crypto Unit

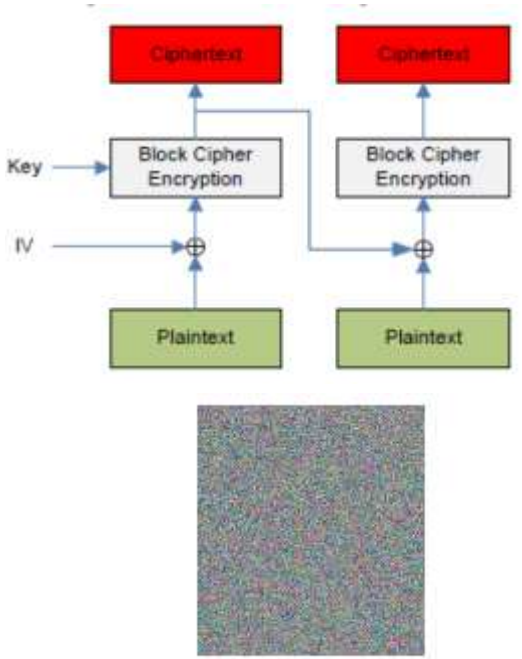
- Crypto and decryption algorithm: AES-128
- AES Encryption/Decryption in ECB or CBC mode
- Miyaguchi-Preneel



Plaintext



ECB:Ciphertext



CBC:Ciphertext

SHE Specification – CMAC Generator

- Cipher based Message Authentication Code (CMAC)
- A MAC algorithm inputs:
  - Secret key
  - Message of arbitrary length
- A MAC algorithm output:
  - MAC value
- The MAC value protects both a message's data integrity as well as its authenticity.

SHE Specification – Keys

Key values moved from public memory space to secure memory space.

The secure memory space is only accessible by the security module. Application work with key references!

	Write Protection	Secure Boot Failure	Debugger Activation	Wildcard UID	Key Usage	Plain Key	Counter	Overall data bits
MASTER_ECU_KEY	X	X	X	X			X	160
BOOT_MAC_KEY	X		X	X			X	159
BOOT_MAC	X		X	X			X	159
KEY_<n>	X	X	X	X	X		X	161
RAM_KEY						X		129
SECRET_KEY		X[1]	X[1]					128
UID								120

[1] SECRET_KEY inherits its protection flags from MASTER_ECU_KEY

SHE Specification – Memory Update Protocol

- To add user keys the protocol as defined in the SHE specification must be used.
- This ensures confidentiality, integrity, authenticity and protects against replay attacks.
- To update the memory containing the keys the following must be calculated and passed to CSE: K1, K2, M1 ,M2 and M3.

Key	Calculation	Size
K1	$KDF(K_{AuthID},KEY\_UPDATE\_ENC\_C)$ KDF is key derivation function	128 bit
K2	$KDF(K_{AuthID}, KEY\_UPDATE\_MAC\_C)$ KDF is key derivation function	128 bit
M1	UID'|ID|AuthID - 256 bits	128 bit
M2	$ENC_{CBC,K1,IV=0}(C_{ID}'\|F_{ID}'\|"0...0"_{95}\|K_{ID}')$ CBC encryption using K1	256 bit
M3	$CMAC_{K2}(M_1\|M_2)$ CMAC calculation using K2	128 bit

SHE Specification – Functions

#	SHE – Functions	Usage
1 2 3 4	CMD_ENCRYPT_ECB CMD_ENCRYPT_CBC CMD_DECRYPT_ECB CMD_DECRYPT_CBC	Encryption / Decryption
5 6	CMD_GENERATE_MAC CMD_VERIFY_MAC	Signing / Authentication
7 8 9	CMD_LOAD_KEY CMD_LOAD_PLAIN_KEY CMD_EXPORT_RAM_KEY	Key Management
10 11 12	CMD_INIT_RNG CMD_EXTEND_SEED CMD_RND	Random Number System
13 14 15	CMD_SECURE_BOOT CMD_BOOT_FAILURE CMD_BOOT_OK	Secure Boot
16 17 18 19	CMD_GET_STATUS CMD_GET_ID CMD_CANCEL CMD_DEBUG	Module Handling

$K_1 = KDF(K_{AuthID}, KEY\_UPDATE\_ENC\_C)$
$K_2 = KDF(K_{AuthID}, KEY\_UPDATE\_MAC\_C)$
$M_1 = UID' | ID | AuthID$
$M_2 = ENC_{CBC,K1,IV=0}(C_{ID}' | F_{ID}' | "0...0"_{95} | K_{ID}')$
$M_3 = CMAC_{K2}(M_1 | M_2)$

CMD_LOAD_KEY
stores key value in secure NVM

Note:
To be able to update a key you have to know the actual key value or the MASTER_ECU_KEY value.

SHE Specification – Secure Boot

- Secure Boot verifies the custom firmware after POR
- SHE offer these secure boot flows:

1. Parallel Boot – Application core and SHE module comes out of reset at the same time

2. Sequential Boot – SHE module comes out of reset and verifies the custom code, after the application core comes out of reset and execute the code (independent of secure boot result!)

