electronics

MDPI

Article

Research on In-Vehicle Key Management System under Upcoming Vehicle Network Architecture

Zhihong Wu [1,2], Jianning Zhao [1], Yuan Zhu [1,2,*], Ke Lu [2] and Fenglue Shi [3]

1 School of Automotive Studies, Tongji University, Shanghai 201804, China; zhihong.wu@tongji.edu.cn (Z.W.); 1510810@tongji.edu.cn (J.Z.)
2 Sino-German School for Postgraduate Studies, Tongji University, Shanghai 201804, China; luke@tongji.edu.cn
3 System Department, G-Pulse Technology Co., Ltd. Building 5, 787 Kangqiao Road, Shanghai 201315, China; fenglue_shi@foxmail.com
* Correspondence: yuan.zhu@tongji.edu.cn

Received: 16 August 2019; Accepted: 9 September 2019; Published: 12 September 2019

Abstract: The intelligentization and connectedness of vehicles make vehicle cybersecurity an important research topic. In-vehicle key management is a critical function in vehicle cybersecurity countermeasures. After describing previous research on vehicle key management and the development trend of vehicle network architecture, a key management scheme for in-vehicle multi-layer electronic control units (ECUs) is proposed. The scheme is based on authenticated key exchange protocol 2 (AKEP2) and on-the-air (OTA) technology. Then, the key storage and trusted key usage based on secure hardware are analyzed and studied. Moreover, the AES Counter with CBC-MAC (AES-CCM) algorithm, which uses fewer keys, is introduced to in-vehicle secure communication. The simulation analysis for the proposed OTA-based key update protocol verifies the protocol's security. The validity of the hardware-based trusted key usage environment and the feasibility of the AES-CCM algorithm for the CAN FD bus are proven with corresponding experiments.

Keywords: in-vehicle key management; OTA-based key update; trusted key usage; AES-CCM

1. Introduction

Electrification, intelligentization, and connectedness are developing trends of modern vehicles. Intelligentization can lead to changes in the vehicle's electrical/electronic architecture, such as the concepts of a central gateway and domain controller. Electrification and connectedness, especially connectedness, will result in more attacks in the surfaces of vehicles, thus increasing the risk of the vehicle being attacked. The object of these attacks is either the operation and control functions of the vehicle [1,2], representing the most damaging attack, or information about privacy and intellectual property in the vehicle [3,4]. Cryptography is an essential method to ensure cybersecurity, and the key is the essence of cryptography [5]. Key management is extremely important for a vehicle [6]. For the complex and developing vehicle electrical system, which consists of tens of electronic control units (ECUs), studies on in-vehicle key management systems (KMS) are necessary.

The KMS has many functions, such as key generation, key distribution, key update, key storage, and key destruction. In this paper, we mainly study the functions of storage, usage, update, and destruction of vehicle keys. To the best of our knowledge, there is no complete and comprehensive research on in-vehicle KMS. The previous research mainly focused on key generation and key distribution for the in-vehicle CAN or CAN FD bus.

1.1. Related Research

In 2012, Lin et al. proposed a CAN frame authentication protocol based on an ID table and frame counter [7]. The protocol uses a pair-wise symmetric key (PWSK) to generate a message authentication code (MAC) to realize message authentication and message integrity. PWSK is distributed via a pre-shared method; however, key update, key storage, and key destruction were not discussed. Another CAN frame authentication scheme called a timed efficient stream loss-tolerant authentication (TESLA)-like protocol was suggested in 2013 [8]. The main principle of this scheme is to use the TESLA protocol for timed loss-tolerant authentication. However, other researchers showed that the TESLA-like protocol cannot provide real-time processing of a CAN frame [9]. In 2015, Mundhenk et al. presented a lightweight authentication and authority protocol for in-vehicle networks [10]. The protocol has two steps, firstly using the elliptic curve cryptography (ECC) algorithm to authenticate identification, and then using the Kerberos algorithm to distribute keys or update keys for data stream encryption. A more complete CAN bus security protocol was put forward by Woo et al. in 2015. The protocol uses authenticated key exchange protocol 2 (AKEP2) and a key distribution center (KDC) to implement key generation, key distribution, and key update [11]. Woo et al.'s protocol has a good real-time performance. Fassak et al. proposed a session key establishment protocol between two ECUs based on ECC, and the security of the protocol was validated by simulation software [12]. All of the above studies focused on CAN bus and mainly used an MAC to protect the CAN frame.

However, the CAN bus has small bandwidth and every frame has a payload of at most 8 bytes. Utilizing a secure CAN protocol, such as by adding a MAC in communication, will reduce the data payload or heavily increase the CAN bus load rate. Based on their former work, Woo et al. put forward a practical automotive security level (ASL) concept on a CAN FD bus according to different security requirement levels in 2016 [13]. Their experiments proved that the scheme has little key derivation delay and fast communication response on the CAN FD bus, and it can be used in the practical scene.

1.2. The Work in This Paper

Existing researches mainly focused on lightweight key distribution, key update, and the design of a secure frame on the CAN and CAN FD bus; however, there was little discussion about remote key update. However, the remote key update is a convenient and low-cost function in the KMS of future vehicles. Furthermore, only Woo et al. provided a brief description on key storage in 2016 [13]. We believe that the security of the key usage environment is as important as key storage. Research topics about in-vehicle KMS from other researchers and this paper are shown in Table 1.

Table 1. Main research topics from different researchers.

Researchers	Key Generation	Key Distribution	Key Update (Local Update, Remote Update)	Key Storage	Key Destruction and Certificate Revocation
Lin et al.	×[*]	○[*]	×, ×	×	×
Groza et al.	○	○	○, ×	×	×
Mundhenk et al.	○	○	○, ×	×	×
Fassak et al.	×	○	○, ×	×	×
Woo et al. [13]	○	○	○, ×	○	×
This paper	×	×	○, ○	○	○

[*] ○ means included, × means not included.

This paper provides two main novelties. The first is that a remote key update scheme for in-vehicle multi-layer ECUs is proposed, and its security properties are analyzed and verified. To our best knowledge, the remote key update scheme was not discussed in previous studies. The second is that key storage, the key usage environment, and the AES-CCM algorithm which uses fewer keys are discussed and experimentally verified, especially for the in-vehicle embedded environment.

The structure of this paper is as follows: Section 2 shows the trend of vehicle network architecture. Section 3 describes the extended AKEP2-based key update scheme and the proposed on-the-air (OTA)-based key update and destruction scheme for multi-layer ECUs. Key storage and a trusted key usage environment based on hardware are discussed in Section 4. The AES counter with CBC-MAC (AES-CCM) algorithm is introduced for secure communication in multi-layer ECUs to lower the number of stored keys in Section 5. Related experiments are shown and analyzed in Section 6. In Section 7, conclusions and future work are explained.

