FTF 2016
TECHNOLOGY FORUM

RECENT ADVANCES IN SECURE MCU SECURITY OFFERINGS

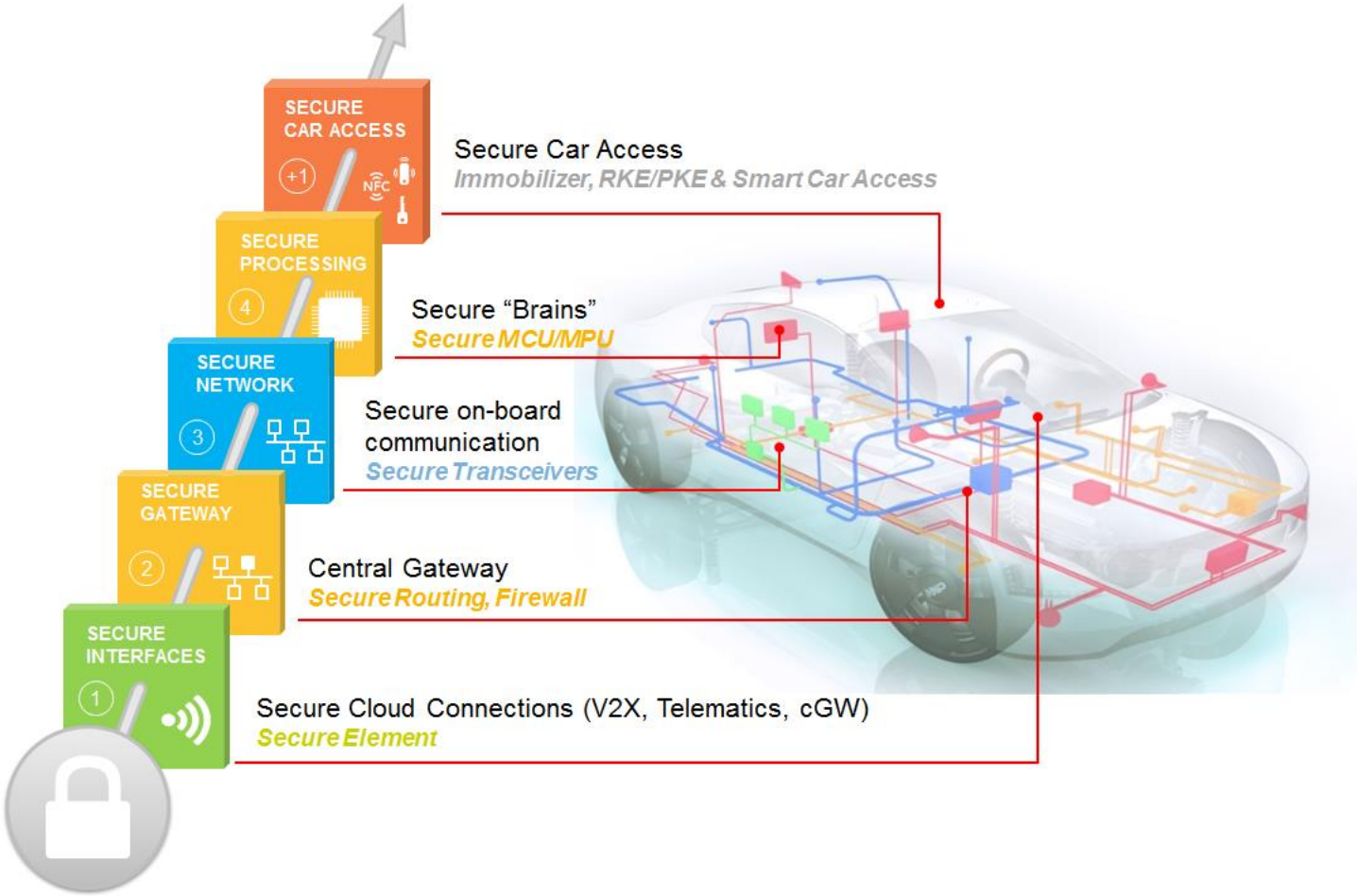
FTF-AUT-N1812

JUERGEN FRANK
SR. SYSTEM ENGINEER
FTF-AUT-N1812
MAY 16, 2016

AGENDA

- Security Use Cases and Attacks
- The HIS - SHE Specification Overview
- Crypto Service Engine (CSE) Overview
- CSE3 on Flash-less Devices (e.g. S32V243)

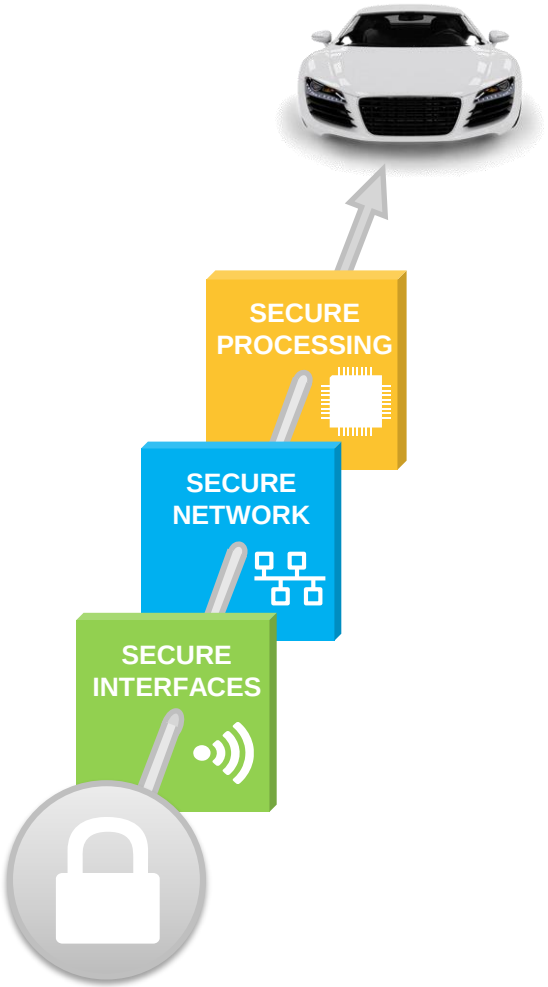
NXP Automotive Vehicle Security Architecture (4 +1 Solution)