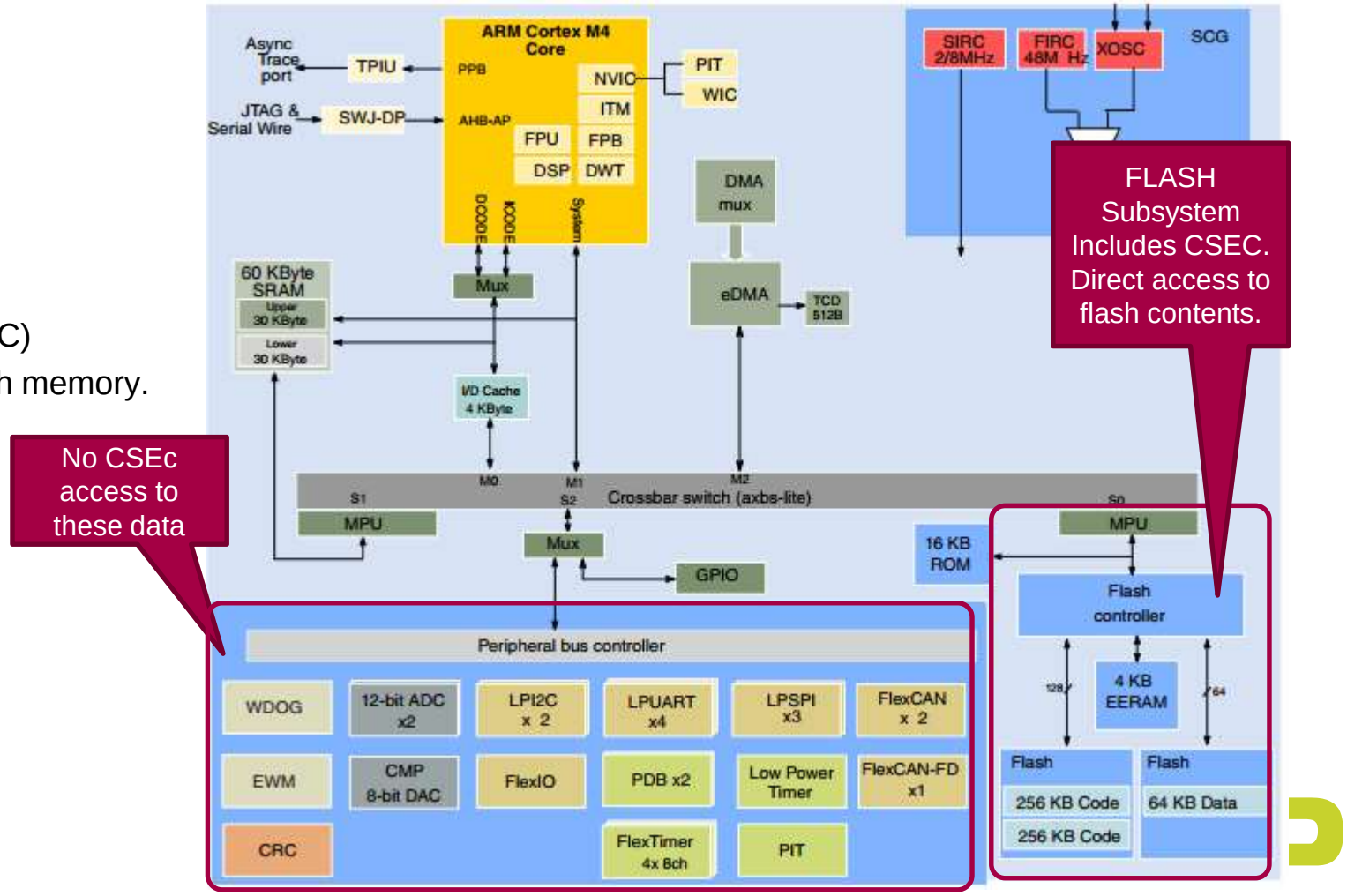
3. Strict Sequential Boot – SHE module comes out of reset and verifies the custom code, after the application core comes out of reset and execute the code if secure boot finalized positive

S32K144
CSEC

S32K Security Module (CSEc) – Overview

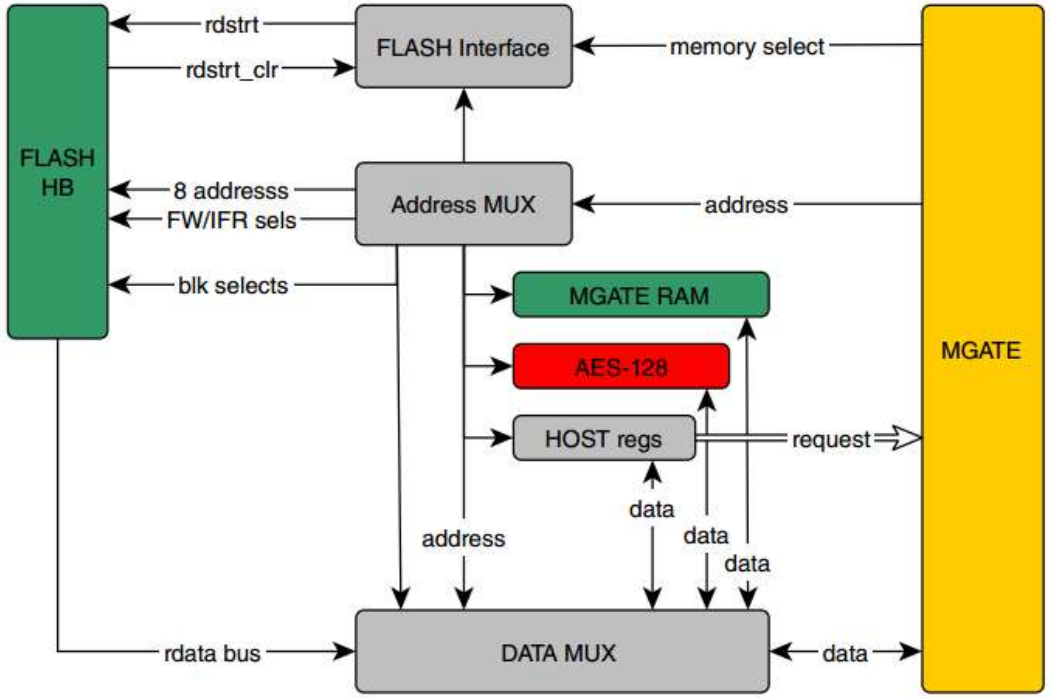
- SHE functionality moves from dedicated master module into the flash system
- Full SHE Specification compliant and support of all Global-B security requirements
- Secure key storage only accessible by CSEc
- True Random Number System
- Sequential boot / parallel boot supported
- CSEc supports AES-128 with ECB, CBC and CMAC mode
- Crypto Keys
  - Several General-Purpose keys
  - Special Purpose keys (e.g. Secret, Master and Secure-Boot Key & CMAC)
  - Support of additional encrypted keys in public flash memory.

