FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Keeping Cars Secure
Solutions for Implementing Security in the Era of the Connected Vehicle
FTF-AUT-F0112

Johnny Chen
Sr. Automotive Field Application Engineer
Steven Pan
Sr. MAD SOC platform Engineer

August 2012

Agenda

- Introduction
  - Security Status
  - Automotive Security concerns
- In a nutshell: Security Algorithms
  - Crypto Algorithms
  - Hashes and CMACs
- 32bit Qorriva Controllers
  - Security Hardware features
  - Use Cases
- i.MX Application Processors
  - Security Hardware features
  - Use Cases
- Summary

Security Status 1

- HIS-SHE specification
  - Version 1.1 published April 2010
  - Freescale implements the first SHE module (CSE) 2011
  - SHE specification now part of HIS (*)

- Evita project
  - Funded by the EU, finished November 2011
  - Specification of 3 different module classes
    - Evita Full: V2X communication, H/W based asymmetric cryptography
    - Evita Medium (HSM): Multiple-purpose ECUs, H/W based symmetric cryptography
    - Evita Light: Sensors and actuators, cost optimized symmetric crypto H/W, secure NVM optional
  - Gaining global acceptance as “quasi standard”

(*) Herstellerinitiative Software

Security Status 2

- Tier1 specification of security module HSM
  - First samples 2010/11
- HIS is working on a specification of a medium-level security module
  - Re-use of Evita Medium and Tier1 results
  - Expect Evita Medium/HSM to become part of HIS spec in 2012

- Evita follow-on project

- PRESERVE (Preparing Secure Vehicle-to-x Communication Systems)
  - Mission: Design, integrate and test a secure and scalable V2X Security Subsystem for FOTs and Pilot Deployments
  - Based on Evita Full module
  - Jan 2011 - Dec 2014

Automotive Security concerns

- **Safety and reliability**
  - Various publications highlighted how vehicle networks were hacked from inside the cabin. With the connected car, the attacker could be anywhere.
  - Software must be secure, especially in safety relevant systems.
  - Cloned parts and ECUs can impact vehicle reliability and safety.
  - Vehicle networks must be safeguarded against consumer electronics.
- **Financial assets protection**
  - Feature activation through software downloads and enablement.
  - DRM for multimedia content download and distribution.
  - Low insurance rates through effective car theft prevention and component protection.
  - Protection from mileage manipulation.
- **Privacy and Confidentiality**
  - Car data, personal driver data and preferences, location, etc. must remain invisible and untraceable.

Security: Necessity or Feature ?

- What needs to be protected?
- What types of attack can be expected?
- What are the attack motivations and methods?

- How much security do we really want?
- How much are we willing to pay for it?

- What is the impact on system complexity?
- How can the security system be maintained and upgraded over time?

Security Use Cases
In vehicle security

- **Secure Boot and Chain of Trust**
  - Verification of authenticity and integrity of vehicle software
- **Secure Communication**
  - Protection of the vehicle network from unauthorized access
- **Mileage Protection**
  - Disabling mileage manipulation
- **Component Protection**
  - Detection of replacement or modification of components or ECUs
- **Feature Activation**
  - Enabling businesses and after-sales car features

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
  - Establishing a secure boot chain
- **DRM for content download/streaming**
  - Authentication that content is licensed
- **Remote ECU firmware update**
  - Enforcing permission limitations between open and safety critical domains

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

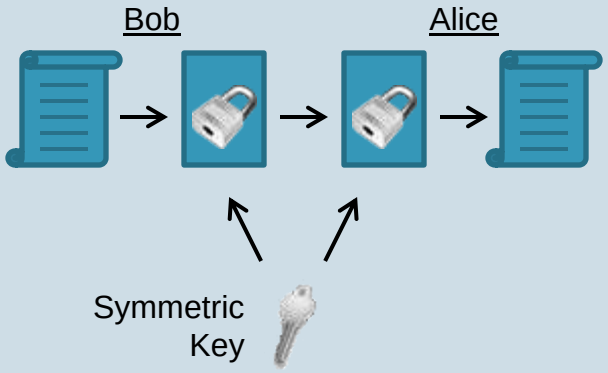
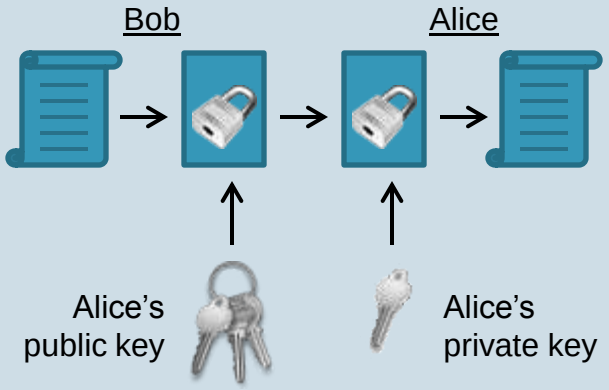
In a nutshell: Security Algorithms

Johnny Chen
Sr. Automotive Field Application Engineer

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 lenght (<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

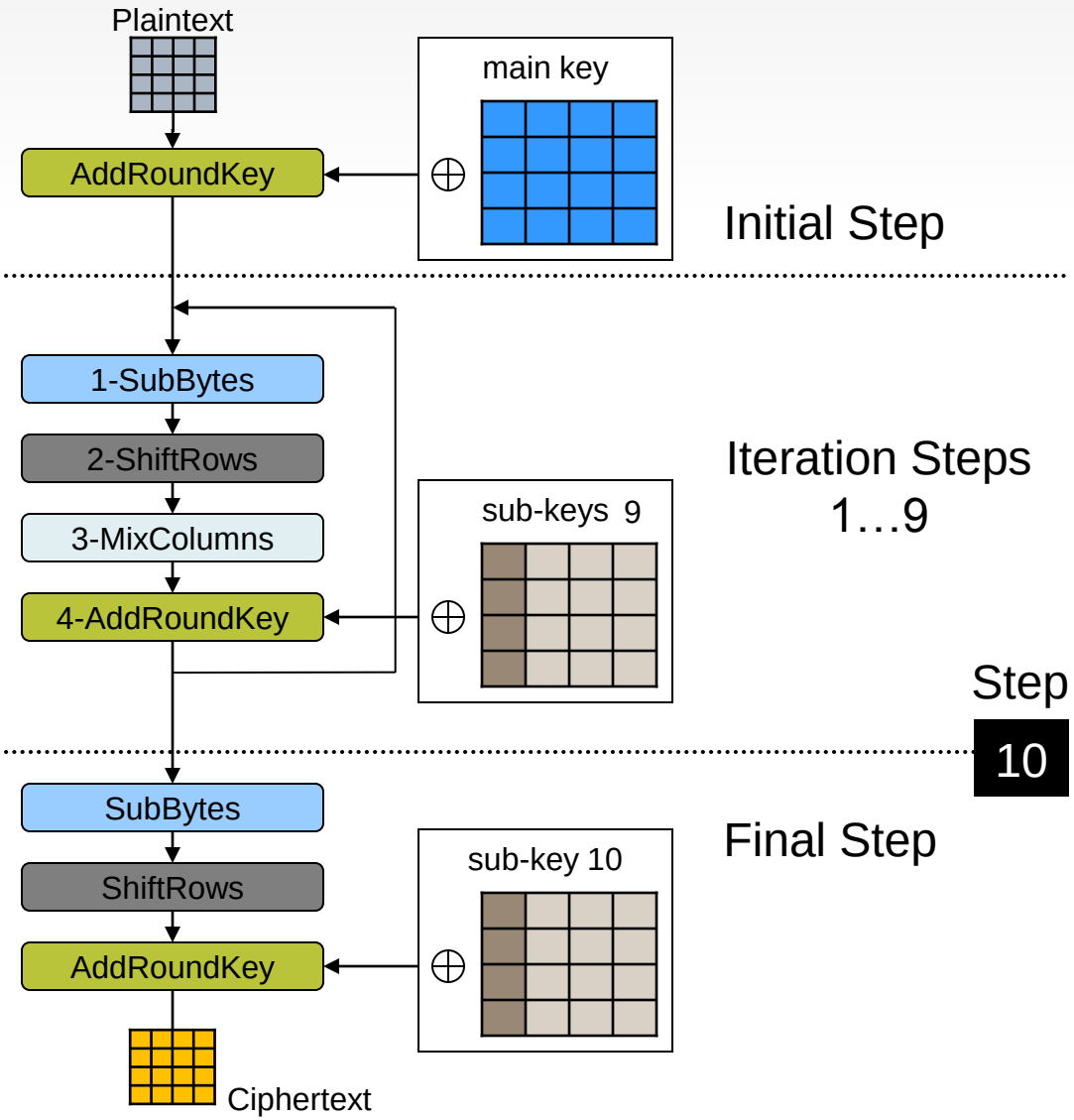
	DES	AES
Developed by	IBM	Vincent Rijmen Joan Daemen
Standardized since	1977	2002
Block size [bits]	64	128
Key size [bits]	64 (reduced to 56 bits)	128, 192 or 256
Rounds	16	10,12 or 14

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

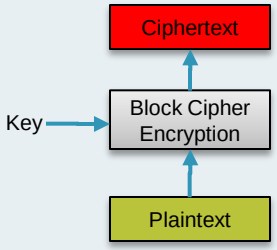
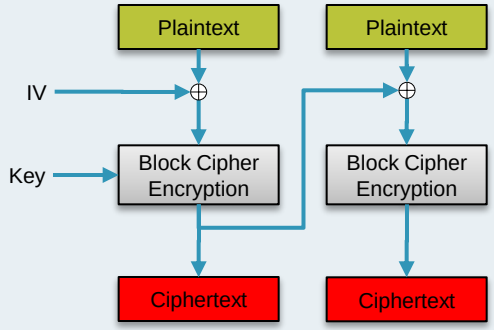
⇒

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)
  - Scientists expect RSA-1024 will be insecure by 2014
- The RSA algorithm involves three steps
  - Key generation
  - Encryption
  - Decryption

RSA - The Algorithm

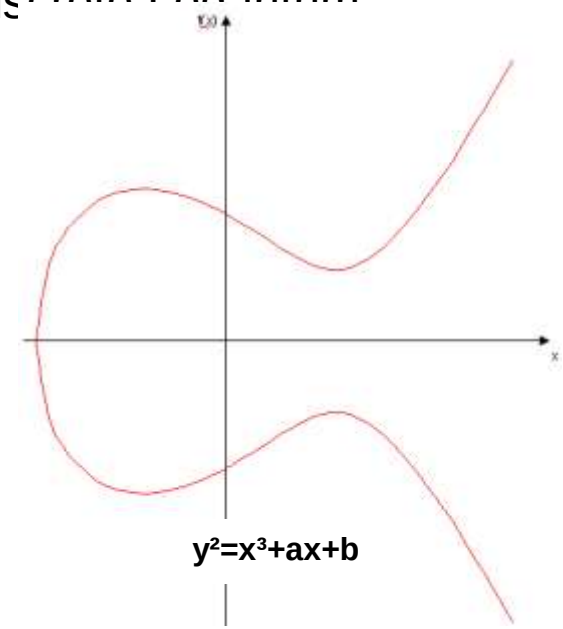
	Algorithm	Example
Key generation	Choose two prime numbers p and q	$p = 223$ $q = 199$
	Compute n=pq (modulus for public & private keys)	$n = 223 \cdot 199$ $= 44377$
	Compute *Totient function:* $\varphi(p,q)=(p-1)\cdot(q-1)$	$\varphi(p,q)=(p-1)\cdot(q-1)$ $=(223-1)\cdot(199-1)$ $=43956$
	Choose an integer e Rules: $1<e<\varphi(p,q)$ e and $\varphi(p,q)$ share no divisors other than 1	$e=5$
	Determine d with the extended Euclidean algorithm $[de \equiv 1 \pmod{\varphi(pq)}]$	$d=35165$ $ed-1=5\cdot35165-1$ $=175824$
	public key=(n;e) / secret key=(n;d)	public key = (44377; 5) secret key = (44377; 35165)
Encoding	$m^e \equiv c \pmod{n}$	m="Bo"=[0x42,0x6F]=0x426F=17007 $c=17007^5 \bmod 44377 = 23986$
Decoding	$c^d \equiv m \pmod{n}$	c=23986 $23986^{35165} \bmod 44377 = 17007$