The research methodology in this paper is based on the existing research and our best knowledge of the vehicle KMS, where some worthy issues were found such as the remote key update, key storage, and key usage. The proposed solutions and experiments are also presented. The main work in this paper is summarized as follows:

1. The frame counter is a practical method to prevent replay attacks in in-vehicle networks [7]. The key distribution method based on AKEP2 has low overhead and good real-time performance in vehicle environments [13]. Based on the research of the above two groups and OTA technology, a key management scheme involving in-vehicle multi-layer ECUs is proposed and its security properties are analyzed and verified.
2. Key storage and a trusted key usage environment are studied. The related experiment proves the availability of tampering detection based on a trusted platform module (TPM).
3. An alternative algorithm, i.e., an authenticated encryption algorithm called AES-CCM, is studied for CAN FD bus communication. This algorithm can reduce the number of keys from two to one while ensuring security. Measurement results of the execution time and bus load rate guarantee the algorithm's feasibility.

2. The Trend of Vehicle Network Architecture

For current vehicles in the market, their network architecture is a distributed network. Their main communication pattern is a CAN bus, and other patterns include LIN, MOST, and FlexRay. All of these vehicles have a low-performance gateway. The main function of the gateway is communication protocol transactions, for example, transforming FlexRay format data to CAN data, or transforming CAN data to LIN data. The general communication network architecture is shown in Figure 1.

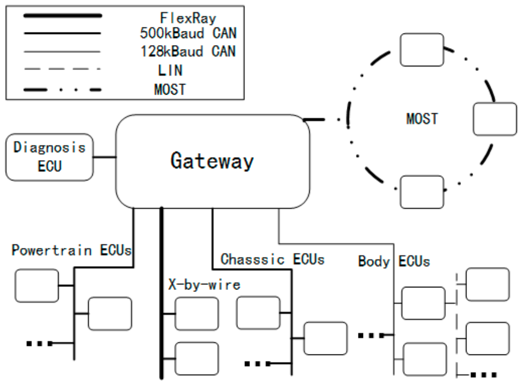


Figure 1. CAN bus serving as a backbone in most current vehicles.

The scheme proposed by Woo et al. in 2016 considered using CAN FD communication between the gateway and domain controllers. Its responding network architecture is illustrated in Figure 2. Its characteristic is a central gateway, and the domain controllers serve as a backbone.

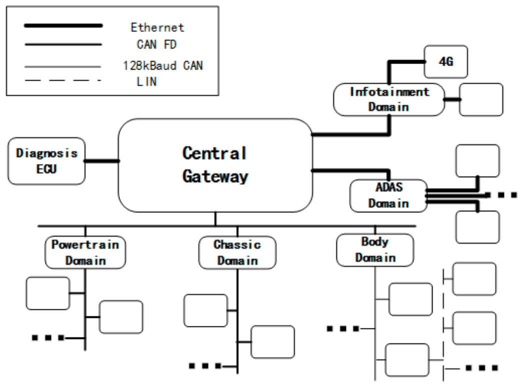


Figure 2. CAN FD and Ethernet serving as a backbone in vehicles.

Another very promising trend of vehicle networks is to use Ethernet as the backbone. Ethernet is the communication mode among the central gateway and all domain controllers as presented in Figure 3. Ethernet allows for flexible networking, such as star topology or bus topology.

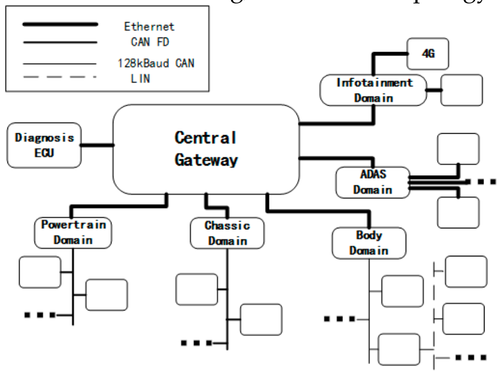


Figure 3. Ethernet serving as a backbone in vehicles.

Based on the discussion of vehicle network architecture above, it can be considered that the main communication methods of the in-vehicle network in the future will be as follows:

Case 1. CAN FD as the backbone network communication protocol, and CAN FD as the in-domain network communication protocol.

Case 2. CAN FD as the backbone network communication protocol, and CAN as the in-domain network communication protocol.

Case 3. Ethernet as the backbone network communication protocol, and CAN FD as the in-domain network communication protocol.

Case 4. Ethernet as the backbone network communication protocol, and CAN as the in-domain network communication protocol.

In the above four cases, it is assumed that the backbone network consists of a central gateway and domain controllers, and the domain here refers to the powertrain domain or chassis domain, which has higher security requirements.

In this paper, we describe our research for case 1. For case 3 and case 4, if the Ethernet network uses a bus topology, the Ethernet frame on the Ethernet bus can use the same arbitration protocol as the CAN FD frame. If the Ethernet network uses a star topology, which is the most likely, then our scheme for case 1 is still applicable. The reason is that our scheme is based on the research results of Woo et al. in 2016, which has a KDC that can be used for both bus topology and star topology. In short, our scheme for case 1 is also applicable for case 3 and case 4. If CAN is the communication protocol within the domain, then we use AKEP2 to derive keys and update keys for the in-domain network in the same way as Woo et al. in 2016; furthermore, for the encryption communication secure frames which include counter and truncated, a MAC value can be used as proposed by Lin and Sangiovanni-Vincentelli in 2012.

3. Extended AKEP2-Based Key Update Scheme and Proposed OTA-Based Key Update and Destruction Scheme for Multi-Layer ECUs

In this paper, the typical in-vehicle ECUs are classified as follows: gateway ECU (GECU), domain ECU (DECU), and function ECU (FECU). The related notations and nouns used in the ECUs are described in Table 2. A function ECU is an ECU that connects an actuator or a sensor. In this classification, the telematics unit is integrated in the GECU.

Table 2. Notations used in key update for multi-layer engine control units (ECUs).

Notation or Noun	Description
KDF	Key derivation function. It is a keyed one-way function used for key derivation.
CRL	Certification revocation list
Long-term key	A pair-wise symmetric key which cannot be updated when the vehicle is running. However, it can be updated by on-the-air (OTA) technology if needed. It can be used for key derivation and identity authentication.
Session key	A group symmetric key used in the AES-CCM or AES algorithm. It can be updated periodically according to the architecture proposed by Woo et al. in 2016.

We extend the application span of the AKEP2-based self-management scheme proposed by Woo et al. from DECU to FECU and additionally propose an OTA-based key update and destruction mechanism to realize all functions of the KMS, namely, key generation, key distribution, key storage, key update, and key destruction. When designing a specific solution, the two principles below are followed.

(1) For each vehicle, the vehicle manufacturer should manage as few keys as possible. Through this principle, the total number of keys which are managed by the manufacturers is reduced.

(2) For different vehicles of the same product series, their keys should be different. Through this principle, security incidents in which a key leak for one vehicle leads to a large number of key leaks for several are avoided.