- KEY-Properies
  - Write-protection
  - Secure-Boot-Failure
  - Debug-Connect
  - Wildcard-UID
  - Key-Usage (key or CMAC)
  - Verify-Only
  - 28bit-Update-Counter



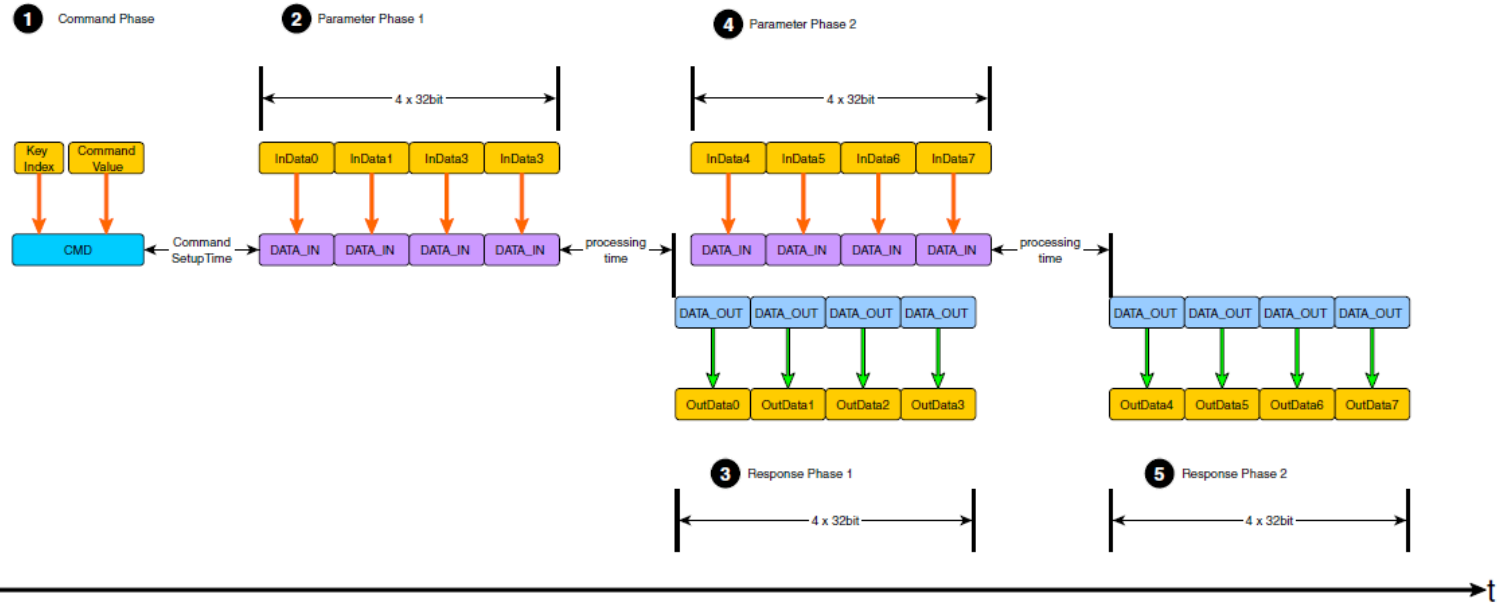
S32K Security Module (CSEc) – Overview

- Implemented directly in the flash system (close to the secure information)
- Direct memory access to the flash data for fast and simple secure boot support
- Data in SRAM / Peripheral are accessable via Core or DMA transfers.
- Supports the complete SHE Specification and the enhanced SHE+ features (more keys etc.)
- Small easy to use security implementation



S32K Security Module (CSEc) – Commands

- CSEc Commads to FTFC.
- CCOB command set is effectively extended to include SHE commands related to ECB, CBC and CMAC features.
- Similar protocol to the CCOB commands, CCOB interface will be locked until completion.
- CSEc command constructed by writing data to a Parameter Memory (PRAM) followed by a command header.
- Operation Start as indicated by CCIF, transition from 1 to 0.
- Operation complete: CCIF transition from 0 to 1. User read PRAM to verify results.



