Safety and Security Features in AUTOSAR

Nagarjuna Rao Kandimala, Michal Sojka
Czech Technical University in Prague
166 27 Praha 6, Czech Republic

Thursday 15th November, 2012

Contents

1 Introduction **2**

2 ISO 26262 **2**

3 AUTOSAR safety features **3**

3.1 Memory partitioning . . . 3
3.2 BSWM defensive behavior . . . 4
3.2.1 Protection against unauthorized use of BSW . . . 4
3.2.2 Protection of data . . . 4
3.3 End-to-End communication protection . . . 4
3.3.1 SW-C E2E communication protection . . . 4
3.4 Program flow monitoring . . . 5
3.4.1 Logical program flow monitoring . . . 5
3.5 Timing related features . . . 6
3.5.1 Provision of synchronized time bases . . . 6
3.5.2 Synchronization of processing of asynchronous processing units . . . 6
3.5.3 Support time deterministic implementation of applications . . . 6
3.5.4 Protection against timing violations . . . 7
3.6 E-Gas monitoring related feature . . . 7
3.7 Communication stack related feature . . . 7

4 AUTOSAR security use cases **8**

4.1 Secure access to ECUs . . . 8
4.2 Immobilizer . . . 8
4.3 Transport interfaces: Generic security/protection . . . 8

5 AUTOSAR security concepts **9**

5.1 Security and cryptographic architecture . . . 9

5.2 Alarm clock concept 9
5.3 DCM manages security access level handling 10
5.4 Format checking of diagnostic services 10
5.5 NV block security of ECU reprogramming 10
5.6 Memory read access 10

Acronyms **12**

Glossary **13**

References **15**

1 Introduction

Growing number of on-road vehicles increases the need for safety and security of passengers, pedestrians and the vehicle itself. The new safety standard ISO 26262 [1] provides the basis for development of safety-related systems comprising of electric, electronic and software components for road vehicles. The standard addresses generic concepts only and their implementation is up to the implementer of the developed component.

AUTomotive Open System ARchitecture (AUTOSAR) 4.0[1] specifies many aspects of automotive systems development. Among others, it defines concepts useful in development of safety related subsystems that comply with ISO 26262. It also defines several security related concepts. As both automotive safety and security is the domain addressed by SESAMO project[2], it is important to know what is supported by AUTOSAR and what is not. AUTOSAR is pretty big standard (it is published as a set of about 170 PDF files) and it is not easy to look it up for concepts and features. For this reason, this document tries to summarize the safety and security related features (or concepts) specified in AUTOSAR.

2 ISO 26262

Before we start dealing with AUTOSAR itself, we mention the introductory requirements of ISO 26262 [1] for functional safety of on-road vehicles.

- ISO 26262 provides an automotive safety lifecycle (management, development, production, operation, service and decommissioning) and supports tailoring the necessary activities during these lifecycle phases.
- ISO 26262 provides an automotive specific risk-based approach for determining Automotive Safety Integrity Levels (ASILs).
- ISO 26262 uses ASILs for specifying the item's necessary safety requirements for achieving an acceptable residual risk.
- ISO 26262 provides requirements for validation and confirmation measures to ensure a sufficient and acceptable level of safety being achieved.
- ISO 26262 provides requirements for the relation with suppliers.

[1] http://www.autosar.org
[2] http://sesamo-project.eu

3 AUTOSAR safety features

AUTOSAR provides a useful document [2] that summarizes requirements related to functional safety introduced in the release 4.0 of AUTOSAR. This section is heavily based on this document.

3.1 Memory partitioning

This feature [3, BRF00115] is the extension of the Operating System (OS) and Run-Time Environment (RTE) functionality that enables the groups of Software Components (SW-Cs) to run in separate memory partitions in order to avoid interference between them. It ensures that memory-related faults in a SW-C do not propagate to other SW-C's. Additionally SW-C executed in user-mode has restricted access to CPU instructions that can change global CPU state [2].

Components running in a separate memory partition have to use inter-OS-Application communication across boundaries of memory partitions, but this is not yet defined in AUTOSAR.

Memory partitioning provides protection by means of restricting access to memory and memory-mapped hardware. In particular, code executing in one partition cannot modify memory of a different partition. Moreover, memory partitioning enables to protect read-only memory segments, as well as to protect memory-mapped hardware.

Partitioning of BSW is not in the scope of the concept/feature – only SW-C is covered. These extensions are also useful for debugging and testing of SW-Cs.

