Keeping Cars Secure: Solutions for Implementing Security in the Era of the Connected Vehicle

APF-AUT-T0112

Jürgen Frank
Automotive Sr. Systems Engineer

September 2013

Agenda

- **Introduction – Security, why?**
  - **Use-case overview**
  - **Attack examples**
- **In a nutshell: Security Algorithms**
- **Automotive Standards**
- **Freescale Qorivva Security Modules**
  - **CSE & CSE2**
  - **HSM**
- **Use-Cases**
  - **Secure Boot**

Security Use Cases: In Vehicle Security

- **Immobilizer / Component Protection**
  - Detection of replacement or modification of components or ECUs
- **Mileage Protection**
  - Disabling mileage manipulation
- **Secure Boot and Chain of Trust**
  - Verification of authenticity and integrity of vehicle software
- **Secure Communication**
  - Protection of the vehicle network from unauthorized access

Security Use Cases: Connected Vehicle Security

- **Telematics and GPS**
  - Authentication of map data
  - Enforcing permission limitations between open and safety critical domains
- **Android application download**
  - Ensuring permissions of apps are not exceeded
- **DRM for content download/streaming**
  - Authentication that content is licensed
- **Remote ECU firmware update**
  - Enforcing permission limitations between open and safety critical domains

Automotive Security Threats I

Research Study 1: **Experimental Security Analysis of a Modern Automobile**,
Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage
IEEE Symposium on Security and Privacy, Oakland, CA, May 16–19, 2010
Center For Automotive Embedded System Security
[http://www.autosec.org/pubs/cars-oakland2010.pdf]

Research Study 2: **Security and Privacy Vulnerabilities of In-Car Wireless Networks:**
A Tire Pressure Monitoring System Case Study
Ishtiaq Rouf, Rob Miller, Hossen Mustafa, Travis Taylor, Sangho Oh, Wenyuan Xu, Marco Gruteser, Wade Trappe, and Ivan Seskar
in Proceedings of the 19th **USENIX Security** Symposium, Washington DC, Aug. 11-13, 2010
[http://www.cse.sc.edu/~wyxu/papers/TPMS2010.pdf]

Mileage manipulation (in Germany)

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

Automotive Security Threats II

- 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 (!)

Video created & owned by National Highway Traffic Safety Administration (NHTSA) [http://www.nhtsa.gov/]

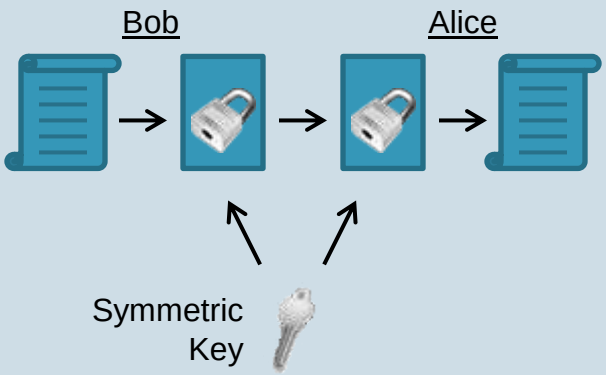
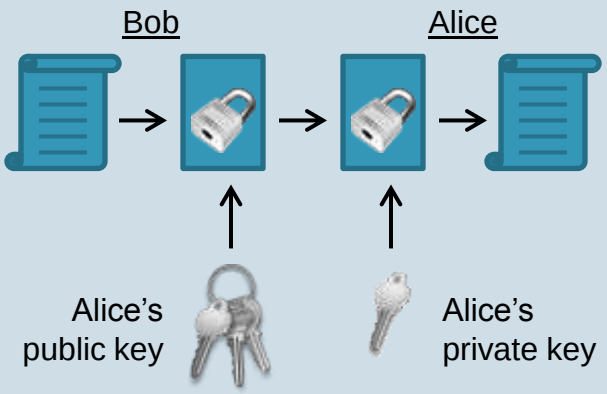
- DARPA Funded Researchers Take Control Of two Cars
- Dr. Charlie Miller and Chris Valasek.
- July, 2013, Defcon: Adventures in Automotive Networks and Control Units [http://illmatics.com/car_hacking.pdf]

In a Nutshell: Security Algorithms

Cryptographic Algorithms

- **Symmetric**
  - DES, AES
  - Cipher Modes
- **Asymmetric**
  - RSA, ECC
- **Secure Hashes**
  - SHA-1 and CMAC

Cryptography – Types of Algorithms

	Symmetric	Asymmetric
Scheme	One key for encoding and decoding 	Key pair (public/non-public) encoding and decoding 
Pro	→ Compact implementation → High performance → Key length (<512-bits)	→ No key exchange problem → Supports verification
Contra	→ Key exchange problem	→ Long calculation time → No message broadcasting
Algorithm	DES, AES	RSA, ECC