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



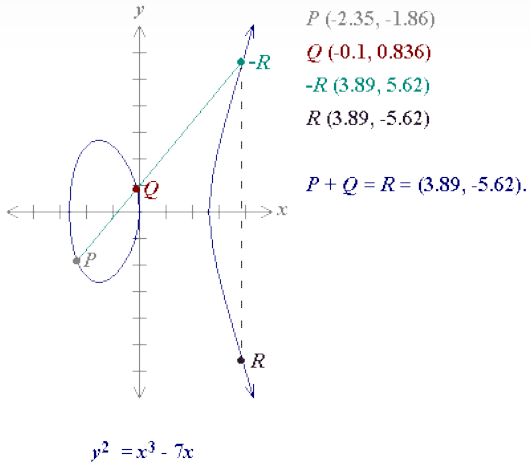
Security Level(*) Restricted → Confidential →Secret→Top secret	NIST Recommended Key Sizes [bits]		
	AES	RSA	ECC
Secret	128	3072	256
Top Secret	192	7680	384
	256	15360	512

(*) Security Levels proposed by NIST

Elliptic Curve Addition: A Geometric Approach

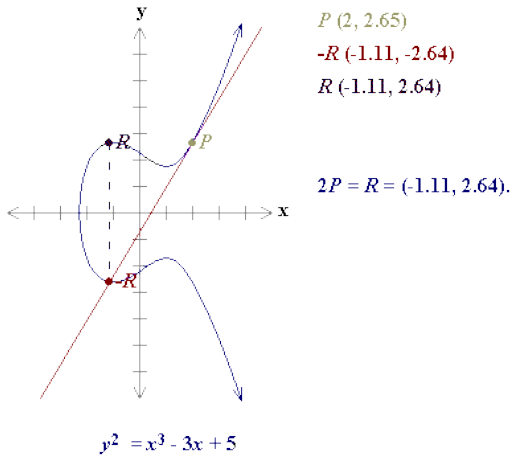
P + Q = R

To add the points P and Q, a line is drawn through the two points. This line will intersect the elliptic curve in exactly one more point, call -R. The point -R is reflected in the x-axis to the point R.



P + P = 2P = R ; k=2

To add a point P to itself, a tangent line to the curve is drawn at the point P. If yP is not 0, then the tangent line intersects the elliptic curve at exactly one other point, -R. -R is reflected in the x-axis to R.



Elliptic Curve Groups over Prime Fields Fp

For performance and precision reasons, usage of finite fields.

<u>Elliptic Curves for cryptography:</u>

Formula:

$k P = Q$

or

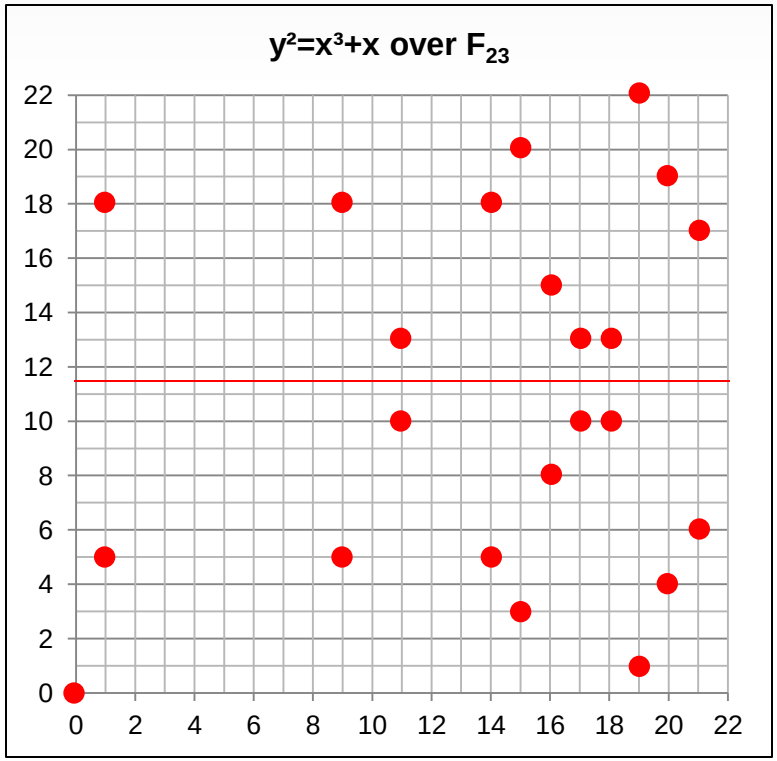
(k times P=+P) = Q

Known:

Points P and Q

Task:

Find k, the discrete logarithm of Q to base P



Example: F_{23}; a=1; b=0

$(y^2) \bmod 23 = (x^3 + x) \bmod 23$

The 23 points which satisfy this equation are:
(0,0) (1,5) (1,18) (9,5) (9,18) (11,10) (11,13) (13,5) (13,18) (15,3) (15,20) (16,8) (16,15) (17,10) (17,13) (18,10) (18,13) (19,1) (19,22) (20,4) (20,19) (21,6) (21,17)

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	

freescale™

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Secure Hashes

SHA and CMAC

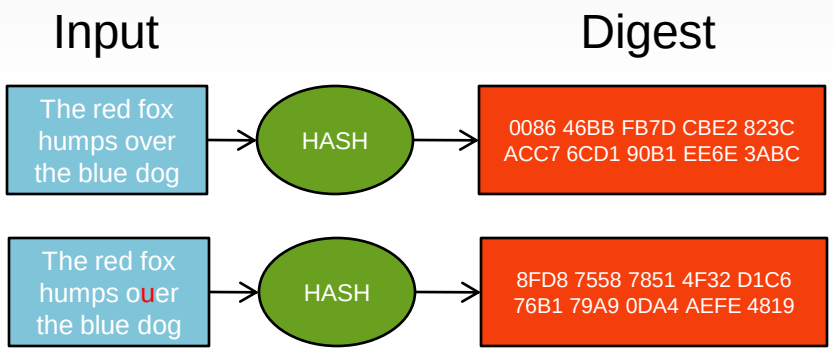
Secure Hash Algorithms

A cryptographic hash function is a deterministic procedure that takes an arbitrary block of data and returns a fixed-size bit string.

Detection of accidental or intentional change.

Main or significant properties:

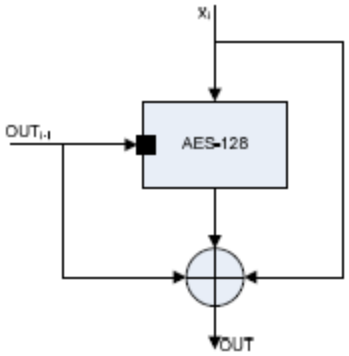
- It is easy to compute the hash value for any given message,
- It is infeasible to generate a message that has a given hash,
- It is infeasible to modify a message without changing the hash,
- It is infeasible to find two different messages with the same hash.



Algorithm	Output size [bits]	Internal state size [bits]	Block size [bits]	Length size [bits]	Word size [bits]	Collision attacks	Preimage attacks
MD5	128	128	512	64	32	Yes	Yes
SHA-1	160	160	512	64	32	Yes	
SHA-256/224	256/224	256	512	64	32	No	No
SHA-512/384	512/384	512	1024	128	64	No	No
WHIRLPOOL	512	512	512	256	8	No	

Cipher-based MAC (CMAC)

- MAC = message authentication code
- MACs are used for data authentication
- Cipher key is the “identifier”, only the secret owner can produce the right CMAC for a given message



AES based CMAC system

freescale™

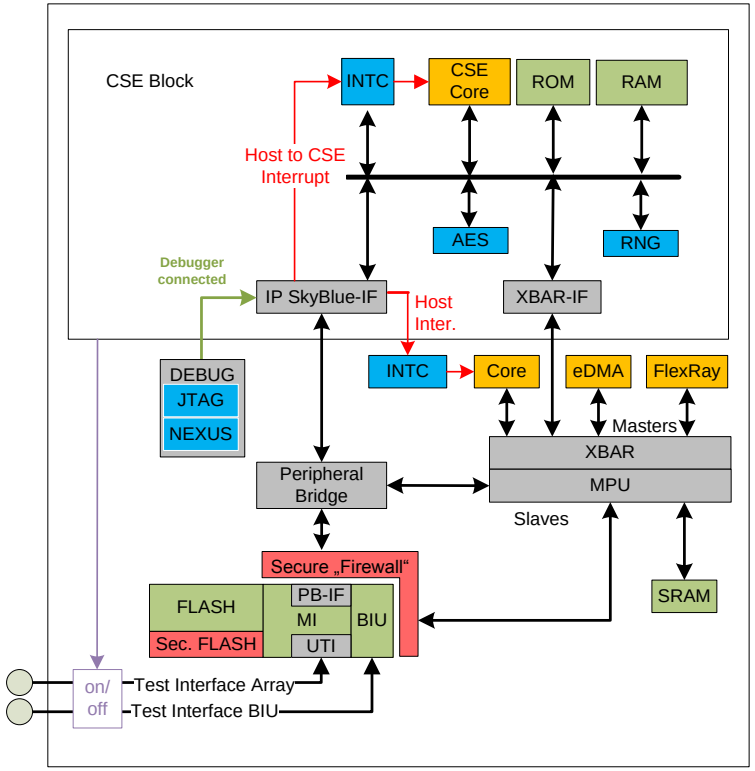
FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

