mathematics

MDPI

Article

A Provable Secure Session Key Distribution Protocol Based on NSSK for In-Vehicle CAN Network †

Long Yin, Jian Xu *, Zihao Wang and Chen Wang

Software College, Northeastern University, Shenyang 110167, China
* Correspondence: xuj@mail.neu.edu.cn
† This paper is an extended version of our paper published in SPNCE 2021: Security and Privacy in New Computing Environments, Virtual Event, 10–11 December 2021; pp. 35–52.

Abstract: Many CAN-based session key sharing approaches are based on the group key scheme, which can easily lead advanced adversaries to infiltrate all ECUs (electronic control units) in the network if the sharing key is leaked. To address the above problem, we propose a provable secure session key distribution protocol based on the improved NSSK (Needham–Schroeder shared key) protocol for the in-vehicle CAN network. We applied the mechanisms of message authentication and digital signature to fix the defects of the original NSSK regarding its lack of resistance to the Denning–Sacco attack. Then, we analyzed the provable security of the proposed protocol on the random oracle model and verified the security goals of the protocol by using the simulation tools AVISPA and Tamarin Prover; the results reflect that the protocol met the security requirements for key distribution such as session key secrecy, injective agreement, and known key secrecy. Finally, we compared our new protocol with other key distribution protocols in CAN bus communication to evaluate the performance of the new protocol in actual scenarios. The result shows that the protocol is secure against many payload-based attacks and is practical for in-vehicle CAN networks.

Keywords: CAN security; security protocols; vehicle cybersecurity; the NSSK protocol

MSC: 68M12

Citation: Yin, L.; Xu, J.; Wang, Z.; Wang, C. A Provable Secure Session Key Distribution Protocol Based on NSSK for In-Vehicle CAN Network. *Mathematics* **2022**, *10*, 2903. https://doi.org/10.3390/math10162903

Academic Editors: Ding Wang, Qi Jiang and Chunhua Su

Received: 21 July 2022
Accepted: 11 August 2022
Published: 12 August 2022

1. Introduction

The controller area network (CAN) bus is a widely deployed Fieldbus protocol that has been used in in-vehicle networks for more than three decades. Due to the characteristics of the CAN bus broadcasting mechanism, the raw CAN packet data are in plain text and easily eavesdropped by an adversary through an OBD (on-board diagnostics) tool connected to the network. There are two major types of attacks on the in-vehicle CAN bus, namely, the frequency-based attacks such as DoS or bus-off attacks, and the payload-based attacks such as impersonation or replay attacks. If the adversaries exploit the compromised ECU to transmit the masquerade or replayed CAN packets, due to a lack the protection of access control, some safe-critical ECUs may receive the malicious command within the forged CAN packet and cause an unexpected failure during vehicle driving.

A majority of research works on CAN-based security protocols focus on providing message authentication to combat impersonation and replay attacks. In many proposed schemes, all ECUs share a group key for in-vehicle secure communication. The sharing key can, however, be compromised by an adversary through wireless or physical access vulnerabilities of CAN bus networks. The adversary may inject malicious CAN packets with forged MACs by exploiting the compromised sharing key, then all ECUs in the network may accept the masquerade CAN packet and consider these packets to originate from a trustable ECU. So, when an ECU is compromised, other ECUs in the network are not secure anymore.

To alleviate these unfavorable factors of using a group-key-based scheme, an effective tactic is to establish a dynamic session key distribution process based on a centralized node-based key management scheme. Considering the actual scenarios of in-vehicle CAN communication, the ECUs in the different domains are connected to the gateway module in the center of the network, which is responsible for session key establishment and updates. The centralized node-based key distribution scheme is that the gateway module contains the different long-term keys with each ECU, and each two ECUs within the same domain transfer the shared session key under the participation of the centralized gateway module. Even if an ECU is compromised and the adversary possesses the session key of one domain, the adversary cannot communicate with the other ECUs in different domains. If the adversary sends a request to the gateway module for the session key of the other domains because it does not have the correct long-term key to decrypt the response from the gateway module, the adversary therefore fails to corrupt the ECUs in the other domains. Thus, a centralized node-based key distribution protocol is more secure and suitable for the CAN bus communication scenarios.

1.1. Related Works

Many CAN-based secure protocols for message authentication and session key sharing have arisen in recent years. Woo et al. produced a malicious self-diagnostic Android app to perform a long-range wireless attack by transmitting the masquerade control data from the attacker's server, and they proposed a security protocol based on AKEP2 to construct a secure initial session key distribution [1]. Wang et al. proposed a trusted group-based security framework, VeCure, which adopted a trust group-based structure combined with a self-designed message authentication scheme BME [2]; their proposed protocol reduced a large amount of computing consumption of traditional heavy-weight cryptographic function and online message processing delay by using an offline computation technique. Nürnberger et al. proposed a backward-compatible framework, vatiCAN, which provided a series of merits such as message authentication, replay-attacks resistance, spoof attacks detection, and prevention [3]. Radu et al. proposed an AUTOSAR-compliant, backward-compatible, and lightweight authentication protocol, LeiA, which has proven security and unforgeability under chosen message attacks [4]. Kang et al. proposed a lightweight authentication protocol SAP to resist masquerade attacks and replay attacks, being based on the one-way hash chain with the group key; they also proposed an attack-resilient mechanism, ART, to avoid hash collision attacks of the CAN environment [5]. Bella et al. proposed a protocol TOUCAN to secure the CAN bus against eavesdropping by unauthorized adversaries, one that uses a fast hash algorithm called Chaskey for message authentication and AES-128 for encryption [6]. Mun et al. proposed a model-based design to optimize the latency and the number of transmitted messages over the CAN bus for safety purposes and applied the HMAC for message authentication to enhance the security against spoofing attacks [7]. Youn et al. proposed sender authentication and key management schemes able to invalidate the impersonation and replay attacks, which update the session key seamlessly without requiring additional communication [8].

Palaniswamy et al. identified the weaknesses of the protocol in [1] and proposed a new protocol suite by adding a secure message exchange protocol for RTR frames and a key update protocol for the group session key renewing when the external device is released [9]. Schmandt et al. proposed Mini-MAC, which utilizes a provably secure HMAC to protect the ECUs against masquerade attacks, applying authentication keys shared among groups rather than sending separate messages to different recipients [10]. Groza et al. proposed a TESLA-like protocol that can perform broadcast authentication keys and MAC codes on the basis of a key-chain in the environment of CAN-bus, which operates without any process of session key distribution or negotiation [11]. Kurachi et al. proposed a lightweight authentication approach that employs a point-to-point authentication mechanism by using the monitor node to check the MAC in data frames and making minor modifications to the existing CAN controller to be easily used [12]. Wang et al. proposed a hardware-based

module security ECU, providing key distribution and management, as well as the MAC authentication against the adversaries transmitting malicious CAN packets to take over a vehicle, and uses Huffman coding with CAN frame compression to reduce the redundancy of messages and storage costs [13]. Jo et al. proposed an authentication protocol MAuth-CAN, wherein each ECU has a unique authentication key to protect itself from masquerade attacks initiated by a compromised ECU fabricating authentication values; the authors applied a dual CAN-controller to enhance the robustness of the authenticator in MAuth-CAN protocol against the bus-off attacks [14]. Wu et al. proposed a key management scheme based on AKEP2 and OTA techniques and utilized a trusted key usage environment based on hardware for key storage [15]. Pullen used implicit certificates to derive authenticated public keys for ECUs, then applied the ECDH to calculate a sharing key and derived actual group keys by using an OTP secure cryptosystem [16]. Pan proposed a dynamic key generation scheme based on extracting the transient signal when the vehicle starts and generating the master key by the ECC algorithm [17]. Jain utilized the CAN bus arbitration mechanism to cause a collision by transmitting data simultaneously from two ECUs, deriving a secret group key from the arbitration result [18]. King proposed a solution that uses the HMAC algorithm and timestamping mechanism against denial of service attacks or replay attacks [19]. Fassak proposed a protocol based on ECC for ECU authentication and the session key establishment in CAN bus networks, then derived symmetric keys from the established session key for authenticating CAN frames [20].

1.2. Our Goal and Contribution

After introducing the related works, we here provide a comparison of the session or temporary key distribution in these works, as shown in Table 1. In some group-key-based schemes [2,3,6,7,10], there does not exist a key distribution scheme, and all the authentication keys are pre-assigned at the manufacturing stage. The other group-key-based schemes [1,5,8,9] adopt a key update scheme based on a variant of AKEP2, which transmits a secret seed in plain text and derives the session keys from a group shared long-term key. Once the group shared long-term key is compromised, the session key can be easily derived by the adversary who possesses the secret seed. The works [1,9,16,17,20] used the asymmetric authentication key exchange or ECDH protocols to update the session keys, bringing some considerable computational overhead to the resource-constrained ECUs. The centralized-node-based schemes [11–14] are based on the hash-chain-based authentication method, producing a group of authentication keys and broadcasting them periodically. The authentication keys are transmitted in public with the messages simultaneously to check the validity of the message, which increases the CAN bus load. The NSSK protocol [21] proposed a symmetric encryption key distribution approach to deliver the session key generated by the trust agent; however, it cannot recognize and resist the known key attacks such as the Denning–Sacco attack. The other NSSK-like protocols [22,23] eliminate this defect by adopting asymmetric encryption and cyclic group multiplication. These approaches have some considerable computational costs on cryptographic operations that cannot meet the real-time demand of resource-constrained ECUs in automotive CAN. After analyzing the above key distribution schemes, we proposed a lightweight provable secure centralized-node-based key distribution protocol based on symmetric encryption to solve the above shortcomings.

Our main contributions are as follows:

(1) We summarize the related cryptographic protocols and applications of in-vehicle CAN and analysis the disadvantages in their key distribution or update protocols. To prevent the risk of group-key-based schemes being compromised by exploiting vulnerabilities for launching impersonation attacks, we adopt the tactic of centralized node-based schemes and propose a new secure centralized node-based key distribution protocol that is based on the NSSK (Needham–Schroeder shared key) distribution protocol.