Symmetric Ciphers – DES & AES

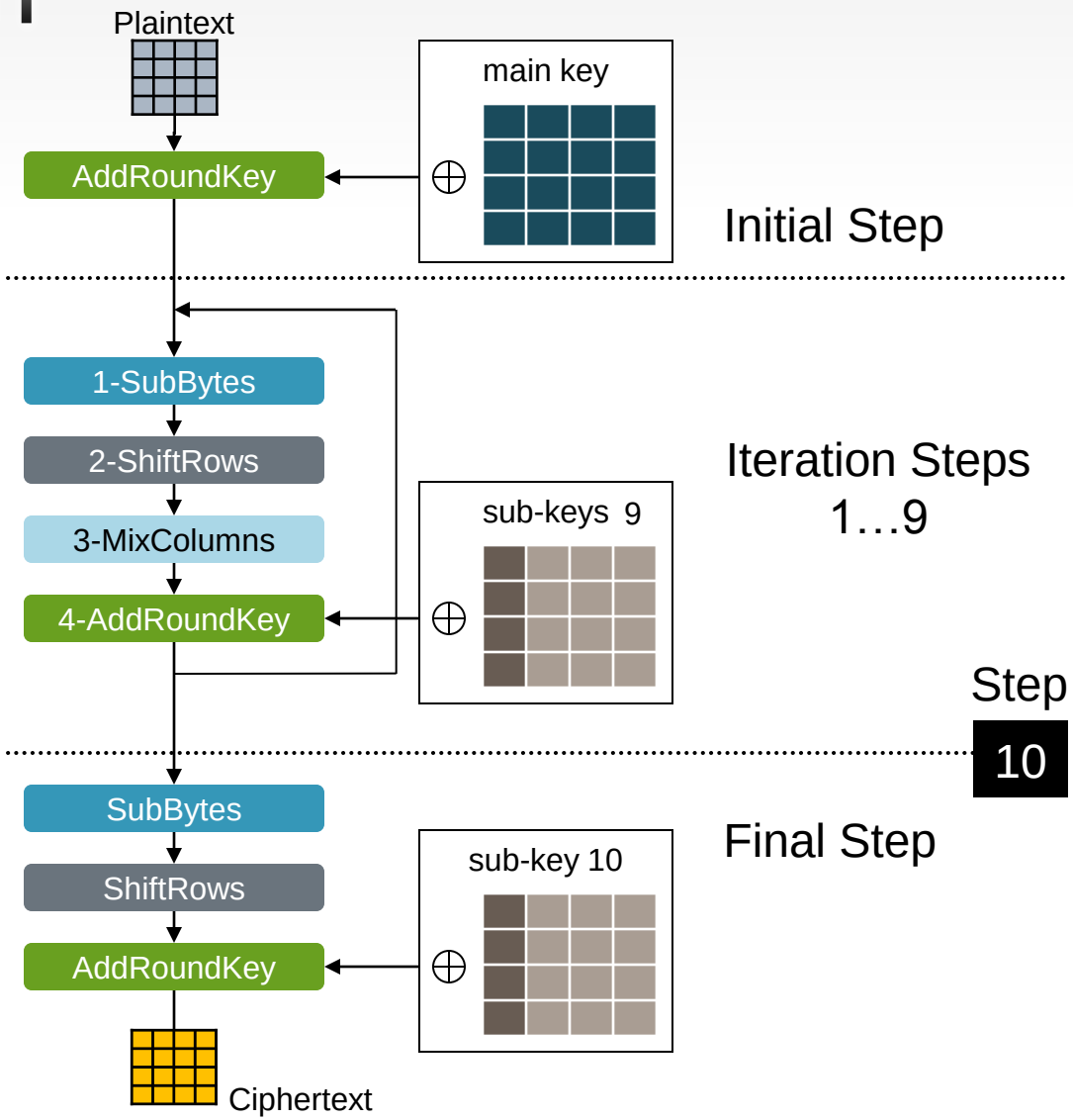
	DES	AES
Developed by	IBM	Vincent Rijmen Joan Daemen
Standardized since	1977	2002
Block size [bits]	64	128
Key size [bits]	64 (reduced to 56 bits)	128, 192 or 256
Rounds	16	10,12 or 14

DES is now considered insecure for many applications!

The AES Algorithm I

The main encoding transformations are:

- 1-SubBytes
- 2-ShiftRows
- 3-MixColumns
- 4-AddRoundKey



The AES Algorithm II

1-SubBytes

input:

19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

output:

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

x = high-nibble
y = low-nibble

hex		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	1	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	5	9a	7	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	0	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	2	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	6	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	8
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	3	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

S-Box

The AES Algorithm III

2-ShiftRows

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

← rotation by 1 byte
← rotation by 2 byte
← rotation by 3 byte

⇒

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

3-MixColumns

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

●

d4
bf
5d
30

=

04
66
81
e5

⇒

04	e0	48	28
66	cb	f8	06
81	19	d3	26
e5	9a	7a	4c

Every column will be modulo multiplied with a given matrix.

4-AddRoundKey

04	e0	48	28
66	cb	f8	06
81	19	d3	26
e5	9a	7a	4c

⊕

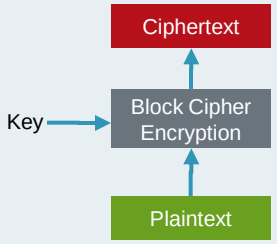
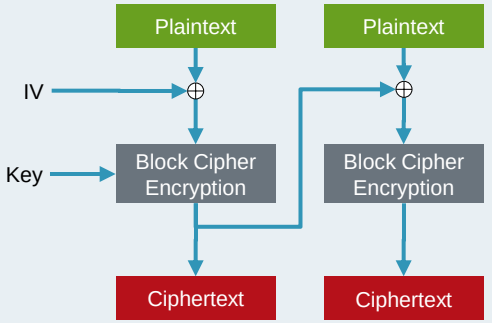
a0	88	23	2a
fa	54	a3	6c
fe	2c	39	76
17	b1	39	05

⇒

a4	68	6b	02
9c	9f	5b	6a
7f	35	ea	50
f2	2b	43	49

Every column will be xor-ed with the same column of the actual iteration sub-key.

