symmetry

MDPI

Article

Research on a Vehicle Authentication and Key Transmission Protocol Based on CPN

Lu Zheng and Tao Feng *

School of Computer and Communication, Lanzhou University of Technology, Lanzhou 730050, China
* Correspondence: fengt@lut.edu.cn

Abstract: With the rapid development of the Internet of Vehicles, the increase in vehicle functional requirements has led to the continuous increase in complex electronic systems, and the in-vehicle network is extremely vulnerable to network attacks. The controller area network (CAN) bus is the most representative in-vehicle bus technology in intra-vehicular networks (IVNs) for its flexibility. Although the current framework to protect the safety of CAN has been proposed, the safety communication mechanism between electronic control units (ECUs) in the vehicle network is still weak. A large number of communication protocols focus on the addition of safety mechanisms, and there is a lack of general protocol formal modeling and security assessment. In addition, many protocols are designed without considering key updates and transmission, ECUs maintenance, etc. In this work, we propose an efficient in-vehicle authentication and key transmission scheme. This scheme is a certificateless framework based on identity cryptography, which can not only ensure the security of the in-vehicle network but also meet the real-time requirements between ECUs. Moreover, this scheme can reduce the complexity of key management for centralized key generators. To evaluate the security of this scheme, we adopt a protocol model detection method based on the combination of the colored Petri net (CPN) and the Dolev–Yao attack model to formally evaluate the proposed protocol. The evaluation results show that the proposed scheme can effectively prevent three types of man-in-the-middle attacks.

Keywords: CAN protocol; CPN Tools; Dolev–Yao; formal analysis; security assessment

Citation: Zheng, L.; Feng, T. Research on a Vehicle Authentication and Key Transmission Protocol Based on CPN. *Symmetry* **2022**, *14*, 2398. https://doi.org/10.3390/sym14112398

Academic Editors: José Carlos R. Alcantud, Ding Wang, Weizhi Meng and Jian Shen

Received: 8 October 2022
Accepted: 10 November 2022
Published: 13 November 2022

1. Introduction

The rapid integration of vehicles and information technology has led to the rapid rise of connected cars, and cars have moved from being independent and closed to being interconnected and open. The increase in open interfaces brings attack risks to the in-vehicle network [1]. The nodes of electronic control units in the vehicle use different bus protocols for network communication, such as, CAN, FlexRay, LIN, MOST, and other protocols [2]. Among them, the CAN protocol and the Ethernet protocol are the most used, but they lack corresponding security mechanisms [3]. Only simple verification cannot guarantee the security of the in-vehicle network, nor can it adapt to the development of connected cars.

The in-vehicle network security architecture can be divided into four levels according to different functions. Each level has different definitions and requirements for security. The information interaction process between different levels is strictly controlled to meet a series of network security requirements. ECUs at Level 4 have relatively the highest risk among the four levels. Therefore, the security of in-vehicle network communication protocol is extremely important. The in-vehicle network security architecture is shown in Figure 1.

For the security protection of the in-vehicle network, the symmetric cryptosystem combined with the fixed key is used between the ECUs, which is mainly used for symmetric encryption and message authentication code (MAC) mechanisms. Although the symmetric cryptosystem has high computational efficiency, each ECU needs to store a large number of keys and lacks effective key management [4]. At the same time, manufacturers need to

register and maintain all keys, which further increases the difficulty and complexity of key usage. Therefore, such schemes are difficult to implement in real-world scenarios [5].

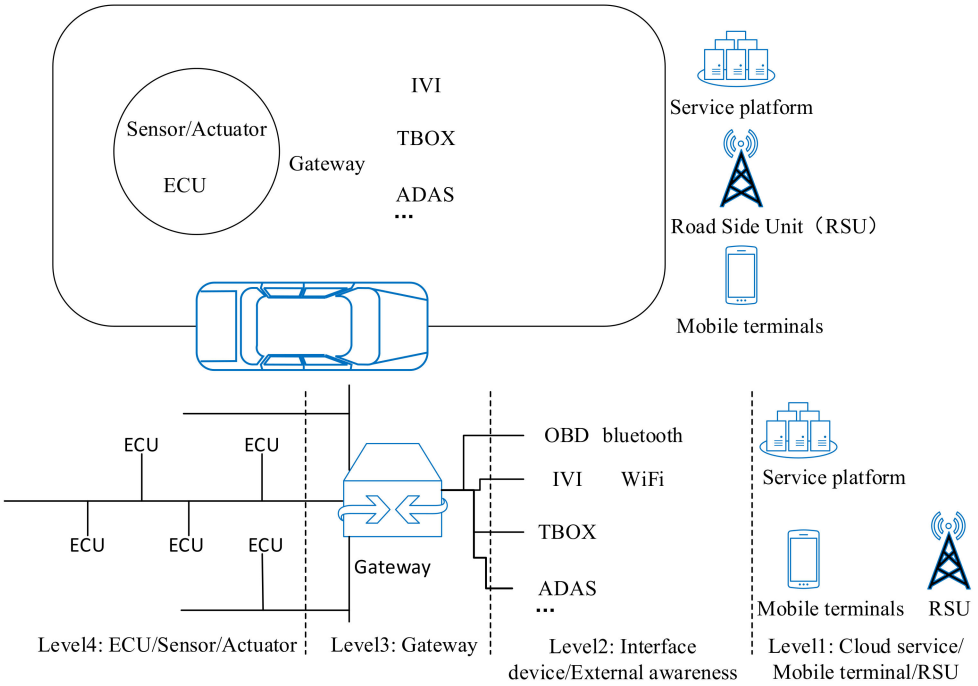

Figure 1. In-vehicle network security architecture.

The public key cryptosystem uses a framework based on digital certificates for authentication. Although digital certificates have a strong authentication mechanism, they require third-party infrastructure. Although key management has extremely high security, the process is relatively complicated, which leads to the high complexity of the security mechanism in the in-vehicle network and is not feasible in real scenarios [6]. To solve the information security problem of the in-vehicle network in the IVN environment [7], many experts and scholars are devoted to the research of secure in-vehicle network architecture. E-safety vehicle intrusion protected applications (EVITA) defines the reference architecture of the in-vehicle network [8], which mainly includes in-vehicle electronic components, such as ECUs, sensors, actuators, etc., the links between components and ECUs, and various software in ECUs. EVITA mainly focuses on research on trust-based hardware security modules (HSM) [9]. The HSM is responsible for performing important cryptographic applications and plays a vital role in the in-vehicle network. Open vehicular secure platform (OVERSEE) builds an open in-vehicle information platform [10], which reduces the risk of vehicle system paralysis caused by malware running through a modular operation method.

2. Related Work

For the specific protocol design in the vehicle system architecture, researchers have also developed different solutions. Koscher et al. [11] completed a comprehensive experimental analysis of the attack surface of vehicle information security and summarized the specific vulnerabilities in CAN bus communication. Attackers can use the OBD interface in the vehicle diagnostic mechanism to launch malicious attacks. Wi-fi, phone synchronization, etc. provide attackers with new opportunities for malicious injection. Szilagyi et al. [12] proposed the use of time-triggered message authentication to improve the efficiency of the protocol, but it lacks consideration in real-time performance. Koopman and Lin et al. [13,14] used HMAC to authenticate CAN message, but it has certain defects in communication overhead. Van et al. [15] proposed CANAuth, which uses HMAC for replay attacks, but still

relies on the assumption of pre-shared keys between ECUs. Furthermore, these protocols above lack an efficient key management mechanism.

Some protocols use shared keys based on ECUs groups. Goza et al. [16] designed mechanisms such as EPSB and Libra-CAN to complete CAN bus authentication. Wang et al. [17] proposed a trust-based key distribution method, but it relies on a fixed group, which reduces the security accordingly.

Woo et al. [18] considered the key management scheme, and proposed the AES lightweight symmetric encryption algorithm for the case of attackers using wireless devices to attack the CAN bus, and verified the efficiency of the algorithm based on ECUs. Mun et al. [19] separated the key functions and used the HMAC mechanism to optimize the ECUs. Palaniswamy et al. [20] proposed a symmetric key-based key management solution to solve the problem of mutual authentication between ECUs. Mundhenk et al. [21] proposed LASAN, a lightweight authentication protocol for secure in-vehicle networks. LASAN uses a hybrid cryptosystem that combines symmetric and asymmetric cryptography for key management. The SM acts as a fully centralized key distribution unit responsible for ECUs authentication and session key generator for each message flow. Groza et al. [22] analyzed the performance of identity-based signature and identity-based key exchange protocols, they studied the classic WolfSSL library in automotive microcontrollers. Han et al. [23] proposed an attribute-based encryption solution to protect the in-vehicle network, which implements key management flexibly. Table 1 provides a comparative analysis of the above articles.

Table 1. Comparison of related works.

	Main Contribution	Drawbacks
Szilagyi et al. [12]	Proposed a time-triggered message authentication protocol	Real-time performance is not considered
Koopman and Lin et al. [13,14]	Proposed authentication protocols based on symmetric encryption	The cost of computation is not considered
Van et al. [15]	Proposed a broadcast authentication protocol based on HMAC	Without key management
Goza et al. [16]	Proposed a lightweight broadcast group authentication protocol	Inefficient key exchange
Wang et al. [17]	Proposed a protocol based on a trust-based key distribution method	Relies on a fixed group, which reduces the security accordingly
Woo et al. [18]	Proposed a protocol based on the AES lightweight symmetric encryption	Inefficient key management
Mun et al. [19]	Proposed a protocol based on the segregation of critical functionalities	Lack receiver authentication and message confidentiality
Palaniswamy et al. [20]	Proposed a symmetric key-based key management protocol	Relies on the assumption of pre-shared keys
Mundhenk et al. [21]	Proposed LASAN, a lightweight authentication protocol	Use a third external party and impractical for large-scale vehicle production
Groza et al. [22]	Proposed an identity-based key exchange protocol	The cost of computation and implementation performance for practical use is not considered
Han et al. [23]	Proposed an attribute-based encryption solution protocol	Isolated ECUs according to their attributes and impractical for vehicle production

To sum up, a secure IVN environment needs to be protected from physical attacks and remote attacks. Physical attacks such as OBD-II interfaces, malicious nodes, etc. Remote attacks through wireless interfaces such as Bluetooth, Wi-Fi, etc. To protect the IVN from attacks, the communication between the ECUs and the gateway should be focused, and a cryptographic protocol with high security should be constructed.