(2) To fix the defect of the original NSSK distribution protocol on resisting the Denning–Sacco attack, we adopt the message authentication and signature verification mechanism on NSSK against the invalid replayed encrypting message attack. Considering the application in the actual communication scenarios of the automotive CAN, we split the key distribution protocol into two-stage protocols, namely, the initial SKDP (initial session key distribution protocol) and second SKDP (second session key distribution protocol). We prove the security and robustness of the initial SKDP and second SKDP by formally analyzing with the ROM (random oracle model) and verifying the security goals in the simulation tool AVISPA and Tamarin Prover. The results showed that the proposed protocol is safe and fits the security demand of session key secrecy and known key secrecy.

(3) We evaluated the performance of our proposed protocol on a CAN prototype test suite. We compared our proposed protocol with the other session key distribution on the computation time complexity and the number of storage message bits. The evaluation result showed that our proposed protocol has a competitive performance on both the computation time and storage cost. We measured the actual execution time and data frame number of the protocols. Moreover, we found these observed indicators of our proposed protocol dropped by 70% and 23% when we replaced the ECDSA with a lightweight signature algorithm ED25519, which showed the fitness of our new protocol on the real-time demand of the resource-constrained ECUs.

Table 1. Summary of some related cryptographic works of the automotive CAN.

Scheme	Research Work	Key Usage	Key Distribution Approach	Resistance to Attacks
Group-key-based	Wang et al. [2] Nürnberger et al. [3] Bella et al. [6] Mun et al. [7] Schmandt et al. [10]	Message authentication	No fixed or pre-assigned keys	Impersonation and replay attacks
	Woo et al. [1] Palaniswamy et al. [9]	Message authentication and encryption	Based on AKEP2 and asymmetric DH protocol	Impersonation, replay, and MITM attacks
	Kang et al. [5] Youn et al. [8]	Message authentication	Based on AKEP2	Impersonation and replay attacks
	Wu et al. [15]	Message authentication and encryption	Based on asymmetric encryption	Impersonation, replay, and MITM attacks
	Püllen et al. [16] Pan et al. [17]	Encryption	Based on ECDH	Impersonation, replay, and MITM attacks
Centralized node-based	Groza et al. [11] Kurachi et al. [12] Wang et al. [13] Jo et al. [14]	Message authentication	Based on hash-chain	Impersonation and replay attacks
Two parties	Fassak et al. [20]	Message authentication	Based on ECC-AKE	Impersonation, replay, and MITM attacks
Three parties	Needham et al. [21]	Encryption	Based on symmetric encryption	Impersonation attack
	Yu et al. [22]	Encryption	Based on symmetric and asymmetric encryption	Impersonation, replay, MITM, and Denning–Sacco attacks
	Arora et al. [23]	Encryption	Based on symmetric encryption with cyclic group multiplication	Impersonation, replay, MITM, and Denning–Sacco attacks
Centralized node-based	This work	Message authentication and encryption	Based on symmetric encryption and signature verification	Impersonation, replay, MITM, and Denning–Sacco attacks

2. Preliminaries

2.1. System Architecture

The system model of the in-vehicle network is shown in Figure 1. The ECUs belong to different communication subnets of the CAN bus network. The ECUs are connected to the central gateway module directly or indirectly. In some research [1,9], the gateway module represents a host ECU node that organizes the ECUs negotiating a sharing session key. We call the gateway GECU. In our proposed protocols, it plays the role of a trusted agent to generate a secret seed for session key derivation.

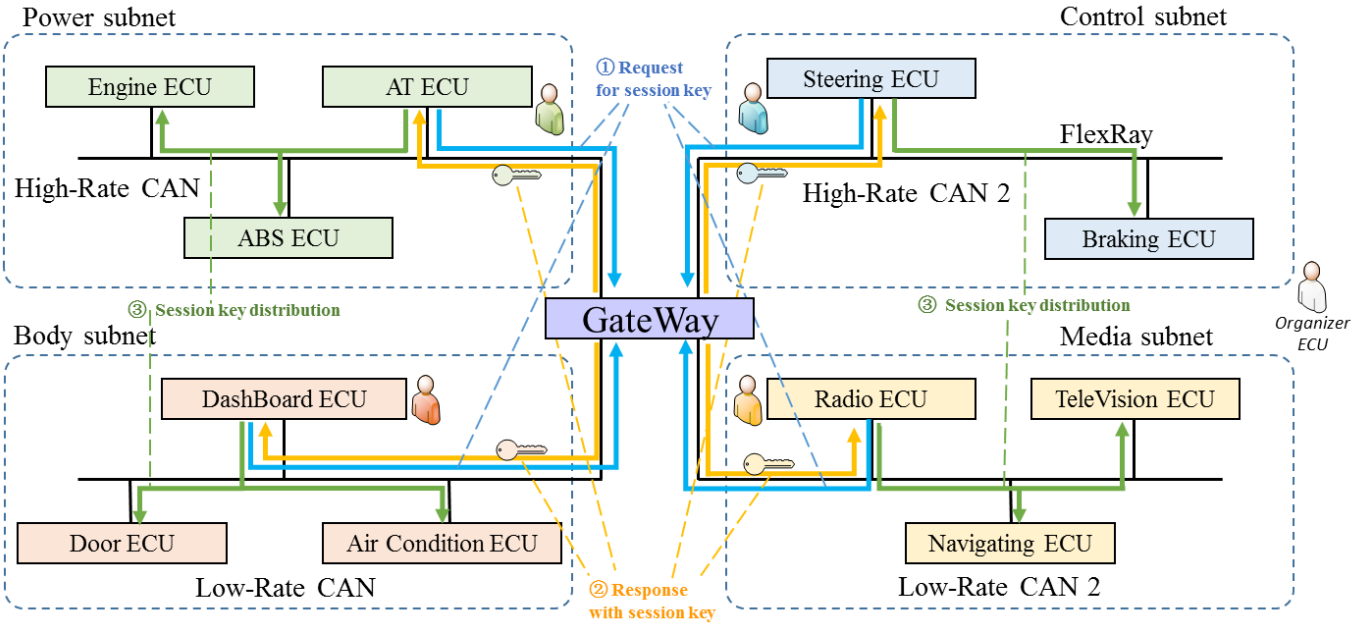


Figure 1. The structure of the in-vehicle network.

2.2. Session Key Sharing Approaches

There are two common session key sharing approaches often used in the automotive CAN environment: (1) a group-key-based approach, and (2) a centralized node-based approach. Some advantages and disadvantages of these approaches are listed as follows.

(1) Group-key-based approach [1–9]: All ECUs in the CAN bus network share one key to authenticate or encrypt CAN packet in automotive CAN. Figure 2a shows the group-key-based approach's network topological structure. Due to all ECUs sharing the same key, the sender ECU only sends a CAN message once. All ECUs in the network can receive the message and verify or decrypt the message data. Although the group-key-based approach is convenient, it cannot, however, completely resist the impersonation attack initiated by an advanced adversary. The group sharing key for message encryption and authentication may have a risk of being compromised by an advanced adversary to penetrate the ECU. Although each group sharing key is usually only bound to one group, some cases may exist where an ECU is located in two different domains at the same time. Thus, the connection points between different groups are more likely to be exploited to launch attacks by advanced adversaries.

(2) Centralized node-based approach: some works [11–14] applied a centralized node-based approach for key sharing. In this approach, each ECU establishes its secret key with a centralized ECU to defend against masquerade attacks launched by compromised ECUs. The centralized ECU is responsible for message authentication and encryption. Figure 2b shows the network topological structure of the centralized node-based approach and message data flows. In general, each ECU shares a distinct key with the central ECU. Each sender ECU encrypts its CAN packets or generates

MACs by using the shared key with the central ECU. The central ECU uses its sender ECU's shared key to decrypt the CAN packets and verify the MACs. Finally, the centralized ECU informs other ECUs about the verification results, which are encrypted by different receiver ECUs' shared keys. These studies mostly employed the hash-chain-based authentication method.

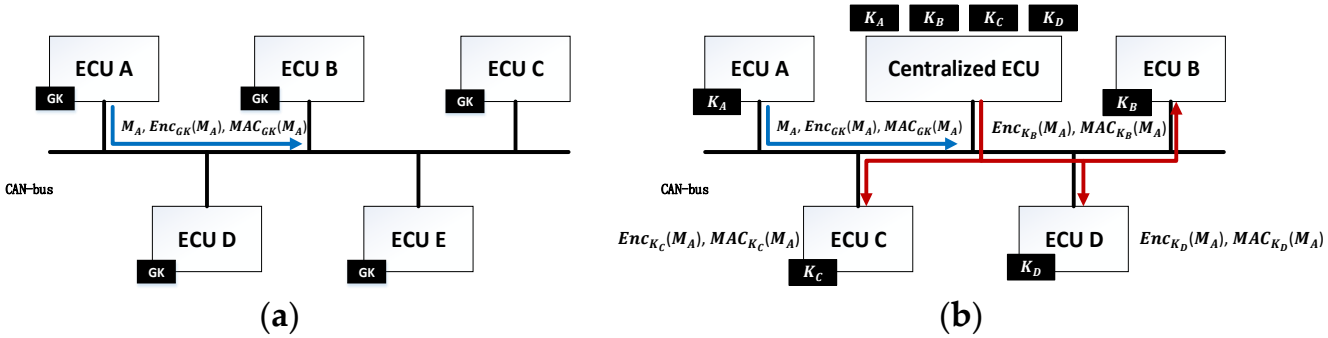


Figure 2. Session key sharing methods: (**a**) group-key-based approach; (**b**) centralized-node-based approach.

