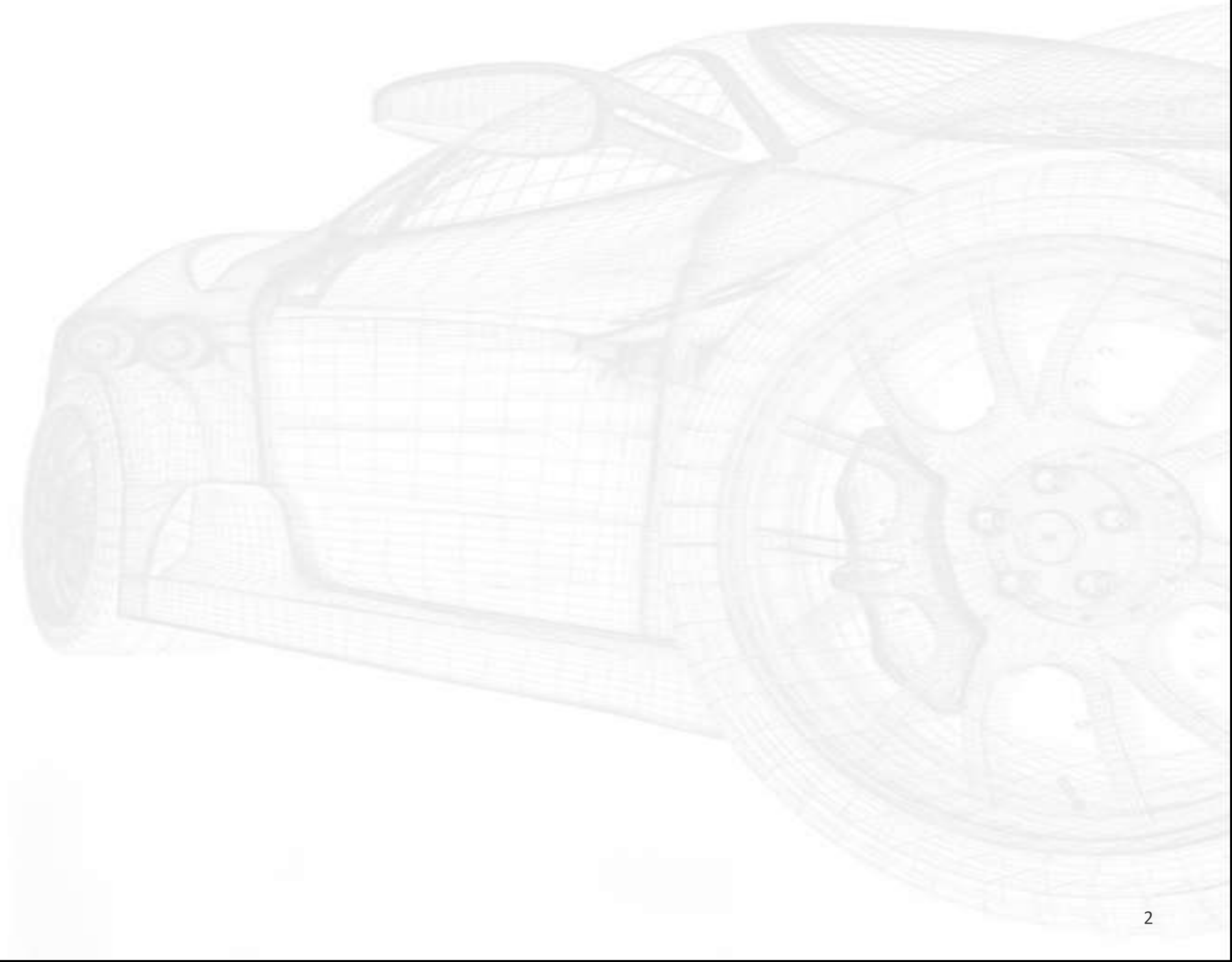
EB AUTOSAR training - Security stack

Elektrobit

Chapter overview

- Basic Security Concept
- Security Stack Modules
  - Csm, CryIf and Crypto
  - KeyM
  - SecOC: Secure Onboard Communication
  - DBKeyM and FvM
- Rte Usage



Basic Security Concept

Safety vs. Security

Safety is about **„Protect humans from system failure“**

- Risks and hazards that come from the system – protection against itself
- Robustness against unexpected faults and HW failure, e.g.:
  - Programming errors
  - Hardware errors
  - Transmission errors

Security is about **„Protect the system from humans“**

- When a chip is connected it has security issues
- Risks and hazards that come from outside the system
- Protection against attacks, e.g.:
  - Manipulate key signal
  - Influence car‘s functionality from outside
  - Manipulate odometer
  - Steal components

Security Properties

- **Confidentiality**
  - Data is only available to authorized users
  - Usage of encryption
- **Integrity**
  - Data cannot be modified in an unauthorized and undetected manner
  - Usage of authentication (e.g. signatures, MAC, Freshness value)
- **Availability**
  - Measures against denial of service attacks
- **Non-repudiation**
  - Sender cannot claim not having sent the message or different content (as he authenticated himself)
  - Usage of pin, fingerprint, trusted third parties

Cryptographic Methods

- **Encryption/Decryption**
  - Make sure that an attacker cannot get access to certain data
- **Authentication / Authorization**
  - **Authentication**:
    - Make sure that your communication partner is who they say they are
  - **Authorization**:
    - Make sure that your communication partner is allowed to do what they want to do
- **Hash**
  - Maps data of arbitrary size to fixed size
  - Reduce amount of data that has to be verified
- **Random number generator**
  - Provides a degree of randomness to cryptography

Symmetrical and asymmetrical cryptography

- **All cryptographic methods are based on knowledge of secrets**
- **Symmetrical cryptography:**
  - Both communication partners have the same secret key and must keep it secret
- **Asymmetrical cryptography:**
  - There is a pair of keys, a private key and a public key
    - One entity has a private key and must keep it secret
    - Everyone else can have the corresponding public key

Crypto Stack

Cryptography in AUTOSAR (since AUTOSAR 4.3)

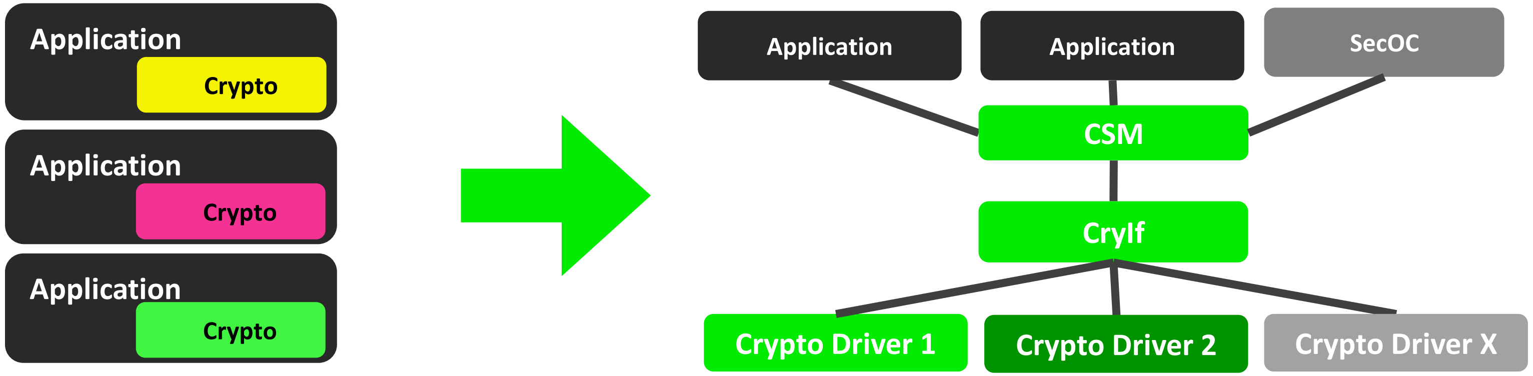
– A single ECU today hosts multiple security related applications

➡ Demand for a standardized approach to basic cryptographic routines

– AUTOSAR 4.3.x defines standardized crypto modules

- Crypto Service Manager – Csm
- Crypto Interface – CryIf
- Crypto Driver – Crypto
- Secure Onboard Communication – SecOC

➡ AUTOSAR does not provide a complete security solution, but building blocks that can be used by applications.



Terms and Definitions

Crypto Primitive:
A crypto primitive is an instance of a configured cryptographic algorithm realized in a Crypto Driver Object.

- i.e. AES-ECB, CMAC, RSA-PSS-Verify, etc.

Crypto Driver Object:
A Crypto Driver implements one or more Crypto Driver Objects.

The Crypto Driver Object can offer different crypto primitives

- in hardware
- or software.

The Crypto Driver Objects of one Crypto Driver are independent of each other.

There is only one workspace for each Crypto Driver Object

- i.e. only one crypto primitive can be performed at the same time

[Specification of Crypto Service Manager; AUTOSAR CP Release 4.3.0; 2.1 Glossary of Terms
Specification of Crypto Driver; AUTOSAR CP Release 4.3.0; 2.1 Glossary of Terms]

Terms and Definitions

Channel:
A channel is the path from a Crypto Service Manager queue via the CryIf to a specific Crypto Driver Object.

Job:
A job is a configured Object which refers to

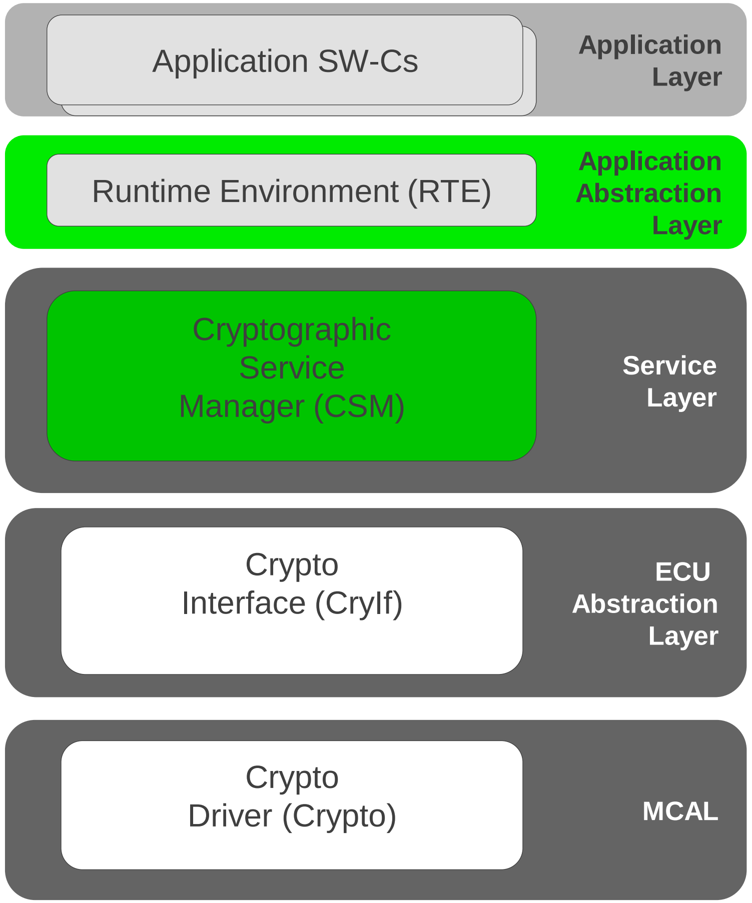
- a key,
- a cryptographic primitive,
- a channel,
- A callback

A job is configured via the Csm.

Its instance is forwarded via the CryIf to the dedicated Crypto Driver, if it is requested.

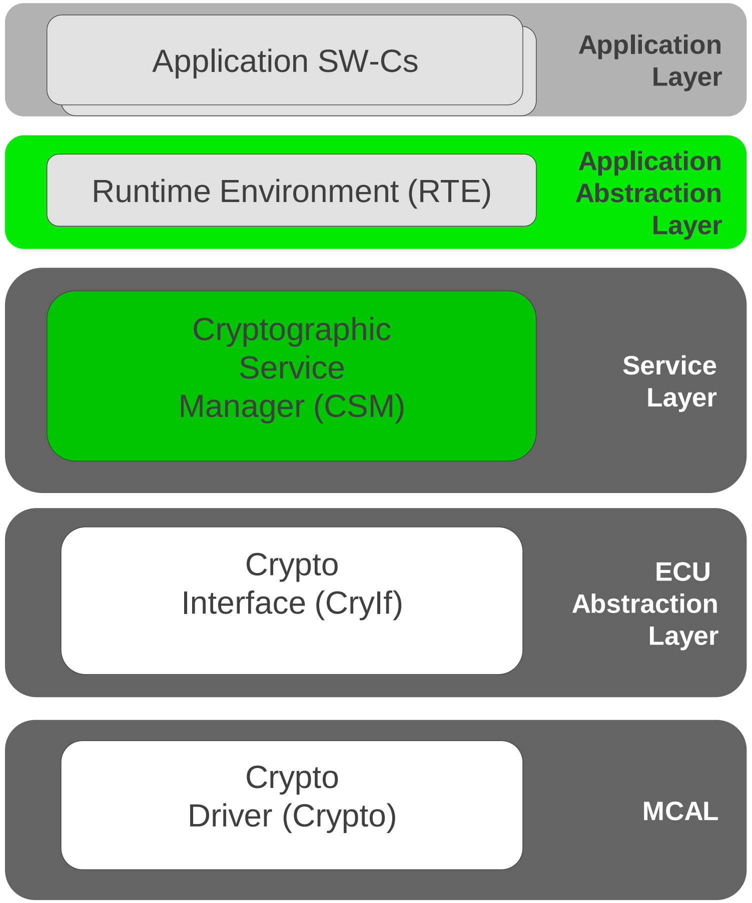
Crypto Service Manager – Csm

- **Provides algorithm-independent service interface to application**
  - E.g. interfaces for En-/Decryption, Signature Generation / Verification, Key Extraction
- **Provides interfaces for key management**
  - Application components only need to call Csm, without knowledge of the key
  - Key is determined by static configuration
  - Change the crypto algorithm without modifying the data paths in the application
- **Job-concept**.
  - multiple independent jobs can be processed in separate queues or channels quasi in parallel within the Csm.
- **Support of streaming and single-call within one Interface**
  - An Operation-Mode parameter defines the behavior
  - Improves performance in cases where streaming is not required
- **Prioritized queues**