3.1. Extended AKEP2-Based Key Update for Multi-Layer ECUs

Woo et al. proposed a security architecture to realize key generation, key distribution, and key update based on long-term keys, the AKEP2 algorithm, and a KDC in 2016 [13]. Its application span can be extended, and, in this paper, it is named the in-vehicle key self-management scheme. Keys stored in the ECUs of each layer are shown and explained in Table 3.

Table 3. Keys and functions included in the ECUs of each layer.

ECU Layer	Keys included and explanation
GECU	1. Long-term key. One GECU long-term key and some DECU long-term keys are included. Every vehicle has a unique GECU long-term key; 2. Vehicle manufacturer public key. One vehicle manufacturer public key is included; 3. Session key. One session key to DECUs; 4. KDF; 5. Vehicle's own certificate; 6. CRL, the list consists of invalid certificates of vehicles, suppliers, and telematic service providers (TSP).
DECU	1. Long-term key. One GECU long-term key, one DECU long-term key, and some FECU long-term keys are included. Every DECU has a unique DECU long-term key;

	2. DECU supplier public key. One supplier public key is included. Different DECUs have different DECU supplier public keys; 3. Session key. One session key for GECU and DECUs, and one session key for FECUs are included; 4. KDF.
FECU	1. Long-term key. One DECU long-term key and one FECU long-term key are included; 2. Session key. One session key for DECU and FECUs; 3. FECU supplier public key. One supplier public key is included; 4. KDF.

In the in-vehicle key self-management scheme, the session key is derived by the KDF, which requires the input of a nonce and long-term key. The session key is distributed and updated by the AKEP2 algorithm and a KDC. For GECU, the long-term key can be stored by the TPM storage mechanism. For DECU or FECU, the long-term key is stored in a hardware security module (HSM). For long-term key update and revocation, we can use an OTA-based management scheme.

3.2. Proposed OTA-Based Key Update, Key Destruction, and Certificate Revacation

The key management system mainly implements two functions via OTA technology. One is long-term key update or destruction. The other is the certificate revocation, which is implemented by updating the CRL. Figure 4 shows the related entities and the message stream.

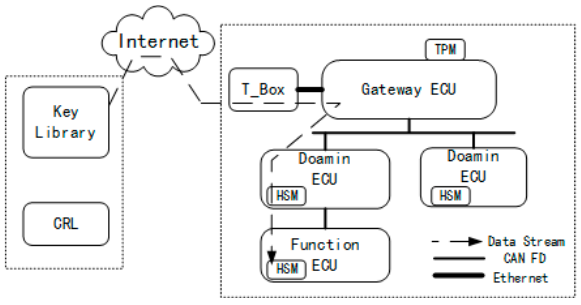


Figure 4. On-the-air (OTA)-based key update and certification revocation.

A procedure to update an FECU long-term key for the *n*-th time based on OTA technology is proposed and described below.

Figure 5 is the simulation stream of the proposed long-term key update. Role-M, role-G, role-D, and role-F stand for manufacture, GECU, DECU, and destination FECU, respectively. Steps 1–6 show the nonce transmission, while steps 7–12 show the update process of the FECU's *n*-th long-term key. Notations and nouns used in Figure 5 are explained in Table 4.

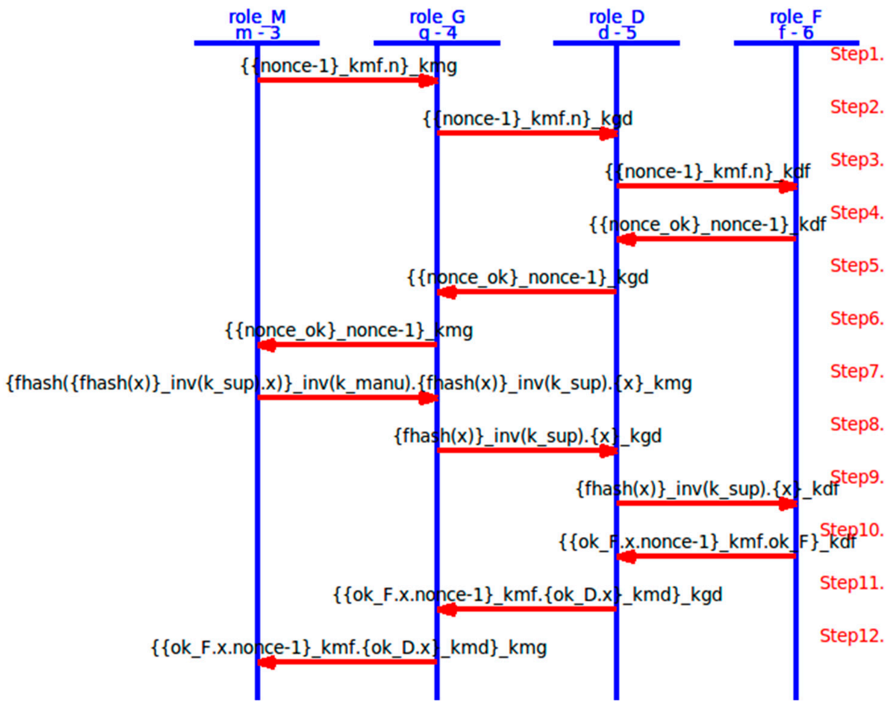


Figure 5. Simulation of the proposed OTA-based key update protocol.

Table 4. Notation or functions used for proposed OTA-based key update mechanism.

Notation or Noun	**Description**
Manu, k_manu	Manufacturer (Manu) and its public key
Sup, k_sup	Supplier (Sup) and its public key
kij	the ($n-1$)-th shared long-term symmetric key between entity i and entity j by default. In this paper, m, g, d, and f represent manufacture, GECU, DECU, and FECU, respectively.
{p}_k	Encrypt plaintext p by key k, which is either a symmetric or an asymmetric key.
n	n-th update.
x	x means the cipher of FECU's n-th long-term key kmf_n which is encrypted by (n − 1)-th long-term key kmf_(n − 1), i.e., x = {kmf_n}_kmf_(n − 1).
.	Notation "." connects different contents.
nonce-1	New nonce
fhash()	Hash function
inv(k_i)	k_i represents entity i's public key; inv(k_i) means i's private key.
nonce_ok, ok_F	Response message from FECU
ok_D	Response message from DECU

Table 5 explains the procedure of the FECU long-term key update for the *n*-th time. The security analysis and simulation experiment of the procedure are presented in Section 6.1.

Table 5. Explanation of OTA-based long-term key update procedure.