2.3. Attack Surfaces of Automotive CAN

There are two major attack surfaces for the in-vehicle network adversaries: (1) physical attack surfaces and (2) wireless attack surfaces.

(1) Physical attack surfaces: Modern vehicles have many exposed physical or hardware access interfaces such as the OBD or USB port, which can be used for ECU status of diagnosis and firmware updates. Attackers may access these interfaces to compromise ECUs. For example, Checkoway et al. [24] plugged an OBD-II dongle device into the automotive CAN to find vulnerabilities. They demonstrated the OBD-II dongle device could be easily exploited by attackers within the same network due to a lack of access control. The adversary can easily install a malicious program on the device to compromise the ECU via firmware updates. In addition, they found the vehicle's audio system updated automatically by re-flashing the firmware at a specific USB drive file path. They demonstrated that an adversary can exploit the defect of the target audio system to plant a modified firmware or an executable file that can transmit malicious packets.

(2) Wireless attack surfaces: Physical-based attacks on the vehicle need physical access to penetrate the in-vehicle network. However, some physical access points to a vehicle may be removed before leaving the factory. Therefore, the wireless access attack is proposed to overcome the limitations of physical access requirements. The adversary can transmit malicious CAN packets remotely through wireless channels between the adversary and telematics devices. Checkoway et al. demonstrated that some telematics devices equipped with Bluetooth may more likely offer remote access to the CAN bus. They reverse-engineered the target vehicle's Bluetooth protocol stack and found the vulnerability that the program did not check the input size of allocated memory. They launched a buffer overflow attack on a Bluetooth pairwise device, then they proved that the penetrated device can execute malicious code on the target telematics device by sending arbitrary packets remotely [24]. Miller et al. found remote access vulnerabilities in Jeep's Uconnect system. They manipulated a femtocell device to establish the wireless access channel to the Uconnect device and sent arbitrary packets remotely to the CAN bus network [25].

2.4. Adversary and Attack Types

We summarized the actual in-vehicle network adversaries into two types: common adversary and advanced adversary.

(1) Common adversary: They attempt to launch an attack on automotive CAN through the OBD port. They can exploit the vulnerabilities of OBD-based services to access

safety-critical ECUs and cause a CAN bus network hazard by injecting many CAN packet into the vehicle.

(2) Advanced adversary: They can compromise ECUs through vulnerable physical or wireless access points. They can exploit the vulnerabilities of remote opening ports of external networks and physical access ports to gain access to compromise the ECUs. By injecting malicious packets to the CAN bus network through these attack surfaces, they can control the safety-critical ECUs or cause a network hazard without using physical OBD-based services or devices.

The attack on the CAN bus can be classified into two categories: frequency-based attacks and payload-based attacks. Most of the cryptographic-based research focuses on protecting the ECUs from payload-based attacks.

(1) Frequency-based attacks: The frequency-based attacks contain two major types: DoS (denial of service) and bus-off attacks. The DoS attack is shown in Figure 3a. It is initiated by a common or advanced adversary who injects the highest priority CAN packet to the automotive CAN network, such as the CAN ID of 0x00. When the network is flooded by the highest priority CAN packet, the ECUs that send the packets with lower priority will lose the arbitration of the CAN bus. The bus-off attack, which is shown in Figure 3b, exploits the defect of the CAN bus arbitration mechanism to execute the attack. A common or advanced adversary knows the regularity of the target ECU data transmission period and sends the same CAN-ID packets with bit errors simultaneously while the target ECU sends legitimate packets. The adversary causes a series of arbitration conflicts to suspend the target ECU. While the target ECU enters the bus-off state, the compromised ECU will take over the traffic with forged packets.

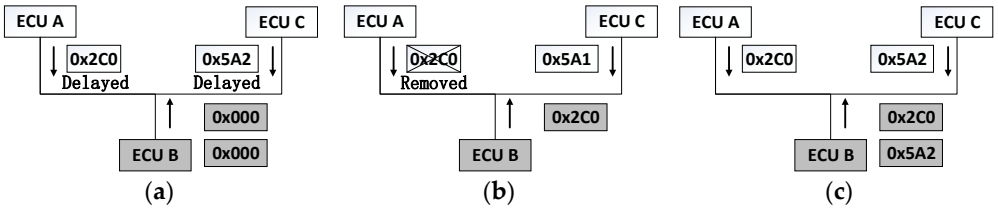


Figure 3. The CAN bus attack types: (**a**) DoS attack; (**b**) bus-off attack; (**c**) payload-based attack.

(2) Payload-based attacks: The payload-based attacks consist of many different types such as impersonation attacks, replay attacks, and MITM (man-in-the-middle) attacks. The impersonation attack is a so-called spoofing attack that is launched by a packet payload modification adversary. The impersonation adversary changes the content of CAN packets to disturb the safe-critical ECUs' function and judgment. Conversely, the replay attack adversary retransmits the past CAN packets without any modification to prevent the vehicle from operating properly in a normal state. The MITM attack is a combination attack of an impersonation attack and a replay attack—it will make the pair-wise ECUs think that the adversary is the real communicating entity with them and finally cause data leakage. Figure 3c shows the payload-based attacks.

(3) Denning–Sacco attack: This attack is a kind of known-key attack for the Needham–Schroeder SKDS scheme [21]. After the role Alice and Bob in NSSK have executed many rounds, the Denning–Sacco adversary cracks the session key from the past transfer messages and replays an old encrypted message with a form such as $\{K_{ab}, ID_{Alice}\}K_{bs}$ to Bob, where K_{ab} represents the cracked session key, ID_{Alice} is the identity of Alice, and K_{bs} is the long-term symmetric key for Bob. Due to the lack of timestamp validation stage in NSSK, the receiver Bob will accept the replayed message from the adversary and confirm the replacement of the sharing session key between the actual role of Alice and Bob [26].

We acknowledge that it is hard to resist the frequency-based attacks at the protocol level. To deal with these attacks requires the help of physical level methods by isolating the compromised ECU who broadcasts the flooding traffic in the CAN bus. Thus, the focus of this paper is to prevent an adversary to permeate the in-vehicle network from payload-based attacks and avoid the adversary getting the vulnerabilities at the protocol level of automotive CAN.

3. Proposed Scheme

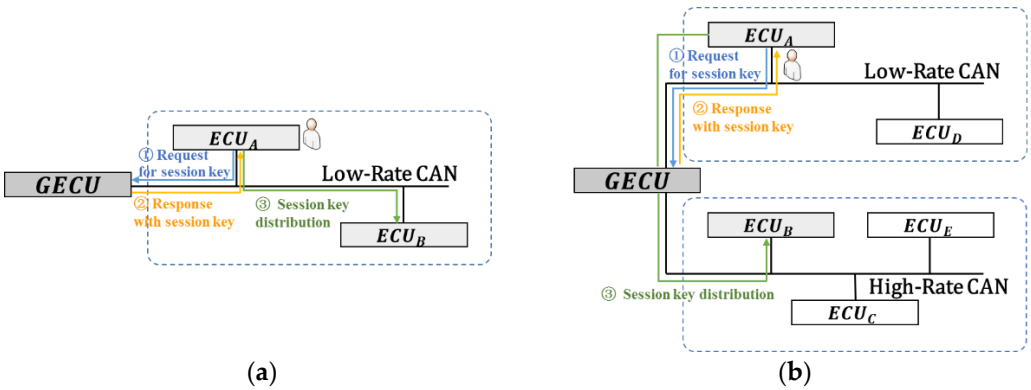
The reason that the original NSSK cannot resist the Denning–Sacco attack is mainly due to a lack of time stamping. In addition, the receiver cannot recognize the identity of the message sender, whether it is an honest party or a Denning–Sacco adversary. To overcome these issues, we proposed the improved protocols: initial session key distribution protocol (abbreviated as initial SKDP) and second session key distribution protocol (abbreviated as second SKDP). The declarations we used in the protocols are shown in Table 2.

Table 2. Notations of some cryptography terms.

Notation	Description
ECU_i	ith electronic control unit
ID_{ECU_i}	The identifier of ECU_i, i = A, B, C, ...
$GECU$	Gateway ECU
$Seed$	A seed value of the session
E_K	Encryption key of the session
A_K	Authentication key of the session
GK	The root key for session key derivation
$K_{A,\,i}$	Long-term key of $\mathrm{ECU_A}$; $K_{A,1}$ is the encryption key and $K_{A,2}$ is authentication key
$KDF_x()$	Keyed one-way function used for key derivation
$H_K(.)$	Keyed-hash message authentication code (HMAC)
$E_K(.)/D_K(.)$	Symmetric encryption/decryption function
$\{.\}_{sk}$	Signature-generating function, where sk is the private key

3.1. Initial Session Key Distribution Protocol

When the ECUs within the communication domain initiate the session key distribution, they elect an organizer ECU to send a session key update request to the gateway module GECU. The ECUs may be clustered in a subnet (as shown in Figure 4a) or scattered in different subnets (as shown in Figure 4b). The elected organizer ECU_A plays the role of the initiator of NSSK and initiates the key distribution with GECU and another ECU, i.e., ECU_B. The GECU plays the role of a trusted agent to generate the secret session key materials. We call the first stage of key distribution the initial session key distribution protocol, which is abbreviated as initial SKDP.



(**a**) (**b**)

Figure 4. The initial SKDP in different scenarios: (**a**) one single subnet; (**b**) multiple subnets.

The initial SKDP is shown in Figure 5. The ECU_A firstly generates a random number R_A and transmits the request containing the random number R_A to GECU. Then, GECU generates a secret seed and seals it into cipher-text t_B by the long-term key $K_{B,1}$. Following this, GECU encrypts the message sequence $R_A||ID_{ECUB}||ID_{GECU}||Seed||t_B$ as y_1 by the long-term key $K_{A,1}$. Thus, the t_B and y_1 are cipher-texts that contain the secret seed.

