EVS30 Symposium
Stuttgart, Germany, October 9 - 11, 2017

Introducing Hardware Security Modules to Embedded Systems for Smart Charging (ISO/IEC 15118)

Fabian Eisele[1]

[1] *Vector Informatik GmbH, Ingersheimer Str. 24, 70499 Stuttgart, fabian.eisele@vector.com*

Summary

ISO/IEC 15118 uses X.509 certificates for the automated authorization of electric vehicles at the charging station. As such, confidential information has to be stored in the electric vehicle. To protect this data from unauthorized access and to unload the host controller from calculating cryptographic operations, Hardware Security Modules (HSM) are introduced. They secure certain areas of the device's memory and can execute required operations like the creation of signatures or encryption of data in a protected environment. This paper introduces the certificate usage of ISO/IEC 15118, the derived requirements to an HSM and assesses which type of HSM fulfils those requirements.

Keywords: V2G (Vehicle To Grid), Smart Charging, Standardization, EV (Electric Vehicle)

1 Introduction

Security is a topic that has gained increased attention over the last few years. Complete systems or databases have been compromised and millions of data sets containing personal information have been stolen. With the introduction of infotainment systems, backend connectivity and payment information stored in the vehicle, vehicles can become a target for hackers as well. If hackers succeed in discovering a backdoor in the vehicle's backend connectivity, they might be able to take over the car remotely or extract personal information. The same applies for the payment information that might be stored in the vehicle, so that the driver of the car is able to pay for charged energy. Therefore, it's more important than ever to secure sensitive data stored in microcontrollers (ECUs) all over the car. Since it's very easy to extract unprotected information from ECUs, some kind of protection mechanism is required.

Hardware Trust Anchors (HTA) are used to secure data stored in ECUs. They prevent unauthorized third parties from accessing the memory of the ECU. But as there are many different types of HTAs that could be used, the question is which HTA is the correct one for the intended use case. The three common types of HTAs, Secure Hardware Extension (SHE), Hardware Security Module (HSM) and Trusted Platform Modules (TPM), will be compared and analyzed to assess which one is best for the smart charging use case. To come to a result, it needs to be known what type of data needs to be secured and which cryptographic operations are required by the used components? These questions will be answered based on the use case specified by ISO/IEC 15118.

ISO/IEC 15118 is a standard that defines the charging communication between an electric vehicle (EV) and a charging station. ISO/IEC 15118 uses a high-level communication over powerline to exchange information between the two actors. For EVs using AC for charging, using high-level communication is

optional, albeit having its advantages. EVs that use DC for charging have to use high-level communication, since there is no other way to control the charging process. Next to the pure charging communication, ISO/IEC 15118 also defines authentication modes. The External Identification Means (EIM) authentication mode requires that the charging station itself offers a method of authentication, since the EV isn't able to authenticate itself. The other authentication mode, Plug and Charge (PnC) is more comfortable for the driver, since the EV will authenticate itself without any interaction required by the driver. To accomplish this, certificates need to be brought into the EV. This confidential information needs to be secured. The standard therefore needs to be analyzed with regards to its requirements to an HTA. Derived from those requirements, it can be determined what type of HTA needs to be used.

Chapter 2 gives an introduction to smart charging in general. Chapter 3 provides an overview over the certificate infrastructure that ISO/IEC 15118 defines. Based on this analysis, chapter 4 will derive the requirements for an HTA. Afterwards, chapter 5 will provide an introduction to HTAs in general, when finally chapter 6 will show how HTAs are used from within an AUTOSAR stack.

2 Smart Charging – An Overview

The term smart charging describes a high-level communication protocol for charging EVs, as compared to low-level charging via a PWM signal or by simply connecting to the power outlet. There are several protocols currently in use that have different capabilities and properties. The following list provides an overview over the smart charging protocols that are in use.

- ISO/IEC 15118
  - AC and DC charging
- DIN SPEC 70121 / SAE J2847
  - DC charging
  - High-Level Communication of DIN SPEC 70121 and SAE J2847 is compatible
- CHAdeMO / GB/T 27930
  - DC charging

While CHAdeMO and GB/T 27930 only communicate over CAN (Controller Area Network) and therefore have a limited bandwidth (0.5 to 1 Mbit/s for regular CAN), ISO/IEC 15118 and DIN SPEC 70121 are using PLC (Powerline Communication) specified by HomePlug GreenPHY[1], which offers a much higher bandwidth (100 Mbit/s). This higher bandwidth enables the latter standards to transmit complex elements such as schedules that contain information about the available power and its price over the next 24 hours, or payment information in a short amount of time.