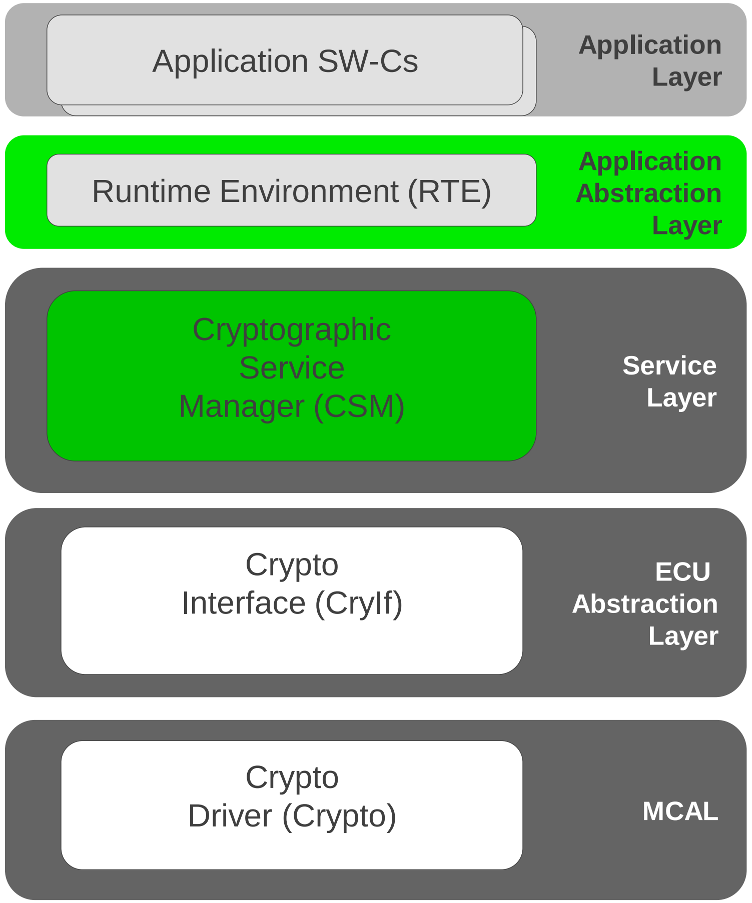
Crypto Service Manager – Csm

- **Defined Operation Modes for Job processing**
  - CRYPTO_OPERATIONMODE_START
  - CRYPTO_OPERATIONMODE_UPDATE
  - CRYPTO_OPERATIONMODE_FINISH
  - CRYPTO_OPERATIONMODE_STREAMSTART
  - CRYPTO_OPERATIONMODE_SINGLECALL
- **Processing modes for Primitives**
  - Synchronous
    - The results are in the provided Buffers, when the function call is finished
  - Asynchronous
    - The calculation is done during the main function call
    - A Callback function has to be provided
    - The results are in the provided Buffers, when the Callback was called



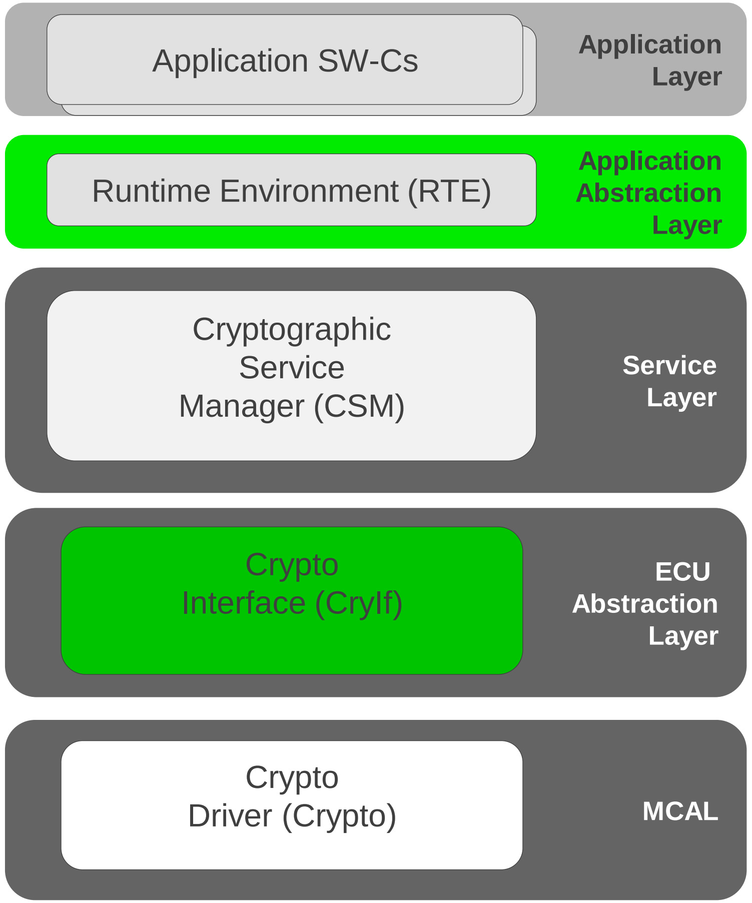
Csm - configuration

- **Job configuration**
  - Reference to Primitive
  - Reference to Key
  - Reference to Callback
- **Primitive configuration**
  - **Primitive algorithm family**
    - E.g. CRYPTO_ALGOFAM_RSA
  - **Primitive secondary family**
    - E.g. RSA-PKCS_1_7 is using the Hash Primitive
      - CRYPTO_ALGOFAM_SHA2_256
      - CRYPTO_ALGOFAM_SHA2_512
  - **Primitive processing type**
    - Synchronous or asynchronous



Crypto Interface - CryIf

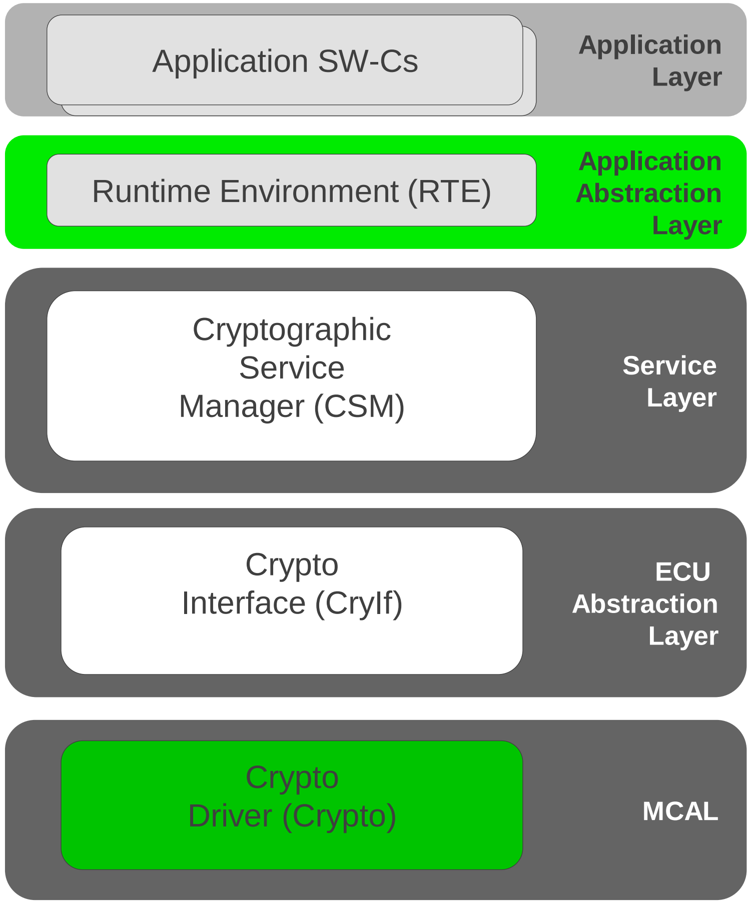
- Provides abstraction from Crypto Driver
- Maps one Crypto Driver Object via a Channel to a Csm queue
- Can handle multiple Crypto Drivers
  - But only one Csm



© Elektrobit (EB) 2022 | Confidential

Crypto Driver – Crypto

- Contains actual cryptographic algorithms (Primitives)
  - E.g. RSA-PSS signature verification, AES-CBC Encryption, CMAC
- Provide the functionality for Key storage and handling
  - Definition of the key structure with key elements
- Provides Crypto Driver Objects
  - A Crypto Driver Object includes a set of Primitives
  - The same Primitive can be in different CDO’s
    - Per CDO the Primitive has its own workspace
- A pre- and recommend configuration is provided
  - Key elements and key types for the implemented primitives are already defined



Supported Primitives from Crypto SW Generic

Csm service	Crypto Primitve
AEADEncrypt/AEADDecrypt	• AES-GCM
Encrypt/Decrypt	• AES-CBC (128, 192, 256) • AES-ECB-128 (only one block per update) • AES-CFB(128, 192, 256) • RSAES-OAEP_SHA2-(224, 256, 384, 512)
Hash	• SHA1 • SHA2-(224, 256, 384, 512)
MacVerify/MacGenerate	• AES-CMAC-(128,192,256) • HMAC-SHA256 • SipHash-2-4
Random	• RNG (Self-shrinking-Generator) • AES-Ctr-DRBG-256

Supported Primitives from Crypto SW Generic

Csm service	Crypto Primitve
SignatureVerify	• RSASSA-PSS • RSASSA_PKCS1_v1_5 • ECDSA SecP256r1 • EdDSA
SignatureGenerate	• ECDSA SecP256r1 • RSASSA_PKCS1_v1_5 • EdDSA

- This is just an extract of the current Crypto driver, for a list which is up to date please look at the chapter “**Supported algorithms“** of the Crypto user manual.

Supported KeyManagement from Crypto SW Generic

Csm service	Crypto Primitve
RandomSeed	• AES-CTRDRBG • SSG
KeyDerive	• HMAC-SHA256 • SHA256
KeyExchange	• ECDH x25519 • ECDH ECCNIST secp256r1 • ECDH ECCNIST secp384r1
KeyElementSet	• CMAC Key Precalculation
SignatureGenerate	• EdDSA (Ed25519)
CertificateParse	• self-descriptive card verifiable (CV)
CertificateVerify	• RSASSA_PSS_SHA256

Crypto Driver – Crypto

HW solution

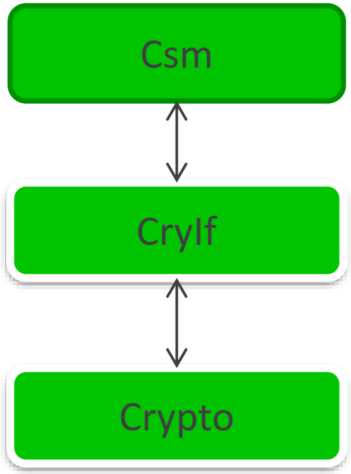
- Automotive Devices
  - Secure Hardware Extension (SHE)
  - Hardware Security Module (HSM)
- Crypto HW Driver module is a driver for interacting with security peripheral
- Secure storage and provision of keys
- Usually equipped with a true random number generator
- Provides Hardware Accelerators e.g. for AES
- Hardware Trust Anchors
  - secure data
  - provide crypto algorithms

SW solution

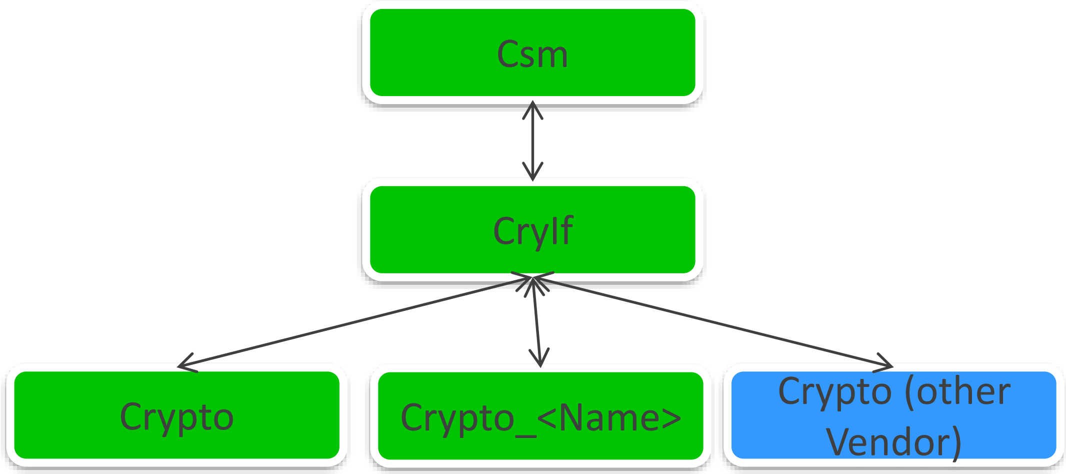
- SW implementation of crypto primitives
- Multiple instances of the SW module
- All possible algorithms can be implemented
- Easier to integrate
  - No dependency to a HSM
  - No need to integrate a special interface for HSM
- No secure storage
- No secure key provisioning protocol