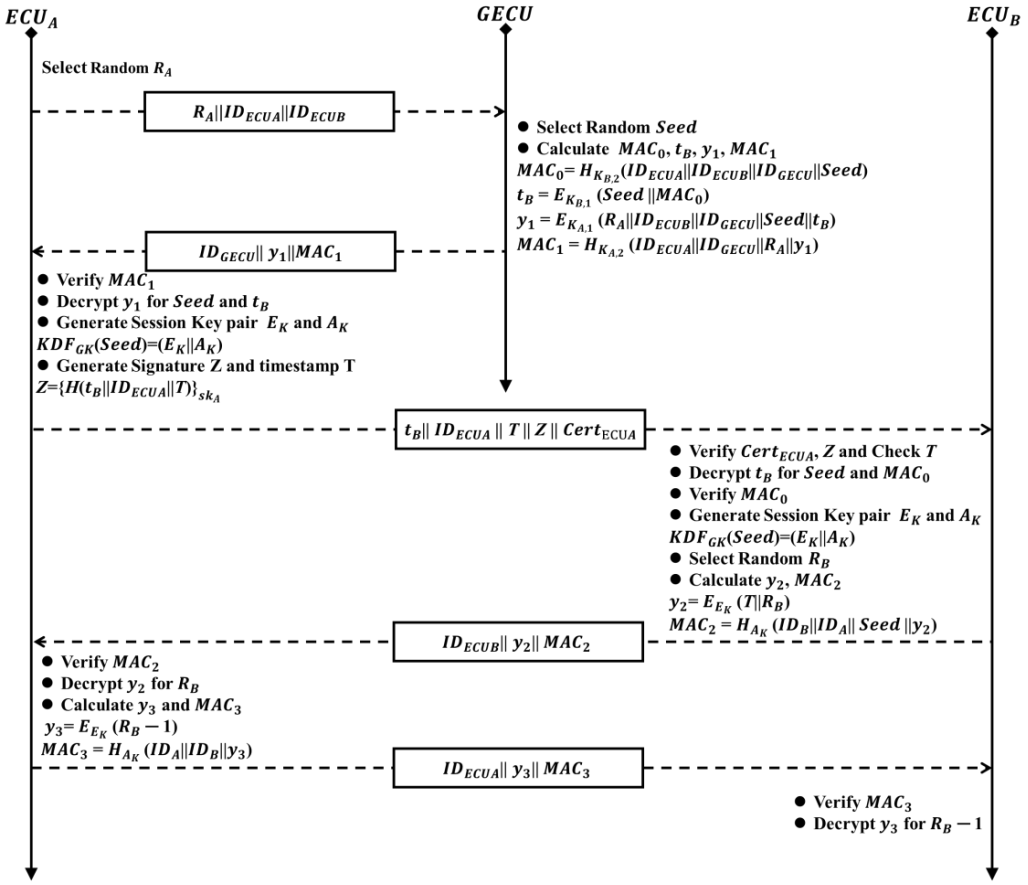


Figure 5. Initial session key distribution protocol (initial SKDP).

When ECU_A and ECU_B receive the y_1 and t_B, respectively, and decrypt the secret seed, they run a key derivation function KDF to generate session key pair E_K and A_K. Before each message transmission, the sender ECU calculates the MAC of the transmitted message for ensuring its data integrity.

At the time that ECU_A commits t_B to ECU_B, ECU_A uses its private key to sign the message H $(t_B||ID_{ECUA}||\mathrm{T})$, in which T is the current timestamp. Then, ECU_A sends t_B, T, and signature Z to ECU_B, and ECU_B checks the T and verifies the Z by the corresponding public key from the ECU_A's lightweight certificate $Cert_{ECUA}$, which can only be sent once only when it updates.

When ECU_B receives t_B, it decrypts t_B and derives the session key pair from the secret seed. Then, ECU_B generates a random number R_B and the cipher-text y_2 by the session key E_K and calculates the MAC_2 by the session key A_K. ECU_A takes them as the confirmation of the secret seed and the timestamp T.

3.2. Second Session Key Distribution Protocol

While the ECU_A finishes the session key distribution with ECU_B, if there is more than one rest ECU in the communication group, ECU_A will enter the second stage of key distribution and transfer the secret seed to the other ECUs such as the ECU_C by the assistance of GECU. The rest of the ECUs may sit in a subnet (as shown in Figure 6a) or be scattered in different subnets (as shown in Figure 6b).

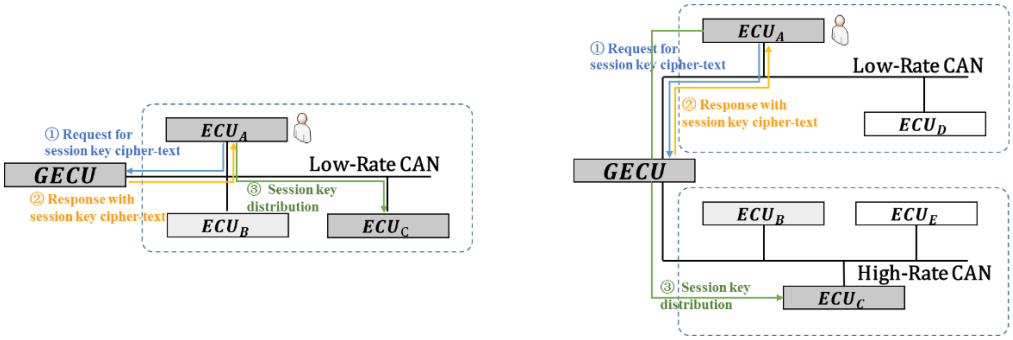


Figure 6. The second SKDP in different scenarios: (**a**) one single subnet; (**b**) multiple subnets.

The second session key distribution protocol, which is abbreviated as second SKDP and is shown in Figure 7, is identical to the initial SKDP, except for only a small difference at the beginning of the protocol. Because ECU_A has learned the secret seed after running the initial SKDP, it therefore sends the MAC_0 to GECU as a confirmation for the secret seed. While GECU receives the request and verifies the MAC, it agrees on the secret seed and then encrypts it with the long-term keys $K_{A,1}$ and $K_{C,1}$, as what has been performed in initial SKDP. The remaining steps of second SKDP are identical to those in initial SKDP as well, only the recipient ECU changes to the ECU_C.

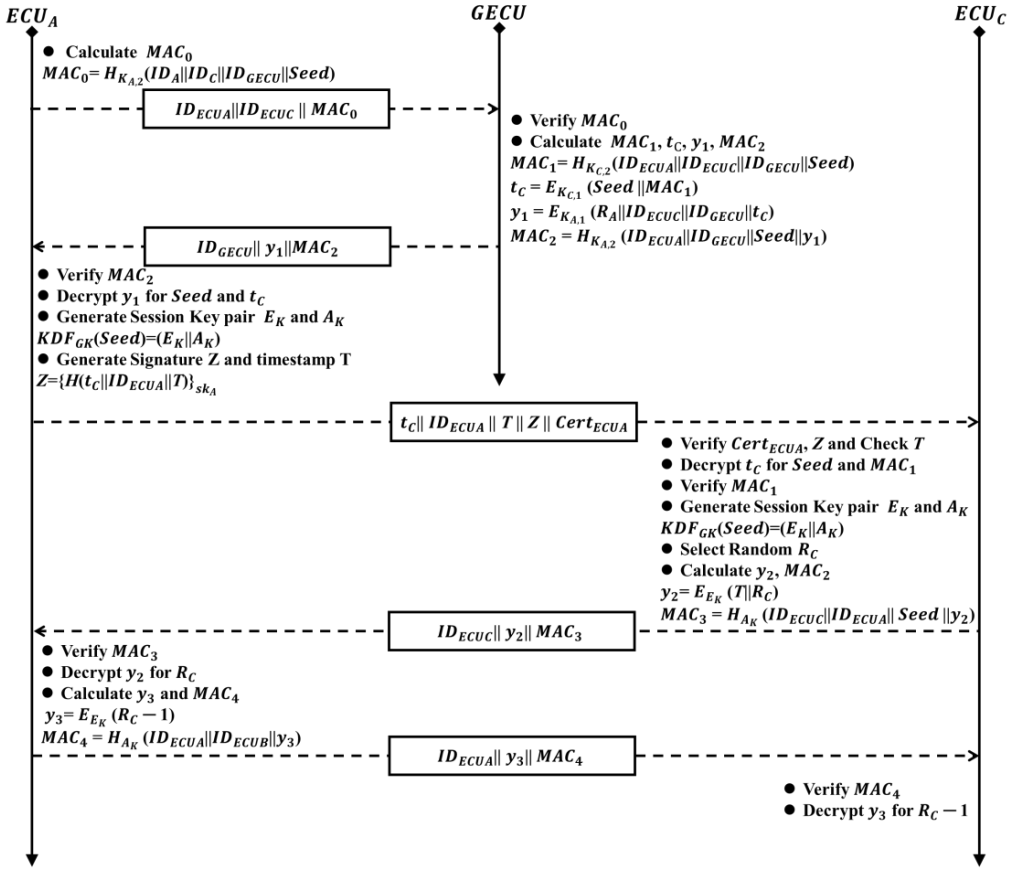


Figure 7. Second session key distribution protocol (second SKDP).

4. Security Analysis

4.1. Formal Security Analysis of the Proposed Protocol

In this subsection, we use the random oracle model to verify the semantic security attributes of the new protocols and define a series of random oracle model queries, as shown in Table 3.

Table 3. The random oracle model operation queries.