- 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

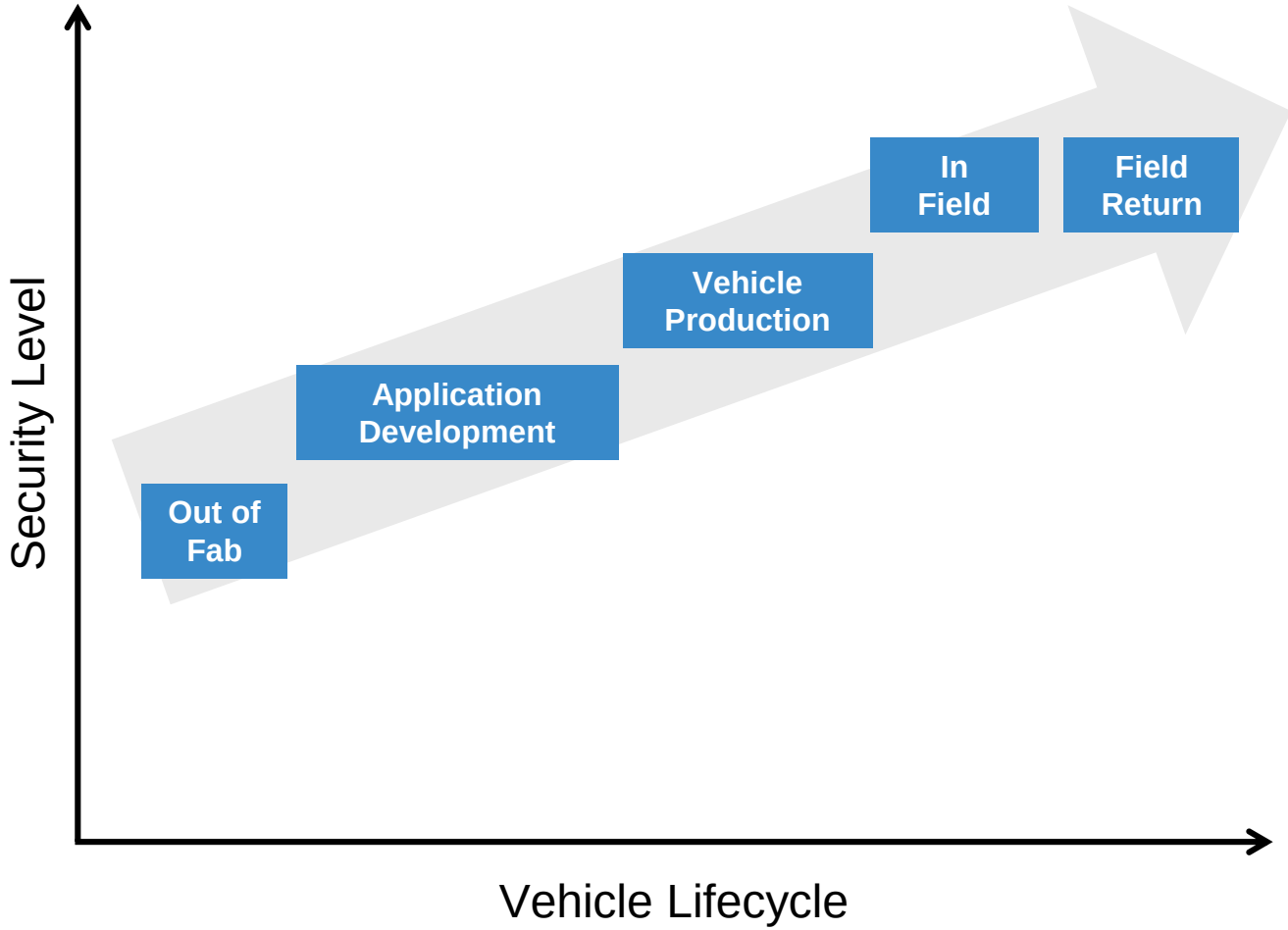
Hardware Security is a Must

- **Crypto accelerators** to guarantee strict performance requirements
  - E.g. V2X message authentication, CAN authentication, secure boot, …
- **Hardware-enforced isolation** to protect against software attacks
  - E.g. system vs. user mode, TrustZone, SHE/HSM, …
- **Tamper-resistant hardware** to protect against advanced, physical attacks
  - E.g. Secure Elements



Security Throughout the Entire Lifecycle

- Increased security level at each stage of the development lifecycle
- Non-reversible, non-revocable
- Enable application development, debugging and failure analysis
- Without compromising security in the production vehicle



Proven History in Driving Automotive Security

Mid 1990s
- Censorship
- Infrastructure

Early 2000s
- Enhanced Censorship
- Infrastructure

Mid 2000s
- High Assurance Boot
- Fault detection sensors

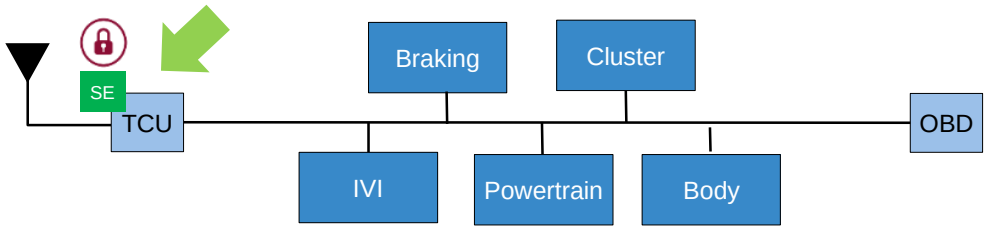
Late 2000s
- Crypto Services Engine (SHE)
- Active shields

2010s +
- Hardware Security Module (HSM)
- Secure Elements (SE)
- Gateway, IVN security

4 Layers to Securing a Car

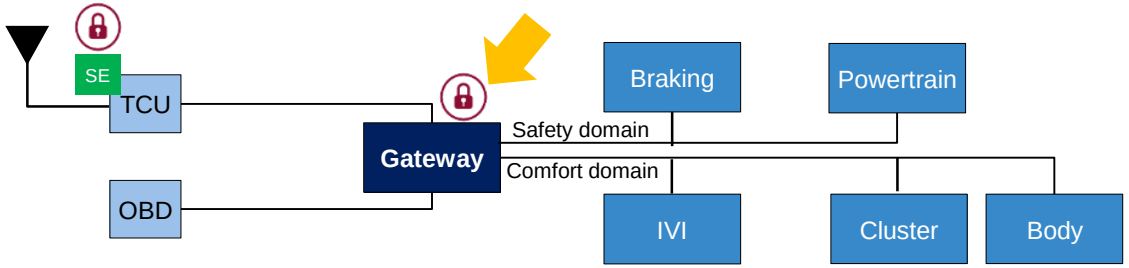
Layer 1: Protect External Interface

Secure M2M authentication, secure key storage



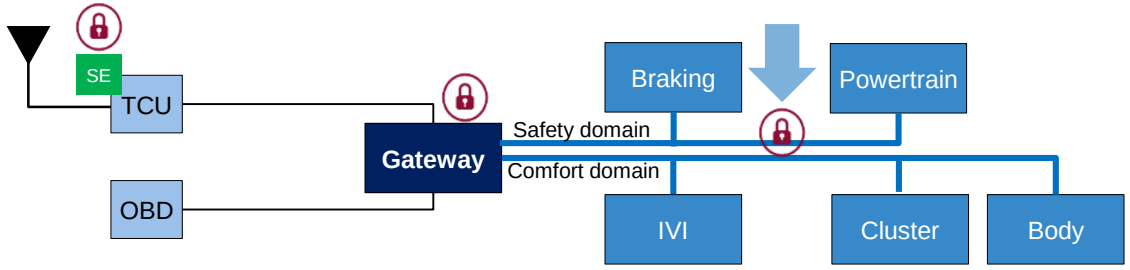
Layer 2: Isolate Network

Domain isolation, firewall/filter, centralized intrusion detection (IDS)



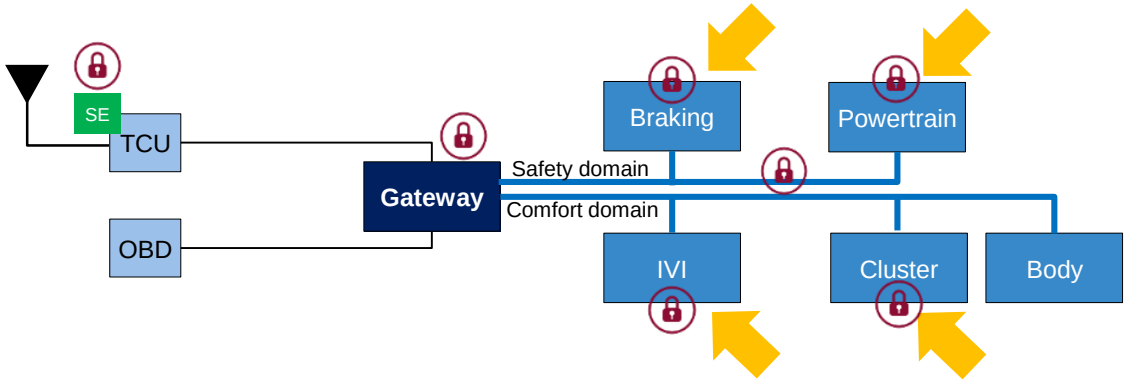
Layer 3: Secure Network

CAN ID Killer, message authentication, distributed intrusion detection (IDS)