32-bit Qorriva Controllers

Security Hardware Features

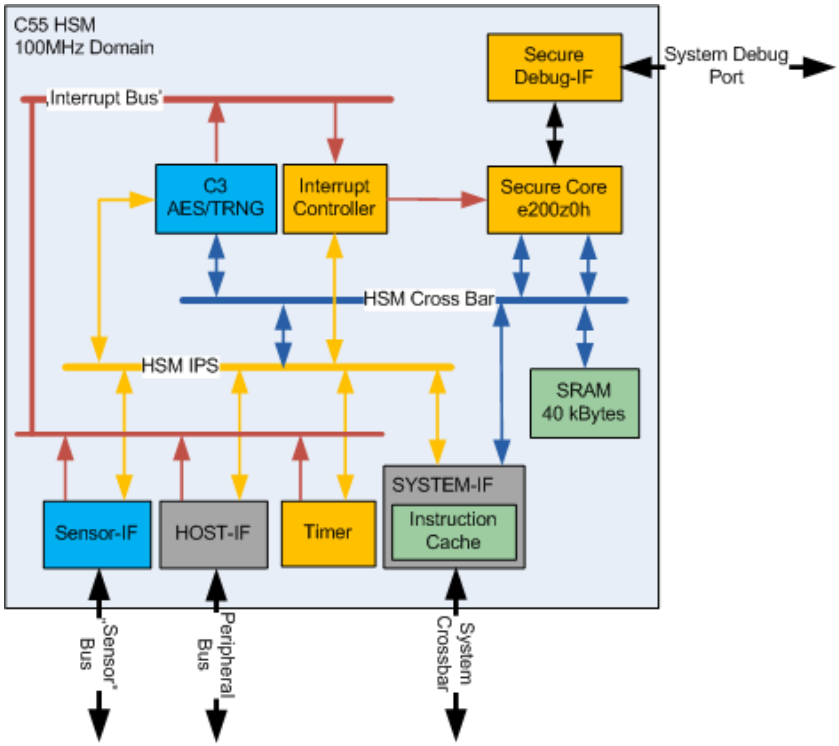
Cryptographic Services Engine (CSE) Qorivva MPC564xB/C

- CSE module implements the official SHE-Specification (Version 1.1), a HIS standard
- 32-bit secure core working at 120 MHz
- AES
  - 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



Hardware Security Module (HSM)
Qorivva MPC5746M

- e200z0h core @ 100MHz
- Crossbar with MPU
- Interrupt controller (32 x HOST to HSM etc.)
- Memory
  - 40KByte internal SRAM
  - Flash
    - Data: 2x 16KBytes
    - Code: 2x 64 KBytes; 1x 16KBytes
- Cryptographic Core (C3)
  - AES-128
  - Random Number Generator
  - DMA functionality
- Sensor Interface – monitor for
  - Voltage
  - Temperature
  - Clock



freescale™

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

32-bit Qorriva Controllers

Use Cases

Secure Boot

Check boot loader for integrity and authenticity

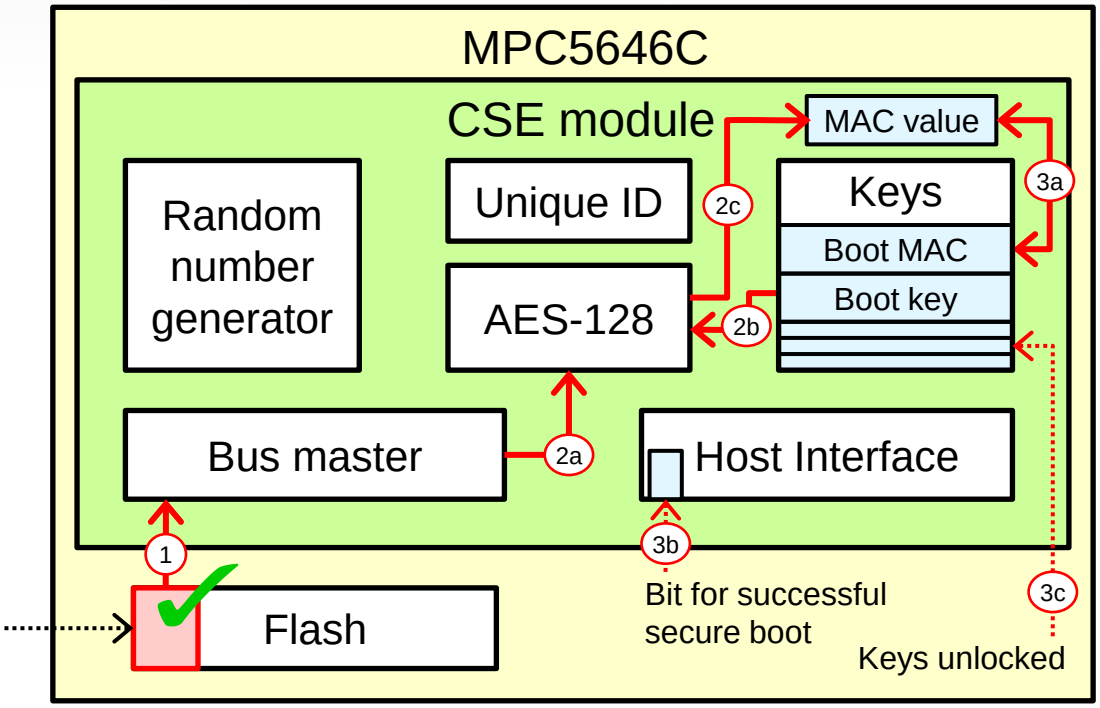
Step 1: After power on: CSE module reads bootloader via its bus master interface.

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

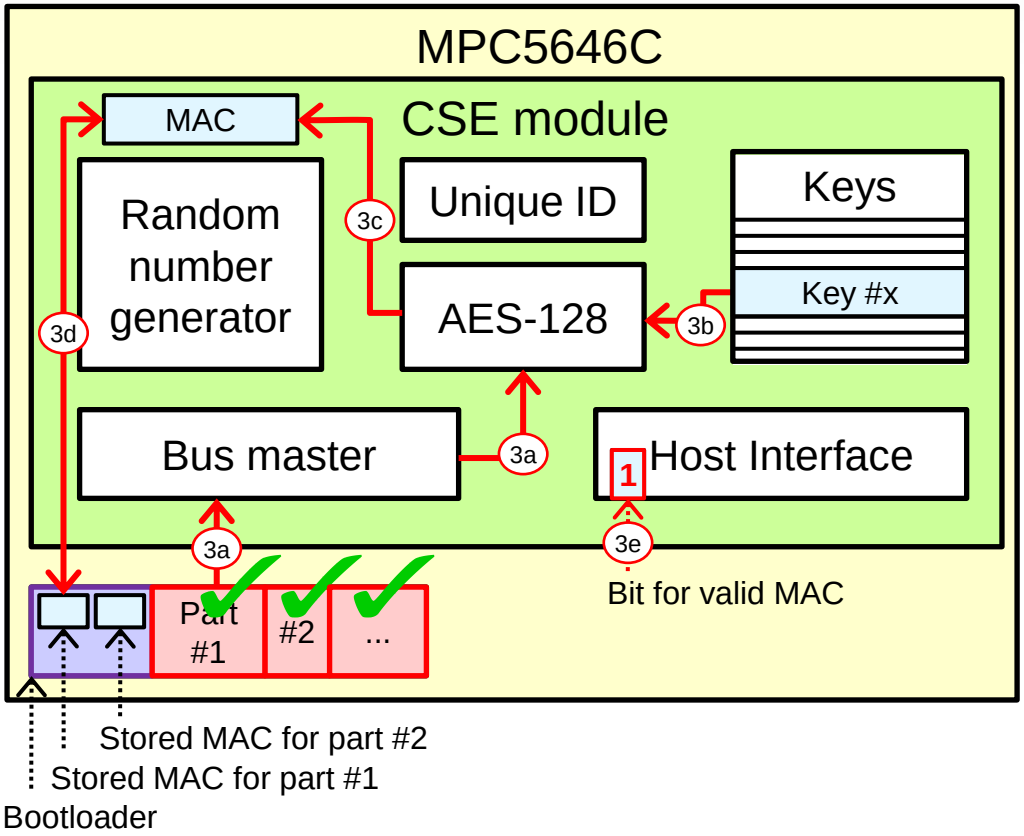
Step 1: Successful secure boot to verify bootloader and to unlock keys. Then the bootloader configures the MCU for full speed, best memory timing, etc.

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.

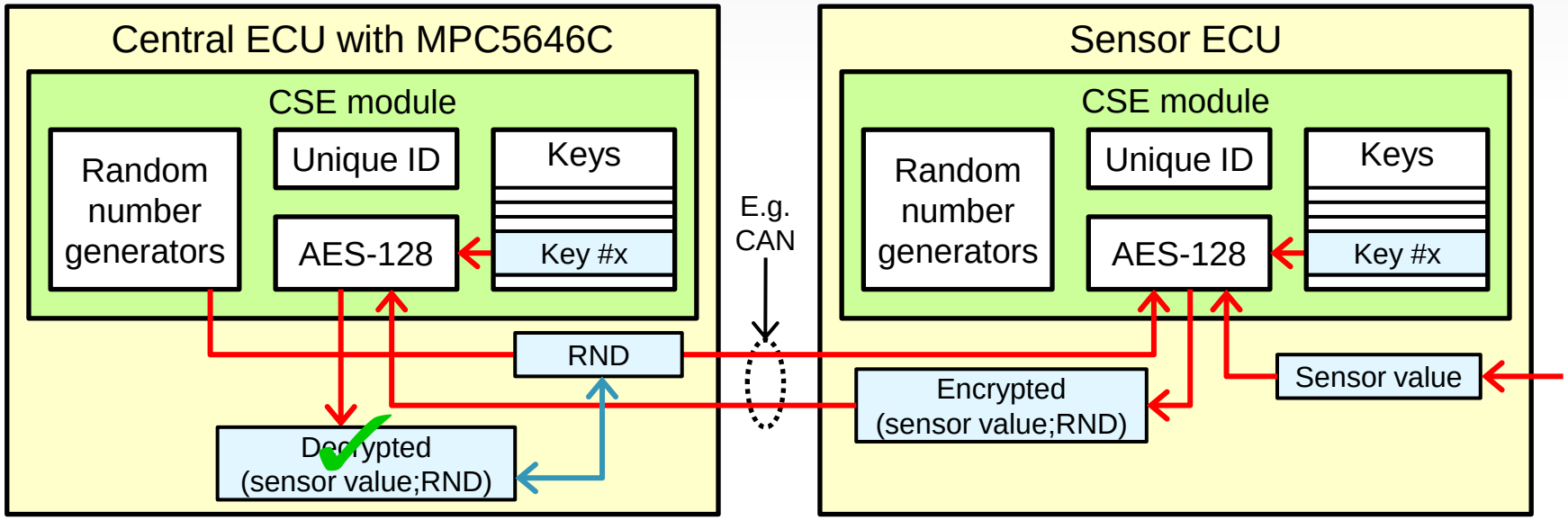
Step 5 etc: Similar to bootloader vs. part #1 of flash: Part #n of flash verifies part #n+1 → chain of trust.