Query Operation	Description
Hash $(\cdot)$	A hash query initiated by an adversary for the correlate plain text M or the ciphertext C. If the correlate record has been already stored in the form of (M, f) or (C, g), the oracle will return the stored hash record f or g; otherwise, it returns a new random number f′ or g′ to the adversary and stores (M, f′) and (C, g′) in the oracle's hash table [26].
Sig $(\cdot)$	A signature query initiated by an adversary for the correlate plain text M; the oracle uses the signature table S to store plain text M and signature S pair in the form of (M, S). The working mechanism of *Sig* $(\cdot)$ is similar to *Hash* $(\cdot)$.
Enc $(\cdot)$	A ciphertext query initiated by an adversary for the correlate plain text M; the oracle uses the ciphertext table E to store plain text M and cipher text C pair in the form of (M, C). The working mechanism of *Enc* $(\cdot)$ is similar to *Hash* $(\cdot)$.
Send $(\cdot)$	The query imitates the message sent by the adversary.
Execute $(\cdot)$	The query imitates passive attacks such as spoofing or replaying the eavesdropped message.
RevealSessKey $(\cdot)$	The query imitates acquiring the session keys of the parties.
RevealSessState $(\cdot)$	The query imitates acquiring the secret internal variables of the parties.
Corrupt $(\cdot)$	The query imitates acquiring the long-term key of the parties.
Test $(\cdot)$	The adversary initiates the query in the final game. The queried party tosses a bit-value b of 0 or 1. If $b = 1$, it returns the real session key; otherwise, it returns a random number. The query succeeds only if an adversary can guess the b's value correctly in every game [26].

We summarize some symbolic definitions and theorems about the description of protocol semantic security as follows:

Definition 1 (Negligible function). *A function $\mu(n)$ is negligible if there exists $n_0 \in Z^+$ for any $c > 0$; it has $|\mu(n)| \leq \frac{1}{n^c}$ for all $n > n_0$* [26].

Definition 2 (Semantic security). *An adversary A initiates the query $Test(\cdot)$ for a given protocol P, only if the advantage of A is negligible or it satisfies $Adv_A^{SS}(P, n) = Pr[A(P, n) = b] - \frac{1}{2} \leq \mu(n)$, the protocol P is semantically secure* [26].

Definition 3 (Difference lemma). *M, N, and F are events defined in the same probability distribution* [26]. *If $M \wedge \neg F \equiv N \wedge \neg F$, then $Pr[M] - Pr[N] \leq Pr[F]$.*

Theorem 1. *For the initial SKDP or second SKDP, we provide the assumptions as follows: The advantage of guessing a long-term key is Adv_P^{SS}, where l represents the MAC length, and m presents the signature size; the encryption function can ensure k bits security. Then, the adversary can break the semantic security of the protocol with an advantage $Adv_P^{SS} \leq \frac{q_{hash}^2}{2^l} + Adv_{sk} + \frac{q_{sig}^2}{2^m} + \frac{q_{Enc}^2}{2^k}$, where q_{hash} represents the number of queries to oracle's hash table, and q_{sig} and q_{Enc} represent the number of queries to signature and encryption function oracle, respectively.*

To prove the theorem of the adversary's advantage to corrupt the initial SKDP and second SKDP, we define five games, namely, G_0, G_1, G_2, G_3, and G_4, where S_i represents the event when adversary A wins the game G_i, i = 0, 1, 2, 3, 4.

Game G_0. This game describes a real attack on the semantic security of initial SKDP (abbreviated as *P*). The adversary first activates protocol by *Send (P, "start")* and then uses *Execute (P)* to eavesdrop on the protocol message. The adversary initiates a series of queries, namely, *RevealSessKey (P)*, *RevealSessState (P)*, and *Corrupt (P)*, to fabricate the identity and takes a probabilistic polynomial-time to initiate the queries *Execute (P)* and *Hash (P)*. Finally, the adversary initiates the *Test (P)* query. The adversary wins the game only if it can correctly guess the bit *b*. According to Definition 2, the adversary wins the game with an advantage Adv_P^{SS}, as shown in Equation (1):

$$Adv_P^{SS} = |Pr(S_0) - \frac{1}{2}| \tag{1}$$

Game G_1. There is a collision with MACs in this game, which may happen in steps 4, 14, and 18 of initial SKDP, as shown in Table 2. If a MAC collision occurs, the adversary can falsify the secret seed and deceive ECUs to accept the fake secret seed. According to the birthday paradox, the advantage of a MAC collision happening is $\frac{q_{hash}^2}{2^l}$, and the difference in the advantage of this game and game $\boldsymbol{G}_0$ is represented in Equation (2):

$$|Pr(S_1) - Pr(S_0)| \leq \frac{q_{hash}^2}{2^l} \tag{2}$$

Game G_2**.** This game is relevant to the known key secrecy of the protocol. The adversary initiates *RevealSessKey (P)* for attaining the expired session keys [26]. Despite possessing the expired session keys, the adversary must acquire the secret *seed* to compute the current session key. The advantage of the adversary attaining *seed* is represented as Adv_{sk}, which denotes the probability of replacing the *seed* with a random number. Thus, the advantage difference is represented in Equation (3):

$$|Pr(S_2) - Pr(S_1)| \leq Adv_{sk} \tag{3}$$

Game G_3. The game is identical to the previous game G_2 except that there is a collision with the signature Z in this game, which may happen in step 7 of initial SKDP, as shown in Table 2. If a signature collision occurs, the adversary can forge a valid signature Z′ that is generated in step 8 of initial SKDP shown in Table 2 and cheat the other ECUs to accept what they have received. The advantage of a signature collision happening is $\frac{q_{Sig}^2}{2^k}$. Thus, the advantage difference is represented in Equation (4):

$$|Pr(S_3) - Pr(S_2)| \leq \frac{q_{sig}^2}{2^l} \tag{4}$$

Game G_4. This game is identical to the previous game G_3 except that the adversary queries the encryption table E to find the collision of t_B and y_1 in step 4 of initial SKDP shown in Table 2. If the adversary finds a collision, it can inject a forged secret seed and send it to the other ECUs. The advantage of a cipher-text collision happening is $\frac{q_{Enc}^2}{2^k}$. Thus, the advantage difference is represented in Equation (5):

$$|Pr(S_4) - Pr(S_3)| \leq \frac{q_{Enc}^2}{2^k} \tag{5}$$

On the basis of the above proofs, the advantage $\mathrm{Adv}_{\mathrm{P}}^{\mathrm{SS}}$ is calculated according to the triangle inequality as in Equation (6):

$$\begin{aligned} &\mathrm{Adv}_{\mathrm{P}}^{\mathrm{SS}} = |\mathrm{Pr}(\mathrm{S}_4) - \mathrm{Pr}(\mathrm{S}_0)| = |\mathrm{Pr}(\mathrm{S}_4) - \mathrm{Pr}(\mathrm{S}_3) + \mathrm{Pr}(\mathrm{S}_3) - \mathrm{Pr}(\mathrm{S}_2) + \mathrm{Pr}(\mathrm{S}_2) - \mathrm{Pr}(\mathrm{S}_1) + \\ &\mathrm{Pr}(\mathrm{S}_1) - \mathrm{Pr}(\mathrm{S}_0)| \leq |\mathrm{Pr}(\mathrm{S}_4) - \mathrm{Pr}(\mathrm{S}_3)| + |\mathrm{Pr}(\mathrm{S}_3) - \mathrm{Pr}(\mathrm{S}_2)| + |\mathrm{Pr}(\mathrm{S}_2) - \mathrm{Pr}(\mathrm{S}_1)| + |\mathrm{Pr}(\mathrm{S}_1) - \\ &\mathrm{Pr}(\mathrm{S}_0)| \leq \frac{q_{hash}^2}{2^l} + Adv_{sk} + \frac{q_{sig}^2}{2^l} + \frac{q_{Enc}^2}{2^k} \end{aligned} \quad (6)$$

The result of Equation (6) concludes the semantics security theorem. It means that if the secret seed is not compromised, the initial SKDP and second SKDP are semantically secure. During the cryptographic games executed on the random oracle model, the active adversary launches the impersonation attack and the replay attack by employing the *Execute (P)* queries and tries to launch the key stealing attack by employing the *RevealSessKey (P)* and *RevealSessState (P)* queries. Due to the provable security of the proposed protocol, the secrecy of the session key is secure on the robustness of the protocol and the secrecy of the secret seed. We have proven that our protocol is safe in Section 5 so that the adversary cannot guess the secret seed by employing the *Corrupt (P)* queries without the knowledge of the secret seed. Then, the semantic security of the proposed protocol is proven in that the proposed protocol is secure to resist the impersonation; replay; Denning–Sacco attack; and the MITM attack, which is a combination of the replay attack and impersonation attack.

4.2. Informal Security Analysis of the Proposed Protocol

With the proof of proposed protocols on provable security, we discuss the capability of protocols for resisting some common session key stealing attacks by taking an informal security analysis of proposed protocols. There are some merits of our proposed protocols that are listed as follows.

(1). Session key freshness: The initial SKDP and second SKDP guarantee the session key freshness by a random generated *Seed* and generate session key pair (E_K, A_K) by using the key derivation function KDF.

(2). Resistance to replay attack: For the initial SKDP and second SKDP, any replay attack is invalid. For the proof of initial SKDP, when an adversary impersonates ECU_A to replay cipher-text t_B, signature Z, and timestamp T in a stored message to ECU_B, ECU_B will examine the timestamp T and whether it is expired at first; if the examination fails, then the replayed data will be discarded. For the data transmission and reception, the receiver uses a long-term key shared with the GECU to authenticate MACs as soon as the data are received. If the MAC authentication fails, then the receiver will discard the received data. The proof of second SKDP is identical to initial SKDP, and therefore the protocols can resist replay attacks.

(3). Resistance to impersonation and Denning–Sacco attack: For the initial SKDP and second SKDP, the probability of an adversary succeeding to guess the session key by launching the impersonation attack is negligible. According to the proof we have analyzed above, the adversary can win this game only by gaining the leaked long-term keys of ECU_A and ECU_B; otherwise, they cannot solve the encrypted seed value and derive the session keys. In addition, the initial SKDP adversary has to provide a valid signature to ECU_B for authenticating the identity. It means that the adversary requires the knowledge of the signing private key to fake the signature. The difficulty of cracking the signing private key is equivalent to the difficulty to solve a discrete logarithmic problem, which might fail with an overwhelming probability in probabilistic polynomial time. Thus, the proposed protocols can resist the Denning–Sacco attack by rejecting the invalid replayed message.

