Approaches for Secure and Efficient In-Vehicle Key Management*

Takeshi SUGASHIMA Dennis Kengo OKA Camille VUILLAUME

Modern vehicles utilize various functionalities that require security solutions such as secure in-vehicle communication and ECU authentication. Cryptographic keys are the basis for such security solutions. We propose two approaches for secure and efficient in-vehicle key management. In both approaches, an ECU acting as a Key Master in the vehicle is required. The first approach is based on SHE. The Key Master generates and distributes new keys to all ECU based on the SHE key update protocol. The second approach performs key establishment based on key derivation. The Key Master sends a trigger in form of a counter and all ECUs derive new keys based on the received counter value and pre-shared keys. It is thus possible to handle in-vehicle key management without the need for an OEM backend to manage all keys. This reduces cost and complexity of the solution. It avoids using the same keys in a vehicle for long periods of time since keys can be updated regularly within the vehicle without any external interaction. We have implemented the approaches on a test bench and performed an evaluation.

Key words :

security, key management, key distribution

Introduction

Modern vehicle systems have transitioned from being isolated, stand-alone systems to interconnected systems with external interfaces. As a consequence, vehicles are now the target of cyber security attacks. There already exist several works presenting security vulnerabilities on vehicle systems, threat and risk analysis methods and relevant security countermeasures such as SecOC in AUTOSAR[1]. The foundation for security solutions is often strong cryptography and as such the management of cryptographic keys is extremely important to ensure a high level of security.

For example, each ECU in a vehicle may require one or several cryptographic keys. The challenge is how to load such keys in a secure and efficient manner. One existing approach is that keys in the ECUs are managed by an OEM backend and loaded into the ECUs during, for example, production. However, in some cases, it is better to manage the keys for the in-vehicle network in each vehicle itself rather than on an OEM backend. Therefore, we are proposing approaches for in-vehicle key management. The focus is on key establishment of symmetric keys among ECUs within a vehicle.

The proposed approaches require a Key Master in

＊ 2016 年 8 月 22 日　原稿受理

the vehicle which is responsible for the in-vehicle key management. Each ECU in the vehicle can then load new keys that are either generated and distributed by the Key Master or derived on each ECU based on a trigger from the Key Master. These approaches have the following advantages:

- Improve security strength, by separating the necessary key management to be handled on an OEM backend and on the in-vehicle network, and making it easy to have different, short-lived session keys within each vehicle
- Enable faster assembly as it is not necessary to generate and load all keys onto the ECUs during production and thus reducing the time needed to be connected to the OEM backend
- Enable replacement of ECUs at dealer locations without the need to be connected to the OEM backend to download keys
- Reduce cost and complexity as it is not necessary to manage all keys in one central location on an OEM backend.

The main contributions of this paper are:

- We propose approaches for secure and efficient in-vehicle key management, which handle key management of ECUs in each vehicle itself rather than on an OEM backend.
- We have designed simple protocols based on key distribution and key derivation approaches, respectively, where a Key Master in a vehicle generates and distributes new keys to ECUs or triggers each ECU to derive new keys.
- We have implemented the protocols on a test bench and performed an evaluation of the results in terms of performance and security.

Related work

HIS, a car consortium consisting of several major German auto manufacturers, has developed an implementation specification for secure hardware called SHE (secure hardware extension)[2)3)]. SHE provides various security functionalities such as MAC generation and verification based on a hardware AES engine and loading of symmetric keys into secure key storage[4)]. To be able to support security use cases where SHE functionality is not sufficient, Bosch has developed a Bosch HSM (hardware security module) specification[5)]. HSM provides further support for security functionalities as it has in addition to a hardware AES engine, a dedicated secure CPU and secure memory allowing it to be programmable to support a vast range of use cases. The AUTOSAR specification 4.2.1 includes CSM (crypto services manager) and describes how cryptographic keys can be used to support use cases such as to protect the in-vehicle communication. For example, the SecOC module provides functionality to enable MAC generation and verification for in-vehicle CAN communication[1)]. There are some APIs for key management in AUTOSAR CSM such as key derivation and key generation[6)]. However, there is at this time no clearly defined APIs for, e.g., key distribution, session key activation and loading keys into key storage. Thus, AUTOSAR does not explicitly specify how keys should be managed within the vehicle.

Furthermore, there exist an international standard which provides mechanisms for key establishment and key management[7)]. These mechanisms are based on symmetric key and asymmetric key techniques, respectively.