Compared with the existing research results, the main contributions of this paper include three aspects:

1. For the in-vehicle network lack strong security mechanisms, we propose an identity-based certificateless framework that considers device updates and maintenance;
2. According to the specification of our proposed protocol, we use CPN Tools to formally model the proposed protocol and complete the model consistency verification;

3. We employ a protocol model detection method that combines the Colored Petri Nets (CPN) theory and the Dolev–Yao attack model for security evaluation of the proposed protocol.

3. Background

3.1. Vehicle Architecture

The vehicle contains various functional modules, and each module consists of corresponding ECUs [24]. The vehicle architecture is shown in Figure 2. Different bus technologies connect the ECUs. In the future, the demand for automotive functions will continue to increase, and the number of complex electronic systems and ECUs will increase [25]. In particular, how the safety-related electronic system can improve the safety at the communication level of the bus and meet the real-time performance is crucial.

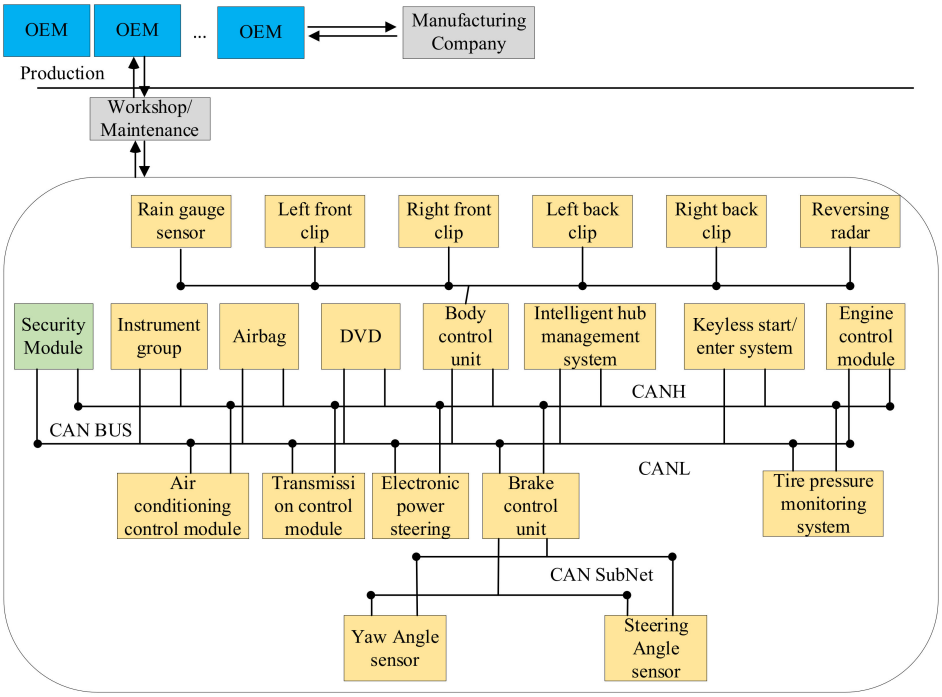

Figure 2. Vehicle architecture.

As the most representative technology in automobiles, the CAN bus will also be a mandatory standard for most vehicles in the future. The CAN bus is internally connected to the various functional modules composed of the ECUs of the vehicle. The CAN data bus is responsible for data transmission and is divided into two data lines: CAN-high (CAN-H) and CAN-low (CAN-L). Data are transmitted through the potential difference between the two data lines [26].

The SM (security module) is the centralized key manager for the ECUs. The SM is the centralized point of key management for each ECU within the vehicle network IVN, and key distribution can be simplified by using a security module (SM) or gateway as the centralized point for key management.

A manufacturing company (MC) mainly assembles and produces based on the needs of customers and the design of the company, etc. In specific applications, it can act as a private key generator (PKG), allowing public key mechanisms to be implemented without relying on third parties such as public key infrastructure (PKI).

3.2. Identity-Based Encryption

The IBE uses the identity identifier to generate the user's public key. As long as the identifier involved in the application system has unique characteristics, it can be directly

used as a public key for cryptographic applications. In this system, users do not need to apply for and exchange certificates and do not need additional management between entity identifiers and public keys. Users can directly use identity information to perform cryptographic operations, thus solving the problem of public key authenticity and fully reflecting the flexibility of public keys, and greatly simplifying the complexity of key system use and management.

Compared with traditional public key infrastructure (PKI), IBE has great flexibility, scalability, and simplicity. A centralized key manager for IBE, called the private key generator (PKG), allows public key mechanisms to be implemented without relying on third parties. Because the system encryption process directly uses the identifier involved in the application as the public key, there is no need to distribute, query the certificate, or apply for a policy certificate, which fully reflects the convenience of encryption operations.

3.3. CPN Tools

A large number of research works have shown that only formal analysis methods can make the verification of security protocols more efficient and credible [27]. BAN logic, string space, and state machine are the methods used in the formal analysis of protocols in the early days. These methods focus on the formal analysis of theorem proving without formal verification of protocol semantics. In recent years, the emergence of powerful formal analysis tools such as ProVerif, Scyther, Tamarin Prover, and CPN Tools can finish formal security verification and semantic analysis for protocols [28].

The CPN Tools modeling tool we adopted is based on the colored Petri net (CPN) theory. CPN is based on the original Petri net theory and effectively combines the standard ML [29], a network system programming language, which belongs to the category of advanced nets. CPN can not only express data sets by color, but also multiple sets of states can be represented by places and tokens, and the ability of data expression has been greatly improved compared with the original Petri net [30]. Before analyzing the functional consistency of the protocol and conducting the security assessment, it is necessary to establish the original specification model of the protocol. The hierarchical model established by CPN can describe the concurrent state and behavior of the system at the same time, which lays a foundation for model consistency and security assessment.

Compare CPN with current mainstream security protocol detection tools:

1. ProVerif is a protocol security detection tool based on logic programming, which can calculate the attack path. However, it only contains one attack path, and the calculated attack path is not comprehensive, which makes it difficult for the modeler to extract attack nodes;
2. Scyther is a protocol model detection tool with high performance and has the function of multiple path analysis. However, it uses the same algorithm to analyze all security protocols. Although the attack path can be found, the error rate is high, and the attack path cannot be comprehensively found for more complex protocols;
3. Tamarin Prover has a powerful state space search capability, which can collect all the state space of the protocol execution and analyze it, but the establishment of the protocol model is not intuitive, and cannot provide the observer with an intuitive description of the protocol execution.

Compared with the above formal analysis tools, CPN Tools require the modeler to dynamically simulate the protocol model according to the protocol process. The visual interface intuitively and vividly shows the steps of the protocol, the modeler can add, delete or modify according to the execution specification of the protocol [31]. CPN Tools can deal with common concurrent systems. The state space generated by the model can clearly indicate the attack state. To query data related to system attack events, Modelers can add necessary termination points and conditions to stop the system process in the model, and establish information collectors in the model. CPN Tools modeling can implement specific modeling and analysis methods for different protocols, so it is usually more effective than other verification tools to verify security protocols.

CPN features state space analysis capabilities that enable incremental syntax checking and code generation while building models, generate and analyze complete state spaces, and use simulation capabilities for state space analysis and model checking to verify system model properties. Its expression is easy to understand and is widely used in many fields.

3.4. Dolev–Yao Attacker Model

The Dolev–Yao attacker model is a model specially designed to verify secure cryptographic protocols [32]. So far, most of the attacker models introduced in the research of security protocols adopt the Dolev–Yao model. The Dolev–Yao model is generalized as a black-box security analysis with five assumptions:

1. The cryptographic system is absolutely secure; the security protocol itself is distinguished from the cryptographic mechanism used within the security protocol. We do not research the security of the specific cryptographic algorithms of the protocol but take the inherent security properties of the protocol as the research goal;
2. The attacker can act as one of the legitimate entities and can communicate with the entities in the protocol;
3. Attackers have powerful computing power and can eavesdrop, block and intercept, replay, and tamper with all information in the system network;
4. Attackers can store, encrypt, decrypt, synthesize, and decompose the intercepted information and participate in the protocol interaction process as a legal entity;
5. If the attacker obtains the corresponding key, the attacker can decrypt the ciphertext.

4. Modeling of Vehicle Safety Protocol Framework

4.1. The Message Flow Model of the Protocol

We design a certificateless framework for identity-based encryption, which includes the initialization phase and the protocol interaction phase.

4.1.1. Initialization Stage

MC generates the identity information ID_{ECUi} and ID_{SM} of SM and n ECUs, and identity information is managed by MC. The validity period Ti indicates the validity of the identity information, which is convenient for the manufacturer to maintain and update the equipment.

MC generates IDs of SM and n ECUs, ID_{ECUi} for ECUs, ID_{SM} for SM:

1. The MC defines the serial number SNi for each ECU, defines the effective period Ti, and generates the identity of the ECUi as $ID_{ECUi} = (Ti||SNi)$;
2. The MC defines the serial number SN_{SM} and generates the identity of the SM as $ID_{SM} = (Ti||SN_{SM})$.

MC generates master key s and system parameters p, defines bilinear mapping and hash function, and lays the foundation for subsequent protocol process phases:

1. MC selects a large prime number q, G1 and G2 are two groups of large prime numbers q, and defines the bilinear map e: $G1 \times G1 \rightarrow G2$;
2. MC selects the random number $s \in Z^*_q$ as the master key, defines a random number generator $P \in G1$, and generates the public key PK = sP of the system;
3. MC defines the hash H_1: $\{0,1\}^* \rightarrow Z^*_q$, H_2: $\{0,1\}^* \times G1 \times G2 \rightarrow Z^*_q$, H_3: $G2 \rightarrow \{0,1\}^n$;
4. MC generates system parameters p = {q, G1, G2, e, n, g, PK, H1, H2}.

