MICROSAR - SwcSecAccessFord

Technical Reference

FORD - Software Component Vector Security Access
Version 1.0.0

Authors	Markus Schneider
Status	Released

Document Information

History

Author	Date	Version	Remarks
Markus Schneider	2015-04-13	1.0.0	Creation

Reference Documents

No.	Source	Title	Version
[1]	Vector	TechnicalReference_Asr_Dcm_Vector.pdf	3.33.00
[2]	Vector	TechnicalReference_Asr_Csm.pdf	1.02.00
[3]	Vector	TechnicalReference_Asr_Cry.pdf	1.01.00
[4]	FORD	EESE DIAGNOSTIC APPLICATION SECURITY ALGORITHM	001

Scope of the Document

This technical reference describes the general use of the Security Access software component.

Contents

Illustrations

Tables

1 Component History

The component history gives an overview over the important milestones that are supported in the different versions of the component.

Component Version	New Features
1.00.00	Initial creation

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the AUTOSAR SWC SwcSecAccessFord, which implements a security access as specified in [4].

Supported AUTOSAR Release*:	4

* For the detailed functional specification please also refer to the corresponding AUTOSAR SWS.

2.1 Architecture Overview

The following figure shows where the SwcSecAccessFord is located in the AUTOSAR architecture.

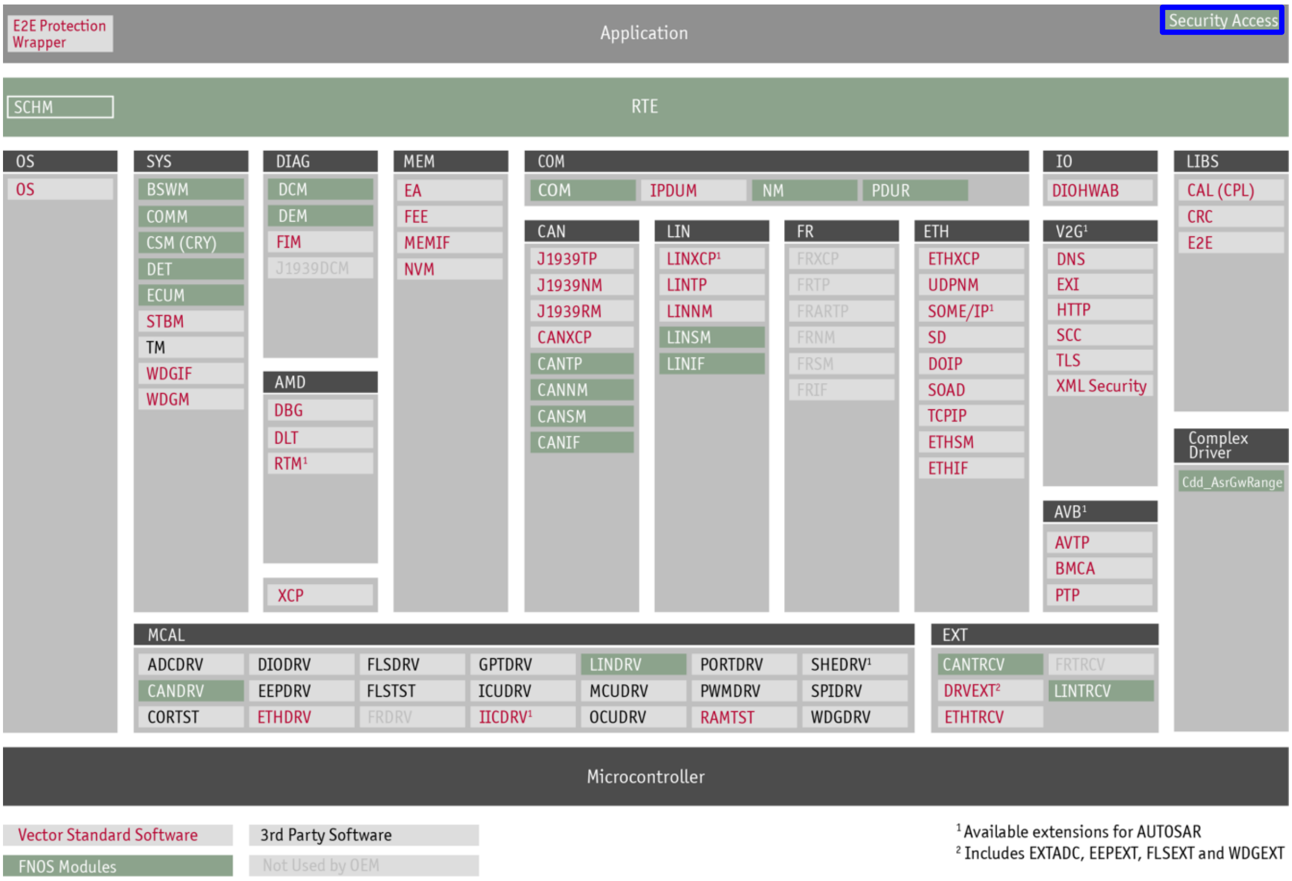

Figure 2-1 Architecture Overview

The next figure shows the port interfaces to adjacent modules of the SwcSecAccessFord. These interfaces are described in chapter 5.

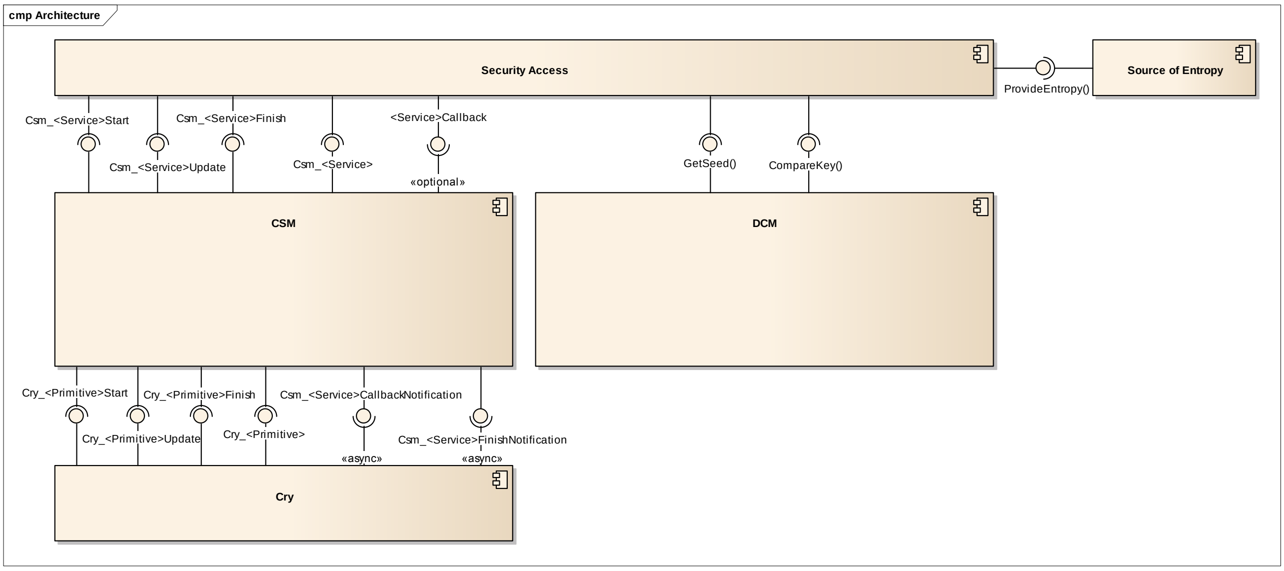

Figure 2-2 Interfaces to adjacent modules of the SwcSecAccessFord

3 Functional Description

3.1 Initialization

The initialization of the component SwcSecAccessFord takes place within the RTE by calling SwcSecAccessFord_Init.