Layer 4: Secure Processing

Secure boot, run time integrity, OTA updates



SECURITY USE - CASES & ATTACKS

Security Use Cases

In-Vehicle Security

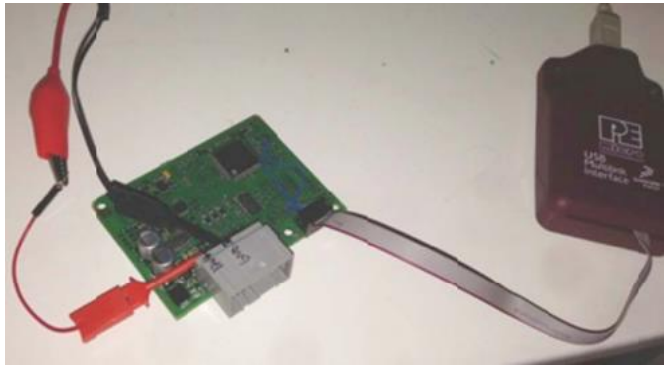
- Immobilizer / Component Protection
- Mileage Protection
- Secure Boot and Chain of Trust
- Secure Communication
- DRM for Batteries

Connected Vehicle Security

- Android application download
- DRM for content download/streaming
- Remote ECU firmware update
- Black-box for due government or insurance
- Car-to-Car communication

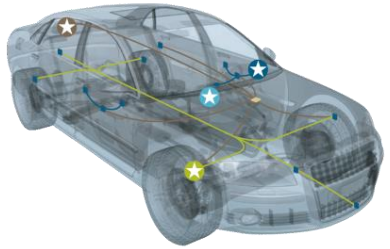
Other Automotive Security Threats

Transportation Department Warns Against Counterfeit Air Bags
October 10, 2012, NHTSA estimates it affects 0.1% of US Fleet, availability of such replacement systems traces back to 2003 (!)



DARPA Funded Researchers Take Control Of Two Vehicles
Using a Macbook connected to the On-Board Diagnostics Port Dr. Charlie Miller and Chris Valasek. July, 2013, Defcon: Adventures in Automotive Networks and Control Units [http://illmatics.com/car_hacking.pdf]

Mileage Manipulation (in Germany)
- 2 million manipulated cars per year
- Average increases in value per car ~3000€
- Total loss 6 billion euro



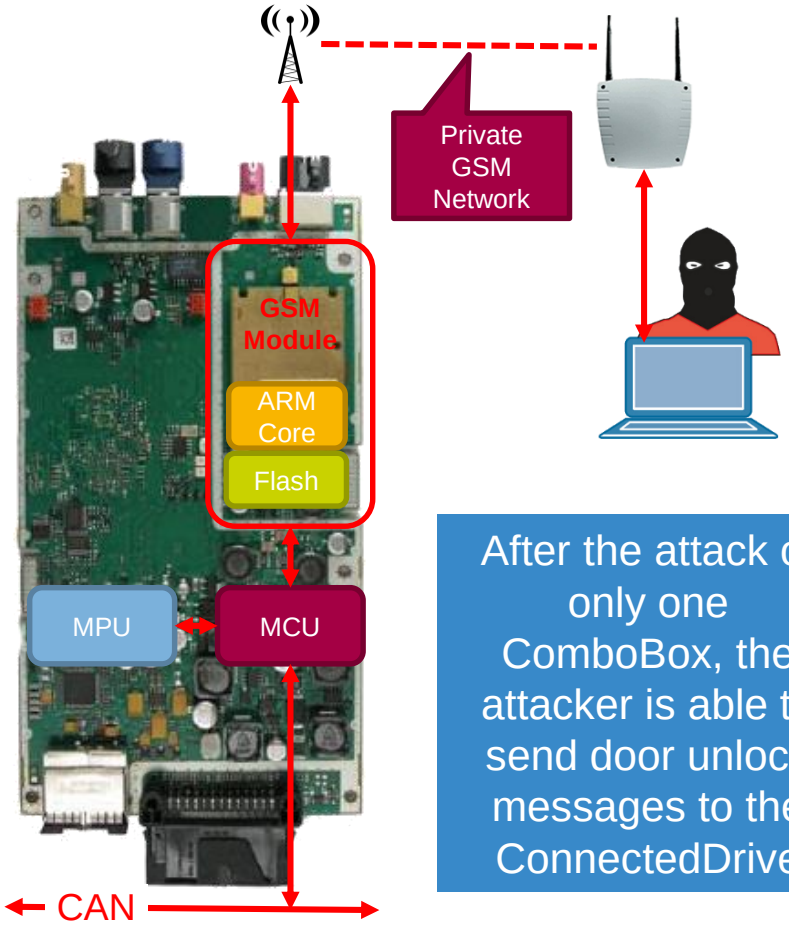
The ConnectedDrive – Unlock the Doors

Issue/Hack:

- No individual keys per car
- Keys stored in readable flash / Firmware readable
- Debug-port active
- Outdated or no encryption on some services
- No integrity check of the device configuration
- No authentication of the counterpart station
- ~ 2.2 million affected cars

Security Requirements:

- Improve key management
- Use existing device features (e.g. disable debug port)
- Crypto modules with:
- Secure key storage
- Actual cipher algorithm (e.g. AES-128) support



After the attack of only one ComboBox, the attacker is able to send door unlock messages to the ConnectedDrive

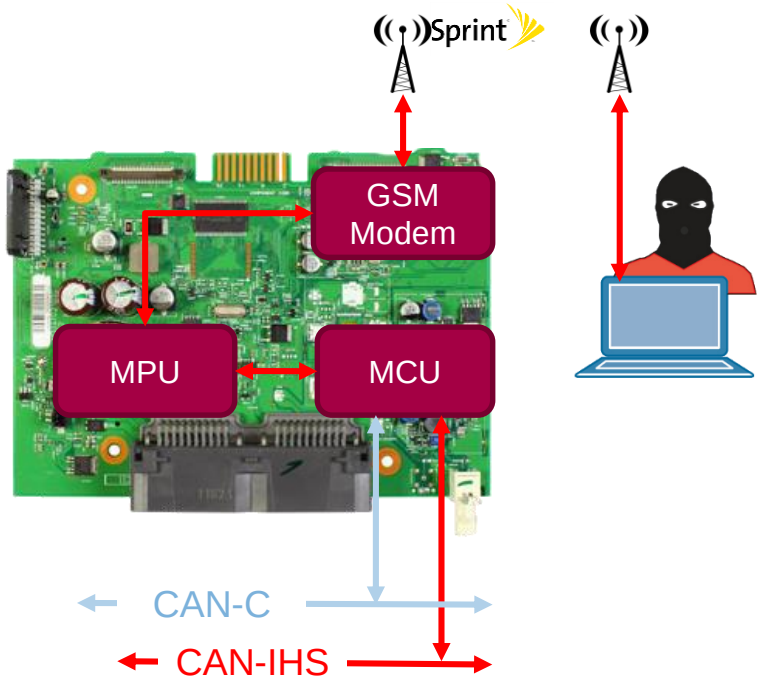
Vehicle – Out of Control

Issue/Hack:

