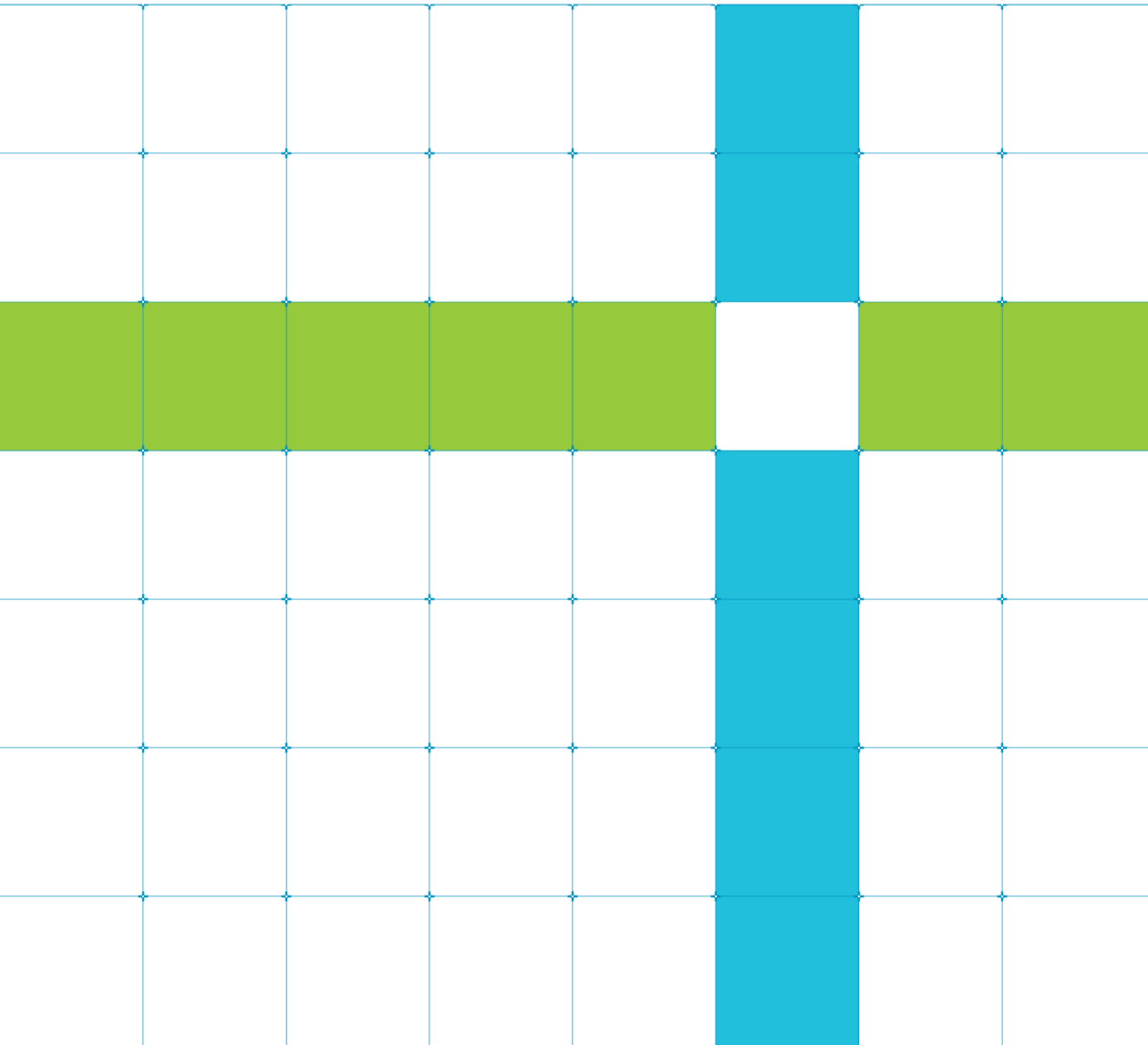
arm

Architecting Secure Automotive Systems

Arm technology for next generation vehicular microcontrollers

Andrew Michael Jones, System Architect. Architecture and Technology Group, Arm

October 2017



Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third-party patents.

THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm's trademark usage guidelines at **http://www.arm.com/company/policies/trademarks**.

Copyright © [2017] Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

Contents

Copyright © 2017 Arm Limited or its affiliates. All rights reserved.

1 Introduction

The automotive industry has recognized the need for rigorous security analysis of its embedded microcontrollers. Major advances in the functionality of these Electronic Control Units (ECUs), as well as a rapid and widespread increase in their implementation, have boosted incentives to reassess security specifications. This whitepaper describes how existing Arm technology can implement Hardware Security Modules (HSMs) that are based on the E-safety Vehicle Intrusion proTected Application (EVITA) framework, in a simple and low-cost manner within the automotive industry.

The development of standards for automotive software, such as AUTOSAR, has allowed a large increase in the functional complexity of deployed Electronic Control Units (ECUs). This development has been accompanied by the need for regular updates in the field of security, and by separate vehicular systems, to manage communication. Modern cars can be thought of as heterogenous automotive networks.

Several future e-safety applications are based on wireless vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) communications. V2I and V2V, collectively called V2X, have the potential to increase the efficiency and operational performance of all vehicles, in an intelligent transportation system. Together with the growing deployment of e-Toll, tachographic, odometric, and location-based functions, this means that the number of assets in automotive systems that require security can be expected to increase rapidly. From the perspective of potential security threats, this increased complexity implies that, not only will the attack surface expand for each vehicle, the number of vehicles a potential attacker can target with the same exploitation technique will grow.

The EVITA project was a European Commission-funded project, the primary aim of which was to specify a secure architecture for automotive networks, in order to provide a basis for the secure deployment of electronic safety aids, based on V2I and V2V communications. EVITA analyzed use cases, assets, threats, and attack methods, in order to derive a set of security requirements for automotive ECUs. The project designed an architectural framework to meet those requirements. Building on the Trusted Platform Module (TPM) specification for secure cryptographic processors, EVITA specified the design and use of Hardware Security Modules (HSMs), to act as trusted subsystems within ECU systems-on-chip.

The EVITA specification describes how an HSM can provide security benefits to a network of ECUs, in a vehicle. The principle use cases are proving the identity of an ECU, providing secure communications between ECUs, reporting the identity of software executing on the ECU, and the remote deployment of maintenance updates.

An HSM is implemented as an isolated subsystem, which typically includes a boot ROM, secure Non-Volatile Memory (NVM) for holding keys and other assets, some hardware accelerators to meet real-time cryptographic functions, a high-entropy source of random numbers, and a dedicated CPU with which it coordinates security functions and manages the interface with the application processor.