Multi Instantiation

Single Instantiation



Multi Instantiation



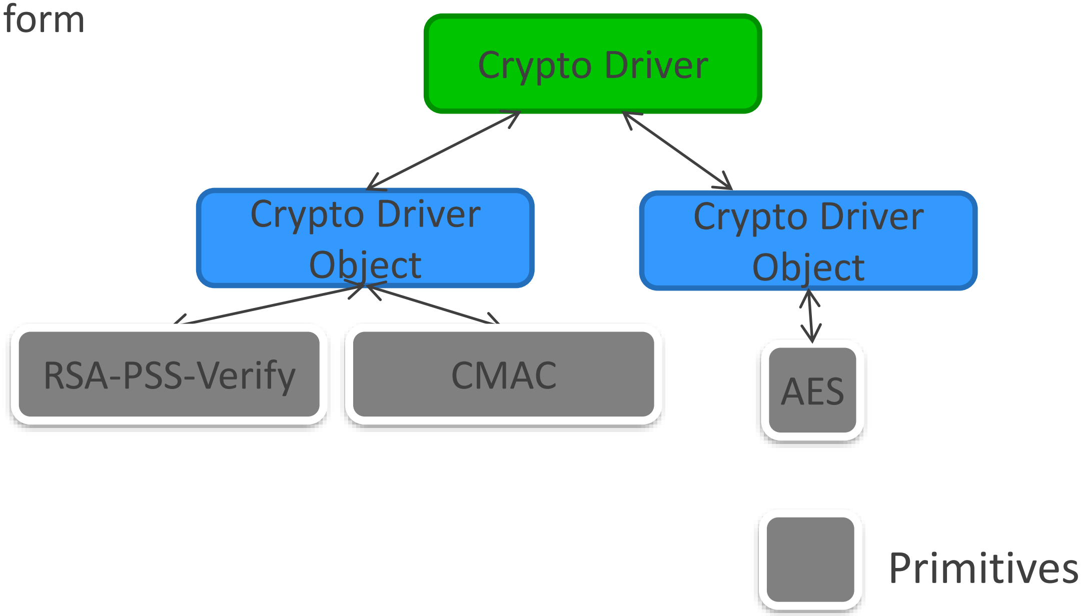
Files:
Crypto_1_<Name>.c

Files:
Crypto_xVIx_xAIx.c

Files:
Crypto_2_1.c

Crypto Driver Object

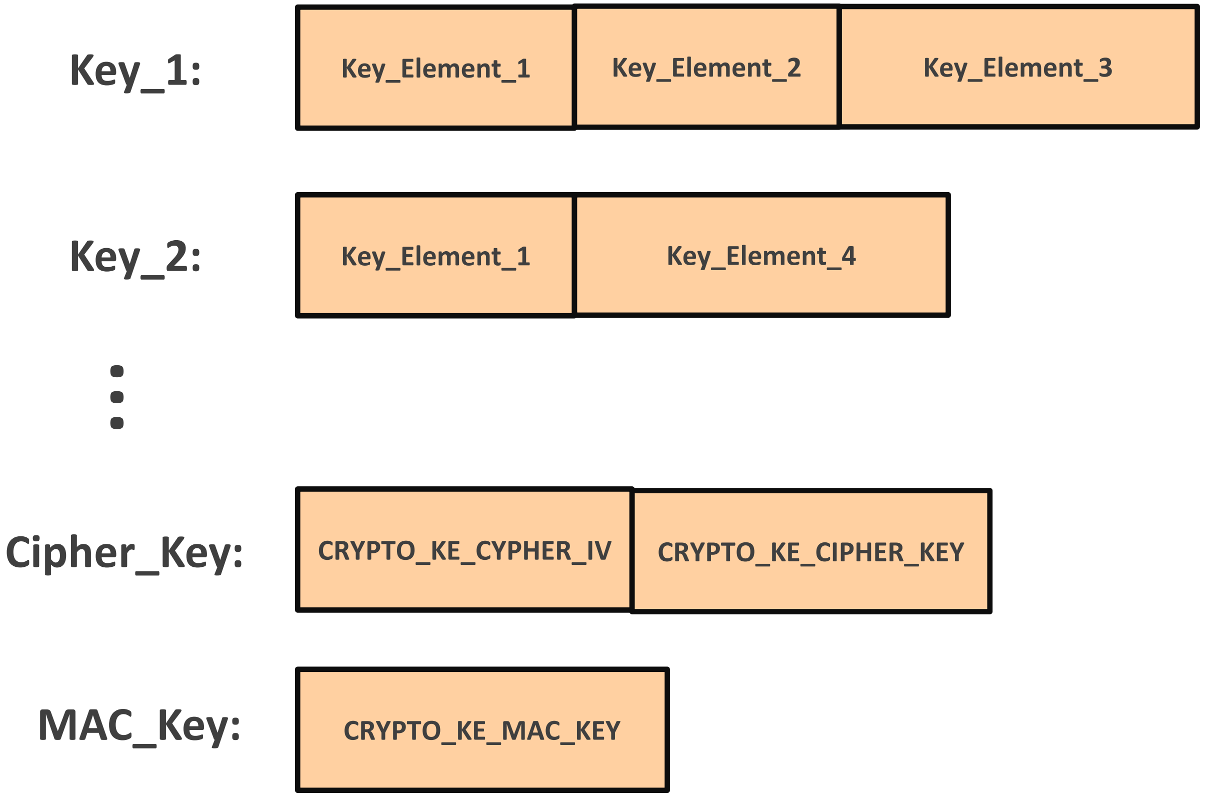
- Crypto Driver implements one or more* Crypto Driver Objects.
- Crypto Driver Object of one Crypt Driver (SW/HW) are independent form one another
- Allows parallel execution of jobs
- One Driver Object can offer different Primitives
  - But only one Crypto Primitive can be performed at a time per Driver Object
- Number of Crypto Driver Objects can be configured
- Only one workspace for each Crypto Driver Object
- One Csm queue/ CryIf Channgel per Crypto Driver Object



* this feature is currently in development

Key management

- A key element can be mapped to different key types
- A key type consists of one or more key elements
  - A key element can be mapped to different key types
- A key references a specific key type
  - A Job needs only one key reference
  - Single key elements can be updated, without changing the Csm configuration
- Key types are preconfigured for the provided primitives
  - Also own key elements can be configured
- Storage is done by the Crypto
  - Can be stored persistent in persistent memory
  - Key elements read/write access can be configured
  - Keys have a valid/invalid state
- Crypto Stack provides APIs to read or write a key



Key_1: Key_Element_1 | Key_Element_2 | Key_Element_3

Key_2: Key_Element_1 | Key_Element_4

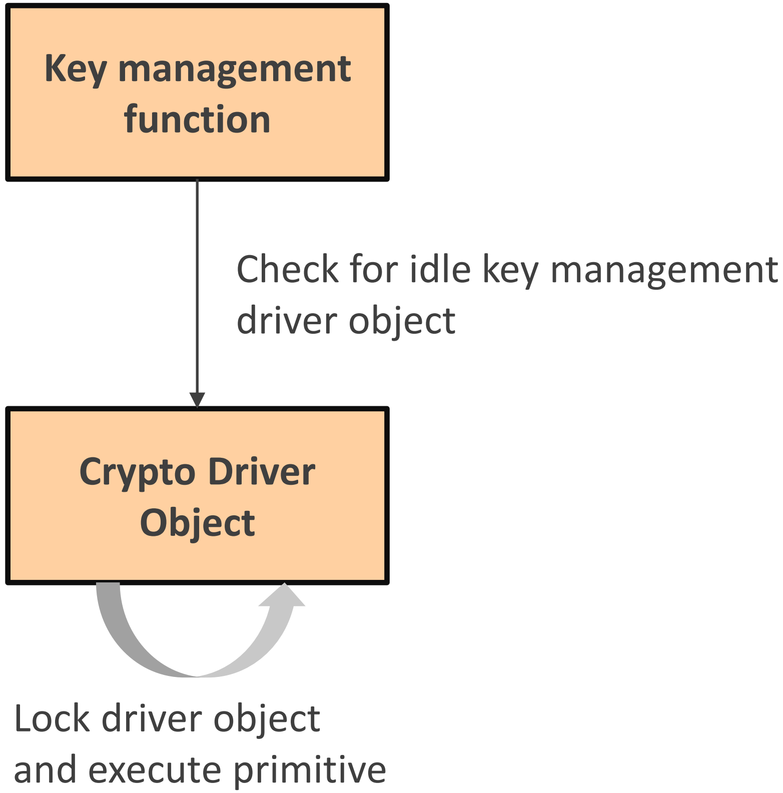
⋮

Cipher_Key: CRYPTO_KE_CYPHER_IV | CRYPTO_KE_CIPHER_KEY

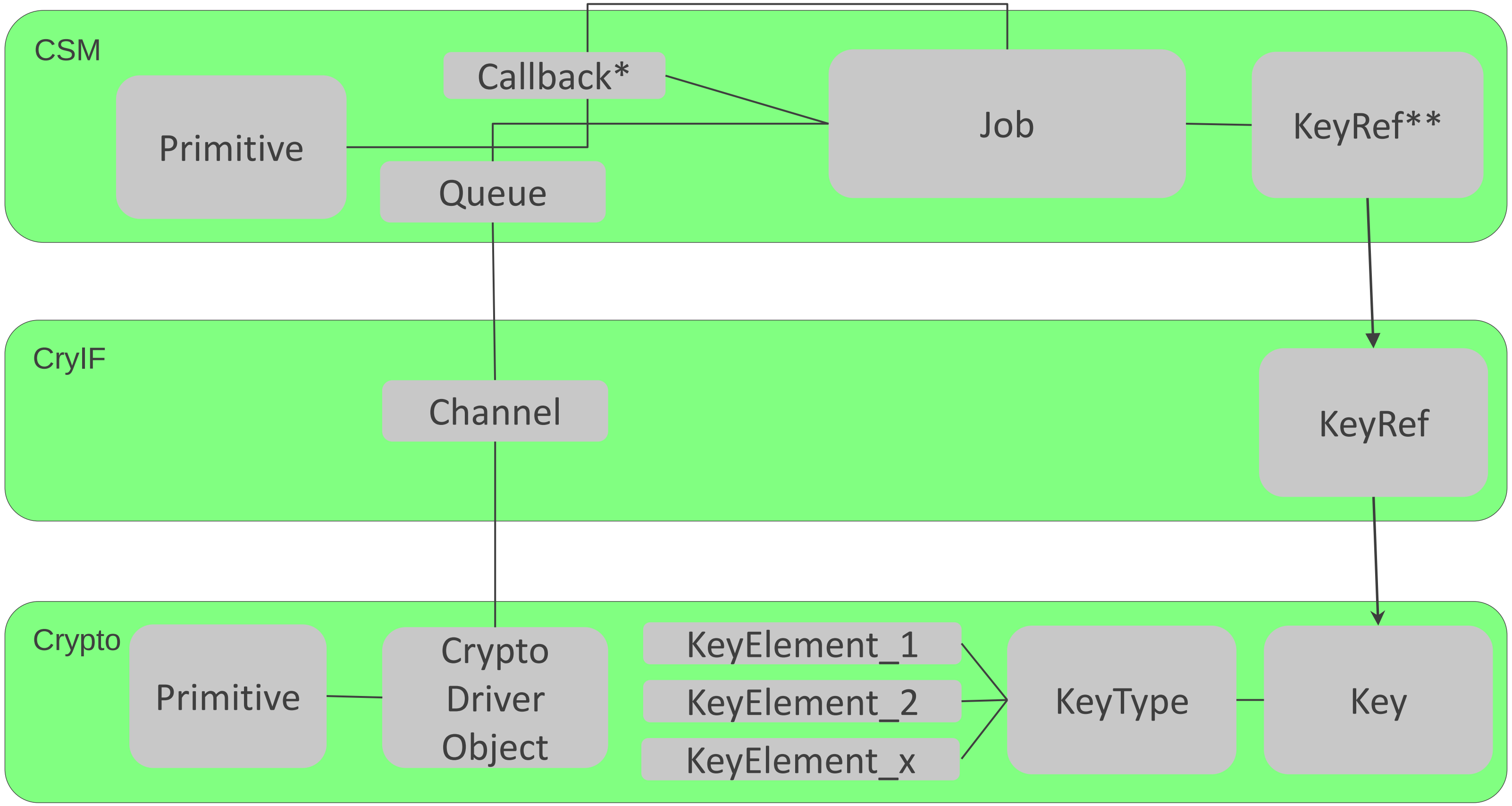
MAC_Key: CRYPTO_KE_MAC_KEY

Key management Crypto Driver Object

- Crypto Driver Objects can be enabled for key management
- Key management functions can lock those Driver Objects to run the primitives on
- This allows parallel calls of key management functions and primitives

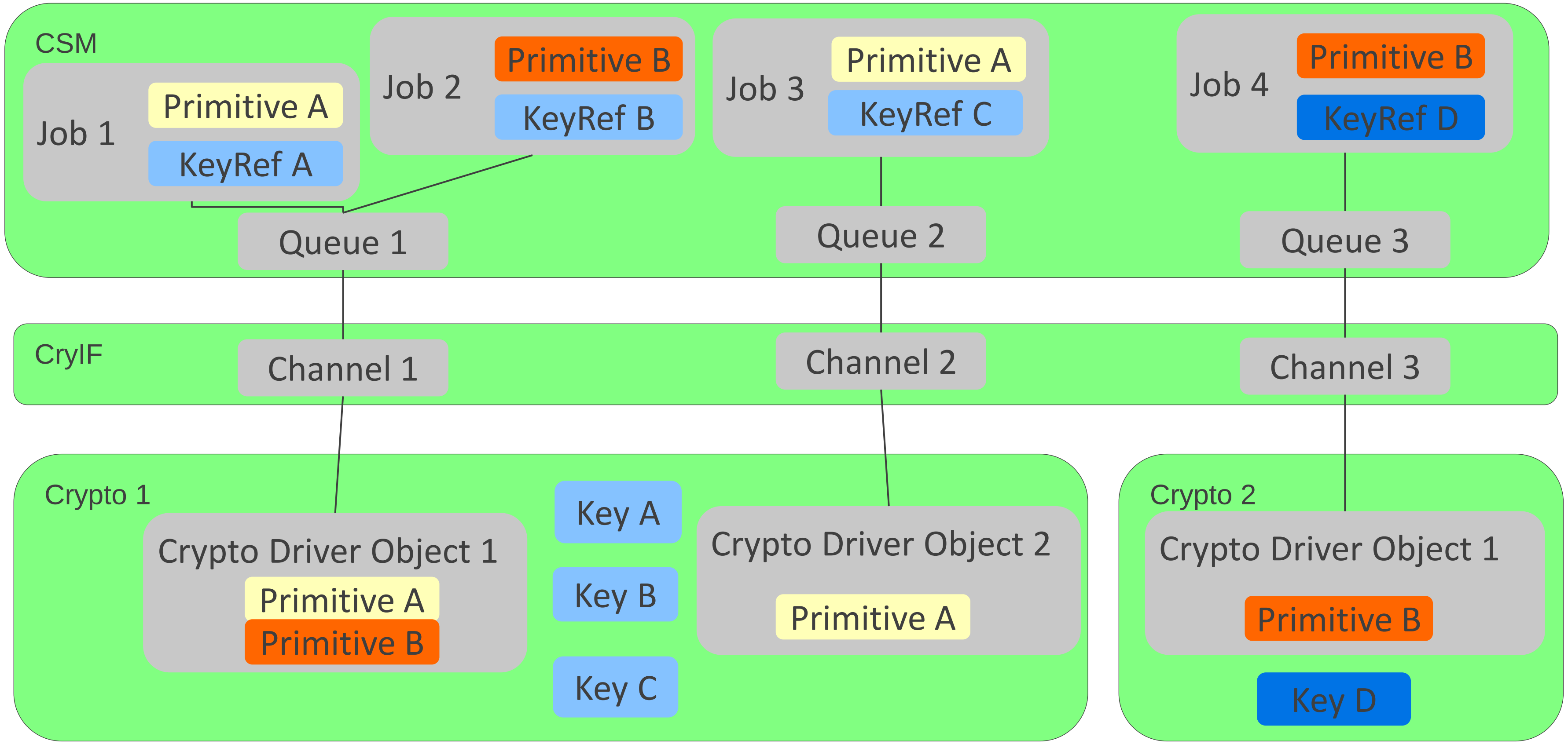

Key management function
Check for idle key management driver object
Crypto Driver Object
Lock driver object and execute primitive

Configuration dependencies



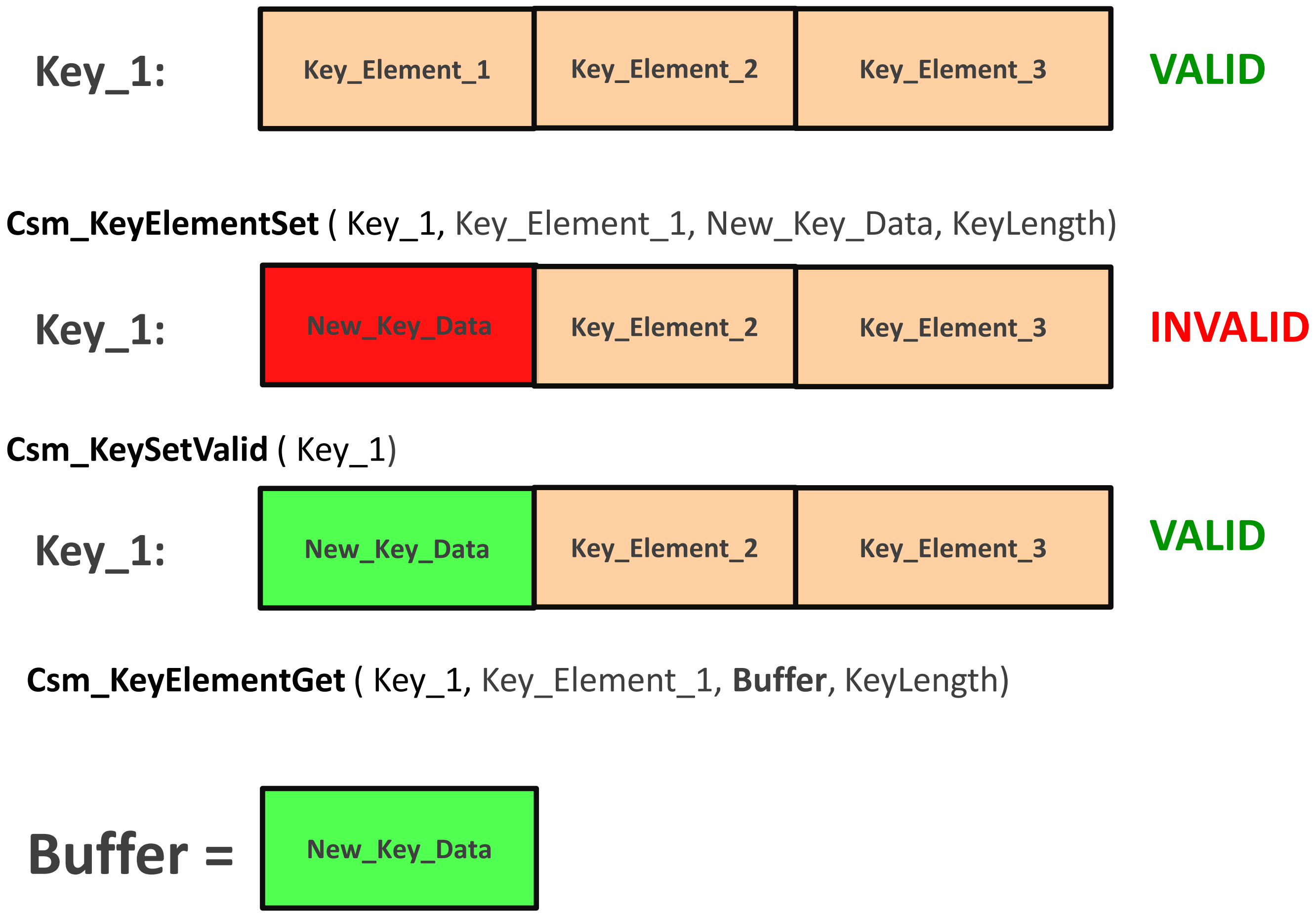
* Callbacks are only needed for asynchronous processing
**Some Primitives don‘t need a key e.g. Hash