Problem statement

There are many use cases which require cryptographic keys in ECUs to be used. A few representative use cases are listed in **Table 1**.

From **Table 1**, it is clear that for some use cases it is more suitable and even required to use in-vehicle key management. In particular, in-vehicle message authentication requires extensive in-vehicle key management for the following reasons. First, it is much easier to manage just the relevant keys within each vehicle rather than managing all keys in an OEM backend. Second, in the SecOC, a freshness value is included in the authentication of messages [1], and to prevent replay attacks, the same pair of freshness value and key can never be used twice. To ensure that they are never used twice, there are two possibilities:

- Storing the current freshness value in non-volatile memory (NVM) when turning the ignition off so that it is possible to increment and continue using the freshness value with the same key.
- Distributing new session keys when turning the ignition on, in which case the freshness value can be reset.

The former approach might be difficult in practice as writing the freshness value in NVM while the ECU is in shutdown sequence may cause the write operation to fail, and might lead to situations where the freshness values are out of sync. The latter approach is more robust, but requires key management.

Using an OEM backend to manage all keys required in ECUs is a costly and complex endeavor. Moreover, since all the necessary keys in the ECUs typically need to be loaded into the ECUs during production, it is a time-consuming process. Furthermore, in general, vehicles are in the field a very long time. If the keys cannot be updated frequently, there is a risk that the same keys are used for long periods of time and thus susceptible to various attacks. For example, if a key is compromised, the key must be renewed. If an OEM backend is used, the vehicle is required to establish a wireless connection to the OEM backend which may not be possible for all vehicles or is required to be taken to a workshop. Even if the vehicle is at a workshop, the dealer technician may not be able access the OEM backend due to lack of network connectivity. In addition, if all ECU keys are managed by an OEM backend, corresponding keys are required to be sent to each ECU. In this case, it will take a long time to distribute all keys to all relevant ECUs. In-vehicle key management would solve these problems.

Threats to the in-vehicle key management approaches we consider in this paper are as follows:

- Attacker distributes own keys to ECUs
- Attacker illegitimately obtains legitimate keys
- Attacker forces ECUs to establish/share keys which were used in the past (replay attack)

Table 1 Representative use cases

Use Case	Suitable Key Management	
	OEM Backend	In-Vehicle
In-vehicle message authentication *1		✔
ECU authentication		✔
Authenticated ECU key updates	✔	✔
UDS "security access" *2	✔	
ECU reprogramming *3	✔	

*1; In-vehicle message is verification MACs for Secure On-board Communication (SecOC) in AUTOSAR[1]. In this case, cryptographic keys are used only in a vehicle.

*2; Using keys to generate and verify responses in the Challenge and Response protocol (service $27) [8].

*3; Update program; RSA for signature and AES for encryption are used [9].

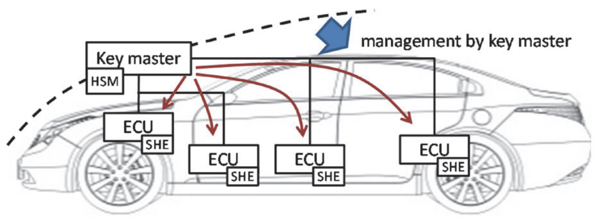


Fig. 1 In-vehicle key managements

Key management proposal

In this section, we propose two key establishment approaches: 1) key distribution based on SHE [2], and 2) key establishment based on key derivation. For both these approaches, a Key Master is required in the vehicle, e.g., a Central Gateway ECU could play this role. Our suggested approaches would be especially suitable for in-vehicle message authentication.