This allows:

- Confirming that an authentic OS starts in a trusted environment and can subsequently be trusted.
- Attesting the authenticity of a platform, and its OS, to third parties (attestation).
- Enabling security capabilities for a trusted OS and its applications.

Copyright © 2017 Arm Limited or its affiliates. All rights reserved.

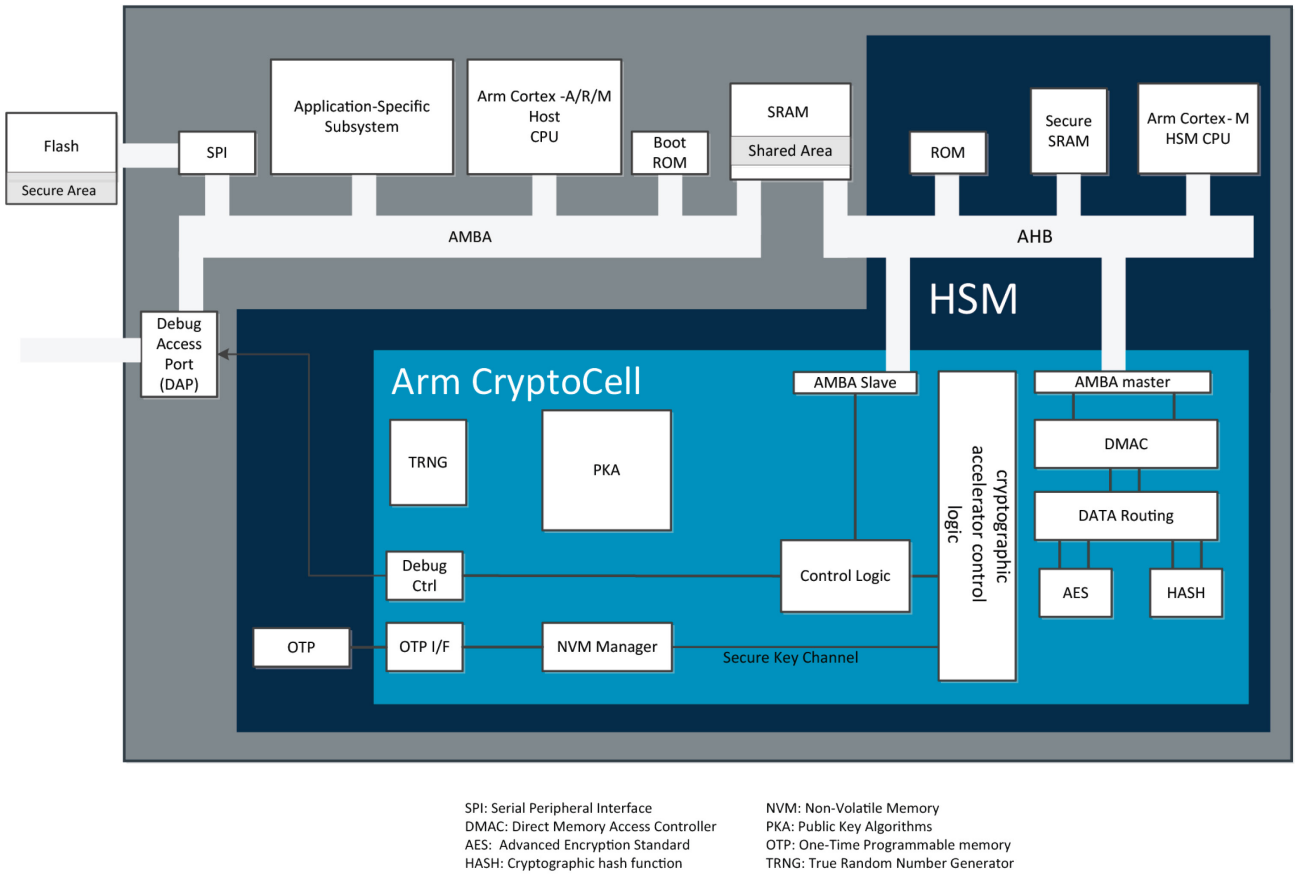


Figure 1 - A Generic Automotive ECU with HSM

A key aspect of an HSM design is certification. Arm recommends that third party, independent testing labs are used to verify that the HSM satisfies its security claims. Automotive OEMs might gain timescale and cost advantages by specifying a common HSM architecture for all their silicon suppliers. Commonality would allow re-use of the security audit of an HSM, across and within vehicle families.

Copyright © 2017 Arm Limited or its affiliates. All rights reserved.

2 Platform Trust

In order to trust a platform, the identity of the current execution environment and the identity of any software that could have influenced the security of the current execution environment must be established.

The standard way of confirming code integrity involves computing a cryptographic hash function over the code, a process known as *measurement*. It is best to measure a software module before it begins execution, because it will be in a known state and will not have started to generate any divergent local state. This is commonly done by using the code which was previously loaded on the platform, and is therefore trusted, to measure the successor code before it relinquishes control. As a result of this, a chain of trust can be constructed in which each loaded software module measures the next one, before transferring control. This sequence of measurements creates a log of the chain of trust. However, the question naturally arises of how the first code to be executed on the platform, becomes trusted.

EVITA solves this problem by using a security anchor that has to be trusted implicitly. In practice, the security anchor is a small immutable piece of verifier code, ideally isolated within ROM, that is the first code that is executed during any boot process, and that initializes the root of trust (RoT) of the ECU. Here, the RoT is a computing engine with code and data that are co-located on the same ECU platform as the application, and that provide security services for the ECU application.

In a *trusted boot* scenario, the HSM constructs the chain of trust by measuring the bootloader and subsequent software finishing with the operating system.

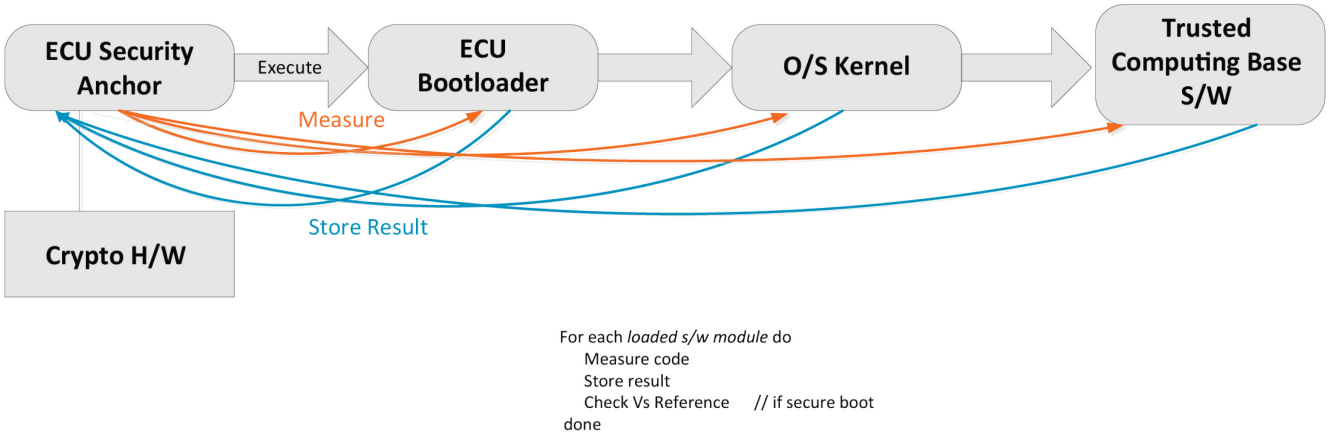


Figure 2 - EVITA boot chain for integrity protection

The process of building the measurement chain is typically extended to include application code that runs under an OS, in a less privileged execution mode. For an automotive ECU, the chain can be used to decide if the platform is in a trusted state, because many operating systems are not sufficiently secure to provide this information directly.