Simple configuration



Set and Get the key

- **Csm_KeyElementSet**
  - Updates one key element
  - After a change to key it will be set to „INVALID“
- **Csm_KeySetValid**
  - Sets the key back to valid
- **Csm_KeyElementGet**
  - Extracts the key to a provided buffer
    (If the key has read rights configured)



Key_1: Key_Element_1 | Key_Element_2 | Key_Element_3 VALID

Csm_KeyElementSet (Key_1, Key_Element_1, New_Key_Data, KeyLength)

Key_1: 🟥 New_Key_Data | Key_Element_2 | Key_Element_3 INVALID

Csm_KeySetValid (Key_1)

Key_1: 🟩 New_Key_Data | Key_Element_2 | Key_Element_3 VALID

Csm_KeyElementGet (Key_1, Key_Element_1, **Buffer**, KeyLength)

Buffer = 🟩 New_Key_Data

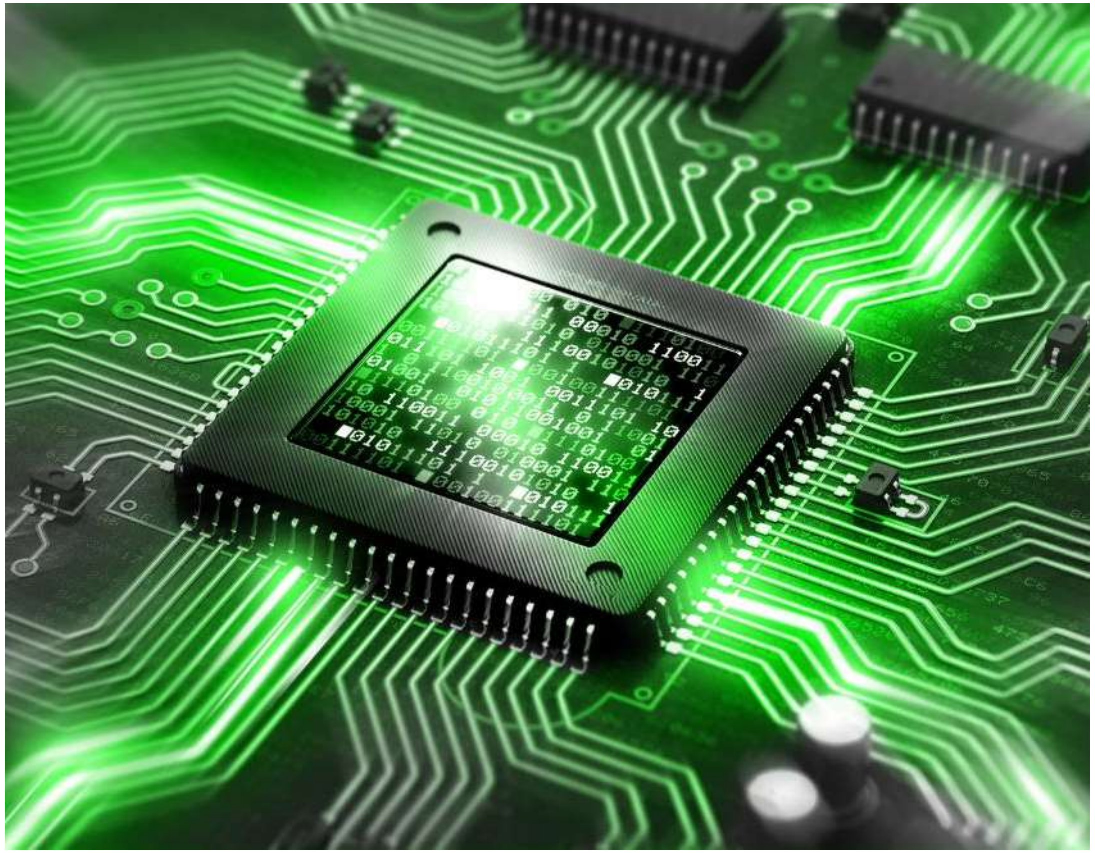
Queuing – Csm and Crypto

- **Csm**
  - Priority-based processing
  - Configurable size
  - Only asynchronous jobs will be enqueued
  - Synchronous jobs skip the queue, if the job priority is higher than the enqueued ones else a synchronous job returns busy.
    - If the Crypto Driver Object is busy then the synchronous job returns also with busy
  - If the queue is full the next job will be rejected
  - Multiple queues are possible
    - One instance of module Csm is allowed per Tresos project -> multiple configurations are not allowed
- **Crypto**
  - Priority-based processing
  - Configurable size
  - If the queue is full the next job will be rejected in Crypto
  - Each Crypto Driver Object can have its own queue

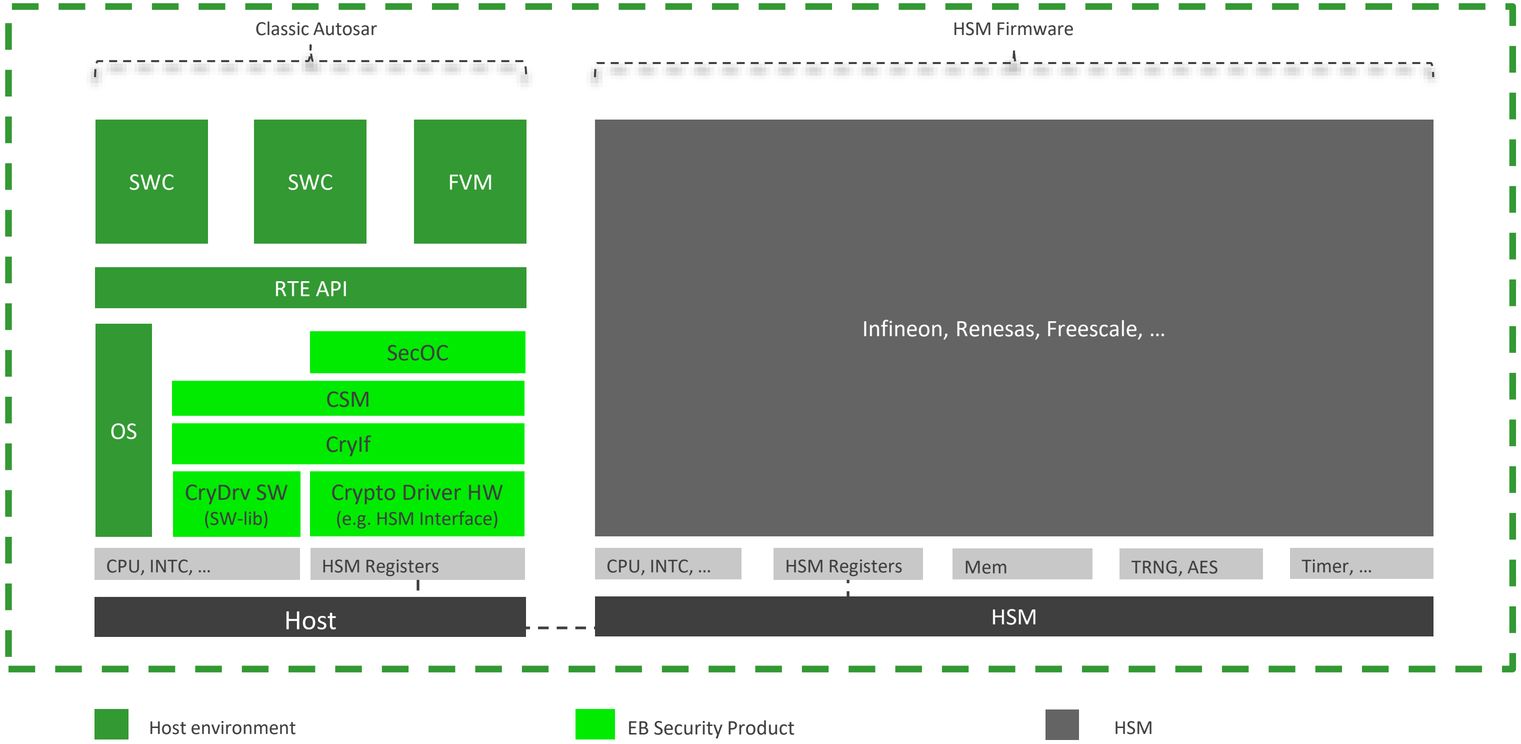
Root of trust

Hardware security module (HSM)

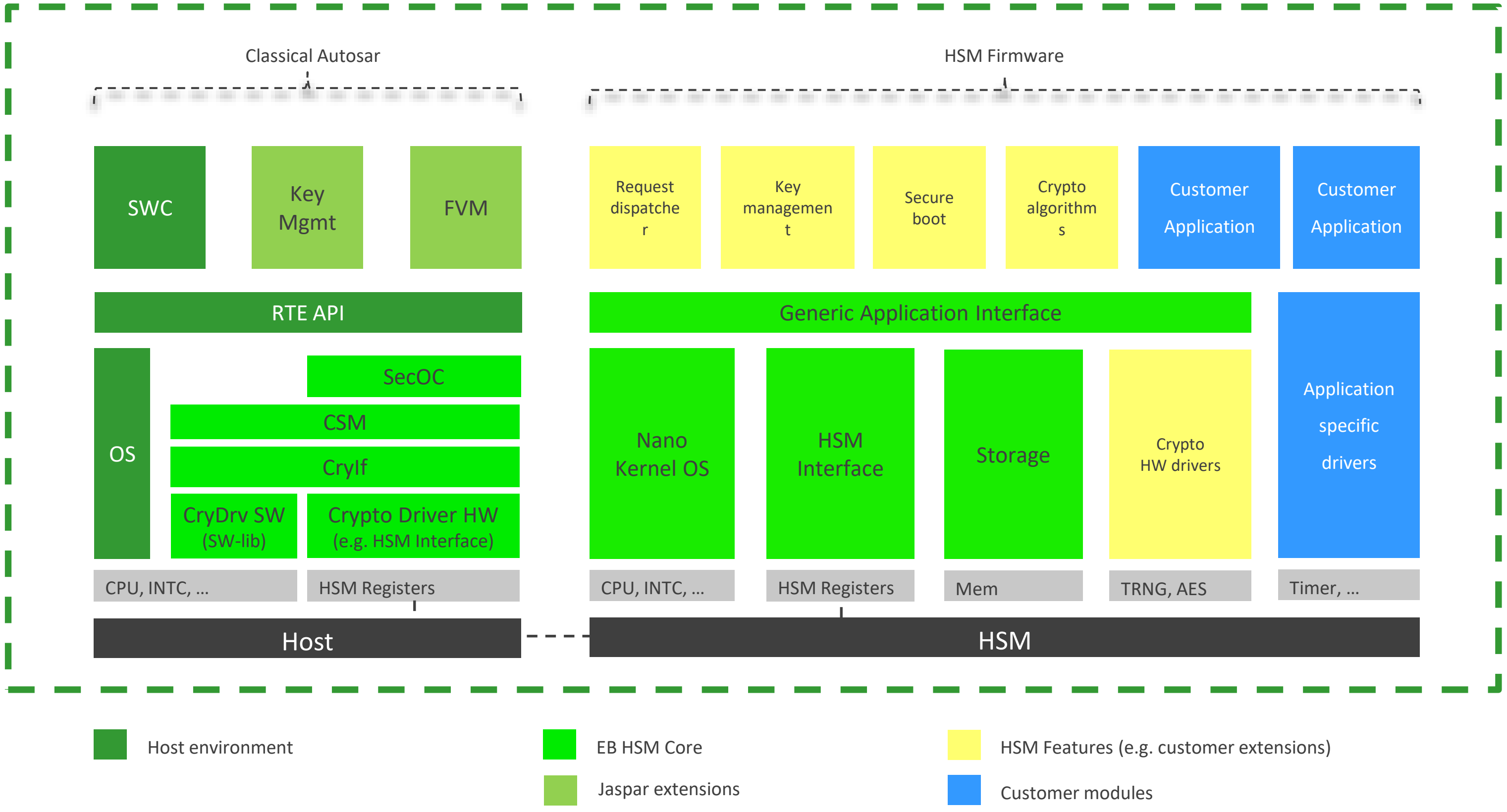
- Processor dedicated for cryptography and security
- Hardware accelerated cryptography
- Random number generator
- Support of secure boot mechanism
- Secure key store
- Programmable to run user specific applications



Crypto Driver for existing 3rd party HSM firmware



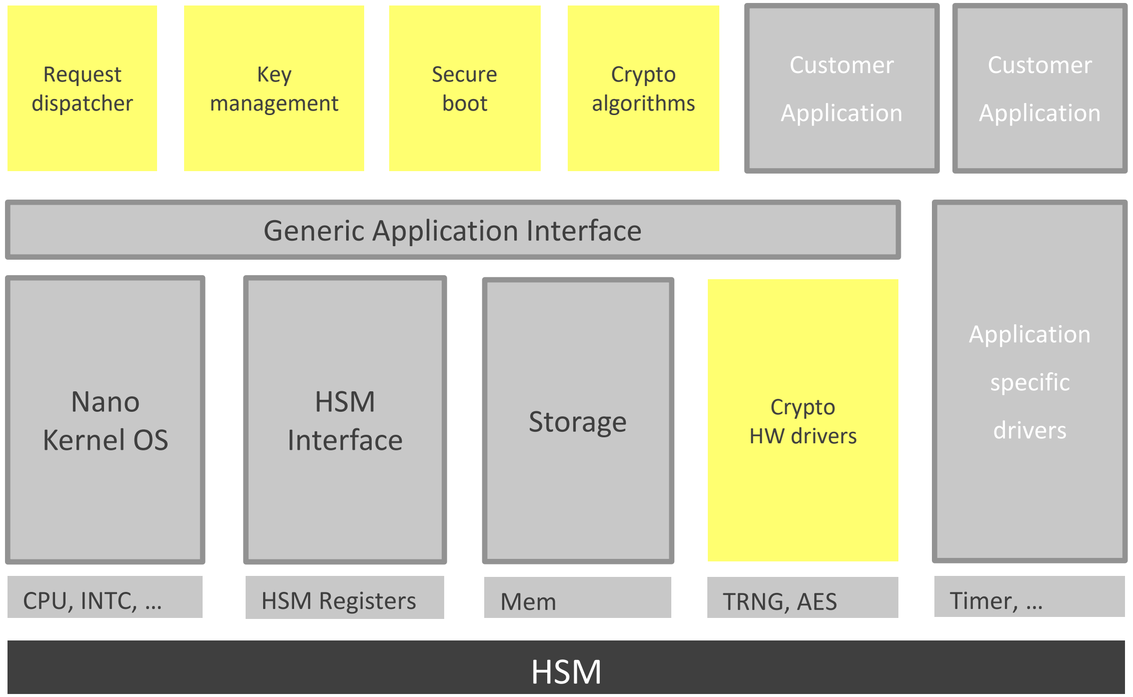
HSM software platform architecture



HSM Features (SHE+)