- 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

Mileage Protection

Protect mileage meter against modification

Step 1: Application asks CSE module to verify MAC of stored mileage (using key #x)

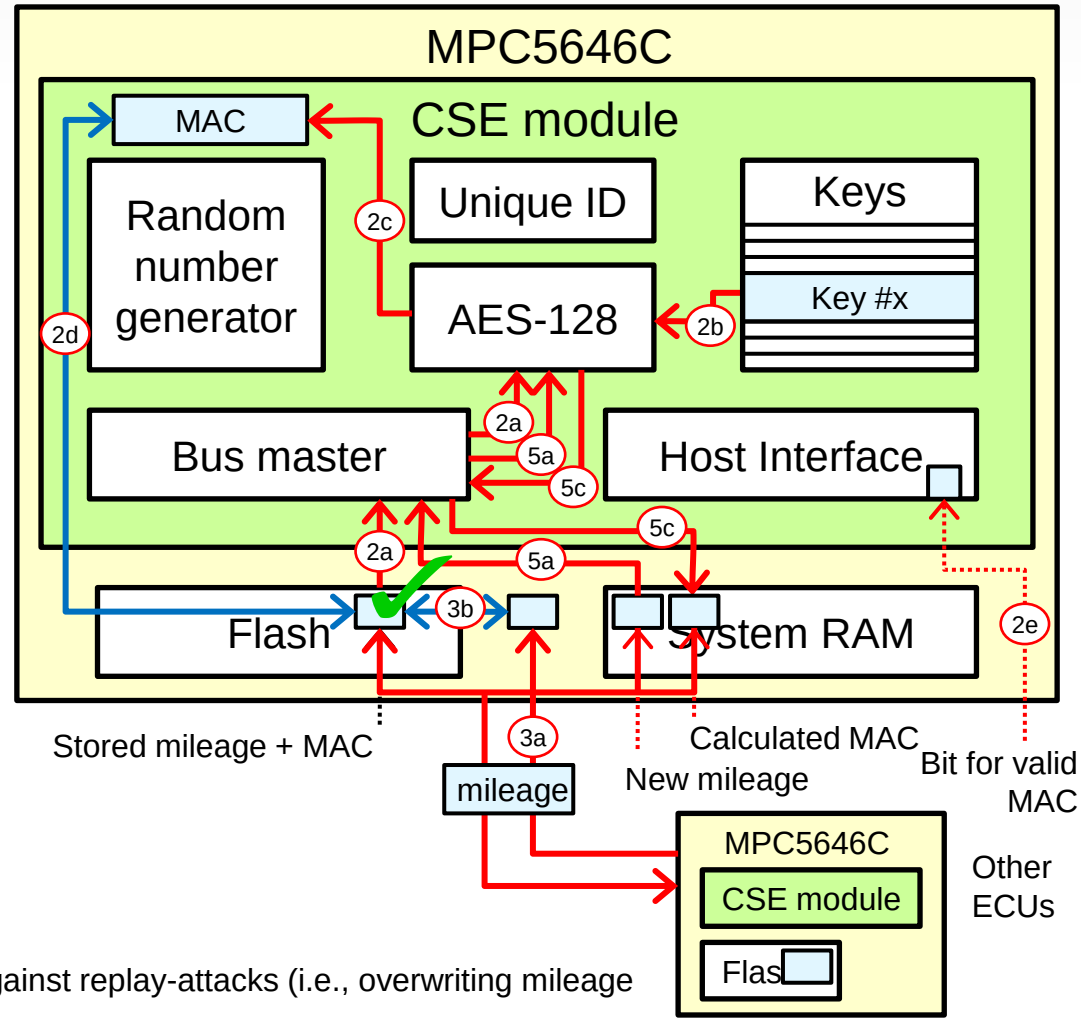
Step 2: CSE module reads mileage and MAC. CSE module uses key #x to calculates MAC. CSE module compares both MACs. If identical: CSE module sets bit in host interface.

Step 3: Application checks bit and asks other ECUs for mileage (via secure communication). If bit is set and other ECUs reports same mileage: stored mileage is ok.

Step 4: ECU gets new mileage. Application asks CSE module to generate MAC of new mileage (using key #x).

Step 5: CSE module reads new mileage. CSE module uses key #x to calculates MAC. CSE module writes MAC to system RAM.

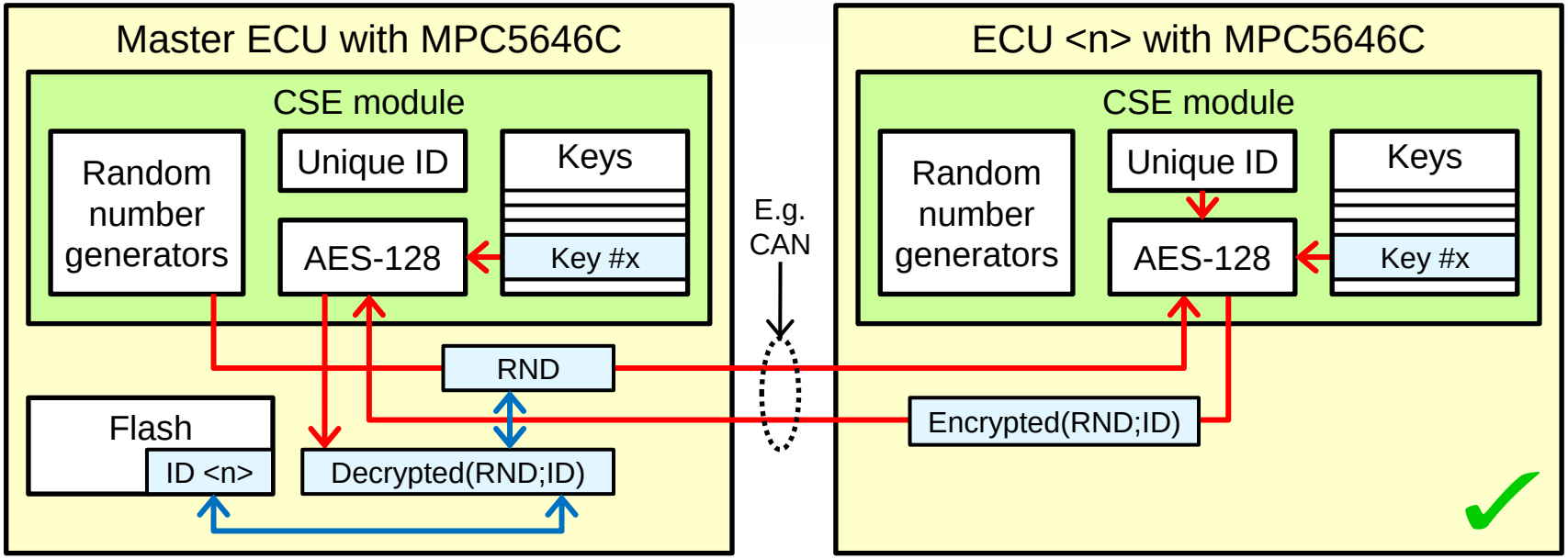
Step 6: Host writes new mileage and its MAC into flash. Host sends new mileage to other nodes (secure communication)



- MAC protects mileage against modification.
- Distributing mileage on other ECUs protects against replay-attacks (i.e., overwriting mileage and MAC with read old mileage and its MAC).

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

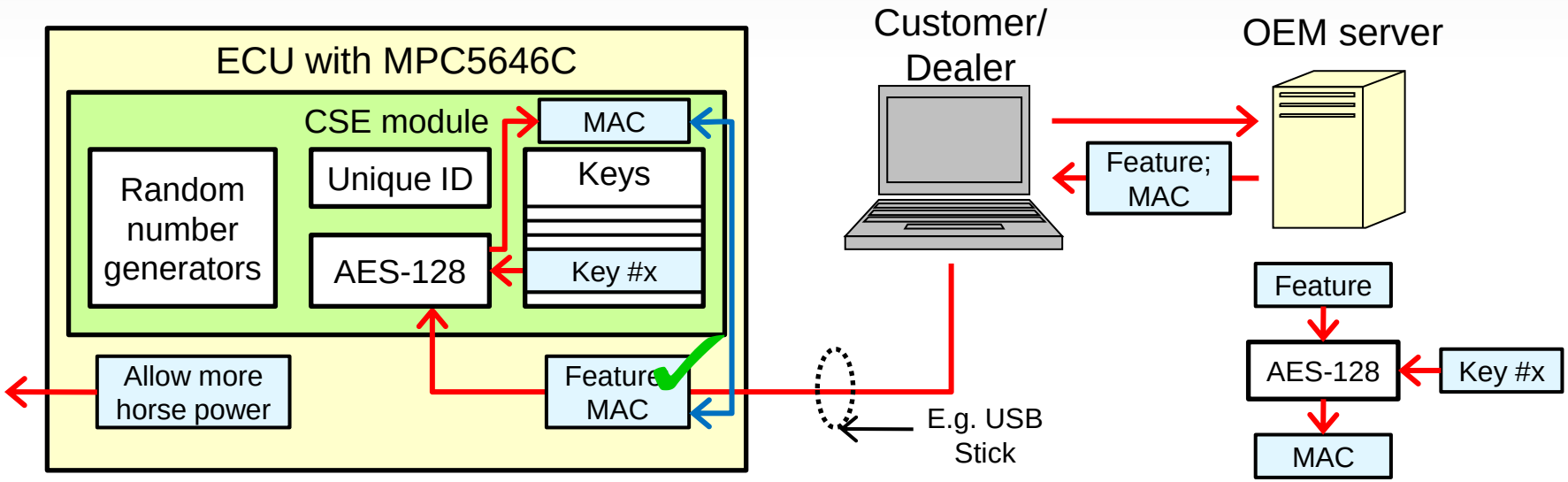
Step 3: Master ECU decrypts received message with key #x.

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.

Feature Activation

Enable activation of car features



Step 1: Customer/Dealer request feature (e.g., more horse power) for the car from OEM's server.

Step 2: OEM server creates message: feature code and MAC using key #x of ECU in customer's car. Message sent to customer

Step 3: Customer applies message (feature;MAC) to car. ECU receives message.

Step 4: CSE module calculates MAC for received feature using key #x. If calculated MAC is identical to received MAC: feature is accepted (e.g., more horse power)

Only features with correct MAC/AES-128 key are accepted.
OEM server to create the MAC for feature.
MAC protects integrity and authenticity of feature.