MC calculates the private key S_{ECUi} of each ECU and the private key S_{SM} of SM, sends identity information ID_{ECUi}, ID_{SM}, parameter p, and private key S_{ECUi} to each ECU, and sends identity information ID_{SM}, ID_{ECUi}, parameter p, and private key S_{SM} to SM.

1. MC generates the private key for each ECUi as $S_{ECUi} = H_1(ID_{ECUi})^s$;
2. MC generates the private key for SM as $S_{SM} = H_1(ID_{SM})^s$.

Table 2 shows some symbols in the protocol and their specific meanings.

Table 2. Notation representation in the protocol.

Notation	Description
ID	Identity
$Q_{ID_{SM}}$	The public key of SM
$Q_{ID_{ECUi}}$	The public key of ECUi
$S_{ID_{SW}}$	The private key of SM
$S_{ID_{ECUi}}$	The private key of ECUi
K_{sym}	Session key
$H_x()$	Hash function
n_x	Random number
T_x	Timestamp
m_x	Vehicle additional information
s	Master key
PK	System public key
M,	Plaintext
C	Ciphertext

4.1.2. Protocol Process Stage

The vehicle runs this stage to verify the identity of each ECU, mainly for regular mutual verification or maintenance between the ECUs and the SM. Therefore, to effectively resist replay attacks, random numbers and timestamps are added to the design of the protocol. To effectively resist tampering attacks, the Hash function is introduced to ensure the security of the protocol interaction process.

To effectively resist spoofing attacks, ECDSA digital signature technology is introduced to ensure two-way identity authentication in the entity interaction process. ECU and SM use the same system parameters and keep their public and private keys. The sender of the signature calculates the hash value of the message to be sent, generates the signature with the private key, system parameters, and hash value, and sends it to the receiver. The verifier of the signature verifies the message with the public key and system parameters of the signer. The signature algorithm is that the sender computes $h = H(m)$, computes $s = e(hS_{ID}$, U). The verification algorithm is that the receiver computes $signature' = e(h'Q_{ID}$, W).

The protocol flow is represented by the specific message flow. Figure 3 shows the protocol message flow model.

1. SM selects a random number, n0, generates the message $M_1 = (n_0||T_{SM}||m_{SM})$, selects a random number r, generates U and W. SM uses the public key $Q_{ID_{ECUi}}$ to calculate $k_1 = H_3\left[e\left(Q_{ID_{ECUi}}, \mathrm{W}\right)\right]$, generates the ciphertext $C_1 = M_1 \oplus k_1$, calculates the hash value $h_1 = H_2(M_1)$, and uses the private key $S_{ID_{SM}}$ to calculate the signature $V_1 = H_3\left[e\left(h_1 S_{ID_{SM}}, \mathrm{U}\right)\right]$, SM encapsulates the ciphertext C_1, signature V_1, U to message request, and sends it to ECUi.
2. ECUi uses the private key $S_{ID_{ECUi}}$ to calculate ${k_1}' = H_3\left[e\left(S_{ID_{ECUi}}, \mathrm{U}\right)\right]$, calculates ${M_1}' = C_1 \oplus {k_1}'$. ECUi calculates the hash value ${h_1}' = H_2\left({M_1}'\right)$, and uses the public key $Q_{ID_{SM}}$ to verify the signature ${V_1}' = H_3\left[e\left(h_1 Q_{ID_{SM}}, \mathrm{W}\right)\right]$.
3. ECUi selects a random number, n1, and the key Ksym generates the message $M_2 = (n_1||T_{ECUi}||m_{ECUi}||K_{sym})$, uses the public key to calculate $k_2 = H_3\left[e\left(Q_{ID_{SM}}, \mathrm{W}\right)\right]$. ECUi generates the ciphertext $C_2 = M_2 \oplus k_2$, calculates the hash value $h_2 = H_2(M_2)$, and uses the private key $S_{ID_{ECUi}}$ to calculate the signature $V_2 = H_3\left[e\left(h_2 S_{ID_{ECUi}}, \mathrm{U}\right)\right]$. ECUi encapsulates the ciphertext C_2, signature V_2 to message respond and sends it to the SM.
4. SM uses the private key $S_{ID_{SM}}$ to calculate ${k_2}' = H_3\left[e\left(S_{ID_{SM}}, \mathrm{U}\right)\right]$, generates ${M_2}' = C_2 \oplus {k_2}'$. SM calculates the hash value $h_2 = H_2\left({M_2}'\right)$, and uses the public key $Q_{ID_{ECUi}}$ to verify the signature ${V_2}' = H_3\left[e\left({h_2}' Q_{ID_{ECUi}}, \mathrm{W}\right)\right]$.
5. SM selects the message $data$ to be sent, calculates the hash value $H_2(data)$, encrypts it with the key Ksym, generates the message $M_3 = E_{K_{sym}}[n_2||T_S||data||H_2(data)]$, and sends it to ECUi.
6. ECUi decrypts with the key Ksym to get the message $data'$, calculates the hash value $H_2(data')$, and verifies the data.

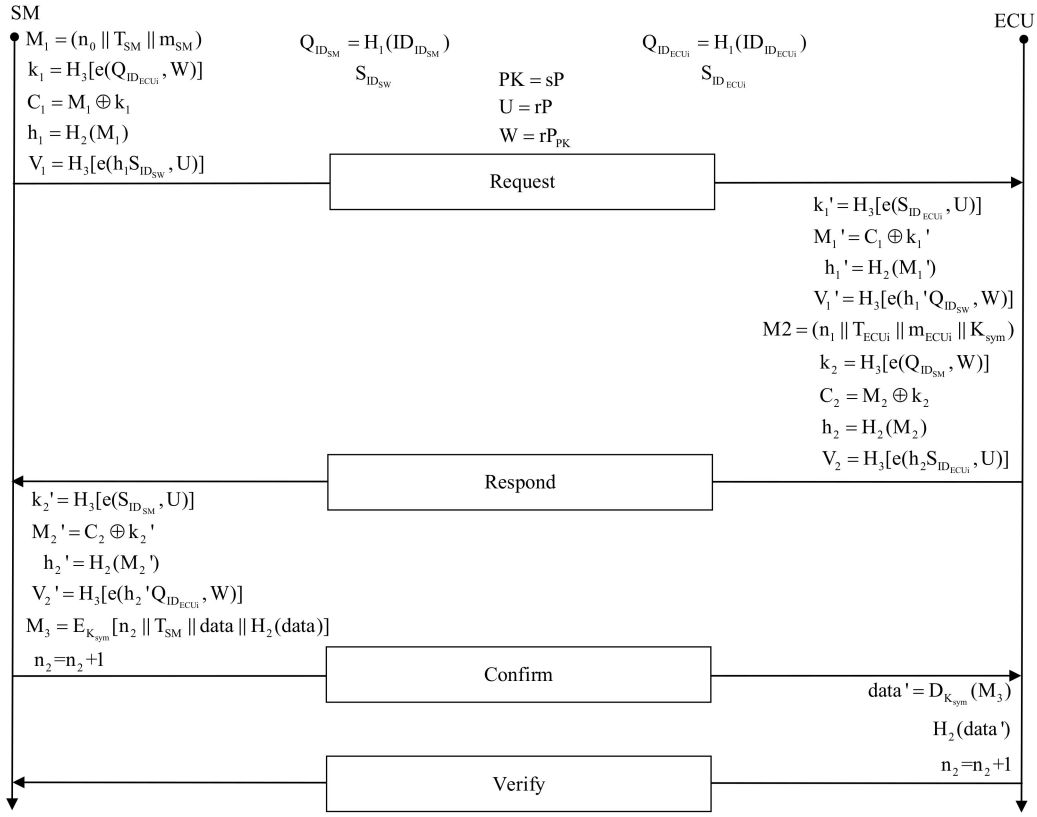

Figure 3. Protocol message flow model.

4.2. Modeling the Protocol Based on CPN

The color set is established for the messages exchanged between the receiver and the sender of the proposed protocol. The specific color set definitions are shown in Table 3. The MESS type indicates the message that combines random numbers, time stamps, and vehicle additional messages. The K type and K′ type represent the parameters used to complete the bilinear algorithm. The C type indicates that the plaintext is encrypted with the key to obtain the ciphertext. The HDATA type indicates that the hash algorithm is used to obtain the hash value of the data. The V type and V′ type indicate that the system parameters are used to complete the digital signature. The tran type indicates the message that combines system parameters, ciphertext, and digital signatures. The mess3 type indicates that the session key is used to encrypt messages with random numbers, timestamps, messages, and their hash values.

Table 3. Color set definitions for the protocol.

Key Elements	Color Set Definition
MESS	colset MESS = record n: R * ts: TS * am: AM
K	colset K = record p: W * hid: HID * hash: HASH
K′	colset K′ = record p: U * hid: SID * hash: HASH
C	colset C = record k: K * m: MESS
HDATA	colset HDATA = record h: HASH * d: DATA
MESS3	colset MESS3 = product R * TS * AM * DATA * HDATA
V	colset V = product HASH * H * U * SID
V′	colset V′ = product HASH * H * W * HID
Tran	colset Tran = record c: C * v: V * u: U

When building a large-scale CPN network, the single-page CPN is not intuitive and cannot clearly describe the functions of each module in the protocol interaction process. We

adopt the idea of modular programming and take advantage of the substitution transition of CPN to split the CPN network structure into multiple sub-blocks, each sub-block is called a subpage, and the split CPN network is called the parent page of the subpage [33]. The top layer of the multi-layer model established in this section is the parent page with substitution transitions that hide the details of the model in the top layer, simplify and present the model in a high-level description, and define the full picture of the model in a broad sense.

The hierarchical model established in this section includes top-level, middle-level, and bottom-level models. The top layer is the abstract description of the overall protocol, the middle layer is the further functional refinement of the top layer, and the bottom layer is the detailed description of the implementation details of the protocol we proposed; it is the specific implementation of the protocol.