Provides:

- AES-128 supported cryptographic operations
  - Encryption/decryption
  - MAC generation/verification
- Random number generator
  - AES based pseudo random number generator
  - True random number generator
- Secured key storage
  - 20 (10) key slots
  - Key update protocol
- Secure boot
  - Key slot for BOOT_MAC_KEY and MAC are provided



Request dispatcher

Key management

Secure boot

Crypto algorithms

Customer Application

Customer Application

Generic Application Interface

Nano Kernel OS

HSM Interface

Storage

Crypto HW drivers

Application specific drivers

CPU, INTC, …

HSM Registers

Mem

TRNG, AES

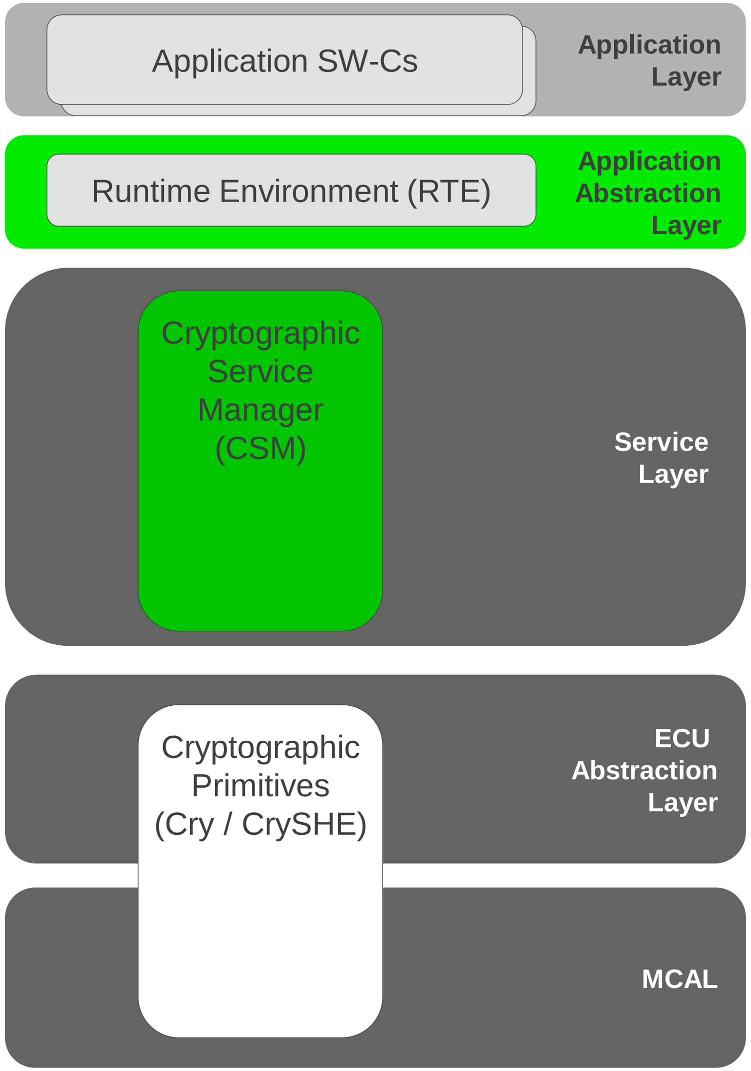
Timer, …

HSM

🟨 HSM Features (SHE+, customer extensions)

Crypto Service Manager (from AUTOSAR 4.0 – 4.2)

- CSM provides algorithm-independent interface to application
- Examples:
  - En-/Decryption
  - Signature Generation/Verification
  - MAC Generation/Verification
  - Key Wrapping / Key Extraction
- The actual cryptographic algorithms are contained in the Cry module


Application SW-Cs
Application Layer
Runtime Environment (RTE)
Application Abstraction Layer
Cryptographic Service Manager (CSM)
Service Layer
Cryptographic Primitives (Cry / CrySHE)
ECU Abstraction Layer
MCAL

CSM Principles (from AUTOSAR 4.0 – 4.2)

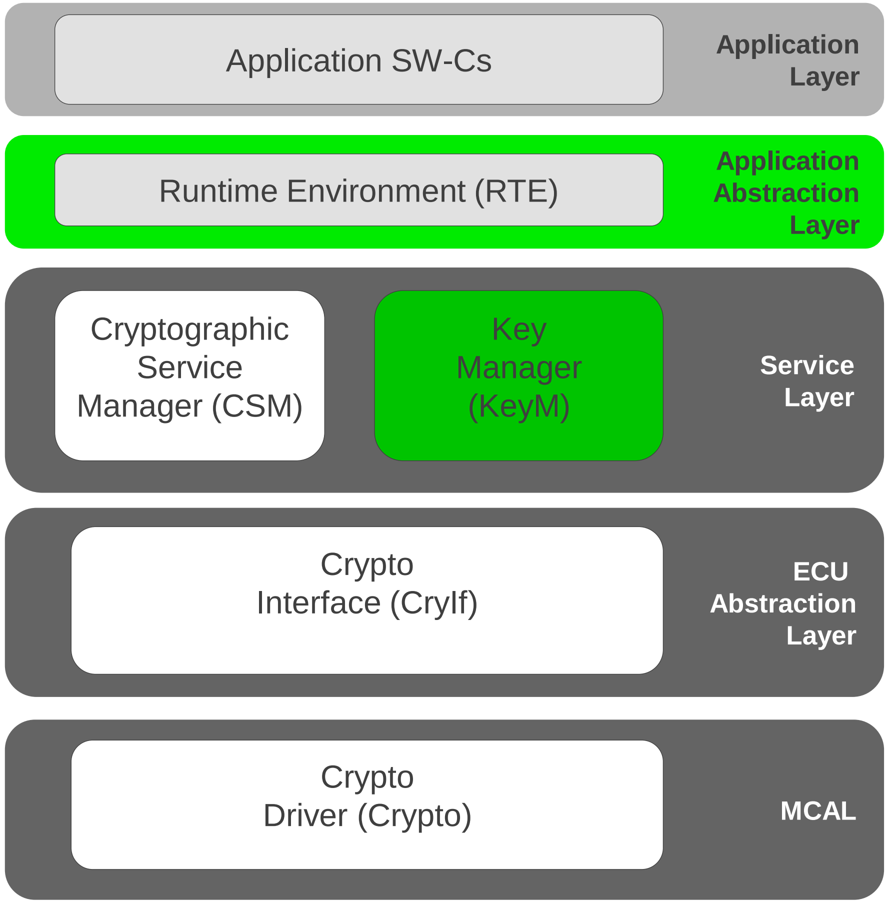
- All CSM interfaces follow the streaming paradigm
  - `Csm_<Service>Start`
    - initialize operation
  - `Csm_<Service>Update`
    - provide input data
    - retrieve output data
    - can be called multiple times
  - `Csm_<Service>Finish`
    - retrieve remaining output data
    - finish calculation
- Only one calculation can be performed at a time

KeyM: Key Manager

KeyM

Autosar KeyM (Key Manager)

- Introduced with Autosar 4.4
- Only the certificate submodule is provided
- Maintains certificates and/or certificate chains



KeyM

Autosar KeyM (Key Manager)

- Provides functions to get/set certificates and/or certificate chains
  - Certificates can be stored in chains eventually leading to a root certificate
  - In case of verification, the whole certificate chain will be checked before verifying the certificate
- Provides functions to Verify certificates considering its chain
  - To start the verification process all certificates of the chain have to be parsed to access the certificate elements
  - Checks are performed on the elements (e.g. Checking for correct version, issuer, ...)
  - If parsing of the whole chain was successful, the verification using the signatures is triggered by calling the CSM
- Provides functions to get specific elements inside the certificates
- Provides parsing of certificates in the background
  - To start the verification process all certificates of the chain have to be parsed
  - If there is no verification request present, the KeyM Module will start parsing previously unparsed certificates in its storage
  - This will speed up future verification requests

SecOC: Secure Onboard Communication

Elektrobit

Secure Communication Terminology

- **Authentic PDU (non-secured PDU, normal message)**

  PDU generated by the authentic sender, containing sensitive information

- **Secured PDU**

  PDU that is protected in a way such that confidentiality and integrity of the payload is protected

- **Message Authentication Code (MAC)**

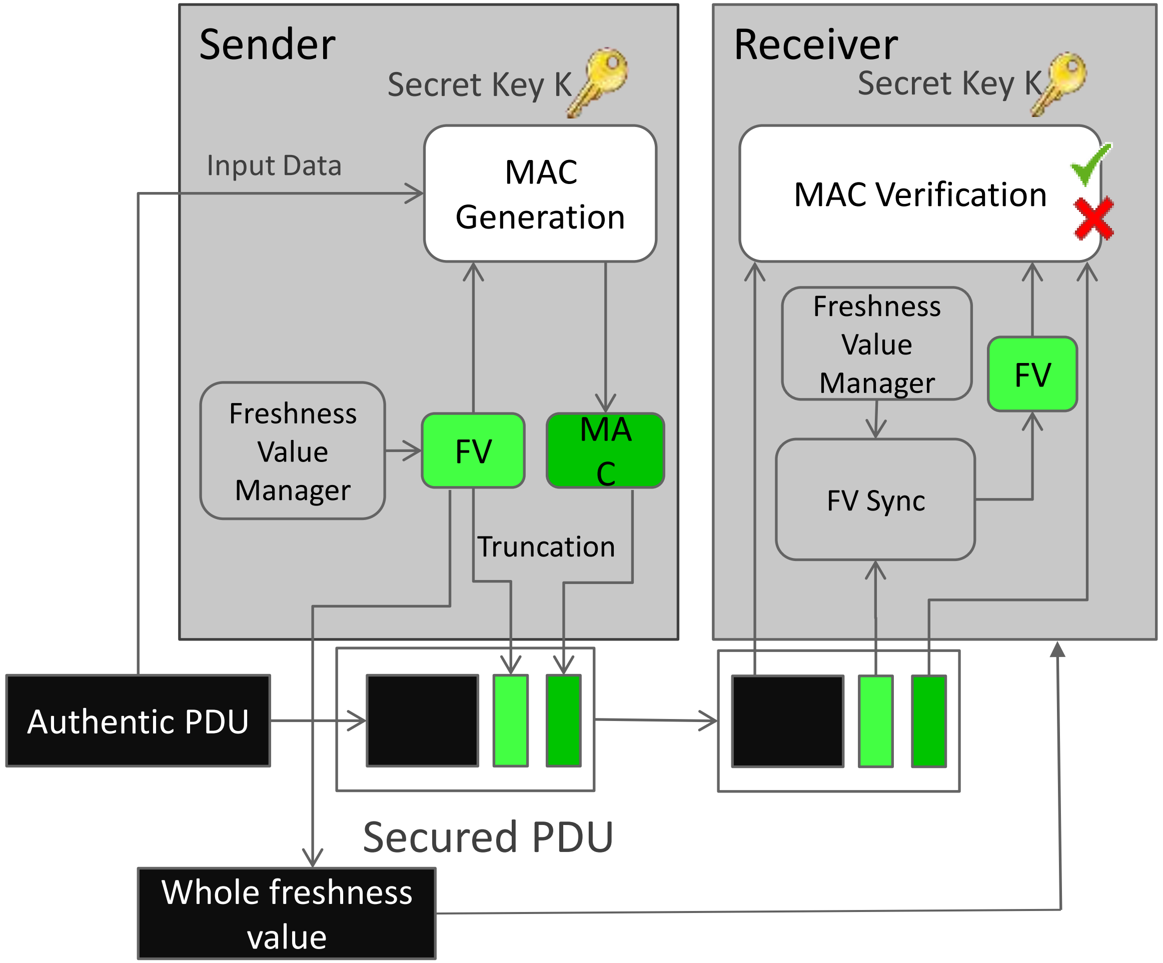
  Checksum that is generated using a cryptographic algorithm based on a secret key

- **Freshness Value**

  Used for preventing replay attacks, i.e. the repetition of previously recorded messages