Approach 1 - Key distribution based on SHE

ECUs enabled with SHE functionality allow for ECU key loading – refer to [4] for details about the SHE load key command. The typical use case using the key update protocol in SHE would require an external entity to generate keys and the messages relevant to key loading (M1, M2, M3) to be able to load the keys onto the SHE-enabled ECUs. The M1, M2 and M3 contain among other the target UID (unique identifier) in M1, the new key encrypted with another key in M2, and a MAC calculated over this data in M3. The receiving ECU replies with the messages M4 and M5 which provide digital evidence that the new key has been loaded properly. M4 contains information which key was updated, and M5 is a MAC calculated over M4. A SHE-enabled ECU cannot be used to securely load keys onto other SHE-enabled ECUs because SHE is implemented in hardware and cannot be programmed; therefore, there is a need for a (non-SHE) Key Master in the vehicle.
The SHE-based approach is depicted in **Fig. 2**. In our proposal, the Key Master generates new keys and sends the corresponding M1, M2, and M3 to each ECU. M2 and M3 are created using a pre-shared MASTER_ECU_KEY. Using a wildcard value for UID, all ECUs' keys can be updated at the same time based on the same set of M1, M2 and M3, i.e., broadcast key distribution. Alternatively, keys can be loaded into each ECU individually by specifying the target UID. M1, M2 and M3 are 16 bytes, 32 bytes and 16 bytes respectively. The Key Master needs to generate keys and corresponding M1, M2, M3 for each Key_1, Key_2 etc. that must be loaded into an ECU. Once the keys have been loaded into the memory slots in SHE, the respective keys can be used at will by SHE. MASTER_ECU_KEY also can be updated by same sequence.
Typically, M4 and M5 are generated by the ECU in the SHE key update protocol. M4 and M5 are 32 and 16 bytes, respectively and only used to provide verification that the new key was properly loaded; if this verification is not required, M4 and M5 can be discarded. However if verification is necessary, since M4 and M5 are large in size and we are concerned about performance, we suggest that the ECU replies to the Key Master with an 8-byte Res value instead. The Res value provides the same verification that the new key was properly loaded, albeit with a shorter message length so that it would fit into one CAN frame. The Res value can be calculated by using the generate MAC function on SHE using the newly created key over the UID of the ECU. The value can then be truncated to 8 bytes and returned to Key Master. Including the unique UID in the calculation of Res allows the Key Master to identify which ECUs have properly loaded the keys.
Additionally, it would be possible for the ECU to trigger a key update by initially sending a specific request to the Key Master.

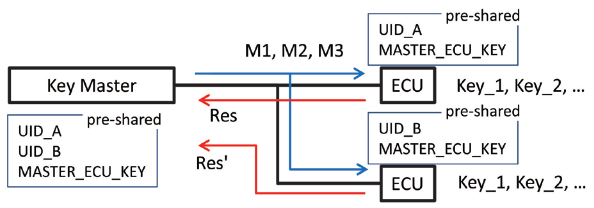


Fig. 2 Broadcast key distribution based on SHE

Approach 2- Key establishment based on key derivation

The second approach is based on key derivation and illustrated in **Fig. 3**. The Key Master and all ECUs have a pre-shared 4-byte counter cnt and a pre-shared key that will be used to protect the counter value (e.g., Key_0). Additionally, all ECUs have pre-shared keys that will be used as session keys (Key_1, Key_2 etc.) stored in secure key storage in SHE. The Key Master increments the counter and adds a 4-byte MAC generated using the pre-shared key Key_0 over the counter value. The 8-byte message is then sent to all ECUs as a trigger to update the keys. Each ECU first verifies the MAC of the received message and that the received counter value is higher than the local counter value, and if so stores it as its local counter value. Each ECU then derives session keys (SKey_1, SKey_2 etc.) using KDF (key derivation function) and the locally stored 4-byte counter and respective pre-shared key (Key_1, Key_2 etc.). In this paper, we assume SHE-enabled ECUs, and therefore we use CMAC as KDF. Each ECU stores the derived session keys in RAM. For SHE to be able to use a specific key, that key needs to be loaded into SHE as a RAM_KEY. There is only one RAM_KEY slot in SHE; therefore, every time a specific session key needs to be used by SHE, it needs to be first loaded as a RAM_KEY.

Since this approach is performance-focused, Res values are not calculated. However, if the Key Master needs to confirm that the session keys have been loaded properly, the same method as in Approach 1 could be used.

Requirements for the key management approaches are shown in **Table 2** and **Table 3**.

In our prototyping, we tried to use AUTOSAR CSM; however, the APIs relevant to key management are not defined in detail. Therefore, in **Table 4**, we propose some details for APIs that are suitable for our suggested approaches.