Copyright © 2017 Arm Limited or its affiliates. All rights reserved.

EVITA proposes HSMs that are based on cryptographic hash chains. This makes efficient use of secure memory because it requires a set of fixed size ECU configuration registers (ECRs), to store an arbitrarily long concatenation of code measurements (the log). The HSM places these measurements in secure RAM, with each measurement associated with a reference value that is provisioned before deployment, and held in NVM.

The HSM initializes each secure ECR to a known value at boot. Each entity in the chain of trust calls a secure HSM function, that computes a cryptographic hash over the code and data that is to be loaded next, together with the current value of the ECR. Given the cryptographic strength of the hash function and the hardware security of the storage of the ECR, the integrity of the measurement log is secure. Short of rebooting the machine and losing control of the platform, this effectively prevents malware from hiding its presence from the log.

Even though code measurement at boot time is a powerful technique, it is sometimes not enough to assure trust in the current state of the platform. Simply because a platform starts in a trusted state, does not guarantee that it is not subsequently compromised by external inputs. There are two common defenses against this. One is to augment the software with monitors that impose certain dynamic properties, such as control flow and stack integrity. These monitors then become part of the measured code base. The other defense is for the platform to run service daemons that oversee and impose behavioral policies on less privileged code. EVITA proposes a number of these services, which can dynamically monitor communication, memory access patterns, and API usage. This is specified as part of the EVITA software security modules.

2.1. Secure Boot

Each ECU in the EVITA architecture implements secure boot, supported by its HSM. Starting with a secure boot ROM, the code measurement of each component is compared to a reference value that is provisioned by the root of trust owner, or by an entity which the root of trust owner has authorized to this. On detecting a difference, the secure boot procedure terminates the boot so that final applications, simply because they are actually executing, can be assured that the platform started in an approved state.

2.2. Attestation

Attestation is the process by which a remote agent determines the hardware and software configuration of the platform. The remote agent can use this information, in order to make authorization decisions for the platform.

For example, a remote agent can be the cloud-based software update server of an OEM. Even following a secure boot, the remote agent communicating with an ECU can only infer that an automotive ECU has booted into some authorized state, which might be an unknown state. In addition, the remote agent must also be sure that a specific measurement chain authentically represents the software state of the ECU before it can make a trust decision, regarding the ECU. Because communication between a remote agent and an ECU will typically be mediated by untrusted software running on the application processor, this untrusted software must be authenticated back to the root of trust of the ECU. Following boot, the HSM derives an attestation keypair from its device-unique key, the public half of which is already known to the remote agent. Attestation proceeds by the remote agent requesting a quote and sending a nonce to the attestor, to protect against the replay of stale attestations. The HSM generates the quote by retrieving the current measurement log held in secure storage, concatenating the nonce, and digitally signing it with its private attestation key. The log contains the measurement chain values and associated meta data, to allow the verifier to parse the chain.

The HSM is able to generate random session keys for the communication with the remote agent, as a counter-measure against a reboot attack, that might occur in the middle of attestation and exploit a difference between the system, at the time of check and the time of use.

The precise details of the quote and measurement chain depend on the attestation protocol that is used. However, it is intended that the hardware features supported by EVITA HSMs, together with trusted firmware running on the CPU in an HSM (for example, an Arm® Cortex®-M processor), will give sufficient flexibility for OEMs to adopt or develop appropriate attestation infrastructures.

Copyright © 2017 Arm Limited or its affiliates. All rights reserved.

2.3. Device Security Lifecycle Management

A key aspect of automotive ECUs is the process of supporting the security of the device, from manufacture through to disposal. At each stage in the real-world existence of a device, its security profile is different, and this can be reflected in the set of allowable features.

The HSM supports the transition of the device, through four possible lifecycle states (LCS), illustrated below.

LCS	Description	Event triggering transition into this LCS
Chip Manufacturing	Affiliation: IC vendor. The default state of the chip coming out of fabrication and while on the IC vendor premises.	Initial state post manufacturing of the IC
Device Manufacturing	Affiliation: OEM. State of the chip during device assembly, on the production line of the OEM.	Population of chip manufacturer flag in the on-chip NVM
Deployed	Affiliation: User (Car Owner) The state of the chip (device) when it is deployed in the market (all security features switched ON).	Population of device manufacture flags in the on-chip NVM
RMA	Affiliation: lab (for example, lab of the OEM). The state of the chip (device) when it is returned for failure analysis. Functional aspects are operational but RMA must not compromise car owner data/privacy and must not extend the attack surface.	Population of an RMA indication in the on-chip persistent memory by an authorized entity (post authentication).

Table 1 - ECU lifecycle states

The RMA state is intended for post-field deployment. Debug and testing features can be restored in a manner which does not reveal user or service sensitive data. Entering RMA state requires certified authentication that is cryptographically checked by the HSM. Also, at the transition into the RMA state, all key storage is programmed into a recognized erased state, for example, a zeroed state. However, the LCS state continues to be protected. This can act as a countermeasure against the grey market deployment of the device, as a security-disabled but otherwise functional ECU.

Because the lifecycle state determines the security level of the device, it must remain protected within the secure NVM of the chip. In this way, access is restricted to the HSM and the authentication of its transitions are enforced.

Copyright © 2017 Arm Limited or its affiliates. All rights reserved.