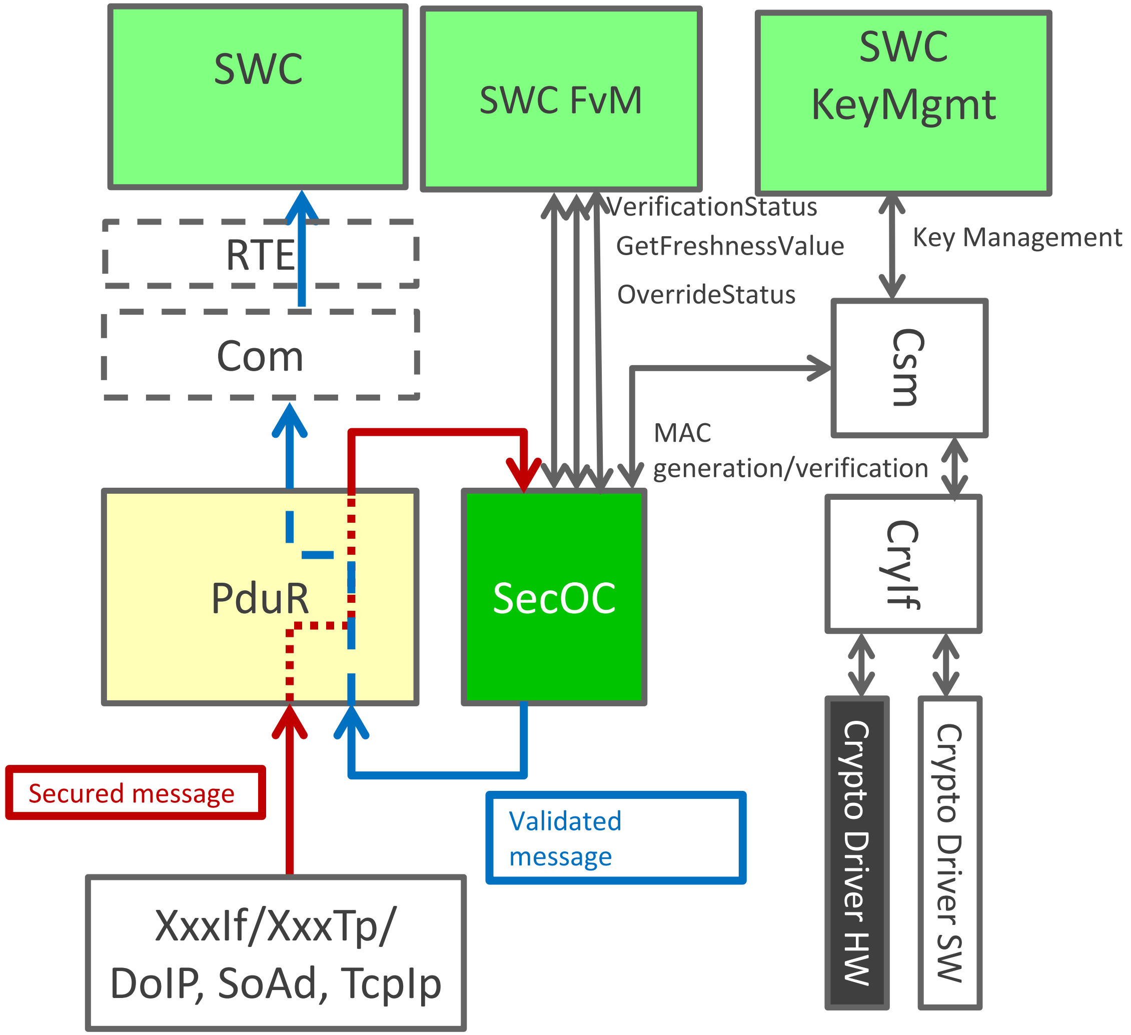
Overview

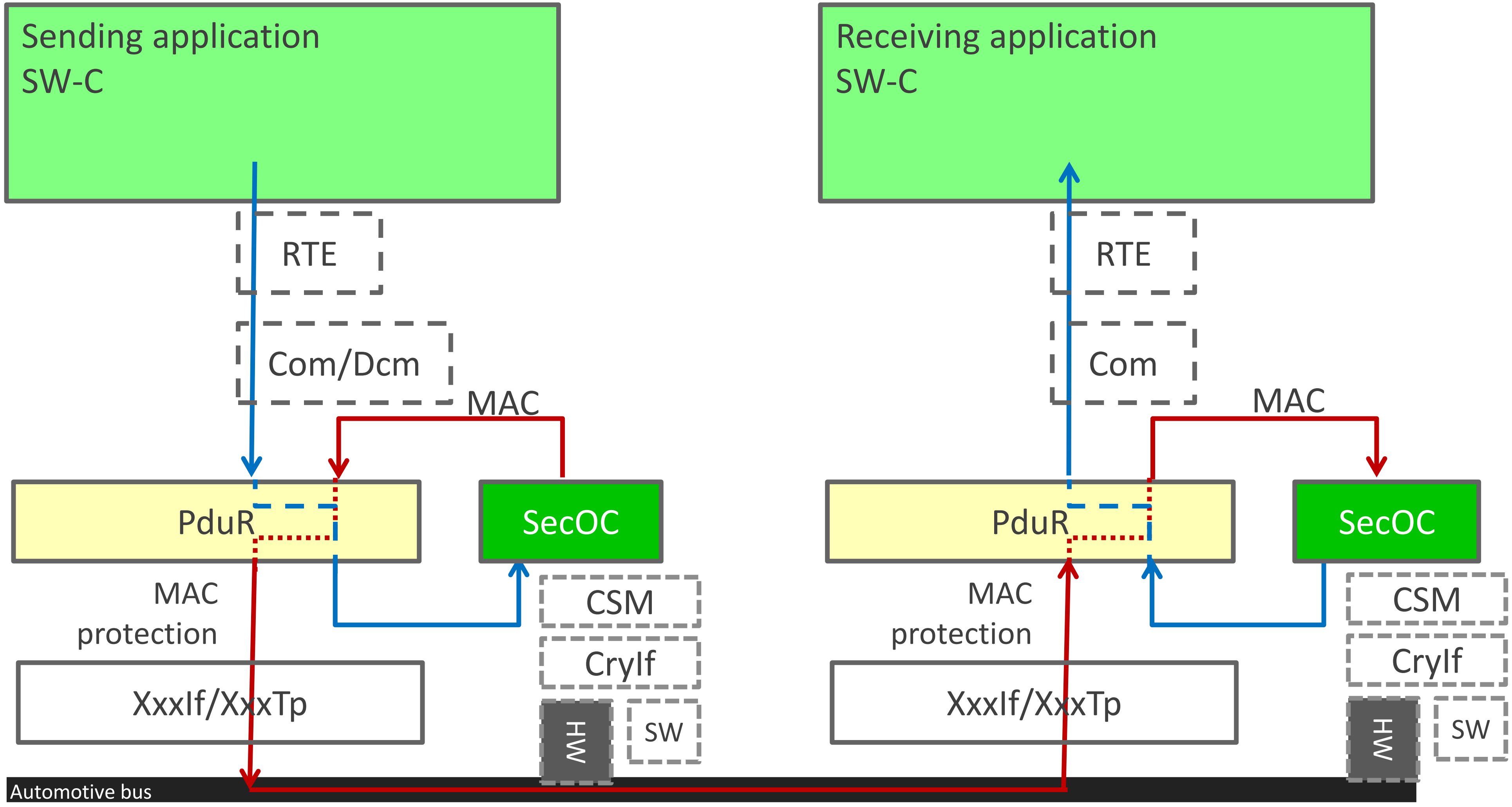
- EB is an active member of the AUTOSAR concept group „Secure Onboard Communication – SecOC“
- Main Features
  - Security protection on bus level
  - Integrity is ensured using MAC
  - Replay protection with freshness value
  - Protection/Verification on PduR level
  - Independent from Bus or protocol
  - If verification fails on receiver side, PDU is not provided to PduR → timeout on upper layer



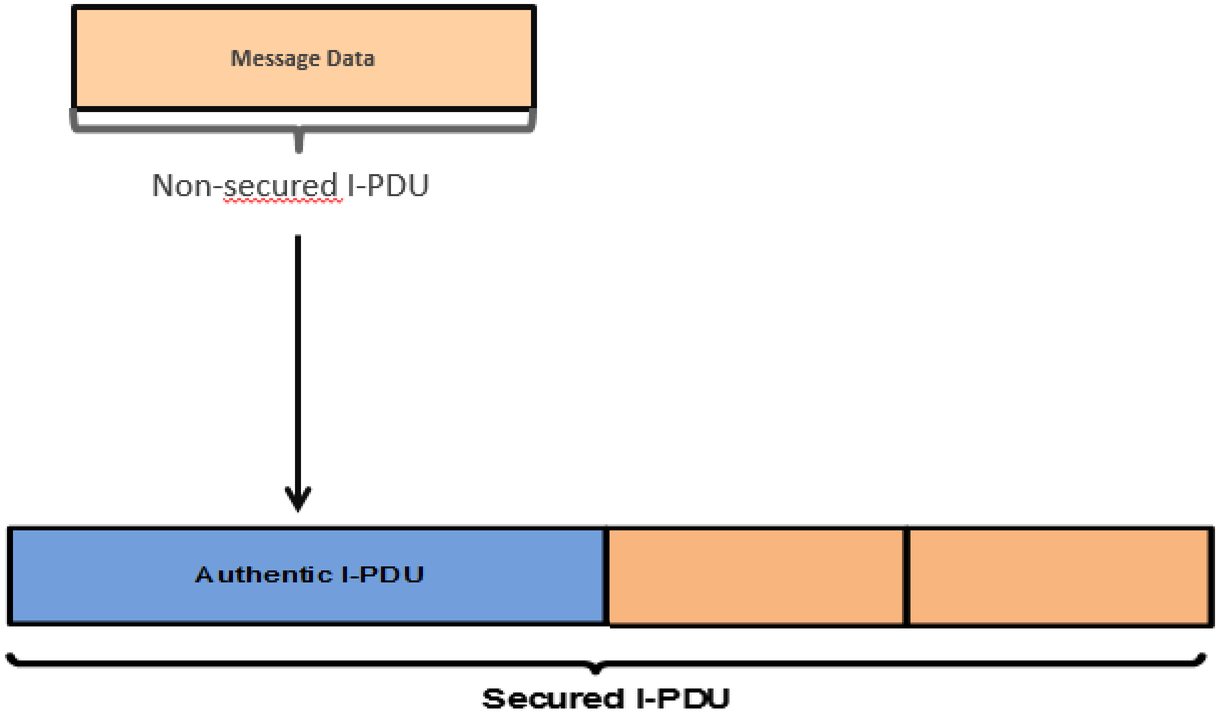
Overview

- SW-C
  - Sends/Receives Authentic PDU
  - Key Management
- PduR
  - Routes secured PDU
- SecOc
  - Generates/Verifies a secured PDU
- CSM
  - Provides interface for cryptographic primitives
- Crypto
  - Provides CMAC primitives

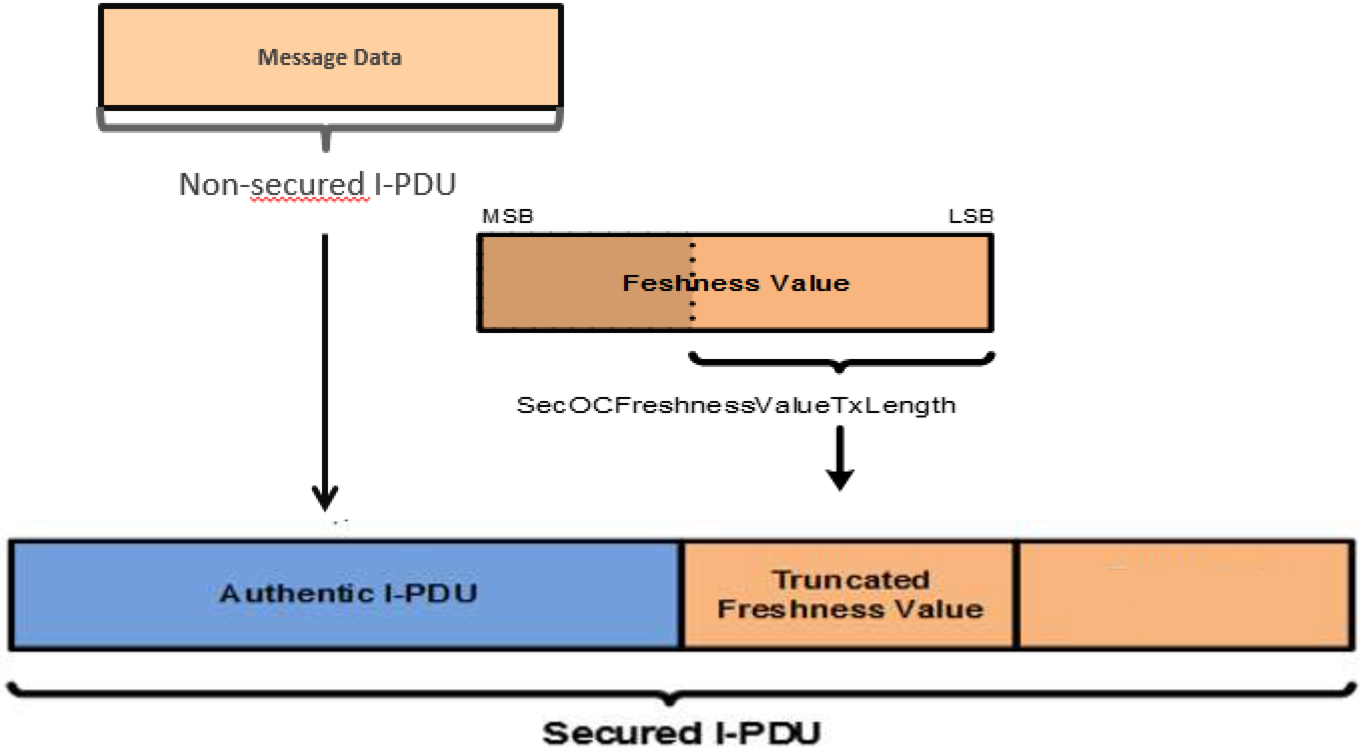



Sending application
SW-C
RTE
Com/Dcm
MAC
PduR
SecOC
MAC
protection
CSM
CryIf
HW
SW
XxxIf/XxxTp
Automotive bus
Receiving application
SW-C
RTE
Com
MAC
PduR
SecOC
MAC
protection
CSM
CryIf
HW
SW
XxxIf/XxxTp