- Radio/Infotainment system is directly connected both CAN busses
- Weak Wi-Fi password system and network configuration (e.g. open D-Bus)
- Weak firmware update process
- Debug-port active
- No secure boot
- Flash content readable
- No encrypted firmware image, no signatures
- OEM has to recall 1.4 Million Cars Over Hacking

Solution:

- Improve network architecture
- Firmware image authentication during update
- Use Secure Boot
- Use Message Authentication for safety relevant messages (e.g. Break / Steering Wheel control)
- Use existing device features (e.g. disable debug port)



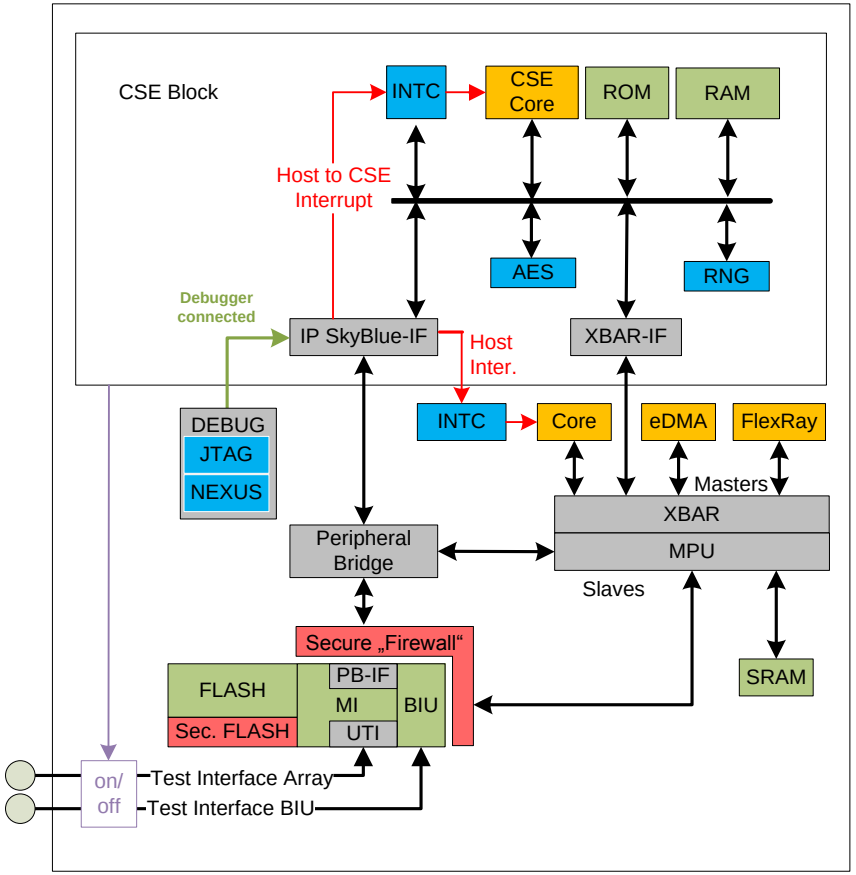
Due several weakness it‘s possible to execute code on the MPU remotely via the GSM network. Additional it‘s possible to modify the MCU firmware and send faked CAN messages via the MCU into the car network.
Finally it was possible to deactivate the breaks remotely!

THE HIS - SHE SPECIFICATION - OVERVIEW

SHE Specification I – Secure Data

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] ECRET_KEY inherits its protection flags from MASTER_ECU_KEY

SHE Specification II – 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

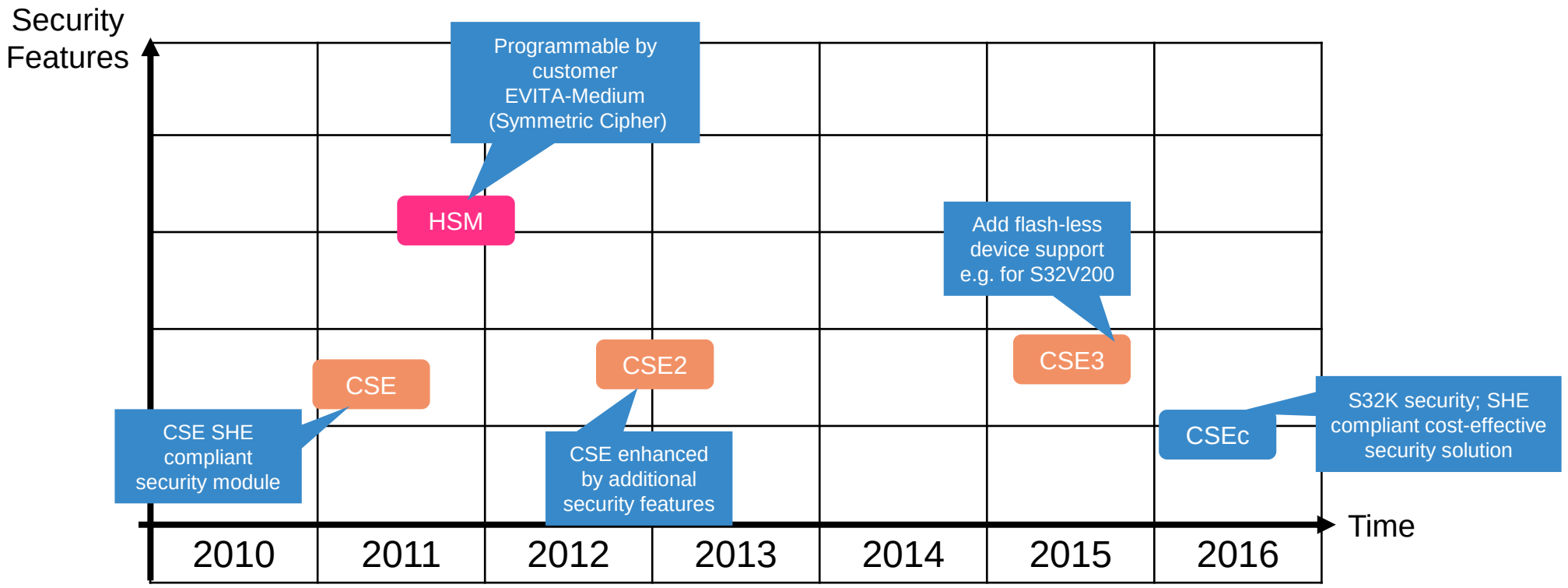
2. Sequential Boot – SHE module comes out of reset and verifies the custom code, after the application core comes out of reset and execute the code (independent of secure boot result!)

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

Note: SHE feature - Autonomous boot-strap is critical!

