electronics

MDPI

Review

Security Issues with In-Vehicle Networks, and Enhanced Countermeasures Based on Blockchain

Narayan Khatri [1], Rakesh Shrestha [2] and Seung Yeob Nam [1,*]

[1] Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, Korea; narayankhatrig@ynu.ac.kr
[2] Yonsei Institute of Convergence Technology, Yonsei University, Incheon 21983, Korea; rakez_shre@yonsei.ac.kr
* Correspondence: synam@ynu.ac.kr

Abstract: Modern vehicles are no longer simply mechanical devices. Connectivity between the vehicular network and the outside world has widened the security holes that hackers can use to exploit a vehicular network. Controller Area Network (CAN), FlexRay, and automotive Ethernet are popular protocols for in-vehicle networks (IVNs) and will stay in the industry for many more years. However, these protocols were not designed with security in mind. They have several vulnerabilities, such as lack of message authentication, lack of message encryption, and an ID-based arbitration mechanism for contention resolution. Adversaries can use these vulnerabilities to launch sophisticated attacks that may lead to loss of life and damage to property. Thus, the security of the vehicles should be handled carefully. In this paper, we investigate the security vulnerabilities with in-vehicle network protocols such as CAN, automotive Ethernet, and FlexRay. A comprehensive survey on security attacks launched against in-vehicle networks is presented along with countermeasures adopted by various researchers. Various algorithms have been proposed in the past for intrusion detection in IVNs. However, those approaches have several limitations that need special attention from the research community. Blockchain is a good approach to solving the existing security issues in IVNs, and we suggest a way to improve IVN security based on a hybrid blockchain.

Keywords: in-vehicle networks; Controller Area Network (CAN); automotive Ethernet; FlexRay; security; vehicle; blockchain; machine learning; intrusion detection system (IDS)

Citation: Khatri, N.; Shrestha, R.; Nam, S.Y. Security Issues with In-Vehicle Networks, and Enhanced Countermeasures Based on Blockchain. *Electronics* **2021**, *10*, 893. https://doi.org/10.3390/electronics10080893

Academic Editor: Mehdi Sookhak

Received: 15 March 2021
Accepted: 6 April 2021
Published: 8 April 2021

1. Introduction

Recent decades have seen significant advancements in technology for self-driving vehicles and smart cars. Vehicular networks are networks of vehicle nodes providing various facilities, such as traffic management, parking management, accident avoidance, critical message dissemination, etc. [1]. There are various research fields where these vehicle nodes act as a communication messenger, such as Vehicular Ad-hoc Networks (VANETs), the Internet of Vehicles (IoV), Vehicle-to-Everything (V2X) communications, etc. There is a separate research field for in-vehicle networks (IVNs), dealing with the connections between the Engine Control Unit (ECU), the Transmission Control Unit (TCU), the Anti-lock Braking System (ABS), Body Control Modules (BCMs), and the various sensors inside the vehicle. There are protocols like Controller Area Network (CAN), FlexRay, and Ethernet, which help in the smooth functioning of in-vehicle networks [2].

The automotive industry is moving toward autonomous and connected vehicles. There are various benefits to going through this option. First of all, connected vehicles provide convenience and flexibility to passengers. Another benefit is that, with the interconnections between vehicles, more information is shared regarding safety and security, which leads to more secure transportation systems. Up to now, VANETs have mainly focused on providing efficiency and safety for drivers and passengers in a vehicular network. Today, Connected and Autonomous Electric Vehicles (CAEVs) is a new technology that is

booming [3]. The main idea is that vehicles can interact with the external world through smart devices. This leads to various security issues as vehicles connect to the internet [3]. The main goal of the VANET is to provide vehicles with critical information (such as accident reports) with guaranteed performance in terms of latency and accuracy. However, this is a challenging task, as discussed in [4]. The increase in connectivity among transportation facilities, and the advancements in technologies, such as V2X-communications, have expanded the security holes, through which attackers can break into an in-vehicle network. Researchers have shown that it is possible to hack into a vehicle (either physically or remotely) through telematics systems, the on-board diagnostics (OBD-II) port, infotainment systems, and so on. Miller and Valasek were able to remotely kill a jeep on the highway by accessing its in-vehicle networks via the CAN-bus [5,6]. There are various researchers who focus on the drawbacks of the in-vehicle network protocol design that may allow a hacker access to in-vehicle networks.

The CAN protocol is widely used for in-vehicle communications in controlling and providing various functionalities to a vehicular system. It helps critical real-time communications for engine management, brake control, airbags, body systems control, etc. The CAN protocol helps in the transmission of CAN packets between various ECUs via inter-connected buses inside a vehicle network. Initially, CAN was used in automobiles owing to its simplicity, deterministic contention resolution mechanism, reduced network complexity, and low wiring costs. However, it was not designed with security in mind. CAN is based on broadcast communications, i.e., every ECU connected to the bus can send/receive messages on the bus. There is no authentication mechanism for messages, and encryption is not applied. Furthermore, the identifier (ID)-based priority mechanism for arbitration processes makes the CAN vulnerable to attack [7]. Hackers can mount attacks to disable functionalities in the vehicle, either locally by using an OBD-II diagnostics tool or through remote access to the telematics or infotainment systems. Thus, the security of in-vehicle CAN networks is vital to the overall security in modern transportation systems. This paper provides a comprehensive state-of-the-art survey of various security attacks that are mounted against in-vehicle network protocols. We also explore intrusion detection systems (IDSs) and other security solutions that were developed as countermeasures to those attacks. Furthermore, we provide security solutions for in-vehicle networks by using a hybrid blockchain framework.

Blockchain is a secured framework that is resistant to data modification. It has features like decentralization (i.e., no need to rely on a centralized node for recording, storing, and updating data), transparency (data records are visible to all nodes in the blockchain network), and it is open-source (i.e., open to the public), autonomous (i.e., nodes can update or transfer data by themselves), immutable (i.e., records are not corrupted once they become part of the chain), and anonymous (i.e., transactions are private) [8,9].

We consider a hierarchical (or hybrid) blockchain, i.e., the combination of a public and a private blockchain, so the information can be securely processed within autonomous vehicles while those vehicles are moving in a decentralized manner. We consider a private blockchain for each vehicle when recording internal vehicle logs, events, etc., occurring within the in-vehicle network. The public blockchain is considered for recording the specific hash information of the private blockchain of each vehicle, along with traffic events occurring around the vehicle. The road-event information (inter-vehicle communications, accident events, etc.) is stored in the public blockchain for sharing with neighboring vehicles.

Table 1 provides a list of commonly used abbreviations in this paper along with their expansions. This paper considers security issues in CAN, FlexRay, and automotive Ethernet because the majority of the research on in-vehicle networks focuses on these three protocols [10].

Table 1. List of abbreviations used in this paper.

Abbreviation	Full Form
ABS	Anti-lock Braking System
BCMs	Body Control Modules
CAN	Controller Area Network
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CAEVs	Connected and Autonomous Electric Vehicles
DoS	Denial of Service
DPoS	Delegated Proof of Stake
DNN	Deep Neural Network
DCU	Domain Controller Unit
ECU	Electronic Control Unit
E/E	Electrical and Electronics
GAN	Generative Adversarial Network
IVN	In-Vehicle Network
IoV	Internet of Vehicles
ID	Identifier
IPFS	Interplanetary File System
LSTM	Long Short-Term Memory
LIN	Local Interconnect Network
OBD	On-Board Diagnostics
PuBC	Public Blockchain
PvBC	Private Blockchain
PoA	Proof of Authority
PCA	Principal Component Analysis
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SVM	Support Vector Machine
TCU	Transmission Control Unit
V2X	Vehicle to Everything
VANETs	Vehicular Ad Hoc Networks

The main contributions of this paper are listed below.

- Detailed security issues with in-vehicle network protocols (CAN, FlexRay, and automotive Ethernet) are provided.
- A detailed survey of various security attacks on CAN bus networks is provided, along with the intrusion detection systems that were developed to prevent those attacks. We investigate those IDSs and address their advantages and disadvantages.
- A survey of attacks and security solutions for automotive Ethernet and FlexRay protocols is provided.
- We suggest a way to improve the security of in-vehicle networks by using a hybrid blockchain framework.
- We suggest open research issues and future research directions for in-vehicle network security.

The rest of the paper is organized as follows. Section 2 provides background on in-vehicle networks and on the vulnerabilities prevalent under various protocols. Section 3 provides a survey of related work done with in-vehicle networks, while Section 4 discusses the various types of security attacks launched against the various protocols, along with countermeasures to those attacks. Section 5 suggests a solution for in-vehicle network security using a hybrid blockchain framework. Section 6 discusses open research issues and future work that can be considered for IDS design. Finally, Section 7 concludes the paper.

2. In-Vehicle Networks

In-vehicle networks are specialized internal communication networks that interconnect various components inside a vehicle. The components inside the vehicle include ECUs, gateways, sensors, actuators, etc. It is estimated that modern high-tech cars have up to 70 ECUs with 2500 electronic signals being exchanged among the various components [11,12]. There are various types of electronic control units, such as the ECU, the TCU, the ABS, BCMs, the Speed Control Unit, the Battery Management System (BMS), the Powertrain Control Module, and the Door Control Unit (DCU). These electronic units receive input from various sensors for computations. Sensors are incorporated into the vehicle to help recognize and solve possible problems, including needing repair, servicing, etc. Sensors play a key role in automobiles. Typical functions include monitoring the crankshaft's rotation speed, managing the car speed, verifying the speed of the wheels, checking fuel temperature, monitoring tire pressure, monitoring the exhaust gases to check the oxygen ratio, computing air density in the engine, etc. All these functionalities provided by various sensors help drivers with early problem detection and accident prevention. The electronic control modules exchange data during normal operation of the vehicle. For instance, the ECU provides information about engine speed to the TCU, and the TCU tells other components when a gear shift will take place. Each unit has its predefined function and controls specific components using a standard protocol over the vehicle network. The protocols used for in-vehicle networks include CAN, Local Interconnect Network (LIN), FlexRay, Ethernet, and Media Oriented Systems Transport (MOST) [2]. These protocols are specifically designed to transmit messages within a pre-defined time limit with assurances on message delivery. Table 2 compares various in-vehicle networks in terms of bandwidth, application domain, advantages and disadvantages.

Table 2. Comparison of various pfrotocols used in in-vehicle network communications.

Protocol	Bandwidth	Application Domain	Advantages	Disadvantages
CAN	125 Kbps–1 Mbps	Widely used in powertrain and body control domains	Low cost, no need of central coordinator	Less bandwidth
LIN	1 Kbps–20 Kbps	Widely used in simple and less time-critical applications	Low cost, easy to implement	Low speed
FlexRay	10 Mbps	Widely used in advanced chassis control	High speed, better fault tolerance than CAN and Lin	High cost
MOST	24 Mbps	Widely used in infotainment applications	High speed	High cost
ETHERNET	100 Mbps	Widely used in the future in applications requiring high bandwidths	High speed (100 times faster than CAN bus)	High cost per node

The in-vehicle network architecture can be of the classic Electrical and Electronics (E/E) type with a central gateway, or a domain-based E/E type with several functional domains connected through a central gateway [13]. The communication overhead is increasing in the classical E/E architecture, because a variety of ECUs have to communicate and route data through a central gateway. In order to reduce this bottleneck, a domain-based architecture was developed, where various functional domains with a domain controller are interconnected through the central gateway. Here, most of the data communication occurs within these functional domains, reducing the communication load on the gateway. Furthermore, the architecture is scalable to allow adding more functional domains.

2.1. The Controller Area Network Protocol

CAN is one of the well-known vehicle bus standards for in-vehicle networks. It is popular in automotive and industrial applications due to its low cost and flexible design, thereby reducing the wiring harness. For example, the number of wires was reduced by 40% in the Peugeot 307, which embeds two CAN buses [14]. CAN is a message-based protocol;

the packets do not have information about the sender and receiver of the messages, and every node can read the messages transmitted over the bus. Functions supported by the protocol in the automotive domain include auto start/stop, electric parking brakes, parking assistance, automatic lane detection, collision avoidance, etc. Figure 1 shows a CAN bus node. It consists of the central processing unit (CPU), the CAN controller, and a transceiver. The function of the CPU is to decode the received messages and to decide on the messages to transmit. Each node can send or receive messages, but not simultaneously. A message or frame consists of an ID and a data payload of up to eight bytes (64-bits).

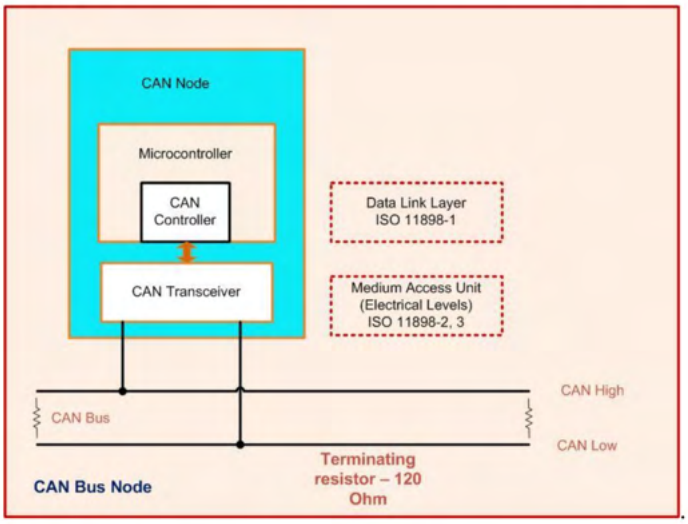

Figure 1. Controller area network bus node.

