Agenda

- Security Overview
  - Vector vHSM Features
  - Vector vHSM Architecture
  - vHSM with HW Platforms
Overview

MICROSAR.HSM: Confidentiality & Integrity

Confidentiality

- How can security asset be classified

Integrity & Authenticity

- How can fabricated message be detected?

MICROSAR.HSM – Functionality

- Large library of crypto algorithms/primitives
  - Cryptographic basic functions (hash, random numbers)
  - Message authentication code (HMAC, CMAC)
  - Symmetric and asymmetric crypto ciphers
  - Supporting signature generation and verification
- Providing protected storage for security assets (e.g. keys)
- Providing secure startup protocol
- Supporting hardware acceleration for better performance
- Providing possibilities to securely update vHSM software

MICROSAR.HSM – Configuration

- Modular architecture with extensive configuration space
- Adaptable HSM firmware to match requirements and footprint
- Comprehensive configuration tool DaVinci Configurator
- Add-Ons to support OEM specific requirements
Classical Approaches for Cryptographic Solutions

Microcontroller

- CPU
- SW Crypto
- RAM
- Flash
- Network Interface

internal connection

vehicle network

Microcontroller

- CPU
- HW Crypto
- RAM
- Flash
- Network Interface

internal connection

vehicle network
Secure hardware extension (SHE)

Microcontroller

Secure Zone

- SHE – Secure Hardware Extension
  - HW Crypto
  - Secure Memory

Control Logic

Peripherals (CAN, UART, ...)

Fundamentals
Fundamentals

Hardware security module (HSM)

Microcontroller

Secure Zone

- HSM
- CPU
- SW Crypto
- HW Crypto
- Secure Memory

CPU

internal connection

- RAM
- Flash
- Network Interface

vehicle network
## Differences of Evita HSM Types

<table>
<thead>
<tr>
<th></th>
<th>HSM (small) ~ SHE</th>
<th>HSM (medium)</th>
<th>HSM (full)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensure integrity of crypto material</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Ensure confidentiality and integrity of crypto material</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Symmetric crypto material can securely stored and updated</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Asymmetric crypto material can be securely stored and updated</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Dedicated CPU</td>
<td>No</td>
<td>Yes</td>
<td>Yes (High Performance)</td>
</tr>
<tr>
<td>HW support for symmetric cryptography</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HW support for asymmetric cryptography</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Firmware changes possible</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Comments</td>
<td>+ Good availability of HW and drivers + Support for AES, MAC and secure boot - No support for asymmetric cryptography</td>
<td>+ Allows firmware changes + SW security libraries can be executed in HSM - No HW support for asymmetric cryptography</td>
<td>+ Long term, extendable, security solution + Crypto accelerators for symmetric and asymmetric cryptography - Cost</td>
</tr>
</tbody>
</table>
Fundamentals

AUTOSAR Host CRYPTO Stack

- Csm
  - Primitive
  - Function(Queue)
  - Key

- CryIf
  - Mapping job to obj

- CryDrv
  - Job Conducting
  - Key Mgmt: Init, Length
Agenda

Security Overview

- **Vector vHSM Features**
  - Vector vHSM Architecture
  - vHSM with HW Platforms
MICROSAR.HSM - Features

MICROSAR.HSM: vHSM – Feature Overview

- Root of trust (Crypto, HSM)
- Crypto Primitives (e.g. Crypto Algorithms)
- Secure firmware updates
- Key management (e.g. Keys, Certificates)
- Secure Startup / Software Authentication
- Secure communication (SecOC, TLS, IPsec)

- Secure vehicle-external interfaces (e.g. TLS, Ipsec)
- Isolation of execution context (e.g. OS, Hypervisor)
- Intrusion Detection and Prevention (e.g. Firewall, Security Event Memory)
- Secure time (Time synchronization)
- Policing (Minimum rights)
- Message forwarding/routing (e.g. VLANs)
Supported Basic Functions and Cryptographic Algorithms