Figure 1 shows an example of partitioning. Partitions are used as the fault containment regions. When an error is detected in a particular partition then that partition can be terminated or restarted during run-time. These partitions are configured in Electronic Control Unit (ECU) configuration file. Let us consider a violation/error occurrence in the partition 1 of Figure 1, Partition 1 is then terminated by the OS services and possible communication is stopped and the partition is restarted [4].

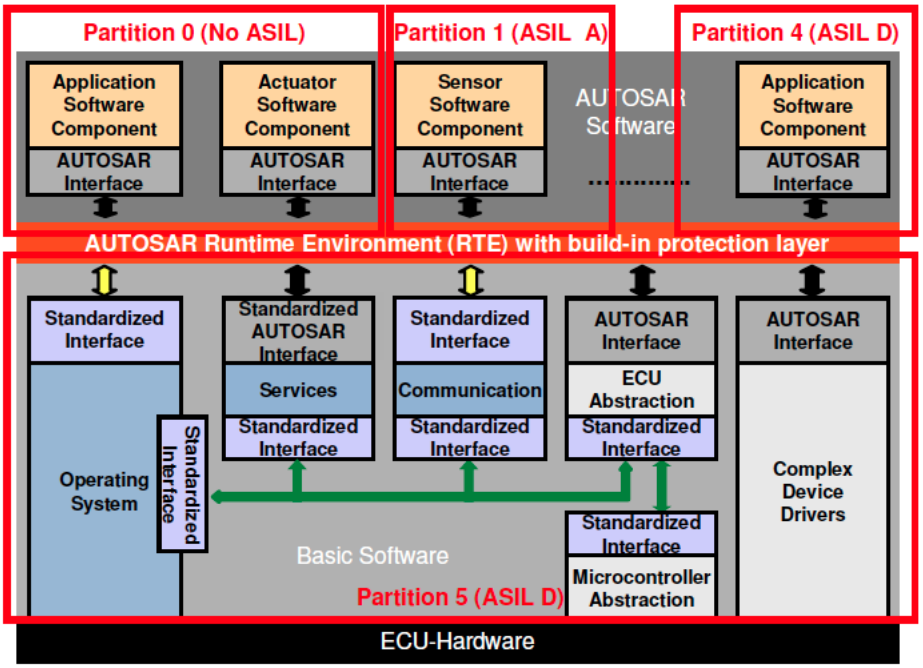

Figure 1: Example of Partitioning (source [4])

3.2 BSWM defensive behavior

Basic Software Module (BSWM) defensive behavior prevents data corruption and wrong service calls in the AUTOSAR Basic Software, this is achieved by the concepts described in the following subsections.

3.2.1 Protection against unauthorized use of BSW

AUTOSAR provides protection for BSWM from unauthorized use by SW-Cs and other BSWMs, to avoid SW-Cs from corrupting or interfering with AUTOSAR service calls of safety-relevant SW-Cs and also to prevent non-safety related BSWMs from corrupting or interfering with BSW used by safety-related BSWMs [3, BRF00128].

Example use case for this concept: Only selected software modules are allowed to request the shutdown of ECU.

3.2.2 Protection of data

AUTOSAR protects safety-related data in RAM and non-volatile memory against corruption (e.g. by using checksums) to handle internal data in a safe manner [3, BRF00129].

3.3 End-to-End communication protection

In an embedded system the exchange of data between sender and receiver(s) can affect the functional safety if its functional safety depends on integrity of such data. Therefore such data are transmitted using mechanisms to protect them against the effects of faults within the communication link [2, Sec. 1.1.5].

The End-to-End (E2E) communication protection related features are implemented in AUTOSAR 4.0 as a standard library providing protection mechanisms that enable sender to protect such data and the receiver(s) to detect and handle errors in the communication link at runtime. This library provides mechanisms for E2E protection, adequate for safety-related communication having requirements up to ASIL D.

SW-C end-to-end communication protection, described in the next section, aims to protect safety-related data exchange among SW-Cs [2].

3.3.1 SW-C E2E communication protection

This feature [3, BRF00114] helps SW-Cs located on remote ECUs to exchange safety-related data. Safe communication is possible by using extensions to RTE and ECU configuration file. This feature also helps to detect and tolerate faults in RTE, communication software and other BSWMs, as well as in communication hardware.