The CAN standard frame format consists of an 11-bit identifier. The identifier of the CAN frame represents the message priority. If a message has a lower identifier value, it will have a high priority on the bus. This field is used in the arbitration process to avoid conflict when two nodes or more than two nodes are transmitting the messages simultaneously on the bus. Figure 2 shows the CAN frame format. It consists of seven fields: Start of Frame (SOF), arbitration, control, data, Cyclical Redundancy Check (CRC), Acknowledge (ACK), and End of Frame (EOF).

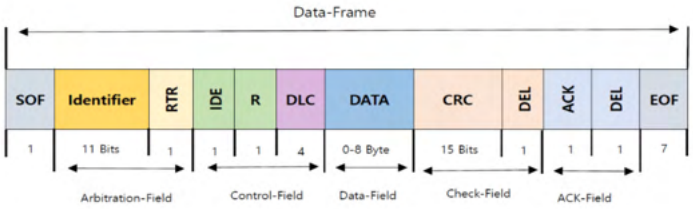

Figure 2. The controller area network (CAN) frame format (11-bit identifier).

CAN has four different frame types [15,16]. They are the data frame, the remote frame, the error frame, and the overload frame. The data frame is used for actual data transmission from a transmitter to other nodes (receivers). The remote frame is used by a node to request a certain message with a particular identifier. If any of the nodes on the bus detect an error, that node will transmit an error frame. The overload frame is used to inject a delay between the data and remote frames.

CAN Bus Attack Interfaces

The attack surface in the connected car environment was demonstrated by Aliwa et al. [17]. It consists of telematics units, infotainment systems, the OBD-II port, and sensors. Figure 3 illustrates the CAN bus attack surfaces. The attacker may inject messages into

the network through a direct connection, like an OBD-II connection, or wirelessly through telematics or infotainment systems.

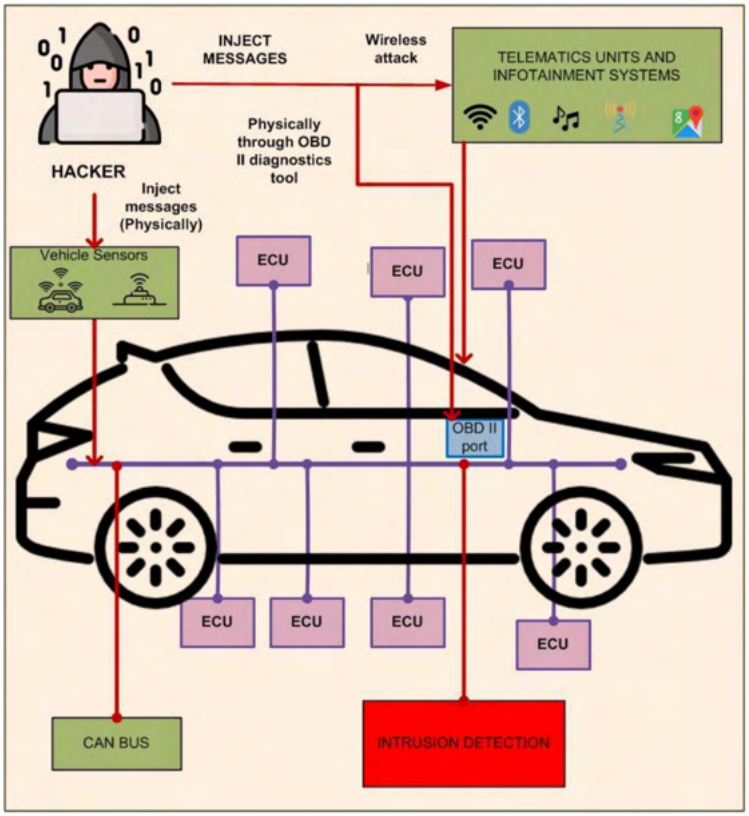

Figure 3. CAN bus attack interfaces.

2.2. Automotive Ethernet Protocol

The high-bandwidth requirements of modern vehicle applications motivated the introduction of automotive Ethernet (AE) as an essential component of in-vehicle networks [18]. Automotive Ethernet is going to be the future in-vehicle network protocol that satisfies the bandwidth requirements for multimedia applications, autonomous driving, and safety applications, such as the advanced driver-assistance system (ADAS) [10,19]. Other features of this protocol include efficient communication, lower latency, scalability, reduced wiring harness, and low cost. The various protocols used for AE are 100-Base-TX, BroadR-Reach (100Base-T1), Audio Video Bridge-Time-Sensitive Network (AVB/TSN), Diagnostics over Internet Protocol (DoIP), and Scalable Service-Oriented Middleware over Internet Protocol (SOME/IP), etc. [20]. 100Base-T1 is an AE standard that provides bandwidth of up to 100 Mbps over a single unshielded twisted-pair cable [19]. 1000Base-T1 is in development and will support data rates of up to 1000 Mbps [21]. Once a vehicle is on the market, the in-vehicle network protocols cannot be updated. Thus, the use of protocols and specifications should be handled mindfully during the design process. Since bus protocols are insecure, this emerging technology has room for enhanced security features. The security features that have to be implemented in AE networks are access control to the network, secure on-board communication, a data access policy, anomaly detection and prevention mechanisms, etc. [10]. There is no authentication mechanism to enforce access control against unknown devices in AE networks to protect the various software and hardware components, such as the operating system, drivers, IP stack, Ethernet interfaces, the CPU, etc. [21]. There is a need for trust among devices participating in the network, which can be guaranteed through authentication mechanisms, as in IEEE 802.1x. The topology for in-vehicle network communication is a hierarchical structure where various protocols are connected to a central gateway [21]. However, this topology is neither dynamic nor exten-

sible. If a hacker gets access to one port of the in-vehicle network, there is a chance he/she will also get access to other parts of the network that are connected through gateways. Electric Vehicles (EVs) are connected to the outside world through V2X communications, which uses Ethernet as a backbone [22], e.g., the communication between an EV and a charging station during negotiations for the charging service. There is an issue regarding the authenticity of the communicating bodies in vehicular networks. There should be a mechanism to evaluate authenticity and integrity, and protect the confidentiality of messages flowing between in-vehicle networks and outside networks [23]. The AUTOSAR consortium has not covered the issue of initializing pre-shared secret keys between senders and receivers of messages in Secure Onboard Communication (SecOC) [24]. Ethernet technology has been well studied in the cybersecurity community. Hackers can use previous knowledge from the IT domain and can launch sophisticated attacks against vehicle systems. Thus, future AE technology should be designed with these considerations in mind. Furthermore, cybersecurity attacks and their countermeasures should be investigated extensively in order to secure in-vehicle networks.

2.3. FlexRay Protocol

FlexRay is a reliable, time-triggered protocol that provides a higher bandwidth of up to 10 Mbps, compared to CAN networks, which provide data rates of up to 1 Mbps. All the ECUs connected by this protocol are synchronized to global time, and the data frames are transmitted and received within pre-defined time slots. This protocol has high fault-tolerance, compared to CAN. The properties of the FlexRay protocol regarding transmission capabilities include large payloads, flexibility in terms of network topologies, and the transmission of deterministic, as well as dynamic, data in one cycle. However, they have the drawbacks of having high cost and increased complexity in in-vehicle networks. The FlexRay frame consists of a header segment, a payload, and trailer segments. The header consists of the slot ID, payload length, cycle counter, etc. The payload segment contains the data. The trailer provides a CRC for the frame. This protocol is particularly used in drivetrain and chassis applications with time-critical and event-triggered messages. The vulnerabilities in this protocol are that it lacks confidentiality, authentication, and data freshness mechanisms [25]. The protocol is not designed to guarantee security from external attack. According to Shrestha and Kim [19], it is vulnerable to various attacks, such as eavesdropping, masquerading, injection, and replay attacks. Nilsson and Larson [26] mentioned that the application layer is missing, making it insecure. The CRC section of the FlexRay frame can protect the integrity of the data against transmission errors. Availability is guaranteed through time division multiplexing. These features help in providing safety to the network, but do not guarantee protection against security attacks.

3. Related Works

There have been many surveys of in-vehicle network security issues and countermeasures. Zeng et al. [2] provided a comprehensive survey of the five most common protocols used for in-vehicle networks. They analyzed the protocols in terms of cost, data transmission ability, and fault-tolerance capability. They discussed various security threats to in-vehicle networks and the security solutions that can be implemented to mitigate those threats, including cryptography, authentication schemes, and controlled physical access to the vehicle. They believe that Ethernet will be the leader in next-generation in-vehicle communications due to its high bandwidth and low-cost PHY techniques. However, they did not survey various security attacks that are launched against in-vehicle networks or the IDSs developed to prevent those attacks. Lokman et al. [7] provided a survey of IDSs implemented for CAN networks. Their paper presents the deployment strategies of these IDSs. The authors investigated anomaly-based IDSs for securing a CAN bus. However, the review did not cover detailed security attacks on other types of in-vehicle networks and the corresponding countermeasures. Security concerns over the CAN bus were provided by Bozdal et al. [27]. The authors exploited various attack surfaces that are typical in the CAN

bus environment. That paper lacked a systematic review of various attacks that are prevalent against in-vehicle networks. Wu et al. [28] provided a survey of intrusion detection schemes for in-vehicle networks. The authors explained the limitations and challenges in designing intrusion detection systems for in-vehicle networks, and state-of-the-art IDSs for IVNs were provided. Tomlinson et al. [29] explained the various IDSs for CAN networks. Those authors studied recent work on CAN intrusion detection systems and addressed in detail the research challenges for the evaluation of those IDSs.

In this paper, we provide a survey of various attack types that can be launched against in-vehicle networks and the IDSs developed to mitigate those attacks. The advantages and disadvantages of these IDSs are also investigated. A comparison of our work against previous surveys of in-vehicle network security is in Table 3. Our work investigates the vulnerabilities and possible attacks against various in-vehicle network protocols, such as CAN, FlexRay, and automotive Ethernet, along with countermeasures. We focus on these three protocols because they are widely used for transmitting engine control and critical safety messages. Our choice for selecting automotive Ethernet in this work is due to the fact that the protocol has a high potential to be used in wider application domains in future automotive vehicles. Furthermore, we provide a suggestion for in-vehicle network security using a hybrid blockchain technology.

Table 3. Comparison of various surveys of in-vehicle network security.

Reference	In-Vehicle Network Vulnerability Analysis	Detailed Survey of IDSs in IVNs Based on Various Attack Types	Includes a Blockchain-Based IDS Suggestion for In-Vehicle Network	Discusses Open Issues and Future Works
Zeng et al. [2]	✔	✘	✘	✔
Avatefipour and Malik [30]	✔	✘	✘	✘
Wu et al. [28]	✔	✘	✘	✔
Bozdal et al. [27]	✔	✘	✘	✘
Lokman et al. [7]	✔	✘	✘	✘
Tomlinson et al. [29]	✘	✘	✘	✔
This article	✔	✔	✔	✔

4. Security Issues and Countermeasures for In-Vehicle Networks

This section investigates security attacks against, and countermeasures for, various protocols, including CAN, automotive Ethernet, and FlexRay. We also compare various security solutions and intrusion detection systems developed to detect attacks against in-vehicle networks. An IDS is a device or software application that monitors network traffic for malicious activities and issues an alert when malicious activity is detected. The alert is typically sent to the network administrator or is centrally collected using a security information and event management (SIEM) system. The SIEM system then uses an alarm-filtering method to identify malicious activities from the collected reports. IDSs can be classified into two types: the network intrusion detection system (NIDS) and the host-based intrusion detection system (HIDS). IDS approaches can also be classified into two types: signature-based/rule-based detection and anomaly-based detection. Anomaly-based detection is the process of identifying rare events, items, or observations that are suspicious and that differ significantly from the majority of the data. Anomaly detection is usually based on a machine learning technique.

Modern vehicles use sophisticated sensors providing various functionalities to the vehicle, and this trend is rising. This has led to a huge amount of data being generated by in-vehicle networks. Thus, the data-driven approach to intrusion detection is one of the interesting topics that researchers focus on nowadays. With an analysis of these data, we can get meaningful information about in-vehicle networks.

IDSs can be compared from diverse aspects, e.g., in terms of detection speed, accuracy, algorithm complexity, learning time, robustness, and detection coverage.

4.1. Security Issues and Countermeasures for CAN

4.1.1. Security Issues

The CAN protocol was not designed with security in mind and has many security holes that hackers might use to attack an in-vehicle network. Several authors have explained the security vulnerabilities in the CAN bus [31,32]. Practical attacks have been demonstrated in the literature. Hoppe et al. demonstrated a possible attack on CAN networks with practical examples [33]. The authors executed frame sniffing and replay attacks to control the window lift, the warning lights, and the ABS. Woo et al. showed that it is possible to execute a long-range wireless attack in real vehicles by using malicious smartphone applications in a connected car [34]. The attack surface can be telematics, a Wi-Fi port, Bluetooth, or an OBD port. Similarly, Miller and Valasek demonstrated security flaws in the Jeep Cherokee by executing practical attacks on the CAN bus and thereby disabling critical functions in the vehicle (e.g., disabling the brakes and stopping the engine) [35]. The attack was launched remotely through the infotainment system via the Sprint cellular network. This attack led to a recall of around 1.4 million vehicles. A group of researchers from Keen Security demonstrated practical attacks on the Tesla X [36]. The attackers were able to take control of the brakes, unlock the doors, activate signals, etc. We now investigate the security issues in CAN networks in more detail.