(4). Resistance to MITM attack: The initial SKDP and second SKDP can resist MITM attacks. The adversary may utilize the past session's eavesdropped messages or newly forged messages to launch an MITM attack. This is a combination of a replay attack and a masquerade attack. According to the results discussed above in (2) and (3), our proposed protocols can resist MITM attacks as well.

5. Security Verification

For the proposed scheme security and robustness verification, we used the well-known simulation tool AVISPA to prove the validation results of the proposed schemes. We verified our proposed schemes on the backends of AVISPA such as OFMC and CLAtSe. The role establishment and validation results are shown in the following figures.

5.1. The Role of HLPSL Codes

In this subsection, we explain the role of the ECUA, GECU, and ECUB in our proposed scheme. The ECUA requests the key distribution process with the GECU and exchanges the session key materials with the ECUB. The brief HLPSL code for the ECUA role is shown in Figure 8.

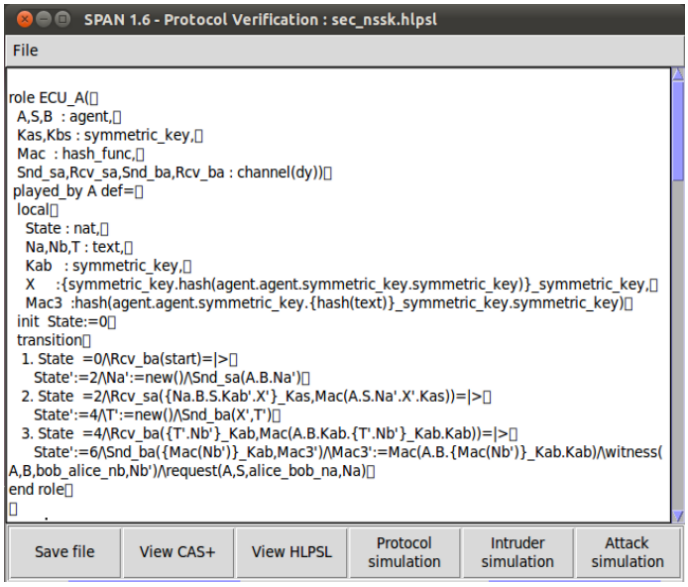


Figure 8. The ECUA role HLPSL code.

Then, we present the role of the GECU in our proposed scheme. The GECU receives the key distribution request from the ECUA and responds to the messages with an encrypted secret seed for session key derivation and a sealed credential encrypted by a distinct long-term key with the ECUB. The HLPSL code for the GECU role is shown in Figure 9.

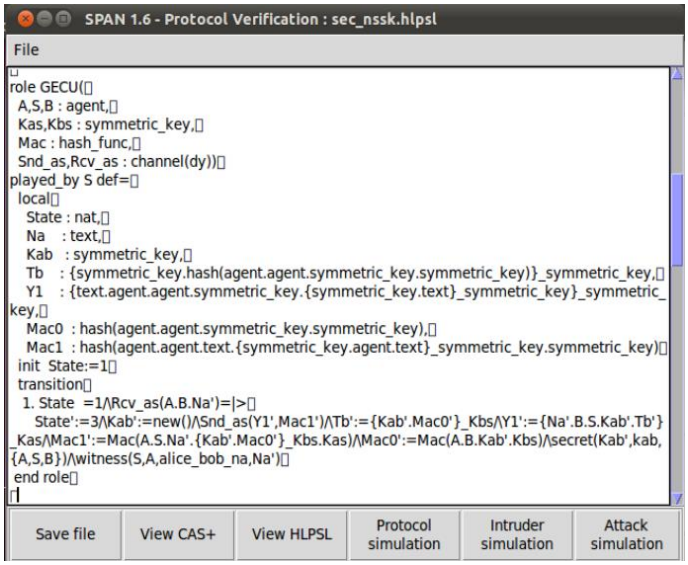


Figure 9. The GECU role HLPSL code.

Finally, we explain the role of the ECUB in our proposed scheme. The ECUB receives the sealed credential containing the secret seed with the valid signature and timestamp generated by the ECUA. It verifies whether the MACs, signature, and timestamp are valid or not for the session key confirmation. The brief HLPSL code for the ECUB role is shown in Figure 10.

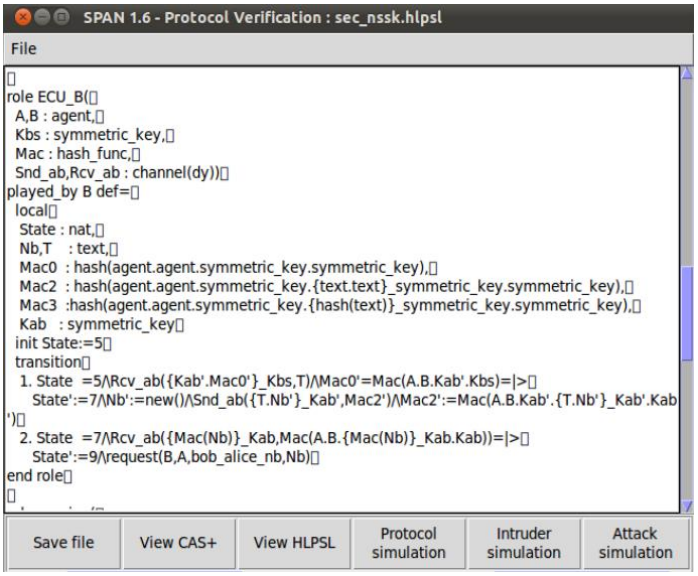


Figure 10. The ECUB role HLPSL code.

5.2. The Environment and Verification Goals

The HLPSL code for proposed scheme verification and environment definition is shown in Figure 11. The goals defined for secret seed *kab*'s secrecy and authentication on the random numbers generated by the ECUA and ECUB are listed. In addition, the intruder knowledge and session compositions are also defined in the role environment. If these specified goals are fulfilled, then the proposed scheme's security can be proven.

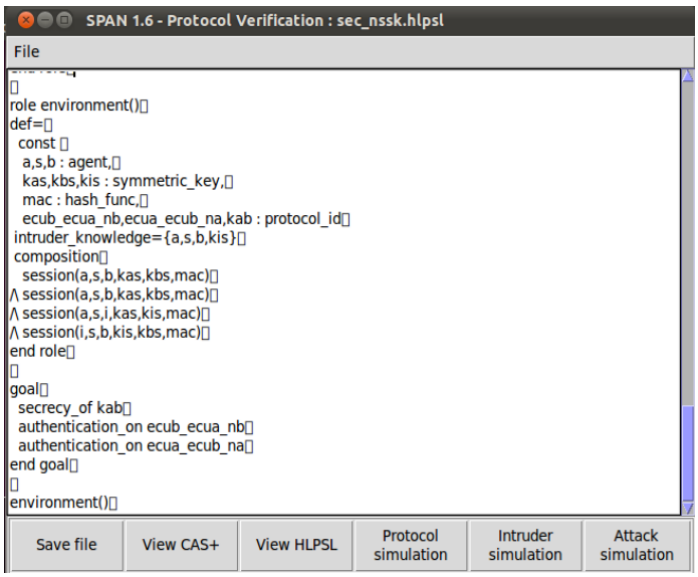
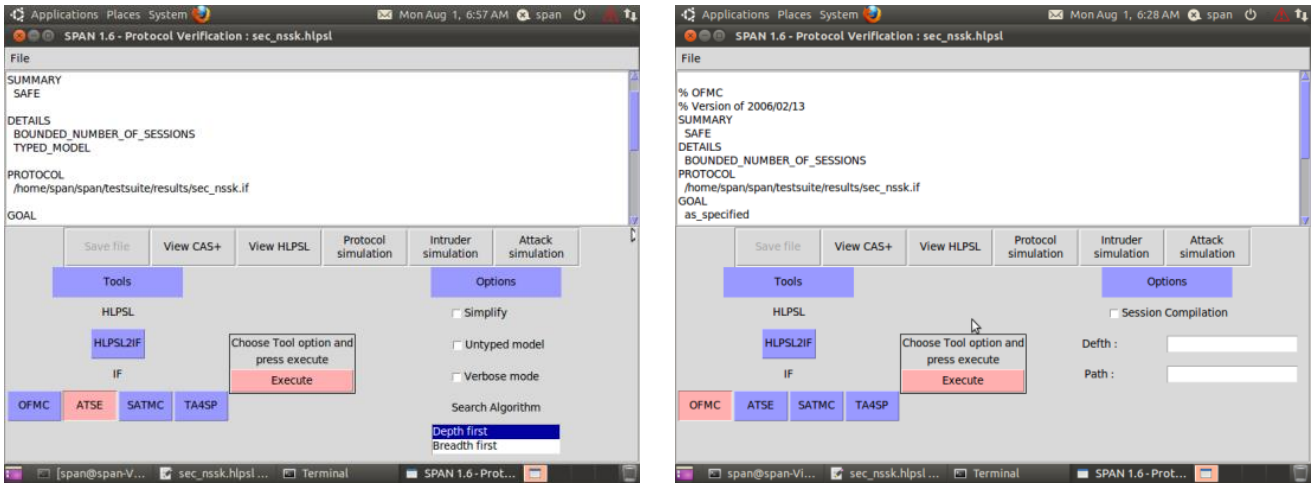


Figure 11. The environment of HLPSL code.

5.3. Simulation Result in AVISPA

We present the simulation results according to the HLPSL code. We applied HLPSL code to the following two protocols: OFMC and ATSE.