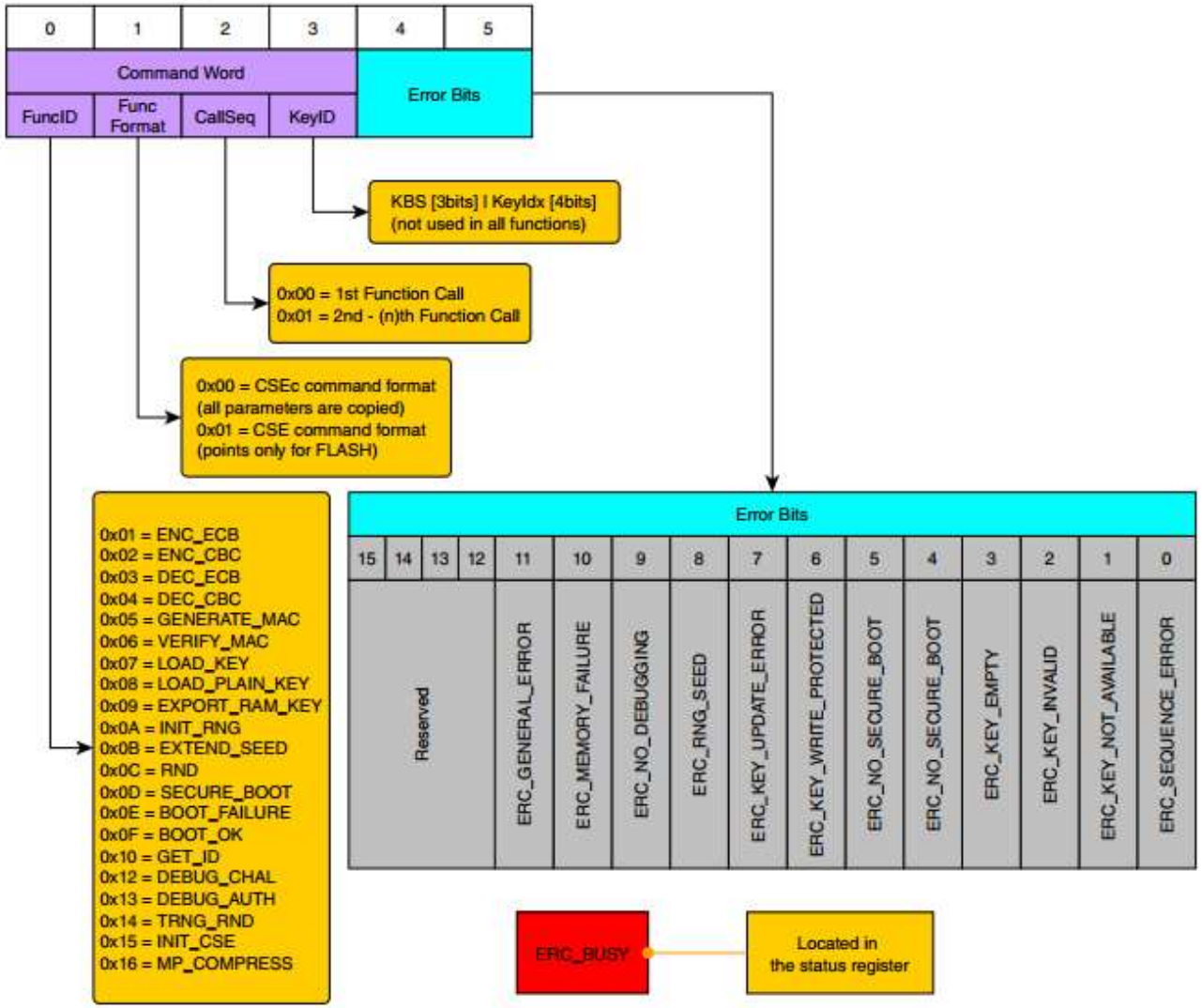
S32K Security Module (CSEc) – PRAM

- 128-bit (16 bytes) SRAM with 8 x128-bit (16 bytes) pages.
- Command header must be las data written
- Write to the command header locks PRAM.

Bits	[127:0]															
Bits	31:24	23:17	15:8	7:0	31:24	23:17	15:8	7:0	31:24	23:17	15:8	7:0	31:24	23:17	15:8	7:0
WD	Word 0				Word 1				Word 2				Word 3			
	Byte															
Page	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	FUN ID	CMD FOR MAT	CALL SEQ	KEY ID	ERROR BITS				COMMAND SPECIFIC I.E. PAGE LENGHT							
1	DATA INPUT OR OUTPUT FORM CSEc															
2																
3																
4																
5																
6																
7																

S32K Security Module (CSEc) – Commands Header

- FuncID: CSEc ID to execute
- Func Format: specify data transfer to CSEc: parametrs directly copied to PRAM or pointer method
- CallSeq: long data could be managed
- Key ID: SHE key index (KeyIdx) and key block selec (KBS)
- Error bits: Located in FCESTAT



S32K Security Module (CSEc) – Keys

Key name	KBS	Key Index
SECRET_KEY	X	0x0
UID	X	0x0
MASTER_ECU_KEY	X	0x1
BOOT_MAC_KEY	X	0x2
BOOT_MAC	X	0x3
KEY 01 – KEY 10	0	0x4-0xA
KEY 11 – KEY 17	1	0x4-0xA
RAM_KEY	X	0xF

S32K Security Module (CSEc) – CBC Encryption Command

Input Parameters

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0x02	0x00	0x00	KeyID	Error Bits											
1	IV[0-15]															
2	PlainText 1 [0-15]															
3	PlainText 2 [0-15]															
4	PlainText 3 [0-15]															
5	PlainText 4 [0-15]															
6	PlainText 5 [0-15]															
7	PlainText 6 [0-15]															