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

i.MX Applications Processors

Security for the Connected Car

Steven Pan
Sr. MAD SOC platform Engineer

Connected Car – Emphasis for i.MX

Connected car

- Opens new opportunities for driver information & assistance features
- Enables new threats going beyond current automotive security
- Focuses attention on isolation between open and critical subsystems

i.MX Trust Architecture

- Provides multiple cohesive protection features
- Guards against sophisticated attacks
- Assures software security measures

Use Cases

High-speed, high-bandwidth network access

- Digital Rights Management of streaming audio/video content
- Remote ECU diagnosis & update
- Remote status & control via the Telematics system
- Enhanced navigation
- Richer user experience
  - Cloud services, apps, social networking…

Platform protection

- Software
  - Protection against root-kits, viruses, malicious hacks
- Separation of safety critical domains and other domains
  - Data & content exchange between Cluster and Infotainment domain

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Trust Architecture Features

i.MX Trust Architecture Features

Trusted Execution

- Isolates execution of critical SW from possible malware
- TrustZone Secure & Normal Worlds (processor modes)
- Hardware firewalls between CPU & DMA masters and memory & peripherals

High Assurance Boot

- Authenticated boot: prevents unauthorized SW execution
- Encrypted boot: protects SW confidentiality
- Digital signature checks embedded in on-chip boot ROM
- Run every time processor is reset

HW Cryptographic Accelerators

- i.MX family dependent
- Symmetric: AES-128, AES-256, 3DES, ARC4
- Message Digest & HMAC: SHA-1, SHA-256, MD-5

i.MX Trust Architecture Features (continued)

Secure Storage

- Protects data confidentiality and integrity
- Off-chip: cryptographic protection including device binding
- On-chip: self-clearing Secure RAM
- HW-only keys: no SW access

HW Random Number Generation

- Ensures strong keys and protects against protocol replay
- On-chip entropy generation
- Cryptographically secure deterministic RNG

Secure Clock

- Provides reliable time source
- On-chip, separately-powered real-time clock
- Protection from SW tampering

i.MX Trust Architecture Features (continued)

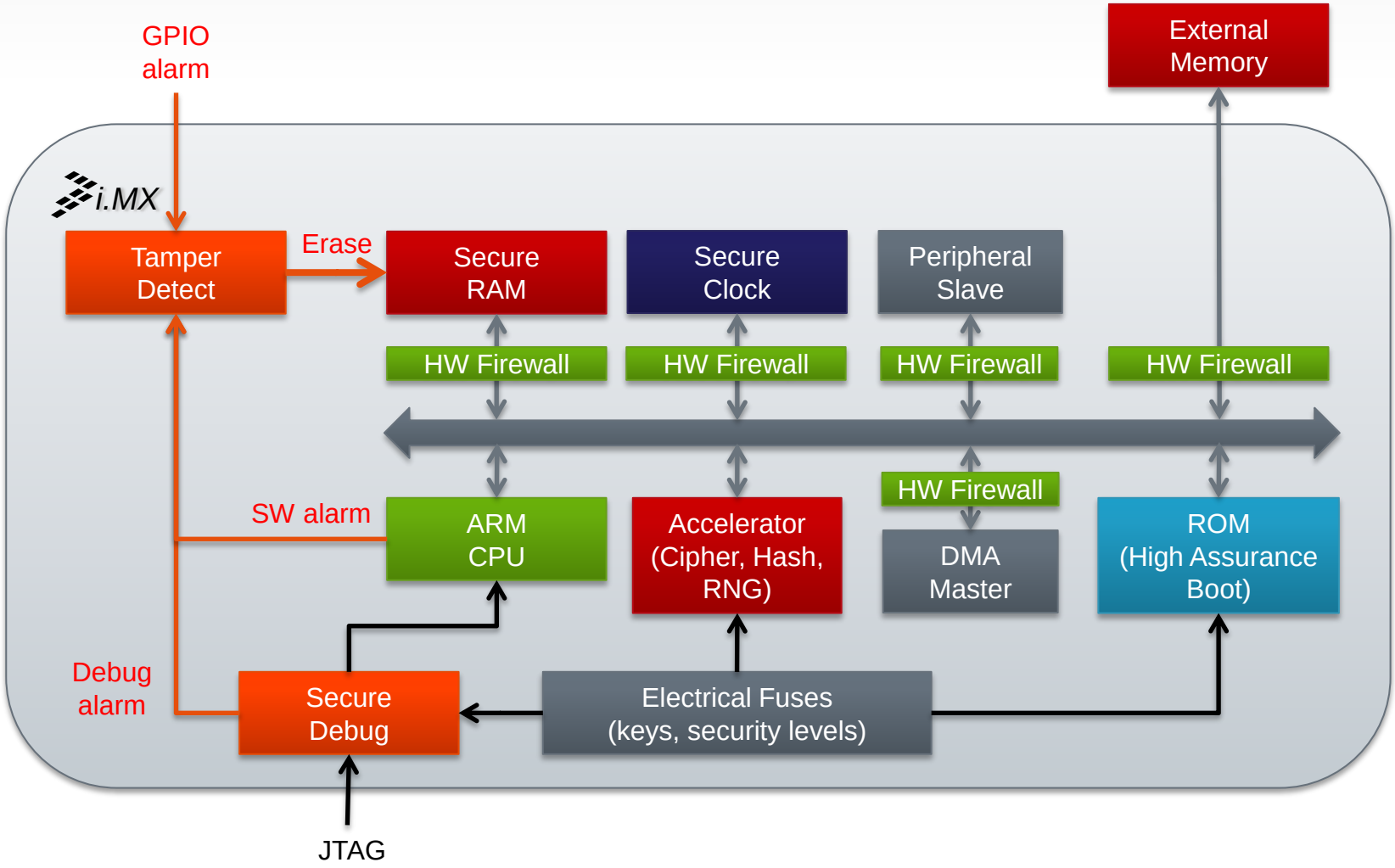
Secure Debug:

- Protects against HW debug (JTAG) exploitation for:
  - Security circumvention
  - Reverse engineering
- Three security levels + complete JTAG disable

Tamper Detection

- Protects against run-time tampering
- Monitoring of various alarm sources
  - Debug activation
  - External alarm (e.g. cover seal)
  - SW integrity checks
  - SW alarm flags
- HW and SW tamper response

i.MX Trust Architecture – Overview



Trust Architecture features Use Case Matrix

Feature	Content protection with DRM	Remote ECU updates	Remote status and control via telematics	Content exchange between cluster and infotainment domains	Software Protection
Trusted Execution	✓	✓	✓	✓	✓
High Assurance Boot	✓	✓	✓	✓	✓
Secure Storage	✓	✓	✓	✓	
Hardware RNG	✓	✓	✓	✓	
Secure Clock	✓				
Secure Debug	✓	✓	✓	✓	✓
Tamper Detection	✓	✓	✓	✓	✓

freescale™

freescale™

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

High Assurance Boot

High Assurance Boot – Purpose

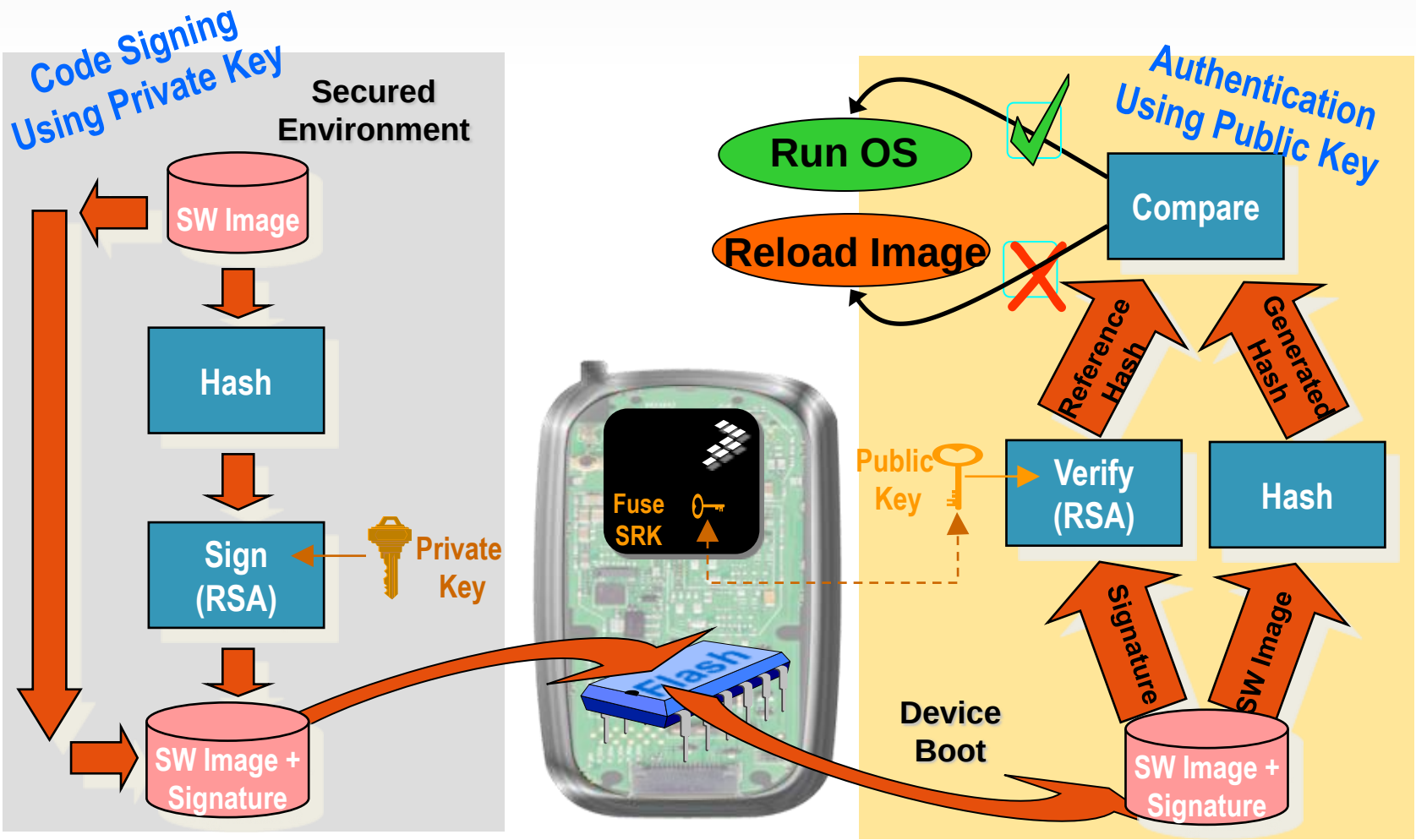
High Assurance Boot ensures the boot sequence

- Uses authentic SW
- Establishes a “known-good” system state
- Essential for secure systems
  - Use case independent