The top-level CPN model is shown in Figure 4.

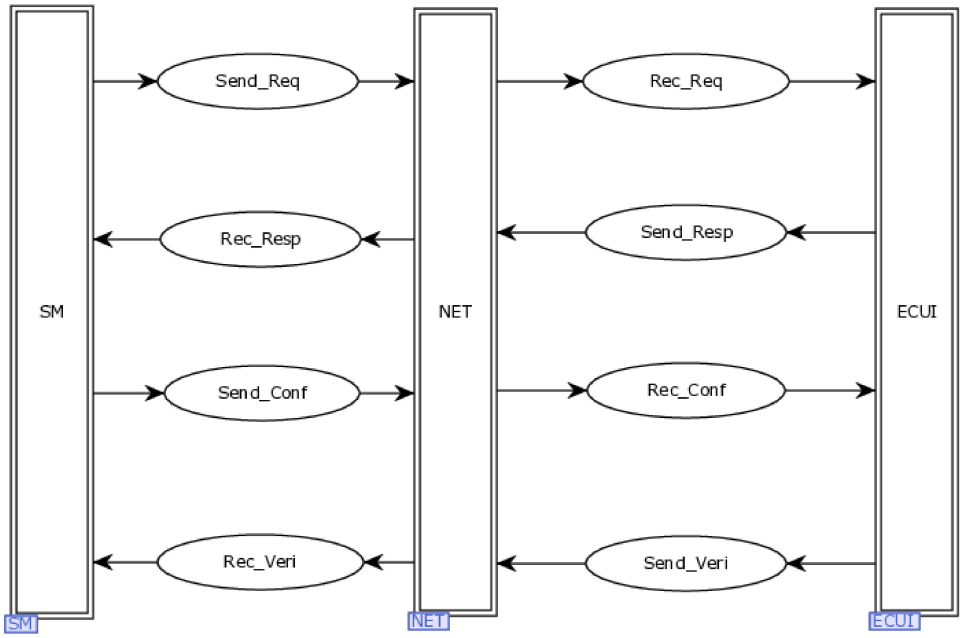

Figure 4. Top-level CPN model.

The top-level model is composed of communication between two parties, SM and ECUi, communication network NET, communication process, and intuitively simulates the conversation process of the whole protocol. The double-layered rectangle in Figure 3 represents the substitution transition, representing the communication between two parties, SM, ECUi, and the communication network NET, respectively, and the ellipse represents the message places; it is used to store the messages in the communication process. Arrows represent the flow of transmission of messages. They are used to transfer messages in the communication flow to the parties.

The middle-level CPN model is shown in Figure 5. The middle-level model includes five substitution transitions and nine places. The substitution transition Connet represents the process that the SM initiates a request to the ECU. Substitute transition Connet' represents the process that the ECU responds to the request initiated by the SM. Substitute transition Confirm means that the SM initiates the verification process to the ECU. Substitute transition Confirm' represents the process that the ECU matches the verification initiated by the SM.

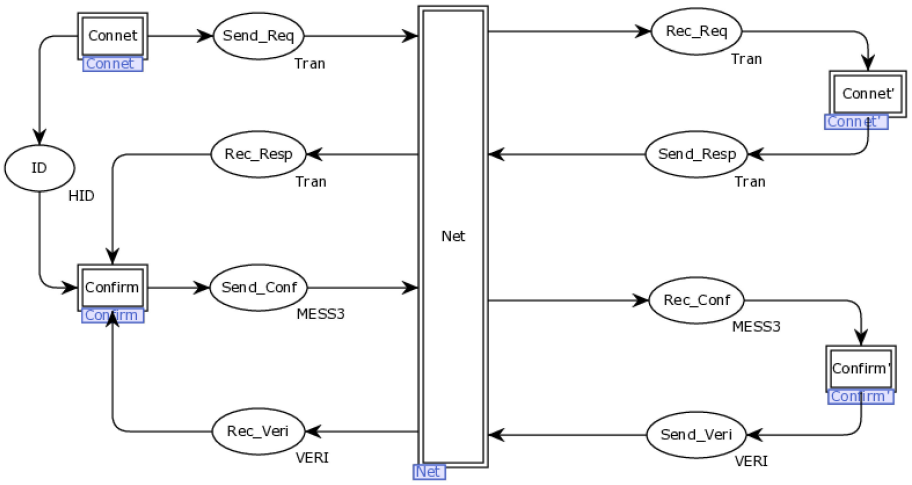

Figure 5. Mid-level CPN model.

The bottom layer CPN model of the protocol consists of four parts. The bottom layer CPN model is established according to the protocol process and the interaction details between the SM and the ECU. It is a detailed description of the protocol details for protocol request responses and verification matches.

Figure 6 shows the internal CPN model of the substitution transition Connect. The transition Mess1 combines the random number N1 generated by SW, the message string AMsm, and the timestamp TSsm into a plaintext message and stores it in place M1. The transition K1 runs the bilinear mapping on the parameter W and the public key of the ECU and runs the Hash operation on the result to combine it into place K1. After the transition, K1M1 runs the XOR operation on the messages K1 and M1. They are combined into a ciphertext message and stored in place C1. The transition HM1 runs the Hash operation on the plaintext message M1 and stores the result in the place hM1. Transition V1 runs the bilinear mapping on the hash value hM1, the private key of SW and parameter U, runs the Hash operation on the result, combines it into the signed message, and stores the result in place V1. Transition Connect combines the parameter U1, the ciphertext C1, and the signature V1 into the request message and sends it to the ECU through the Send_Req of the place.

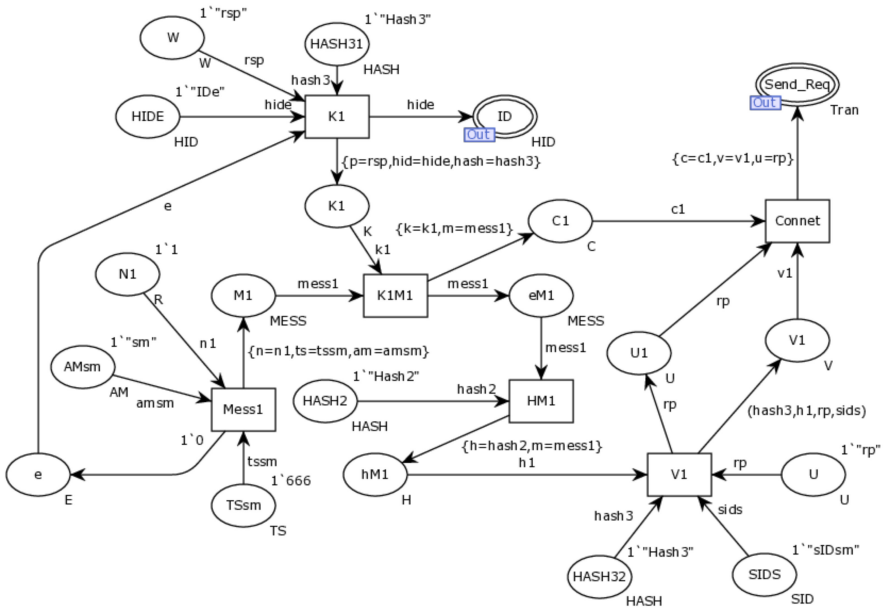

Figure 6. CPN model of substitution transition Connect.

Figure 7 shows the internal CPN model of the substitution transition Connect′. Transition Connect′ first sends the received data to each place. The transition K1′ runs the bilinear mapping on the parameter U and the private key of the ECU, and runs the Hash operation on the result to combine it into the place K1′. The transition M1′ decrypts the received ciphertext messages C1 and K1′, obtains the plaintext information sent by the SM, and stores it in the place M1′. The transition HM1 runs the Hash operation on the plaintext message M1′, and stores the result in the place hM1′. The transition V runs the bilinear mapping of the hash value hM1′, the public key of SW, and the parameter W, runs the Hash operation on the result, combines it into the local hash value, and stores the result in the place V′. The transition compare1 verifies the hash value V and the local hash value V′. If the verification fails, the process is reset through the place Authentication_1. If the verification succeeds, the process will enter the next phase.

The transition Mess2 combines the random number N2 generated by the ECU, the session key Ksym, and the timestamp TSe into a message and stores it in the place M2. The transition K2 runs the bilinear mapping on the public key of the SW and the parameters W, runs the Hash operation on the results, and combines them into place K2. The transition K2 runs the bilinear mapping of the parameters W and the public key of SW and runs the Hash operation on the result to form a place K2. After the transition, K2M2 runs the XOR operation on the messages K2 and M2. They are combined into a ciphertext message and stored in place C2. The transition HM2 runs the Hash operation on the plaintext message M2 and stores the result in the place hM2. Transition V2 runs the bilinear mapping on hash value hM2, the private key of ECU and parameter U, runs Hash operation on the result, combines it into the signed message, and stores the result in place V2. Transition Connect combines the parameter U2, the ciphertext C2, and the signature V2 into the request message and sends it to the SM through the place Send_Resp.