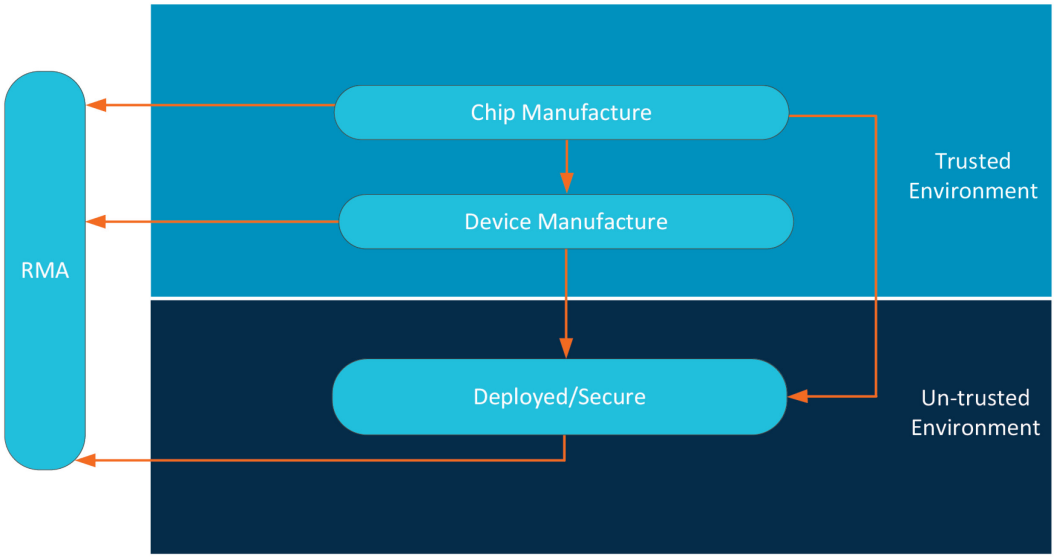


Figure 3 - Device Security Lifecycle

Arm CryptoCell is a secure subsystem, comprising functions for cryptographic acceleration and management of secure resources. Secure lifecycle management is fully integrated within an Arm CryptoCell-based implementation of an HSM, and enforces policies, including:

- Execution of unsigned code.
- On-chip NVM programming policy.
- On-chip NVM read policy.
- Confidential (that is, encrypted) code accessibility.
- Secure debug.

Copyright © 2017 Arm Limited or its affiliates. All rights reserved.

2.4. Secure Debug

Secure debug relates to the granting of debugging rights, based on a signature from a trusted entity. Arm CryptoCell uses a certificate-based scheme, in which a debugger first has to present a certificate for inspection by the HSM CPU, before initiating a debug session. The CPU uses a root of trust (or predecessor in a chain of trust) to validate the certificate and enables debug permissions, as indicated by the certificate. Authenticated debug can be enforced by the lifecycle state management - the default policy (which can be changed by the IC vendor) is shown below.

LCS	Debug Status
Chip Manufacturing	All debugging enabled
Device Manufacturing	All debugging enabled
Deployed	Authenticated debug (certificate based) active
RMA	All debugging enabled

Table 2 - Secure Debug

CryptoCell implements a unified root of trust for the verification of certificates that originate from the same entity (that is, those where both the code-loading certificate and the debug authentication certificate are signed by the same key). It is important to ensure that a debugger does not exceed its debugging permissions. CryptoCell makes use of a bus filtering unit that partitions the address space into areas of accessibility and inaccessibility, for each valid certificate. This allows isolation between agents with similar access rights, for example, software belonging to separate application vendors.

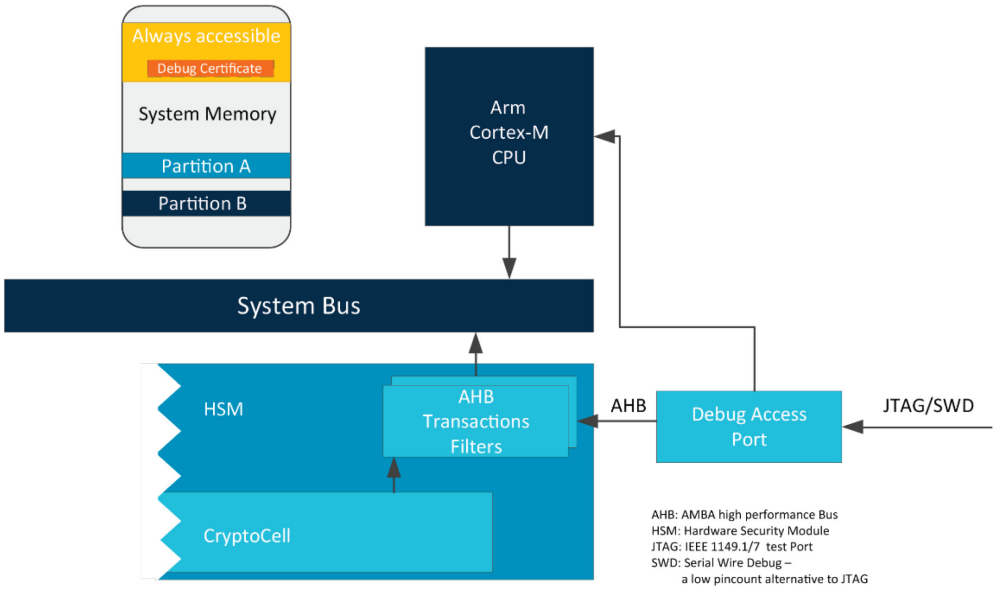


Figure 4 - Example Authenticated debug implementation

Copyright © 2017 Arm Limited or its affiliates. All rights reserved.

2.5. CryptoCell NVM management

NVM management is fundamental to the operation of HSMs because significant amounts of persistent data serve as trust anchors for the platform. The integrity, and in some cases the confidentiality, of these roots of trust must be assured. This data must be stored on-die in NVM, as indicated in the EVITA framework.

1. There are three main ways in which the root of trust fields can be populated:
2. The IC Vendor does not populate the chip with any sensitive or confidential data, and ships it in the Chip Manufacturer LCS. In this case, code encryption and asset provisioning by the IC vendor can be done using embedded RTL secrets. This is low cost but less secure than other methods.
3. The IC vendor populates the HW Unique Key (HUK), relevant flags and, optionally, any key fingerprints. This transitions the chip to Device Manufacturer LCS, before shipping it to the OEM.

The IC vendor is authorized to populate all the fields (some on behalf of the OEM), and ships the IC in the Deployed LCS.

Many platforms will use some combination of mask ROM, OTP, eFlash, and external Flash, to provide sufficient (RoT and Non-RoT) NVM resource for the ECU. CryptoCell provides support for directly or indirectly interfacing to these implementations.

ECUs that are designed to hold much of their persistent state off-chip, can make use of secure external flash partitions, called Replay-Protected Memory Blocks (RPMB), that are present in standard eMMC and UFS devices. CryptoCell is able to derive and protect a device-unique key, suitable for use in securing RPMBs. In the Device Manufacture lifecycle state (in a secure environment), CryptoCell supports the export and insertion of a shared secret into the Flash device. As the CryptoCell is the only entity capable of accessing the RPMB key, a CryptoCell-based HSM can guard access to this resource and manage the storing of non-RoT secrets, for example, derived keys and storage of confidential code and data.