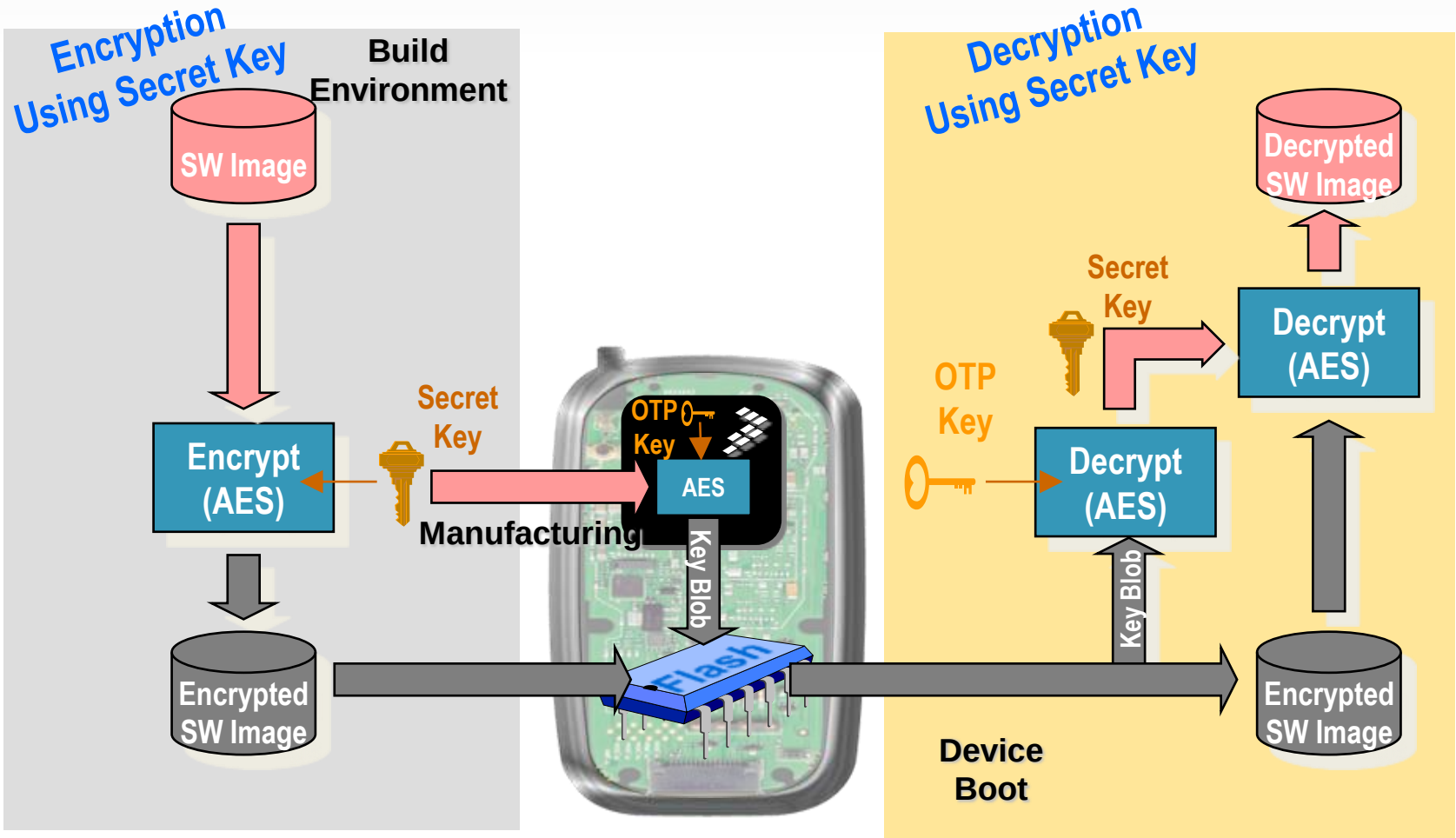
High Assurance Boot protects against

- Platform re-purposing
- Rootkits and similar unauthorized SW designed to
  - Harvest secrets
  - Circumvent access controls

High Assurance Boot – Operation



High Assurance Boot – Encrypted



High Assurance Boot – Tools

Freescale Reference Code Signing Tool (CST)

- Offline process of creating digital signatures
- Signing Keys and signatures generated by device manufacturers
- Supports code signing for: i.MX258, i.MX28, i.MX35x, i.MX508, i.MX51x, i.MX53x and i.MX6x

Manufacturing Tool

- Platform software provisioning
- One-Time Programmable e-fuse burning

- Both can be downloaded from: http://www.freescale.com/webapp/sps/site/overview.jsp?code=IMX_DESIGN