Cipher Modes – Electronic Codebook (ECB) and Cipher-block Chaining (CBC)

	Electronic codebook (ECB)	Cipher-block chaining (CBC)
Scheme	Each block is encoded/decoded independantly from the others	Previous result is XORed with actual plaintext
Diagram		
Pro	Random access possible	Secure for messages longer than block size
Contra	Insecure for message longer than the block size (statistical analysis)	No random access possible, (before the last block can be decoded all others must be decoded)
Example		

RSA – Public Key Algorithm

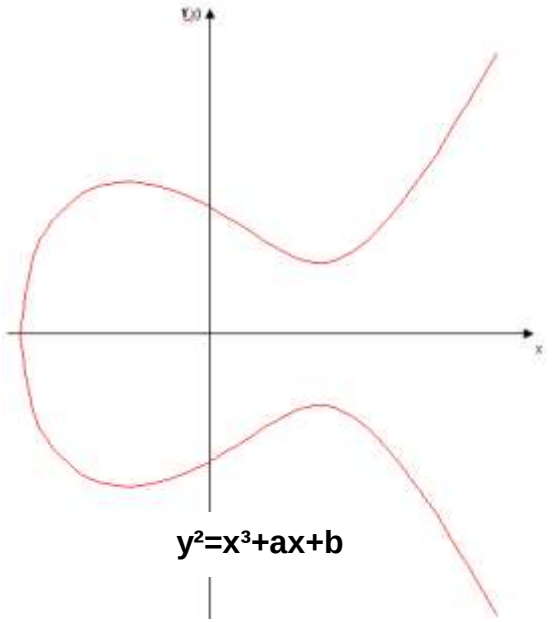
- **Facts**
  - Developed in1978 by Ron Rivest, Adi Shamir and Leonard Adleman at MIT
  - Key size: ≥ 1024-bits (CAN-Message offers 8 data bytes)
  - Algorithm based on very big numbers → optimized libraries (e.g. GMP)
  - Expectations: RSA-1024 will be insecure by 2014
- **The RSA algorithm involves three steps**
  - Key generation
  - Encryption
  - Decryption

Elliptic Curve Cryptography (ECC)

- 1985: Victor Miller and Neil Koblitz use elliptic curves for cryptography
- It's an alternative asymmetric crypto algorithm
- ECC relies upon the difficulty of the Elliptic Curve Discrete Logarithm Problem (ECDLP)

$$a^x \equiv m \bmod p$$

  - it's easy to calculate m, when a, x and p is known
  - it's hard to find x when a, m and p is known
- ECC offers smaller key size than RSA



$y^2=x^3+ax+b$

Security Level(*) Restricted → Confidential →Secret→Top secret	NIST Recommended Key Sizes [bits]		
	AES	RSA	ECC
Secret	128	3072	256
Top Secret	192	7680	384
	256	15360	512

Cipher Summary

	DES	AES	RSA	ECC
Secure for the next few years	✗	✓	✓ Key size > 2048	✓
Type	symmetric	symmetric	asymmetric	asymmetric
Typical key size [bits]	56	128, 192, 256	1024, 2048, 3072	180, 224, 256,320, 512
Execution time	short	short	long	long
Authentication / verification	✗	✗	✓	✓
Implementation	HW – good SW – lot of bit ops	HW / SW - good	Could combine into one module, req. big number math functions (e.g. GMP)	
Comments	Message broadcasting isn‘t an issue		Message broadcasting is an issue	

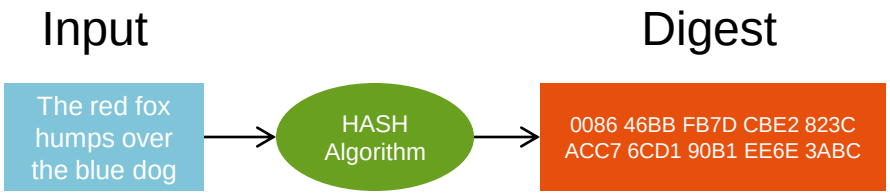
Secure Hashes

SHA and CMAC

Hash Algorithms vs. CMACs

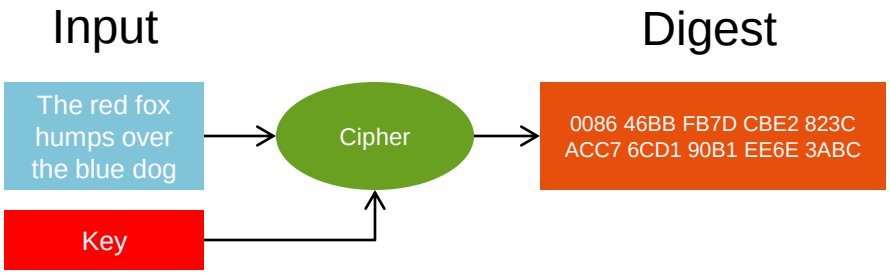
Hashe:

- Integrity check
- Public known algorithm
- Input: data blocks
- Output: HASH value
  (e.g 256-512bits)