CRYPTO SERVICE ENGINE (CSE) – OVERVIEW

S32 Freescale Security Modules

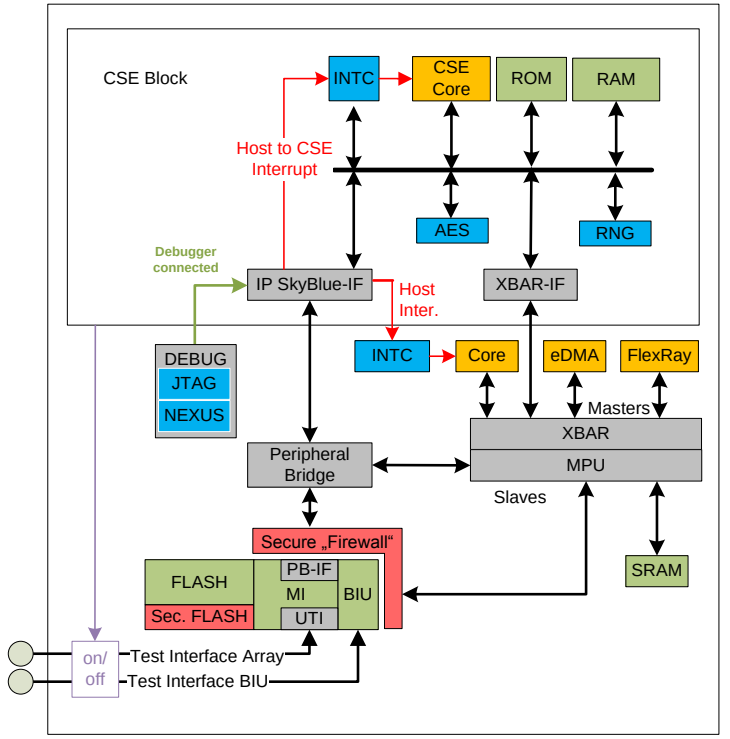


Cryptographic Services Engine (CSE)

The 1st Implemented SHE Spec. Module in the World

The CSE moves the crypto keys from public memory into the secure memory. The applications works only with key references instead with key values. Only the CSE can access the keys, for an key update the actual value must be known.

- **CSE module implements the official HIS SHE-Specification**
- **32-bit secure core working up to 133 MHz**
- **AES-128**
  - Supports the cipher modes: ECB & CBC
  - Throughput of 100 Mbit/sec & 2µs latency per encoding/decoding operation
- **CSE module interfaces:**
  - Crossbar master interface → full memory window access
  - Configuration interface → offers easy to use programming model
- **Unique ID – 120bit**
- **Secure Boot support**
- **Secure flash blocks are fix assigned to the CSE module. No other masters can access these memory**
- **PRNG seed generation via TRNG**



CSE2 Enhancements

To improve and support safety & secure communication better these enhancements are implemented in cooperation with US car OEMs:

- Introduction of a new security flag per GPR-Keys
  The flag defines the communication role (sender or receiver) of the ECU. Only the sender is able to publish an CMAC value. Receiver are only able to verify the CMAC.

- Double the number of general purpose keys (from 10 to 20)
  More keys allows more communication domains; each with his own individual key!

- Secure Boot result stored in NVM (configurable by customer)
  The secure boot result is storable in the flash memory, for this it's re-useable after leaving power down modes. Otherwise secure-boot sensitive keys are not useable anymore.

- Reset Generation on Secure Boot Fail (configurable by customer)
  For high-security ECUs its useful to stay in reset, if secure boot fails.

- SHE-Compression (Miyaguchi-Preneel construction) available via API

juergen.frank@freescale.com

CSE3 Enhancements – SHE for Flash-less Devices

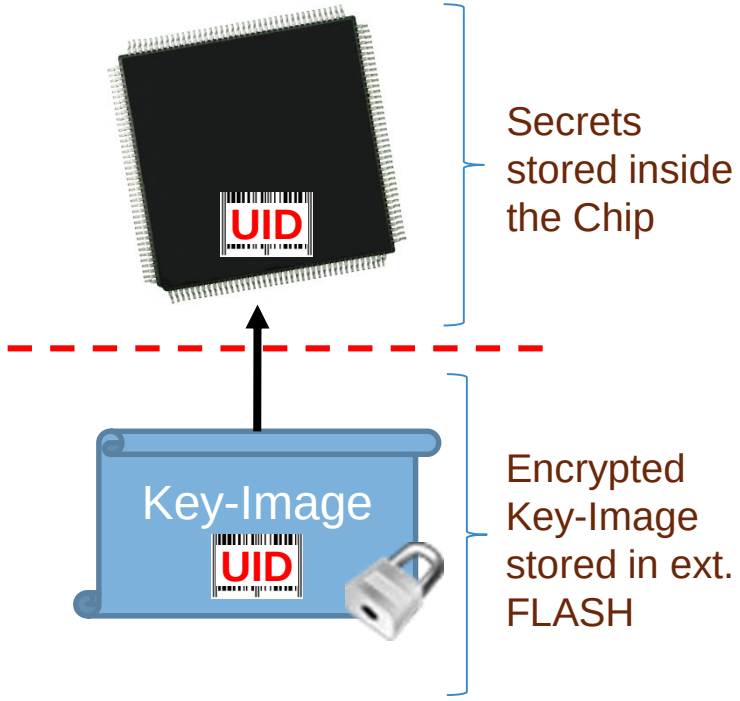
CSE3 offers an alternative security key storage solution and brings the SHE Spec. API to flash-less devices without loss of security[1]:

- CSE3 is an enhanced version of the CSE2

  Uses the same API and functions, so existing software drivers are easy portable to the new architecture.

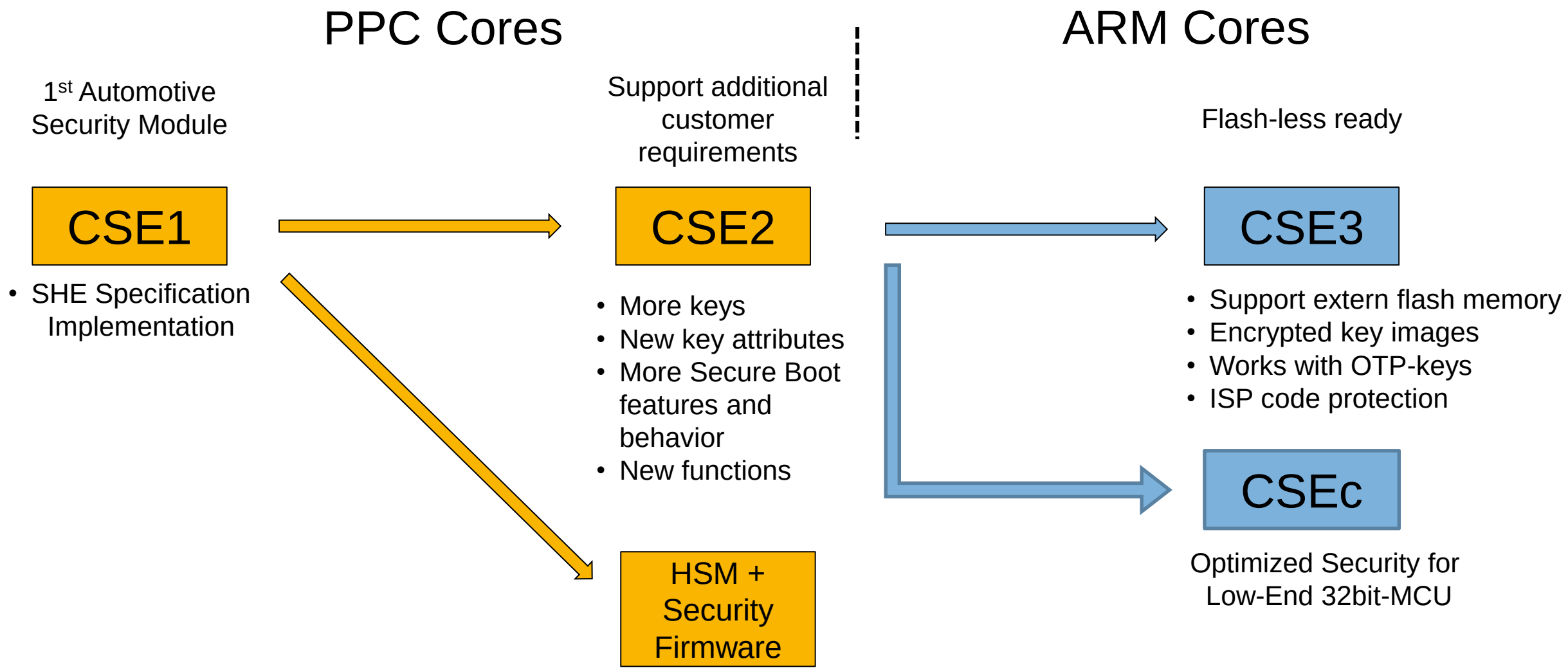
- CSE3 handles the main keys in an encrypted key file
  The key-file could be stored in any NVM memory, the encryption key for the key-file is stored in a fuse array on the device. The file is protected against cloning and re-play attacks and is bound to the device.