(1) ATSE protocol result: The simulation results of our proposed scheme based on the ATSE protocol are shown in Figure 12a, showing that our scheme is safe as it prevents the key sharing process from all kinds of common stealing session key and payload-based attacks.



(**a**) (**b**)

Figure 12. Results of the proposed scheme: (**a**) on the basis of the ATSE protocol; (**b**) on the basis of the OFMC protocol.

(2) OFMC protocol result: The simulation results of our proposed scheme on the basis of the OFMC protocol are shown in Figure 12b, which shows the safety of our proposed scheme and it is efficiency in resisting all kinds of the common stealing session key and payload-based attacks.

5.4. Simulation Result in Tamarin Prover

In this section, we use Tamarin Prover to verify the semantic secure attributes of our proposed protocol. We have described the tamarin-prover in our previous work [26]. In order to prove the semantic security of our proposed scheme, we need to verify the lemmas for security properties of initial SKDP as *Executability*, *SessionKey_secrecy*, *Injectiveagreement_ECUA*, and *ECUA_KKS*, which are listed in Table 4.

Table 4. The initial SKDP's lemmas in Tamarin Prover.

The Lemmas of Initial SKDP
lemma Executability: exists-trace "Ex ECUA ECUB RB EK AK #i #j #k. Commit_strongB(Ri)@#i & Commit_strongA (Ri)@#j & #j<#i & Running_strongA(Ri) @#j & Running_strongB(Ri)@#k & #k<#j & not(Ex #r1. Reveal(ECUA)@#r1) & not(Ex #r2. Reveal(ECUB)@#r2)"
lemma SessionKey_secrecy: "not(Ex ECUA ECUB EK AK #i. Secret(ECUA,ECUB,EK,AK)@#i & (Ex#r. K(<EK,AK>)#j)& not(Ex #j. Reveal(ECUA) @ #j & not(Ex #j. Reveal(ECUB) @ #j)"
lemma Injectiveagreement_ECUA: "(All Ri #i. Commit_strongA(Ri)@ #i ==> (Ex #j. Running_strongA(Ri) @ #j & #j < #i) & (not (Ex #j. Commit_strongA(Ri) @ #j & not (#i = #j))))"
lemma ECUA_KKS: all-traces "All ECUA EK AK #i #m #n. Secret(ECUA,EK,AK) @ i & Honest(ECUA)@ m &Sesreveal(ECUA) @ n & #i<#n ==> not(Ex EK AK #r.K(<EK,AK>)@r)"

The *SessionKey_secrecy* denotes that the session key pair <EK, AK> is secret if it is generated by honest parties that are not revealed to the adversary; then, the lemma ensures the session key secrecy. The *Injectiveagreement_ECUA* denotes the verification of the injective agreement property of the protocol. The lemma describes that for any committed action of ECUA at the time *i*, it must have a prior receiver awaiting for the committed variable. The lemma states that for a committed variable R*i* at the time *i*, there exists a running Ri at *j* [26], where *j* is earlier than *i*, and *j* is not identical to *i* at any time. The *ECUA_KKS* lemma denotes that the revealed session keys EK and AK of the honest ECUA are secret if the adversary does not know about them. In addition, the adversary cannot infer them by utilizing the knowledge of the past session keys, which are irrelevant with EK and AK. Thus, the lemma is relevant to the known key secrecy of the initial SKDP.

We verified the same security lemmas of the second SKDP as initial SKDP; the differences between the two protocols were very small, so we omitted the derivation of the second SKDP. We list four lemmas about the semantic security issues that we are concerned about. The executable lemma verifies the executability of the protocol. The injective agreement lemma confirms the two parties exchanging their messages sequentially within an injective mapping. The session key secrecy lemma ensures that the knowledge of the session key or other secrets is not compromised by adversaries. The known key secrecy lemma ensures that the adversary cannot derive the current session key by using past session keys.

Table 5 shows the verification results generated by the Tamarin Prover for the proposed protocols. It can be seen that all the protocols passed the semantic security verification, and thus the proposed key distribution protocol fit the security goals for resisting the common session key stealing attacks.

Table 5. Verified lemma for the initial SKDP and second SKDP.

	Executable	**Injective Agreement**	**Session Key Secrecy**	**Known Key Secrecy**
Initial SKDP	Yes	Yes	Yes	Yes
Second SKDP	Yes	Yes	Yes	Yes

6. Performance

As described in this section, we conducted simulation experiments with embedded devices to evaluate the performance of the protocol suite. We used the LANCHXL-TMS57004 evaluation kit to simulate ECU_A and ECU_B by programming the CAN bus application based on the FreeRTOS system. At the same time, we used Raspberry Pi 4B with MCP2515 to simulate the gateway device and used *python-can* to develop a test application for establishing a key distribution protocol between the ECU_A and ECU_B. The key distribution protocol used the ISO-TP protocol to realize the data communication between the ECUs and the gateway. Then, we used the CAN analyzer to capture and analyze the data frames sent on the bus to evaluate the time overhead in the execution of the protocol suite.

We used the 500 kbps CAN bus speed to perform multiple rounds of performance testing and obtained the results shown in Figure 13. We used AES as the encryption algorithm and SHA1 as the HMAC algorithm by default for the proposed initial SKDP and second SKDP. For the data signature and verification part, the ECDSA and the ED25519 [27] were used for comparison. In addition, we measured the computation time of calculating HMAC in SHA-256, encrypting and decrypting messages in AES-128 or RSA-2048, and signing and verifying with ECDSA-256 on the test suite. Moreover, we compared the actual computation complexity and the number of storage message bits of our proposed protocol with the other key updating protocols. We used the following notations shown in Table 6 to present the approximate computation time of the basic cryptographic primitive operations in the protocols. The comparison results are shown in Table 7.

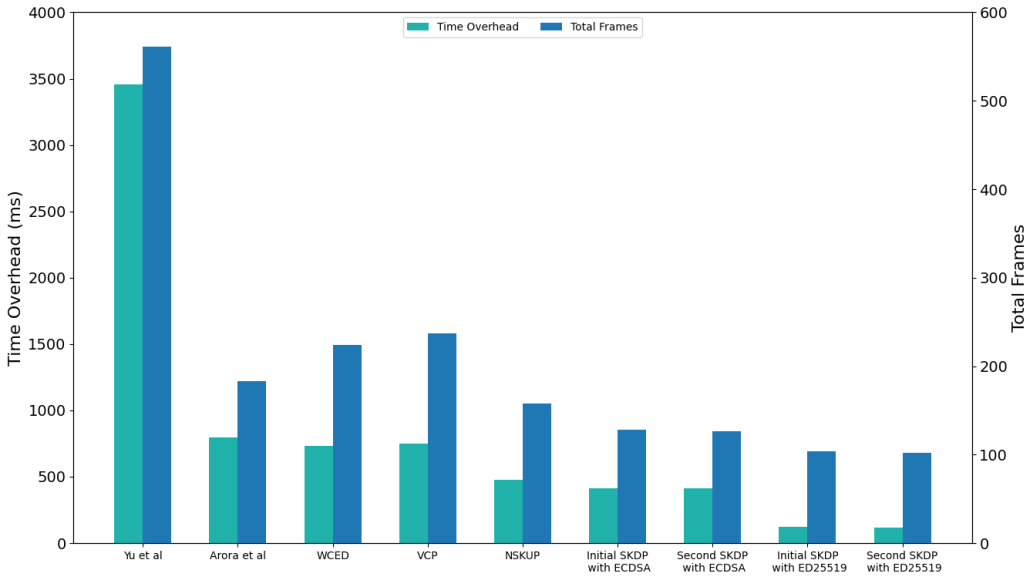


Figure 13. Comparison of the protocol executed time and total frames with the other protocols.

Table 6. The notations of the cryptographic primitive operation computation time.

Notation	Description	Computation Time (ms)
T_H	The time to perform an HMAC or hash operation in SHA-256	1.21
$T_{E/D}$	The time for symmetric encryption or decryption operation in AES-256	0.35
T_S	The time to sign the message in ECDSA-256	64
T_{C_v}/T_{S_v}	The time of verifying a certificate/signature in ECDSA-256	96
T_{E_p}	The time for asymmetric encryption operation in RSA	230
T_{D_p}	The time for asymmetric decryption operation in RSA	184
T_{C_p}	The time to perform the cyclic group multiplication	265

Table 7. Comparison of computation complexity and storage bits with other protocols.

	WCED [1]	**VCP [9]**	**Initial SKDP**	**Yu et al. [23]**	**Arora et al. [25]**
EDEV	$T_S+T_{C_v}+T_{S_v}$	$T_S+T_{C_v}+T_{S_v}+T_H$	$6T_H+6T_{E/D}+T_S+T_{C_v}+T_{S_v}$	$4T_{E_p}+4T_{D_p}+3T_{E/D}$	$8T_{E/D}+2T_{C_p}$
GECU	$T_S+T_{C_v}+T_{S_v}+T_{E/D}$	$T_S+T_{C_v}+T_{S_v}+T_{E/D}$	$2T_H+2T_{E/D}$	$3T_{E_p}+3T_{D_p}$	$6T_{E/D}$
Computation complexity	$2T_S+2T_{C_v}+2T_{S_v}+T_{E/D} \approx 512$ ms	$2T_S+2T_{C_v}+2T_{S_v}+T_{E/D}+T_H \approx 513$ ms	$8T_H+8T_{E/D}+T_S+T_{C_v}+T_{S_v} \approx 266$ ms	$7T_{E_p}+7T_{D_p}+3T_{E/D} \approx 2896$ ms	$14T_{E/D}+2T_{C_p} \approx 515$ ms
	WCED [1]	NSKUP [9]	Second SKDP	Yu et al. [23]	Arora et al. [25]
EDEV	-	$T_{S_v}+T_S$	$7T_H+6T_{E/D}+T_S+T_{C_v}+T_{S_v}$	-	-
GECU	-	$T_S+T_{S_v}$	$3T_H+2T_{E/D}$	-	-
Computation complexity	-	$2T_S+2T_{S_v} \approx 320$ ms	$10T_H+8T_{E/D}+T_S+T_{C_v}+T_{S_v} \approx 269$ ms	-	-
Total computation complexity	$2\times(2T_S+2T_{C_v}+2T_{S_v}+T_{E/D}) \approx 1024$ ms	$4T_S+2T_{C_v}+4T_{S_v}+T_{E/D}+T_H \approx 833$ ms	$18T_H+14T_{E/D}+2T_S+2T_{C_v}+2T_{S_v} \approx 535$ ms	$14T_{E_p}+14T_{D_p}+6T_{E/D} \approx 5792$ ms	$28T_{E/D}+4T_{C_p} \approx 1030$ ms
Number of storage message bits	3578	3164	1890	8976	2528

The results showed that the average execution time of the initial SKDP and second SKDP execution by the ED25519 signature algorithm were about 121 milliseconds and 117 milliseconds, respectively, while the average execution times of the protocol obtained by applying the ECDSA algorithm were about 414 milliseconds and 413 milliseconds, respectively. Therefore, we recommend using the ED25519 algorithm as the lightweight signature and verification algorithm for the proposed key distribution protocol.