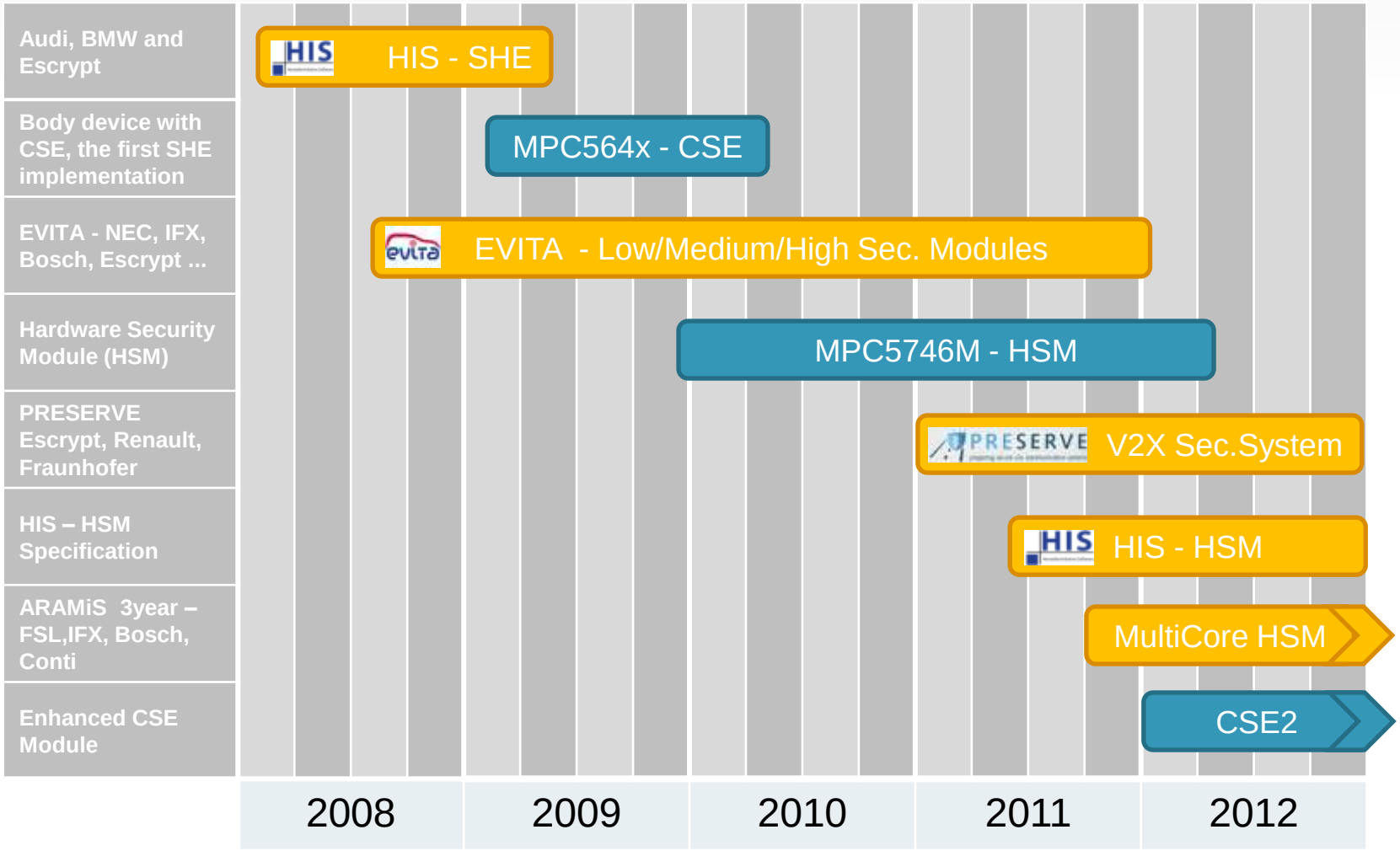


CMAC:

- Integrity & authenticity check
- Public known algorithm
- Input: data blocks & key value
- Output: CMAC value
  (e.g 128-256 bits)



Automotive Security Standards and Projects



juergen.frank@freescale.com

HIS – SHE Specification

- 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 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

juergen.frank@freescale.com

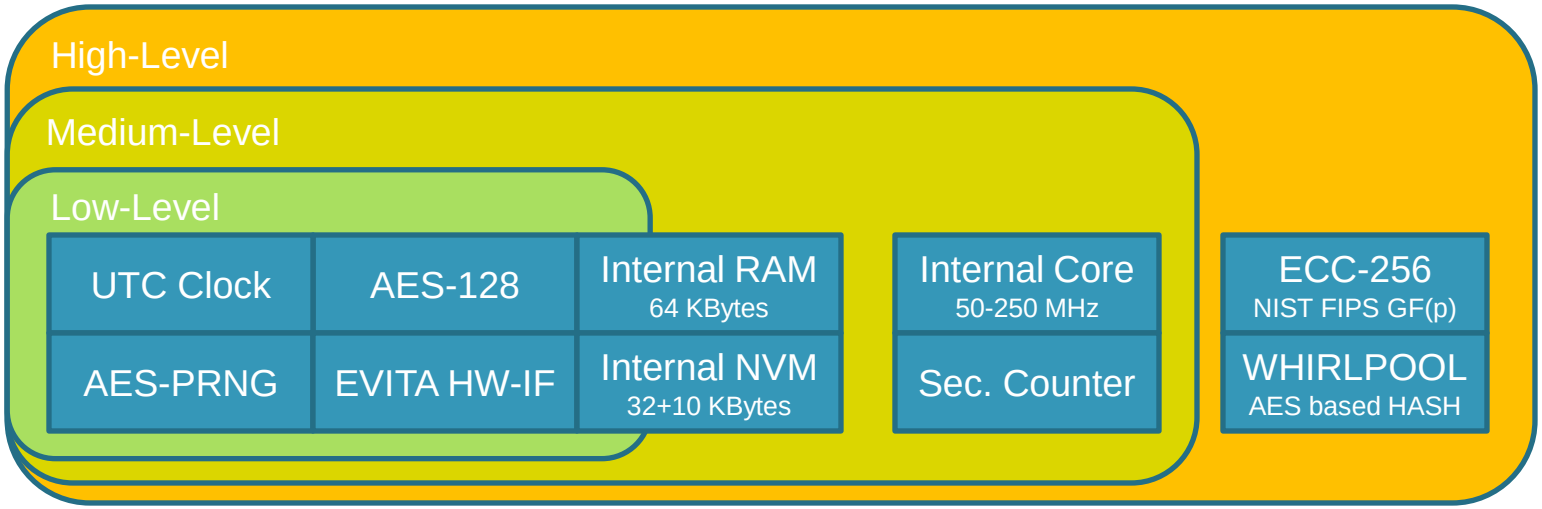
Evita

http://www.evita-project.org

EVITA a project co-funded by the European Union.

The objective of EVITA is to design, verify, and prototype an architecture for automotive on-board networks where security-relevant components are protected against tampering and sensitive data are protected against compromise.

