13AE-0069

Secure Key Management – A Key Feature for Modern Vehicle Electronics

Christian Schleiffer, Marko Wolf, André Weimerskirch, and Lars Wolleschensky
ESCRYPT

ABSTRACT

The need for vehicular data security and privacy protection is already enormous and increases even further. Prominent application areas are for instance theft protection, anti-counterfeiting, secure data storage and secure communication inside the vehicle and from the vehicle to the outside world. However, most of the vehicular security and privacy protection solutions involve modern cryptography and require availability of cryptographic keys in the vehicle and in related backend infrastructure. A central aspect for ensuring this availability and a controlled usage of such cryptographic keys is a secure key management, which affects the whole lifecycle of the key, from creation and distribution, usage, backup and update up to key deactivation. Even though secure key management is quite well understood in the standard computer world, the situation in the automotive world is quite different, as we have different functional requirements (e.g., sporadic low-bandwidth connections) and different security requirements (e.g., physical insider attacks).

We hence analyze the requirements and give best practice approaches for a secure key management solution in the automotive context. We highlight potential security concerns that are encountered during each phase on a lifecycle of a cryptographic key used in a typical vehicular security solution. Knowing the security vulnerabilities, we will introduce open solutions and best practice approaches for secure key management implementation both in the embedded in-vehicle domain as well as for the supporting backend infrastructure.

INTRODUCTION

There is an increasing demand for data security in vehicles. Vehicular data security use-cases include theft protection, protected in-vehicle communication, secure software update, secure infotainment systems, as well as secure communication of vehicles to the outside world. The main security goals behind these use-cases are protecting functional safety, protecting business models, or enforcing legal requirements. Most of the security solutions involve modern cryptography and require availability of cryptographic keys in the vehicle and in related backend infrastructure. Since such cryptographic keys determine the reliability of a security solution, they have to be protected accordingly. This protection affects the whole lifecycle of the key, which goes from secure creation and distribution, secure usage, backup and update up to secure deactivation. Key lifecycle management is quite well understood in the PC world, with for instance SSL connections protecting financial communication today (e.g. online banking). However, the situation is very different in the automotive world, where we have different functional requirements (e.g., millions of different keys for ensuring privacy for vehicle-to-X use cases), and different security requirements (e.g., inside attackers). Hence, we need adequate key management solutions inside the vehicle and for related backend servers. Secure in-vehicle key management comes down to a proper in-vehicle key concept together with a secure distribution and the secure in-vehicle usage and storage of keys. Secure in-vehicle key management is typically realized by dedicated security controllers. However, there are only few automotive grade security controllers available. A low-cost but powerful hardware extension is the Secure Hardware Extension (SHE) specified by the German HIS consortium (Herstellerinitiative Software). SHE is a proper solution for security functions such as secure boot, secure on-board communication, component theft, and counterfeit protection. Other, more powerful hardware extensions are still in the concept phase. A sound in-vehicle key management will minimize likelihood and impact of successful key extraction (e.g., by avoiding global keys that are used in more than one vehicle). Secure backend server key management for vehicles usually requires high scalability, efficient and specially adapted data format, and a well-elaborated key distribution approach that can support different offline scenarios. While typical solutions for the PC world support only one or few certificates per node (e.g. per webserver or per email sender), vehicles often require several keys, up to thousands of keys, for instance, to realize privacy-preserving vehicle-to-X communications. Vehicle applications further do not only require typical PC-style certificates (X.509 certificates) but a variety of cryptographic keys, including X.509 certificates, wireless transport layer security (WTLS) certificates, plain public/private key pairs, and symmetric keys. Furthermore, the processes to inject keys in vehicles are very different to the PC

world. Often, keys need to be injected by a supplier in a manufacturing plant, possibly in another country or continent.

The article will analyze the special needs and constraints regarding security and functionality for key management realizations in automotive domain. Based on this analysis, the article will suggest a secure and efficient key management approach for both in-vehicle and backend server for today's and future vehicular data security applications.

BACKGROUND

This section gives a short introduction into the terms and processes used within this article.

Cryptographic Keys

This subsection shortly describes the central objects of a key management solution, the cryptographic key material, and corresponding certificates (if applicable).