Step	Entity	Explanation
0		BEGIN
1	Manu	Generates nonce-1, and sends {{nonce-1}_kmf}_kmg to GECU via private channel.
2	GECU	Decrypts received data and encrypts it, then sends {{nonce-1}_kmf}_kgd to DECU.
3	DECU	Decrypts received data and encrypts it, then sends {{nonce-1}_kmf}_kdf to FECU.
4	FECU	Sends {{nonce_ok}_nonce-1}_kdf to DECU.
5	DECU	Decrypts received data and encrypts it, then sends {{nonce_ok}_nonce-1}_kgd to GECU.
6	GECU	Decrypts received data and encrypts it, then sends {{nonce_ok}_nonce-1}_kmg to Manu via private channel.
		Steps 1–6 realize nonce transfer
7	Manu and Sup	Manu generates x, then x is signed by Sup and Manu successively, and sends {fhash({fhash(x)}_inv(k_sup).x}_inv(k_manu).{fhash(x)}_inv(k_sup).{x}_kmg to GECU via private channel.
8	GECU	Verifies the signature of Manu and sends {fhash(x)}_inv(k_sup).{x}_kgd to DECU.
9	DECU	Verifies the signature of Sup and decrypts x to get the *n*-th key of FECU. Decrypts received {x}_kgd and encrypts it, then sends {fhash(x)}_inv(k_sup).{x}_kdf to FECU.
10	FECU	Verifies the signature of Sup and decrypts x to get the *n*-th key, then sends {{ok_F.x.nonce-1}_kmf.ok_F}_kdf to DECU.
11	DECU	Verifies ok_F and sends {{ok_F.x.nonce-1}_kmf.{ok_D.x}_kmd}_kgd to GECU.
12	GECU	Decrypts received data and encrypts it, then sends {{ok_F.x.nonce-1}_kmf.{ok_D.x}_kmd}_kmg to Manu via private channel.
13	Manu	Verifies ok_F and ok_D, updates the *n*-th value of FECU in the local database.
		Steps 7–13 realize x transfer and FECU's *n*-th long-term key update.
		FINISH

The in-vehicle key can be destroyed by an OTA-based mechanism when a vehicle needs scrapping. The procedure is the same as key update, except for the contents transmitted between the vehicle manufacturer and the vehicle.

Certificate revocation can be achieved by updating the CRL. In the OTA-based key management scheme, when the vehicle is scrapped, a vehicle certificate can be revoked by updating the CRL in the manufacturer's server. Similarly, if the vehicle manufacturer, supplier, or TSP claims that its certificate is invalid, its certificate can be revoked by updating the CRL in a vehicle gateway.

4. Application of Hardware-Based Key Storage and Trusted Key Usage in Proposed KMS

In an embedded system such as vehicle ECUs, key storage mainly relies on hardware isolation or encryption by a key encryption key (KEK). Usually, a hardware security module (HSM) is used to store a session key and provide an isolated environment for secure operations [14]. Except for key storage based on a KEK, TPM utilizes a trust anchor and integrity measurement function to construct a trusted key usage environment on a host microcontroller. Table 6 shows the differences between them.

Table 6. Comparison for hardware security module (HSM) and trusted platform module (TPM).

Item	HSM	TPM
Economy cost	Low	Medium
Resource cost	Low	Medium
Integrated pattern	Inside of microcontroller	Outside of microcontroller
Security	Medium	High
Function diversity	Medium	High
current product	Infineon AURIX Series, NXP MPC577x and Renesas RH850/P1x-C	Infineon TPM SLx96xx, STM ST33TPxF2E and NTC TPM NPCT75x
Current usage in vehicles	Domain controllers, functional controllers	Few in typical ECUs, probably in central gateway and head unit
Standards or specification	EVITA HSM [14]	TPM 2.0 Specification [15], TPM 2.0 Automotive Thin Profile [16]

HSM is a simpler and low-cost method compared to TPM. The security of HSM depends on the hardware security. The secure access mechanism and dedicated memory mapping of HSM and other security strategies protect the data in the HSM. TPM is not integrated in a microcontroller, but is connected to the host microcontroller via an Inter-Integrated Circuit (I^2C) bus or other kinds of buses.

KEK is securely stored in the TPM. The ciphertext of the key is stored in a specific area in the host microcontroller because the read-only memory (ROM) of the TPM is limited. However, the area where the key ciphertext is stored is not as secure as the inside of the TPM. In order to prevent tampering with the key ciphertext or unauthorizedly reading the decryption result using KEK, it is necessary to construct a trusted key usage environment on the host microcontroller. The trusted environment can be built by measuring the integrity of the host's bootloader file and the operation system (OS) file using TPM. Figure 6 shows the integrity measurement function of TPM.

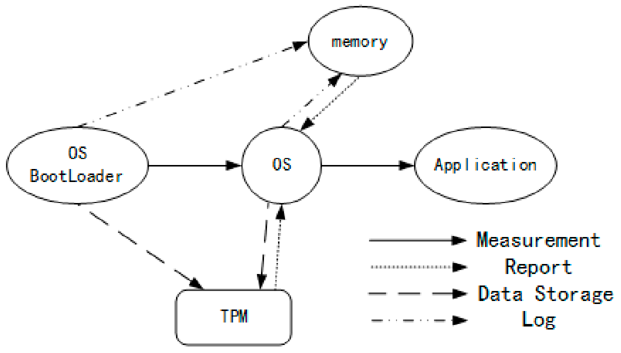


Figure 6. Integrity measurement based on trusted platform module (TPM) and construction of the chain of trust.

5. Application of AES-CCM Algorithm in Proposed Multi-Layer ECU Communication

As mentioned above, Woo et al. used an encryption key and authentication key to calculate ciphertext and MAC separately. Under the premise of ensuring the secrecy of keys, the same key can be used for both the authentication and encryption operations. This is the concept of the authentication encryption algorithm [17]. It has the advantage of reducing the number of keys used, which makes sense in embedded systems with limited storage space. In the TLS1.3 specification, authenticated encryption with associated data (AEAD) algorithms are adopted as the only stream encryption algorithms. As one of the AEAD algorithms in TLS1.3, AES-CCM has a short operation time and is considered secure [18]. Furthermore, AES-CCM has a good reuse of the AES code and this also means a lower cost of future AES-CCM hardware modules.

Figure 7 shows the steps for calculating the ciphertext and MAC values proposed by Woo et al. and the steps of the AES-CCM that we studied. Table 7 shows the meaning of the notations in the steps in Figure 7. The parameters and functions in Figure 7b are from or derived from the National Institute of Standards and Technology (NIST) Special Publication 800-38C [19].