3.2 Provide Entropy

After the initialization and before the usage of the security access a source of entropy has to be provided at least once by calling SwcSecAccessFord_ProvideEntropy.

Note

Depending on the used (pseudo) random number generator, it may be necessary for the entropy to have a certain minimum length.

E.g. in case FIPS186-2 is used, the length of the entropy has to be at least 20 bytes of random data.

The entropy will be triggered in the main function task.

While the entropy is passed to the random number generator a seed request will report a pending operation result.

3.3 States

3.3.1 Sec_EntropyOp

The state machine is used to feed the entropy into the Csm_RandomSeed service. Internal processing in the main function task will be started by invoking the SwcSecAccessFord_ProvideEntropy while the component is in the 'Initial' state.

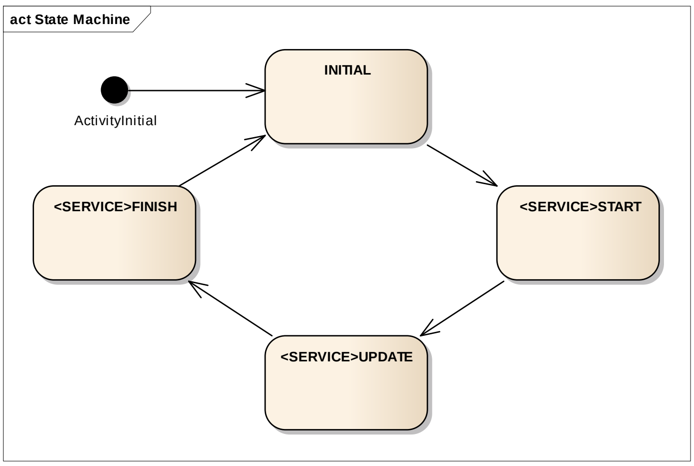

Figure 3-1 State machine Sec_EntropyOp and Sec_VerifyOp

3.3.2 Sec_VerifyOp

The state machine is used to verify the received key from the tester using the Csm_MacVerify service.

3.3.3 Sec_SeedState

The state machine is used to determine the progression of seed generation.

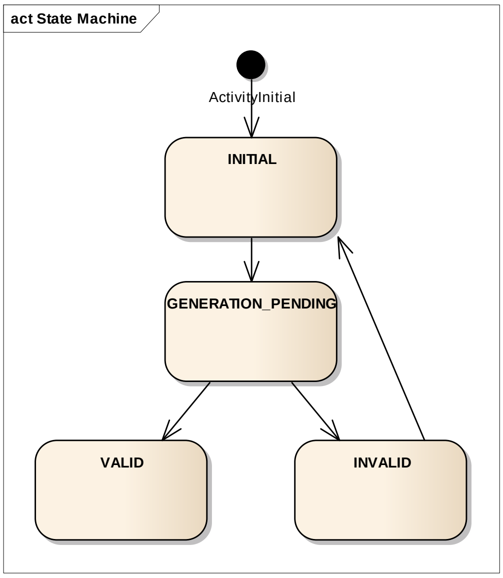

Figure 3-2 State machine Sec_SeedState

3.3.4 Sec_VerifyState

The state machine is used to determine the progression of the MAC verification.

3.4 Main Functions

The SwcSecAccessFord_MainFunction triggers the processing of the provided entropy and the MAC verification. The function has to be called cyclically from RTE level.

4 Integration

This chapter gives necessary information for the integration of the MICROSAR SwcSecAccessFord into an application environment of an ECU.

4.1 Scope of Delivery

The delivery of the SwcSecAccessFord contains the files which are described in the chapters 4.1.1 and 4.1.2:

4.1.1 Static Files

File Name	Description
SwcSecAccessFord.c	Implementation of the SwcSecAccessFord
SwcSecAccessFord.h	Header file of the SWC
SwcSecAccessFord_Cfg.c	In this file the secret keys can be stored
SwcSecAccessFord_Cfg.h	Configuration header file of the SWC

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool DaVinci Configurator5.