EVITA Security Modules



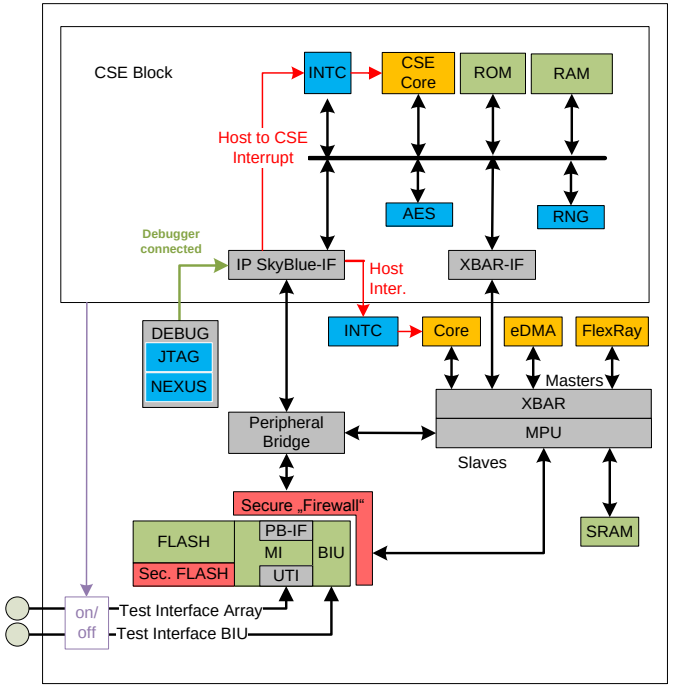
juergen.frank@freescale.com

32-bit Qorivva Controllers

Security Hardware Features

Cryptographic Services Engine (CSE)
Qorivva MPC564xB/C

- **CSE module implements the official HIS SHE-Specification**
- **32-bit secure core working at 120 MHz**
- **AES-128**
  - Supported crypto modes: ECB & CBC
  - Throughput 100 Mbit/sec
  - Latency 2µs per one encoding/decoding ops
- **CSE module interfaces:**
  - Crossbar master interface
  - Configuration interface
- **Secure flash blocks assigned to the CSE module. Accesses from other masters are impossible.**
- **PRNG seed generation via TRNG**
- **CSE Core not programmable by customer**



juergen.frank@freescale.com

CSE2 Enhancements to CSE

In future designs CSE2 will only be used

- **Introduce new security flag per GPR-keys**
- **Increased number of GPR-keys from 10 to 20**
- **Secure Boot result storage in NVM (configurable by customer)**
- **Reset Generation on Secure Boot Fail (configurable by customer)**

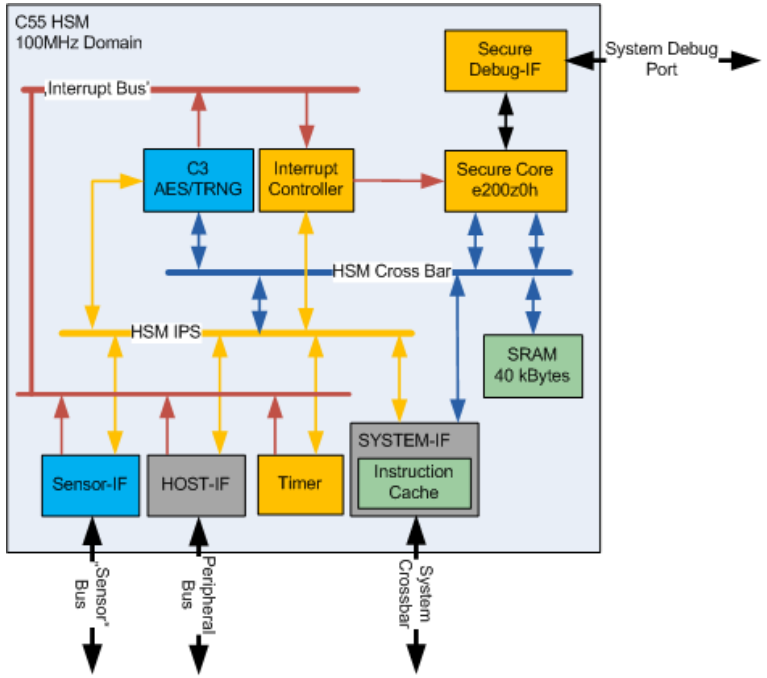
juergen.frank@freescale.com

Hardware Security Module (HSM)
v1: MPC5746M / MPC5777M & v2: MPC5748G

HSM is free programmable by the customer,
additional security algorithm could implemented in software

Features:

- **e200z0h core (v1: 100MHz / v2: 80 MHz)**
- **4Kbytes Instruction cache**
- **Secure Debugger Interface**
- **Cryptographic Modules with AES-128, Random Number Generator, DMA**
- **Sensor Interface – monitor for voltage, temperature and clock (v1 only)**
- **Memory**
  - SRAM (v1: 40 Kbytes / v2: 32 Kbytes)
  - Flash
    code: 2 x 64 Kbytes + 1 x 16KBytes
    data : 2 x 16 Kbytes