$\text{if(j==1):}\ \ SECU_{ij}(7)\xrightarrow{M\|MAC}RECU_{ij}(7)(9)$

$\text{if}(2 \le j \le 3)\text{:}\ \ SECU_{ij}(5)(6)\xrightarrow{C\|MAC}RECU_{ij}(6)(8)(9)$

$(5)\ \ C = E_{EKk}(M \oplus CTR_{SECUij})$

$(6)\ \ MAC = H_{AKk}(C\|\ SECU_{ij}\|\ CTR_{SECUij})$

$(7)\ \ MAC = H_{AKk}(M\|\ SECU_{ij}\|\ CTR_{SECUij})$

$(8)\ \ M = D_{EKk}(C) \oplus CTR_{SECUij}$

$(9)\ \text{if}\ \ MAC == \textit{correct},\ \ \text{Accept M.}$

$\quad \text{else return False.}$

(**a**)

$\text{if(j==1):}\ \ SECU_{ij}(6^*)\xrightarrow{P\|T^*}RECU_{ij}(6^*)(8^*)$

$\text{if}(2 \le j \le 3)\text{:}\ \ SECU_{ij}(5^*)(6^*)\xrightarrow{C^*\|T^*}RECU_{ij}(7^*)(6^*)(8^*)$

$(5^*)\ C^* = \text{AES-CCM}_{EN}(K,P,Ctr[x])$

$(6^*)\ T^* = \text{AES-CCM}_{AU}(K,N,A,P,Ctr[x])$

$(7^*)\ P = \text{AES-CCM}_{DE}(K,C^*,Ctr[x])$

$(8^*)\ \text{if}\ \ T^* == \textit{correct},\ \ \text{Accept P.}$

$\quad \text{else return False.}$

(**b**)

Figure 7. Original steps by previous researcher (**a**) and steps using the AES-CCM algorithm (**b**).

Table 7. Notation used in steps in Figure 7.

Notation	Description
ECU_{ij}	ECU using identity "i" and belonging to the subnetwork with ASL grade "j" [13].
M	Message; it is plaintext used in Woo et al.'s work [13].
EK_k	Encryption key of k-th session [13].
AK_k	Authentication key of k-th session [13].
CTR_{SECUij}	ECU_{ij} data frame counter. It can be managed and synchronized in $SECU_{ij}$ and $RECU_{ij}$ by ACK bit [13].
K	Key used in AES-CCM.
N	Nonce, means random number.
A	Associated data; in our experiment, it is fixed to 8 bytes.
P	Plaintext; it has same meaning as M in step 8 in Figure 7a.
Ctr[x]	Array Ctr[] has m elements: $m=\lceil \lvert p \rvert / 128 \rceil$. In our experiment, Ctr[]'s x-th element is Ctr_x and $Ctr_x = N + x$.
C*	P's corresponding cipher. We define that C* is equal to $(P \oplus MSB_{\lvert P \rvert}(S))$ [19].
T*	P's corresponding tag. We define that T* is equal to $(T \oplus MSB_{r}(S_0))$ [19].
T*′	P's corresponding tag calculated by receiver ECU.
AES-CCM_{EN}	Equivalent function of Steps 5–8 in generation–encryption process [19].
AES-CCM_{AU}	Equivalent function of Steps 1–4,8 in generation–encryption process [19].
AES-CCM_{DE}	Equivalent function of Steps 1–5 in decryption–verification process [19].

The steps in Figures 7a,b show two independent algorithms for realizing confidentiality and authentication between two ECUs. With respect to the AES-CCM algorithm we studied, when j is equal to 2, steps 5* and 6* are executed in the sender ECU. After the receiver ECU receives $C^* \| T^*$, steps 7*, 6*, and 8* are executed successively. In the case that j is equal to 2, the security requirements are data confidentiality and entity authentication [13]. The difference between our study in Figure 7b and the former study in Figure 7a is that we use one key as opposed to two keys to calculate the ciphers and authentication value, which may mean less storage and easier management.

Before the AES-CCM algorithm can be used on the CAN FD bus between two ECUs, a secure communication time should be evaluated in the embedded system. Equation (1) is the one-way secure communication time containing AES-CCM operations between two ECUs. It mainly contains the encryption time, bus transmission time, and decryption time. The theoretical calculations and experimental results are illustrated in Section 6.3.

$$\tau_{\sec\_com} = \tau_{en} + \tau_{bus} + \tau_{de} \,. \tag{1}$$

6. Security Analysis and Experiment

6.1. Security Analysis on In-Vehicle KMS

The in-vehicle KMS in this paper consisted of two patterns: in-vehicle key self-management and OTA-based key management. The former involved extending the key management mechanism proposed by Woo et al. in 2016 to the CAN FD subnetwork within a certain domain. Thus, the security was the same as the scheme by Woo et al. in 2016. The security of the OTA-based key management was analyzed using two methods: provable security theory and simulation analysis based on the Delov–Yao model [20,21]. This section details the analysis of the security of the FECU *n*-th long-term key kmf_n in the OTA-based update procedure.

6.1.1. Analysis Based on Provable Security Theory

The principle of the provable security theory is to attribute the security of the cryptosystem to the security of the basic modules.

Confidentiality. Confidentiality is ensured because kmf_n is encrypted by kmf_(n − 1). As kmf_(n − 1) is stored and used in the secure hardware module of manufactures and the FECU, in our scheme, we assume it is secure.

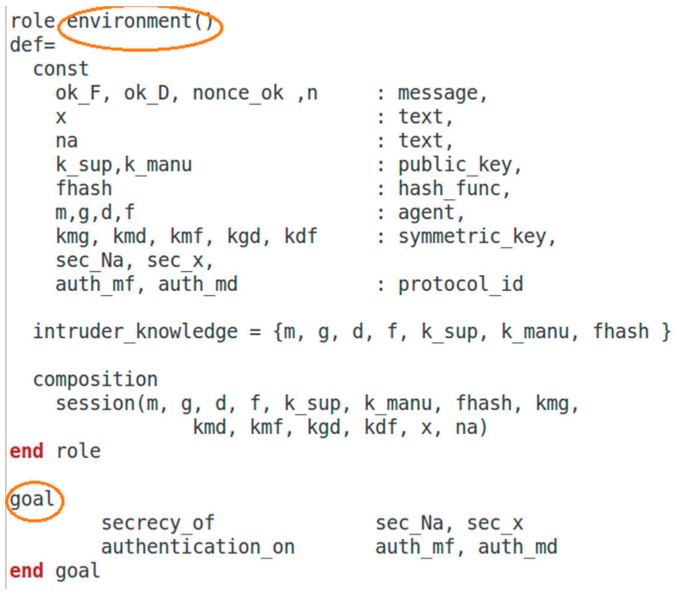
Source authentication and integrity. The two properties are ensured because the ciphertext of kmf_n is signed successively by the supplier and vehicle manufacturer. Moreover, the public keys or certificates used for verifying signatures are secured by the HSM, TPM, and CRL of entities.

Prevention for replay attacks. In our scheme, nonce and identifier n are used to prevent replay attacks.

6.1.2. Simulation Analysis According to Dolev–Yao Model

Dolev–Yao model analysis uses a formula to analyze the security property of a certain protocol, especially one which that contains public key encryption. The Dolev–Yao model has two basic assumptions: the cryptography is secure, and the intruder has full control over the network. SPAN is a software that supports the security analysis of a protocol according to the Delov–Yao model [22].