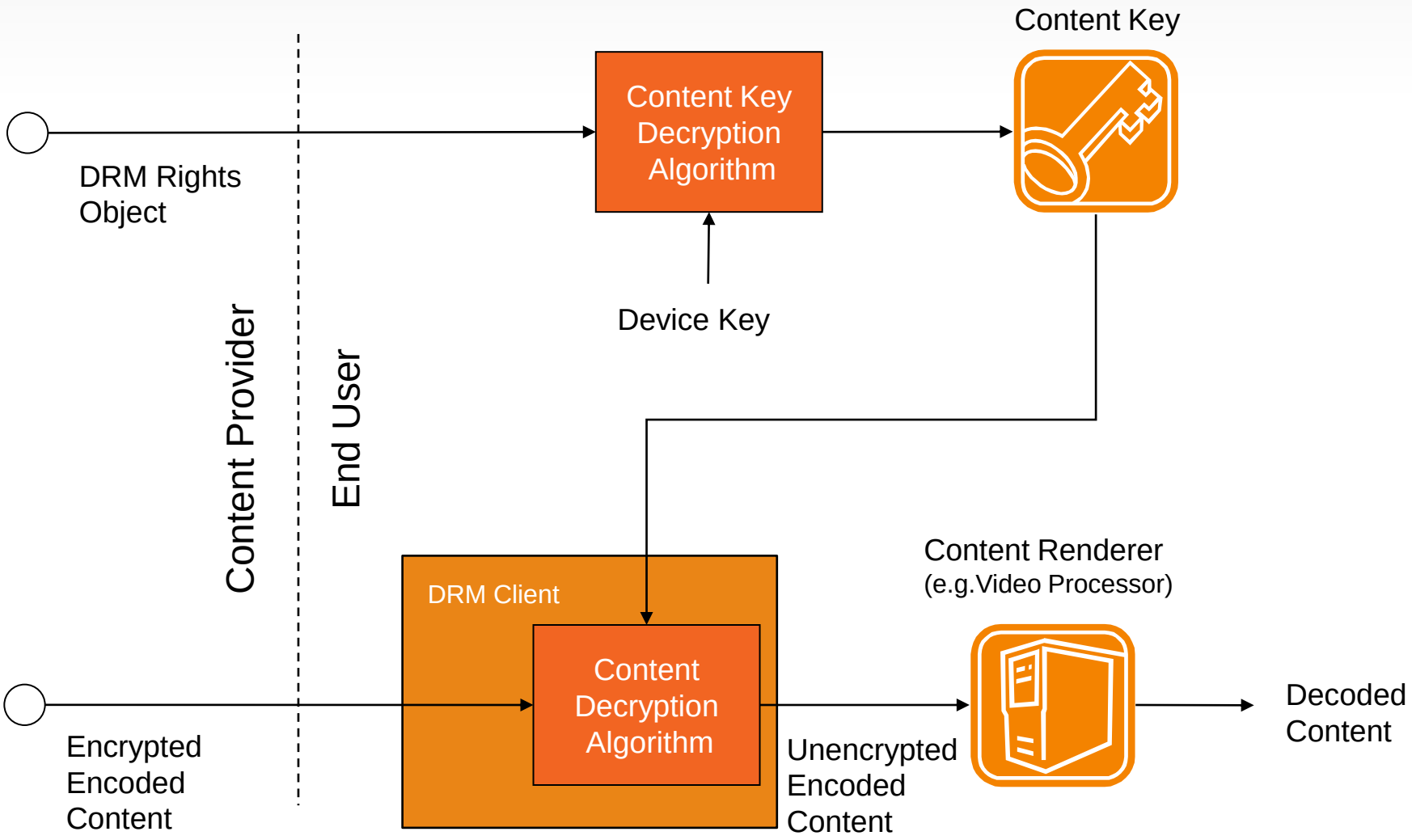
freescale™

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Use Case

Digital Rights Management

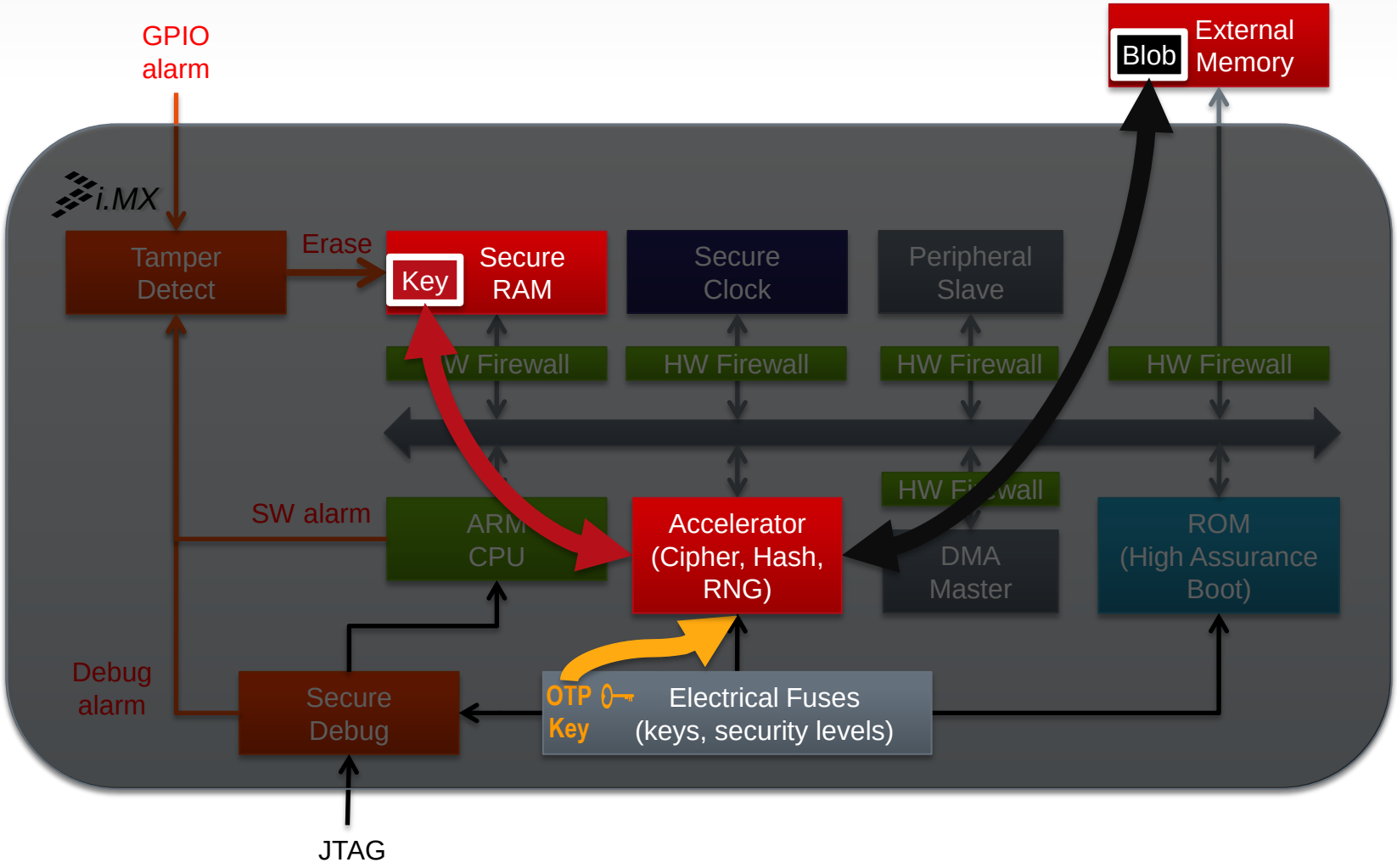
Simplified DRM Overview



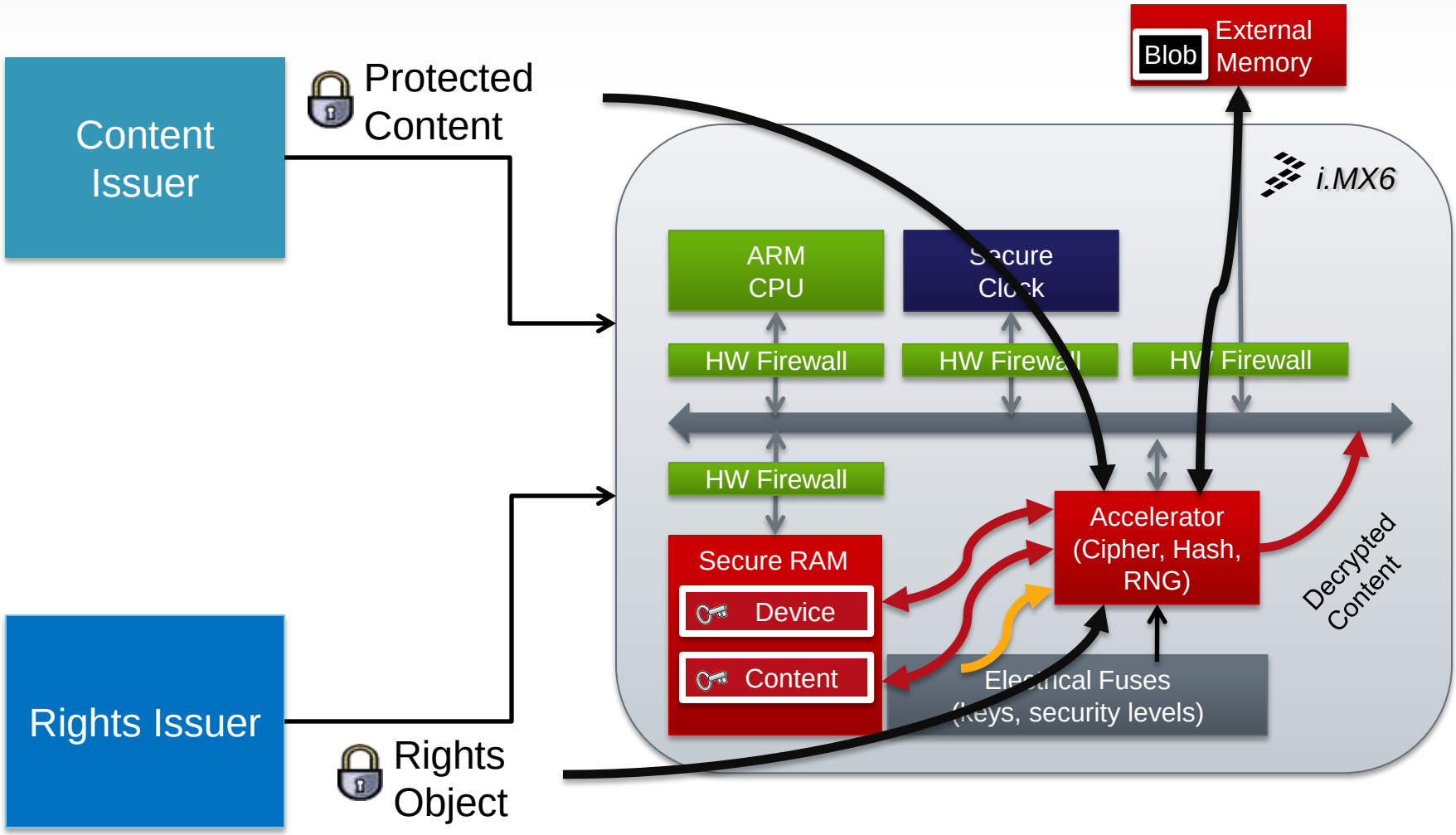
DRM Protection

- How can we protect the DRM keys and content?
  - Secure Storage
  - Hardware Cryptographic Algorithm Support
- How can we protect the DRM client software?
  - High Assurance Boot
  - Trusted Execution
  - Secure Debug
- What other trust architecture features are required?
  - Random Number Generator
    - Needed for key exchange algorithms
  - Secure Time
    - Required by DRM scheme robustness rules

DRM Device Key Protection (Secure Storage)



DRM Content Decryption



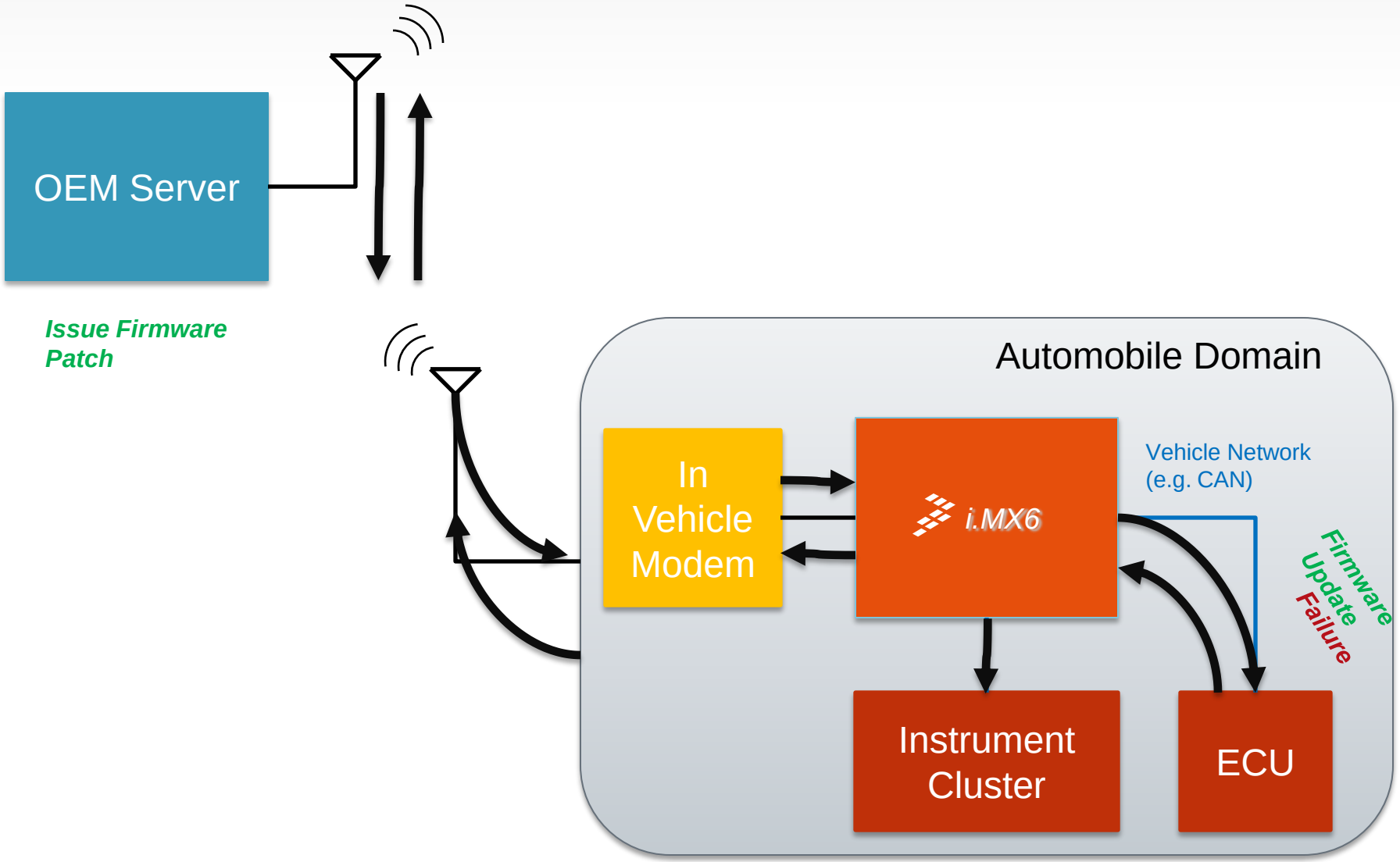
freescale™

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Use Case

Remote ECU Diagnosis & Update

Remote ECU Diagnosis and Update



Remote Update Protection

- How can we ensure the update mechanism is not abused?
  - High Assurance Boot
  - Trusted Execution
  - Hardware Cryptographic Algorithm Support
  - Secure storage for authentication public key certificates
  - Secure Debug

freescale™

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Trusted Execution

Trusted Execution – Purpose

Trusted execution allows critical SW to co-exist with a rich platform SW environment on a single IC

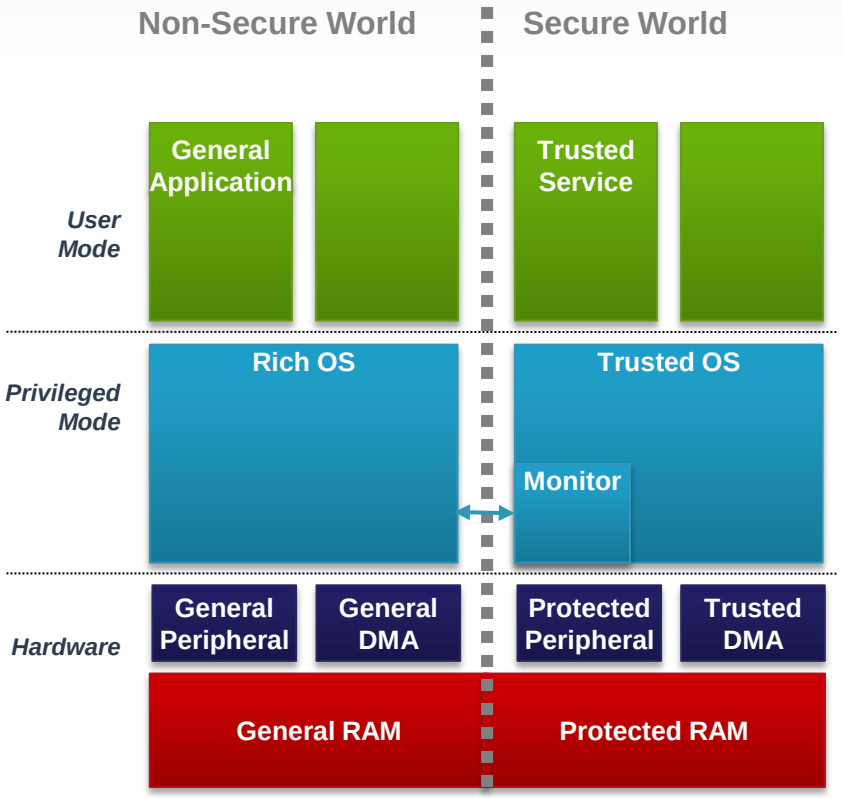
- Isolates trusted SW performing critical services
- Protects confidentiality and integrity of sensitive data
- Protects critical peripherals and memory
- Enables access for trusted DMA masters only

Trusted execution protects against

- Attacks from compromised platform SW
- Access to protected peripherals and memory
- Backdoors using untrusted DMA masters
- Starvation of resources available to critical services