DIN SPEC 70121 was derived from ISO/IEC 15118 before its completion by German vehicle manufacturers (OEM). As the release of ISO/IEC 15118 was delayed, the goal was to create DIN SPEC 70121 to be able to faster release a standard that focuses on DC charging and omits all optional features of ISO/IEC 15118. Initially, the premise was that in the end, DIN SPEC 70121 should be aligned with ISO/IEC 15118 to make them fully compatible, but this didn't play out. DIN SPEC 70121 deviates from ISO/IEC 15118 in some minor parts because technical reasons make them completely incompatible. Therefore the two standards are coexisting, but only ISO/IEC 15118 will be continued and extended in the future.

ISO/IEC 15118 is currently the only standard that offers an encrypted connection and automated payment as well as the exchange of detailed schedules that makes the charging of EVs plannable for both the driver and the infrastructure. The focus of this paper will therefore be only on this smart charging standard.

A new version (2nd edition) of ISO/IEC 15118 is currently being specified that will additionally support inductive charging (Wireless Power Transfer – WPT), pantograph charging (Automatic Connection Device – ACD) and Bidirectional Power Transfer (BPT).

[1] http://www.homeplug.org/tech-resources/green-phy-iot/

3 ISO/IEC 15118 – Certificate Usage

Before taking a look at the PKI of ISO/IEC 15118, the first sub-chapter will introduce how asymmetric cryptography works, explained by the Diffie-Hellman key exchange. This is important to understand why certificates are used in a PKI in the first place. The X.509v3 PKI that is used by ISO/IEC 15118 will be introduced with a focus on the involved certificates. Finally, it will be shown how ISO/IEC 15118 uses those certificates to create its PKI.

3.1 Diffie-Hellman key exchange

Today most of the communication happens between different hosts within the Internet. Those hosts usually don't have a protected end-to-end connection. Therefore, when trying to establish a secure connection, using only a symmetric encryption will not work. Symmetric encryption requires a secret that is known on both sides, since this secret will be used to encrypt the data. It is thus necessary to be able to agree on a secret on both sides, but over an unprotected connection.

One way to do this is by using the Diffie-Hellman key exchange. The name originates from Whitfield Diffie and Martin Hellman who published the idea in 1976[2]. Several steps are part of this key exchange. First each host creates a pair of keys. One of those keys will never be revealed to others, it's consequently called private key. The other key which will be sent to the other host and which will therefore probably be seen by 3rd parties is called public key. The public key is allowed to be known by anyone, this doesn't affect the security of this key exchange. Once the hosts have received the public key from their communication partner, they will use it together with their own private key to create a secret. This secret is called shared secret, since it will be the same for both communication partners, even when they didn't use the same keys, but always their own private key and the public key of the other party.

In theory there is a way for 3rd parties to extract the private key from the public key, but even with modern supercomputers, this mathematical problem would not be solvable in a feasible amount of time. This key exchange is therefore currently considered as being secure.

Since there are various participants around the world that communicate through the internet, simply transmitting the raw public key across it would lead to issues, such as not being able to identify which public key belongs to whom. This is the point where certificates come into the play, namely X.509v3 certificates that are used by ISO/IEC 15118.

3.2 X.509v3 Certificates

What kind of data has to be stored in the HTA and which cryptographic operations need to be supported depends on the use case. For charging communication, there is currently one standard available that makes use of cryptographic material. This standard is called ISO/IEC 15118. The part 2[3] document of ISO/IEC 15118, which was released in 2014, describes the communication protocol and the requirements for a Public Key Infrastructure (PKI) that can be used together with this protocol. A PKI describes how and by whom certificates are issued and how they have to be used.

The PKI of ISO/IEC 15118 is based on X.509v3 certificates, which are defined by an ITU-T standard[4]. X.509v3 certificates are widely used for electronic communication, especially for Transport Layer Security (TLS), which is used by all browsers to establish secure connections to Hypertext Transfer Protocol (HTTP) servers. Connections to HTTP servers that are protected by TLS are called HTTPS connections, where the 'S' stands for secure. Those secure connections were used mostly for online banking, but more and more websites start using HTTPS for privacy protection, i.e., to prevent 3rd parties from knowing which content has been accessed on a certain website.

X.509v3 certificates consist of several elements. Figure 1 gives an overview about the X.509v3 certificate in total and of the certificate part in detail.

[2] https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange
[3] https://www.iso.org/standard/55366.html
[4] http://www.itu.int/rec/T-REC-X.509/en

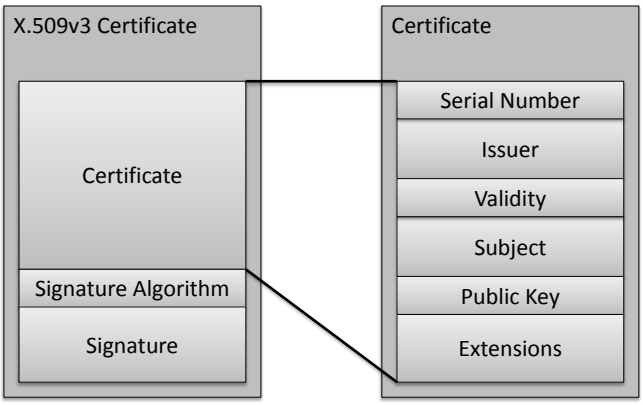

Figure 1: The structure of an X.509v3 certificate and of the enclosed certificate

On the root level are three elements that make up the X.509v3 certificate.