Logically, this feature creates a layer between Virtual Function Bus (VFB) and SW-Cs, which is realized by the following means:

- Safety protocol library – it is a set of stateless library functions that verify the communication (e.g. if a Cyclic Redundancy Check (CRC) of a message is correct or is it on time) and which are invoked by RTE or SW-Cs.
- Introduction of additional configurable attributes (fields) for SW-C ports (e.g. port address), used by safety protocol library.

The port attributes have the state information of the communication whereas the stateless library function does the checks. With these extensions any inter-ECU communication can be possibly used to transmit safety related data. The safety protocol will work on any network/bus that is supported by AUTOSAR.

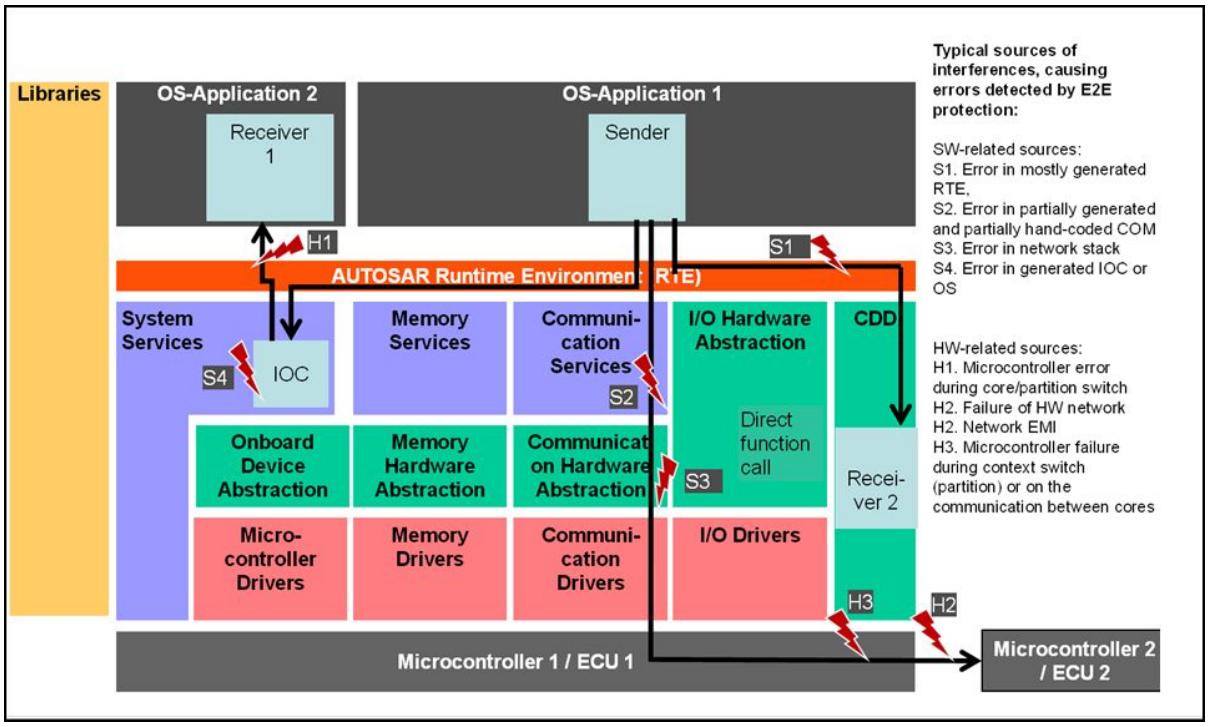

Figure 2: Example of faults detected by E2E protection (source [4])

The protocol needs to be configured depending on reliability and type of network used, size and criticality of transmitted data and fault tolerance of application. The configuration involves selection of used mechanisms and mechanism strength (e.g. CRC8 or CRC16), the selection is done by the integrator.

3.4 Program flow monitoring

Program flow monitoring is a mechanism to check the correct execution of software. The focus of this concept is the detection of program flow errors, i.e. a divergence from the valid program sequence. An incorrect program flow occurs if one or more program instructions are processed either in the incorrect sequence, not in time or are not even processed at all. Program flow errors can for example lead to data corruption, data inconsistencies or the SW failures [2, Sec. 1.1.1].

Logical and temporal program flow monitoring is used in the automotive industry and mentioned e.g. in ISO 26262 as a measure to detect failures of the processing units and as measure for the detection of failures of the HW clock [2, Sec. 4.1.1].

3.4.1 Logical program flow monitoring

Logical monitoring of execution sequence of a program [3, BRF00131] enables the detection of errors that cause a divergence from valid program sequence during the error-free execution of the application.