File Name	Description
Rte_SwcSecAccessFord.h	Generated data types and function calls

Table 4-2 Generated files

4.1.3 CSM Configuration in DaVinci Configurator 5

Before the integration it is necessary to create the configurations of the random number generator primitives.

Step 1
Create the configurations for the CsmRandomSeed, CsmRandomGenerate and CsmMacVerify Services

On further Information about the configuration of the CSM please refer to the Technical Reference of the CSM [2] and CRY [3].

4.1.4 Integration in DaVinci Developer

The SwcSecAccessFord consists of one AUTOSAR SWC component.

Step 2
Import software components / compositions

To use the software component the software component description must be imported in an existing DaVinci Developer workspace "**File | Import XML file…**"

Step 3
Instantiate software components

Step 4
Create a SWC to provide a source of entropy to the open SWC port.

When FIPS186-2 is used as random number generator this SWC has to provide at least 20 byte of random data.

The quality of randomness highly depends on the quality of the entropy input.

4.1.5 Connecting ports and task mapping

To connect the ports and map the task you have to use the DaVinci Configurator 5

Step 5
Connect the port prototypes from the SWC with the specific component prototypes

Step 6
Mapping of the SwcSecAccessFord_MainFunction to a cyclically task and the SwcSecAccessFord_Init to the initialization

Step 7
Use DaVinci Configurator 5 to generate and add the provided files, as described in 4.1.1 and 4.1.2, to your project.

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Service Ports

5.1.1 SwcSecAccessFord_CallbackNotificationRandomGenerate

Prototype	
`Std_ReturnType` **`SwcSecAccessFord_CallbackNotificationRandomGenerate`** `(Csm_ReturnType retVal)`	
Parameter	
retVal	Result of CSM operation
Return code	
Std_ReturnType	RTE_E_OK – allways succeed
Functional Description	
This function is being called by the CSM after completion of the random number generation. The seed state will be set to SEEDSTATE_VALID if successful	
Call context	
Called by CSM	

Table 5-1 SwcSecAccessFord_CallbackNotificationRandomGenerate

5.1.2 SwcSecAccessFord_CallbackNotificationRandomSeed

Prototype	
`Std_ReturnType` **`SwcSecAccessFord_CallbackNotificationRandomSeed`** `(Csm_ReturnType retVal)`	
Parameter	
retVal	Result of CSM operation
Return code	
Std_ReturnType	RTE_E_OK – allways succeed
Functional Description	
This function is being called by the CSM after completion of a random seed operation. The state of Sec_EntropyOp will be set to next processing step if successful.	
Call context	
Called by CSM	

Table 5-2 SwcSecAccessFord_CallbackNotificationRandomSeed

5.1.3 SwcSecAccessFord_CompareKey_L<LEVEL>

Prototype	
`Std_ReturnType` **`SwcSecAccessFord_CompareKey_L<LEVEL>`** `(const UInt8 *Key, Dcm_OpStatusType OpStatus)`	
Parameter	
Key	Points to the requested key.
OpStatus	Status of the current operation.
Return code	
Std_ReturnType	RTE_E_OK RTE_E_ SecurityAccess_E_COMPARE_KEY_FAILED RTE_E_ SecurityAccess_E_NOT_OK RTE_E_ SecurityAccess_E_PENDING
Functional Description	
The routine calculates a key based on a previously calculated seed and compares the given key (from the parameter) against the calculated key.	
Call context	
Called by DCM	

Table 5-3 SwcSecAccessFord_CompareKey_L<LEVEL>

5.1.4 SwcSecAccessFord_GetSeed_L<LEVEL>

Prototype	
`Std_ReturnType` **`SwcSecAccessFord_GetSeed_L<LEVEL>`** `(Dcm_OpStatusType OpStatus, UInt8 *Seed, Dcm_NegativeResponseCodeType *ErrorCode)`	
Parameter	
`Seed`	Points to the response seed data
`OpStatus`	Status of the current operation.
`ErrorCode`	NRC to be sent in the negative response in case of failure (E_NOT_OK)
Return code	
Std_ReturnType	RTE_E_OK RTE_E_ SecurityAccess_E_NOT_OK RTE_E_ SecurityAccess_E_PENDING
Functional Description	
This function is connected to the GetSeed port of the DCM to provide a seed for the tester.	
Call context	
Called by DCM	