Symmetric Keys

Symmetric keys are randomly created secret bit strings k_{SEC} of typically 56 to 256 bits (depending on the level of security) used in symmetric crypto algorithms such as DES and AES for ensuring information confidentiality via symmetric encryption and/or information authenticity via message authentication codes (MAC). In contrast to asymmetric cryptography (cf. next subsection), symmetric crypto algorithms need the same secret k_{SEC} used for data encryption also during data decryption, and similar, the same secret k_{SEC} used for MAC generation also during MAC verification. The deployment of symmetric keys hence requires a secure distribution of the secret k_{SEC} and makes k_{SEC} especially susceptible to unauthorized read-outs.

Asymmetric Keys

Asymmetric keys are well-defined mathematical structures (e.g., special prime numbers) securely created by an individual key generation scheme which belongs to the respective asymmetric crypto algorithm such as RSA or elliptic curve cryptography (ECC). In fact, asymmetric keys are normally a pair of two associated keys, which means, a public k_{PUB} and a corresponding private k_{PRV}. Asymmetric crypto algorithms are also used to ensure confidentiality via asymmetric encryption and/or authenticity of information via digital signatures. However, in contrast to symmetric cryptography, different keys are used during encryption (k_{PUB}) and decryption (k_{PRV}) as well as for signature generation (k_{PRV}) and signature verification (k_{PUB}). The deployment of asymmetric keys hence considerably eases the key deployment problem, since for data encryption and signature verifications only k_{PUB}, which contains no secret information, is involved.

Certificates

Certificates comprise various metadata such as issuer, lifetime, or usage restrictions (e.g., who, when, where, what) associated to a certain (public) key. To prevent any unauthorized modifications of this certificate data, this certificated itself is protected by cryptographic measures (e.g., MAC or digital signatures), which in turn involve further cryptographic keys (cf. key hierarchies in the subsequent section).

Key Lifecycle

This subsection gives a short introduction of the stations during a typical lifecycle of a cryptographic key relevant for most key management solutions.

Creation

Mostly new crypto keys are created by applying a (true) random number generator either directly to create a random bit string (cf. symmetric keys) or the random numbers are used as input for a key generation scheme, which belongs to the respective crypto algorithm (cf. asymmetric keys). Moreover, new keys can be created using key derivation functions, which take already existing "master" keys and other metadata as input. Key derivation enables everyone in possession of the master key to create the same derived keys, for instance, to build key hierarchies. Key derivation functions are usually realized by specially adapted one-way functions using cryptographic encryption or hashing algorithm as building blocks.

Distribution

Once keys are created, they become securely delivered to their actual usage location. The key distribution channel usually has to be protected against disclosure, manipulation, malicious substitution, or replay, which often involves cryptographic operations based on other key material in turn or strong organizational protection measures (e.g., trusted carrier).

Store

Once securely delivered to their actual usage location, the keys have to be stored securely, which usually means protected against disclosure, manipulation, malicious substitution, or deletion. Secure is usually realized by physical protection (e.g., protected memory) or involves again other cryptographic protection mechanisms, which involve other keys in turn.

Store operations can include key update, which means authorized overwriting of existing key material, and key backup, which means storing the key to another location to improve availability.

Use

Use of cryptographic keys means applying them as one input to cryptographic functions such as encryption, decryption, hashing, digital signing, or signature verification. The cryptographic key usage involves usage restrictions, which have to be enforced by requesting and verifying necessary usage authorization before applying the requested key usage.

Deactivation & Deletion

In the last step of a key lifecycle, the key becomes unrecoverable destroyed or disabled (at a certain location) often proven to the key owner with a cryptographic receipt.

SECURE VEHICULAR KEY CONCEPT

There is an increasing demand for cryptographic key management in vehicles to strengthen existing use-cases and to implement future applications. In this section, we introduce some exemplary use-cases and list security and functional requirements of a key management solution in the automotive context.

Use Cases

Security mechanisms are already used today, in particular for

1. theft protection and counterfeit protection, and
2. secure software updates.