- Certificate
  - Contains the public information this certificate carries.
- Signature Algorithm
  - Describes which signature algorithm is used for the attached signature, e.g., Elliptic Curve Digital Signature Algorithm with SHA-256 (ecdsa-with-SHA256).
- Signature
  - Signature that was created by the issuer of this certificate. With the issuer's certificate it's possible to validate this signature and therefore confirm that this certificate is valid.

The actual certificate that is part of the X.509v3 certificate contains various kinds of information. The following list explains the most important parts.

- Serial Number
  - Unique for this issuer to be able to exactly identify this certificate.
- Issuer
  - Provides information about who issued this certificate. This information usually includes the company, its address and contact information.
- Validity
  - Every certificate is only valid during a certain period. Longer validity periods increase the risk of compromised certificates, i.e., certificates whose private key has been exposed, being used overly long, shorter periods require more frequent updates.
- Subject
  - Organization or individual for whom this certificate was created. Also usually contains information about organization, address and contact information.
- Public Key
  - This key can be used to perform asymmetric cryptographic operations, e.g., to validate signatures that have been created with the belonging private key.
- Extensions
  - Optionally extensions can be added that, for example, limit how many certificates can be derived from this certificate or for what purposes this certificate might be used.

3.3 PKI of ISO/IEC 15118

The PKI of ISO/IEC 15118 consists of several certificate chains that belong to different roles within ISO/IEC 15118. Those roles might either represent independent organizations or a single organization might fulfill multiple roles. As of the writing of this paper, there exists no actual implementation of the PKI, thus there is no practical experience yet as to how those roles will be distributed across the various actors of the market. Thus the following list only provides an overview about the roles as described by ISO/IEC 15118; they might be different for the real PKI.

- Charge Point Operator (CPO)
    - As the name suggests, the Charge Point Operator is the company that operates the charge points at which the EV can charge its battery. The CPO may be an independent company, but it can also belong to a utility.
- Certificate Provisioning Service (CPS)
    - The Certificate Provisioning Service is responsible for distributing the contract certificates that the owner of an EV has concluded with a Mobility Operator.
- Mobility Operator (MO)
    - The Mobility Operator is the entity that is in direct contact with the owner of the EV. These two parties agree on the terms of the contract, e.g., the price the EV owner has to pay for the energy.
- EV Manufactures (OEM)
    - The EV manufacturer is also an actor within the PKI, since the OEM needs to provide an initial set of certificates with every EV, otherwise the EV wouldn't be able to automatically install the contract certificate from the MO.
- 3rd Parties
    - It's possible for 3rd parties to create their own private environment, i.e., their own set of certificates. This makes them independent from the official ISO/IEC 15118 Private Key Infrastructure.

Figure 2 gives an overview of the certificates that are required for ISO/IEC 15118 and their relation with each other.

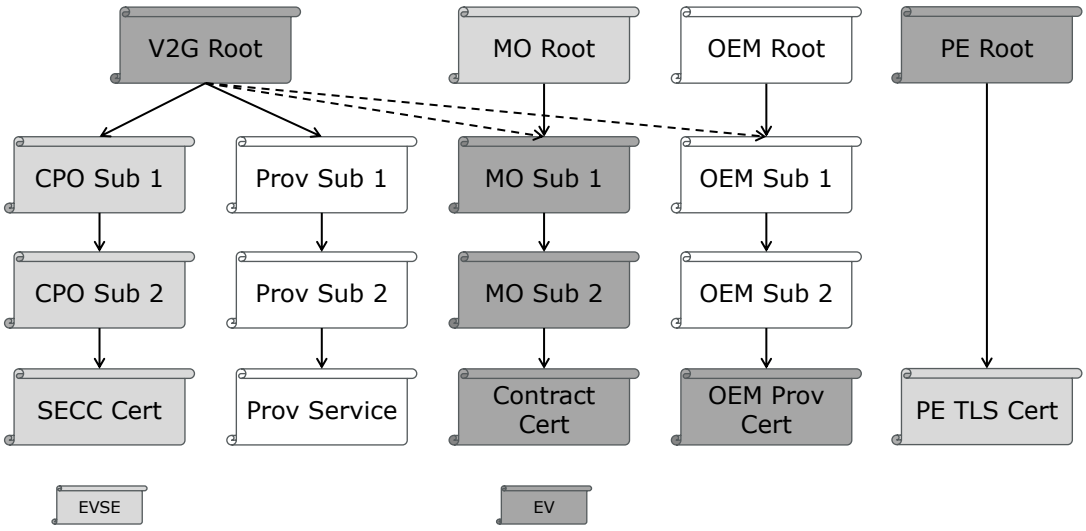

Figure 2: PKI according to ISO/IEC 15118 [Source: ISO/IEC 15118-2]

As shown in Figure 2, the PKI of ISO/IEC 15118 mainly consists out of four different certificate chains that all potentially trace back to the V2G Root Certificate.