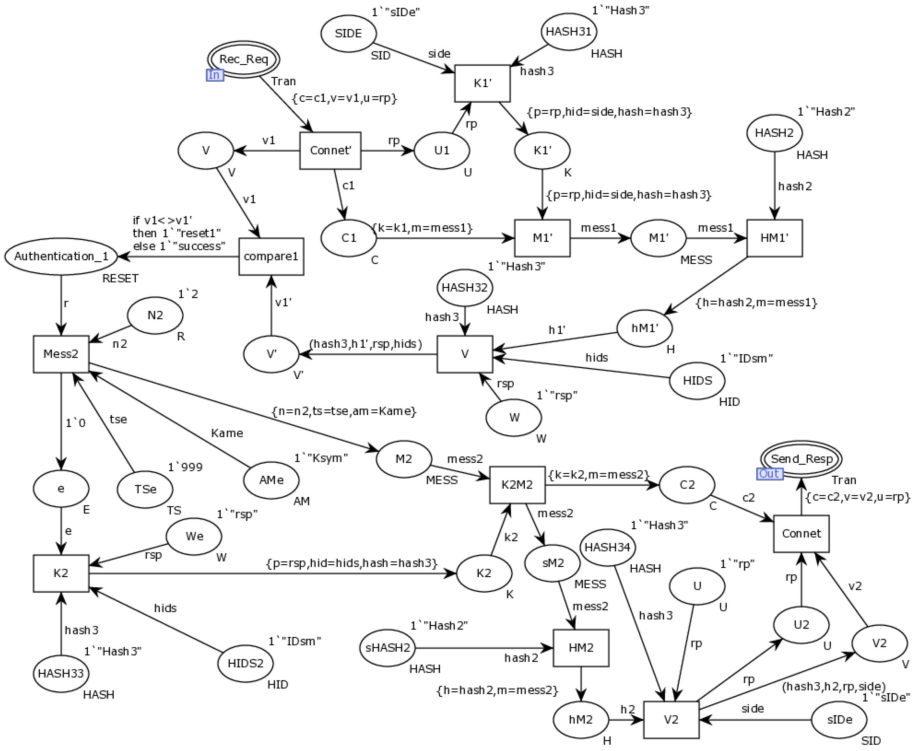

Figure 7. CPN model of substitution transition Connect′.

Figure 8 shows the internal CPN model of the substitution transition Confirm. The transition Confirm first sends the received data to each corresponding place for processing. The transition K2′ runs the bilinear mapping of the parameters U and the private key of SM

and runs the Hash operation on the result to form the place K2′. The transition M2′ decrypts the received ciphertext messages C2 and K2′, obtains the plaintext information sent by the ECU, and stores it in the place M2′. The transition HM2′ runs the Hash operation on the plaintext message M2′ and stores the result in the repository hM2′. The transition V2 runs the bilinear mapping on the hash value hM2′, the public key of the ECU, and the parameter W, runs the Hash operation on the result, combines it into the local hash value, and stores the result in the place V2′. The transition compare2 verifies the hash value V and the local hash value V2′. If the verification fails, it will be reset through the place Authentication_2. If the verification succeeds, the process will enter the next phase.

The transition KM2′ stores the session key sent by the ECU in the place K, and the transition DATA combines the random number N3 generated by the SW, the timestamp TSsm, the data frame data, and the corresponding hash value hdata into the data frame transmission connection message, encrypts it with the session key Ksym and sends it to the ECU through the place Send_Conf.

If the SM receives the verification success message fed back by the ECU, it means that the data frame transmission is completed, and the place Rec_Veri enables the transition of HDATA and starts the next data frame transmission.

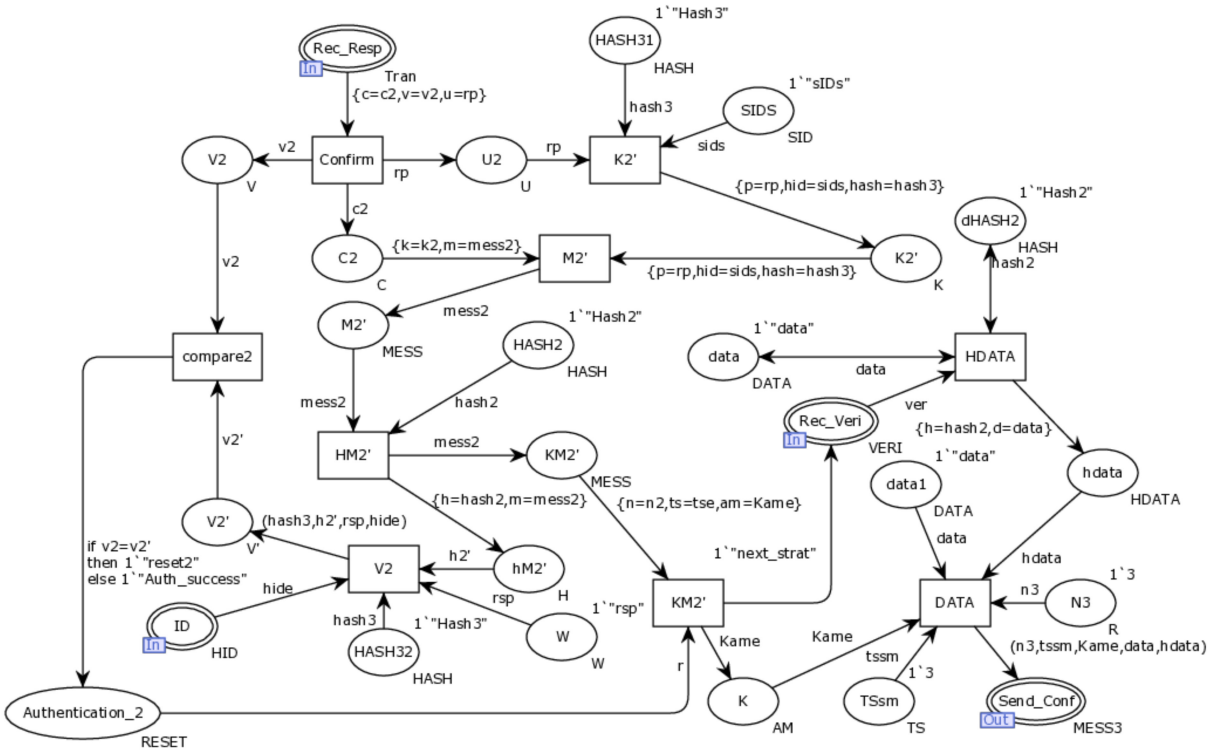

Figure 8. CPN model of substitution transition Confirm.

Figure 9 shows the internal CPN model of the substitution transition Confirm′.

Transition Confirm′ decrypts the received message and stores the decrypted message in the corresponding place. The transition HDATA runs a local hash operation on the data frame data generated by decryption and stores the hash value in the place dHash2. The transition compare3 verifies the hash value hdata and the local hash value *hdata*′ sent by the SM. If the verification fails, the process is reset through the place Reset3. If the verification succeeds, the process will be fed back to the SM through the place Send_Veri.

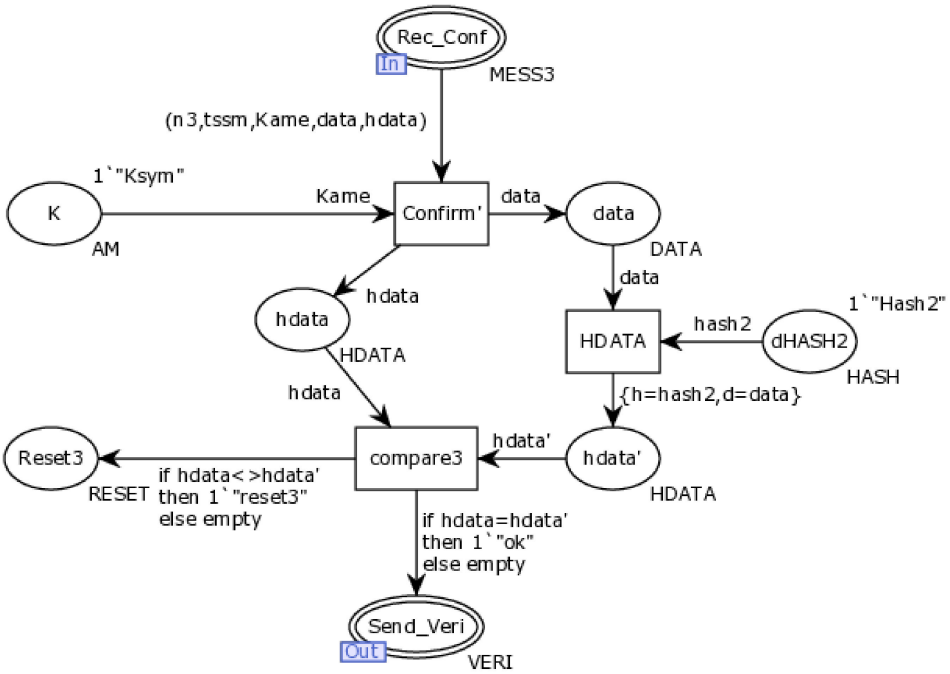

Figure 9. CPN model of substitution transition Confirm′.

Figure 10 shows the internal CPN model of the substitution transition NET. The arrows on the left and right of the transition represent the direction of information transmission in the interaction process of the protocol. The transition trans1 simulates the transmission path that the SM sends the connection request to the ECU, including ciphertext, parameters, and digital signature. Trans2 simulates the process that the ECU receives the connection request message, decrypts to obtain the plaintext, and verifies the signature. Subsequently, trans2 simulates the transmission path for the ECU to reply to the connection request sent to the SM, including ciphertext, parameters with session keys, and digital signatures. trans3 simulates the process that the SM receives the reply message, decrypts to obtain the session key, and verifies the signature. Subsequently, trans3 simulates the transmission path of the SM to send the ciphertext data to the ECU. The ciphertext data encrypts the parameters, the data frame, and the hash value of the data frame with the session key. trans4 simulates the transmission path that the ECU receives the ciphertext information sent by the SM, decrypts and verifies it with the session key, and sends the successful verification feedback message to the SM.

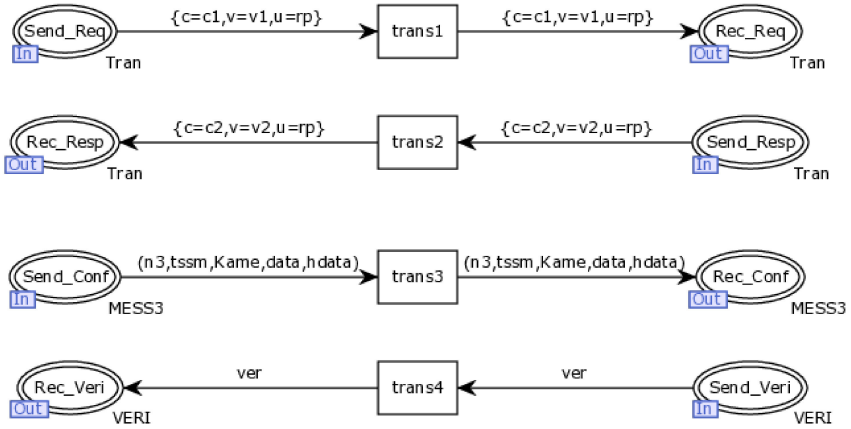

Figure 10. CPN model of substitution transition NET.