At the same time, we compared the proposed key distribution protocol suite with the other protocols under the same experimental environment. The protocol execution time of

Yu et al. [22] was about 3457 milliseconds, and a total of 561 data frames were sent on the CAN bus. Arora and colleagues' [23] protocol execution time was 797 ms with 183 data frames sent on the bus. WCED's [1] execution time was 735 ms with 224 data frames sent, while VCP's [9] and NSKUP's execution times were 751 ms with 237 data frames and 479 ms with 158 data frames, respectively. Our proposed protocol suite initial SKDP with ED25519 sent 114 data frames, and second SKDP with ED25519 sent 112 data frames (initial SKDP with ECDSA sent 128 frames, second SKDP with ECDSA sent 126 frames). Thus, adopting the lightweight ED25519 as the signature verification algorithm in our proposed protocols more accurately fit the real-time constraints of the in-vehicle CAN network.

7. Conclusions

In this work, by considering the common payload-based attacks in actual in-vehicle network scenarios, we proposed a provable secure key distribution protocol based on NSSK for the in-vehicle network to prevent these malicious impacts. First, we applied the mechanisms of message authentication and digital signature to fix the defect of the original NSSK's session key distribution process in resisting the known Denning–Sacco attack. Next, we present a formal security analysis of the new protocol in a random oracle model. The proof shows the new protocol is provably secure and can resist the attacks, namely, replay, impersonation, and MITM. Then, we verified the security attributes of the protocol's semantics by using the AVISPA and Tamarin Prover, with the results ensuring that the protocol met the security requirements for sharing keys, such as session key secrecy, injective agreement, and known key secrecy. Finally, we provided a comparison of our protocol with other key distribution protocols in CAN bus communication to evaluate the performance of the proposed protocol in actual scenarios. Compared with the other schemes of adopting the group-key-based approach, our scheme is more secure and feasible in its application in the session key distribution for different independent domains of an in-vehicle network.

In future works, we will take the security issue of the long-term shared keys into account and design the lightweight key exchange protocol suite for the in-vehicle CAN network to enhance the resistance against the compromise of long-term shared keys.

Author Contributions: Methodology, L.Y.; software, Z.W.; validation, L.Y., J.X. and C.W.; formal analysis, C.W.; writing—original draft preparation, L.Y.; writing—review and editing, L.Y.; supervision, J.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the National Natural Science Foundation of China under grants 61872069, 62072090 and 62173101, and in part by the Fundamental Research Funds for the Central Universities under grant N2017012 and N2217009.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. 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. [CrossRef]
2. Wang, Q.; Sawhney, S. VeCure: A practical security framework to protect the CAN bus of vehicles. In Proceedings of the 2014 International Conference on the Internet of Things (IoT), Cambridge, MA, USA, 6–8 October 2014; pp. 13–18.
3. Nürnberger, S.; Rossow, C. vatiCAN–vetted, authenticated CAN bus. In Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems, Santa Barbara, CA, USA, 17–19 August 2016; pp. 106–124.
4. Radu, A.I.; Garcia, F.D. LeiA: A lightweight authentication protocol for CAN. In Proceedings of the European Symposium on Research in Computer Security, Heraklion, Greece, 28–30 September 2016; pp. 283–300.
5. Kang, K.D.; Baek, Y.; Lee, S.; Son, S.H. An Attack-Resilient Source Authentication Protocol in Controller Area Network. In Proceedings of the 2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), Beijing, China, 18–19 May 2017; pp. 109–118.

6. Bella, G.; Biondi, P.; Costantino, G.; Matteucci, I. TOUCAN: A protocol to secure controller area network. In Proceedings of the ACM Workshop on Automotive Cybersecurity, Richardson, TX, USA, 27 March 2019; pp. 3–8.
7. Mun, H.; Han, K.; Lee, D.H. Ensuring safety and security in CAN-based automotive embedded systems: A combination of design optimization and secure communication. *IEEE Trans. Veh. Technol.* **2020**, *69*, 7078–7091. [CrossRef]
8. Youn, T.-Y.; Lee, Y.; Woo, S. Practical Sender Authentication Scheme for In-Vehicle CAN with Efficient Key Management. *IEEE Access* **2020**, *8*, 86836–86849. [CrossRef]
9. Palaniswamy, B.; Camtepe, S.; Foo, E.; Pieprzyk, J. An efficient authentication scheme for intra-vehicular controller area network. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3107–3122. [CrossRef]
10. Schmandt, J.; Sherman, A.T.; Banerjee, N. Mini-MAC: Raising the bar for vehicular security with a lightweight message authentication protocol. *Veh. Commun.* **2017**, *9*, 188–196. [CrossRef]
11. Groza, B.; Murvay, S. Efficient protocols for secure broadcast in controller area networks. *IEEE Trans. Ind. Inform.* **2013**, *9*, 2034–2042. [CrossRef]
12. Kurachi, R.; Matsubara, Y.; Takada, H.; Adachi, N.; Miyashita, Y.; Horihata, S. CaCAN-centralized authentication system in CAN(controller area network). In Proceedings of the 14th International Conference on Embedded Security in Cars (ESCAR 2014), Hamburg, Germany, 19 November 2014; pp. 1–9.
13. Wang, E.; Xu, W.; Sastry, S.; Liu, S.; Zeng, K. Hardware module-based message authentication in intra-vehicle networks. In Proceedings of the 2017 ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS), Pittsburgh, PA, USA, 18–20 April 2017; pp. 207–216.
14. Jo, H.J.; Kim, J.H.; Choi, H.-Y.; Choi, W.; Lee, D.H.; Lee, I. Mauth-can: Masquerade-attack-proof authentication for in-vehicle networks. *IEEE Trans. Veh. Technol.* **2020**, *69*, 2204–2218. [CrossRef]
15. Wu, Z.; Zhao, J.; Zhu, Y.; Lu, K.; Shi, F. Research on in-vehicle key management system under upcoming vehicle network architecture. *Electronics* **2019**, *8*, 1026. [CrossRef]
16. Püllen, D.; Anagnostopoulos, N.A.; Arul, T.; Katzenbeisser, S. Using implicit certification to efficiently establish authenticated group keys for in-vehicle networks. In Proceedings of the 2019 IEEE Vehicular Networking Conference (VNC), Los Angeles, CA, USA, 4–6 December 2019; pp. 1–8.
17. Pan, Q.; Tan, J. A dynamic key generation scheme based on CAN bus. In Proceedings of the 2019 10th International Conference on Information Technology in Medicine and Education (ITME), Qingdao, China, 23–25 August 2019; pp. 564–569.
18. Jain, S.; Guajardo, J. Physical layer group key agreement for automotive controller area networks. In Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems, Santa Barbara, CA, USA, 17–19 August 2016; pp. 85–105.
19. King, Z. Investigating and securing communications in the Controller Area Network (CAN). In Proceedings of the 2017 International Conference on Computing, Networking and Communications (ICNC), Silicon Valley, CA, USA, 26–29 January 2017; pp. 814–818.
20. 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.
21. Needham, R.M.; Schroeder, M.D. Using encryption for authentication in large networks of computers. *Commun. ACM* **1978**, *21*, 993–999. [CrossRef]
22. Jin-Gang, Y.; Zhi-Gang, Z. An improved NSSK authentication protocol and its formal analysis. In Proceedings of the 2017 10th International Conference on Intelligent Computation Technology and Automation (ICICTA), Changsha, China, 9–10 October 2017; pp. 132–136.
23. Arora, S.; Hussain, M. Secure session key sharing using symmetric key cryptography. In Proceedings of the 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), South Campus, Bangalore, 19–22 September 2018; pp. 850–855.
24. Checkoway, S.; Mccoy, D.; Anderson, D.; Kantor, B.; Kohno, T. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In Proceedings of the 20th USENIX Security Symposium (USENIX Security 11), San Francisco, CA, USA, 8–12 August 2011; pp. 447–462.
25. Miller, C.; Valasek, C. *A Survey of Remote Automotive Attack Surfaces*; Black Hat: Isanti, MN, USA, 2014; p. 94.
26. Long, Y.; Xu, J.; Wang, C.; Wang, Z. An Improved Needham-Schroeder Session Key Distribution Protocol for In-Vehicle CAN Network. In Proceedings of the International Conference on Security and Privacy in New Computing Environments, Xi'an, China, 27–28 November 2022; pp. 35–52.
27. Bernstein, D.J.; Duif, N.; Lange, T.; Schwabe, P.; Yang, B.Y. High-speed high-security signatures. *J. Cryptogr. Eng.* **2012**, *2*, 77–89. [CrossRef]