- Charge Point Operator Certificate Chain (CPO Sub 1 to SECC Cert)
    - The certificate chain of the CPO is used within the charging station. It is used during the TLS connection establishment to authorize the charging station at the EV.
- Provisioning Service Certificate Chain (Prov Sub 1 to Prov Service)
    - The CPS uses its certificate chain to sign the new contract certificate chain that is sent from the MO to the EV. With the attached signature the EV can be sure that no-one in between the MO and the EV has modified the contract certificate chain.
- Contract Certificate Chain (Mo Sub 1 to Contract Cert)
    - In order for the EV to identify itself at the charging station and the MO respectively, the EV needs some form of authentication. In case Plug and Charge (authentication profile of ISO/IEC 15118) is used, the EV provides its contract certificate chain that was provided

by the MO for authentication, as compared to the driver having to show his RFID card in case of the External Identification Means (EIM) authentication profile.

- OEM Provisioning Certificate Chain (OEM Sub1 to OEM Prov Cert)
    - The OEM needs to provide at least one unique provisioning certificate per EV. The sub CA certificates are not necessary to be stored in the EV. With this certificate the owner of the EV can provide a means of identification to the MO. Because when there is no valid contract certificate installed in the EV yet, the MO needs some method of determining which customer has to receive which contract certificate.

Since this PKI offers a contract certificate that uniquely identifies the driver of the car as a customer of a certain Mobility Operator, the driver doesn't need to carry some other form of authentication method with him in form of, for example, an RFID card. It's also more easily possible to define validity periods of the certificates compared to RFID cards. Those validity periods allow, e.g., offline charging stations to let EVs charge. Because once the current contract certificate has expired, it may only be renewed as long as the contract with the MO is still valid. In case of an RFID card this would imply that the RFID card would need to be exchanged ever so often or the customer would need an RFID card reader to change the certificate on the card. Both solutions are not comfortable for the customer of the Mobility Operator, since there would always be an interaction necessary. But with the contract certificates in the EV that are also autonomously handled by the EV, the customer doesn't need to do anything. This exchange of payment information between the EV and the charging station is typically referred to as Plug and charge (PnC).

ISO/IEC 15118 currently specifies a new use case that drives the usage of PnC authentication mode. This use case is inductive charging. Conductive charging requires still a lot of interaction from the driver compared to vehicles with combustion engines. In case the battery needs to be charged, the charging cable needs to be connected to the charging station and when not supporting PnC, the driver needs to authenticate himself at the charging station. The argument against PnC often was that the driver needs to approach the charging station anyway; therefore, it's not that much of an additional effort to present the RFID card. But with inductive charging, the driver has even less interaction than with vehicles using combustion engines. The driver, or later the autonomous EV, will only need to park at a parking lot that offers inductive charging. The EV will automatically establish a wireless communication with the charging station, exchange the authorization credentials and start charging. There is no need for a further involvement of the driver.

4 ISO/IEC 15118 – Derived Requirements

Having introduced smart charging in chapter 2 and the corresponding certificate usage in chapter 3, this chapter derives the corresponding requirements towards an HTA for the given scenario. There are mainly two functionalities that define the requirements to an HTA. One is TLS and the other is the installation or update of contract certificates.

4.1 Transport Layer Security (TLS)

TLS supports many different algorithms to encrypt the communication channel. Additionally, it's not just that a single algorithm will be used during the lifetime of a TLS session. Various algorithms need to be selected, e.g., how the key exchange shall happen, how the data shall be encrypted or how the checksum shall be calculated. In order to reduce the amount of possible combinations TLS defines several so-called cipher suites. Those cipher suites define which algorithm shall be used for which step. ISO/IEC 15118 specifies that the following two cipher suites need to be supported.

- TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256

Based on the names of the cipher suites, the following requirements can be derived.

- Elliptic Curve Diffie Hellman (ECDH[E])
    - The Diffie Hellman key exchange needs to be supported for elliptic curves. It is used to agree on a shared secret which is then used for the symmetric encryption of the data.