2.6. Cryptographic Hardware

Cryptographic specific hardware is widely used to accelerate processor-intensive encryption and decryption computations. There are several reasons why they are deployed in automotive HSMs.

- Real time performance is important. Secure boot might rely on cryptographic operations being performed on large amounts of code and data, before an application can launch. Rapid establishment and maintenance of secure communication channels is critical to V2X applications.

- Significantly less power is used when performing cryptographic operations in hardware, even relative to small CPUs. On an automotive ECU, this might ease thermal management for intensive use cases.

- Cryptographic computation is vulnerable to side channel attacks (SCAs). The three main classes of SCAs, power and electromagnetic signature analysis, timing analysis, and fault injection, all have good countermeasures that are generally known, and that are implemented by CryptoCell. The alternative, hardening a CPU and software against SCAs, can be challenging if there is a tight cost and performance budget.

- There is a requirement for high entropy, random number generation. These numbers are heavily used in cryptographic computation for deriving keys, creating nonces, and producing initial values for encryption and decryption schemes.

Copyright © 2017 Arm Limited or its affiliates. All rights reserved.

The EVITA HSM architecture defines a set of cryptographic functions, all of which are implemented by Arm CryptoCell, including a rich set of symmetric and asymmetric computation capabilities, together with a set of Hash and MAC functions, and a digital true random number generator, that complies with the AIS-31 and NIST SP 800-90B standards1.

1 BSI: Application Notes and Interpretation of the Scheme (AIS) 31 – Functionality Classes and Evaluation Methodology for Physical Random Number Generators
NIST: SP 800-90 Recommendation for Random Number Generation Using Deterministic Random Bit Generators

Copyright © 2017 Arm Limited or its affiliates. All rights reserved.

3 The Design of Secure ECU Architectures

Secure automotive ECU architectures are heavily influenced by cost concerns. In order to minimize the silicon area required for security hardware, EVITA specifies three types of HSM, according to the type of communication in which the host ECU is expected to engage. These types are:

- V2X messages. These messages require high-speed, asymmetric encryption and key storage because the time to establish and authenticate secure connections, is critical. HSMs that support this profile are what EVITA refers to as full HSMs.
- Communication between ECUs within the same vehicle. This communication requires lower-speed, asymmetric cryptography and high-speed, symmetric encryption, together with secure key storage. HSMs that support this profile are what EVITA refers to as medium HSMs.
- Simple and secure communication between small ECUs and clusters of sensors and actuators. This communication requires static symmetric encryption. HSMs that support this profile are what EVITA refers to as light HSMs.

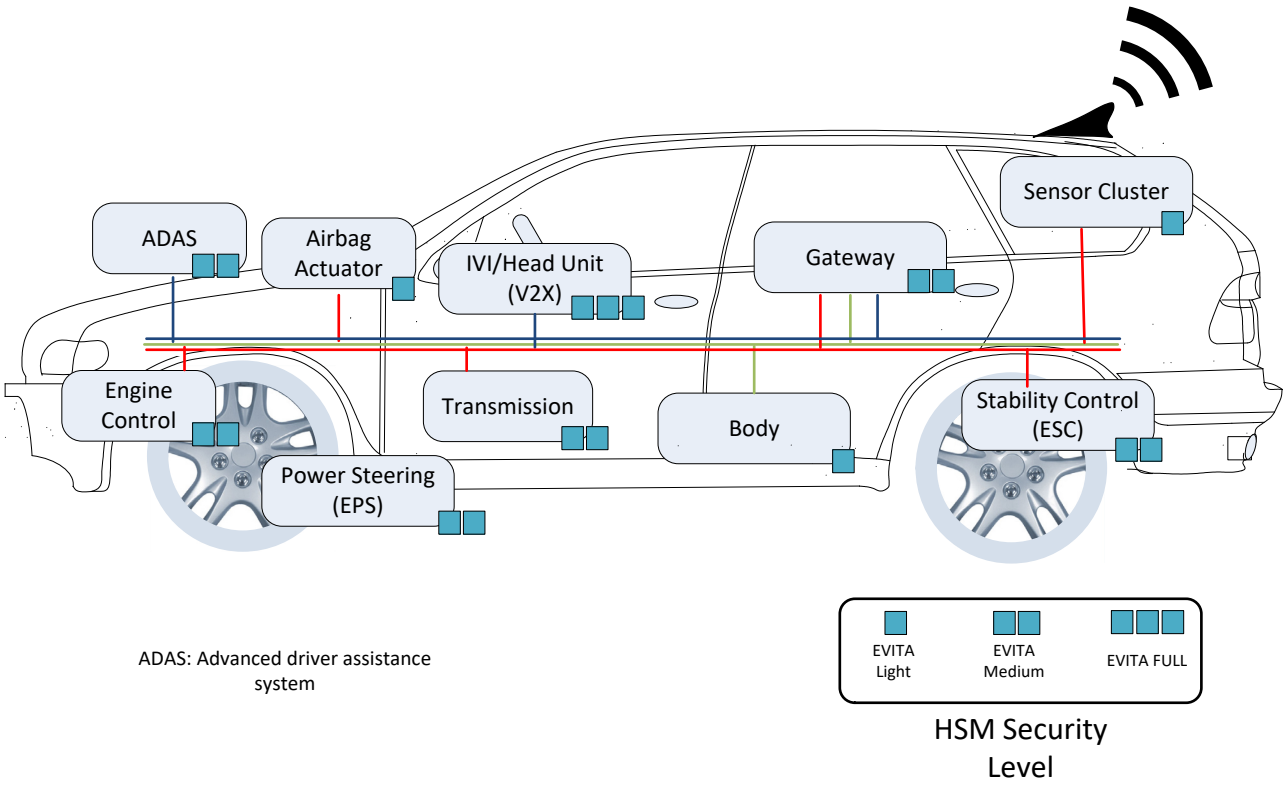


Figure 5 - Example of EVITA hardware security modules deployment

Copyright © 2017 Arm Limited or its affiliates. All rights reserved.

Figure 5 is an example of an automotive network topology that shows a distribution of HSM types, among ECUs. It is increasingly common to see architectures making use of a gateway ECU, whose role is to ensure that only permitted messaging routes are employed, and to convert between protocols.

The EVITA full HSM profile specifies hardware ECC-256 and Whirlpool2 Hash functions. The Medium HSM profile does not require these accelerators, and its isolated CPU is dimensioned accordingly. The Light HSM profile is focused on meeting the smallest cost and therefore, its sole major accelerator is an AES-128 accelerator, to support end-to-end protection of data exchange with sensors and actuators, with critical data.