Output Parameters

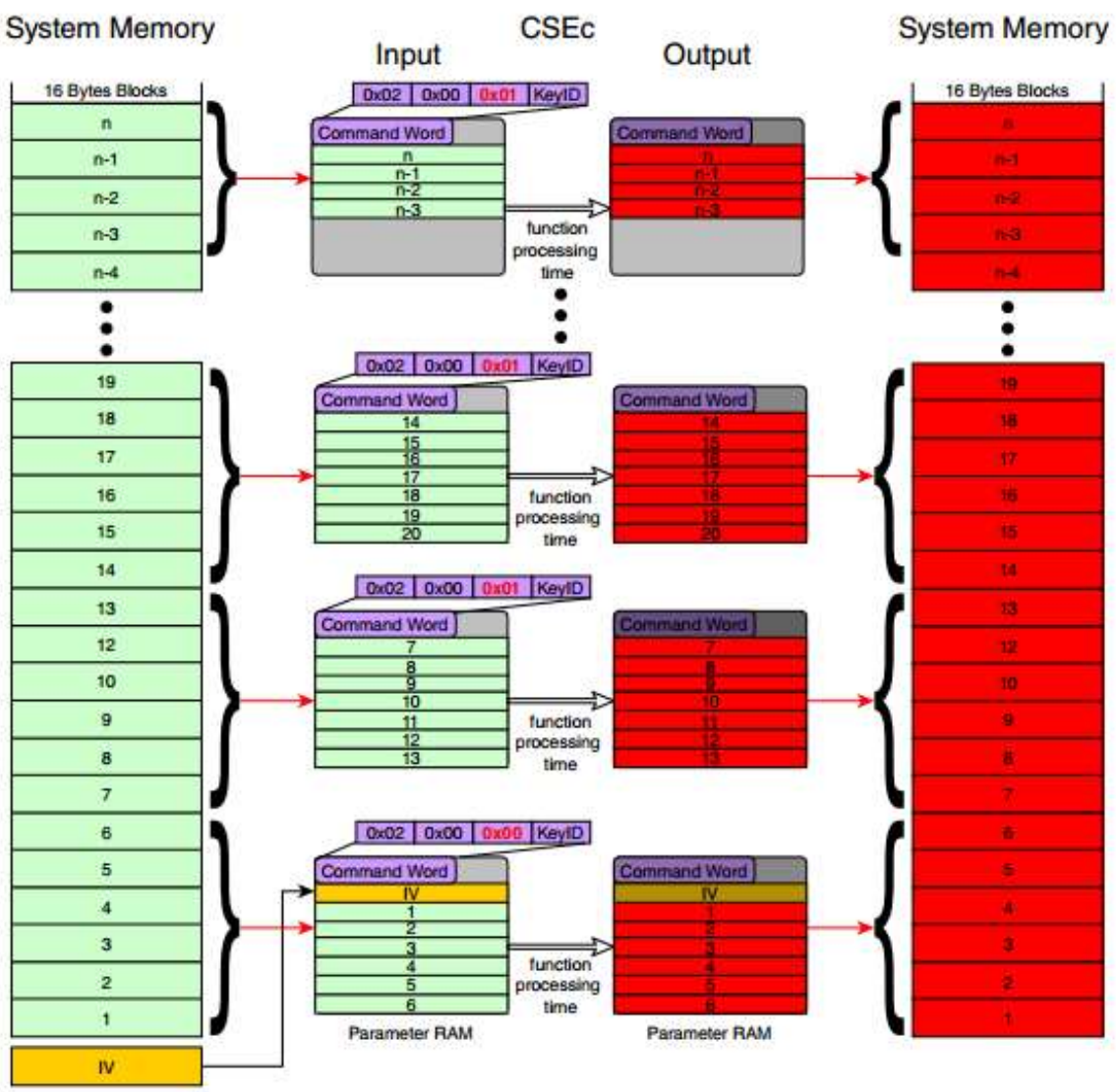
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0x02	0x00	0x00	KeyID	Error Bits											
1	IV[0-15]															
2	CipherText 1 [0-15]															
3	CipherText 2 [0-15]															
4	CipherText 3 [0-15]															
5	CipherText 4 [0-15]															
6	CipherText 5 [0-15]															
7	CipherText 6 [0-15]															

Input Parameters

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0x02	0x00	0x01	KeyID	Error Bits											
1	PlainText 7 [0-15]															
2	PlainText 8 [0-15]															
3	PlainText 9 [0-15]															
4	PlainText 10 [0-15]															
5	PlainText 11 [0-15]															
6	PlainText 12 [0-15]															
7	PlainText 13 [0-15]															

Output Parameters

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0x02	0x00	0x01	KeyID	Error Bits											
1	CipherText 7 [0-15]															
2	CipherText 8 [0-15]															
3	CipherText 9 [0-15]															
4	CipherText 10 [0-15]															
5	CipherText 11 [0-15]															
6	CipherText 12 [0-15]															
7	CipherText 13 [0-15]															



S32K Security Module (CSEc) – CMAC Command

- Generate MAC command operates on a MESSAGE using a key

- Two options:
  - Data Directly copied to PRAM
  - Pointer method

- Command Parameters
  - Key ID
  - Message Length
  - Message

Command Parameters

Parameter	Direction	Width
KEY_ID	IN	5
MESSAGE_LENGTH	IN	64
MESSAGE	IN	n * 128
MAC	OUT	128
MAC = CMACKEY, KEY_ID (MESSAGE, MESSAGE_LENGTH)		
Error Codes: ERC_NO_ERROR, ERC_SEQUENCE_ERROR, ERC_KEY_NOT_AVAILABLE, ERC_KEY_INVALID,ERC_KEY_EMPTY, ERC_MEMORY_FAILURE, ERC_BUSY, ERC_GENERAL_ERROR		

Data Directly Copied to PRAM

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0x05	0x00	0x00	KeyID	Error Bits		Reserved						MESSAGE_LENGTH			
1	DATA 1 [0:15]															
2	DATA 2 [0:15]															
3	DATA 3 [0:15]															
4	DATA 4 [0:15]															
5	DATA 5 [0:15]															
6	DATA 6 [0:15]															
7	DATA 7 [0:15]															