    - In case of ECDH, the server (charging station), uses a static key pair while the client (EV), needs to create a new key pair for each session.
    - The additional ‘E’ in the ECDHE cipher suite denotes that the server needs to use an ephemeral key pair for the Diffie Hellman key exchange, too.
- Elliptic Curve Digital Signature Algorithm (ECDSA)
    - ECDSA needs to be supported to be able to generate and validate signatures that guarantee authenticity of the communication partner and integrity of the sent data.
- Advanced Encryption Standard 128 (AES-128)
    - AES-128 is used to encrypt and decrypt the data, i.e., the messages defined by ISO/IEC 15118. The secret that is used as the key has been determined by ECDH[E]. The used block cipher mode is Cipher Block Chaining (CBC).
- SHA-256
    - TLS uses a hash-based message authentication mode (HMAC) to guarantee integrity during the transmission. The HMAC can use MD5, SHA-1 or SHA-256. The cipher suites supported by ISO/IEC 15118 use SHA-256.

An HTA therefore needs to support asymmetric cryptography and needs to be able to store asymmetric credentials in form of private and public keys, public keys preferably in the form of X.509v3 certificates.

4.2 Installation and Update of Contract Certificates

The installation and update of contract certificates uses similar cryptographic operations as TLS. But there are additional requirements, since the installation and update of contract certificates deviates from the normal procedure. Usually, a key pair that consists of a public and private key is created locally at the owner of the new private key. The owner then sends only the public key to a certificate authority (CA) and receives a certificate that uses the provided public key. With this approach, the private key never has to leave the owner’s control. Using this normal approach was deemed to be unfeasible by ISO/IEC 15118. It would require the PKI to be able to issue a new contract certificate within seconds after the EV has requested this from the charging station. The short time limit originates from not wanting the driver of the car having to wait too long until the EV is able to signal that the charging process has started successfully. Since a rather complex architecture of the PKI was expected and it was expected that the process for creating a new certificate the usual way would take too much time under those conditions, another approach was defined.

The contract certificate as well as its private key are created by the Mobility Operator. It’s therefore necessary to securely transmit the private key to the owner of the contract certificate. To make this possible, the private key needs to be encrypted, which is done in the following way.

- An asymmetric key exchange is done using ECDH[E].
- The symmetric key is derived from the shared secret by using the concatenated key derivation function (KDF) according to NIST Special Publication 800-38A.
- The Mobility Operator uses symmetric cryptography (AES-128) to encrypt the private key.
- The EV receives the encrypted private key and now needs to decrypt the private key and store it securely.

The concatenated KDF and the last step are functionalities that are usually not supported by an HTA. The ability to provide encrypted data, decrypt it within the secure area and set it as a new private key, without removing the decrypted private key from the secured area between those two steps. Normally data that shall be decrypted will be removed from the secure area once it’s done, since the user of the HTA needs to work with this data. But the decrypted private key shall never leave the secured area. The reason is that everyone that has access to the private key might charge on the costs of the actual owner of the Contract Certificate.

Therefore, an additional requirement to an HTA is, next to supporting the concatenation KDF, being able to decrypt and set a private key in a single step. The algorithms for this are already required by TLS.

4.3 Impact on Embedded Systems

ISO/IEC 15118 has high demands on the cryptographic capabilities of a system acting as the communication controller of the EV, a so-called Electric Vehicle Communication Controller (EVCC). Especially three tasks are very demanding.

- Secure storage of the private keys of the contract certificate and OEM provisioning certificate
- Generation and validation of ECDSA signatures
- Encryption and decryption of data using AES-128

Without the hardware support provided by an HTA, processing of especially asymmetric cryptographic operations takes a relatively long time. Generating or validating an ECDSA signature on a common microcontroller, which has a clock frequency of roundabout 120 MHz, takes over 100 milliseconds. This is a very long time for embedded systems, considering that a task usually has a period between 5 to 20 milliseconds. Mechanisms that make sure that the timing constraints are held, like a watchdog, will fail in case such a processing is done synchronously. An HTA with a dedicated core and hardware acceleration can do such operations quicker and it has a smaller impact on the host controller. This prevents issues, e.g., with bus systems like CAN that are sensitive to deviations in the timing behavior.

5 Hardware Trust Anchors – General Introduction

Hardware Trust Anchors (HTA) play an important part in making sensitive data inaccessible to unauthorized parties. While in the past, sensitive data may have been securely stored in non-volatile memory, during runtime, it was always accessible, since otherwise the ECU couldn't have worked with it. Using HTAs, however, it is now possible to secure predefined areas of the ECU's memory (both volatile and non-volatile) to store sensitive data. In addition, some HTAs can offload the host controller of the ECU by calculating security operations on their own either via a dedicated hardware accelerator or on a separate core.