The valid program sequences are identified and modeled during design phase and the component for Logical Monitoring of Program Sequence uses this model to supervise or monitor the proper execution of program sequences during runtime. If any divergence is detected usually the system is reset.

To reduce the overhead caused by logical monitoring of program sequence, in AUTOSAR it is possible to restrict the definition of Supervised Entities (SE) to safety-related tasks/runnables. At least those have to be monitored but non safety-related tasks can be monitored as well.

The logical program flow monitoring of SW-Cs and Basic Software (BSW) modules is implemented by means of extension of watchdog manager.

3.5 Timing related features

Timing related features [2, Sec. 4.2] can be divided into

- provision of synchronized time bases,
- synchronization of processing of asynchronous processing units,
- support for time deterministic implementation of applications and
- support for protection against timing violations.

Those individual groups are described in the following subsections.

3.5.1 Provision of synchronized time bases

A synchronized time base is a SW time base existing at a processing entity that is synchronized with SW time-bases at different processing entities. A synchronized time-base can be achieved by time protocols or time agreement protocols that derive the synchronized time-base in a defined way from one or more physical time-bases. Examples are the Network Time Protocol (NTP) and FlexRay time agreement protocol [2, Sec. 4.2.1.1].

Synchronization will apply to the clock rate and optionally apply to the clock absolute value. The synchronized time bases are established by the synchronized time-base manager BSW module.

Provision of synchronized time bases within a cluster is restricted to FlexRay and TTCAN clusters in AUTOSAR release 4.0 [3, BRF00120].

AUTOSAR specifies to provide a service to access synchronized time bases available to BSWMs and SW-Cs, to enable SW-Cs to perform time-dependent actions and in particular synchronization & monitoring [3, BRF00127].

Possibility to synchronize AUTOSAR OS with global time from providing bus system in a well-defined and fast way, enabling applications to run their tasks synchronous to the global time from providing bus system [3, BRF00278].

3.5.2 Synchronization of processing of asynchronous processing units

This concept is implemented to synchronize runnables within a set of SW-Cs. They have to be attached to a synchronized RTE timing.

Synchronization is possible within a single ECU as well as across networks [3, BRF00126].

This feature is restricted to RTE timing events only. The events are used to trigger runnables. The synchronization of runnables that are controlled by different AUTOSAR OS instances (e.g. if they are running on different ECUs or different microcontrollers within one ECU) is only possible if they are located on ECUs within the same FlexRay or TTCAN network cluster.

3.5.3 Support time deterministic implementation of applications

Time deterministic implementation of applications requires to be able to specify timing constraints and analyze timing properties at different stages of development, i.e. during virtual integration on VFB level, development of SW-Cs and finally the integration of SW-C into ECUs and of ECUs into a system of ECUs [2].

It shall be possibile to develop implementations based on AUTOSAR with verifiable timing constraints on jitter, latency and execution time. This requirement relates to task scheduling, communication

scheduling and responsiveness to external events [3, BRF00122].

AUTOSAR specifies to enable the use of external events as an initiator for scheduling. As certain external events require a timely response to ensure correct behavior, these events must be able to initiate tasks [3, BRF00123].

3.5.4 Protection against timing violations

Depending on the Scalability Class, the AUTOSAR OS can provide protection mechanisms against timing violation. As the OS is only aware of tasks and not of runnables, the OS provides protection mechanisms on the task level with the fault containment region being the OS application.

Timing protection of SW-Cs at runtime requires monitoring of runnables and preventing the propagation of timing faults from one SW-C to another. If SW-Cs require protection from each other, then their runnables have to be placed into different OS application, which implies that they are placed into different task bodies.

AUTOSAR specifies to provide statically configured runtime timing protection and monitoring. This includes monitoring that tasks are dispatched at the specified time, meet their execution time budgets and do not monopolize OS resources [3, BRF00121].

AUTOSAR recommends to provide a mechanism that monitors ECU local time. This is a necessary basis for deterministic execution of safety functions and for detection of failures of the system by safety integrity functions, within the guaranteed time intervals [3, BRF00125].

3.6 E-Gas monitoring related feature

Electronic Gasoline (E-Gas) monitoring concept [5] is standardized by AKEGAS working group and is not a part of AUTOSAR standard, but still it is a standardized automotive safety concept.

The E-Gas monitoring related SW is located in a Complex Device Driver (CDD). The CDD allows direct access to the related inputs and outputs.