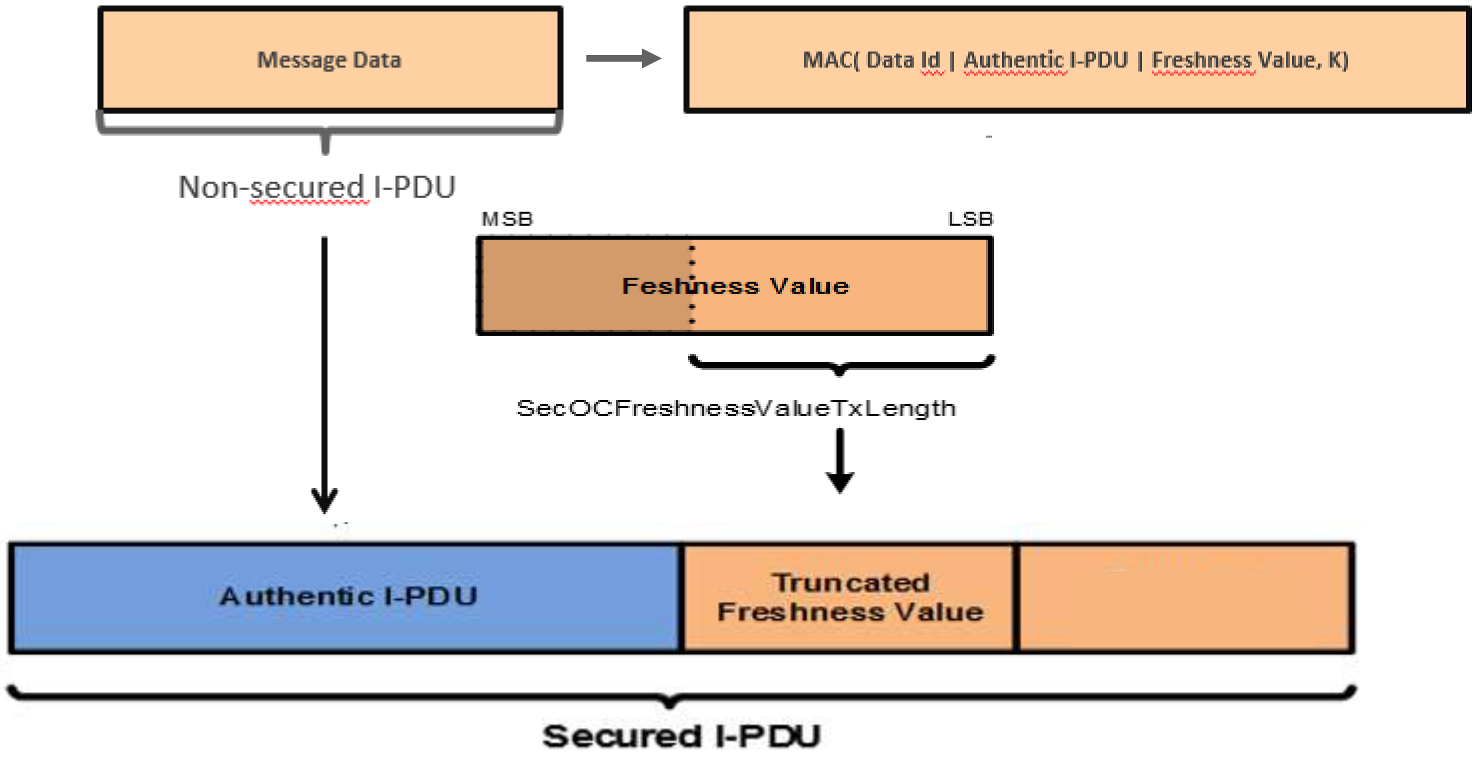
Composition of a Secured PDU



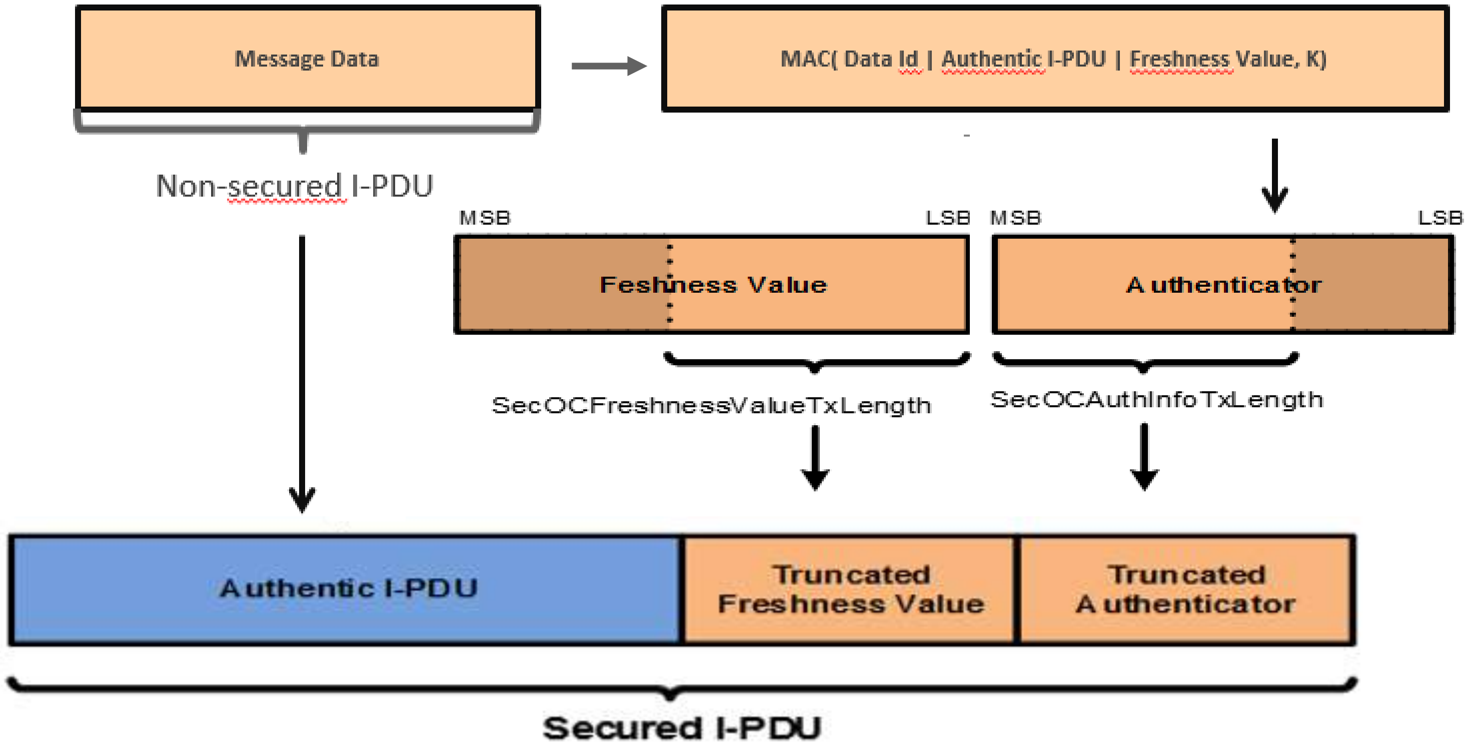
Composition of a Secured PDU



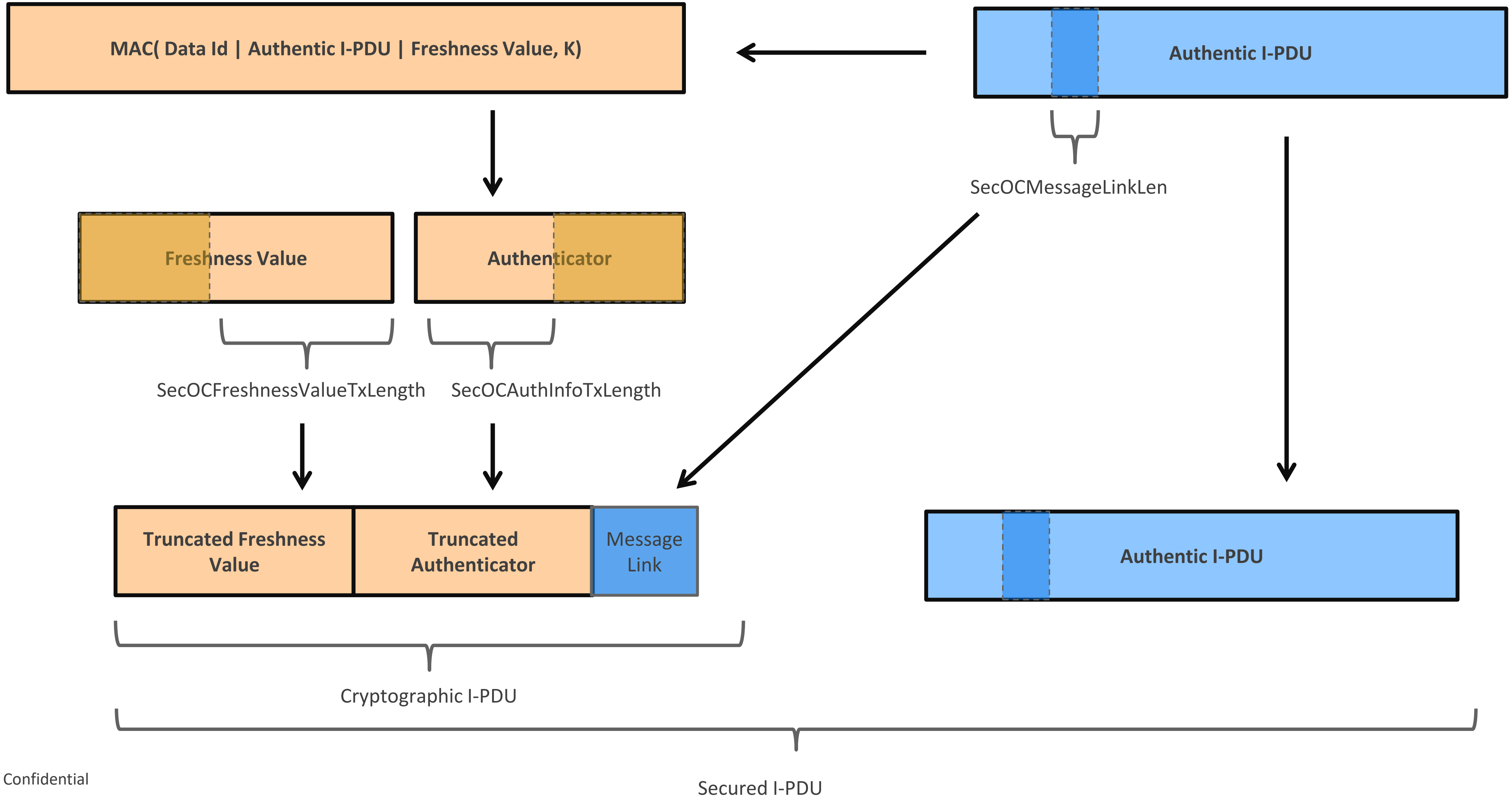
Composition of a Secured PDU


Message Data
MAC(Data Id | Authentic I-PDU | Freshness Value, K)
Non-secured I-PDU
MSB
LSB
Feshness Value
SecOCFreshnessValueTxLength
Authentic I-PDU
Truncated Freshness Value
Secured I-PDU

Composition of a Secured PDU



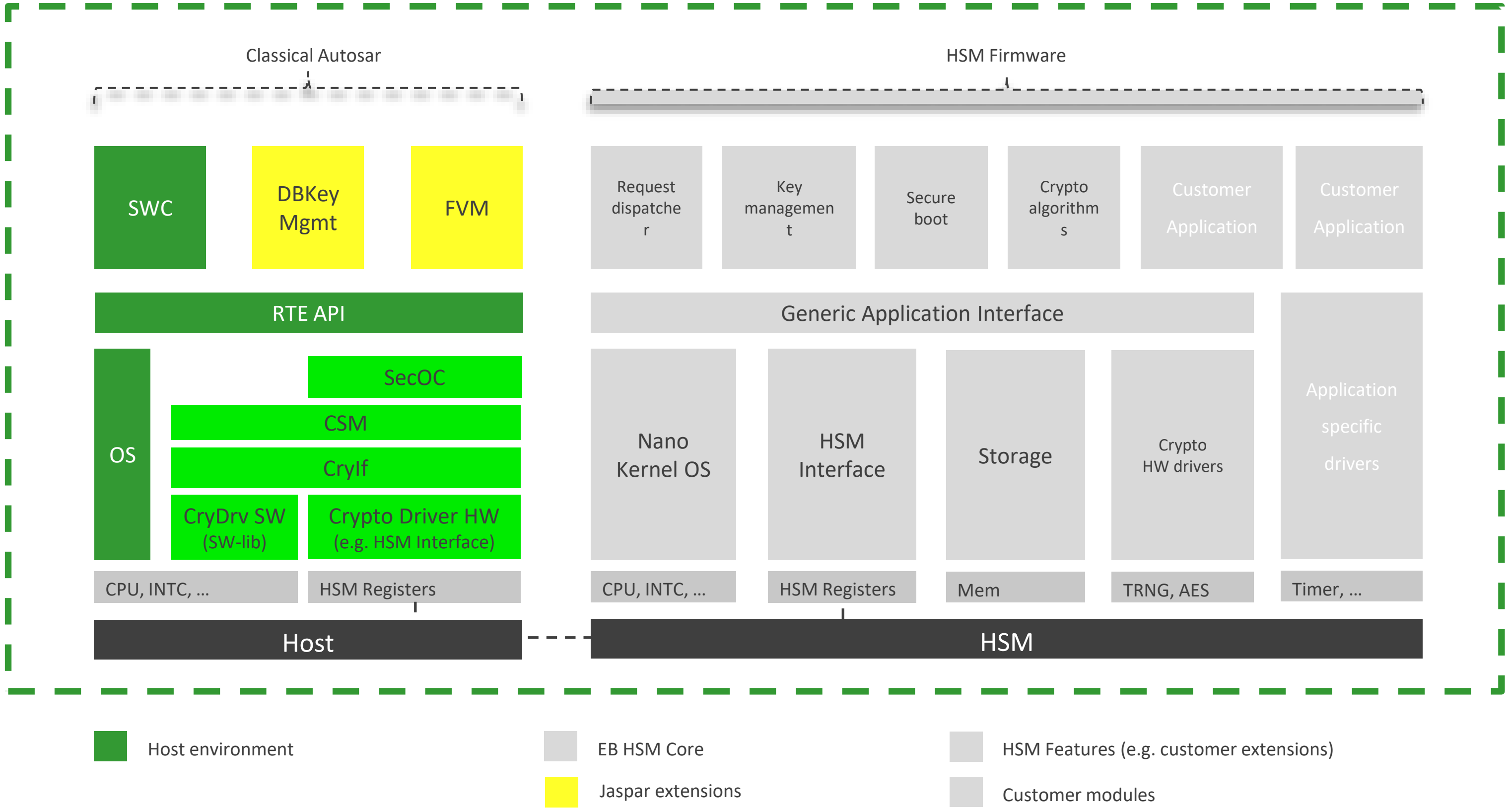
Secured PDU Collection



Limitations

- AUTOSAR SecOC only provides basics
- Application has to handle
  - Resynchronization of freshness counter
  - Persistent storage of freshness counter (e.g. when to store, how often)
  - Key distribution
  - Error handling/recovery strategy

EB’s Jaspar extensions



Freshness Value

- Freshness value can be understood as some sort of counter or time stamp to detect and counteract attacks such as replay, spoofing, and tampering of PDU-based communication
- Typically truncated before sent on a bus
- Non-truncated value generated and distributed in particular message by a module
  - Using c-function
    - User can implement easy counter to something more complex
  - Using SW-C
    - Basically time stamp of when freshness value sender (ECU which generates it) was waken up
    - Sync mechanism
      - Process Freshness Value message received from freshness value sender and based on own internal and received freshness value perform synchronization
    - Check mechanism
      - E.g. if received freshness value is within specified time window

Freshness Value Manager - FvM

- Not defined by AUTOSAR
- Mentioned in the SecOC specification with three proposals how to realize a FvM

1. Freshness Value FV is based on a Freshness Counter.
   Freshness Counter is provided for each Freshness Value ID.
   Freshness Counter is incremented prior to providing the FV.
   Freshness Counter on both sides (receiver and sender) should be incremented synchronously.
2. Freshness Value FV is based on a Freshness Timestamp.
   Global synchronized time can be used.
3. Construction of Freshness value from decoupled counters.
   Master/slave approach for the FvM.
   Master sends synchronization messages to slaves.

- EB provides the 3’rd proposed solution, which is compatible with JASPAR

Freshness Value Manager - FvM

- Not defined by AUTOSAR
- **Mentioned in the SecOC specification with three proposals how to realize a FvM**

1. Freshness Value FV is based on a Freshness Counter.
   Freshness Counter is provided for each Freshness Value ID.
   Freshness Counter is incremented prior to providing the FV.
   Freshness Counter on both sides (receiver and sender) should be incremented synchronously.
2. Freshness Value FV is based on a Freshness Timestamp.
   Global synchronized time can be used.
3. Construction of Freshness value from decoupled counters.
   Master/slave approach for the FvM.
   Master sends synchronization messages to slaves.