An EVITA light HSM is based on enabling sensors and actuators to enforce the authenticity, integrity, and confidentiality of the data that is exchanged. The necessary shared secret for symmetric cryptography can be established in several ways, for example, by pre-configuration by the OEM or by performing a software key establishment routine on the ECU application processor. It must be noted that a standard light HSM has no internal NVM, no internal RAM, and that all secrets are accessible by application software, running on the host CPU.

3.1. EVITA Light HSM Architectures with an Arm Cortex-M processor

This absence of secure RAM, NVM, and isolated processing capability, in the light HSM profile, limits the scope of ECUs in which these low cost HSMs can be used. If the application CPU is an Armv8-M processor, then Arm TrustZone® technology can increase the security level of the EVITA Light profile, with no significant increase in cost. TrustZone technology can isolate partitions within existing NVM and RAM resources, so that they can only be accessed by software running in the secure mode of the CPU. This allows hardware supported separation of secure HSM functions and applications software. This HSM profile is, in Arm terminology, called trusted light. A trusted light HSM could generate, process, and store its secrets (that is, AES keys and secure boot references) more securely in hardware, and is therefore able to enforce the cryptographic boundary and secure boot, more strongly.

The TrustZone technology is a hardware security technology that is incorporated into Arm processors. It consists of security extensions to an Arm system-on-chip (SoC) covering the processor, memory, and peripherals. These mechanisms can be leveraged by system designers, to run secure services in isolation from the OS. With TrustZone technology, the processor can execute instructions in one of two possible security modes. These modes are referred to as the normal world- where untrusted code executes- and the secure world- where secure services run. These processor modes have independent memory address spaces and different privileges.

While code running in the normal world cannot access the secure world address space, code running in the secure world can access the normal world address space, in certain conditions. A special processor bit- the NS bit- indicates which world the processor is currently executing in, and this bit accompanies transactions over the memory bus and certain I/O buses for peripherals. This enables the system designer to allocate memory, solely to the secure world, and to control which devices are accessible from the different worlds. Hardware interrupts can trap directly to the secure world interrupt handler, which then enables flexible routing of those interrupts to either secure world or normal world. As the processor executes in one security mode at a time, it must switch worlds in order to execute software in the other security mode. In a Cortex-M processor that is implementing TrustZone technology, a mode switch is very lightweight and can be achieved by issuing a special branch instruction (BLSF), whose target must be a Secure Gateway (SG) instruction that is located in a memory region marked as being suitable for secure mode switches, Non-Secure Callable (NSC).

These features allow ECUs using an Armv8-M processor, to isolate HSM processing from application code, and also to create secure partitions within existing memory resources. A design for an ECU, based on a trusted light HSM, is outlined in Figure 6.

It is typically less expensive to have a single, large memory device and partition it into secure and non-secure regions, than it is to provide separate, dedicated memories for each secure world and normal world. In some designs, this will permit low-cost, secure partitions to be established in

[2] Or another cryptographically strong hash function.

Copyright © 2017 Arm Limited or its affiliates. All rights reserved.

already existing resources. The TrustZone Memory Adapter (TZMA) enables a design to secure a region within an on-SoC static memory, such as an ROM or an SRAM. The TZMA allows a single static memory, of up to 2MB, to be partitioned into two regions, where the lower part is secure and the upper part is non-secure.

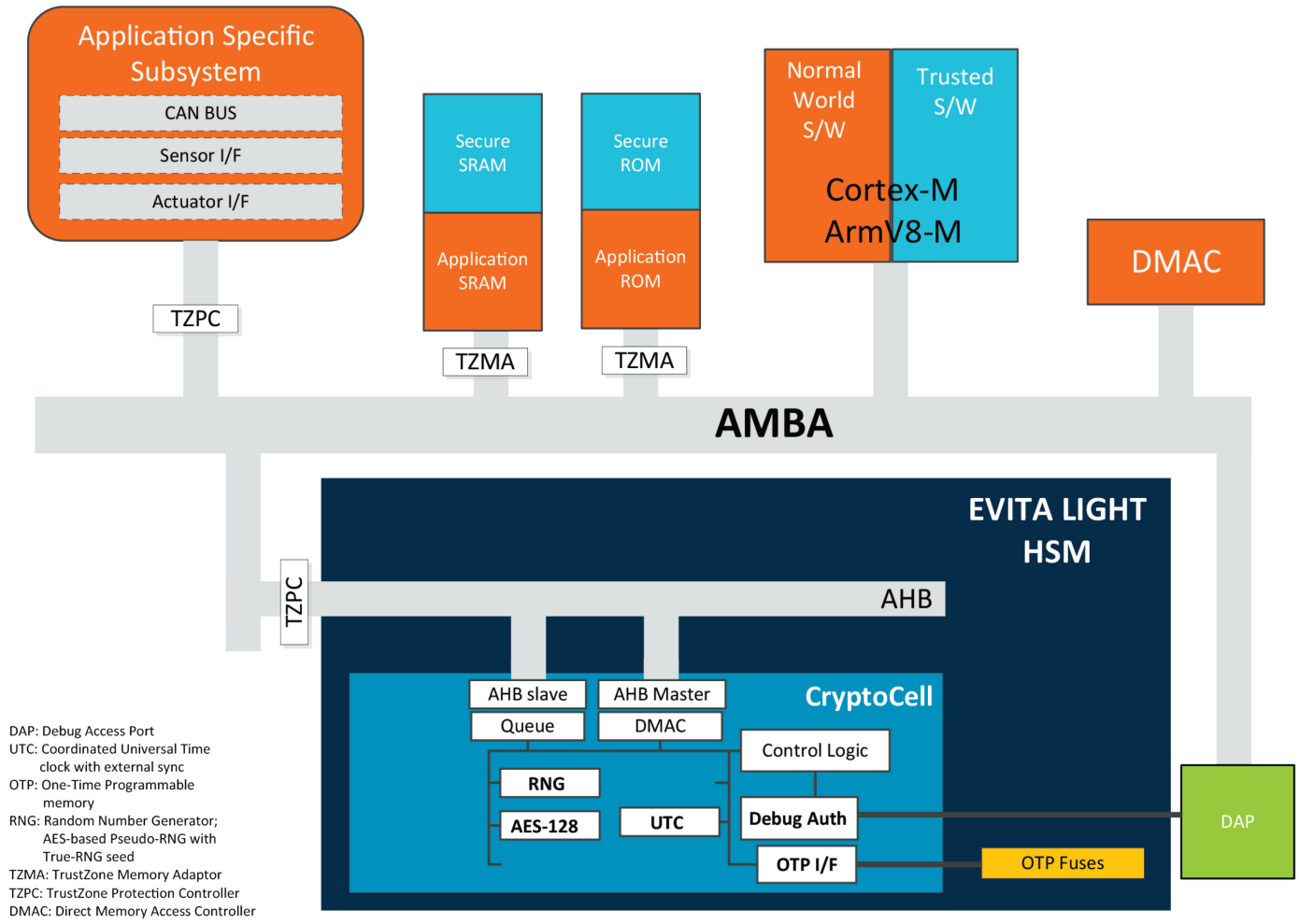


Figure 6 - EVITA Trusted Light HSM architecture

In fact, this technique can also be used to design low cost ECUs that are in accordance with the medium HSM, or full HSM profile. If the application performance profile can be met with an Armv8-M processor, then TrustZone technology can provide hardware-isolated compute and storage resources for the HSM.