If the responsibility of detection is placed in CDD, AUTOSAR BSW requires to provide a mechanism to transmit the communication protections against a corruption or a loss of data to the CDD [2, BRF00243].

Exclusive/Priority access to Serial Peripheral Interface Bus (SPI) bus should be granted to SW modules that carry out timing critical monitoring protocols between the main controller and a monitoring unit converted via SPI bus. This should be possible for both these SW modules being included in an AUTOSAR SW-C, and these modules being included in a complex device driver [3, BRF00251].

AUTOSAR specifies to allow the use of mechanisms for the testing and monitoring of I/O HW elements as well as the safety related values received/transmitted using the I/O HW elements, to detect errors in measured sensor data or output actuator data, and to detect failures in I/O HW [3, BRF00248].

3.7 Communication stack related feature

This feature aims at enhancing fault detection in order to cover communication failure modes and also providing possible recovery through redundancy [2]. The following concepts are from [3].

AUTOSAR specifies to provide mechanisms for data sequence control, enabling the possibility to check whether a signal is received in sequence [3, BRF00111]. For example, if a distributed safety related powertrain control system receives a torque request signal via CAN with a sequence counter with a value higher than expected, then this error is interpreted as several messages are lost and there might be an inconsistent state within the powertrain system, in this case usually the powertrain system is reinitialized.

AUTOSAR specifies to provide a mechanism that detects wrong routing and supports the correct routing

of signals between SW-Cs, to detect when a signal is coming from an unexpected SW-C sender or when providing the information that it is not supposed to provide. Such messages are reported as errors and discarded. [3, BRF00112]

AUTOSAR specifies to provide a mechanism that detects if periodic signals are not exchanged within defined time interval. Timeouts are used to detect if communication system is functioning or if the individual ECU is communicating. [3, BRF00113]

AUTOSAR specifies to support multiple communication links, to tolerate faults on one of the channels. [3, BRF00241]

AUTOSAR specifies to monitor network communication, to detect high-level issues with network communication and to increase the fault detection capabilities of the system [3, BRF00242]

4 AUTOSAR security use cases

In this section some of the security use cases are documented.

4.1 Secure access to ECUs

AUTOSAR specifies to provide secure access to ECUs (e.g. by user authentication), including standardized up- and download of data and software. For this mechanisms and methods shall be defined [6, Main170].

- The update and upgrade feasibility provided by AUTOSAR includes technical challenges (e.g. standardized up-/download protocol, partly update of the software) and mechanisms (e.g. authorizing the user)
- To fulfill this requirement it is also necessary that the environment that is not standardized by AUTOSAR (like boot loader) needs to match the same security requirements.

4.2 Immobilizer

Immobilizer represents the anti-theft system, protecting the vehicle from unauthorized driving. In case of any attempt for unauthorized engine start, immobilizer requests for visual and acoustical alarms to be generated.

The core functionality of immobilizer is implemented within immobilizer manager. It incorporates the basic functionalities like key learning, learnt key verification, immobilizer diagnostics etc. The data shared between these functionalities should be encrypted to ensure security operation. For further details on immobilizer, refer to [7, p. 77]

4.3 Transport interfaces: Generic security/protection

Protection against unauthorized access in production phase is absolutely required. In production phase Diagnostic Log and Trace (DLT) module shall use the security mechanisms provided by Diagnostic Communication Manager (DCM) to handle the access to the log and trace messages. The mechanism is absolutely required to be implemented to permanently enable the communication module for testing [8, BSW35000029].

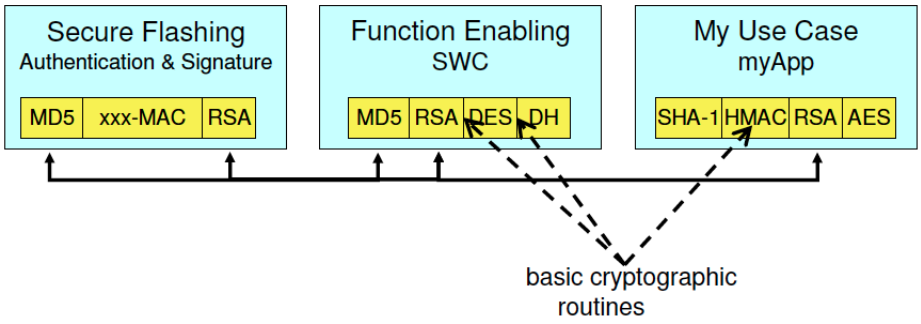

Figure 3: Security Application and Cryptographic routines (source [4])