Trusted Execution – i.MX Hardware Features

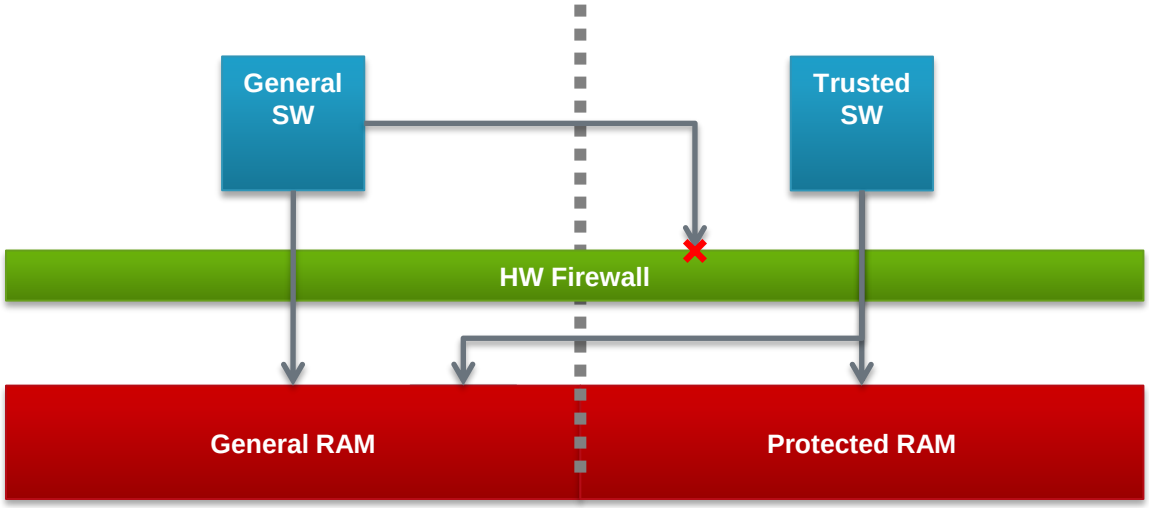
- HW system partitioning
- ARM TrustZone
  - Secure & non-secure worlds
  - Multicore support
- Memory isolation
  - Virtual & physical
- Peripheral isolation
  - Master & slave
- Interrupt separation
- Watchdog protection



Trusted Execution – Physical Memory Isolation

HW Firewall

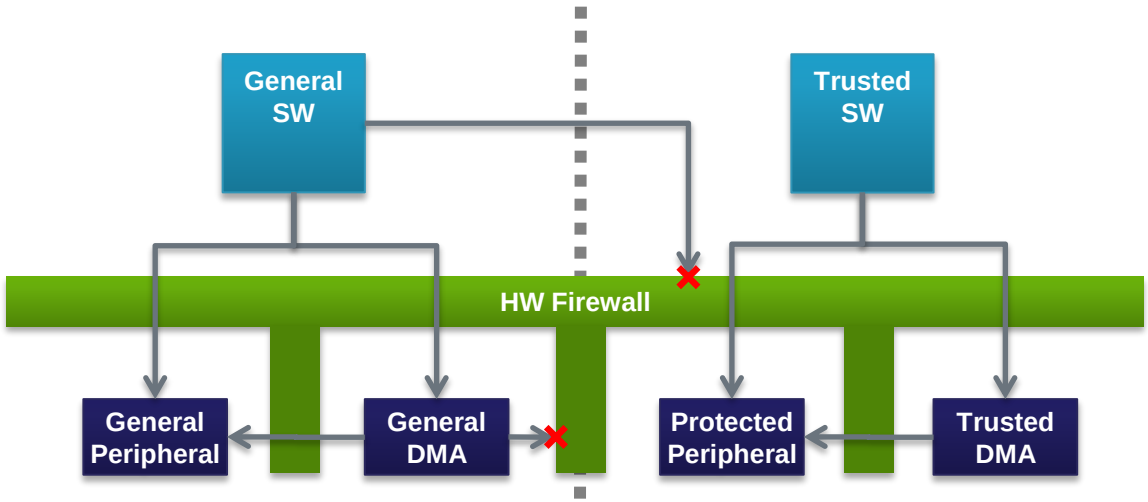
- Monitors internal bus to external memory controller
  - Secure world access only vs shared access
- Programmed by Secure World



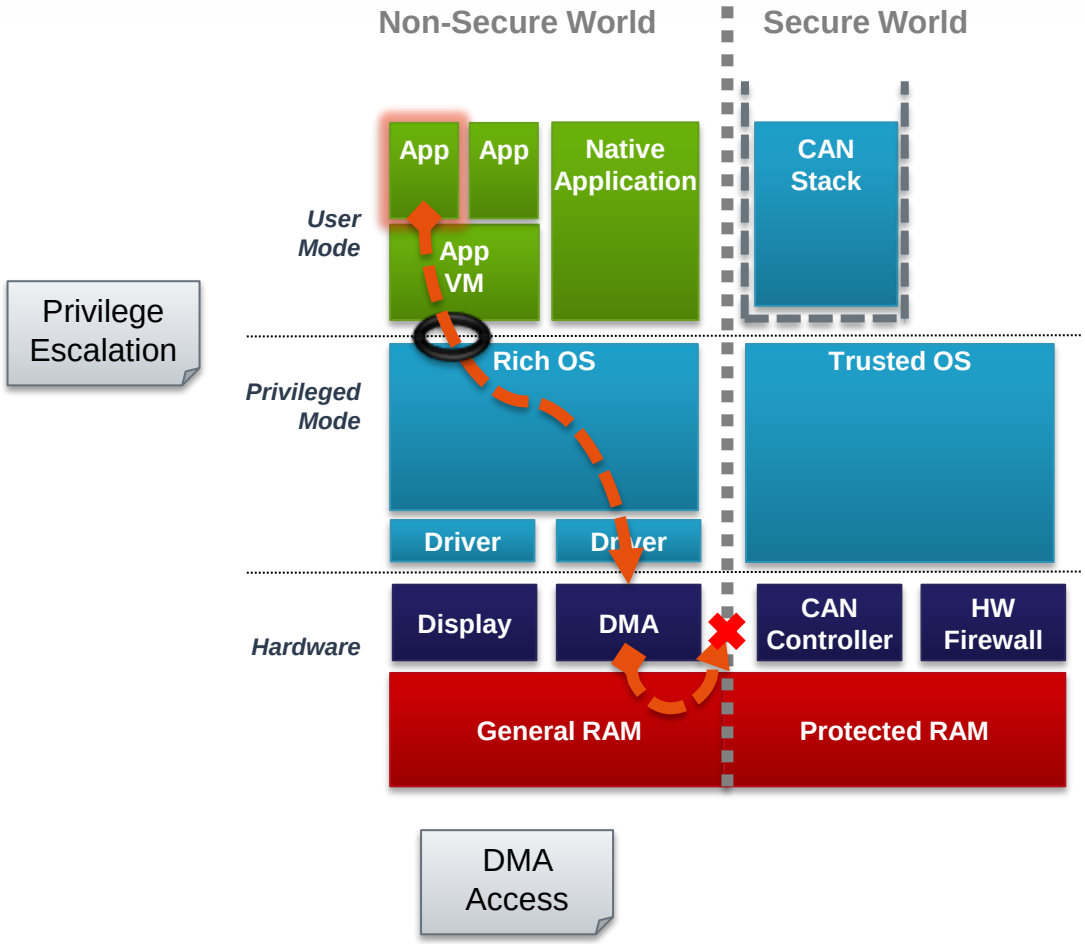
Trusted Execution – Peripheral Isolation

HW Firewall

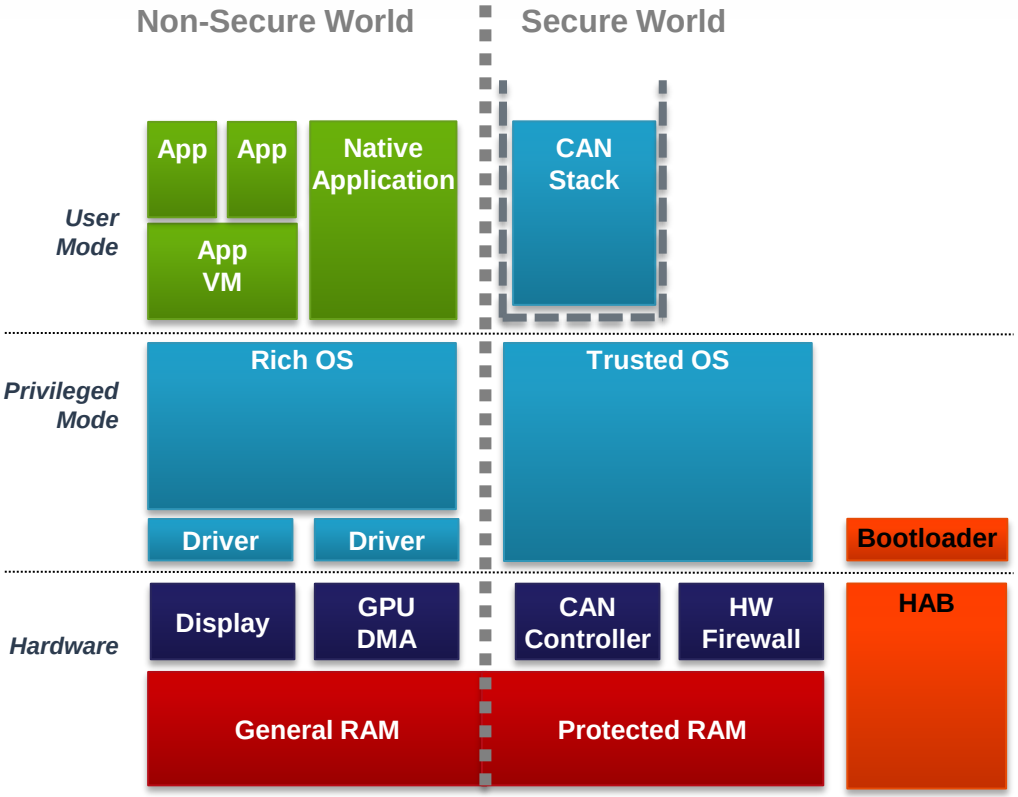
- Monitors peripheral access
  - Secure world access only vs shared access
- Monitors DMA transactions
  - Secure vs non-secure privileges
- Programmed by Secure World



DMA Attack Example



Trusted Execution and High Assurance Boot Example



FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Additional i.MX Trust Architecture Features

Other i.MX Security Features
Not covered in depth here

- Cryptographic Algorithms
  - AES, 3-DES, SHA1, SHA-256, ARC4, and others
- Random Number generation
  - Essential for key generation
- Secure Clock
  - Required for some Digital Rights Management (DRM) schemes
- Secure Debug
- Tamper Detection
  - Capabilities vary depending on i.MX family
- See FTF-CSD-F0211 for additional information

freescale™

FTF | FREESCALE TECHNOLOGY FORUM
POWERING INNOVATION

Summary

Summary

- Automotive Security: Protection and Enablement
  - Necessity and feature
  - Challenge and opportunity for new driver experiences
- Automotive Security: A global concern, requirement and new trend
  - Automotive industry in search for standardization
- Two types: In-vehicle Security and Connected Vehicle Security

Freescale Automotive Security Solutions				
Type		Device Family	Platform	Security Module
In-Vehicle Security	MCU (internal Flash)	MPC564xB/C (90nm Body)	32bit Qorivva Power Architecture	CSE
		MPC5746M (55nm Powertrain)		HSM
		Body (55nm)		HSM
Connected Vehicle Security	MPU (no Flash)	Vybrid R-Series	ARM Cortex-A5	Trust Zone Sahara CAAM
		i.Mx Application Processors	ARM9/11 Cortex-A8/A9/A15	

Q&A

Freescale on Kaixin
Tag yourself in photos
and upload your own!

Weibo?
Please use hashtag
#FTF2012#

Session materials will be posted @ www.freescale.com/FTF


freescale™