ID-Based Arbitration Mechanisms

The CAN frame format does not have a field for the sender or receiver of a message. The CAN protocol uses an arbitration scheme called Carrier Sense Multiple Access with Collision Detection (CSMA/CD). When two or more nodes are transmitting messages on the bus, the node with a lower ID value will have higher priority. If any node is transmitting a dominant bit (i.e., bit 0), all the other nodes in the bus will read this dominant bit regardless of the bit value they have transmitted. When a node on the bus observes a higher priority message, it stops transmission and backs off. This is the arbitration scheme used in the CAN protocol [37]. The arbitration mechanism is illustrated with an example in Figure 4 [30,38]. Let us consider two nodes with 11-bit CAN IDs: Node 15 (binary representation: 00000001111) and Node 16 (binary representation: 00000010000). When these two nodes transmit simultaneously, they first transmit a start bit and then transmit the first six zeros of their identifier. When the seventh ID bit is transmitted, Node 16 transmits a recessive bit and observes that Node 15 has transmitted a dominant bit. Thus, Node 16 stops transmission until Node 15 transmits all of its bits.

	Start Bit	ID Bits										The rest of the frame
		10	9	8	7	6	5	4	3	2	1	0	
Node 15	0	0	0	0	0	0	0	0	1	1	1	1	
Node 16	0	0	0	0	0	0	0	1	Stopped Transmitting				
CAN Data	0	0	0	0	0	0	0	0	1	1	1	1	

Figure 4. The CAN bus arbitration mechanism.

With this arbitration mechanism, the CAN bus is vulnerable to a denial of service (DoS) attack. Hackers can send high-priority messages within a short interval of time causing the legitimate node to stop transmission [30]. Song et al. demonstrated a kind of injection attack on an in-vehicle CAN bus [39].

Lack of Confidentiality

The CAN protocol broadcasts frames over the bus, and every node connected to the bus can decide whether to accept or reject the message. Therefore, a malicious node can keep track of all the messages flowing on the bus. Messages are not encrypted in CAN bus

networks. As a result, every node can see the messages passing through the bus. Attackers can analyze those messages and use them to hack into the in-vehicle network.

Lack of Authenticated Messages

Since the CAN frame format does not contain sender and receiver information, the receiver of the message cannot determine if a message is from a legitimate source or not. Thus, the authenticity of the message cannot be guaranteed in CAN networks.

Classification of attacks prevalent in CAN networks is shown in Figure 5. There are five types: DoS, fuzzy attacks, replay attacks, spoofing, and impersonation. These major types of attack are well-studied in the literature and can have serious effects on the security of in-vehicle systems.

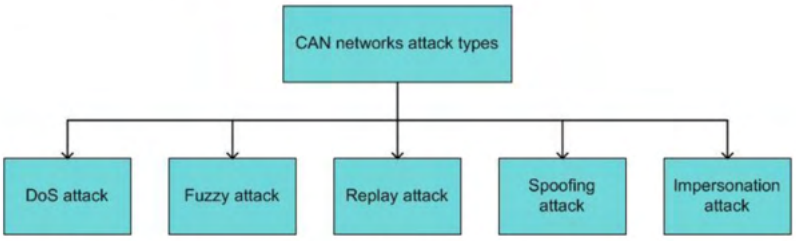

Figure 5. Classification diagram for CAN network attack types.

Denial of service is a type of cyber-attack where the attacker floods the target network with a large number of fake requests in order to make the information systems, services, or network resources unavailable to legitimate users [40]. The CAN network is vulnerable to DoS attacks due to its arbitration mechanism for contention resolution. Figure 6 shows a scenario of a DoS attack on CAN networks.

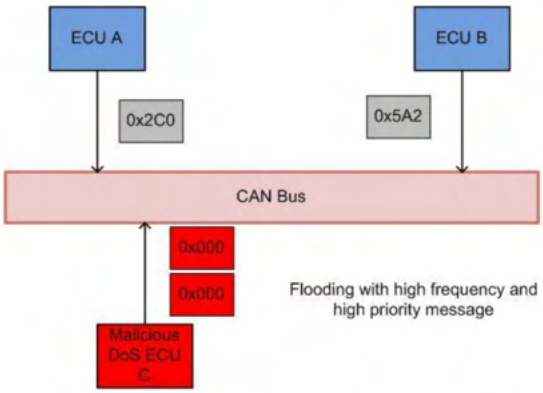

Figure 6. A sample denial of service (DoS) attack on a CAN network.

In the fuzzy attack, a malicious ECU sends random frames with spoofed IDs with arbitrary data values. This type of attack is simple to implement because of the small range of valid CAN frames flowing over the bus, and it does not require reverse engineering. Figure 7 shows a scenario for a fuzzy attack on a CAN network. Researchers have shown that fuzzy attacks can cause vehicles to malfunction and exhibit unintended behavior, thereby putting the lives of the passengers at risk.

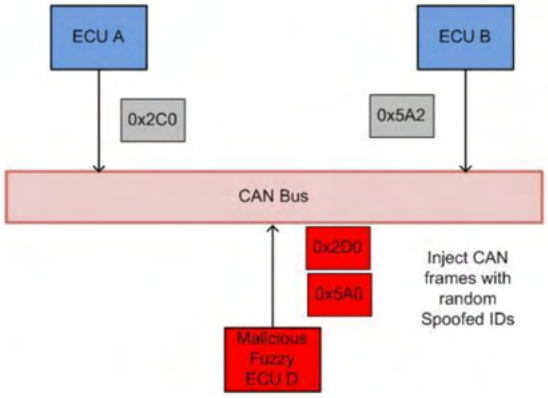

Figure 7. A sample fuzzy attack on a CAN network.

Hackers with control of an ECU can sniff (collect) messages transmitted over the bus, and then replay them to create a hazardous situation. This type of attack is hard to detect because the message syntax copies a legitimate CAN message. Figure 8 depicts an example of a replay attack.

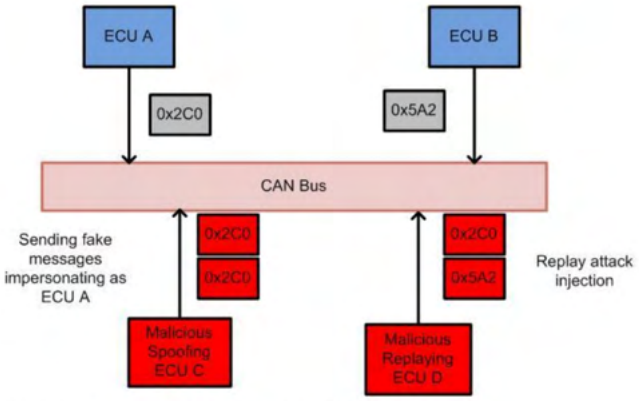

Figure 8. Sample CAN bus attacks (spoofing and replay).

Spoofing refers to a type of attack in which a malicious node will send messages with a fake ID that is the same as a legitimate node. Thus, the receiver node might determine the message is from a legitimate node, and it is difficult to differentiate malicious ones from legitimate ones, since there is no authentication of messages on the CAN bus. The spoofing attack is also illustrated in Figure 8.

Impersonation is a type of attack in which a malicious node is planted in the CAN bus, impersonates a legitimate node, and takes over its functionality to completely disable the operation of the legitimate node [15,41]. This type of attack can make the vehicle perform in unintended ways, endangering the vehicle and its surroundings. Figure 9 depicts an impersonation attack.

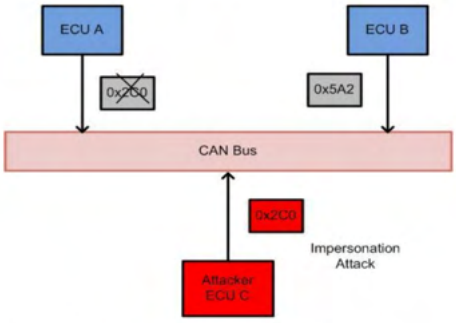

Figure 9. Illustrating an impersonation attack.

4.1.2. Countermeasures

A classification diagram for anomaly-based intrusion detection systems is shown in Figure 10. IDSs can be categorized into three types: timing or frequency analysis, machine learning-based, and entropy analysis. Furthermore, machine learning-based IDS schemes can be categorized as supervised learning, unsupervised learning, or deep learning.

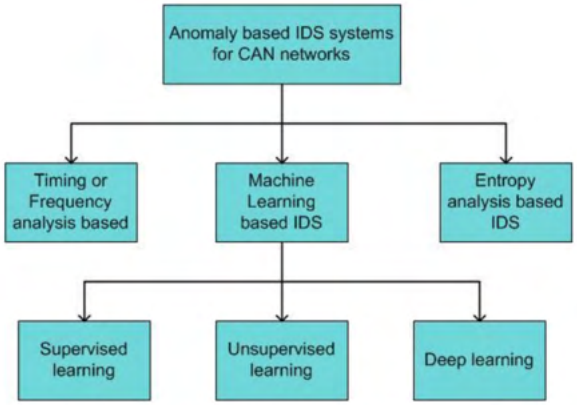

Figure 10. Classification of intrusion detection systems (IDSs) for CAN networks.

The following subsections explain attack types and IDSs in more detail.

Hoppe et al. [33] did pioneering work in the field of CAN bus security attacks with practical experiments using the CANoe simulation tool by Vector Informatik. The authors developed four attack scenarios and demonstrated the vulnerability of the CAN protocol from the lack of an authentication mechanism. Vulnerabilities in the electric window lifts, warning lights, airbag control system, and gateway ECU have been explored. By injecting malicious code into the ECU of the simulation framework, the authors were able to open a window until the attack finished. This attack led to denial of service in the electric window lift. The authors also provided countermeasures to the demonstrated attacks through intrusion detection systems [42]. The proposed IDS for the electric window lift and warning lights has a simple design and was implemented with reduced costs. However, this detection system cannot detect malicious messages that occur aperiodically, as in an event-based system. The system assumes an authentic sender is transmitting the message. Furthermore, they considered simple attack patterns based on message frequency, and did not consider other attack types, such as spoofing messages or ECU impersonation. Koscher et al. [43] proposed practical attacks to control the various functionalities of a vehicle, such as the engine, the brakes, the heating and cooling systems, the lighting system, the instrument cluster, the radio, the locks, etc. The authors used Carshark, which can inject packets and analyze the CAN bus. Through an analysis of the security vulnerabilities in an intra-vehicular network, the authors executed various security attacks in a real car on the road. The generic denial of service disabled the functioning of the CAN bus components. With this attack, the authors showed the possibility of a DoS attack on the speedometer reading service by generating false information. Their research exploited the vulnerability in modern cars, but they did not provide countermeasures. Other works [44,45] showed practical attacks on the CAN bus along with an analysis of various automotive attack surfaces but did not provide countermeasures to mitigate attacks.

Most of the previous IDSs for detection of DoS attacks depend on timing analysis or frequency analysis of CAN bus messages [46,47]. The injection of a huge number of messages changes the frequency of certain IDs, as shown in Figure 11. Hackers usually launch DoS attacks through injection of a huge number of high-priority messages in a short time span, thereby increasing the rate of message injection [37]. Song et al. [46] demonstrated a DoS attack on a real vehicle by injecting messages through the OBD-II port of the car. The authors used the KVASER CAN interface to connect to the CAN bus, and injected messages for five to 10 s at a rate 30 times higher than normal messages. The IDS

evaluation result showed 100% detection accuracy. However, this model is very simple and capable of detecting attacks that inject a larger number of CAN messages in a short time interval. The model can detect malicious messages that occur regularly, but is unable to detect malicious messages that arrive at irregular intervals. Furthermore, the IDS cannot detect attacks that include changes in message semantics. More research on CAN-message frequency-based IDSs was presented in the articles by Young et al., and Taylor et al. [48,49].

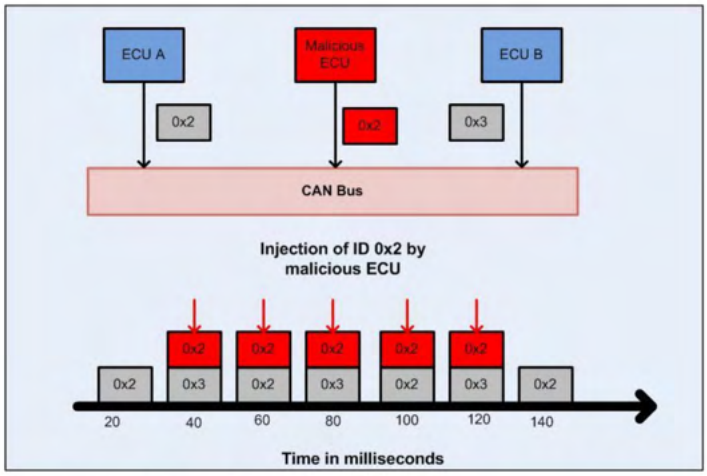

Figure 11. Timing analysis of a DoS attack.