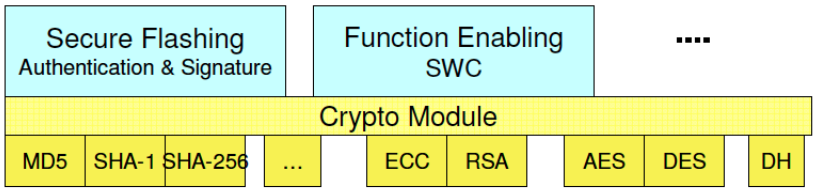

Figure 4: Separation of Security Application and Cryptographic routines (source [4])

5 AUTOSAR security concepts

5.1 Security and cryptographic architecture

Each main security use case corresponds to a security application and each security application uses a different set of cryptographic services. The Crypto Service Manager (CSM) will make it possible for different applications to use the same service but using different underlying cryptographic primitives and cryptographic schemes/routines [9]. For example one application might need to use the hash service to compute an MD5 digest and another might need to compute an SHA1 digest as shown in Figure 3.

Commonality of cryptographic routines may lead to slightly different crypto implementation or to duplicated code, due to which separation of security application and cryptographic routines is required. This is achieved by crypto module as shown in Figure 4.

Crypto module manages requests for cryptographic services from applications and dispatches to a pool of cryptographic basic routines. To achieve this, crypto module exposes an interface for security application to allow for a generic access to standardized cryptographic routines and an interface for cryptographic routines to allow for arbitrary implementations to plug-in crypto module and for use by security applications, as shown in Figure 5.

CSM will make it possible to configure which services are needed and to create several configurations for each service where schemes and primitives can be chosen. Figure 6 shows the embedding of crypto module. CSM is located in system services of service layer. Also shown is the support of Crytographic hardware, which is optional.

5.2 Alarm clock concept

Provides a real-time clock for wakeup purposes. SW-C can set or cancel wakeup alarms. Wakeup alarm allows them to leave a sleep state at a point in time in the future [3, BRF00196].

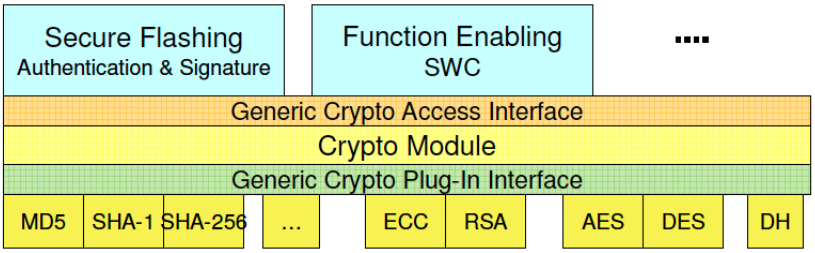

Figure 5: Cryptographic Architecture (source [4])

5.3 DCM manages security access level handling

The Diagnostic Communication Manager (DCM) is a basic software module that provides a common API for diagnostic services. The functionality of the DCM module is used by external diagnostic tools during the development, manufacturing or service. DCM module ensures diagnostic data flow and manages the diagnostic states, especially diagnostic sessions and security states. DCM software specification document [10] describes the functionality, the API, and the configuration of DCM.

The DCM shall manage the handling of the Unified Diagnostic Service (UDS) security access and also the security level handling. The accessibility of the services (service identifier) in the actual security level shall be checked by the DCM [11, BSW04005].

Some diagnostic services are in dependence to a security access level. Therefore it is necessary that the DCM has knowledge about the current level and no service that is restricted by security will be processed without authorization. Not all diagnostic services are allowed in each security level [11, BSW04005].

5.4 Format checking of diagnostic services

The DCM absolutely requires to check the format of diagnostic service. An incorrect service shall be rejected by a negative response, so that the application won't get a request with incorrect format [11, BSW04036].

5.5 NV block security of ECU reprogramming

After ECU reprogramming some Non-Volatile Random Access Memory (NVRAM) data has to be kept secure. This means that some of the Non-Volatile memory blocks (NV blocks) in the NVRAM should never be erased nor be replaced with the default ROM data after first initialization, for example immobilizer code or vehicle identification number [12, BSW08015].