Table 2 Requirements for key management approach 1

Key Master		HSM(Secure storage, Secure execution environment)
		TRNG for key generation Be able to generate M1, M2, and M3 ・ Key encryption: AES-CBC ・ Key message authentication: AES-CMAC ・ KDF based on Miyaguchi-preneel compression
	Parameters	Pre-shared MASTER_ECU_KEY with ECUs UIDs of all ECUs Counter values for respective key to be stored in ECUs.
ECUs		Compliant with SHE
	parameters	Pre-shared MASTER_ECU_KEY with Key Master

Table 3 Requirements for key management approach 2

Key Master		HSM(Secure storage, Secure execution environment)
	Parameters	Pre-shared key (Key_0) with ECUs (to protect cnt) Pre-shared counter value for key derivation with ECUs
ECUs		Compliant with SHE
	parameters	Pre-shared keys with other ECUs Pre-shared key (Key_0) with Key Master (to verify MAC for counter) Pre-shared counter value for key derivation with Key Master

Table 4 APIs proposal

AUTOSAR CSM	Problem	Proposal
Csm_SymKeyGenerate	This API may be used but details are not be defined.	Key generation from TRNG (for Approach 1)
Csm_SymKeyWrapSymStart Csm_SymKeyWrapSymUpdate Csm_SymKeyWrapSymFinish	This API may be used but details are not be defined.	M1-M3 generation (for Approach 1)
Csm_SymKeyExtractStart Csm_SymKeyExtractUpdate Csm_SymKeyExtractFinish	This API may be used but details are not be defined.	Load key (e.g., loading the keys for Approach 1)
(Missing)	There is no key activation API.	Key activation (e.g., which key to activate for Approach 2)
Csm_KeyDeriveSymKey	This API may be used but details are not be defined.	Key derivation (e.g., generating CMAC from counter for Approach 2)

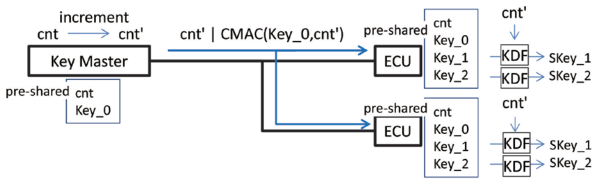


Fig. 3 Key establishment based on key derivation

Evaluation and analysis

We have implemented the two approaches presented in previous section and performed an evaluation of performance and analysis of the implementation. The setup is described as follows:

Table 5 implementation environmentl

Key Master	Host CPU:160MHz, HSM:80MHz
ECU A	Compliant w/ SHE CPU:80MHz, SHE:40MHz
ECU B	
Compiler	GHS, option –osize
Communication	High speed CAN (500kbps)
Task period	5ms (i.e.. one task can star executing every 5 ms)
CAN transmitting period	1ms (i.e., one packet can be transmitted every 1ms)