Table 5-4 SwcSecAccessFord_GetSeed_L<LEVEL>

5.1.5 SwcSecAccessFord_Init

Prototype
`void SwcSecAccessFord_Init (void)`
Functional Description
This function initializes the state machines of the SwcSecAccessFord
Call context
Called by RTE on initialization

Table 5-5 SwcSecAccessFord_Init

5.1.6 SwcSecAccessFord_MainFunction

Prototype
`void SwcSecAccessFord_MainFunction (void)`
Functional Description
The main function triggers the internal processing in the background.
Call context
Called by RTE cyclically

Table 5-6 SwcSecAccessFord_MainFunction

5.1.7 SwcSecAccessFord_ProvideEntropy

Prototype	
`Std_ReturnType SwcSecAccessFord_ProvideEntropy (const UInt8 *entropyBuffer, UInt32 entropyLength)`	
Parameter	
entropyBuffer	Points to the source of entropy
entropyLength	Length of the entropy buffer
Return code	
Std_ReturnType	RTE_E_OK RTE_E_SwcSecAccessFord_ProvideEntropy_E_BUSY RTE_E_SwcSecAccessFord_ProvideEntropy_E_NOT_OK
Functional Description	
A provided source of entropy will be used to instantiate the random number generator for the GetSeed() function.	
Particularities and Limitations	
In case of FIPS186-2 the provided source of entropy has to be at least 20 bytes.	
Call context	
Called by a SWC	

Table 5-7 SwcSecAccessFord_ProvideEntropy

5.1.8 Client Server Interface

A client server interface is related to a Provide Port at the server side and a Require Port at client side.

5.1.8.1 Provide Ports on SwcSecAccessFord Side

At the Provide Ports of the SwcSecAccessFord the API functions described in 5.1 are available as Runnable Entities. The Runnable Entities are invoked via Operations. The mapping from a SWC client call to an Operation is performed by the RTE.

Operation	Mapping
CsmCallbackMacVerify	Mapped to <Instance>_Callback of MacVerify primitive (CSM)
CsmCallbackRandomGenerate	Mapped to <Instance>_Callback of RandomGenerate primitive (CSM)
CsmCallbackRandomSeed	Mapped to <Instance>_Callback of RandomSeed primitive (CSM)
SecurityAccessL<LEVEL>	Mapped to SecurityAccess_<SecurityLevelName> of DCM
ProvideEntropy	SWC which provides a source of entropy at least once

Table 5-8 Provided ports by the SWC

5.1.8.2 Require Ports on SwcSecAccessFord Side

At its Require Ports the SwcSecAccessFord calls Operations. These Operations have to be provided by the SWCs by means of Runnable Entities. These Runnable Entities implement the callback functions expected by the SwcSecAccessFord.

Operation	Mapping
CsmRandomSeed	Mapped to <Instance> of RandomSeed primitive (CSM)
CsmRandomGenerate	Mapped to <Instance> of RandomGenerate primitive (CSM)
CsmMacVerify	Mapped to <Instance> of MacVerify primitive (CSM)

Table 5-9 Required ports by the SWC

6 Glossary and Abbreviations

6.1 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
CRY	Cryptographic library module
CSM	Crypto Service Manager
DCM	Diagnostic Communication Manager
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
PPORT	Provide Port
RPORT	Require Port
RTE	Runtime Environment
SWC	Software Component
SWS	Software Specification

Table 6-1 Abbreviations

7 Contact

Visit our website for more information on

> News
> Products
> Demo software
> Support
> Training data
> Addresses

www.vector.com