The exact capabilities of an HTA depend on its type. There are mainly three different types of HTAs:

- Secure Hardware Extensions (SHE)
  - Released in 2009 as public specification by the "Hersteller Initiative Software (HIS)", a German OEM consortium, now organized by AUTOSAR.
  - Realized as an on-chip extension within an MCU, providing a set of cryptographic services to the application layer. It isolates secret keys from the rest of the MCU resources.
  - Cryptographic services such as encryption & decryption, random number generation or boot loader verification are based on AES-128.
- Hardware Security Modules (HSM)
  - Hardware Security Modules describe dedicated computing devices that are able to securely store cryptographic material and provide hardware support for cryptographic operations.
  - Special requirements for the automotive industry were defined in an EU-sponsored project called EVITA[5] by a consortium that consisted amongst others of Bosch, BMW and Infineon. The specifications are available on the EVITA website.
  - The design objectives were to harden ECUs against attacks, provide hardware acceleration for cryptographic services by offloading the application core and support ECU to ECU communication protection.
  - Three different HSM profiles were defined to make the HSM solution scalable. The "small" profile only supports simple block ciphers to secure critical sensors and actuators. But therefore the modules are low cost. The "medium" profile can be used for secure ECU to ECU communication, since it supports asymmetric cryptographic operations, which are required for a secure key exchange over an unsecured connection. The "full" profile

[5] https://www.evita-project.org/

supports even stronger authentication, e.g., elliptic curves (ECC) and complex block ciphers, and has a high performance.

- Trusted Platform Modules (TPM)
    - Developed by the Trusted Computing Group[6], a consortium of different companies of the IT sector, such as AMD, IBM, Intel and Microsoft. Specified in 2009 as ISO/IEC 11889.
    - A TPM is a powerful, dedicated hardware, which wasn't used in the automotive industry until recently. The usual "smaller microcontrollers" don't support TPMs, but in the future each vehicle will have one or few "larger microcontrollers" that run Linux, which will use TPMs. But since the smart charging functionality usually resides on the smaller microcontrollers, TPMs will not be further discussed.

Currently SHEs are still mostly used when an HTA is required. But as security demands on ECUs rise, HSMs will be required. Table 1 gives a detailed overview about the supported features of different HTAs. For the simplification of the table, SHEs and small HSMs share a column, since they have the same properties.

Table 1: Comparison of the SHE/HSM(small) against the HSM(medium) and HSM(full)

	SHE ~ HSM (small)	**HSM (medium)**	**HSM (full)**
Ensure Confidentiality	Yes	Yes	Yes
Ensure Integrity	Yes	Yes	Yes
Secure Storage of Symmetric Crypto Material	Yes	Yes	Yes
Secure Storage of Asymmetric Crypto Material	No	Yes	Yes
Dedicated CPU	No	Yes	Yes
HW Support for Symmetric Crypto Operations	Yes	Yes	Yes
HW Support for Asymmetric Crypto Operations	No	No	Yes
Firmware Changes Possible	n/a	Yes	Yes

As the table shows, SHEs and small HSMs have a limited set of features. Thus they are the best choice when a cost effect solution is desired and there is no need to support asymmetric cryptography. Since SHEs and small HSMs don't have a dedicated CPU, they also don't have a changeable firmware. It's thus not possible to extend the feature set at a later point in time. "Medium" HSMs have the advantage of being updateable and extendable because the firmware might be changed at a later time. Asymmetric cryptographic operations are supported, albeit not in hardware. "Full" HSMs are required when demanding asymmetric cryptographic operations are to be expected and the HSM shall be used in the long-term, e.g., when it shall have power reserves. Of course the cost of a "full" HSM is higher than that of a "medium" HSM.

Figure 3 and figure 4 provide a comparison of a "small" HSM / SHE and a "medium" HSM. As can be seen, the "small" HSM only has its host controller and a peripheral hardware module for calculating AES-128. The "medium" HSM doesn't only consist of a peripheral module, but has its own core that executes code independent from the host controller. This separate core has its own memory and hardware modules that allow hardware acceleration for cryptographic operations.

[6] https://trustedcomputinggroup.org/

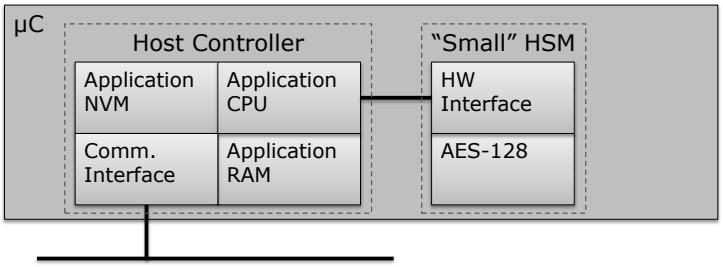

Figure 3: Topology of a "small" HSM / SHE, based on figure from [7]