- 1st implementation of the CSE3 in S32V2



[1]The SHE Specification requires device internal flash

CSE – The Evolution Steps

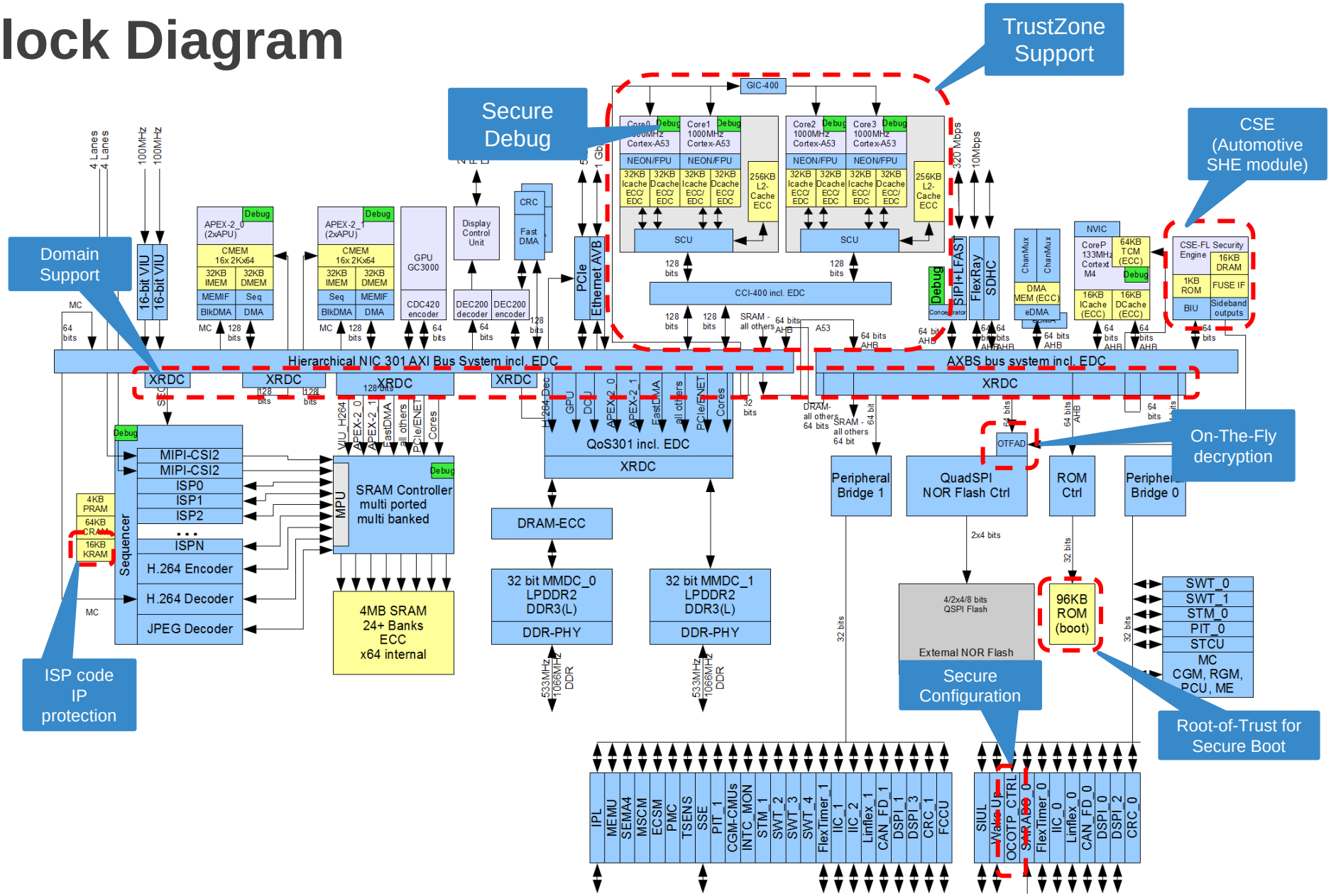


HSM Security Firmware – A SHE Implementation on HSM

- Release 1.0 is available for MPC574xG (3M & 6M)
- Firmware implements the CSE2 feature set
  (SHE firmware + Global-B requirements) on the HSM
- Firmware „emulates“ the CSE register interface, to simplify porting of existing SW stacks
- Firmware is delivered pre-programed in the device
  - No SHE firmware programming and DCF configuration required by customer