Copyright © 2017 Arm Limited or its affiliates. All rights reserved.

3.2. Virtualized EVITA Full HSM/Medium HSM with Arm Cortex-R processors

Automotive networks show a clear trend towards the centralization of functionality. The consolidation of functions that were previously performed by individual ECUs, into a single ECU, allows for a significant reduction of the required number (and type) of in-vehicle ECUs, and therefore helps to mitigate the overall complexity. There are automotive standards and interfaces that have been designed, in order to create platforms where software from different suppliers can be progressively integrated, for example AUTOSAR.

Security architectures, based on virtualization technologies, use a single computing platform to securely execute multiple, independent (fully featured) runtime environments (also known as virtual machines) concurrently, while enabling an efficient sharing of available physical resources. In addition to the cost-efficient utilization of available hardware resources, the most important safety and security advantage that virtualized security architecture provides, is strong runtime isolation. This feature ensures that subsystems, components, or even individual applications can communicate only through strictly controlled communication channels, in a way that provides strong defenses against illegally accessing data, functions or affecting the execution or performance of other applications, without having proper authorization. To enforce strong isolation, virtualization architectures employ a very small, highly efficient kernel. In practice, this can be realized in hardware, in software, or by a combination of both.

Applying virtualization techniques on consolidated ECUs provides for separation of processes and applications. Crucially, these techniques allow both lower-cost deployment models and independent certification of applications, with different supply chains and different trust levels, to exist on the same ECU.

Figure 7 illustrates a consolidated ECU architecture. This consolidated ECU consists of a virtualized hardware platform, with an Armv8-R architecture CPU, running a hypervisor for securely scheduling and isolating a number of virtual machines (VMs). In addition, a medium profile HSM incorporating a dedicated Armv8-M architecture processor (full profile if the ECC and HASH engines are instantiated) is shown. Armv8-A architecture CPUs provide several options for implementing the most computationally intensive tasks.

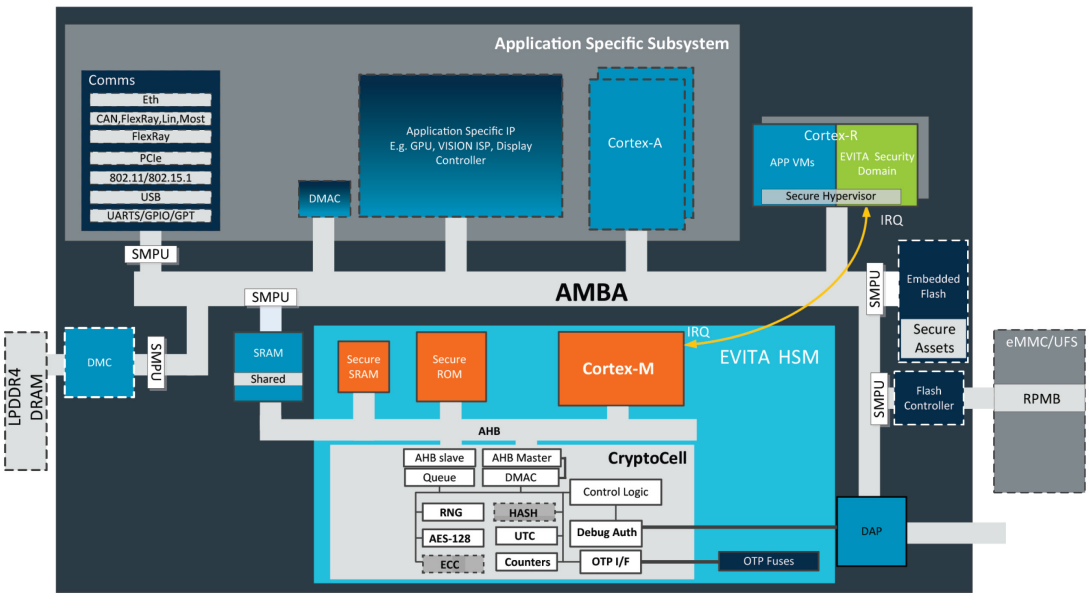


Figure 7 - A Hypervisor-based ECU design

Copyright © 2017 Arm Limited or its affiliates. All rights reserved.

A VM runs on the interface to the bare hardware, constructed by the hypervisor. Each VM runs its own software environment, which consists of a fully instantiated and configured application, with its run time environment (RTE) and operating system. Each VM is usually configured to host only a single automotive application (the firmware of a single-function ECU), and the RTE and OS that are included are adapted (that is, minimized) to the essential requirements of the application. This method promotes ease of porting and implementation efficiency, and also simplifies validation procedures, by isolating certification.

Figure 8 shows an example of software distribution for a multi-purpose ECU that is consistent with the EVITA framework. Here, the hypervisor implements and isolates a VM for each application domain and runs directly on the CPU hardware (type 1 hypervisor). The hypervisor can be implemented as a microkernel, to minimize the trusted computing base. The EVITA Security Domain is a trusted VM that includes modules that are required to provide security services for applications, together with platform integrity. These modules provide confidentiality, integrity authentication, access control, secure communication, and system monitoring. Platform attestation services might also be located here. One implementation option is to include a virtual EVITA HSM, to allow AUTOSAR VMs to interact with a dedicated virtual HSM, through a security domain virtual machine. This is to allow AUTOSAR VMs to be implemented, ported and certified within such an environment, with the highest efficiency in terms of time and security.