- Basic functions and symmetric crypto algorithms
  - Hash (e.g. SHA-256, SHA-384, SHA-512,...)
  - Random number generation
  - MAC Generation and Verification
  - Cipher : AES in the modes ECB, CBC, GCM

- Key derivation functions and key exchange
  - KDF
  - Key exchange protocol EC-DHE

- Asymmetric crypto algorithms*
  - RSA
  - ECC
  - ECDSA
  - etc

Example: Symmetric encryption and decryption

Example: ECC signature for integrity protected flash programming

* These algorithms require vHSM Add-On Asymmetric Crypto
MICROSAR.HSM - Features

Secure Key and Data Storage

- Data can be passed in plaintext to and stored inside the HSM
  - Includes symmetric/asymmetric keys and certificates
  - Any other (security related) data
    (e.g. mileage, freshness values, certificates)

- Key Installation of symmetric keys according SHE 1.1
  - Support of counter handling (M1, M2, M3)
  - UID can be read out

- Extensive configuration options per key/data:
  - RAM-Key or persisted Key
  - Redundant and reset safe key storage
  - Pre-loading / cashing of keys on startup reduce loading times
  - No limitation regarding the amount of keys*
  - Locking of keys until secure boot has finished
  - Write once keys
Secure Startup Support - Basics

- Support of different secure startup modes:
  - Secure Boot / Sequential mode
  - Authenticated Boot / Parallel mode
  - Trusted Boot / mixed mode

- Configurable secure boot sanctions are:
  - System reset
  - HSM Halt
  - None
  - Custom

Example: Signatures for code signing and secure boot update
MICROSAR.HSM - Features

Secure ECU Startup – Secure Boot

Host/Application Core(s)  |  HSM Core
---|---
Bootsloader  |  vHSM Stack

- Decide which area shall be verified
- Perform Job for Area 1
- Start Appl
- Perform Job for Area 2...N
- Inform vHsm: last job request for secure boot set
- After finishing all pending jobs: Enable secure boot protected keys if all verified slots are valid

- Startup
- Verify Area 0 (Bootsloader)
- Start Bootsloader if valid
- Verify Area 1
- Verify Area 2...N

© 2019. Vector Korea IT Inc. All rights reserved. Any distribution or copying is subject to prior written approval by Vector. V1.01.05 | 2019-08-23
Secure ECU Startup – Authenticated Boot

Host/Application Core(s)

- **Bootloader**
  - Startup bootloader
  - Start Appl

- **Application**
  - Startup application
  - Application running

HSM Core

- **vHSM**
  - Startup
  - Verify Area 0 (e.g., Bootloader)
  - Verify Area 1
  - Verify Area 2..N
  - After finishing all SecureBoot jobs:
    - Release SecureBoot protected secrets for use
MICROSAR.HSM - Features

Secure ECU Startup – Trusted Boot

Host/Application Core(s)

- Bootloader
  - Decide which area shall be verified
  - Trigger Job for Area 1..N
  - Inform vhsm: last job request for secure boot set
  - Start Appl

- Application

HSM Core

- vHSM
  - Startup
  - Verify Area 0 (e.g. Bootloader)
  - Start 1st entity if valid
  - Verify Area 1..N
  - After finishing all pending jobs: Enable secure boot protected keys if all verified areas are valid

Application running

Startup application
Secure Startup Support – Secure Boot Segments

- Secure boot protected memory area \( n \)
- Secure boot protected memory area ... 
- Secure boot protected memory area 2
- Secure boot protected memory area 1

"Segment"

- SecBoot Key
- SecBoot Address: 0xFABE0000
- SecBoot Size: 0x5410000
- SecBoot CMAC
- SecBoot Sanction
- SecBoot Mode: SEQUENTIAL
MICROSAR.HSM - Features

Secure Startup Support – Secure Boot Groups

Host Code Flash Layout

Module LB#0 (Fbl)