Lee et al. [15] performed experiments by launching DoS attacks on the Kia Soul. Furthermore, they developed an IDS based on the offset ratio and the time interval between request and response messages over the CAN bus. If a particular node sends a request for a message by transmitting a remote frame with a certain ID, the node corresponding to that ID responds quickly with the data frame. The authors used the concept where, during an injection attack, if the response time and offset ratio (i.e., the order and the time interval from request to response message) is higher than a certain threshold, an anomaly is detected. In the CAN bus, nodes receive messages sequentially, and therefore, it is possible to determine the order and time interval between request/response message pairs. Halder et al. [50] proposed a clock offset-based intrusion detection system for the CAN bus. These authors developed a way to fingerprint the transmitting ECU's clock by measuring the clock offset values of inter-departure messages, and they used that as a basis for normal clock behavior. The deviation in the clock frequency due to message injection will result in a change in the clock offset of inter-departure messages. The experimental results showed that their IDS performed better than OTIDS in detecting attacks. However, their model has higher computational and memory requirements.

Machine learning-based intrusion detection systems for the CAN bus have been widely researched [51,52]. Kang and Kang [53] proposed an IDS based on a deep neural network (DNN). Statistical features of CAN data were captured using an unsupervised deep-belief network, and a decision was made on any anomaly in the data. The experiment results showed the model can detect 3900 frames within 7 to 8 milliseconds, and they obtained 99% accuracy while keeping false positives under 1–2%. Song et al. experimented with various attack scenarios in a real vehicle with message injection attacks, such as DoS and fuzzy attacks, and by spoofing the gear and the RPM [39]. The authors analyzed the sequence of messages for intrusion detection. Datasets were constructed by using a CAN data logger from a real vehicle during message injection-attack periods. The authors proposed an IDS based on a deep Convolutional Neural Network (CNN) architecture called Inception-ResNet [39]. It consists of a training step and a detection step. In the training step, the CNN classifier is trained, and during the detection phase, real CAN data frames are passed through this model to determine whether they are normal or attack situations. The evaluation results showed that the proposed model had a high DoS detection rate (100%

precision). The model showed a low error rate, compared to previous work. However, the proposed algorithm has high computational and memory requirements.

Seo et al. [54] executed DoS attacks by injecting fake messages into CAN networks. They developed a GIDS algorithm for intrusion detection systems using a Generative Adversarial Network (GAN). This model is based on deep learning and can detect unknown attacks using only the normal CAN data for training. The CAN data are first converted to images using one-hot vector encoding. Two discriminators are used in the intrusion detection model. The first discriminator is used for training with known attacks, and the second is used for training with unknown attacks. The experimental results showed that a DoS attack detection rate from the model is 99.6%. GIDS takes 0.18 s to detect about 1954 CAN messages, which shows that the detection speed is good. The drawback with this IDS is that it cannot distinguish between anomalous traffic from attackers and anomalous traffic from malfunctioning ECUs [54].

A Long Short-Term Memory (LSTM) neural network-based IDS was proposed in [55]. The authors investigated various attacks on the CAN bus, such as DoS, fuzzy attacks, and spoofing attacks. They captured normal messages for 120 s by using the CAN Vehicle Spy 3 tool. DoS attacks were executed by flooding the network with messages using a CAN ID of 00, a data length code (DLC) of 8, and a data field of 00. The dataset was collected for normal messages and attacks. A real car was used to create the dataset for DoS, fuzzy attacks, and spoofing attacks [56]. The LSTM architecture consists of an input layer, a hidden layer, and an output layer. The input for the model is CAN packet payloads. The activation function is softmax. The output layer classifies incoming packets as either normal or an attack. The IDS can detect attacks with high accuracy, but the model has high computational time for intrusion detection, with increased latency in communications. Lin et al. proposed a deep learning-based anomaly detection system for CAN networks [57]. The training phase consists of feature extraction and training with a deep denoising autoencoder. Compared to other machine learning algorithms, their work exhibited a high detection rate. Edge computing technology has been applied for reducing the computation time for intrusion detection using LSTM [58]. The assumed attack interface is an on-board unit (OBU) that connects in-vehicle networks to external networks. They executed flooding attacks leading to malfunctions in the CAN network. The LSTM framework for intrusion detection consists of a prediction phase and a detection phase. CAN traffic information consisting of data and time-interval features is passed to the LSTM network. During the prediction phase, features of the CAN message are predicted through data and time-interval analyses. The neural network is trained with real CAN bus messages. The detection phase determines whether a message is anomalous or not. Since the computation time with LSTM is high, the authors proposed using a mobile edge-assisted multi-task LSTM model. The evaluation results showed that the model achieved 90% accuracy with detection latency of 0.61 milliseconds. Xiao et al. [41] proposed a lightweight machine learning algorithm called SIMATT-SECCU. The authors used datasets developed from hacking and a countermeasure lab, which are publicly available [15]. The training is performed with normal and attack datasets. A random forest algorithm is used for the anomaly detection process. The performance evaluation results showed that the model performed better than other state-of-the-art algorithms. A deep learning algorithm for anomaly detection was developed in [59]. The authors developed a simulation model for experimenting with DoS attack detection. The CAN Bus Message Attack Detection Framework (CAN-ADF) was proposed by Tariq et al. [60], which uses a combination of rule-based and Recurrent Neural Network (RNN) anomaly detection models. The dataset was formed by collecting 7,875,792 CAN messages from 24 h of driving two vehicles (Kia Soul and the Hyundai Sonata). A DoS attack was launched by injecting high-priority messages in a short time interval into the CAN bus. The limitation in that work is that the authors did not consider diverse attack types. A data-driven approach to anomaly detection based on the support vector machine (SVM) model was proposed by Al-Saud et al. [61]. The data were gathered from an electric car

and consisted of normal data and DoS-attack data. Comparison results with other machine learning algorithms, such as k-Nearest Neighbor (kNN), the Decision Tree, RBF NN, and a conventional SVM showed that their detection scheme had better intrusion detection accuracy. CANet is an unsupervised intrusion detection system developed by considering the dimensionality of CAN data using LSTM, an autoencoder, and an exponential linear unit (ELU) [62]. This scheme is based on deep learning, and has the ability to detect known and unknown attack types from a stream of CAN messages arriving from the CAN bus at a high rate. The measured dataset and a synthetic dataset were used for experimentation, and 16.5 h of CAN data were used for training, with 7.5 h of CAN data used for testing. Experimental results showed the detection accuracy of the model was very high (up to 99.6%) on both synthetic and real datasets. The fundamental drawback of that model is that neural networks require a huge amount of training data and are computationally expensive. Lokman et al. [63] proposed a deep learning-based anomaly detection scheme for CAN networks using autoencoders. The CAN bus data were collected from three different vehicles (the Toyota Camry, the Hyundai Elantra, and the Perodua Axia) while the vehicles were in driving mode under a DoS attack. Experimental results showed that their model provided an anomaly detection rate of 91% to 100%, which is high compared to state-of-the-art autoencoders. The information entropy-based intrusion detection system has been investigated in several studies [64,65]. In information theory, entropy refers to the average level of information, and uncertainty in the outcome of a random variable. The entropy in CAN bus messages during an attack will change, compared to the entropy calculated during normal operation of the vehicle. This criterion is used to determine whether an intrusion has occurred or not. An unsupervised Kohonen Self-Organizing Map (SOM) approach for the intrusion detection system was proposed in [66]. This intrusion detection system is based on a k-means clustering algorithm that has good features in terms of a high detection rate, minimum training time, and high versatility.

The authors in [43] demonstrated practical fuzzy attacks on a vehicle, and were able to control the engine and brake units. With the attack, they were able to increase the engine RPM, disable the engine, lock up the brakes, etc. Lee et al. [15] tested fuzzy attacks on the Kia Soul. The specific CAN identifier was chosen by analyzing the in-vehicle messages, and a fuzzy attack with arbitrary data was launched. The results of the attack included shaking in the steering wheel, irregular operation of signal lamps, and unintended gear shift changes. Furthermore, the authors suggested an intrusion detection system based on the correlation of offset and time intervals between the request and response messages. Specification-based anomaly detection using CAN timing was proposed in [67]. The specification-based model develops specifications for real-time CAN features, such as timing behavior. The model was developed based on real-time characteristics of the CAN bus, and violation of these specifications is considered anomalous behavior by the IDS. The fundamental idea is that the CAN bus messages appear on the bus in a regular manner, and the authors used a supervised learning algorithm to capture this pattern. The drawback of the model is that it may not detect an anomaly with an aperiodic interval. Time series analysis-based anomaly detection was proposed in [68]. An adaptive cumulative sum (CUSUM) algorithm was proposed to detect change-points in the time-series data. Islam et al. [69] proposed a graph-based IDS that uses the chi-squared test for analyzing data distributions of CAN messages. The distribution of an attack-free dataset was compared to the distribution for an attack dataset. They built the graph based on real CAN message data representing a sequence of messages. The data analysis showed that the attack-free dataset had a Gaussian distribution, whereas the fuzzy attack dataset showed bimodal distribution with different peaks. Experimental results showed that this model has high attack-detection accuracy and a low misclassification rate. The authors claimed that their work was the first graph-based IDS proposed for CAN bus networks, and that it had higher detection accuracy compared to other ID sequence-based methods. A one-class support vector machine-based IDS was proposed in [37]. The fundamental drawback of the algorithms is that they are based on supervised learning. As a result, they

may not detect unknown attack patterns. Furthermore, the algorithms are computationally complex and cannot be handled by resource-constrained in-vehicle computing systems. A lightweight machine learning algorithm using an RNN was proposed in [41], which was further optimized to reduce the computational time of the neural network model. Han et al. [70] proposed an anomaly detection system based on a survival analysis method. This method is based on a statistical time analysis until some event happens. During a fuzzy attack, random CAN packets were sent every 0.0003 s. The authors performed this experiment in a real vehicle by developing software using the PiCAN2 tool connected through the OBD-II port. Using semantic knowledge of the CAN ID function for anomaly detection, their algorithm might be applied to diverse vehicle models. The drawback is that it cannot detect messages injected in an irregular pattern.

Zhang et al. [71] executed replay attacks through experiments on three real cars using various models: the Honda Accord, an Asian brand, and an American brand. Normal data were collected at first from these cars, and a replay attack was launched to create three attack datasets for all the vehicles. They developed an IDS to detect this type of attack through a combination of rule-based and machine learning-based methods. The detection results from the rule-based system, which takes features such as the message ID, the time interval, the message frequency, etc., are passed to a DNN-based system, where they are further processed to detect the type of attack. For anomaly detection, the DNN model uses features such as the message ID, the number of occurrences in the last second, the ID sequence, the relative distance of ID entropy, system ID entropy change, system data entropy change, etc. The experimental results showed that their hybrid IDS had a detection rate of approximately 99% for these datasets from the three different vehicle models. Low-rate replay attacks were implemented by researchers from the University of Louisiana at Lafayette [72]. The replay attack model consisted of four types: dominant ID replay, random ID replay, sequence replay injection, and targeted replay injection. The authors created datasets by accessing the OBD-II port of a 2012 Nissan Leaf, and message injection was performed using the PyCAN (Python CAN) library. They developed an IDS to detect these attacks through a frequency pattern analysis based on a sequence mining technique. The authors argued that the model performed better than a dictionary-based approach. However, the main problem with this model is a longer detection time, which may not be tolerable for real-time anomaly detection. The work in [60] demonstrated replay attacks on the Kia Soul. In that work, the authors selected some IDs and sniffed data with the selected IDs, replaying them over the bus. An attack detection framework was developed based on a combination of an RNN-based method and a rule-based method. The rule-based method is based on message payload distance. In replay attacks, some of the message IDs may be adjacent to each other. Thus, by measuring the distance between those message IDs and deriving a threshold condition for an attack situation, they distinguished replay messages from normal messages. The RNN-based IDS model used an LSTM network with three hidden layers, and ReLU was used for the activation function. Input for the model consisted of 40 consecutive CAN packets, each with 11 features. The output was classified into five different classes: normal messages and four types of attack messages, including the replay attack. Zhang et al. [73] proposed a deep-learning approach to an IDS. The authors used strong correlation parameters, such as vehicle speed, RPM, pressure, throttle position, gear, etc., to train a normal model. The experimental results show that the model had a true positive rate of 98% and a false positive rate of only 1% to 2%. Additional work on replay attacks was shown in [74,75] through the CANoe simulation framework. The work in [74] showed a simulation setup in CANoe, which consisted of a replay node, an adversary node, and an IDS node. Real traffic was collected through the OBD port and was replayed in the simulation framework by the replay node. The authors also provided an intrusion detection system for replay attacks using a kNN algorithm. The work in [75] focused on neural networks for intrusion detection in CAN bus networks. The problems with these approaches are computational complexity and high memory requirements.