5.6 Memory read access

The OS may offer support to protect the memory sections of an OS-Application against read accesses by all other OS-Applications. If a task can read from any memory then it may operate on incorrect data. This could result in failures at run time. Preventing read accesses provides a way of trapping such faults as soon as they occur. A secondary issue is security. While it is not anticipated that there are any security implications between OS-Applications on the same processor, read accesses does provide protection if required [13, BSW11000]

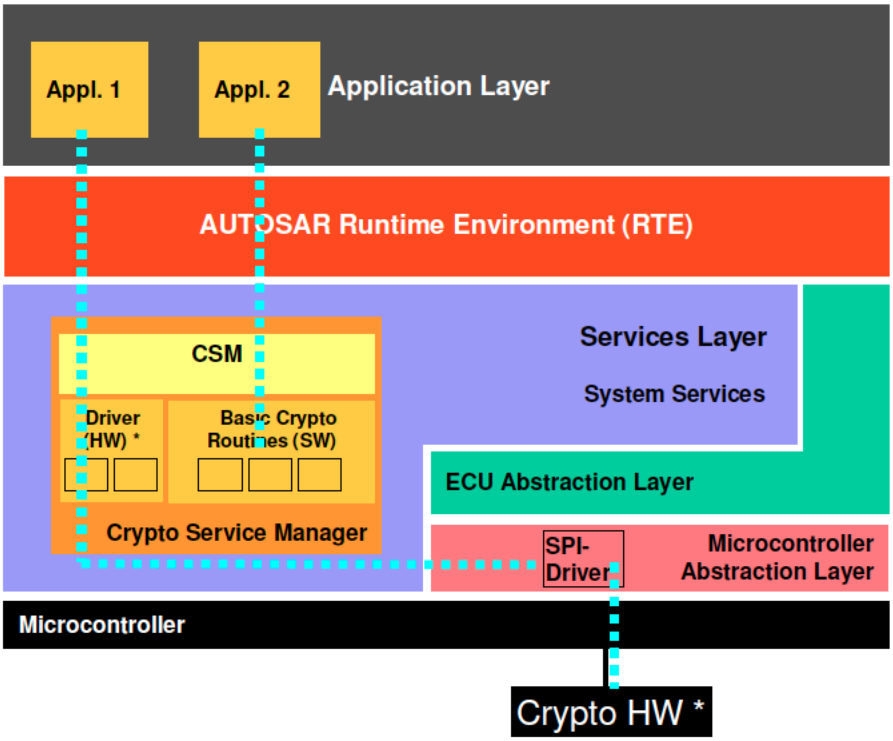

Figure 6: Embedding of crypto module (source [4])

Acronyms

ASIL	Automotive Safety Integrity Level
AUTOSAR	AUTomotive Open System ARchitecture
BSW	Basic Software
BSWM	Basic Software Module
CDD	Complex Device Driver
CRC	Cyclic Redundancy Check
CSM	Crypto Service Manager
DCM	Diagnostic Communication Manager
DLT	Diagnostic Log and Trace
DSD	Diagnostic Service Dispatcher
DSL	Diagnostic Session Layer
DSP	Diagnostic Service Processing
E2E	End-to-End
ECU	Electronic Control Unit
E-Gas	Electronic Gasoline
NV block	Non-Volatile memory block
NVRAM	Non-Volatile Random Access Memory
OS	Operating System
RTE	Run-Time Environment
SPI	Serial Peripheral Interface Bus
SW-C	Software Component
UDS	Unified Diagnostic Service
VFB	Virtual Function Bus

Glossary

Complex Device Driver	An Atomic Software Component that on one side interfaces with the VFB and on the other side directly accesses Peripheral Hardware and/or ECU-Abstraction and/or MCAL. Complex Driver can be accessed via AUTOSAR Interfaces and/or directly by Basic Software Modules[14]
Crypto Service Manager	It offers a standardized access to cryptographic services for applications and system functions.
Diagnostic Communication Manager	This module provides a common API for diagnostic services. The functionality of the DCM module is used by external diagnostic tools during the development, manufacturing or service. This module ensures diagnostic data flow and manages the diagnostic states, especially diagnostic sessions and security states. Further refer[10]
Diagnostic Log and Trace	It is a Basic Software Module, which handles and stores log and trace messages produced by SWC itself or the interactions between SWC and RTE/VFB and by other Basic Sofware Modules. The log and trace messages are generated by calling APIs provided by the DLT module[8]
Diagnostic Service Dispatcher	This is submodule of DCM and it processes a stream of diagnostic data. It receives a new diagnostic request over network and forwards it to a data processor and Transmits a diagnostic response over a network when triggered by the data processor. Further refer[10]
Diagnostic Session Layer	This is a submodule of DCM and ensures data flow concerning diagnostic requests and responses, supervises and guarantees diagnostic protocol timing and manages diagnostic states, especially diagnostic session and security. Further refer[10]
Diagnostic Service Processing	This is a submodule of DCM and it handles the actual diagnostic service requests. Further refer[10]
Non-Volatile memory block	the entire structure, which is needed to administrate and to store a block of NV data.[12]
OS-Application	A block of software including tasks, interrupts, hooks and user services that form a cohesive functional unit
runnable	Runnable entity