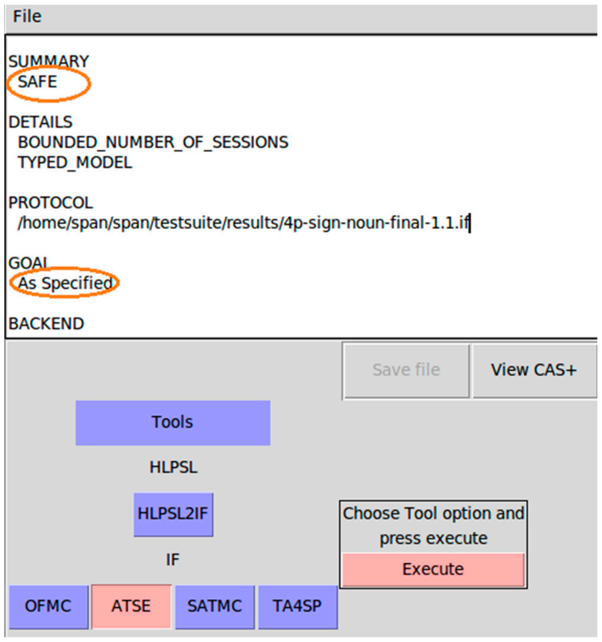
Figure 8 shows a part of the code for the OTA-based protocol in HLPSL under the SPAN software [23,24]. After the definitions of each entity and the sessions, the environment and security goals are defined successively, as presented in Figure 8. The goal is the secrecy of Na and x, authentication between manufacture and the FECU, and authentication between manufacture and the DECU. Under the SPAN software, the simulation for authentication can analyze not only the identification of two entities, but also the prevention ability for replay attacks.



```
role environment()
def=
  const
    ok_F, ok_D, nonce_ok ,n       : message,
    x                             : text,
    na                            : text,
    k_sup,k_manu                  : public_key,
    fhash                         : hash_func,
    m,g,d,f                       : agent,
    kmg, kmd, kmf, kgd, kdf       : symmetric_key,
    sec_Na, sec_x,
    auth_mf, auth_md              : protocol_id

  intruder_knowledge = {m, g, d, f, k_sup, k_manu, fhash }

  composition
    session(m, g, d, f, k_sup, k_manu, fhash, kmg,
                  kmd, kmf, kgd, kdf, x, na)
end role

goal
        secrecy_of                sec_Na, sec_x
        authentication_on         auth_mf, auth_md
end goal
```

Figure 8. Environment and goal definitions in HLPSL under SPAN software [22].

The ATSE tool integrated in SPAN was used to analyze the proposed protocol [25]. The simulation analysis result is shown in Figure 9. The proposed OTA-based protocol is safe, and the goal shown in Figure 8 was realized as specified.

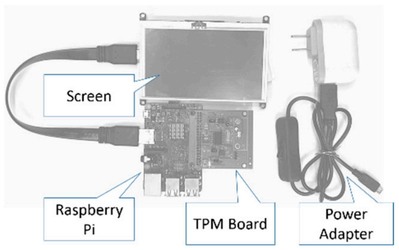


```
SUMMARY
 SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL

PROTOCOL
 /home/span/span/testsuite/results/4p-sign-noun-final-1.1.if

GOAL
 As Specified

BACKEND
```

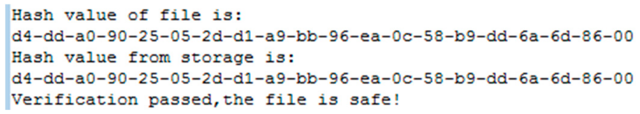
Figure 9. Simulation analysis result according to Dolev–Yao model.

6.2. Tamper Detection Experiment Based on TPM

In this paper, Raspberry Pi 3 was used to simulate a vehicle gateway which has a Linux OS and connects the Infineon TPM SLB9645 expansion board via an I2C bus. The TPM was programmed to perform the integrity check of a critical file after the Linux OS boots. The original file and the tampered file were tested separately, and the result is shown in Figure 10.

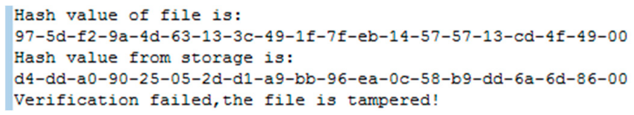


(**a**)



```
Hash value of file is:
d4-dd-a0-90-25-05-2d-d1-a9-bb-96-ea-0c-58-b9-dd-6a-6d-86-00
Hash value from storage is:
d4-dd-a0-90-25-05-2d-d1-a9-bb-96-ea-0c-58-b9-dd-6a-6d-86-00
Verification passed,the file is safe!
```

(**b**)



```
Hash value of file is:
97-5d-f2-9a-4d-63-13-3c-49-1f-7f-eb-14-57-57-13-cd-4f-49-00
Hash value from storage is:
d4-dd-a0-90-25-05-2d-d1-a9-bb-96-ea-0c-58-b9-dd-6a-6d-86-00
Verification failed,the file is tampered!
```

(**c**)

Figure 10. Experiment device for TPM module (**a**) and integrity measurement results (**b**,**c**).

The results show that the OS can effectively perform file integrity detection based on the TPM. Based on the same principle and the trust chain extension, a trusted environment for key usage can be built by performing integrity verification on codes of bootloader files and OS files.

6.3. Security Analysis and Experiment for AES-CCM Algorithm on CAN FD Bus

The security of the AES-CCM algorithm was analyzed [18]. Although there are still some criticisms of the algorithm, it was adopted as the recommended standard of NIST [19]. Thus, we considered it to be secure in this paper.

The execution time of the AES-CCM algorithm in an AURIX TC297 microcontroller, the execution time of secure communication, and the bus load rate were measured. In the experiment, the compiler was a TriCore Eclipse IDE v4.3r3, and more devices are shown in Figure 11.