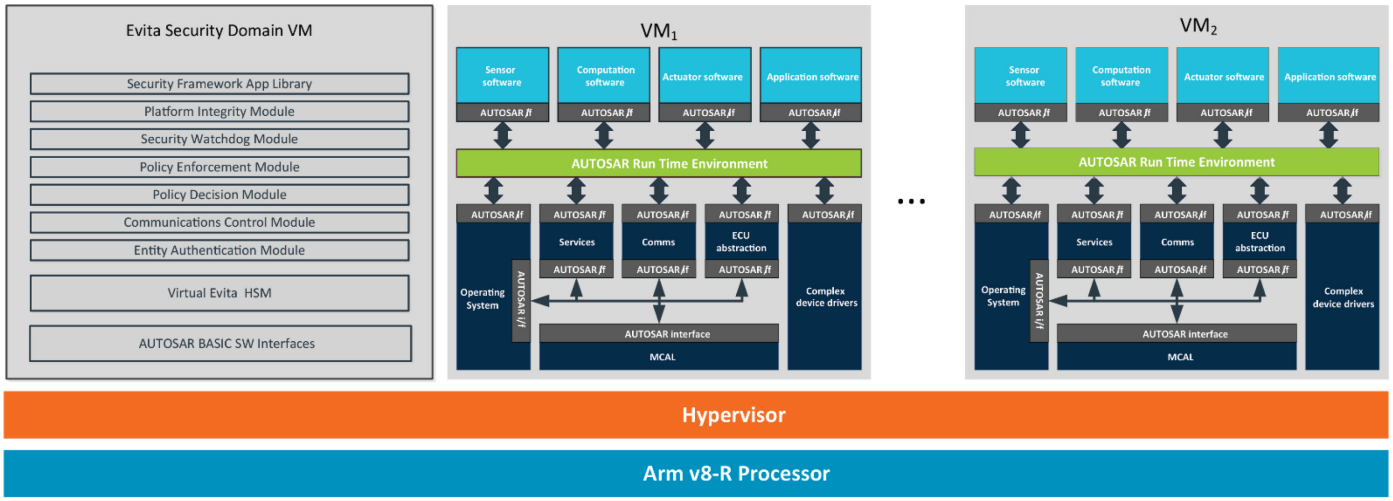


Figure 8 - Software distribution for a medium profile HSM

A head-unit or IVI design will exhibit a software architecture that is similar to Figure 8, with the possible addition of one or more VMs, for richer operating systems (than AUTOSAR), such as QNX or Linux. It must be noted that both Armv8-A and Armv8-R architecture processors can exploit the advantages of virtualization.

Copyright © 2017 Arm Limited or its affiliates. All rights reserved.

3.3. Armv8-R CPU architecture

An important difference between the current Armv8-A and Armv8-R processors is the memory system architecture. The Armv8-A profile uses the Virtual Memory System Architecture (VMSA) to provide virtual memory support. VMSA supports rich operating systems, such as Linux or Green Hills' INTEGRITY, by using translation tables located in memory and cached in a Translation Lookaside Buffer (TLB). In contrast, an Armv8-R processor uses the Protected Memory System Architecture (PMSA), to provide memory protection without translation. The non-deterministic behavior, caused by potential TLB misses, is avoided by use of a memory protection unit (MPU) that is based on registers, and that is tightly linked to the core. Real-Time Operating Systems (RTOS) can use the PMSA to provide memory protection, between tasks.

Armv8-R introduces support for hardware virtualization, through the provision of three Exception Levels (EL) that can be virtualized. When combined with a secure hypervisor, virtualization enables stronger isolation between applications than can be achieved within a single operating system, by virtue of having a smaller code base, that can be more strongly audited.

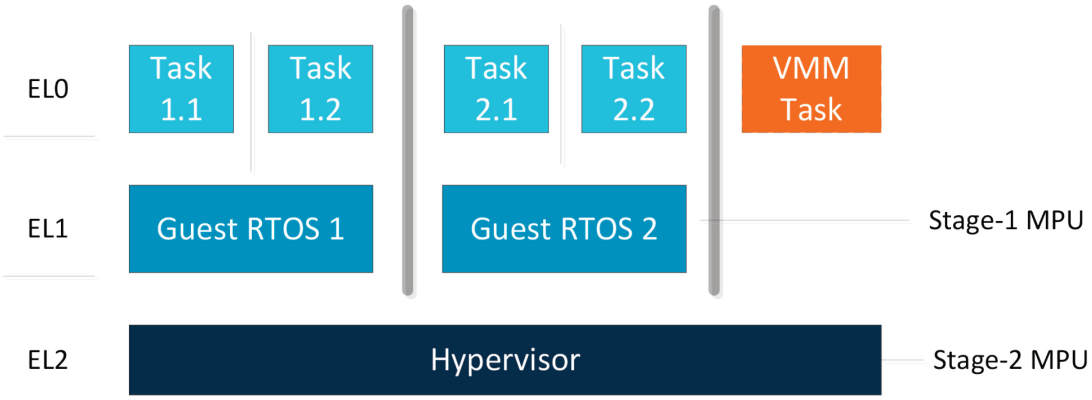


Figure 9 - Armv8-R 3 Exception Levels

Tasks or applications commonly run at EL0 and guest operating systems run at EL1, while hypervisors execute at EL2. Stage-1 MPUs are controlled by a guest OS, that allows kernels to manage their memory protection independently from the hypervisor, although all memory accesses are ultimately subject to memory permissions in the stage-2 MPU. Therefore, the integrity of the hypervisor code, and the address-space isolation of each guest operating system, rely on the stage-2 MPU.

Copyright © 2017 Arm Limited or its affiliates. All rights reserved.

4 Summary

In terms of silicon, certification and deployment infrastructure, the uptake of strong security measures can be expensive. However, the increasing cost of insecurity, in terms of OEM reputation and vehicle recalls, combined with the possibility that safety features relying on V2X communication might be accompanied by regulations, will increase the motivation for achieving meaningful, low-cost security.

The adoption of secure architectures, based on the EVITA framework, allows OEMs to standardize their security approach across the range of ECUs that they deploy. When drawing up requirements, system architects ought to consider that a common HSM architecture, that is reused across several ECU application types, can achieve considerable savings in NRE costs, as well as a significant reduction in certification time. An HSM using an Arm Cortex-M processor, CryptoCell accelerators and security management, provides a flexible, low-cost solution with sufficient longevity to be used across a generation of designs. TrustZone technology for Cortex-M processors can be used throughout low cost HSMs (trusted light architecture), to provide hardware isolation for security assets, without the full cost of implementing physically separate architecture resources.

For more complex ECUs, head units, and communication gateways, implementing a real-time host CPU, with strong virtualization support, is key to lowering cost. Virtualization allows the creation of trusted execution environments, that allow the secure co-habitation of mutually distrustful applications, on the same CPU. Together with enabling technologies, such as AUTOSAR, this simplifies the aggregation of several applications on a single ECU, and hence mitigates the silicon cost of the security and its supporting infrastructure.

For silicon providers, standardizing ECU offerings, on Arm-based HSM designs, can lower the cost of developing and certifying the security of their designs, while focusing on the highly differentiated application performance area.

All brand names or product names are the property of their respective holders. Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder. The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given in good faith. All warranties implied or expressed, including but not limited to implied warranties of satisfactory quality or fitness for purpose are excluded. This document is intended only to provide information to the reader about the product. To the extent permitted by local laws Arm shall not be liable for any loss or damage arising from the use of any information in this document or any error or omission in such information.

© Arm Ltd. 10.17