- SL#2
  - Segment#1
    - 0xA00000
  - Segment#2
    - 0xBADF0000

Module LB#1 (Appl)

- SL#1
  - Segment#1
    - 0xFABE0000
  - Segment#2
    - 0xFFFF0000

vHSM Secure Boot Configuration

SecureBootGroup_Fbl

- SecureBootSegment#2
- SecureBootSegment#1
  - SecBoot Key
  - SecBoot Address
  - SecBoot Size
  - SecBoot CMAC

SecureBootGroup_Appl

- SecureBootSegment#3
- SecureBootSegment#2
- SecureBootSegment#1
  - SecBoot Key
  - SecBoot Address
  - SecBoot Size
  - SecBoot CMAC

© 2019. Vector Korea IT Inc. All rights reserved. Any distribution or copying is subject to prior written approval by Vector. V1.01.05 | 2019-08-23
vHSM Error Log, DETs and Debugging

- vHSM provides an error log, which can be used to log error events which occur on the HSM
- Errors can be written to secure data flash and read out by application

- Reporting of Det Errors on vHSM
- Source code delivery allows better debugging
MICROSAR.HSM - Secure Update of vHSM

Motivation

- vHSM software stack allows to satisfy existing/upcoming Security requirements for automotive ECUs
  - Requirements may change over time
  - Required changes based on field experiences

- Update/upgrade of vHSM needed (e.g. changes in crypto algorithms, performance improvements, addition of features or other issues)
  - Updating vHSM is done using “Add-On vHSMUpdater”
Introduction of vHSMUpdater
- Bootloader-like component as part of HSM code flash (separate DaVinci Configuration project and binary)
- Taking care of updating vHSM code flash content

General Properties of vHSMUpdater
- Data flash and hence keys stored in vHSM remain available after update
  - Extensions of key configuration / update of key layout not intended
  - Changes in the key layout/configuration lead to invalidation of the data flash and loss of keys
- vHSM update is handled on HSM core
  - No other entity allowed to write HSM code flash beside vHSM updater
  - Update download is executed by host software (e.g. flash bootloader or OTA stack)
- vHSMUpdater is not updatable and is typ. started first in the system
- Resistance against power outages and/or unintended reset during update
- Compliant to existing OEM software download infrastructure and specification
- Security considerations
  - Ensure Confidentiality by using encryption
  - Ensure integrity and authenticity by using signatures
  - Ensure freshness by verifying version
MICROSAR.HSM - Secure Update of vHSM

vHSM Update Sequence Overview with Fbl

1. Generation of updated binary (e.g. vHSM.hex)

2. Download of binary by FBL as part of the payload of an OEM download container. Downloading according to OEM standard download strategy

3. Verification of download by vHSM Appl.

4. Storage of update image in host code flash by Fbl

5. Request vHSM update installation by Fbl

6. Start of vHSMUpdater by vHSM firmware

7. Install update by vHSMUpdater after verifying update image
Agenda

Security Overview

Vector vHSM Features

- **Vector vHSM Architecture & Sample Use Case**

  vHSM with HW Platforms
## Internal Architecture of the vHSM

<table>
<thead>
<tr>
<th>HSM OS</th>
<th>HSM SYS</th>
<th>HSM MEM</th>
<th>HSM LIBS</th>
<th>HSM-CRYPTO</th>
<th>HSM-MCAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Det</td>
<td>Fee*</td>
<td>Crc</td>
<td>vhsmAuth</td>
<td>Fls*</td>
</tr>
<tr>
<td></td>
<td>vhsmCore</td>
<td>MemIf</td>
<td></td>
<td>vhsmCustom</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CryIf</td>
<td></td>
<td></td>
<td>vhsmHal</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Crypto(Sw)</td>
<td></td>
</tr>
</tbody>
</table>