Use case 1 is ideally based on cryptographic mechanisms and can be implemented using a symmetric or asymmetric key scheme. Potential protocols and algorithms are described in [9]. It seems reasonable to use a symmetric key mechanism due to the low overhead in terms of expected cost and performance demands. The consortium HIS (Hersteller Initiative Software) of German OEMs specified the Secure Hardware Extension (SHE) [6] that provides a low-cost hardware extension suitable for theft protection and counterfeit protection. SHE offers an AES engine and several automotive-grade SHE-compliant microcontrollers are already available (e.g. in Infineon's TriCore series). A basic key management scheme will use a single symmetric key per vehicle, and each electric control unit (ECU) in the vehicle that implements the theft protection mechanism will learn that key at the assembly line or before being installed in the car. This key needs to be generated and it is wise to store it externally in order to ease the process to replacing ECUs to the vehicle. Counterfeit protection ideally requires an individual key per ECU in order to detect cloning attacks. Due to low-cost and little performance overhead requirements, it might be advantageous to use symmetric keys in this application as well. It might also be advantageous to store these keys in a server, together with the ECU's serial number, in order to be able to validate authenticity of ECUs and detect counterfeits in case of doubt.

Secure software updates are already found somewhere in most cars today, often for infotainment systems and possibly for safety critical and powerful ECUs. Secure software updates are typically implemented based on RSA digital signatures. For each ECU, a private/public key pair is generated. The public key is stored in the ECU and the private key is stored in a server. The server then uses the private key to digitally sign firmware, and the ECU applies the public key to digitally verify the firmware. Details are explained in [8]. In order to reduce the number of key pairs, it is reasonable to generate key pairs that are used by more than one ECU, e.g., one key pair for each ECU model and year.

Future applications might require even more cryptographic keys in vehicles. For instance, vehicle-to-vehicle communication to implement cooperative safety applications might require numerous cryptographic keys per vehicle. It is security practice to use separate cryptographic keys for each application such that it can be foreseen that each vehicle will be loaded with numerous cryptographic keys. This does not only apply to passenger vehicles but also to trucks, construction machines, etc.

Requirements and Best Practice

This section lists security and functional requirements as well as best practice design rules of secure key management.

Key Management Design

It is wise to never use global keys, i.e. keys that are globally used in vehicles or ECUs. This is also the case if a global key is stored in ECUs in order to derive ECU-individual session keys, even if the global key is never applied directly. At the same time it is wise to use a separate key for each application. For instance, separate keys shall be used for encryption and authentication, and separate keys shall be used for a secure software download and a theft protection application.

Cryptographic Algorithms

We recommend the use of standardized cryptographic algorithms, such as AES, RSA, ECC, and SHA-2. The same holds for random number generators (RNG) that are often an unexpected source of security weaknesses. There is hardly ever a need to use a proprietary algorithm, and there are many examples of proprietary algorithms that were compromised (e.g. [2]). Most standardized cryptographic algorithms can be implemented efficiently in software and hardware, and usually cryptographic hardware accelerators are available. Stream ciphers are an exception since there are no standardized stream-ciphers available. Stream ciphers might be of interest to automotive applications that have high performance or low cost requirements. However, there are recommended stream

ciphers, e.g. the ciphers defined in the eSTREAM portfolio [3].

Optimize Performance

It is wise to optimize performance of the key management to minimize cost. Typical key management server solutions provide digital X.509 certificates for use in asymmetric cryptographic schemes. However, in many cases the use of lean certificates (e.g. WTLS certificates), private/public key pairs or symmetric keys will be much more efficient. For instance, lean certificates and private/public key pairs save over-the-air bytes compared to an X.509 certificate, and symmetric key systems will optimize run-time performance compared to public key certificates.

There might be further steps to optimize performance. For instance, key lengths should be considered carefully. If key lengths are chosen too short then security will be endangered, but if key lengths are selected too long then performance will be harmed. There are several recommendations for key lengths available [1], however, most of these target applications of high security level. An individual assessment of each key might be able to tweak performance.

Shift Complexity from Vehicle to Server

Vehicles differ significantly from a PC in terms of security. Vehicles provide constrained resources due to cost restrictions, an attacker often has physical access to the vehicle, and the software in vehicles cannot easily be updated. Therefore, it is wise to shift complexity from the vehicle to a server. For instance, keys can be generated by a key management server and then injected in ECUs at the assembly line. This will ease the implementation and reduce complexity in the ECU, and at the same time shift more control to the server.

Another example continues the theft protection use case. Let us assume that ECUs in a car hold a car-specific symmetric key. In order to install a replacement ECU, the ECU could simply be connected to a key management server and upon successful authorization, the key management server looks up the car-specific key and provides it to the replacement ECU. If the authorization is executed outside of the ECU (e.g. between an approved workshop and the key management server), the complexity in the ECU is greatly reduced. Note that no end-to-end security between ECU and server is implemented and complexity is reduced at the cost of some security.