The work in [55] investigated spoofing attacks on a hybrid Toyota. The assumption for spoofing attack detection is that fake messages are generated from the attacker node, which deceives the receiver ECUs. First, an attack-free dataset was generated from the car, and attacks were launched through injection of messages using the Python programming language. The spoofing attack dataset consisted of messages injected with spoofed IDs for handle angle and vehicle speed. The authors proposed an IDS based on LSTM networks. The binary classification result on the dataset showed 100% accuracy with spoofing attack patterns. However, the problem with the approach is that the model is based on a supervised machine learning algorithm, which has several limitations. Another study of spoofing attacks on the CAN bus was conducted by Yang et al. [76]. The authors proposed a Recurrent Neural Network Long Short-Term Memory (RNN-LSTM) model for spoofing attack detection based on the features of ECU fingerprint signals. A simulation was performed to generate fingerprint signals of 50 ECUs, and the model was trained through an RNN-LSTM classifier to authenticate an ECU's ID on the CAN bus. A Field Programmable Gate Array (FPGA) was used for real-time intrusion detection. The process starts with deep feature extraction from physical CAN signals for the DNN classifier. If the evaluation score of some ECU ID is greater than a set threshold, it is seen as a legitimate message. Otherwise, it is regarded as a spoofing attack. The result showed that their scheme has a low misidentification rate for spoofing attacks, compared to previous work. The authors of [71] proposed a two-stage IDS based on rule-based and DNN models for spoofing detection. The authors in [67] proposed an IDS to detect spoofing based on timing and a frequency analysis of CAN messages. The experiment results with various datasets showed that the model had a low false positive rate. Avatefipour et al. designed an IDS for spoofing detection based on a one-class support vector machine [37]. The proposed algorithm was modified during the training process in order to generate a better structure for CAN data frequency.

The authors in [50] proposed a clock offset-based IDS for detection of impersonation attacks. Their algorithm measures the clock offset of the transmitter ECU's clock to build a fingerprint of normal clocks. Any deviation in the clock offset from the normal fingerprint offset is considered to signal an anomalous event. The experimental results showed their work outperformed OTIDS and can detect the impersonation attacks within a few seconds. Xiao et al. proposed a lightweight machine-learning algorithm based on an RNN for intrusion detection on the CAN bus [41]. Extensive evaluation using suitable hyper-parameters showed that the model had good performance results, compared to LSTM, the Gated Recurrent Unit (GRU), and Generative Adversarial Networks.

Table 4 compares the various IDSs developed to detect attacks in CAN networks. Here, robustness is defined as the ability of the IDS to detect attacks and thereby minimize the false alarm rate. Detection coverage refers to the various types of attacks that the IDS can detect.

4.2. Security Issues and Countermeasures for Automotive Ethernet

4.2.1. Security Issues

Talic [20] analyzed security issues in the automotive Ethernet protocol. Various security attacks that can be launched against AE were described by Gupta and Lee et al. [77,78]. The attack patterns targeting AE can be classified as shown in Figure 12: DoS, frame injection, replay, spoofing, impersonation, Address Resolution Protocol (ARP) cache poisoning, and TCP hijacking. Some of the attack patterns in AE have similar characteristics to the ones in CAN networks, e.g., replay attacks and spoofing. Frame injection refers to the injection of fake frames into the network through malicious nodes or software programs. DoS is a type of attack in which the communication channel is flooded with invalid data to hinder normal communications. The authors in [20] demonstrated DoS attacks against the Transmission Control Protocol (TCP). The objective of the attack is to hinder the normal firmware flashing process. The flashing process is conducted with Diagnostics over Internet Protocol (DoIP), which runs on top of TCP. A network testing program called

Tcpkill is used to sniff the TCP connections in particular hosts, and which interrupts the connection through the insertion of RST (reset) flagged frames into the communication channel, leading to a connection drop. The result of the attack is disconnection of a TCP session with the error message "Communication failed". ARP cache poisoning is where the attacking node transmits spoofed ARP requests or responses in order to compromise the ARP cache of the node under attack. A demonstration of this attack with non-static ARP tables was offered in [20]. The result showed that the ECU replied to two different method authentication code (MAC) addresses having the same ECHO request after the corresponding ARP replies were received. This proves that ARP cache poisoning is possible under the AE protocol.

Table 4. Comparison of various IDSs for attack detection in in-vehicle networks.

References	Detection Algorithm	Detection Speed (milliseconds)	Detection Accuracy (%)	Algorithm Complexity	Learning Time (seconds)	Robustness	Detection Coverage
Song et al. [39]	DCNN	N.A.	>80	High	N.A.	Medium	DoS, fuzzy, spoofing
Seo et al. [54]	GAN	<500	>90	High	N.A.	High	DoS, fuzzy, spoofing
Hossain et al. [55]	LSTM	N.A.	>90	High	N.A.	Medium	DoS, fuzzy, spoofing
Kang et al. [53]	DNN	<10	>90	High	<15	High	N.A.
Lin et al. [57]	Deep Learning	N.A.	>80	High	N.A.	Medium	DoS, fuzzy, impersonation
Zhu et al. [58]	LSTM	<10	>80	High	N.A.	Medium	Spoofing, replay, flood
Xiao et al. [41]	RNN	N.A.	>90	High	N.A.	High	DoS, fuzzy, impersonation
Song et al. [46]	Time interval based IDS	N.A.	>90	Low	N.A.	High	DoS
Katragadda et al. [72]	Frequency Analysis based IDS	<155	N.A.	Medium	9.04	Medium	Replay

N.A. = Not Available.

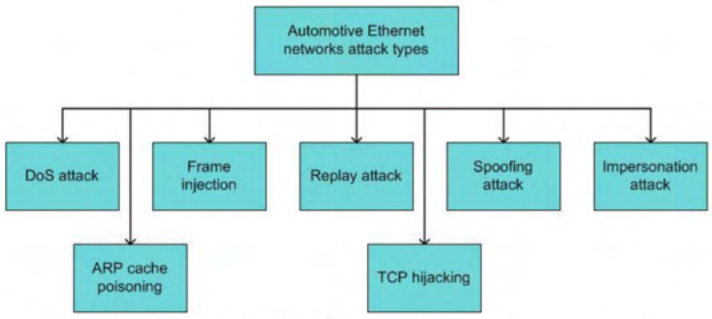

Figure 12. Classification of attack patterns for automotive Ethernet networks.

4.2.2. Countermeasures for Securing Automotive Ethernet

Figure 13 shows the different approaches that have been designed for securing AE networks: anomaly detection, encapsulation and authentication, and encryption.

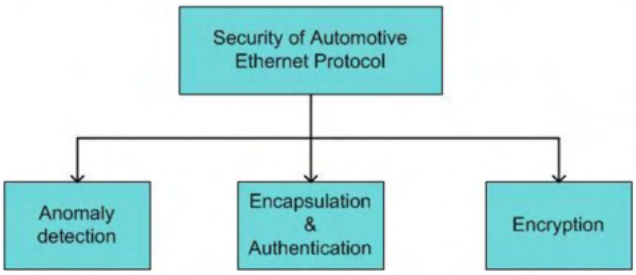

Figure 13. Security solutions for automotive Ethernet networks.

Very little research has been conducted to secure in-vehicle AE networks. The in-vehicle AE is still in its development phase and has not been widely adopted by the vehicle manufacturers. As a result, the security vulnerabilities in AE are not extensively investigated yet. An anomaly detection scheme for automotive Ethernet was proposed by Jeon et al. [10]. They considered the AE architecture based on an Ethernet backbone where domain gateways are used as gateways for CAN and Ethernet networks. The anomaly detection system consists of multiple stages, including data analysis, feature extraction, feature processing, and the actual anomaly detection. The features used for anomaly detection in Ethernet are named Duration, Protocol Type, Src_bytes, Dst_bytes, Count, and Srv_count [10]. Grimm et al. proposed a hybrid anomaly detection system for AE networks using a specification- and machine learning-based method [79]. The authors considered irregularities in message occurrences as the criterion for anomalous behavior in AE networks. The message irregularities may occur if the sender ECU cannot transmit data to a given receiver, or if the message consists of data from unknown protocols. The authors analyzed the performance of three algorithms: Mahalanobis distance, Principal Component Analysis (PCA), and one-class support vector machine (OCSVM) in terms of false positive rate and true positive rate. PCA and OCSVM algorithms can detect attacks with high detection accuracy and a low false alarm rate. However, issues such as feature extraction and real-time anomaly detection should be researched in more detail. Yang et al. [80] proposed a security solution for AE based on encryption and authentication mechanisms. The authors demonstrated, with a CANoe simulation, the secure transmission of video information in AE networks by using authentication and encryption methods. The HMAC-SHA1 algorithm is used for message authentication in Ethernet networks. Based on the reviews, we provide the following recommendations for securing AE-based in-vehicle networks.

- Implementation of firewalls to prevent unauthorized ECUs from sending safety-critical messages
- Intrusion detection systems that can provide prompt feedback to administrators
- Secured on-board communications using authentication and integrity of critical frames based on message authentication code (MAC) with efficient key initialization and management techniques
- Digital signatures and public key infrastructures (PKIs)

Table 5 provides a comparison of various security solutions that have been developed so far for in-vehicle automotive Ethernet networks.

Table 5. Comparison of security mechanisms to detect attacks against in-vehicle Ethernet networks.

References	Security Mechanisms	Detection Speed	Detection Accuracy (%)	Algorithm Complexity	Learning Time (seconds)	Robustness	Detection Coverage
Jeon et al. [10]	Anomaly detection	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
Grimm et al. [79]	OCSVM	N.A.	>90	Medium	N.A.	High	Replay attack
Grimm et al. [79]	PCA	N.A.	>90	Low	N.A.	High	Replay attack
Yang et al. [80]	Authentication and encryption	N.A.	N.A.	Low	N.A.	N.A.	N.A.

N.A. = Not Available.

4.3. Security Issues and Countermeasures for FlexRay

4.3.1. Security Issues

There is a lack of authentication, confidentiality, and data freshness under the FlexRay protocol [25]. Various attacks that can be mounted against this protocol were explained by Shrestha and Kim [19]. Spoofing attacks against this protocol were experimented with by Kishikawa et al. [81]. Similarly, various types of attacks on this protocol were explained by Nilsson et al. [26]. The authors proposed a simulation framework to execute eavesdropping and spoofing against FlexRay networks using the CANoe simulator. The authors were able to take control of brake light functionality in a vehicle and turn the brake lights on, although the vehicle was accelerating and brakes had not been applied. Based on these studies, it is possible to classify security attacks that can be implemented in FlexRay networks, as shown in Figure 14. The flooding attack is a type of DoS. In an eavesdropping attack, the attacker listens to private communications between various ECUs in the network. The drop attack is where a malicious ECU drops a message coming from a legitimate node. In the modify attack, the attacker masquerades as a legitimate node to transmit manipulated messages to in-vehicle networks.

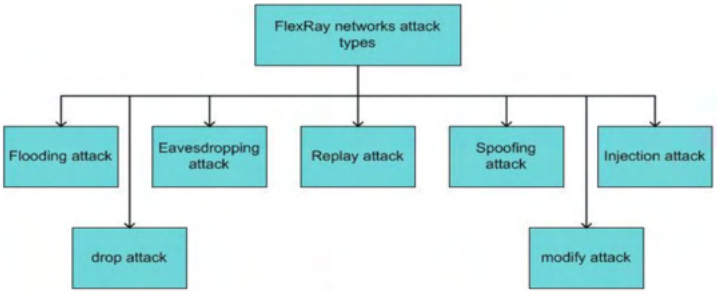

Figure 14. Classification of attack patterns for FlexRay networks.

4.3.2. Countermeasures to Secure the FlexRay Protocol

The possible approaches to securing the FlexRay protocol are shown in Figure 15 [82]. There is very little research focusing on the security attacks and countermeasures for this protocol, despite its vulnerability. Security solutions for the FlexRay protocol can be classified under network design, message authentication, and IDSs. Adopting a bus topology for network design is one of the options to mitigate spoofing attacks. This is because spoofed messages collide with legitimate frames and become invalid frames in this network design. Adopting message authentication using message authentication code can be one option for ensuring the integrity and authenticity of messages. Kishikawa et al. [82] proposed an intrusion detection and prevention system (IDPS) to mitigate spoofing attacks in FlexRay networks. However, the scope of their work is very limited, and IDSs for this protocol need further investigation. A real-time intrusion detection and prevention system is essential for future work. Mousa et al. [83] proposed a lightweight authentication

protocol for secure message exchange in FlexRay networks. Their proposed protocol is scalable, robust, and easy to maintain. Table 6 compares the various mechanisms developed for securing FlexRay networks.

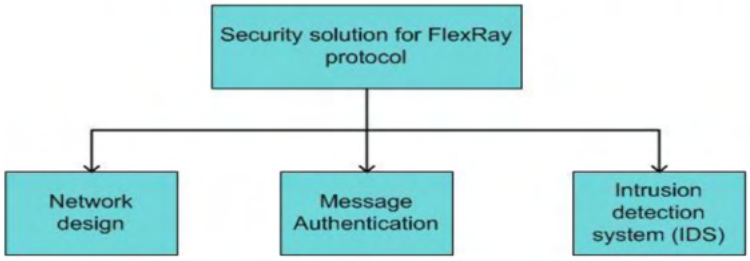

Figure 15. Security solutions for FlexRay networks.

Table 6. Comparison of security mechanisms to detect attacks against in-vehicle FlexRay networks.

References	Security Mechanisms	Detection Speed	Detection Accuracy (%)	Algorithm Complexity	Learning Time (seconds)	Robustness	Detection Coverage
Kishikawa et al. [82]	IDPS	N.A.	Medium	N.A.	N.A.	Low	Spoofing attack
Mousa et al. [83]	Message authentication	N.A.	N.A.	Low	N.A.	High	N.A.
Han et al. [84]	Message authentication	N.A.	N.A.	Medium	N.A.	N.A.	N.A.