4.3. The Attacker Model of the Protocol

According to the strong ability of the attacker in the Dolev–Yao attacker model, various man-in-the-middle attacks such as replay, tampering, and deception are launched into the network channel.

We build the Dolev–Yao attacker model based on CPN to simulate the introduction of a man-in-the-middle attack to the network channel for communication between SW and ECU. Figure 11 shows the Dolev–Yao attacker model of the proposed protocol.

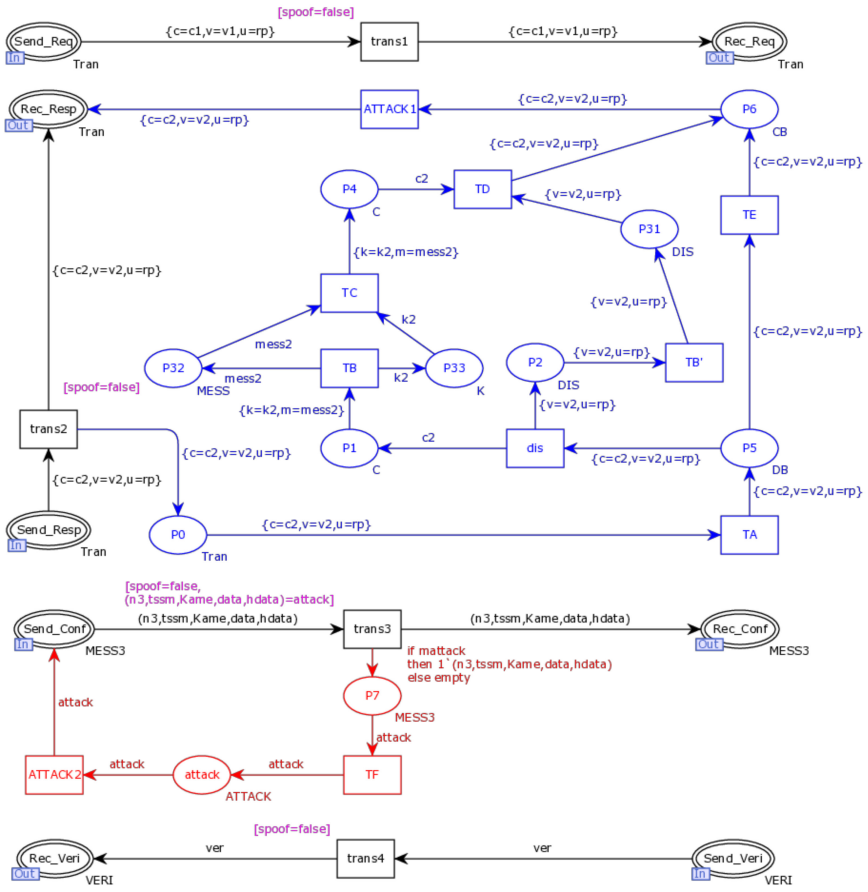

Figure 11. Formal description based on the attacker model.

The replay attack is initiated by the places and transitions of blue type to the Net subpage of the original model. On the trans2 path, the replay attack initiated by the attacker adopts the attacker model based on message splitting and combination. It can reduce the useless information caused by arbitrary splitting by the attacker and avoid the phenomenon of state space explosion caused by a large number of repeated information. This method can guarantee the ability of attackers and effectively reduce the state space. The information transmitted in the network channel is intercepted by place P0 on the replay attack path and transmitted to place P5 through the transition TA for further processing. We define three data types, DB, CB, and DIS, on the path. The data type of place P5 is DB, which is used to store the split and unsplit messages. The transition dis completes the next step of splitting place P5 and stores the corresponding information in place P1 and place P2 storing the DIS data type. Transitions TB and TB' split all kinds of messages into atomic information that cannot be processed and stored in places P31, P32, and P33. The type of place P6 is CB, which is used to store the information that cannot be decrypted in the intercepted information and the information after the synthesis of atomic messages. The transition TE utilizes the Dolev–Yao attacker transition rule to store the messages that cannot be deciphered after interception into the repository P6. Transition TD utilizes Dolev–Yao attacker synthesis rules to synthesize and store atomic

messages in place P6. Transition Attack1 uses place P6 to launch the replay attack to reach the port place of the network channel.

The tampering attack is initiated by the red type of places and transitions to the Net subpage of the original model, and the attacker initiates the tampering attack through the transition Attack2 to reach the port place.

The spoofing attack is initiated by the purple type places and transitions to the Net subpage of the original model. The attacker launches the attacks to all transitions trans1, trans2, trans3, and trans4 on the network transmission path in the original interaction channel.

5. Model Consistency and Security Evaluation of the Protocol

The standard state space report can be generated based on the debugging of the established CPN model. State space reports are used to verify protocol security. Formal properties, such as accuracy, safety, liveness, etc., of the model can be understood through the state space report. The standard state space report provides some basic information about the size of the state space, as well as the standard behavioral properties of the CPN model. In the state space report, the bounded property includes all possible states and how many states a place can hold. A live transition means that for any reachable state, we can always find a sequence of occurrences that includes this transition. The model has a dead transition. Then, they correspond to the part of the model that can never be activated, that is, the end of the state completion.

Table 4 shows the model state space report generated by CPN Tools. The state space report contains the full state of the original model and the attacker-based protocol security evaluation model.

Table 4. State space reports of the original model and the attack model of the protocol.

Type	Original Model	Original Attack Model
State Space Node	42	1063
State Space Arc	40	2867
SCC Graph Node	42	1063
SCC Graph Arc	40	2867
MainState Space Node	0	0
Live Transition Instances	0	0
Dead Transition Instances	0	0
Dead Marking	1	1

The original CPN model state space report reflects the consistency of the model with the protocol standard and the correctness of subsequent protocol model security assessments. The number of all nodes and the number of strongly connected nodes are the same; both are 42. The number of directed arcs in the model is the same as the number of strongly connected arcs; both are 40. This shows that all state nodes of the model are reachable, and there is no cyclic structure. The model runs according to the protocol process and completes request connection and response, data frame transmission, and verification. All the running steps of the model will not trigger the three reset places established in the model, and the phenomenon of model interruption will not occur, so the number of dead transitions is 0. There are no transitions in the model that can always be triggered, so the number of live transitions is 0. The protocol we designed has no uniform termination state, so the number of master state nodes is 0. The original CPN model transmits information according to the protocol specification, and there is a state in which the data transmission is completed. This is consistent with the expected result, so the number of dead nodes is 1.

After running the attacker-based CPN model, its state space report reflects the effectiveness of the introduced attacker model. The model has no infinite loop phenomenon, no master state nodes, and no live transitions. The number of nodes and directed arcs in the state space of the attacker model does not increase significantly compared to the original model because the introduction of the attacker model does not cause the state space to be too large to explode.

There are no dead transitions in the model because there is no situation where a transition cannot be triggered due to the design flaw. The number of dead nodes is 1, which means that there is only one state in which data transmission is completed in the model, and there is no other attack state; that is, it has not been attacked by man-in-the-middle attacks of replay, tampering, and spoofing initiated in the attacker model.

To verify that the attacker model based on CPN eliminates the attack state introduced by Delov-Yao, we run a path search based on the attacker's CPN model and write code to run after the computation of the state space is complete [34]. ListDeadMarkings() is used to list all dead markings identified in the model. In the analysis, we used ListDeadMarkings() to obtain dead markings stated in the report and further simulate them individually. According to the configuration of the model in our work, the expected results will be achieved after the successful implementation of the protocol. The expected marking is a state in which only one initial request is expected to be successfully executed on the result. For a state that has more than one of its initial requests successfully executed, it is an unexpected dead marking. Figure 12 shows the running result of the code for querying the path, and No. 36 is the sequence number of a dead node path included in the attacker model based on CPN. Querying the serial numbers of dead nodes is beneficial to the subsequent state analysis of the attacker model.

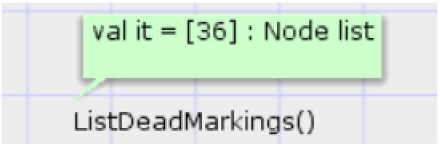

Figure 12. The execution result of querying dead node code.

NodeDescriptor () is used to get the full place state for the corresponding dead marking. We used the NodeDescriptor () within a function to search for all possible states that issued requests from the attacker state. This is to search and return node values. According to the queried dead node path with serial number 36, extract all the contents of the place in the No. 36 path. Figure 13 is an excerpt of the state of the place in path 36 after the state space is calculated. The content contained in the place can reflect the running status of the attack behavior of the Delov-Yao attacker.

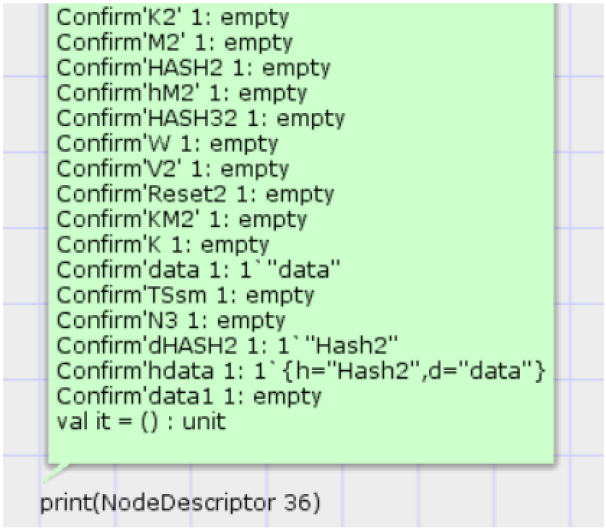

Figure 13. The execution result of querying places the status code in the path.