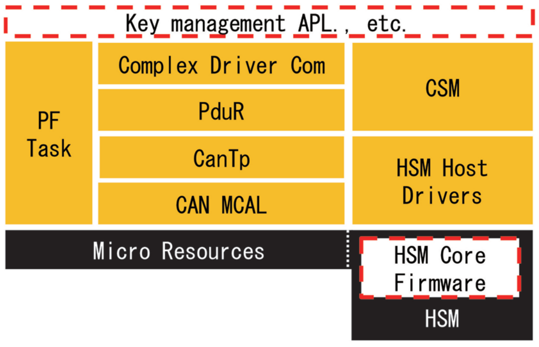


Fig. 4 Implementation of Key master

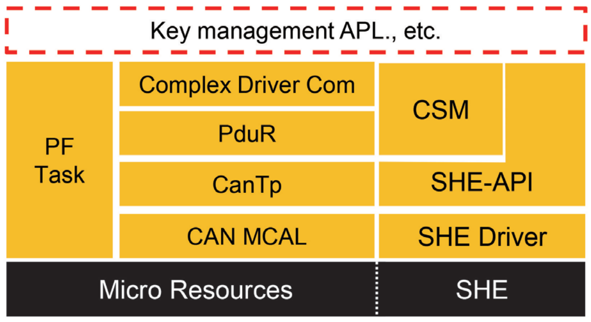


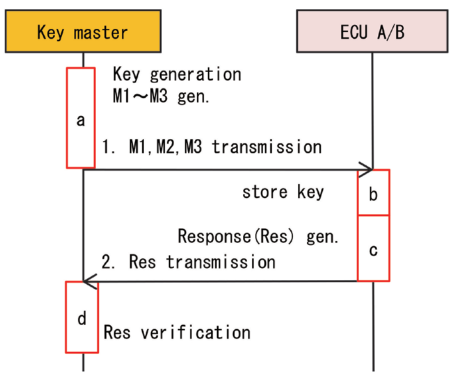
Fig. 5 Implementation of ECUs

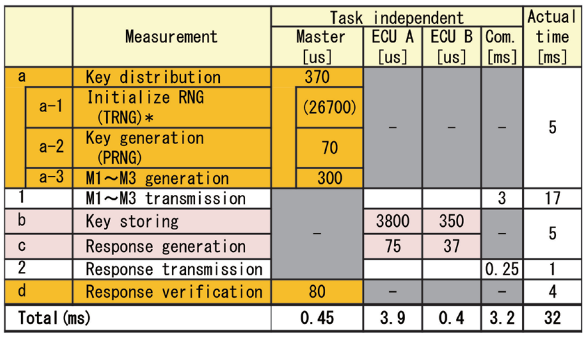
We used two microcontrollers for MCU A and B. Each microcontroller is compliant with SHE, and these are designed by different semiconductor vendors. The task period and CAN transmitting period are based on realistic performance. The software sizes required to run the in-vehicle key management are shown in **Table 6**.

Table 6 Software sizes

	Key Master		ECU
ROM size(kByte)	Host	~3.0	~2.5
	HSM	~1.2	
RAM size(kByte)	~2.0*		~0.6*

*RAM size depends on message buffer size.





	Measurement	Task independent				Actual time [ms]
		Master [us]	ECU A [us]	ECU B [us]	Com. [ms]	
a	Key distribution	370	–	–	–	5
a-1	Initialize RNG (TRNG)*	(26700)				
a-2	Key generation (PRNG)	70				
a-3	M1～M3 generation	300				
1	M1～M3 transmission	–			3	17
b	Key storing		3800	350	–	5
c	Response generation		75	37		
2	Response transmission				0. 25	1
d	Response verification	80	–		–	4
Total (ms)		0. 45	3. 9	0. 4	3. 2	32

Fig. 6 Approach 1 flow and time measurements

Approach 1

The results for Approach 1 are first discussed. The Key Master distributes one key to one or several ECUs. Several ECUs could perform the same steps in parallel to load the same key. To load multiple keys, the protocol is repeated until all keys have been distributed. The flow and time measurements are shown in **Fig. 6**. The two "task-independent" columns show the measured processing and communication time for the specific independent task operations on the microcontroller level. The right column shows the actual measured time for the implementation of the function including any overhead. As the independent tasks for processing (3.3ms) and communication (3.2ms) take about only 6.5ms in total, the total key distribution time could be reduced by optimizing the implementation (current implementation takes 32 ms).

The key generation based on true random number generation (TRNG) and the communication times are the most time-consuming parts in this protocol.

TRNG time depends on the performance of the microcontroller. In this prototyping, the initialization of TRNG takes long time. It shall be done before the key distribution process. To generate and send the M1, M2, M3 messages over CAN takes around 27ms (steps a, 1, b, c in **Fig. 6**). If keys are distributed by broadcast, the Key Master sends M1, M2, and M3 only once per key, and one Res response message is sent per ECU per key, which takes around 5 ms (steps 2, d). As a result:

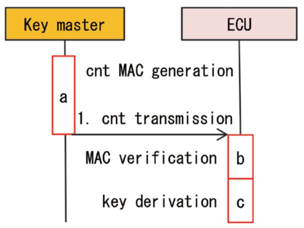
● Loading one key takes: 27 ms + (5 ms * <number of ECUs>).

● Total time: <number of keys> * (27 ms + (5 ms * <number of ECUs>))

It would be possible to perform some steps in parallel. For example, while Key Master is sending M1-M3 to the first ECU, the Key Master can generate keys and M1-M3 for the following ECUs.

Analysis about the threats that we defined

To protect M2 and M3, a pre-shared MASTER_ECU_KEY needs to be stored on the Key Master and the ECUs. This initial key needs to be loaded during, e.g., production. For an attacker to be able to load own keys, the attacker needs to know the MASTER_ECU_KEY to be able to create legitimate M2 and M3. As long as the MASTER_ECU_KEY is secure, an attacker cannot load his own keys. The MASTER_ECU_KEY is stored in the secure storage of the Key Master and in a SHE key slot in each ECU, and the new keys are loaded into SHE secure key storage on each ECU. As a result, it is assumed that an attacker cannot modify or extract the keys. If an attacker replays M1, M2 and M3, the included counter value for the corresponding key will be incorrect and the command to load the key will fail, thus preventing replay attacks.