Reliability

Any security mechanism must not reduce reliability and availability of production and processes during the remaining life-cycle. For instance, if the key management server is involved during production of an ECU (e.g. injection of cryptographic keys), unavailability of the server, e.g. due to communication problems, must not hinder production of ECUs. If a replacement ECU needs to be installed in a car, a communication problem to the key management server should also not significantly disturb the process. The use of redundant systems can help here. The use of local computing and buffer systems can further increase reliability, e.g. provided by the use of locally deployed smart-cards or Hardware Security Modules (HSM). These locally deployed smart-cards or HSMs are able to securely store the server credentials locally and can offer the required key management server features locally, and synchronize with the server when a connection is available. Note that a proper system allows revocation of smart-cards and HSMs in case they were compromised, misused, or stolen.

Further aspects that need to be considered are privacy and access control. On one hand, the key management mechanism and server shall not leak any privacy-sensitive information; on the other hand access to the server shall be limited to a minimum need basis.

SECURE VEHICULAR KEY LIFECYCLE

In order to ensure the security of cryptographic keys throughout its life cycle it has to be ensured that the secret key remains confidential and authentic. If either of these goals is compromised the security of the system is endangered. In the following, we focus on secret keys that are stored and used in the vehicle. During its lifetime a cryptographic key undergoes several phases. This section highlights security concerns that are encountered during each phase.

Key Creation

Keys are either generated by a centralized authority, by an ECU in a vehicle, or they are negotiated between different ECUs. In either case the entropy of a key needs to be maximized. Generating a random key in an ECU is challenging since ECUs only provide limited resources and since they do not offer a proper source of randomness. If a proper source of randomness is available, using symmetric keys is fairly straightforward since a symmetric key is a binary string. However, generating public key pairs is more resource demanding. In particular, generating an RSA key pair is challenging. Therefore it is wise to shift the effort of generating keys to the key management server whenever possible. Also note that random number generators (RNG) are a main component to create cryptographic keys, and poor RNGs are a typical weakness in cryptographic systems.

Key Distribution

If two parties want to communicate securely, both parties need to be able to access the other party’s key. This key might be a shared symmetric key, or the other party’s public key. Furthermore, a party needs to be able to get hold of its private

and possibly public key or certificate. Key distribution ensures availability of the cryptographic key material.

In order to load keys initially into vehicles, it is common to create keys in the key management server, establish a secure channel between key management server and a production server at the assembly line, and then provide the generated keys via secure channel to the production server. The production environment is assumed to be secure and keys are loaded unencrypted into the ECUs. It would be desirable to protect the communication between production server and ECU to load the keys, but in order to do so the ECU would need cryptographic keys to establish a secure channel. The use of asymmetric keys allows such mechanisms. The ECU then create a public/private key pair, provides the public key to the production server, and the production server uses the ECU's public key to encrypt the generated key. Note that this approach is still vulnerable to a man-in-the-middle attack. If a key needs to be injected at a later time, e.g. during the replacement of a module, the same mechanism can be applied at a dealership.

Two parties that were loaded with keys now need to be able to obtain the other party's key. In case of a symmetric key management, the other party's key has to be loaded initially, typically during production time or during car service. It is reasonable to use a single car-specific symmetric key for all ECUs of a given vehicle. This mechanism might be appropriate for applications such as theft protection.

Applications that use public-key cryptography are able to implement key agreement or key exchange protocols such as Diffie-Hellman key agreement. In a key agreement protocol, the two parties provide an input and are able to derive a shared common key. In a key exchange protocol, one party generates a key and encrypts the key with the public key of the other party. Note that after the initial loading phase of keys in a secure environment, the parties are able to protect the channel for all required information exchange based on the initially loaded key material.

Key Storage

Keys must be stored properly to avoid easy extraction or modification. Ideally, secret and symmetric keys are protected from being extracted, and public keys are protected from being modified (otherwise an attacker would be able to generate a new private/public key pair and replace the public key in order to know the private key).

These security goals pose a challenge in a vehicle because an attacker often has physical access to a vehicle. ECUs usually utilize low-cost microcontrollers with no security features to avoid memory extraction or modification. In order to reduce attractiveness of such attacks it should be ensured that attacks do not scale, e.g. by using individual keys for each ECU. The use of secure hardware and hardware security modules (HSM) provides the required level of security but increases cost slightly.