juergen.frank@freescale.com

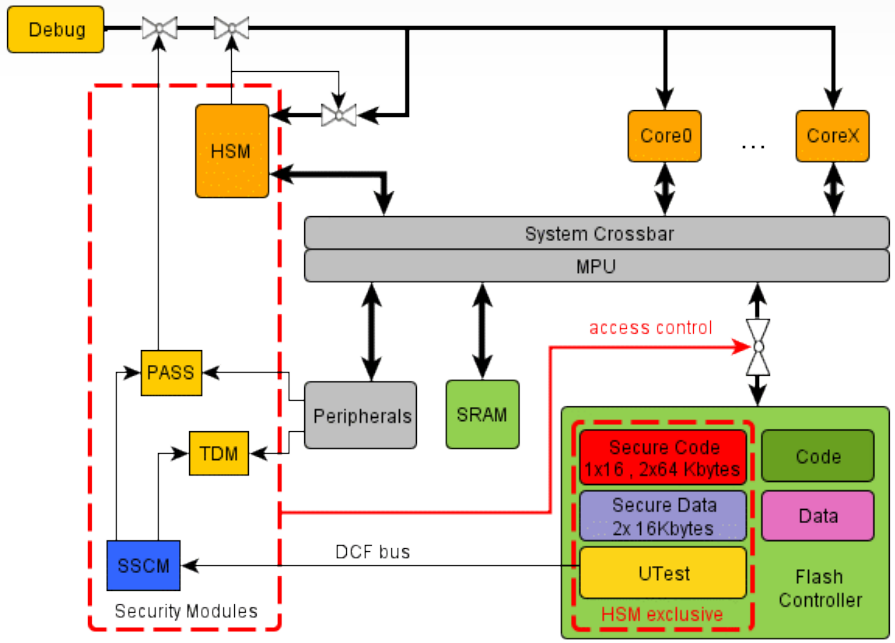
C3 – AES-128 Features

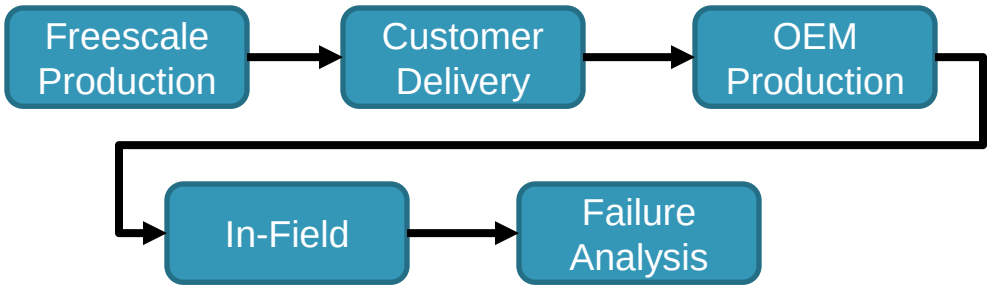
- Cryptographic keys
  - 2 x read-only keys (programmed by FSL)
  - 8 x GPR-keys
  - 1 x PRNG-key
- Cipher Modes: ECB, CBC, OFB, CFB, CTR, XTS, CMAC, GCM and OFB for PRNG
- Multiple contexts; context swapping performed by hardware
- Programming interfaces:
  - Register based mode
  - SmartDMA mode

Qorivva HSM Security Architecture

Features:

- Device life cycle scheme
- Unique ID for each device
- Debugger restrictions
- Flash Protection
  - OTP
  - read / write & erase
  - diary to log erasing-steps





SSCM: System Status Configuration Module
PASS: Password And Device Security Module
TDM: Tamper Detection Module
HSM: Hardware Security Module
MPU: Memory Protection Unit
DCF: Device Configuration Format

Freescale Devices with Security

Freescale Security Solution for Automotive products				
	Device	Platform	Module	
MCU (internal flash)	MPC564xB/C	PowerPC e200	CSE	Automotive
	MPC5746M		HSMv1	
	MPC5777M		HSMv1	
	MPC5748G		HSMv2	
MPU (flash-less)	Vybrid Controller Solutions	ARM Cortex-Ax/Mx & ARM9/11	Trust Zone + Sahara / CAAM	Consumer
	i.Mx 2x / 3x / 5x / 6x / 7x			

no automotive standards

juergen.frank@freescale.com

CSE, HSM and the Security Standards

Security Standards	EVITA- Low	HIS-SHE	EVITA-Medium (HIS-Medium)	EVITA-High
Main features	UID Crypto engine	NVM is mandatory Fix function set	Programmable by customer	Public Key HASH
CSE / CSE2	supported by MPC564xB/C			
HSM (v1/v2)	supported by MPC5746M & Calypso			

juergen.frank@freescale.com

32-bit Qorivva Controllers

Use Cases

Secure Boot

Check Boot Loader for Integrity and Authenticity

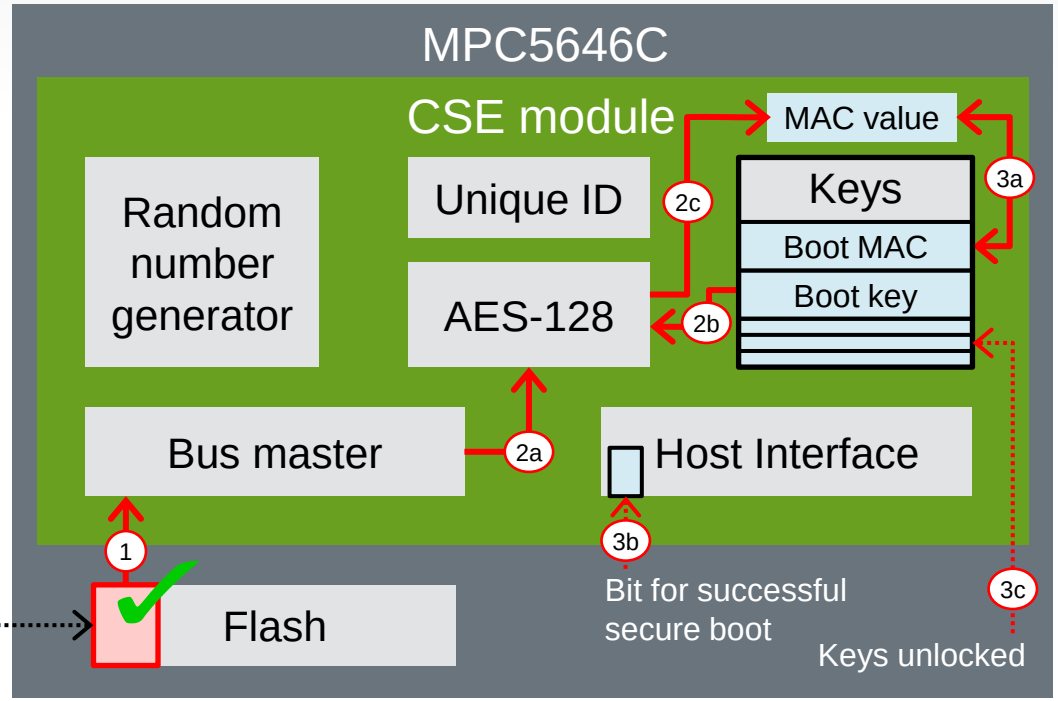
Step 1: After power on: CSE module reads bootloader via its bus master interface.