UpperMultiSet(Mark.page'place n) algorithm can accurately express the corresponding state of the key library on the page in the state space after the execution of the model. It is often used to query the success of some key interaction process in the model.

The place Authentication_2 in the Confirm page indicates the result of the identity authentication between the SM and the ECU. If the Auth_success value is contained in the Authentication_2 place after the Delov-Yao attacker model is added, it indicates that the identity authentication is successful. Otherwise, identity authentication fails. Figure 14 shows the result of the authentication queried by SML. In the authentication phase of the protocol, the security mechanism of digital signature is adopted, which is represented by transition V in the Connect subpage and transition V 'in the Connect' subpage of the CPN model. It results in the attacker's failure to launch spoofing attacks.

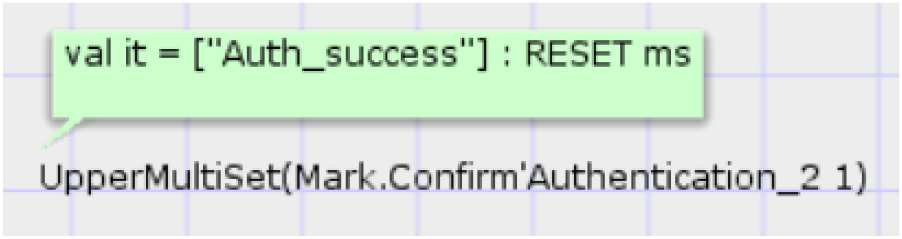

Figure 14. The execution result of querying identity authentication.

The Send_Veri value of the place on the Confirm1 page reflects whether the session key can be used for safe data transmission between the SM and the ECU. Figure 15 shows the result of querying data transmission by SML. The Trans_ok value is contained in the Send_Veri place; it indicates that data are safely transmitted. In the data transmission phase of the protocol, the hash function is used, so the attacker cannot obtain the session key and tamper with the intercepted information.

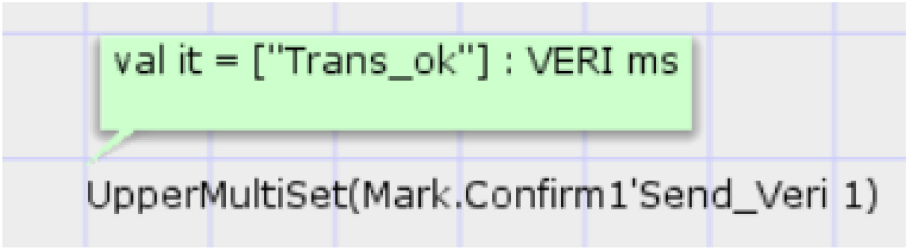

Figure 15. The execution result of querying data transmission.

The Rec_Veri value of the place on the Confirm page indicates whether the next round of data transmission between the SM and ECU will be successful after the last round of data transmission is secured. Figure 16 shows the result of the trigger queried by SML. The Rec_Veri place contains Trans_ok and next_start values, which indicates that the next round of data transmission is successfully triggered. Random numbers are adopted in the CPN model of the protocol. The authentication phase and the data transmission phase of the protocol are not affected after the attacker launches the replay attack.

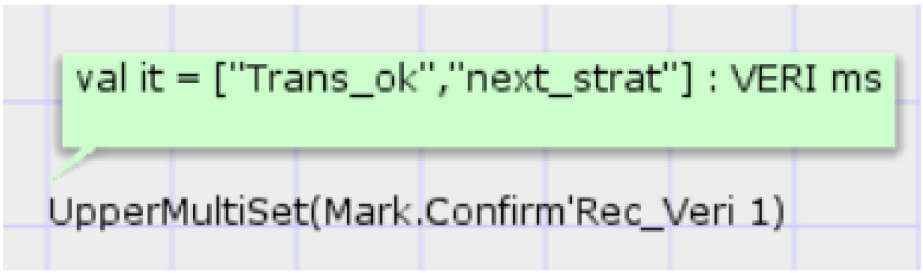

Figure 16. The execution result of querying the next round of data transmission.

By writing an ML program to analyze the transition and the corresponding state of the place, it can be found that the attack state does not exist in the place. Our proposed protocol uses digital signatures in the authentication phase. It can effectively prevent spoofing attacks initiated by attackers. We use the hash function that can effectively resist tampering attacks launched in the model. The random number mechanism in the protocol can well resist the replay attack initiated by the attacker. The mechanism of the random number in the protocol can well resist the replay attack initiated by the attacker.

6. Security Analysis and Performance Analysis

6.1. Security Analysis

The comparison of security properties can reflect the powerful functions satisfied by the protocol. We compare the proposed protocol with the in-vehicle communication protocols proposed by Groza et al. [16], Woo et al. [18], Palaniswamy et al. [20], and Mun et al. [19]. This further shows that our proposed protocol has high security. The comparison of security properties is shown in Table 5.

Table 5. Comparison of protocol security attributes.

Security Attributes	Woo et al. [18]	Groza et al. [16]	Palaniswamy et al. [20]	Mun et al. [19]	Our Scheme
Session key security	No	Yes	Yes	Yes	Yes
Entity authentication	No	No	No	No	Yes
Resistance to reply attack	No	No	Yes	Yes	Yes
Resistance to tampering attack	Yes	Yes	Yes	Yes	Yes
Resistance to spoofing attack	No	No	No	No	Yes
Provable security	No	No	Yes	Yes	Yes
Formal verification	No	No	Yes	No	Yes

Woo et al. [18] proposed a protocol based on symmetric keys that solve the problem of communication between ECUs of in-vehicle networks. It can only achieve verification of the functional correctness of the protocol. However, it does not consider the key exchange and relies on the assumption of pre-shared keys. The security mechanism contained in the protocol can only prevent tampering attacks through hash functions but cannot defend against replay attacks and spoofing attacks.

Libra-CAN, proposed by Groza et al. [16], uses key splitting for sender group communication and MAC mixing for message authentication suitable for CAN bus. The protocol can guarantee the security of the session key. The security mechanism included in the protocol can only prevent tampering attacks through hash functions but cannot defend against replay attacks and spoofing attacks.

Palaniswamy et al. [20] proposed a protocol that includes the key management solution. They use the Tamarin tool to formally verify the security of the protocol and verify that there is no replay attack in the protocol. However, Tamarin cannot show the attacker's status in real time. Although this approach provides efficient computation and key management, the protocol cannot resist spoofing attacks in the data frame transmission phase.

Mun et al. [19] proposed a protocol based on the segregation of critical functionalities, leading to flexible key management. However, due to the lack of formal analysis

of the proposed protocol, the protocol cannot resist spoofing attacks well. In addition, communication cost is not considered in this scheme.

Our proposed framework considers the update and maintenance of the ECUs and reduces the difficulty of key distribution and the complexity of key management. The security mechanism contained in the protocol can ensure a variety of security properties. It can resist replay, tampering, and deceive man-in-the-middle attacks and can ensure the security of entity authentication and session keys. The method of formalized security proof can clearly describe the protocol specification, extract loopholes of the protocol, and ensure that the security mechanism embedded in the protocol does not affect the functionality of the protocol itself.

6.2. Performance Analysis

Computational complexity can reflect the computational and communication efficiency of the protocol. We compare the performance of the proposed framework with the protocols of Groza et al. [16], Woo et al. [18], Palaniswamy et al. [20], and Mun et al. [19].

The comparison results of computational complexity are shown in Table 6.

Table 6. Comparison of protocol computational complexity.

Phase	Woo et al. [18]	Groza et al. [16]	Palaniswamy et al. [20]	Mun et al. [19]	Our Scheme
SM1	$3T_{HM} + T_H$	$2T_{HM} + T_H$	$T_S + T_{CV} + T_{SV} + T_H$	$T_{HM} + 2T_H$	$T_S + T_H + T_{E/D}$
ECU1	$3T_{HM} + T_H$	$2T_{HM} + 2T_H$	$T_S + T_{CV} + T_{SV} + T_{E/D}$	$T_{HM} + 2T_H$	$T_S + T_{SV} + T_H + 2T_{E/D}$
connection	$6T_{HM} + 2T_H$	$4T_{HM} + 3T_H$	$2T_S + 2T_{CV} + 2T_{SV} + T_{E/D} + T_H$	$2T_{HM} + 4T_H$	$2T_S + T_{SV} + 2T_H + 3T_{E/D}$
SM2	$T_{HM} + 2T_H + T_{E/D}$	$T_H + 2T_{E/D}$	$T_S + T_{SV}$	$T_{HM} + 2T_H + 2T_{E/D}$	$T_{SV} + 2T_H + T_{E/D}$
ECU2	$T_{HM} + 2T_H + T_{E/D}$	$T_H + 2T_{E/D}$	$T_S + T_{SV}$	$T_{HM} + 2T_H + 2T_{E/D}$	$T_H + T_{E/D}$
certification	$2T_{HM} + 4T_H + 2T_{E/D}$	$2T_H + 4T_{E/D}$	$2T_S + 2T_{SV}$	$2T_{HM} + 4T_H + 4T_{E/D}$	$T_{SV} + 3T_H + 2T_{E/D}$
computational complexity (ms)	$8T_{HM} + 6T_H + 2T_{E/D} \approx 1062.62$	$4T_{HM} + 5T_H + 4T_{E/D} \approx 1023$	$4T_S + 2T_{CV} + 4T_{SV} + T_{E/D} + T_H \approx 1086.91$	$4T_{HM} + 8T_H + 4T_{E/D} \approx 1029.26$	$2T_S + 2T_{SV} + 5T_H + 5T_{E/D} \approx 1041$

We employ the computation time of [12] to estimate the computational complexity of our protocol and those proposed by others above. The computation time used by the ECU to evaluate cryptographic algorithms. The computation times for AES-128 and SHA-256 are 0.71 ms and 2.20 ms, respectively. The time required to verify the certificate is 122 ms. The computation time of HMAC is about 131 ms. The computation time for EDCSA-256 signature and verification is 88 ms and 122 ms, respectively. The meaning of each symbol is as follows.