Key Use

In order to use a key in an ECU's microcontroller, it needs to be loaded from storage into working memory to execute operations on it. This offers an attacker various opportunities to extract the key. The protection of keys during operation is a challenge on secure microcontrollers and smart-cards, and attacks become known regularly. In the vehicle environment, the challenge is even larger due to the low cost restrictions. Standard automotive controller are not protected against advanced attacks that read out side-channels, such as power consumption and time behavior, to derive information about the secret key. However, even basic attacks can often be successfully mounted to an ECU's microcontroller. For instance, the JTAG debug interface might be used to read out secret key data. Furthermore, the attacker might compromise the application that has access to the cryptographic key to use the cryptographic key, without actually knowing or extracting that key. If a standard microcontroller is used, it is essential to use a robust key management design without the use of any global keys. Protecting cryptographic keys during execution is at a high security level only possibly by using security microcontrollers. Automotive grade security controllers were specified in the EVITA project [4] and by HIS (Hersteller Initiative Software) known as the Secure Hardware Extension (SHE). SHE is a low-cost automotive grade security controller that provides secure key storage and a secure execution environment such that secure keys never leave the controller in unencrypted format. Note that SHE is available today on the market.

Key Backup

There are hardly any use-cases in which a key of a vehicle's ECU needs to be actively backed up during operation. However, it might be advantageous to store a copy of keys that were injected during production in the key management server. For instance, if a car key fob that was part of a theft protection mechanism was lost and a new key fob needs to be initialized, it might be easiest to load the secret key material from the key management to the new key fob in a secure environment, e.g. at a dealership.

Key Update

Key updates are useful to increase security. Regularly updating keys will avoid attacks that are based on the cryptanalysis of a large amount of encrypted messages or authentication information. A key update might also be used to restructure access to protected information. For instance, if a node needs to be removed from a system, all nodes except the selected one could receive an updated key.

Updating keys in a vehicle might be necessary to modify the status of an ECU. For instance, if an ECU is included in the theft protection mechanism of a vehicle, it might be loaded with keys that allow the authentication to other ECUs in the vehicle. If that ECU is switched from one car to another car, the key needs to be updated. A possible implementation is to delete all key material and then re-initialize the ECU with new keys in a secure environment as described above. Since key updates are usually only executed infrequently, if at all, it is important to choose key lengths in such a way that they are expected to protect the desired information for the life-span of the vehicle.

Deactivation & Deletion of Keys

While deactivation, revocation, and deletion of keys are common operations in a PC based security system, it seems unreasonable in a vehicle system. Availability and reliability are the major objectives of an electronic in-vehicle system. The deactivation of cryptographic keys implies non-availability of the protected system and therefore it contradicts the essential objective.

SECURE VEHICULAR KEY MANAGEMENT IMPLEMENTATIONS

There are available various implementations of secure key management. Unfortunately, most of them are proprietary and therefore not available on the market. In the following we are going to elaborate on available solutions and best practice approaches for secure key management implementation both in the embedded domain as well as for the supporting backend infrastructure.

In The Vehicle

In general, it is a good idea to strongly separate and isolate the storage of keys along with its cryptographic processing from the regular application operation environment and data storage. Today, various techniques to achieve this goal are present.

Secure Hardware Extension – SHE

As mentioned before, the German automotive vendors have created a functional specification of a rather minimalistic hardware co-processing unit called “Secure Hardware Extension” with the exact purpose of separating symmetric keys from the regular application of an ECU. SHE follows the approach of bringing a fixed and self-contained functionality in form of an on-chip hardware extension of a regular microcontroller. Its core functionality is the storage of symmetric keys and basic operations (en-/decryption, MAC generation/verification) with these keys. Furthermore, it offers protocols and processes to allow for a secure injection of keys into the ECU by providing end-to-end security from backend servers to silicon to efficiently protect the keys against spoofing, modification and re-injection.

Additional measures of SHE provide a basic access control to keys to, for example, hinder the key usage whenever a debugger is attached or to even use a secure boot scheme to ensure authenticity of the application accessing the keys.