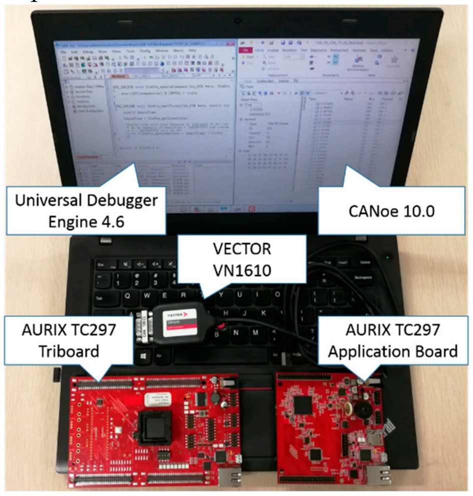
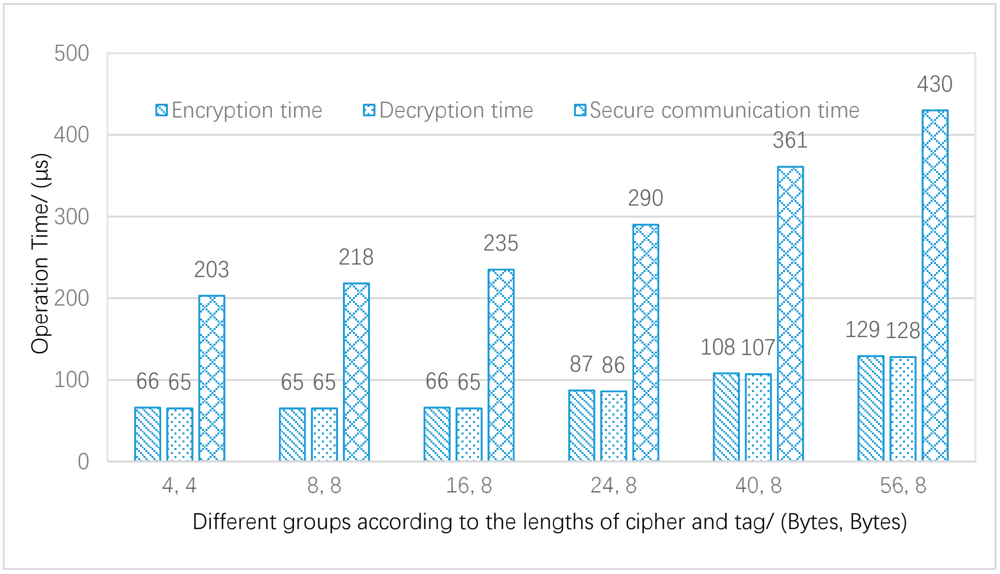


Figure 11. Experiment devices.

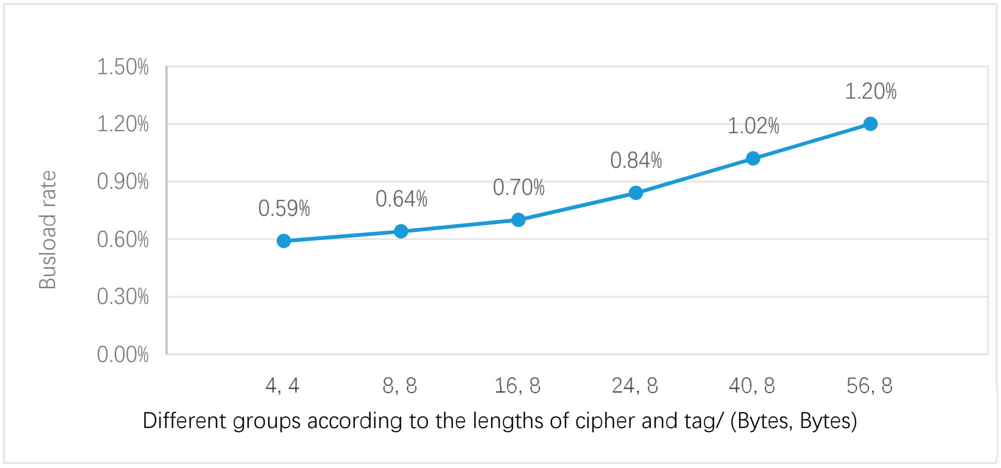
The set-up of the experiment is shown in Table 8. For a more accurate result, we did the measurement 10,000 times to get an average execution time of the algorithm and secure communication. The length of the AES-CCM output was chosen for two reasons. Firstly, 8 bytes was chosen because it can be compatible with a CAN frame which is already defined by manufacturers. Another reason is that the CAN FD frame supports some values of the length in the ISO specification. That is to say, the Data Length Code (DLC) values 8, 10, 12, 13, 14, and 15 stand for 8 bytes, 16 bytes, 24 bytes, 32 bytes, 48 bytes, and 64 bytes, respectively. The results are shown in Figure 12a.

Table 8. Evaluation set-up of AES-CCM on a CAN FD bus. CPU—central processing unit.

Item	Note
CPU frequency	100 MHz
Length of key	128 bits
Arbitration baudrate	500 kb/s
Data baudrate	5 Mb/s
Length of cipher for plaintext	4, 8, 16, 24, 40, or 56 bytes
Length of tag	4 or 8 bytes
Length of AES-CCM output	8, 16, 24, 32, 48, or 64 bytes
AES-CCM code source	tls.mbed.org



(**a**)



(**b**)

Figure 12. Execution time of algorithm and secure communication (**a**) and bus load rate on the CAN FD (**b**).

With Equation (1) and the encryption and decryption time in Figure 12a, the secure communication time can be calculated. Taking 56 bytes and 8 bytes as examples, according to the CAN FD specification, if stuffing bits are not included, there are about 27 bits translated at the arbitration baud rate and 543 bits translated at the data baud rate. It is easy to get the following:

$$\tau_{\sec\_com} = 129\mu s + (27 \times 2 + 543 \times 0.2)\mu s + 128\mu s = 419.6\mu s\,. \tag{2}$$

The result is 97.6% of a measurement time of 430 μs. The difference rate of 2.4% is mainly because the stuffing bits in the transmitted CAN FD frame are not included in Equation (2). After the comparison of theoretical values and experimental values, the experimental results are reasonable.

The result shows that, as the plaintext increases from 4 bytes to 56 bytes, the encryption time is from 66 μs to 129 μs. The decryption costs almost the same time, while the secure communication time is from 203 μs to 430 μs. The execution time of the AES-CCM algorithm is acceptable for in-vehicle embedded systems. Moreover, the time can be further reduced by an AES-CCM hardware module, which is a more efficient method to realize the algorithm. From the ratio of the secure communication time to the ordinary communication time, the time consumption of the secure communication can be compared. The ratio r varies between 2.26 and 2.82 according to Equation (3) and values in Figure 12a.

$$r = \frac{\tau_{\sec\_com}}{\tau_{\sec\_com} - \tau_{en} - \tau_{de}}\,. \tag{3}$$

When the frequency was set to 100 messages per second, the largest bus load rate was no more than 1.2%. The values are acceptable considering that the total bus load rate usually reaches 50%. Moreover, the increment values of ROM and random access memory (RAM) based on codes from tls.mbed.org are 27,686 bytes and 8688 bytes, which occupy only 0.33% of the total ROM and 0.31% of the total RAM.

7. Conclusions

This paper summarizes previous research on in-vehicle KMS and studies an OTA-based key update protocol, key storage, and trusted key usage environment, along with application of the AES-CCM algorithm in in-vehicle communication.

Considering the developing and layered in-vehicle network architecture, a key update scheme based on AKEP2 and OTA technology was proposed for the first time. SPAN software is an efficient tool to find potential attacks on cryptographic protocols. Simulation analysis in SPAN verified the security properties of the proposed OTA-based protocol. The properties verified were key confidentiality, authentication, and resistance to replay attacks. Key storage and a trusted key usage environment based on the secure hardware were discussed. Experiments on a microcontroller with TPM SLB9645 showed that the OS can recognize the behavior of tampering with a certain file by verifying the integrity of critical files or configurations, thus constructing a trusted key usage environment. AES-CCM can decrease the number of used keys. The experiment between two AURIX TC297 microcontrollers showed that the time consumption and bus load consumption of the CAN FD frame with AES-CCM are acceptable for the in-vehicle environment.