Pointer Method

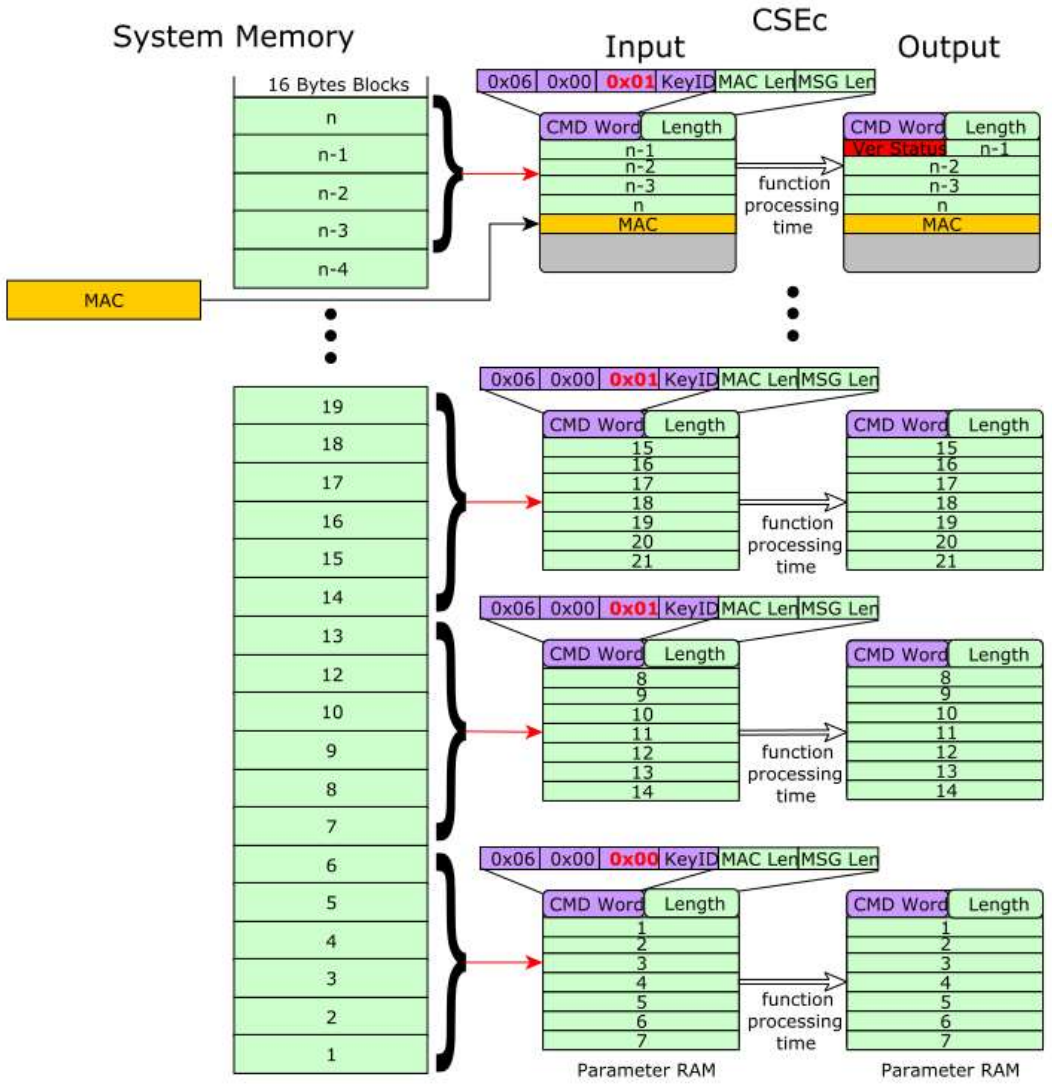
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0x05	0x01	0x00	KeyID	Error Bits		Reserved						MESSAGE_LENGTH			
1	Flash Start Address				Reserved											
2	Reserved															
3																
4																
5																
6																
7																

S32K Security Module (CSEc) – CMAC Verification

- The Verify MAC command verifies a MAC of a given MESSAGE

- Two options:
  - Data Directly copied to PRAM
  - Pointer method

- Command Parameters
  - Key ID
  - Message Length
  - Message
  - MAC
  - MAC Length



S32K Security Module (CSEc) – Load Key

- Update a Key in secure memory per SHE specification
- Command Input Parameters
  - M1
  - M2
  - M3

$ENC_{CBC,K1,IV=0}(C_{ID}'|F_{ID}'|"0...0"_{95}|K_{ID}')$

Check CBC slide

UID'|ID|AuthID

Figure 32-26. Load key input parame

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0x07	0x00	0x01	KeyID	Error Bits		Reserved									
1	M1 [0:15]															
2	M2 [0:15]															
3																
4	M3 [0 :15]															
5	Reserved															
6																
7																

$CMAC_{K2}(M_1|M_2)$

Check CMAC slide

S32K Security Module (CSEc) – Boot Define

- Allow user to define the Boot size
- User to select the boot mode
- Input Parameters
  - Boot size
  - Boot Flavor

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0x11	0x00	0x00	KeyID	Error Bits		Reserved									
1	Reserved											Boot Flavor	BOOT_SIZE			

Table continues on the next page...