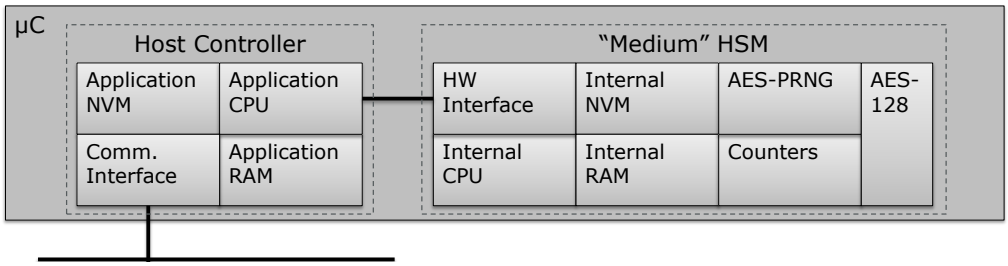

Figure 4: Topology of a "medium" HSM, based on figure from [7]

6 Hardware Trust Anchors – Utilization within AUTOSAR

In general, the availability of HTAs is good. But there are different architectures that define how HTAs can be used within a software stack. If standardized software components shall be used that allow access to the HTAs functionality, a well-defined architecture should be used. Such an architecture is provided by AUTOSAR, short for Automotive Open System Architecture, which is broadly used in the automotive industry and supports the usage of HTAs.

The usage of HTAs has been supported by AUTOSAR already with previous versions, but with the current version 4.3 AUTOSAR's crypto stack was reworked. Figure 5 shows the difference between the two architectures:

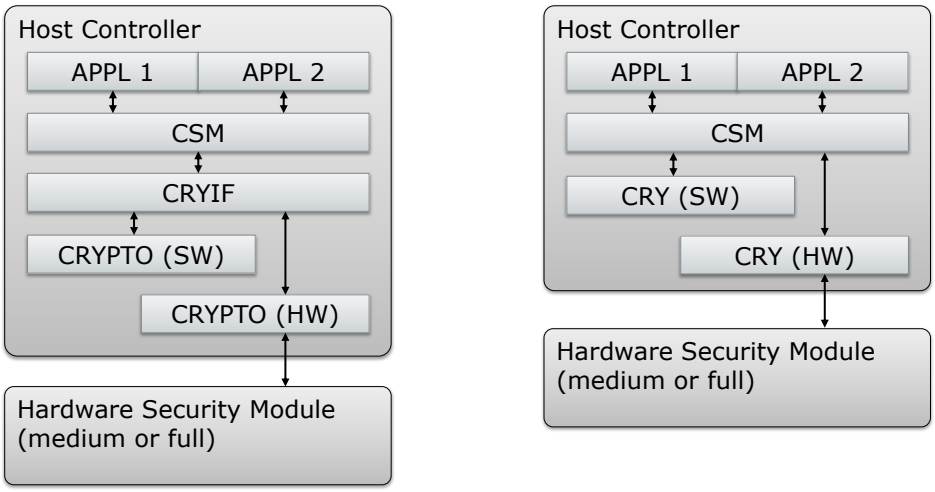

Figure 5: Comparison of the AUTOSAR crypto stack defined by version 4.3 (left) and version 4.2 (right) respectively

Visible on the architecture is one main difference, the introduction of the CRYIF (Crypto Interface) as an interface layer. This will be explained later. Otherwise the architecture remained quite similar, even when

[7] https://www.evita-project.org/Publications/EVITAD1.2.5.1.pdf

there have been major changes that greatly improve the functionality. Three new features shall be highlighted, which are the following:

- API parameter changes
- Introduction of a job queue
- Possibility to mix synchronous and asynchronous services

APIs that were using private keys required a pointer to the memory where the private key is located. Those APIs used the private keys for various cryptographic operations, e.g., encrypting data or generating signatures. In order to be able to provide a pointer to the actual private key, the private key would need to be in unprotected memory, which is not secure. It was therefore necessary to provide an alternative. This alternative has been introduced with a workaround already to AUTOSAR 4.2. Instead of providing the address of the private key, only a reference to the private key in the HTA was provided. The API was thus changed with AUTOSAR 4.3 so that now it's not necessary anymore to misuse the pointer parameter to pass on a reference, but that the parameter is only a reference from the start.

The introduction of a job queue was also important for a software stack with multiple users of the crypto stack to function properly. In version 4.2 each service could only be triggered once at a time, meaning that when a certain user was always second in row to trigger services, the first user could always block the service by continuously using it. To avoid such starvation scenarios, version 4.3 introduced a job queue. Through this queue it was now possible for all users to queue their service requests, independent of their position in the call path and even when the service is currently in use.

Finally, the possibility of mixing synchronous and asynchronous services helps to improve the performance of the system. Version 4.2 only supported one type of API, which led to a difficult decision for architects. When only using synchronous services, the system might be blocked by a certain services for a prolonged time, like the 100 milliseconds mentioned in chapter 4.3. But when switching to asynchronous services, it would mean that services that only run for a short time lead to an unproportionally high delay for the user, since after triggering the asynchronous service, the user needs to wait for his next turn. With version 4.3 the handling of each service can be configured independently of other services.