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.

Bootloader:
Part of flash memory
4 Start 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)

juergen.frank@freescale.com

Chain of Trust

Check parts of Flash Memory for Integrity and Authenticity

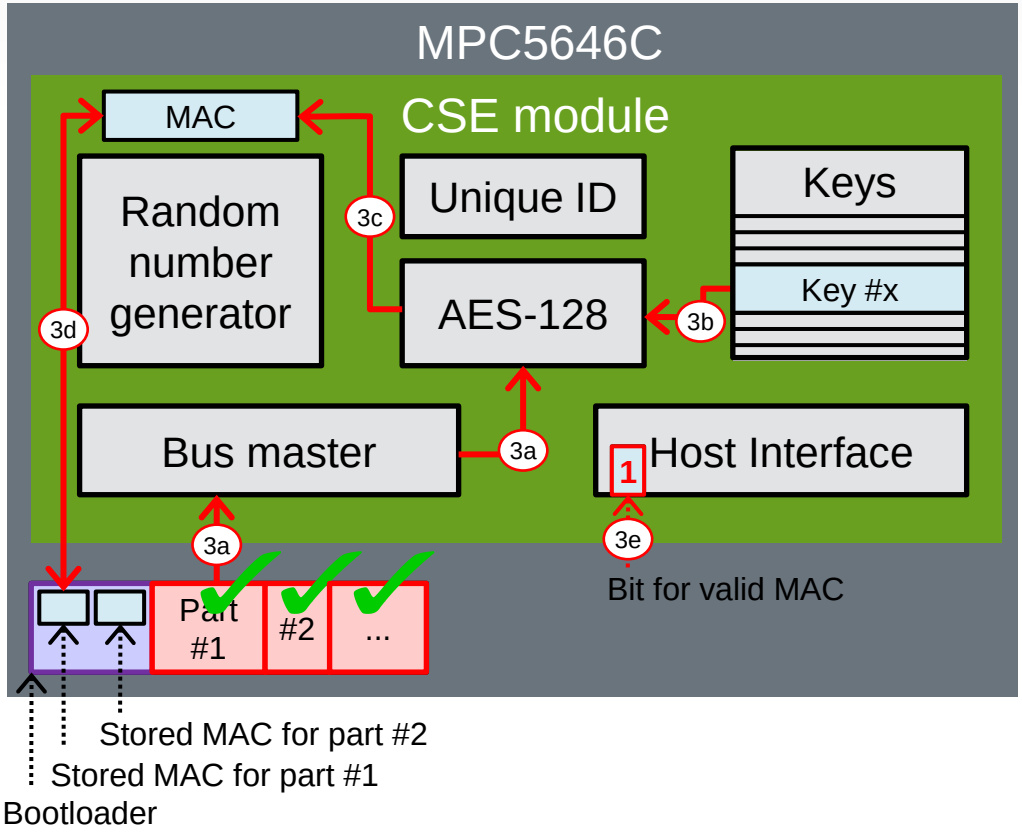
Step 1: Successful secure boot to verify bootloader and to unlock keys. Then the bootloader configures the MCU for full speed, best memory timing, etc.

Step 2: Bootloader asks CSE module to verify MAC for part #1 of flash memory using key #x

Step 3: CSE module reads part of flash, uses key #x to calculate MAC, and compares calculated MAC with MAC for part #1 as stored in bootloader. If identical, CSE module sets corresponding bit in host interface.

Step 4: Bootloader checks bit.
If set: Part #1 of flash ok → execute part #1.

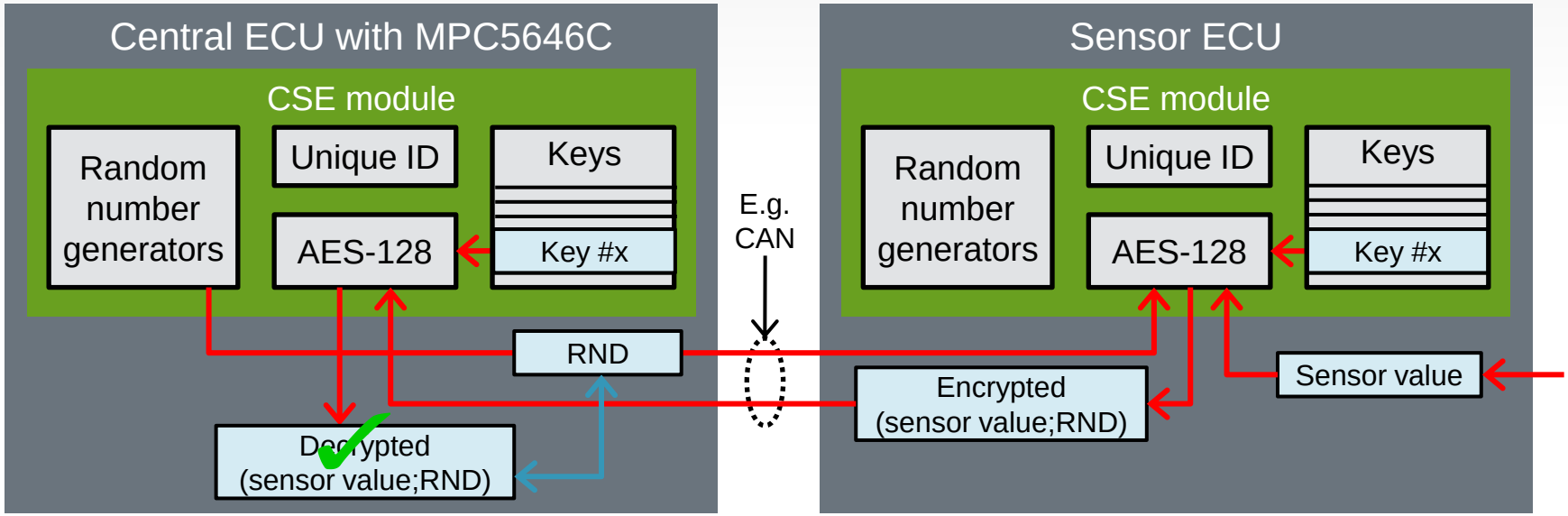
Step 5 etc: Similar to bootloader vs. part #1 of flash: Part #n of flash verifies part #n+1 → chain of trust.