N.A. = Not Available.

5. Possible Direction for In-Vehicle Network Security

In this section, we investigate possible directions to improve the security of in-vehicle networks based on blockchain technology. A blockchain is a collection of records called blocks, which are linked with each other sequentially and encrypted. Each block contains a cryptographic hash of the previous block, a timestamp, and the transaction data. It was proposed by Satoshi Nakamoto, and forms the core of Bitcoin [85]. Since then, Blockchain has found huge applications in the field of healthcare data exchange because it is secure, and it maintains privacy in a decentralized system [86]. The node that adds the next block to the blockchain is determined by a consensus mechanism. The privacy of the user is achieved through the use of IDs derived from public keys. A blockchain can be public, a consortium, or private. In this paper, we focus on public and private blockchains. In a public blockchain, everybody can access and read/write transactions to the blockchain [8]. All the nodes in the network can take part in the consensus mechanism. On the other hand, the private blockchain is limited to a single organization, and access rights to transactions are restricted. It is a permissioned blockchain. The transaction processing speed is fast in the private blockchain, compared to the public blockchain, due to the reduced scale and different consensus mechanism.

Developments in Internet of Vehicles technology has created various challenges that cannot be handled by traditional centralized management and data storage systems [87,88]. A large number of IoV (Internet of Vehicles) nodes access a huge network, and create a large traffic load on a centralized system, which may lead to server failure. Blockchain can be a solution to this problem. The demands on blockchain technology for the IoV include big data storage, complete decentralization, true redundancy, use of resources, privacy, service quality enhancement, and cost efficiency [87]. In ref. [87], the authors proposed a blockchain architecture and network model that can be integrated into the IoV environment. The authors focus particularly on the storage of big data in a distributed manner, considering the large amount of traffic generated by IoV devices.

5.1. Application of Blockchain in In-Vehicle Networks

There are several issues with the classic E/E-based architecture using a central gateway. One of the major issues is the single point of failure problem, arising if an attacker compromises the central gateway. We consider a domain-based E/E architecture with several domains interconnected by a central gateway to the central server. However, attackers might compromise domain-based controllers, either by installing unauthorized ECUs or via the OBD II port. To overcome this type of attack, an IDS can be implemented to detect coordinated and distributed attacks. Even the IDS cannot fully secure an in-vehicle system due to the various attack vectors. Thus, integrating blockchain with the IDS can contribute to improved cybersecurity for in-vehicle systems. The central server is integrated with the IDS and the private blockchain, as shown in Figure 16. The IDS generates a system profile based on legitimate information exchanged between participating nodes or ECUs. These profiles are recorded in the private blockchain as transactions that are shared among participating nodes.

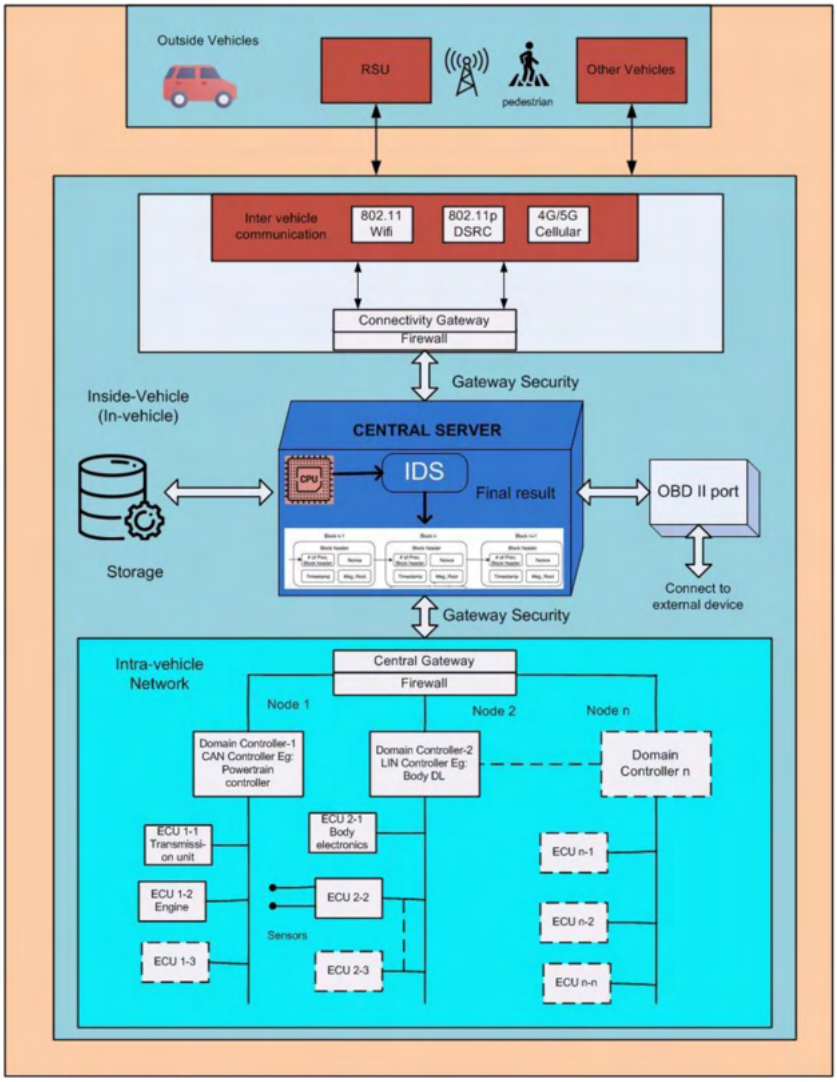

Figure 16. An in-vehicle communication system.

Autonomous vehicles comprise a huge number of sensors, ECUs, cameras, and other equipment, which generate massive amounts of data from onboard or in-vehicle sensors. For safety and security in autonomous vehicles, these massive amounts of data should be processed, analyzed, and exchanged continuously among the various peripherals and

entities inside and outside of the vehicle to prevent critical incidents, such as accidents. In other words, critical information should be exchanged in real time, considering the security issues. To satisfy these conditions, a security system requires a high degree of trust in essential procedures (e.g., vehicle manufacturer firmware updates), and it needs to provide security in real time (e.g., critical information exchange). Blockchain satisfies these conditions based on its high trust level, and it can provide real-time security. Unlike a cryptocurrency, blockchain cannot be used directly as a security tool for autonomous vehicles. A blockchain should be adopted in such a way that it satisfies real-time security needs of in-vehicle and inter-vehicle communications. One such solution is to use a hybrid blockchain.

We consider a hybrid blockchain, so the information can be securely processed within an autonomous vehicle while the vehicle is moving in a decentralized manner. A hybrid blockchain combines the features of the public blockchain and the private blockchain. A public blockchain (PuBC) is used for recording traffic events and V2X communications as transactions, while a private blockchain (PvBC) is used for recording logs of sensitive in-vehicle data and communication. In the hybrid blockchain, the private blockchain for in-vehicle networks can interwork with the public blockchain for external networks through smart contracts. The PuBC can be used to remotely upgrade the vehicle's firmware based on secure Over-the-Air (OTA) updates, and to confirm road event information, etc. Similarly, the PvBC can be used to secure important information exchanged between the ECUs of the vehicle, and for fast validation of sensor information (in a fraction of a millisecond), such as obstacle information, automatic braking system status, etc. In the hybrid blockchain, the private blockchain for the in-vehicle networks communicates with the public blockchain in the external networks through the smart contract. Thus, private in-vehicle data are stored in the private blockchain, whereas the other data that need to be shared with the outside world are stored in a distributed Interplanetary File System (The POA) through smart contracts in the public blockchain.

5.1.1. Private Blockchain for In-Vehicle Networks

A secure vehicular network such as the Internet of Vehicles, based on a hybrid public and private blockchain, will allow a full ecosystem of connected vehicles [89]. In this subsection, we will focus on the in-vehicle blockchain approach. CAN, FlexRay, and Ethernet bus information needs to be protected from multiple threats. Most electric cars incorporate OBD-II interface systems, and can therefore produce reports for self-diagnosis. The OBD-II port provides direct connectivity to all CAN buses inside the vehicle through a physical connection. It collects data, such as noise level, temperature, battery charge, dust levels, etc., from around the vehicle via ECUs and sensors, and sends them to external devices. This interface might put the in-vehicle communication system in danger by allowing hackers to access it.

For the in-vehicle system, there might be one or more CPUs controlling an autonomous vehicle, which act as a central authority to control the activities of other elements. A connectivity gateway is used to connect with V2X systems for inter-vehicle communications. The central gateway is interconnected with a domain-based controller E/E architecture, as shown in Figure 16, where each ECU is under the control of the domain controller. Moreover, all ECUs under each domain gather information from their sensors to process it, make decisions, react based on the decisions, or transmit the processed information to other ECUs.

A PvBC is adopted for the in-vehicle system because the number of CPUs is limited. The private blockchain logs the records of the domain controllers, the ECUs, etc., as shown in Figure 17. On one hand, the private blockchain is fast and uses a simple consensus mechanism; on the other hand, there is no need for an incentive mechanism. The central server may consist of a main CPU, which is considered a supernode, whereas the domain controllers act as nodes for the PvBC. Moreover, the central server consists of several components, such as the CPU, the IDS, the blockchain, etc., that control, manage, and

maintain a list of authorized nodes and ECUs. As stated earlier, the data exchanged between the ECUs and various sensors in the different gateways in intra-vehicle networks are considered transactions, which are similar to transactions in a bitcoin blockchain. The private blockchain structure is similar to the bitcoin blockchain, which consists of the hash value of previous and current blocks, a Merkle tree, and a transaction list for monitoring the history of data exchanges. The central server also verifies and stores the public keys of all the ECUs and nodes (such as various domain controllers). Therefore, unauthorized ECUs cannot flood the in-vehicle network with malicious data. The supernode is responsible for managing the memberships of the blockchain nodes in the PvBC network. It is assumed that all the required vehicle components (DCUs, ECUs, sensors, and other components) are already registered, and that the membership information is stored in the PvBC. The private-chain platforms do not need mining and hence, an alternative consensus mechanism is used, such as Proof of Authority (PoA) [90]. The PoA consensus mechanism uses identity for confirmation of authority, and thus, there is no need for validators to verify the consecutive blocks in the PvBC network. PoA requires very little computation power, and has very low latency, because there is no need for frequent communication between different nodes to reach consensus. In in-vehicle networks, the supernode is assumed to be secure. However, despite using blockchain and firewall as security features in the in-vehicle networks, there are physical attacks and cyberattacks. The attackers are constantly developing new technologies to steal sensitive vehicle data and install malwares. They might attack the supernode via several wireless interfaces, such as Wi-Fi, cellular networks, and multimedia interfaces including OBD II ports. The OBD II ports might be a potential entry point for attackers. If the supernode is hacked, then the attacker might take full control of the vehicle and steal all the private and financial information. If IDS is combined with the blockchain, it can provide additional security. The IDS initially creates a system profile by collecting normal or legitimate information exchanges (i.e., transactions) between the ECUs, nodes, and sensors. The system profile includes information exchanges such as control data, vehicle speed, door lock, multimedia information, etc. Such profiles can be created autonomously based on historical data samples or various AI techniques. The profiles and information are stored in the blockchain as transactions, and the blockchain is replicated among the blockchain nodes, such as ECUs. Then, the IDS monitors newly exchanged messages/transactions between the entities, and after comparing new transactions with stored transactions, the final results are stored in the PvBC. It will generate an alert if it detects a significant diversion from the profile. Any attacks or malicious transactions that alter the standard behavior of the system by manipulating resources or linking to new ECUs outside the trusted network will be considered anomalies.

The attacker can compromise a normal ECU by tampering with its software or by connecting an unauthorized ECU to the in-vehicle network. When the attacker tries to insert malicious information into the CAN bus system, however, the in-vehicle system tries to verify it against the private blockchain. If the information is not consistent with information stored in the private blockchain, the intrusion can be detected by the in-vehicle IDS.

Secure Private In-Vehicle Blockchain Approach

Let us assume that one of the ECUs of the domain controller (like the CAN controller) needs to communicate with the ECUs of other domain controllers like the Local Interconnect Network (LIN) controller as shown in Figure 17. Each ECU transmits to the receiver ECU encrypted data signed with their private key. This provides authenticity and integrity, as well as confidentiality. To be more specific, the transmission unit (i.e., ECU1-1) of the CAN controller (i.e., Node 1) needs to send encrypted data to Body Electronics (i.e., ECU2-1) of the LIN controller (i.e., Node 2). The central server (the supernode) is responsible for monitoring and verifying all domain controllers, ECUs, and sensors [91]. The mechanism of the proposed blockchain-based in-vehicle solution is described as follows.

1. As a first step, the transmission unit (i.e., ECU1-1) sends a request to the CAN controller (i.e., Node 1) for communicating with body electronics (i.e., ECU2-1). All request and response messages are encrypted and signed before transmission.
2. The CAN controller requests authorization from the supernode via the central gateway on behalf of ECU1-1 for communication.
3. The supernode verifies the identities of ECU1-1 and ECU2-1 by checking the stored public keys and their respective signatures.
4. After identity verification, the supernode allows both ECU1-1 and ECU2-1 to communicate with each other. It also provides both ECUs' public keys to their respective domain controllers (i.e., Node 1 and Node 2 in Figure 17).
5. Each domain controller allows their ECU to communicate and shares their respective public keys with the corresponding ECU for further communication.
6. The transmission unit (ECU1-1) encrypts the message with the public key of body electronics (ECU2-1) and signs with its own private key. It then transmits the encrypted message to ECU2-1.
7. ECU2-1 verifies the message received from ECU1-1 by checking its identity, and confirms the received message by sending an encrypted acknowledgement.