It's therefore recommended to use AUTOSAR 4.3 for a smart charging ECU. The reason is that smart charging has multiple users, e.g., TLS and the components that handle smart charging and it uses cryptographic operations that have execution times that vary greatly.

The components that are used by the AUTOSAR 4.3 crypto stack are, as shown on Figure 5, the Crypto Service Manager (CSM), Crypto Interface (CRYIF) and Crypto Driver (CRYPTO). The CSM[8] component offers the supported cryptographic algorithms to the users, which may be application components or other basic software modules. Compared to the old architecture of AUTOSAR 4.2 it's now possible to have multiple users using the CSM, since it now has a job queue. The CRYIF[9] abstracts the hardware or software dependent CRYPTO modules. For the CSM and its users it's therefore not of relevance what kind of HTA is used, whether it's only a SHE that has an additional software cryptographic library to handle the asymmetric algorithms or whether it's a "full" HSM. For each type of software library or HSM (sub-type of HTA), a unique CRYPTO[10] is necessary. This is because the firmware of the HTAs usually has a different interface, which makes other drivers incompatible.

However, it must always be kept in mind that introducing an HSM to an ECU doesn't only have advantages. Because of the separate core, more effort needs to be put into coordinating the access to shared resources. It's also essential that the HSM supports all cryptographic algorithms required by the ECU, as no sensitive information shall leave the HSM. But when the goal is to use automated payment and to securely store the cryptographic material required for this, the advantages of using an HSM clearly outweigh the disadvantages.

[8] AUTOSAR_SWS_CryptoServiceManager.pdf
[9] AUTOSAR_SWS_CryptoInterface.pdf
[10] AUTOSAR_SWS_CryptoDriver.pdf

7 Conclusion

The goal of this paper was to analyze the requirements of ISO/IEC 15118 in regards to security. At first chapter 2 gave a general introduction to the topic of smart charging. Then chapter 3 introduced how ISO/IEC 15118 uses cryptography at all. Afterwards chapter 4 collected the cryptographic algorithms that need to be supported by an ECU wanting to use ISO/IEC 15118 with the PnC authentication mode. Since HTAs are probably the best way to fulfill those requirements, chapter 5 gave an overview about the available HTAs while focusing especially on SHEs and HSMs and determining which type of HTA will be required for this use case. Finally, chapter 6 showed how those HTAs can be used in an ECU for the automotive industry.

The derived requirements were:

- Support for symmetric and asymmetric cryptographic operations
  - ECDH[E], ECDSA, SHA-256, AES-128
- Support for key derivation functions
  - Concatenated KDF
- Secure storage of asymmetric cryptographic material
  - Private and public key of X.509v3 certificates
- API that decrypts and stores private keys with a single call

Those requirements demand at least a “medium” HSM. A “full” HSM is recommended to reduce runtime of asymmetric cryptographic operations through hardware acceleration.

References

[1] HomePlug GreenPHY, http://www.homeplug.org/tech-resources/green-phy-iot, accessed on 2017-06-28

[2] DH Key Exchange, https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange, accessed on 2017-06-28

[3] ISO/IEC 15118-2:2014, https://www.iso.org/standard/55366.html, accessed on 2017-06-28

[4] X.509v3, http://www.itu.int/rec/T-REC-X.509/en, accessed on 2017-06-28

[5] EVITA Project, https://www.evita-project.org, accessed on 2017-06-28

[6] Trusted Computing Group, https://trustedcomputinggroup.org, accessed on 2017-06-28

[7] EVITA Workshop (01.10.2010), https://www.evita-project.org/Publications/EVITAD1.2.5.1.pdf, accessed on 2017-06-29

[8] AUTOSAR Release 4.3.0, Specification of Crypto Service Manager, AUTOSAR_SWS_CryptoServiceManager.pdf

[9] AUTOSAR Release 4.3.0, Specification of Crypto Interface, AUTOSAR_SWS_CryptoInterface.pdf

[10] AUTOSAR Release 4.3.0, Specification of Crypto Driver, AUTOSAR_SWS_CryptoDriver.pdf

Author

Fabian Eisele started working for Vector in 2012. As a software development engineer, he was responsible for developing AUTOSAR basic software according to the standards ISO/IEC 15118 and DIN SPEC 70121. Additionally, he was responsible for holding technical workshops on-site at international customers, explaining the basics of smart charging and integrating Vector’s embedded V2G solution. In 2014 he joined the standardization activities of ISO/IEC 15118. He was part of a small team that prepared the working draft of the 2nd edition of ISO/IEC 15118 that is currently being specified and is now responsible for its XSD schema. Since 2016 he is the product manager for Vehicle to Grid (V2G) at Vector’s embedded software department and is thus the central contact for all topics related to embedded software for e-mobility at Vector.