Our research can provide a reference for in-vehicle secure communication protocols and lifecycle key management for vehicle manufacturers. Future research will include lightweight secure protocols and a performance evaluation of in-vehicle Ethernet communication, as well as the implementation and optimization of trusted environment and key management based on TPM.

Author Contributions: Methodology, J.Z. and Z.W.; software, K.L. and F.L.; validation, J.Z. and K.L.; writing—original draft preparation, J.Z.; writing—review and editing, Y.Z.; supervision, Z.W. and Y.Z.

Funding: This research was supported by the National Key R&D Program of China (grant number 2017YFE0101400) and the Shanghai Automotive Industry Science and Technology Development Foundation (grant number 1703).

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

AKEP2	Authenticated Key Exchange Protocol 2
OTA	Qn-the-Air
AES-CCM	AES Counter with CBC-MAC
TPM	Trusted Platform Module
KMS	Key Management System
MAC	Message Authentication Code
TESLA	Timed Efficient Stream Loss-Tolerant Authentication
ECC	Elliptic Curve Cryptography
KDC	Key Distribution Center
TSP	Telematic Service Provider
KEK	Key Encryption Key

References

1. Miller, C.; Valasek, C. *Adventures in Automotive Networks and Control Units*; DEF CON 21 Hacking Conf.: Las Vegas, NV, USA, 2013.
2. Miller, C.; Valasek, C. *Remote Exploitation of an Unaltered Passenger Vehicle*; Black Hat USA: Las Vegas, NV, USA, 2015.
3. Kang, J.; Yu, R.; Huang, X.; Jonsson, M.; Bogucka, H.; Gjessing, S.; Zhang, Y. Location privacy attacks and defenses in cloud-Enabled internet of vehicles. *IEEE Wirel. Commun*. **2016**, *23*, 52–59, doi:10.1109/MWC.2016.7721742.
4. Wan, Z.; Zhu, W.T.; Wang, G. PRAC: Efficient privacy protection for vehicle-to-Grid communications in the smart grid. *Comput. Secur.* **2016**, *62*, 246–256, doi:10.1016/j.cose.2016.07.004.
5. Hamida, E.B.; Noura, H.; Znaidi, W. Security of Cooperative Intelligent Transport Systems: Standards, Threats Analysis and Cryptographic Countermeasures. *Electronics* **2015**, *4*, 380–423, doi:10.3390/electronics4030380.

6. Hu, Q.; Luo, F. Review of Secure Communication Approaches for In-vehicle Network. *Int. J. Automot. Technol.* **2018**, *19*, 879–894, doi:10.1007/s12239-018-0085-1.
7. Lin, C.W.; Sangiovanni_Vincentelli, A. Cyber-Security for the Controller Area Network (CAN) communication protocol. In Proceedings of IEEE International Conference on Cyber Security, Alexandria, VA, USA, 14–16 December 2012; pp. 344–350.
8. Groza, B.; Murvay, S. Efficient protocols for secure broadcast in controller area networks. *IEEE Trans. Ind. Inform.* **2013**, *9*, 2034–2042, doi:10.1109/TII.2013.2239301.
9. Woo, S.; Jo, H.J.; Lee, D.H. A practical wireless attack on the connected car and security protocol for in-Vehicle CAN. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 993–1006, doi:10.1109/TITS.2014.2351612.
10. Mundhenk, P.; Steinhorst, S.; Lukasiewycz, M.; Fahmy, S.A.; Chakraborty, B. Lightweight Authentication for Secure Automotive Networks. In Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, Grenoble, France, 9–13 March 2015; pp. 285–288.
11. Bellare, M.; Rogaway, P. Entity authentication and key distribution. In Proceedings of the 13th Annual International Cryptology Conference, Santa Barbara, CA, USA, 22–26 August 1993; pp. 232–249.
12. Fassak, S.; Idrissi, Y.E.H.E.; Zahid, N.; Jedra, M. A secure protocol for session keys establishment between ECUs in the CAN bus. In Proceedings of the International Conference on Wireless Networks and Mobile Communications (WINCOM), Rabat, Morocco, 1–4 November 2017; pp. 37–42.
13. Woo, S.; Jo, H.J.; Kim, I.S.; Lee, D.H. A practical security architecture for in-Vehicle CAN-FD. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 2248–2261, doi:10.1109/TITS.2016.2519464.
14. Ludovic, A.; Khayari, E.K.; Olaf, H.; Yves, R.; Hendrik, S.; Hervé, S.; Benjamin, W.; Marko, W. Secure automotive on-Board electronics network architecture. In Proceedings of the FISITA 2010 World Automotive Congress, Budapest, Hungary, 30 May–4 June 2010. Available online: https://evita-project.org/Publications/AEHR10.pdf (accessed on 16 August 2019).
15. Trusted Platform Module Library Specification, Family "2.0". Available online: https://trustedcomputinggroup.org/resource/tpm-library-specification/ (accessed on 16 August 2019).
16. TCG TPM 2.0 Automotive Thin Profile for TPM Family 2.0; Level 0. Available online: https://trustedcomputinggroup.org/resource/tcg-tpm-2-0-library-profile-for-automotive-thin/ (accessed on 16 August 2019).
17. Rogaway, P. Authenticated-encryption with associated-data. In Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington, DC, USA, 18–22 November. 2002; pp. 98–107.
18. Jonsson, J. On the security of CTR + CBC-MAC. In Proceedings of the SAC 2002: Selected Areas in Cryptography, St. John's, Newfoundland, Canada, 15–16 August. 2002; pp. 76–93.
19. Dworkin, M. *NIST Special Publication 800-38C Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*. National Institute of Standards and Technology, Gaithersburg, MD, USA, 2004.
20. Bellare, M. Practice-Oriented provable-Security. In Proceedings of the First International Workshop on Information Security, ISW'97 Tatsunokuchi, Ishikawa, Japan, 17–19 September 1997; pp. 221–231.
21. Delov, D.; YAO, A.C. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *9*, 198–208, doi:10.1109/TIT.1983.1056650.
22. Available online: http://people.irisa.fr/Thomas.Genet/span/ (accessed on 27 August 2019).
23. Oheimb, D.V. The high-Level protocol specification language HLPSL developed in the eu project avispa. In Proceedings of the APPSEM 2005 Workshop, Frauenchiemsee, Germany, 12–15 September 2005; pp. 1–17.
24. Available online: www.davoh.de/cs/talks/AVISPA-HLPSL.pdf (accessed on 27 August 2019).
25. Turuani, M. The CL-Atse Protocol Analyser. In *RTA 2006: Term Rewriting and Applications*; Springer: Heidelberg, German, 2006; pp. 277–286.