* 3rd party or Vector depending on hardware

**vHSM Updater\(^1\)**

<table>
<thead>
<tr>
<th>HSM UPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>vhsmUpdCore</td>
</tr>
<tr>
<td>vhsmUpdPlatform</td>
</tr>
</tbody>
</table>

---

**vHSM modules**

- AUTOSAR /
- MICROSAR Standard Software

---

1. Add-On vhsmUpd
Work flow of vHSM

AUTOSAR 4.3 BSW
- SWC / Application
  - RTE
  - SYS
  - COM
- OS
- MCAL
- Crypto(vHSM)

Microcontroller
- Flash Bootloader
- RAM
- HSM Channel
  - HSM Channel
  - HSM Channel
  - HSM Channel

vHSM
- vHSM KeyM
- Secure Memory
- Crypto Software Library
- Custom Crypto Job
- Crypto Hardware Accelerator
- Secure Boot and Update Support
- vHSM Job Processor
- vHSM CryIf

Application core subsystem
Shared memory(IPC)
HSM subsystem
Use Case: Secure OnBoard Communication (SecOC)

**MICROSAR.HSM - Architecture and Use Case**

**Advantage:** MAC generation and verification on isolated, secure core with secure memory for key storage and higher performance due to hardware acceleration.

**SECOC sends & receives secured PDUs**
- Option to split payload and security data into two messages
  - SECOC waits until both parts have been received before checking the security
  - Advantage: No need to change CAN communication design for legacy ECUs
- Freshness value handling via callout to FVM
  - Allows OEM specific solution for freshness handling
Use Case: SecOC CMAC Generation Using Hardware Acceleration

1. CMAC generation is requested by application
2. CMAC generation job request at Csm is put in queue
3. Csm handles request and hands over to CryIf
4. CryIf dispatches the requested job to driver objects of Crypto(vHSM)
5. vHSM receives job via shared memory
6. Job processor schedules execution
7. CryIf maps job request to crypto HAL driver
8. Generation of CMAC via HW accelerator and response to PE core
Integration Scenarios with Vector Software

**vHSM + MICROSAR**

- **Advantages**
  - Better tool support → DaVinci Configurator provides consistency between MSR and vHSM configurations
  - Combination already in use
  - Better technical support possible
  - Easier debugging

**vHSM + FBL***

- **Advantages**
  - Secure Startup support
  - Updating SecureBoot based on segmentation table update on host
  - Better tool support
  - Usage of vHSM crypto primitives for secure software downloads/updates from FBL
  - Using vHSM key storage from FBL

*For availability of vHSM integration in FBL and supported features please contact Vector*
Agenda

Security Overview

Vector vHSM Features

Vector vHSM Architecture & Sample

vHSM with HW platform
vHSM with HW

HW Architecture

- Dedicated Core

![Diagram of HW Architecture]

- vHSM
  - HSM Core
    - SW Crypto
  - HW Crypto
  - Secure Memory

- MICRO SAR
  - Host Core
  - RAM
  - Flash
  - Network Interface

*Internal connection*
vHSM with HW

HW Architecture

Secure Memory
Summary

Confidentiality

- How can security asset be classified

Integrity / Authentication

- How can fabricated message be detected?

MICROSAR.HSM – Functionality

- Large library of crypto algorithms/primitives
  - Cryptographic basic functions (hash, random numbers)
  - Message authentication code (HMAC, CMAC)
  - Symmetric and asymmetric crypto ciphers
  - Supporting signature generation and verification
- Providing protected storage for security assets (e.g. keys)
- Providing secure startup protocol
- Supporting hardware acceleration for better performance
- Providing possibilities to securely update vHSM software

MICROSAR.HSM – Configuration

- Modular architecture with extensive configuration space
- Adaptable HSM firmware to match requirements and footprint
- Comprehensive configuration tool DaVinci Configurator
- Add-Ons to support OEM specific requirements
For more information about Vector and our products please visit

www.vector.com

Author:
Embedded Software Team

Vector Korea IT Inc