	Measurement	Task-independent				Actual time
		Master [us]	ECU A [us]	MCU B [us]	Com. [ms]	
a	cnt MAC generation	80	-	-	-	3
1	cnt transmit	-			0.25	1
b	cnt MAC verification		40	64	-	1
c	key derivation		40	80	-	
Total(ms)		**0.05**	**0.1**	**0.15**	**0.25**	**5**

Fig. 7　Approach 2 flow and time measurements

Approach 2

The results for Approach 2 are described below. The Key Master initiates the key update by incrementing the counter and calculating a MAC and sending the counter and MAC to the ECUs. The same steps can be performed in parallel on all ECUs to derive the keys. The flow and time measurements are shown in **Fig. 7**. First, the transmission of the 8-byte counter and MAC value from the Key Master to the ECUs takes around 4ms (steps a, 1 in **Fig. 7**). The ECU then performs the necessary calculations to verify the MAC, synchronizes the counter value and then uses KDF to derive the session keys. These steps take around 1ms (steps b, c). The key derivation is repeated for all the keys that need to be generated. The individual task to generate one session key on the microcontroller level takes around 40us (step 3 processing time), thus the implementation time of 1ms would not significantly change even if several keys are generated at this step. As a result:

● Total time to derive one or several keys: 5 ms

Every time a session key needs to be used, it needs to be loaded into SHE first. The load time for a key is about 5us. Since there is only one RAM key slot, the corresponding key needs to be loaded before it can be used. For example, if the keys are used for secure in-vehicle communication, the corresponding key needs to be loaded into the RAM key slot before it can be used to verify the MAC of the message. Since we consider the time to load a new session key into the RAM key slot to be negligible (5 us), it is possible to easily switch between multiple keys.

Analysis about the threats that we defined

For an attacker to be able to load own keys in the ECUs (i.e., be able to load a key in an ECU that the attacker knows), the attacker needs to know the keys Key_1, Key_2 etc. stored in each ECU. Since the keys Key_1, Key_2 etc. are stored in the SHE secure key storage, we consider that it not possible to extract such keys. However, because session keys are stored in RAM, these keys are susceptible to attacks (e.g., disclosure or modification). Simple replay attacks are prevented as the ECUs verify that the received counter value is higher than the locally stored counter value. In addition, since the counter sent by the Key Master is authenticated, it is not possible for an attacker to send fake counters that can trigger key updates in ECUs.

Summary

If security is the most important factor for the use case in question and it is acceptable to allow a slight delay initially to load all keys securely, Approach 1 is suggested. If many keys are used and performance is extremely time-critical and it is not possible to wait to load all keys individually using the corresponding M1, M2 and M3, Approach 2 would be more suitable. There exist some open issues that need to be resolved. During testing on some microcontrollers, for Approach 1, e.g., the time to store a key into ECU B was sometimes abnormally long (~300 ms), probably due to flash erase operations. Flash memory needs to erase operation before writing new data. The microcontroller used for ECU B has large flash block. If the erased area is left, storing new key into ECU B takes about 300us. When the flash block is filled, erase operation has to be run and take over 300ms. The microcontroller used for ECU A has smaller flash block than the microcontroller for ECU B, the erase operation is run on all key load. The key load operation takes over 3ms every time, but this operation time is constant. For this problem, our suggestion is as follows.

– Providing two slots for one key
– Key in first slot is used
– New key is generated, distributed and store to the second slot during the driving session
– Valid key slot is switched during ignition on

For Approach 2, the counters on the Key Master and ECUs may get out of sync, and may require to be reset. There needs to be a secure protocol to handle out-of-sync counters.

A suggestion for synchronized counter approach is as follows. The 4-byte counter consists of two parts: 2 most significant bytes (MSB) counter incremented for every ignition on, and 2 least significant bytes (LSB) counter incremented for each key update. When turning the ignition on, the Key Master increments and distributes the MSB counter with a 6 bytes MAC calculated over the 4-byte counter using the pre-shared key stored in SHE. The LSB counter on the Key Master is reset to zero. The ECUs verify the MAC and compare the received counter to its own MSB counter. If the received counter is greater than stored counter, it replaces the locally stored MSB counter with the received counter value and resets the local LSB counter value to zero. Since this counter is stored early during the driving session, it is assumed the write to NVM is successful. The key update while driving is similar but instead of the Key Master distributing the MSB counter, the Key Master increments and sends the LSB counter to the ECUs. If the received counter is larger than locally stored LSB counter, it replaces the locally stored LSB counter with the received counter value.