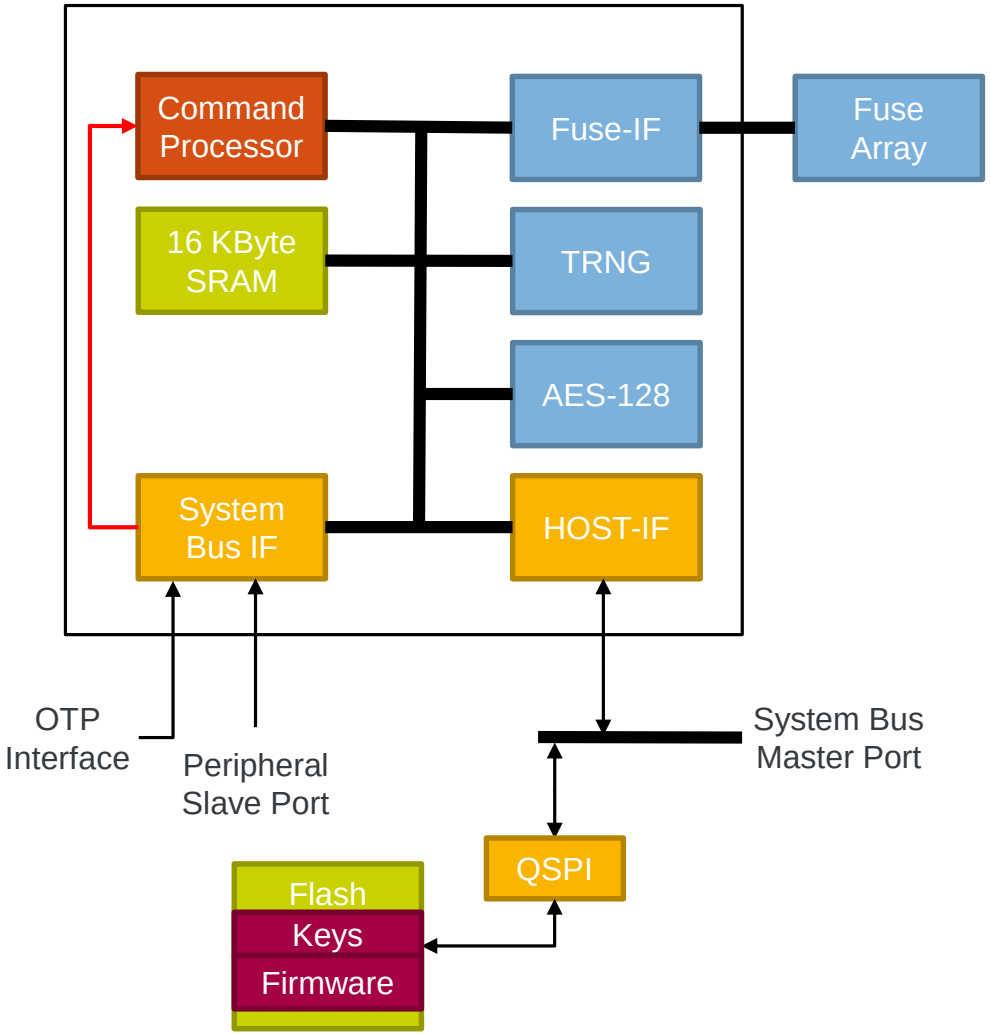
CSE3 ON FLASH - LESS DEVICES

S32V Block Diagram



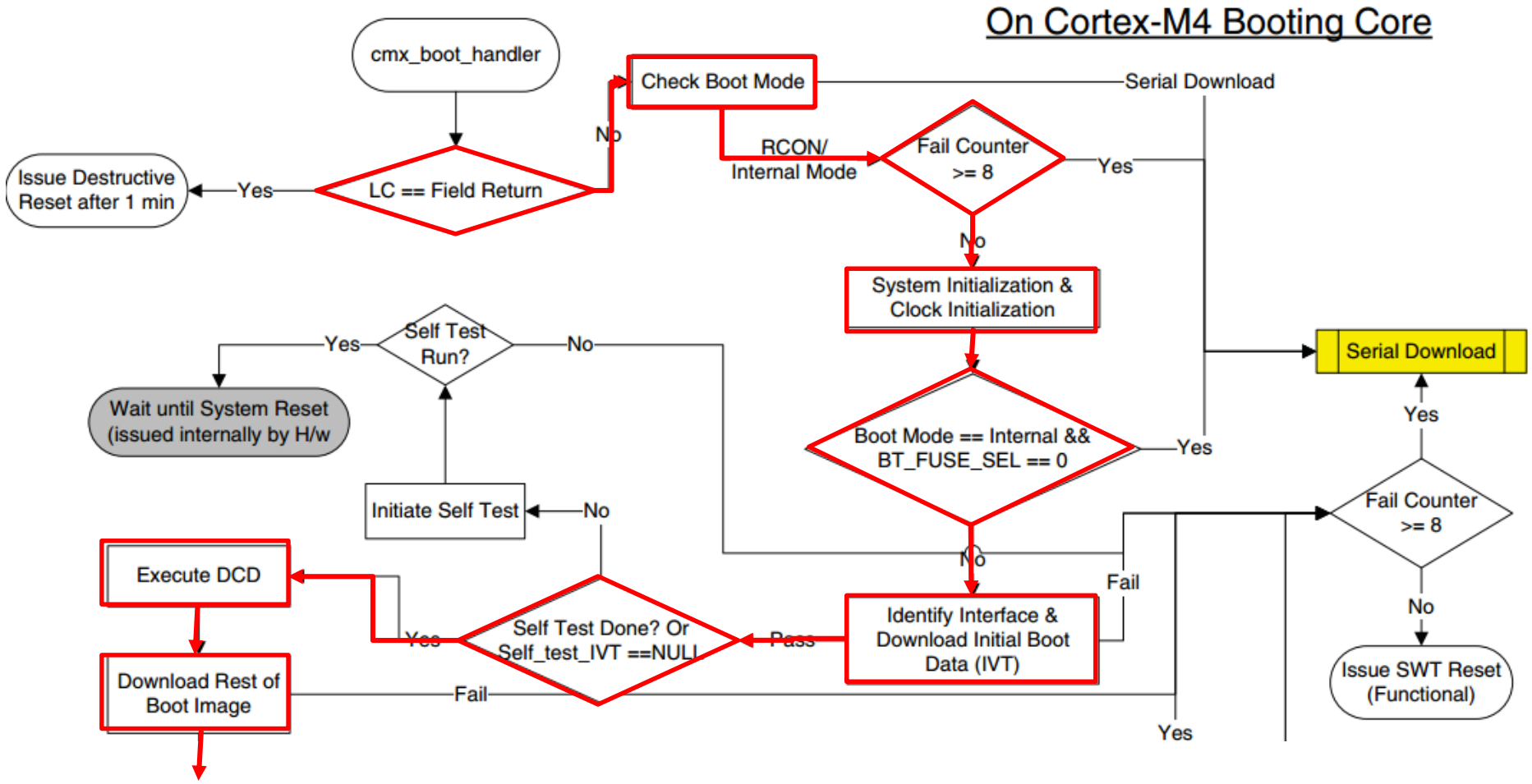
Cryptographic Services Engine v3 (CSE3)

- **CSE module implements main elements of the HIS SHE-Specification**
- **32-bit secure core working at 133 MHz**
- **AES-128**
  - Supported crypto modes: ECB & CBC
  - Throughput ~100 Mbit/sec
  - Latency 2µs per one encoding/decoding ops
- **CSE module interfaces:**
  - System master interface
  - Configuration interface
- **PRNG seed generation via TRNG**
- **CSE Core not programmable by customer**
- **Encrypted code and key image are stored in external memory**
- **Several tempter detect inputs and a tamper detect enable**
- **CSE3 offers the same security features as CSE2**