- MACs stored in bootloader provide integrity and authenticity of the related parts in flash memory.
- Bootloader protected by secure boot (see previous demo).
- Part-by-part checking of flash to execute critical parts of flash (e.g., MCU configuration/IRQ table) as soon as possible.

juergen.frank@freescale.com

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).

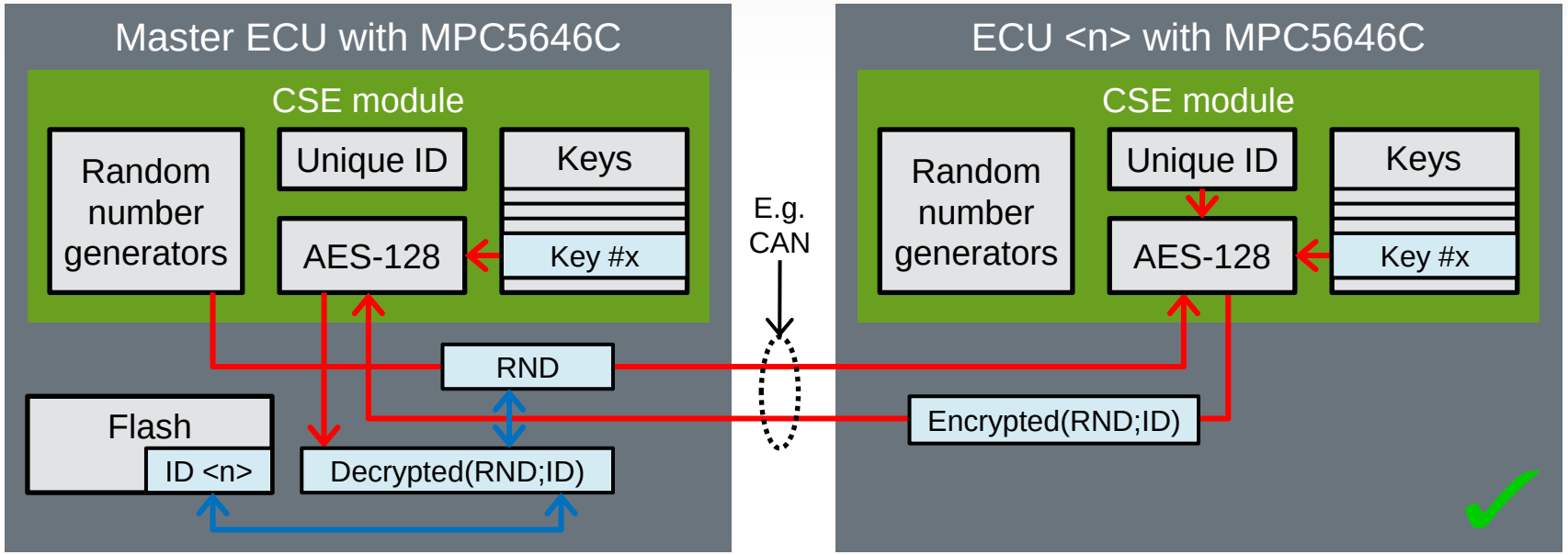
Step 5: Central ECU checks sent random number vs. received/decrypted random number.

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

juergen.frank@freescale.com

Component Protection

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



Step 1: Master ECU obtains random number and sends it to ECU <n>.

Step 2: ECU <n> appends its unique ID to received RND, encrypts this message with key #x, and sends encrypted message to master ECU

Step 3: Master ECU decrypts received message with key #x.

Step 4: Master ECU checks decrypted RND and ID with sent RND and with stored ID <n>. If match: ECU <n> is ok.

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

Summary

Summary

- Automotive Security: Protection and Enablement
- Automotive Security: A global concern, a requirement and a new trend.
- Two types: In-vehicle Security and Connected Vehicle Security
- Freescale offers a variety of solutions today

Freescale Security Solution for Automotive products			
	Device	Platform	Module
MCU (internal flash)	MPC564xB/C	PowerPC e200	CSE
	MPC5746M		HSMv1
	MPC5777M		HSMv1
	MPC5748G		HSMv2
MPU (flash-less)	Vybrid Controller Solutions	ARM Cortex-Ax/Mx & ARM9/11	Trust Zone + Sahara / CAAM
	i.Mx 2x / 3x / 5x / 6x / 7x		

Contact Information

- **Jürgen Frank**
  - Senior Systems Engineer, FSL Munich
  - Phone: +49 - 89 - 92 103 429
  - Juergen.Frank@freescale.com


freescale™