Session keys are then derived using the locally stored 4-byte counter. If ignition on happens 10 times a day, a 2 bytes MSB counter will be effective for about 18 years. This counter can be reset when pre-shared key is renewed. If support for a longer time period is necessary, pre-shared key should be renewed before counter overflow, or a longer counter could be used. Even if the ECU would be in shutdown sequence, there is no need to write the latest LSB counter value to NVM as it will always be reset to zero at ignition on. This approach would prevent out-of-sync counters as long as the MSB counter has been previously written properly to NVM.

Conclusions

In this paper, we propose two approaches for secure and efficient in-vehicle key management, and additionally specify some details for suitable APIs in AUTOSAR CSM to be used with the approaches. In both approaches, an ECU acting as a Key Master in the vehicle is required. The first approach is based on SHE. The Key Master generates and distributes new keys to all ECU based on the SHE update protocol. To reduce the required time, it is possible to broadcast keys to several ECUs at the same time. The second approach performs key establishment based on key derivation. The Key Master sends a trigger in form of an authenticated counter and all ECUs derive new keys based on the received counter value and pre-shared keys. We have implemented the approaches on a test bench and performed an evaluation. Approach 1 takes roughly 32 ms per key to load into SHE on an ECU (several ECUs can be loaded in parallel). Once all keys have been loaded, they are ready to be used. Approach 2 takes roughly 5 ms to generate all new keys on an ECU (can be done in parallel). Every time a key needs to be used, it needs to be loaded into SHE first which takes about 5us. For Approach 2, because keys are stored in RAM before they are loaded into SHE secure key storage, the keys are susceptible to attacks (modification, disclosure). If performance for the use case in question allows a slight delay initially to load all keys securely, Approach 1 is suggested. If many keys are used and performance is extremely time-critical and it is not possible to wait until all keys have been loaded individually using the corresponding M1, M2 and M3, Approach 2 would be more suitable. For example, approach 1 can be used to share the key-derivation key, and approach 2 can be used to derive session keys.

The suggested approaches use a Key Master in the vehicle to handle in-vehicle key management without the need for an OEM backend to manage all keys. This reduces cost and complexity of the solution. It avoids using the same keys in a vehicle for long periods of time since keys can be updated regularly within the vehicle without any external interaction. There exist many use cases where cryptographic keys are used in vehicles such as secure in-vehicle communication and ECU authentication. The suggested proposals for secure and efficient in-vehicle key management can enable and improve security for such use cases.

References

1) AUTOSAR Release 4.2.1, “Specification of Module Secure Onboard Communication, Communication Stack,” 2014.
2) Hersteller Initiative Software (HIS), “SHE- Secure Hardware Extension, Function Specification, Version 1.1,” 2009.
3) Fujitsu Seminconductor, “SHE - Secure Hardware Extension,”in Workshop on Cryptography and Embedded Security at embedded world conference, Nuremberg, 2012.
4) Freescale Semiconductor, “Using the Cryptographic Service Engine (CSE) - An introduction to the CSE module,” 2011.
5) J. G. M. I. J. S. R. S. a. M. E. Oliver Bubeck, “A Hardware Security Module for Engine Control Units,” in escar - Embedded Security in Cars, Dresden, 2011.
6) AUTOSAR Release 4.2.1, “Specification of Crypto Service Manager, System Services,” 2014.
7) ISO/IEC, “ISO/IEC 11770 - Information technology - Security techniques - Key management,” 2010.
8) ISO/IEC, “ISO/IEC 14229-1 - Road vehicles -- Unified diagnostic services (UDS) -- Part 1: Specification and

requirements," 2013.
9) Hersteller Initiative Software (HIS), "HIS Security Module Specification Version 1.1," 2006.

著者

菅島 健司
すがしま たけし

電子基盤システム開発部
車載セキュリティの技術開発

Dennis Kengo Oka
岡 デニス 健五

イータス株式会社　エンベデッドセキュリティ Ph.D.　工学博士
車載セキュリティのコンサルティングに従事

Camille Vuillaume
カミーユ ヴィオム

イータス株式会社　エンベデッドセキュリティ Ph.D.　博士 (システム情報科学)
車載セキュリティのコンサルティングに従事