$T_{E/D}$: Calculates the time required for encryption/decryption operations. T_{Cv}: the time required to verify the certificate. T_S: the time required to calculate the digital signature. T_{Sv}: the time required to verify the signature. T_H: the time required to compute the hash function. T_{HM}: the time required to calculate HMAC. SM1 and ECU1 are the computational complexity resulting from the connection stage. SM2 and ECU2 are the computational complexity resulting from the certification phase.

The comparison of computational complexity in Table 4 shows that our proposed protocol framework has high communication and computational efficiency and strong security properties. It can effectively prevent man-in-the-middle attacks, ensure security and meet real-time requirements.

7. Conclusions

With the integration of the in-vehicle network and the Internet, the insufficiency of the security features of the in-vehicle bus makes the in-vehicle network face many security challenges. We propose a certificateless framework based on the identity system. It has an efficient in-vehicle authentication and key transmission scheme without requiring a series of complex certificate operations such as application, query, and distribution. In addition, related aspects such as ECU update and maintenance are considered in the protocol design. To verify the security of the proposed protocol framework, we first construct a protocol

message flow model according to the protocol specification. Based on the colored Petri network (CPN) and the Dolev–Yao attack model that has been recognized by a large number of protocol researchers, we adopt the CPN Tools to model the proposed protocol and verify the model consistency. Subsequently, we introduce a man-in-the-middle attack on the model using the model detection scheme of the Dolev–Yao attack model. We write ML language for path search and further verify that there are no three types of man-in-the-middle attack states in the protocol path, including replay, tampering, and spoofing. The safety mechanism adopted by our proposed protocol meets the requirements of safety and real-time performance of key components and meets the needs of modern vehicles.

In this work, we mainly consider the use of man-in-the-middle attacks to formally analyze the security of protocols, which can be used as a case study to show the real-time work of the attacker model. The shortcoming of this work is that we did not carefully analyze other attack methods. In future work, while enhancing the security of the protocol, we will try to add other attack methods with the method of formal modeling to verify whether there are other types of security problems in the protocol. We will accumulate more experience in protocol design and explore how to combine formal modeling analysis methods with various attack methods. We will try to implement it within a software environment such as CPN Tools to achieve uncomplicated and automatic verification simultaneously.

Author Contributions: Conceptualization, L.Z. and T.F.; methodology, L.Z.; formal analysis, L.Z.; investigation, L.Z.; supervision, T.F.; funding acquisition, T.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under grant number 62162039, the National Natural Science Foundation of China under grant number 61762060, and the Foundation for the Key Research and Development Program of Gansu Province under grant number 20YF3GA016.

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. Lo Bello, L.; Mariani, R.; Mubeen, S. Recent Advances and Trends in On-Board Embedded and Networked Automotive Systems. *IEEE Trans. Ind. Inform.* **2019**, *15*, 1038–1051. [CrossRef]
2. Inam, M.; Li, Z.; Asghar, A. A Novel Protocol for Vehicle Cluster Formation and Vehicle Head Selection in Vehicular Ad-hoc Networks. *Electron. Inf. Eng.* **2019**, *10*, 103–119.
3. Zhang, H.; Meng, X.; Zhang, X. CANsec: A Practical in-Vehicle Controller Area Network Security Evaluation Tool. *Sensors* **2020**, *20*, 4900. [CrossRef] [PubMed]
4. Xiao, L.; Lu, X.; Xu, T. Reinforcement Learning-Based Physical-Layer Authentication for Controller Area Networks. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 2535–2547. [CrossRef]
5. Ying, X.; Bernieri, G.; Conti, M. Covert Channel-Based Transmitter Authentication in Controller Area Networks. *IEEE Trans. Dependable Secur. Comput.* **2022**, *19*, 2665–2679. [CrossRef]
6. Bella, G.; Biondi, P.; Costantino, G. TOUCAN: A protocol to secure Controller Area Network. 2021. In Proceedings of the ACM Workshop on Automotive Cybersecurity, Richardson, TX, USA, 27 March 2019; pp. 3–8.
7. Aliwa, E.; Rana, O.; Perera, C. Cyberattacks and Countermeasures for In-Vehicle Networks. *ACM Comput. Surv.* **2021**, *54*, 1–37. [CrossRef]
8. Hartzell, S.; Stubel, C.; Bonaci, T. Security Analysis of an Automobile Controller Area Network Bus. *IEEE Potentials* **2020**, *39*, 19–24. [CrossRef]
9. Yan, R.; Dunnett, S.J.; Jackson, L.M. Model-Based Research for Aiding Decision-Making During the Design and Operation of Multi-Load Automated Guided Vehicle Systems. *Reliab. Eng. Syst. Saf.* **2022**, *219*, 108264. [CrossRef]
10. Daohua, W.; Debiao, L.; Tao, T. Qualitative and Quantitative Safety Evaluation of Train Control Systems (CTCS) with Stochastic Colored Petri Nets. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 10223–10238.
11. Koscher, K.; Czeskis, A.; Roesner, F. Experimental security analysis of a modern automobile. In Proceedings of the 2010 31st IEEE Symposium on Security and Privacy (S & P 2010), Berleley/Oakland, CA, USA, 16–19 May 2010; pp. 447–462.

12. Szilagyi, C.; Koopman, P. Flexible multicast authentication for time-triggered embedded control network applications. In Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks, Lisbon, Portugal, 29 June–2 July 2009; pp. 165–174.
13. Szilagyi, C.; Koopman, P. Low cost multicast authentication via validity voting in time-triggered embedded control networks. In Proceedings of the 5th Workshop on Embedded Systems Security, Scottsdale, AZ, USA, 24 October 2010.
14. Lin, C.W.; Sangiovannin, V.A. Cyber-security for the Controller Area Network (CAN) communication protocol. In Proceedings of the 2012 ASE International Conference on Cyber Security, Washington, DC, USA, 14–16 December 2012; pp. 1–7.
15. Herrewege, A.V.; Singelee, D.; Verbauwhede, I. CANAuth-A Simple, Backward Compatible Broadcast Authentication Protocol for CAN bus. In Proceedings of the ECRYPT Workshop on Lightweight Cryptography, Louvain-la-Neuve, Belgium, 28–29 November 2011.
16. Groza, B.; Murvay, S. Efficient Protocols for Secure Broadcast in Controller Area Networks. *IEEE Trans. Ind. Inform.* **2013**, *9*, 2034–2042. [CrossRef]
17. 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 2014), Cambridge, MA, USA, 6–8 October 2014; pp. 13–18.
18. 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.* **2014**, *16*, 993–1006. [CrossRef]
19. 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]
20. Palaniswamy, B.; Camtepe, S.; Foo, E. An Efficient Authentication Scheme for Intra-Vehicular Controller Area Network. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3107–3122. [CrossRef]
21. Mundhenk, P.; Paverd, A.; Mrowca, A. Security in Automotive Networks: Lightweight Authentication and Authorization. *ACM Trans. Des. Autom. Electron. Syst.* **2017**, *22*, 1084–4309. [CrossRef]
22. Groza, B.; Murvay, P.S. Identity-Based Key Exchange on In-Vehicle Networks: CAN-FD & FlexRay. *Sensors* **2019**, *19*, 4919.
23. Han, M.; Wan, A.; Zhang, F. An Attribute-Isolated Secure Communication Architecture for Intelligent Connected Vehicles. *IEEE Trans. Intell. Veh.* **2020**, *5*, 545–555. [CrossRef]
24. Jeong, W.; Choi, E.; Choi, J.W. Autocorrelation-based Intrusion Detection System for Controller Area Network (CAN). *J. Inst. Control. Robot. Syst.* **2021**, *27*, 92–97. [CrossRef]
25. Musuroi, A.; Groza, B.; Popa, L.; Murvay, P.S. Fast and Efficient Group Key Exchange in Controller Area Networks (CAN). *IEEE Trans. Veh. Technol.* **2021**, *70*, 9385–9399. [CrossRef]
26. Lin, H.; Du, L. Optimization and Simulation of Controller Area Network Communication Model Based on Industrial Internet of Things Platform. *Complexity* **2020**, *2020*, 1076–2787. [CrossRef]
27. Gu, Y.; Liu, J.; Li, X.; Chou, Y.; Ji, Y. State space model identification of multirate processes with time-delay using the expectation maximization. *J. Frankl. Inst.* **2019**, *356*, 1623–1639. [CrossRef]
28. Wu, Y.; Feng, T. An Anonymous Authentication and Key Update Mechanism for IoT Devices Based on EnOcean Protocol. *Sensors* **2022**, *22*, 6713. [CrossRef] [PubMed]
29. Assaf, G.; Heiner, M.; Liu, F. Coloured fuzzy Petri nets for modelling and analysing membrane systems. *Biosystems* **2022**, *212*, 104592. [CrossRef] [PubMed]
30. Yao, L.; Liu, J.; Wang, D. Formal Analysis of SDN Authentication Protocol with Mechanized Protocol Verifier in the Symbolic Model. *Int. J. Netw. Secur.* **2018**, *20*, 1125–1136.
31. Liu, Z.; Liu, J. Formal verification of blockchain smart contract based on colored petri net models. In Proceedings of the 2019 International Computer Software and Applications Conference, Milwaukee, WI, USA, 15–19 July 2019; pp. 555–560.
32. Dolev, D.; Yao, C. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1981**, *29*, 198–208. [CrossRef]
33. Wu, Y.; Feng, T. Formal Security Analysis and Improvement Based on LonTalk Authentication Protocol. *Secur. Commun. Netw.* **2022**, *2022*, 8104884.
34. Gong, X.; Feng, T. Lightweight Anonymous Authentication and Key Agreement Protocol Based on CoAP of Internet of Things. *Sensors* **2022**, *22*, 7191. [CrossRef]