runnable entity	A Runnable Entity is a part of an Atomic Software-Component which can be executed and scheduled independently from the other Runnable Entities of this Atomic Software-Component. It is described by a sequence of instructions that can be started by the glsRTE. Each runnable entity is associated with exactly one Entry Point.
Scalability Class	A group of features of the OS (like Memory Protection or Timing Protection) as defined in [15, Sec. 7.11].
Virtual Function Bus	It is an abstraction of the communication between Atomic Software Components and AUTOSAR Services. This abstraction is such that specification of the communication mechanisms is independent from the concrete technology chosen to realize the communication

References

[1] ISO26262, "ISO 26262 standard for functional safety of road vehicles," Dec 2010.

[2] AUTOSAR, "Technical safety concept status report," Oct 2010, R4.0 rev 2. [Online]. Available: http://www.autosar.org/download/R4.0/AUTOSAR_TR_SafetyConceptStatusReport.pdf

[3] AUTOSAR, "Feature specification of the BSW architecture and the RTE," Dec 2011, R4.0 rev 3. [Online]. Available: http://www.autosar.org/download/R4.0/AUTOSAR_RS_BSWAndRTEFeatures.pdf

[4] S. Bunzel, S. Furst, J. Wagenhuber, and F. Stappert, "Safety and security related features in autosar," June 2010. [Online]. Available: http://www.automotive2010.de/programm/content_data/Bunzel-AUTOSAR.pdf

[5] E. W. Group, "Standardized E-Gas monitoring concept for engine management systems of gasoline and diesel engines," Jan 2007, r4.0. [Online]. Available: http://wenku.baidu.com/view/aedb0922bcd126fff7050b51.html

[6] AUTOSAR, "Main requirements," Dec 2011, R4.0 rev 3. [Online]. Available: http://www.autosar.org/download/R4.0/AUTOSAR_RS_Main.pdf

[7] AUTOSAR, "Explanation of application interfaces of the body and comfort domain," Dec 2011, R4.0 rev 3. [Online]. Available: http://www.autosar.org/download/R4.0/AUTOSAR_EXP_AIBodyAndComfort.pdf

[8] AUTOSAR, "Requirements on diagnostic log and trace," Nov 2009, R4.0 rev 1. [Online]. Available: http://www.autosar.org/download/R4.0/AUTOSAR_SRS_DiagnosticLogAndTrace.pdf

[9] AUTOSAR, "Explanation of application interfaces of the body and comfort domain," Nov 2009, R4.0 rev 1. [Online]. Available: http://www.autosar.org/download/R4.0/AUTOSAR_SRS_CryptoServiceManager.pdf

[10] AUTOSAR, "Specification of diagnostic communication manager," Sep 2010, R4, rev 3. [Online]. Available: http://www.autosar.org/download/R4.0/AUTOSAR_SWS_DiagnosticCommunicationManager.pdf

[11] AUTOSAR, "Requirements on diagnostic," Dec 2011, R4.0, rev 3. [Online]. Available: http://www.autosar.org/download/R4.0/AUTOSAR_SRS_Diagnostic.pdf

[12] AUTOSAR, "Requirements on memory services," Dec 2009, R4.0 rev 1. [Online]. Available: http://www.autosar.org/download/R4.0/AUTOSAR_SRS_MemoryServices.pdf

[13] AUTOSAR, "Requirements on operating system," Oct 2011, R4.0 rev 3. [Online]. Available: http://www.autosar.org/download/R4.0/AUTOSAR_SRS_OS.pdf

[14] AUTOSAR, "Glossary," Dec 2011, R4 rev 3. [Online]. Available: http://www.autosar.org/download/R4.0/AUTOSAR_TR_Glossary.pdf

[15] AUTOSAR, "Specification of operating system," Nov 2011, R4.0 rev 3. [Online]. Available: http://www.autosar.org/download/R4.0/AUTOSAR_SWS_OS.pdf