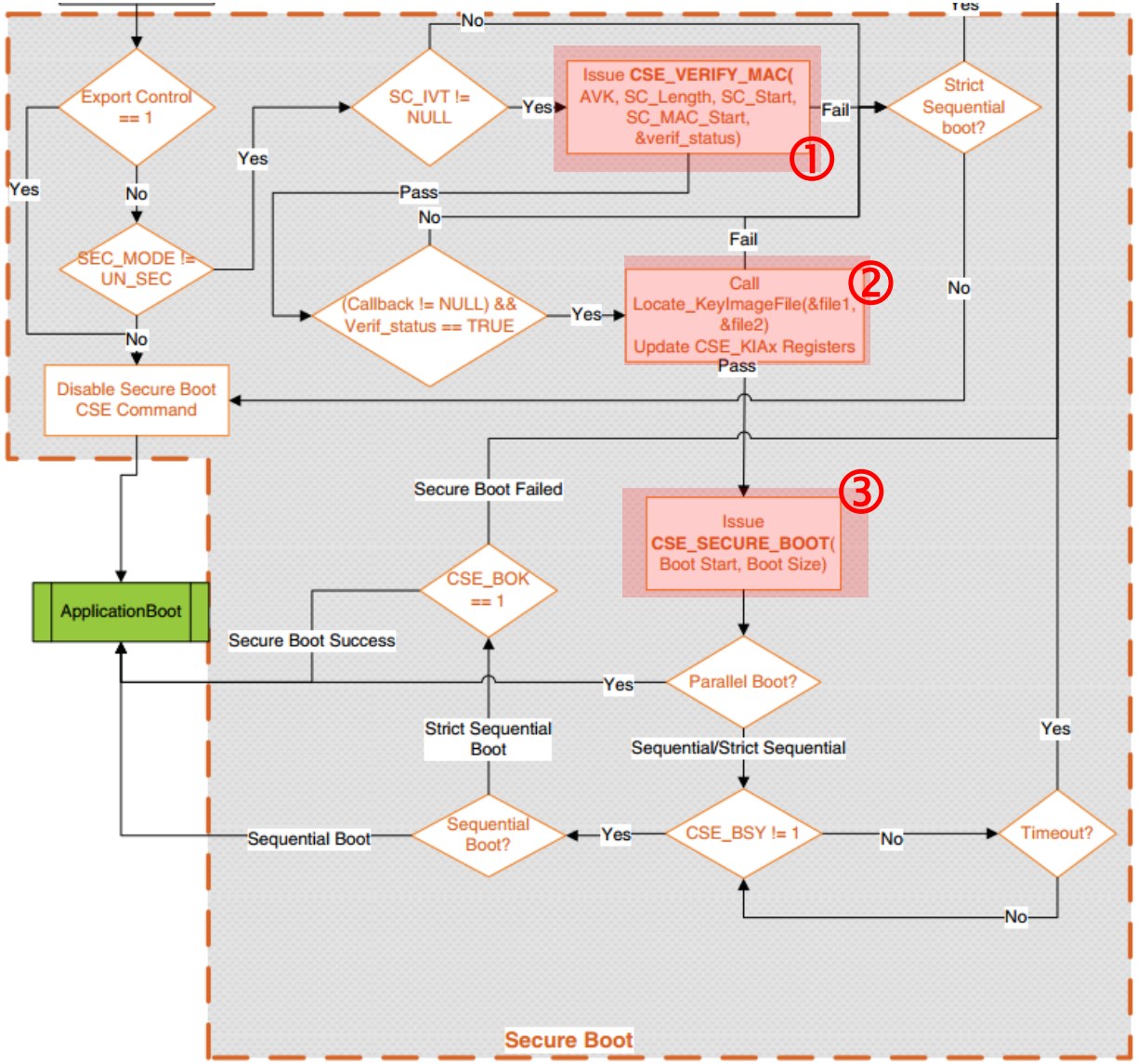


juergen.frank@freescale.com

Boot Flow I – Controlled by ROM Code



Boot Flow II – Controlled by ROM Code



1 Verify & Encode Secure Callback Image - CSE offers limit functionality

2 Call Callback function to determine Key Image location

3 CSE full functional – all SHE keys loaded
➔ Secure Boot possible

CSE Register Interface

Register	Usage
Control Register	Configuration information
Status Register	Status information
Interrupt Register	Command Complete Interrupt Flag
Error Code Register	Contains any error information returned by latest command
CSE TRNG Control Register	TRNG Test Mode on/off
Command Register	Commands are written here (after the Command Parameter registers have been written)
Command Parameter 1-5 Register	General purpose registers whose meaning depends upon command being executed. Often used to point to addresses of input and output data
CSE Key Image Address 0 / 1 Register	Must be initialized with addresses of the latest key images before either the SECURE_BOOT or INIT_CSE commands are issued (e.g. via secure call back functions)
CSE OTP Status Register	Secure Counter Value (up to 42) from OTP memory
CSE Publish Count	Holds the publish counter value of the loaded key image

CSE2 Register Set (Autosar driver available)

CSE3 Additional Register Set

One Time Programmable (OTP) Memory

OTP Field	Size	System Access	Programm by	Usage
Secret Key	128	---	Freescale; programmed and locked during device production	SHE-Secret Key, random number
IAK	128	---		ISP code protection key
UID	120	R		SHE UID value

Flash Key	128	W	Customer; programmed and locked	if (Secure Counter > 1)
				  for encryption of the CSE image files
				if (Secure Counter == 0)
				  use 0 for image encryption

Secure Counter	128	R / W	Customer; programmed	Prevent rollback-attacks (re-usage of old images)

Encrypted CSE Images

- Key Image 1 KByte
- Firmware Image 12 KByte
- The key image is encrypted and authenticated by the CSE using keys derived from the Flash Key and Secret Key. A Secure Counter value of zero indicates the Flash Key is not programmed and a default value of zero is used for the Flash Key.

Steps to prepare the device:

1. Program the Flash Key.
2. Issue the INIT_RNG command.
3. Issue the PUBLISH_KEY_IMG command with the secure counter input parameter set to one.
4. Store the generated key image to non-volatile memory.
5. Program the OTP Secure Counter to one.

Key Image

- Size: 1KByte
- Using the AES standard big endian convention
- Generated by the PUBLISH_KEY_IMG
- Image Address is determine by system boot ROM code

Index	Field	Size (bytes)	Format	Description
a[0], a[1]	Header Mark	2	0,1	0x4B49
a[2]	Reserved	1	0,1	0x00
a[3]	Format	1	0,1	Key image format, 0x00 or 0x01
a[4] ... a[7]	Firmware Address	4	0,1	CSE firmware start address
a[8] ... a[23]	Firmware CMAC	16	1	CMAC covering the firmware image
a[24] ... a[39]	Key CMAC	16	1	CMAC covering the rest of the key image (984 bytes)
a[40] ... a[54]	UID	15	1	Unique ID
a[55]	Reserved	1	1	
a[56] ... a[59]	Publish Counter	4	1	Count of publish operations (unsigned)
a[60] ... a[62]	Reserved	3	1	
a[63]	Secure Counter	1	1	Secure counter (6-bit unsigned)
a[64] ... a[79]	Key IV	16	1	Key slot encryption IV
a[80] ... a[95]	Reserved	16	1	
a[96] ... a[1023]	29 Key Slots (encrypted)	960	1	29 Key Slots (encrypted)

CSE3 Commands

CMD	Command	Description
0x01	ENC_ECB	Encrypt ECB
0x02	ENC_CBC	Encrypt CBC
0x03	DEC_ECB	Decrypt ECB
0x04	DEC_CBC	Decrypt CBC
0x05	GENERATE_MAC	Generate MAC
0x06	VERIFY_MAC	Verify MAC
0x07	LOAD_KEY	Load Key
0x08	LOAD_PLAIN_KEY	Load Plain Key
0x09	EXPORT_RAM_KEY	Export RAM Key
0x0A	INIT_RNG	Initialize RNG
0x0B	EXTEND_SEED	Extend PRNG Seed
0x0C	RND	Generate Random Number
0x0D	SECURE_BOOT	Secure Boot

CMD	Command	Description
0x0E	BOOT_FAILURE	Boot Failure
0x0F	BOOT_OK	Boot OK
0x10	GET_ID	Get ID
0x11	CANCEL	Cancel
0x12	DEBUG_CHAL	Debug Challenge
0x13	DEBUG_AUTH	Debug Authorization
0x14	TRNG_RND	Generate TRNG Random Number
0x15	INIT_CSE	Initialize CSE
0x16	MP_COMPRESS	Miyaguchi-Preneel (MP) Compression
0x17	PUBLISH_KEY_IMG	Publish Key Image
0x18	LOAD_SEC_RAM	Load Secure SRAM (not the CSE SRAM!!)
0x19	OPEN_SEC_RAM	Open Secure SRAM (not the CSE SRAM!!)
0x1A	EXPORT_SEC_RAM	Export Secure SRAM (not the CSE SRAM!!)

new commands of CSE3

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

- NXP overs since years innovative automotive security solutions

- NXP offers security solutions for all 32bit-MCU segments
  - Low – CSEc
  - Mid – CSE
  - High – HSM with CSE firmware

NXP Security Solution for Automotive MCU			
	Device	Platform	Module
MCU (internal flash)	MPC564xB/C	PowerPC e200	CSE
	MPC5746M / MPC5777M		HSMv1
	MPC5748G / MPC5746C		HSMv2
	MPC5777C		CSE2
	Radar MCU		CSE2
	MAC57D54H	ARM Cortex-A5/M4	CSE2
MPU (flash-less)	S32V243	ARM Cortex-Ax/Mx & ARM9/11	CSE3 / OTFAD/ TrustZone
	VFxxx		Trust Zone + CAAM
	i.Mx		

SECURE CONNECTIONS
FOR A SMARTER WORLD

ATTRIBUTION STATEMENT