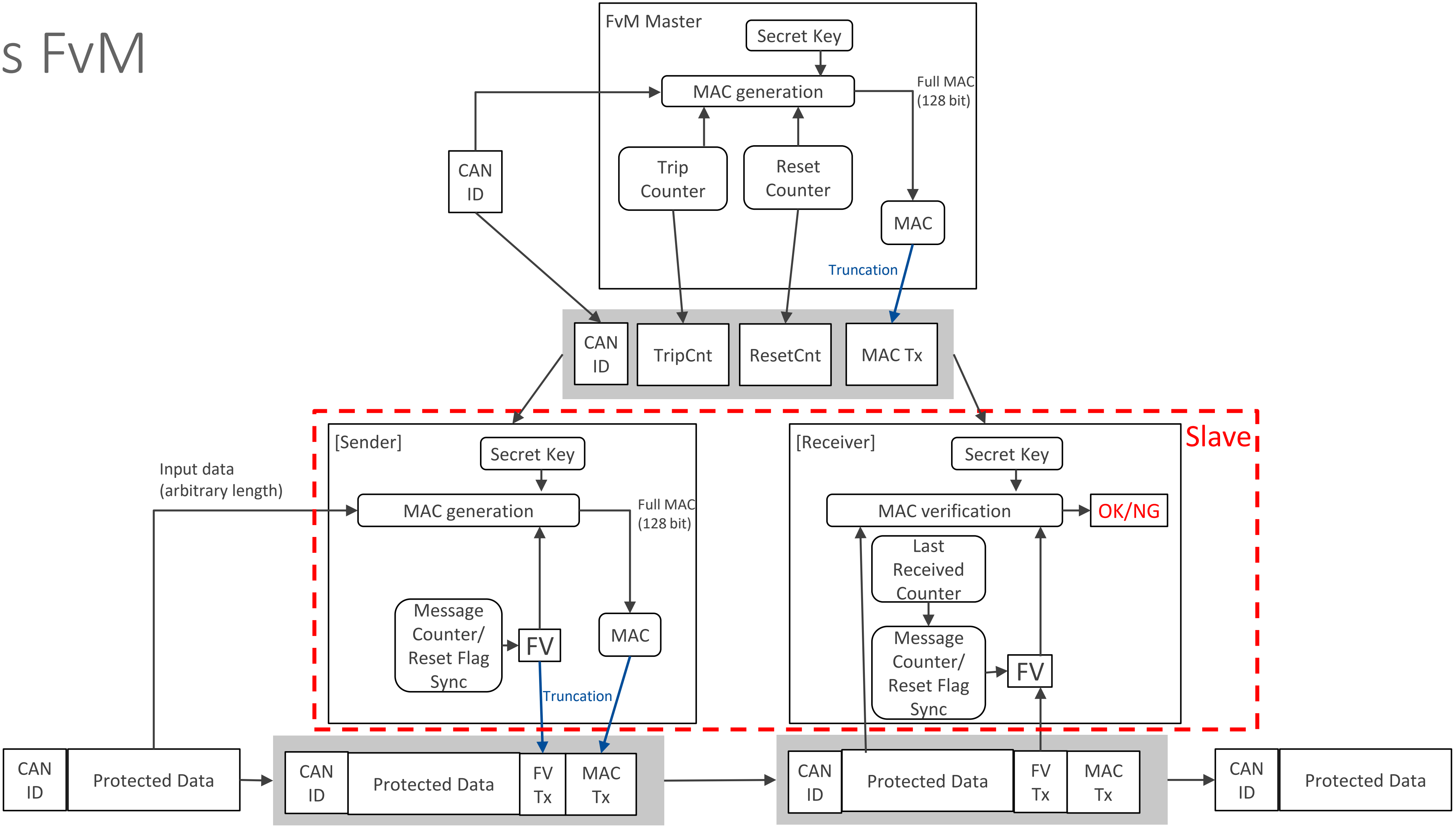
- **EB provides the 3’rd proposed solution, which is compatible with JASPAR**

EB‘s FvM

FvM (Freshness Value Manager)

- Manage freshness values
- Master and Slave
- Sending and Receiving ECUs
- Send/Receive sync messages
- Provide freshness values to SecOC
- Override SecOC verification status in special cases (prototype key, during startup)
- Store Freshness Values persistently
- Report events

EB‘s FvM


FvM Master
Secret Key
MAC generation
Full MAC
(128 bit)
CAN
ID
Trip
Counter
Reset
Counter
MAC
Truncation
CAN
ID
TripCnt
ResetCnt
MAC Tx
Slave
[Sender]
Secret Key
Input data
(arbitrary length)
MAC generation
Full MAC
(128 bit)
Message
Counter/
Reset Flag
Sync
FV
MAC
Truncation
[Receiver]
Secret Key
MAC verification
OK/NG
Last
Received
Counter
Message
Counter/
Reset Flag
Sync
FV
CAN
ID
Protected Data
CAN
ID
Protected Data
FV
Tx
MAC
Tx
CAN
ID
Protected Data
FV
Tx
MAC
Tx
CAN
ID
Protected Data

EB‘s DBKeyM

DBKeyM (Diagnostic-based Key Manager **)**

- Jaspar based Key manager extension implmented as an SW-C
- Maintains keys in the Crypto Stack
- Handles key management tasks via diagnostic interfaces
- Basic Functionality:
  - Update Keys
  - Verify Keys

- **Note: Meanwhile AUTOSAR Introduced the KeyM as a new BSW module with AUTOSAR 4.4 →** More features e.g. for Certificate handling.
- At this moment both solutions are availble: the DBKeyM and the AUTOSAR KeyM

Rte Usage

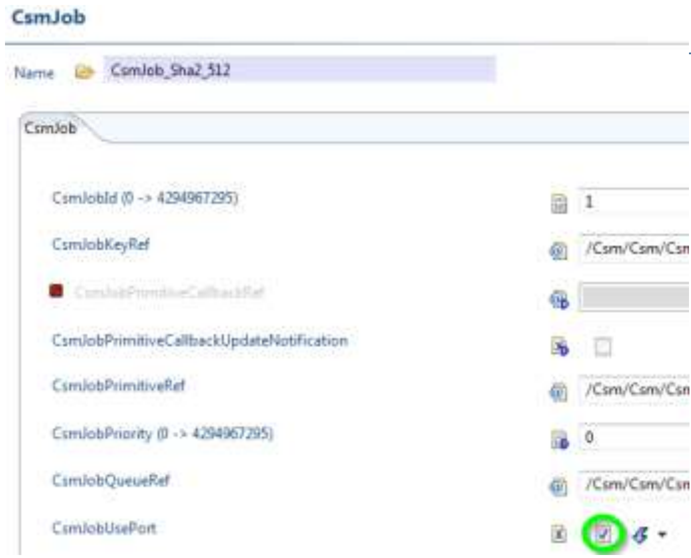
Rte interface

To module Csm

- For every job (only singlecall is defined)
- For every key

To module SecOC

- Freshness value
- Verification status

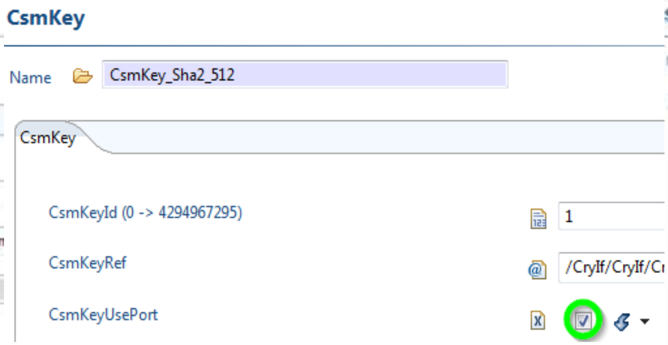


CsmJob

Name: CsmJob_Sha2_512

CsmJob

CsmJobId (0 -> 4294967295)	1
CsmJobKeyRef	/Csm/Csm/Csm
CsmJobPrimitiveCallbackRef	
CsmJobPrimitiveCallbackUpdateNotification	□
CsmJobPrimitiveRef	/Csm/Csm/Csm
CsmJobPriority (0 -> 4294967295)	0
CsmJobQueueRef	/Csm/Csm/Csm
CsmJobUsePort	☑

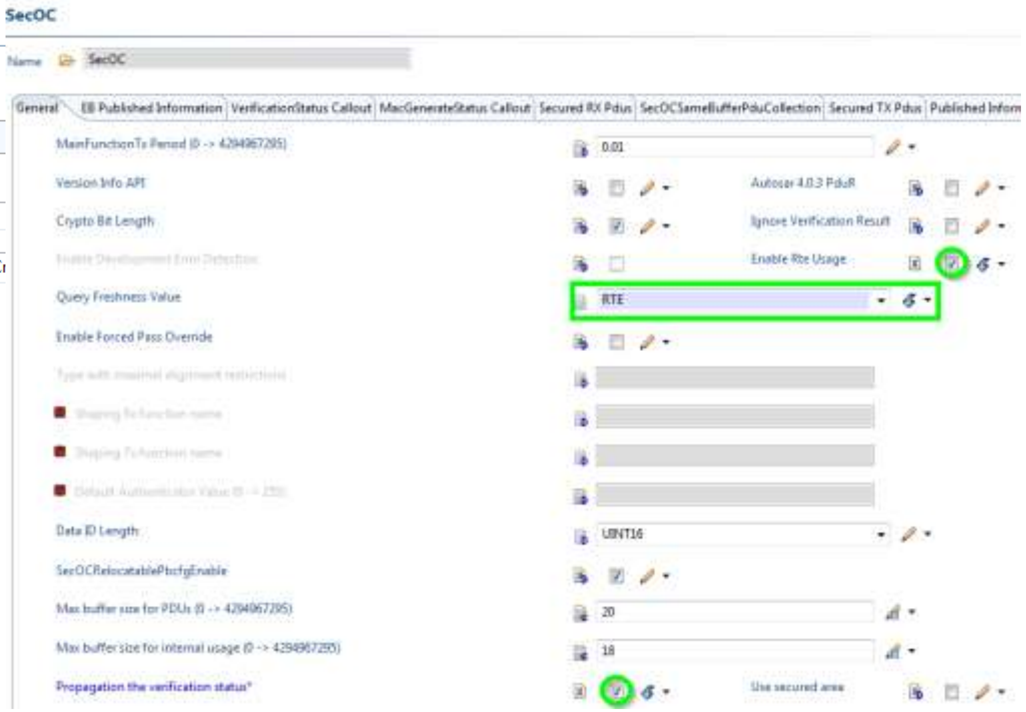


CsmKey

Name: CsmKey_Sha2_512

CsmKey

CsmKeyId (0 -> 4294967295)	1
CsmKeyRef	/CryIf/CryIf/Cr
CsmKeyUsePort	☑



SecOC

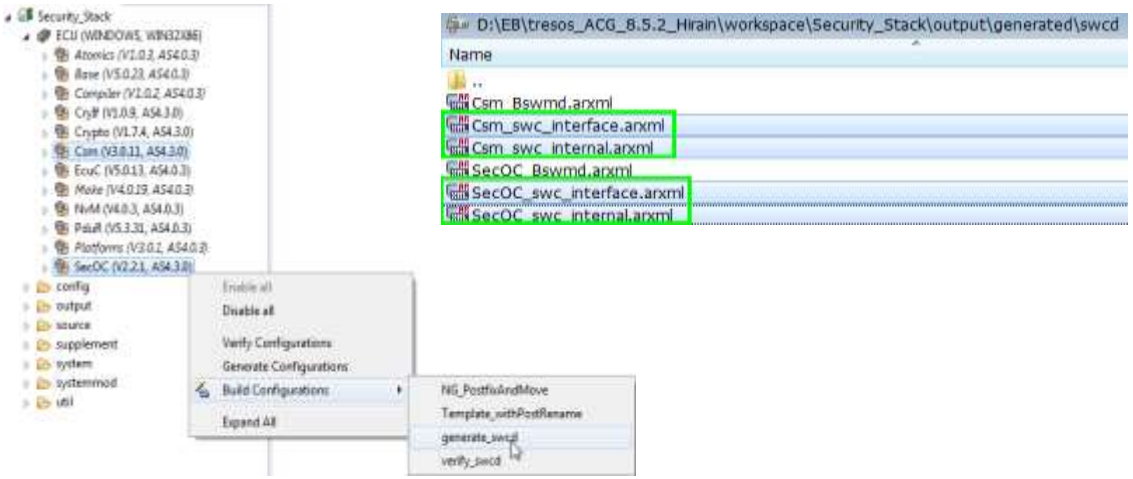
Name: SecOC

General
EB Published Information
VerificationStatus Callout
MacGenerateStatus Callout
Secured RX Pdus
SecOCSameBufferPduCollection
Secured TX Pdus
Published Inform

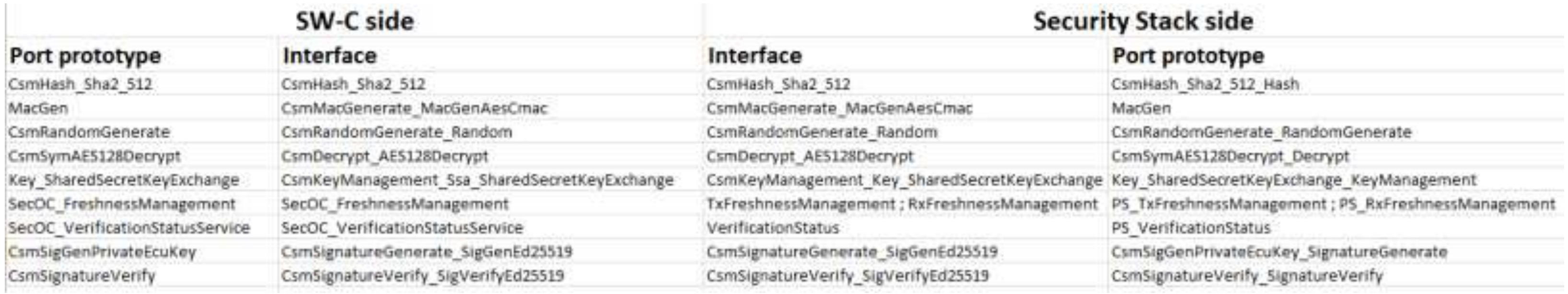
MainFunctionTx Period (0 -> 4294967295)	0.01		
Version Info API	□	Autosar 4.0.3 PduR	□
Crypto Bit Length	☑	Ignore Verification Result	□
Enable Development Error Detection	□	Enable Rte Usage	☑
Query Freshness Value	RTE		
Enable Forced Pass Override	□		
Type with maximal alignment restrictions			
Shaping Rx function name			
Shaping Tx function name			
Default Authentication Value (0 -> 255)			
Data ID Length	UINT16		
SecOCRelocatablePbcfgEnable	☑		
Max buffer size for PDUs (0 -> 4294967295)	20		
Max buffer size for internal usage (0 -> 4294967295)	18		
Propagation the verification status*	☑	Use secured area	□

BswM Interfaces

- Arxml files
  - Internal
  - Interface



- Example of SW-C Rte interfaces



SW-C side		Security Stack side	
Port prototype	**Interface**	**Interface**	**Port prototype**
CsmHash_Sha2_512	CsmHash_Sha2_512	CsmHash_Sha2_512	CsmHash_Sha2_512_Hash
MacGen	CsmMacGenerate_MacGenAesCmac	CsmMacGenerate_MacGenAesCmac	MacGen
CsmRandomGenerate	CsmRandomGenerate_Random	CsmRandomGenerate_Random	CsmRandomGenerate_RandomGenerate
CsmSymAES128Decrypt	CsmDecrypt_AES128Decrypt	CsmDecrypt_AES128Decrypt	CsmSymAES128Decrypt_Decrypt
Key_SharedSecretKeyExchange	CsmKeyManagement_Ssa_SharedSecretKeyExchange	CsmKeyManagement_Key_SharedSecretKeyExchange	Key_SharedSecretKeyExchange_KeyManagement
SecOC_FreshnessManagement	SecOC_FreshnessManagement	TxFreshnessManagement ; RxFreshnessManagement	PS_TxFreshnessManagement ; PS_RxFreshnessManagement
SecOC_VerificationStatusService	SecOC_VerificationStatusService	VerificationStatus	PS_VerificationStatus
CsmSigGenPrivateEcuKey	CsmSignatureGenerate_SigGenEd25519	CsmSignatureGenerate_SigGenEd25519	CsmSigGenPrivateEcuKey_SignatureGenerate
CsmSignatureVerify	CsmSignatureVerify_SigVerifyEd25519	CsmSignatureVerify_SigVerifyEd25519	CsmSignatureVerify_SignatureVerify

- For almost all interfaces Csm is server, SW-C is client
  - Exception: Freshness value

Get in touch!

Elektrobit

sales@elektrobit.com
www.elektrobit.com