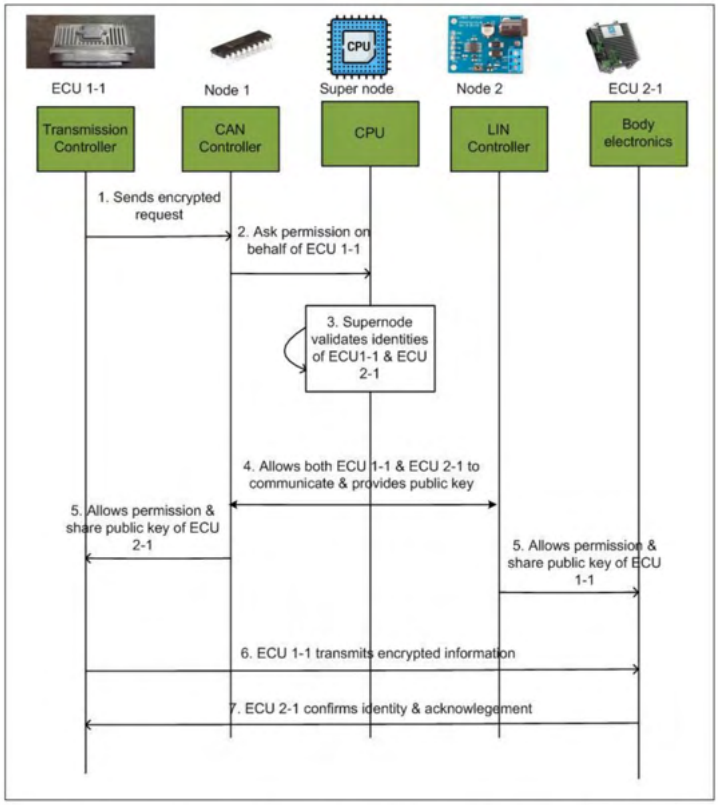

Figure 17. Secure message exchange between engine control units (ECUs) in in-vehicle networks.

The supernode will analyze, verify, and validate all these transactions from authorized ECUs with the help of the IDS. If the IDS detects anomalies in the transactions, an alarm is generated. If it does not detect anomalies, and all nodes reach consensus on validity based on the PoA consensus mechanism, the approved messages are added to the transaction list of a newly created block, and this block is added to the private blockchain. A copy of the block is shared with all member nodes of the private blockchain. However, a large amount of data, such as raw information gathered by the sensors, blackbox information, ECU data, etc., is collected, stored, and updated in the IPFS (Interplanetary File System). Similarly, the PuBC can also store large files related to inter-vehicle communication in the IPFS, which is a completely decentralized system making file exchange over the internet more secure.

The files stored in the IPFS are content addressed so they can easily be retrieved based on the contents. The PuBC only stores the hash of the IPFS files containing the corresponding raw sensor information in the smart contract. Thus, the required information or files can be stored and retrieved easily based on IPFS hashes stored within the public blockchain. Malicious information from an attacker trying to communicate with the in-vehicle system can be detected as an intrusion based on the information recorded in the private blockchain.

5.1.2. Public Blockchain for Inter-Vehicle Networks

Most of the in-vehicle information is very sensitive, so it is recorded in the private blockchain. However, there is information collected by vehicle sensors, or multimedia information, that needs to be exchanged with neighboring vehicles or with edge-cloud devices for event prediction or for offloading data computations. Thus, we need to consider the PuBC for recording safety-related messages, because the centralized or private blockchain method is not appropriate, owing to the single point of failure problem, especially if malicious attackers gain control of the network. The PuBC stores the vehicle's sensor information, the raw and detailed data related to the vehicle (such as maintenance information/history, car diagnostic reports, and event information) as well as reputation information about other vehicles and Roadside Units (RSUs) [92,93]. This information, when recorded in the public blockchain, is transparent and serves as evidence when conflicts occur. The PuBC serves as global ground truth for traffic events, vehicle status, etc. New blocks are created by aggregating a list of unconfirmed messages from the message pool. The unconfirmed messages are the collection of transactions that are not included in the blocks of the main blockchain. The hashes of each block are organized and chained in sequential order to create the blockchain. The public blockchain can use Delegated Proof of Stake (DPoS) as a consensus mechanism to encourage the staking vehicles or nodes to participate in the network by delegating their stakes, as well as to incentivize them with a reward [92,94]. The new block is stored in the blockchain based on the consensus mechanism. The public blockchain holds smart contracts and maintains a list of authorized requestors, such as nodes or vehicles. Moreover, the PuBC can store large files related to inter-vehicle communications in the IPFS, which stores raw sensor data and other information to prevent bloating of the blockchain. The PuBC only stores the hash of the IPFS files with those that correspond to raw sensor information. A requestor with a matching public key can decrypt only certain files in the IPFS. Hence, confidentiality and data integrity are guaranteed while accessing the information in the IPFS. If any vehicle needs to access the required in-vehicle data from the IPFS, the private blockchain owner needs to add the requestor node's public key, privileges, and access requests in a smart contract. The smart contract is stored in the public blockchain, and the contract codes contains rules, conditions, and information related to authorized requestors (e.g., nodes or vehicles) to allow access to the IPFS files. The access policies coded in the smart contract stored in the PuBC provide smooth access to the requested data stored in the IPFS to the authorized nodes. This smart contract automatically executes when all conditions are met, and the required information can be retrieved easily based on IPFS hashes stored in the public blockchain.

We provide a use case of hybrid blockchains for a wireless Over the Air (OTA) software update in intelligent vehicles, as shown in Figure 18. The OTA allows the vehicle's software updates in the ECU software to be done remotely. The centralized OTA update approach is not suitable for thousands of highly distributed vehicles around the world. The decentralized hybrid blockchain allows secure and efficient communication between the participating nodes, eventually leading to installation and update of the software in the intelligent vehicles remotely.

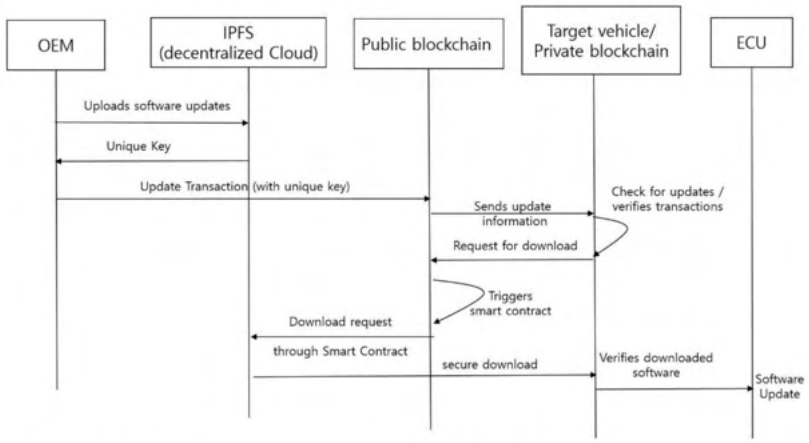

Figure 18. Use case of hybrid blockchain for wireless Over the Air (OTA) software update.

Each intelligent vehicle participates in the blockchain by generating a first transaction that includes basic vehicle information (e.g., vehicle type, production date). Depending upon the situation, the software provider or OEM of the vehicle creates the software update for the ECU after production [95]. The software provider/OEM uploads the software to the IPFS, which acts as a decentralized cloud storage server. The IPFS stores new update software files sent by the OEM. Once the update software is stored in the IPFS, a unique key is supplied to the OEM. It can then be appended to the public blockchain with a transaction. When there is a software update, the OEM sends the update transaction information to the public blockchain and then to the target vehicles, notifying them about the new update. The unique key can be used to retrieve the corresponding update software directly from the IPFS network using HTTP or web-socket interfaces. In IPFS, all the files are linked to hashes and are directly stored in the hash tables [96]. The IPFS provides an access control mechanism so that only the authorized vehicle nodes can download the software update files. The software update transactions contain the related information such as OEM identities, the transaction ID, metadata, etc. It should be noted that only the OEM is allowed to upload the software for security.

A smart contract can be constructed in the public blockchain to run the necessary functions according to specified constraints. When an update alert is sent to each target vehicle, the target vehicle checks the update information in its private blockchain. After verification, it sends the software download request transaction to the PuBC. A smart contract is triggered based on the corresponding transaction information. The smart contract executes autonomously and independently according to the transaction information (i.e., software download). This shows that the smart contract-based blockchain is operating a virtual machine (VM), and the blockchain network functions as a distributed VM. The request and response from the smart contract will be stored as a completed transaction in the public blockchain. The target vehicle downloads the software update files securely from the IPFS through the PuBC. The diagnostic tool in the target vehicle verifies and initializes the downloaded software. It then transfers software to the ECUs and then finally conducts the software update in the corresponding ECUs. The software update information is then stored in the private blockchain.

Similarly, other examples of using a hybrid blockchain is in vehicle accidents, where the insurance company may use the vehicle data for proof and to automate the claims process based on the private blockchain. The insurance company can also be registered in the public blockchain so it can access vehicle information such as model, year, identification number, vehicle owner information, etc., as well as access police reports related to the accident. This helps solve accident and insurance issues easily.

6. Open Challenges and Future Works

The security of the vehicle is of the utmost importance because of the possible loss of life, and damage to property. There is not much advanced attack research that includes message semantics owing to the lack of related datasets for experimentation. Using a car for experimentation in academics is expensive. Thus, there should be more research on a variety of attacks through collaboration among academia, research organizations, and the automotive industry. The open challenges for in-vehicle IDSs are listed below.

- **Supervised learning and labeled datasets:** With the advancements in technology, new attacks are generated, often in a real-time scenario. Deploying a supervised learning model will only be able to detect pre-defined attacks, and the labeling of a dataset is a difficult task because in-vehicle data frames are generated in intervals of milliseconds.
- **Single-layer security:** Most of the existing work focuses on either physical layer or application layer security, and the efficiency of these processes is not sufficient to provide secure data communications for in-vehicle networks.
- **Computational complexity:** The limitations on memory storage and the computation power of in-vehicle computing systems are not considered by the existing research. Most of the existing work used deep learning-based approaches, which require a lot of computation time, compared to the time budget available in practical situations.
- **Diverse attack types:** Most of the prevalent attacks against in-vehicle networks are message injection, which is easy to detect. In the future, hackers might adopt more advanced attacks that cannot be detected easily. For example, attacks might manipulate CAN frame semantics. Thus, an IDS needs to be designed to cover as many attack patterns as possible.

The CAN bus datasets for experimentation are not publicly available from automobile manufacturers because they are considered intellectual property [62]. Thus, there is a need for collaboration between automobile manufacturers and the research community in order to deal with the lack of data for research and academic purposes. Since vehicles can be exposed to diverse types of attacks, and concern for the lives of passengers is increasing, all parties should consider the security of vehicles a high priority. Most IDSs consider attacks executed through physical access using the OBD-II port. Future work should consider the reaction of the vehicle to provide a safe landing after an anomalous event is detected by the IDS [41].

As the number of vehicles connected wirelessly keeps growing, new types of security issues might appear in the near future. One of the open security challenges in a blockchain-based in-vehicle system is the consensus mechanism. The election of a subset of peers for the consensus algorithm is very important, because they are responsible for ensuring the integrity of the data stored in the blockchain. The type of consensus mechanism used for the blockchain will affect the security of the system. In the future, some in-vehicle sensor technology might be interconnected wirelessly, raising issues related to security. An attacker might launch eavesdropping attacks during sensor data transmission, such as from the tire pressure monitor while the vehicle moves, and the attacker might even use reverse engineering to inject false data [97]. Thus, we need to consider wireless sensors and actuators in communications security as well. In the future, we can implement smart contracts that follow specific legal terms to manage and exchange data between domain controllers. The smart contract will execute successfully if all the terms and conditions are met, thus providing enhanced security for the in-vehicle system. Future work should study blockchains for securing in-vehicle networks, such as the CAN bus and AE systems. More research on lightweight machine-learning algorithms for IDSs is needed.

7. Conclusions

This paper investigated in-vehicle network protocols and attacks on security, along with countermeasures to mitigate those attacks. The in-vehicle network is not secure, and adversaries can mount attacks to completely take over controls. Although many intrusion

detection systems have been proposed to protect in-vehicle networks, including CAN, FlexRay, and automotive Ethernet, each scheme has its own limitations in terms of detection coverage, learning time, computational complexity, detection times, robustness, etc. We compared existing approaches for each type of in-vehicle networking protocol. The security of autonomous vehicles is vital and requires more study in order to avoid future hazardous situations. Thus, we argue that future IDSs should employ lightweight algorithms that can be handled by the limited computational capacity of in-vehicle systems. In this paper, we also suggested a hybrid blockchain framework for securing in-vehicle networks. It uses the private blockchain concept to secure message flows among the various sensors and components inside the vehicle, and uses a public blockchain for communicating with the outside world through V2X communications. Since blockchain technology is secure by design, this technology might contribute to the security of in-vehicle networks; we showed an example scenario based on the concept of the hybrid blockchain.