USE CASES

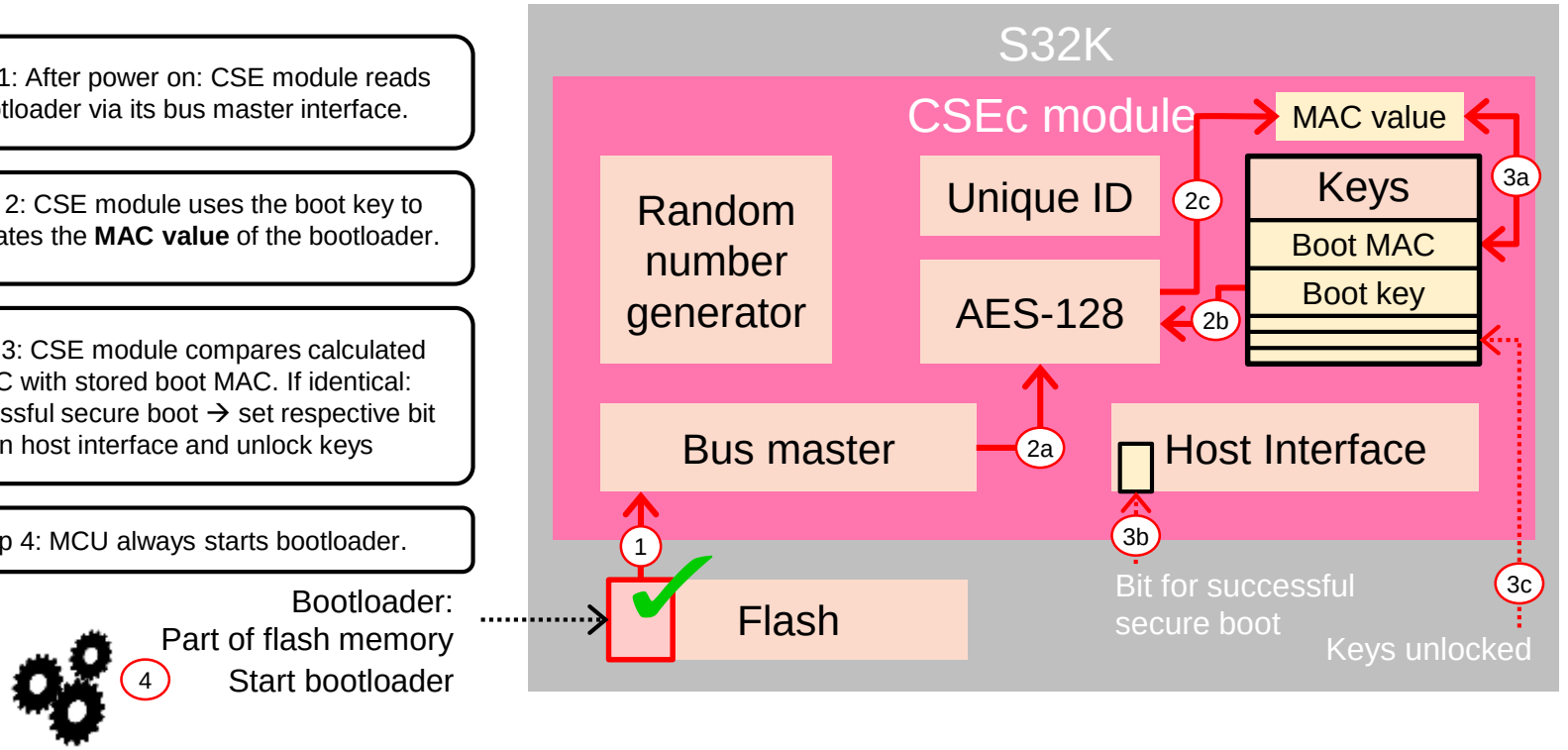
Secure Boot - Check Boot Loader for Integrity and Authenticity

Step 1: After power on: CSE module reads bootloader via its bus master interface.

Step 2: CSE module uses the boot key to calculates the **MAC value** of the bootloader.

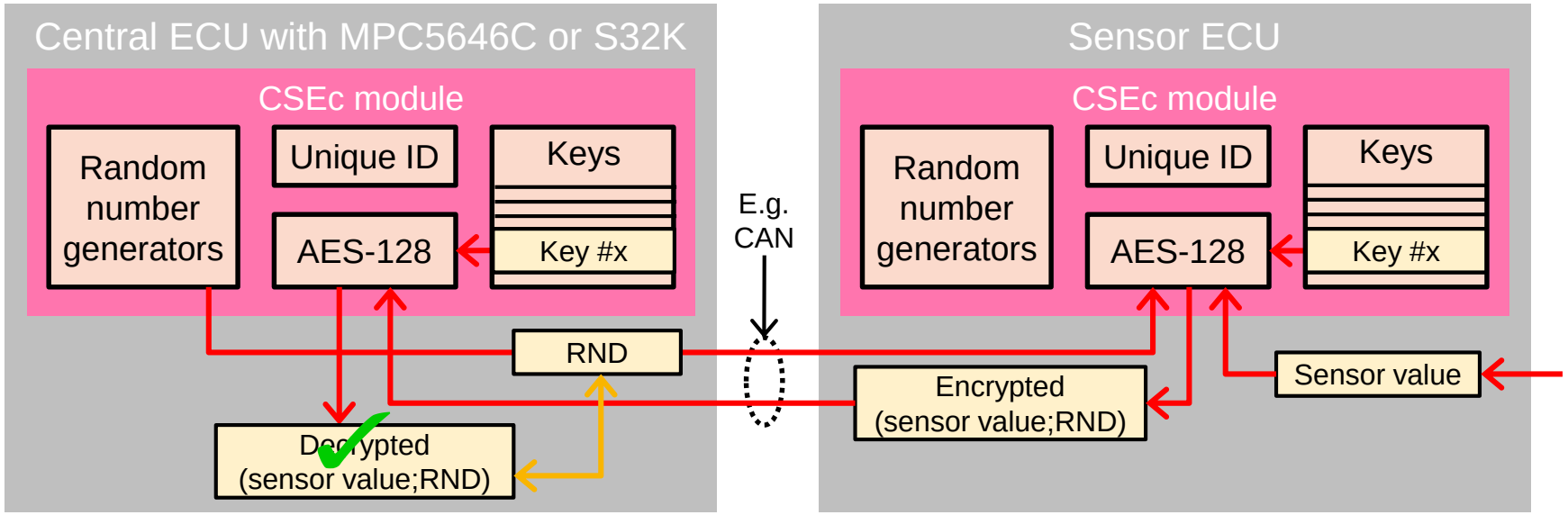
Step 3: CSE module compares calculated MAC with stored boot MAC. If identical: successful secure boot → set respective bit in host interface and unlock keys

Step 4: MCU always starts bootloader.