To ease the integration into existing controllers, the specification of SHE is only on a functional level, leaving the concrete implementation and system integration to the semiconductor vendor. Furthermore, no requirements to the fabrication process are made and thus no hardware security measures to counteract tampering are included.

Today, the Secure Hardware Extension is available and/or announced in different automotive graded microcontrollers covering applications from powertrain, over body and chassis up to cluster instruments and infotainment.

EVITA HSMs

Similar to the industrial approach of SHE, the European research project EVITA developed three hardware security modules (i.e., EVITA light, medium, full) to be implemented as an on-chip peripheral on automotive microcontrollers [4]. Focusing on the on-board security of V2X applications and V2X communication, the three classes of an automotive HSM mainly differ in performance and supported cryptographic algorithms as shortly described in the following. Note that EVITA HSMs do not have a fixed functionality. They are intended to be reprogrammable to support more complex security functionality or the upgrade of supported algorithms and protocols in the future. Further, the definition of the EVITA HSMs prioritizes the functionality and architecture and does not make any statements on the necessity of anti-tampering measures.

EVITA Light

Except the additional internal clock and a somewhat more sophisticated key access and usage control, the smallest version of EVITA, named EVITA light, is almost similar to SHE and explicitly stated to be covered by SHE.

EVITA Medium

The medium sized HSM has similar hardware requirements and functionalities as the light module, that means, only symmetric cryptography (AES) is accelerated by hardware blocks. However, due to its programmability also the medium HSM support the execution of asymmetric cryptographic and hashing algorithms on its internal CPU. It further supports a more sophisticated bootstrap protection such as secure boot and authenticated boot and several monotonic counters to ensure freshness as necessary in many security protocols.

EVITA Full

The full HSM provides highest performance and state of the art acceleration for cryptographic hashing and for asymmetric cryptography (ECC) for external V2X communications with performance demands of up to 1000 signature verifications per second. See Table 1 for further details.

	Full	Medium	Light	SHE	TPM	SmC
Cryptographic algorithms						
ECC/RSA	■/■	■/■	□/□	□/□	□/■	⊞/⊞
AES/DES	■/⊞	■/⊞	■/□	■/□	□/□	⊞/⊞
WHIRLPOOL/SHA	■/■	■/■	□/□	□/□	□/■	⊞/⊞
Hardware acceleration						
ECC/RSA	■/□	□/□	□/□	□/□	□/□	□/□
AES/DES	■/□	■/□	■/□	■/□	□/□	□/□
WHIRLPOOL/SHA	■/□	□/□	□/□	□/□	□/□	□/□
Security features						
Secure/authenticated boot	■/■	■/■	⊞/⊞	■/□	□/■	□/□
Key AC per use/bootstrap	■/■	■/■	■/⊞	□/■	⊞/■	□/□
PRNG with TRNG seed	■	■	■	■	■	■
Monotonic counters 32/64 bit	■/■	■/■	□/□	□/□	■/□	□/□
Tick/UTC-synced clock	■/■	■/■	■/■	□/□	□/□	□/□
Internal processing						
Programmable/preset CPU	■/⊞	■/⊞	□/⊞	□/■	□/■	⊞/⊞
Internal V/NV (key) memory	■/■	■/■	⊞/⊞	■/■	■/□	■/□
Asynchronous/parallel IF	■/⊞	■/□	■/□	■/□	□/□	□/□

Annotation: ■ = available, □ = not available, ⊞ = partly or optionally available

Table 1: Comparison of EVITA hardware security modules [4], SHE [6], TPM [7], and Smart-Cards

Up to now, only prototypical implementations of the Full HSMs are available used in different research projects. The availability of HSMs inspired by the Medium HSM is announced for the next years by semiconductor manufacturers.

Secure Element

The name secure element refers to dedicated hardware modules placed in ECUs. Usually, derivatives of smart card controllers are used as secure element. Smart card controllers are dedicated microcontroller fulfilling highest security standards in terms of functionality and tamper-resistance. Secure elements come originally from the high-security area, where they are used to protect financial transaction, for example in cash cards or access control systems.

Software-based Sandboxing

If no dedicated hardware security mechanisms are available, also software based techniques for protecting key material can be used. One common approach is the sandboxing of different processes and applications, where the operating system or kernel isolates processes by, for example, starting virtual machines to execute the application code. Common hardware functionality, such as memory management units, can be used to assist the separation of processes by denying access to other processes' resources.

Hardware Assisted Virtualization

Also hardware assisted virtualization techniques as known form the classical computers and servers can be used to establish a secure zone to store keys and process cryptographic material. ARM's TrustZone technology is specifically designed to distinguish between a trusted context and the application context on a hardware basis. The trusted context may of course be used for implementation of security and key management related functionality. The hardware based hypervisor can ensure that only code being dedicated to the trusted zone is able to access resources of the TrustZone.

At the backend

Backend systems for embedded key management are rarely available on the market. For asymmetric cryptography there are solutions known as public key infrastructures (PKI) to support the management of keys. However, classical PKIs are designed in a user-centric way, meaning that all interaction is built around a person itself or devices associated to a person. In contrast, for embedded devices the PKI should follow a device-centric approach, where keys are associated to device identities and their specific life-cycles.

Furthermore, there are oftentimes fundamental differences in operation and hosting of device key management systems compared to classical PKIs. In general, they are directly related to production processes and life-cycle management processes of devices and therefore have to be run as a service with high-availability and low latency to ensure timely production of devices. Both properties directly lead to the fact that a high automation of processes is a key feature, again differing from classical PKIs where at least the identity registration involves manual steps.

High availability of the management system and cost sensitivity of the products are regularly a prohibitive factor for running a PKI in a classical trust center being specialized on providing highest security in terms of physical shielding and processing of requests. In contrast, today's international production processes being organized in a multi-tier fashion are longing for distributed and fail-safe server infrastructure to follow the streams of merchandise over the world.

As a matter of fact, key management server solutions are still evolving by following the trend of increasing connectivity of embedded devices (so called cyber physical systems). Usually highly customized and proprietary systems are used today for historical reason. Typically one can also observer that those proprietary solutions are very limited in scalability and their operation causes high cost, which is limiting to the competitiveness of the embedded device manufacturer.

SUMMARY

Securing the key management for cryptographic keys used for vehicle electronic systems is quite more difficult than securing the whole lifecycle of a key used for standard computers. Automotive key management systems have to ensure the availability and also a controlled usage of such cryptographic keys has to fulfill different functional requirements (e.g., sporadic low-bandwidth connections) and different security requirements (e.g., physical insider attacks).

In this article, we analyzed the requirements and gave best practice approaches for secure key management solutions in the automotive context. We showed potential security vulnerabilities, which and introduced open solutions and best practice approaches for secure key management implementations in the embedded in-vehicle domain with the help of hardware security modules as well as for the supporting backend infrastructure.

REFERENCES

1. BlueKrypt, “Cryptographic Key Length Recommendation”, available at http://www.keylength.com.
2. Bono, S. , Green, M. , Stubblefield, A., Rubin, A., Juels, A., Szydlo. M., "Security Analysis of a Cryptographically-Enabled RFID Device", USENIX Security Symposium, August 2005.
3. The eSTREAM Project, available at http://www.ecrypt.eu.org/stream/project.html
4. EVITA Project, “Deliverable 3.2: Secure On-board Architecture Specification”, 2010.
5. Gendrullis, T., Wolf, M., ”Design, Implementation, and Evaluation of a Vehicular Hardware Security Module”, In 14th International Conference on Information Security and Cryptology (ICISC 2011), Seoul, Korea, December, 2011.
6. Hersteller-Initiative Software (HIS), “SHE – Secure Hardware Extension Version 1.1”, 2009.
7. Trusted Computing Group (TCG), “TPM Specification 1.2 Revision 116”, 2011.
8. Weimerskirch, A., “Secure Software Flashing”, In Society of Automotive Engineers (SAE), International Journal of Passenger Cars – Electronic and Electrical Systems, October 2009, 2:83-86.
9. Weimerskirch, A., Paar, C., and Wolf, M. , “Cryptographic Component Identification: Enabler for Secure Inter-vehicular Networks”, 62nd IEEE Vehicular Technology Conference, September 25-28, 2005, Dallas, TX, USA.

CONTACT INFORMATION

Marko Wolf
ESCRYPT GmbH
Leopoldstraße 244, 80807 Munich, Germany
marko.wolf@escrypt.com

André Weimerskirch
ESCRYPT Inc.
315 E Eisenhower Parkway, Suite 214,
Ann Arbor, MI 48108, USA
andre.weimerskirch@escrypt.com