- **MAC** protects against modification of bootloader and depends on the (secret) boot key → integrity and authenticity of bootloader.
- Only if calculated MAC value matches stored boot MAC value: successful secure boot → set respective bit in host interface and unlock keys for further usage (see next demos)

Secure Communication



Step 1: Central ECU obtains random number and sends it to sensors ECU (e.g., after power-on of car)

Step 2: Sensor ECU reads sensor value and asks CSE module to encrypt it and the received random number (using key #x)

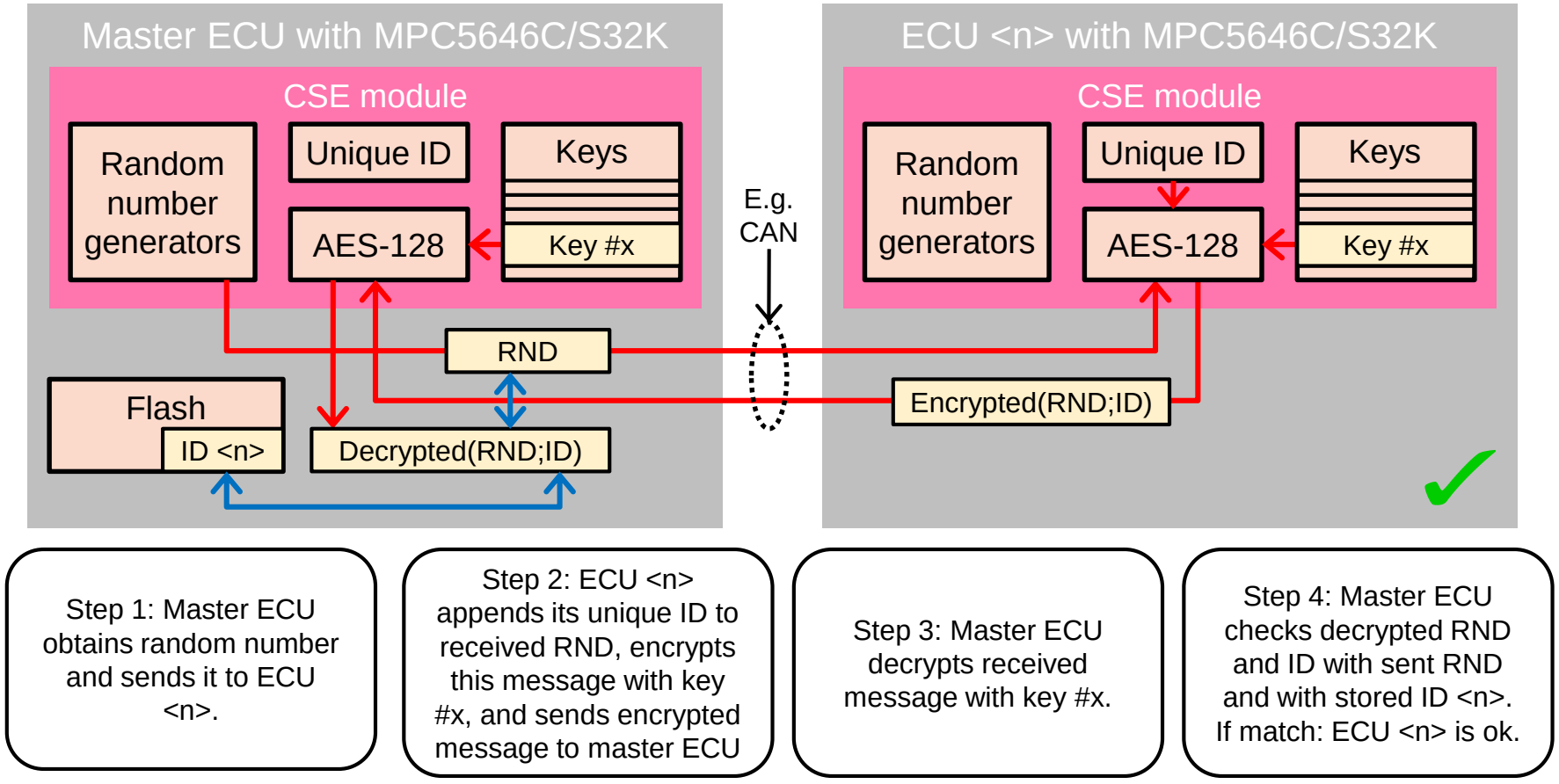
Step 3: Sensor ECU sends encrypted message to central ECU.

Step 4: Central ECU asks CSE module to decrypt received message (using key #x).

Step 5: Central ECU checks sent random number vs. received/decrypted random number.

- Random number: protects against replay attacks.
- Encryption: protects against eavesdropping.
- Random number and encryption: ensures data integrity and authenticity.

Component Protection - Detect replacement or Modification of Components (e.g. ECU)



- Replacement or modification of ECU <n> will change its unique ID and/or keys. Both will be detected with this proposal for component protection.

Application Notes

- AN4234 - Using the Cryptographic Service Engine (CSE)
- AN4235 - Using CSE to protect your Application Code via a Chain of Trust

In <AN4234SW/tools/bin>

AES_CMAC_CMD.exe

Usage AES_CMAC_CMD.out <Key Value> <Message Length in Bits> <Message File name>

AES_ENC_CBC_CMD.exe

Usage AES_ENC_CBC_CMD.out <Key Value> <IV Value> <Message File name>

AES_ENC_ECB_CMD.exe

Usage AES_ENC_ECB_CMD.out <Key Value> <Plaintext Value>

AES_MP_KDF_CMD.exe

Usage is AES_MP_KDF_CMD.c <key value> <key constant>

Summary

- Car hacking is a reality
- CSEc can help you to encrypt your data
- CSEc is able to generate CMAC values.
- Secure communication is possible with S32K144
- Application Firmware authentication is possible with S32K144

SECURE CONNECTIONS
FOR A SMARTER WORLD