Author Contributions: Conceptualization, S.Y.N. and N.K.; methodology, S.Y.N. and N.K.; validation, N.K. and R.S.; formal analysis, R.S.; investigation, N.K. and R.S.; resources, N.K. and R.S.; data curation, N.K.; writing—original draft preparation, N.K. and R.S.; writing—review and editing, S.Y.N.; visualization, N.K. and R.S.; supervision, S.Y.N.; project administration, S.Y.N.; funding acquisition, S.Y.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (2020R1A2C1010366, 2015R1D1A1A01058595). This research was supported in part by the MSIT, Korea, under the ITRC (Information Technology Research Center) support program (IITP-2021-2016-0-00313) supervised by the IITP (Institute for Information and Communications Technology Planning & Evaluation).

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

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

References

1. Hartenstein, H.; Laberteaux, K.P. *VANET: Vehicular Applications and Inter-Networking Technologies*; John Wiley & Sons: Chichester, UK, 2009.
2. Zeng, W.; Khalid, M.A.S.; Chowdhury, S. In-Vehicle Networks Outlook: Achievements and Challenges. *IEEE Commun. Surv. Tutorials* **2016**, *18*, 1552–1571. [CrossRef]
3. Rathee, G.; Sharma, A.; Iqbal, R.; Aloqaily, M.; Jaglan, N.; Kumar, R. A Blockchain Framework for Securing Connected and Autonomous Vehicles. *Sensors* **2019**, *19*, 3165. [CrossRef]
4. Shrestha, R.; Nam, S.Y. Regional Blockchain for Vehicular Networks to Prevent 51% Attacks. *IEEE Access* **2019**, *7*, 95033–95045. [CrossRef]
5. Greenberg, A. Hackers Remotely Kill a Jeep on the Highway-with Me in It. Available online: https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/ (accessed on 21 July 2015).
6. Kim, S.; Shrestha, R. Security and Privacy in Intelligent Autonomous Vehicles. In *Automotive Cyber Security*; J.B. Metzler: Stuttgart, Germany, 2020; pp. 35–66.
7. Lokman, S.-F.; Othman, A.T.; Abu-Bakar, M.-H. Intrusion detection system for automotive Controller Area Network (CAN) bus system: A review. *EURASIP J. Wirel. Commun. Netw.* **2019**, *2019*. [CrossRef]
8. Lin, I.-C.; Liao, T.-C. A survey of blockchain security issues and challenges. *Int. J. Netw. Sec.* **2017**, *19*, 653–659.
9. Shrestha, R.; Nam, S.Y.; Bajracharya, R.; Kim, S. Evolution of V2X Communication and Integration of Blockchain for Security Enhancements. *Electronics* **2020**, *9*, 1338. [CrossRef]
10. Jeon, B.; Ju, H.; Jung, B.; Kim, K.; Lee, D. A Study on Traffic Characteristics for Anomaly Detection of Ethernet-based IVN. In Proceedings of the 2019 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea, 16–18 October 2019; pp. 951–953.
11. Kimm, H.; Ham, H.-S. Integrated Fault Tolerant System for Automotive Bus Networks. In Proceedings of the 2010 Second International Conference on Computer Engineering and Applications; Institute of Electrical and Electronics Engineers (IEEE), Bali, Indonesia, 19–21 March 2010; Volume 1, pp. 486–490.
12. Nilsson, D.; Larson, U.; Phung, P. Vehicle ECU classification based on safety-security characteristics. In Proceedings of the IET Road Transport Information and Control Conference and the ITS United Kingdom Members' Conference (RTIC 2008), Manchester, UK, 20–22 May 2008; p. 102.

13. Brunner, S.; Roder, J.; Kucera, M.; Waas, T. Automotive E/E-architecture enhancements by usage of ethernet TSN. In Proceedings of the 2017 13th Workshop on Intelligent Solutions in Embedded Systems (WISES), Hamburg, Germany, 12–13 June 2017; pp. 9–13.
14. Navet, N.; Simonot-Lion, F.; Delong, C. *In-Vehicle Communication Networks: A Historical Perspective and Review*; Apple Academic Press: Palm Bay, FL, USA, 2017; pp. 50–51.
15. Lee, H.; Jeong, S.H.; Kim, H.K. OTIDS: A Novel Intrusion Detection System for In-vehicle Network by Using Remote Frame. In Proceedings of the 2017 15th Annual Conference on Privacy, Security and Trust (PST), Calgary, AB, Canada, 28–30 August 2017; pp. 57–5709.
16. Bosch. Can Specifications. 1991. Available online: https://www.kvaser.com/software/7330130980914/V1/can2spec.pdf (accessed on 5 January 2021).
17. Aliwa, E.; Rana, O.; Perera, C.; Burnap, P. Cyberattacks and Countermeasures for In-Vehicle Networks. *ACM Comput. Surv.* **2021**, *54*, 1–37. [CrossRef]
18. Kim, S.; Shrestha, R. Intelligent Autonomous Vehicle. In *Automotive Cyber Security*; J.B. Metzler: Stuttgart, Germany, 2020; pp. 15–33.
19. Kim, S.; Shrestha, R. In-Vehicle Communication and Cyber Security. In *Automotive Cyber Security*; J.B. Metzler: Stuttgart, Germany, 2020; pp. 67–96.
20. Talic, A. Security analysis of ethernet in cars. Master's Thesis, Department of Communication Systems, KTH Royal Institute of Technology, Stockholm, Sweden, 2017. Available online: https://people.kth.se/~{}maguire/DEGREE-PROJECT-REPORTS/171006-Ammar_Talic_with_cover.pdf (accessed on 15 January 2021).
21. Corbett, C.; Schoch, E.; Kargl, F.; Felix, P. Automotive ethernet: Security opportunity or challenge? Sicherheit 2016 - Sicherheit, Schutz und Zuverlässigkeit, Bonn, Gesellschaft für Informatik e.V. 2016, pp. 45–54. Available online: https://dl.gi.de/bitstream/handle/20.500.12116/880/45.pdf?sequence=1 (accessed on 27 January 2021).
22. Hank, P.; Muller, S.; Vermesan, O.; Keybus, J.V.D. Automotive Ethernet: In-vehicle Networking and Smart Mobility. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 18–22 March 2013; Volume 2013, pp. 1735–1739.
23. Glass, M.; Herrscher, D.; Meier, H.; Piastowski, M.; Schoo, P. *"Seis"—Security in Embedded IP-Based Systems*; ATZ Elektron Worldw: Berlin/Heidelberg, Germany, 2010; Volume 5, pp. 36–40.
24. Lee, H.-Y.; Lee, D.-H. Security of Ethernet in Automotive Electric/Electronic Architectures. *J. Inst. Internet Broadcast. Commun.* **2016**, *16*, 39–48. [CrossRef]
25. Le, V.H.; Hartog, J.D.; Zannone, N. Security and privacy for innovative automotive applications: A survey. *Comput. Commun.* **2018**, *132*, 17–41. [CrossRef]
26. Nilsson, D.K.; Larson, U.E.; Picasso, F.; Jonsson, E. A First Simulation of Attacks in the Automotive Network Communications Protocol FlexRay. In Proceedings of the Advances in Computer Science and Education; J.B. Metzler: Stuttgart, Germany, 2008; Volume 53, pp. 84–91.
27. Bozdal, M.; Samie, M.; Aslam, S.; Jennions, I. Evaluation of CAN Bus Security Challenges. *Sensors* **2020**, *20*, 2364. [CrossRef]
28. Wu, W.; Li, R.; Xie, G.; An, J.; Bai, Y.; Zhou, J.; Li, K. A Survey of Intrusion Detection for In-Vehicle Networks. *IEEE Trans. Intell. Transp. Syst.* **2020**, *21*, 919–933. [CrossRef]
29. Tomlinson, A.; Bryans, J.; Shaikh, S.A. Towards viable intrusion detection methods for the automotive controller area network. In Proceedings of the 2nd ACM Computer Science in Cars Symposium, Munich, Germany, 13–14 September 2018.
30. Avatefipour, O.; Malik, H. State-of-the-Art Survey on in-Vehicle Network Communication (CAN-Bus) Security and Vulnerabilities. *arXiv* **2018**, arXiv:1802.01725.
31. Carsten, P.; Andel, T.R.; Yampolskiy, M.; McDonald, J.T. In-Vehicle networks: Attacks, vulnerabilities, and proposed solutions. In Proceedings of the 10th Annual Cyber and Information Security Research Conference, Oak Ridge, TN, USA, 7–9 April 2015.
32. Liu, J.; Zhang, S.; Sun, W.; Shi, Y. In-Vehicle Network Attacks and Countermeasures: Challenges and Future Directions. *IEEE Netw.* **2017**, *31*, 50–58. [CrossRef]
33. Hoppe, T.; Kiltz, S.; Dittmann, J. Security threats to automotive CAN networks—Practical examples and selected short-term countermeasures. *Reliab. Eng. Syst. Saf.* **2011**, *96*, 11–25. [CrossRef]
34. 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*, 1–14. [CrossRef]
35. Miller, C.; Valasek, C. *Remote Exploitation of an Unaltered Passenger Vehicle*; BlackHat: Las Vegas, NV, USA, 2015.
36. Nie, S.; Liu, L.; Du, Y.; Zhang, W. *Over-the-Air: How We Remotely Compromised the Gateway, BCM, and Autopilot ECUs of Tesla Cars*; BlackHat: Las Vegas, NV, USA, 2018.
37. Avatefipour, O.; Al-Sumaiti, A.S.; El-Sherbeeny, A.M.; Awwad, E.M.; Elmeligy, M.A.; Mohamed, M.A.; Malik, H. An Intelligent Secured Framework for Cyberattack Detection in Electric Vehicles' CAN Bus Using Machine Learning. *IEEE Access* **2019**, *7*, 127580–127592. [CrossRef]
38. CAN Bus. Available online: https://en.wikipedia.org/wiki/CAN_bus (accessed on 8 February 2021).
39. Song, H.M.; Woo, J.; Kim, H.K. In-vehicle network intrusion detection using deep convolutional neural network. *Veh. Commun.* **2020**, *21*, 100198. [CrossRef]

40. Biron, Z.A.; Dey, S.; Pisu, P. Real-Time Detection and Estimation of Denial of Service Attack in Connected Vehicle Systems. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 3893–3902. [CrossRef]
41. Xiao, J.; Wu, H.; Li, X. Internet of Things Meets Vehicles: Sheltering In-Vehicle Network through Lightweight Machine Learning. *Symmetry* **2019**, *11*, 1388. [CrossRef]
42. Hoppe, T.; Kiltz, S.; Dittmann, J. Applying intrusion detection to automotive it-early insights and remaining challenges. *J. Inf. Assur. Sec.* **2009**, *4*, 226–235.
43. Koscher, K.; Czeskis, A.; Roesner, F.; Patel, S.; Kohno, T.; Chekoway, S.; McCoy, D.; Kantor, B.; Anderson, D.; Shacham, H.; et al. Experimental security analysis of a modern automobile. In Proceedings of the 2010 IEEE Symposium on Security and Privacy, Berkeley/Oakland, CA, USA, 16–19 May 2010.
44. Checkoway, S.; Damon, M.; Kantor, B.; Anderson, D.; Shacham, H.; Savage, S.; Koscher, K.; Czeskis, A.; Roesner, F.; Kohno, T. Comprehensive experimental analyses of automotive attack surfaces. In Proceedings of the USENIX Security Symposium, San Francisco, CA, USA, 8–12 August 2011.
45. Miller, C.; Valasek, C. *A Survey of Remote Automotive Attack Surfaces*; BlackHat: Las Vegas, NV, USA, 2014.
46. Song, H.M.; Kim, H.R.; Kim, H.K. Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. In Proceedings of the 2016 International Conference on Information Networking (ICOIN), Kota Kinabalu, Malaysia, 13–15 January 2016.
47. Gmiden, M.; Gmiden, M.H.; Trabelsi, H. An intrusion detection method for securing in-vehicle CAN bus. In Proceedings of the 2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), Sousse, Tunisia, 19–21 December 2016.
48. Young, C.; Olufowobi, H.; Bloom, G.; Zambreno, J. Automotive Intrusion Detection Based on Constant CAN Message Frequencies Across Vehicle Driving Modes. In Proceedings of the ACM Workshop on Automotive Cybersecurity, Richardson, TX, USA, 27 March 2019; pp. 9–14.
49. Taylor, A.; Japkowicz, N.; Leblanc, S. Frequency-based anomaly detection for the automotive CAN bus. In Proceedings of the 2015 World Congress on Industrial Control Systems Security (WCICSS), London, UK, 14–16 December 2015; pp. 45–49.
50. Halder, S.; Conti, M.; Das, S.K. COIDS: A clock offset based intrusion detection system for Controller Area Networks. In Proceedings of the 21st International Conference on Distributed Computing and Networking, Kolkata, India, 4–7 January 2020.
51. Alshammari, A.; Zohdy, M.A.; Debnath, D.; Corser, G. Classification Approach for Intrusion Detection in Vehicle Systems. *Wirel. Eng. Technol.* **2018**, *9*, 79–94. [CrossRef]
52. Nazakat, I.; Khurshid, K. Intrusion Detection System for In-Vehicular Communication. In Proceedings of the 2019 15th International Conference on Emerging Technologies (ICET), Peshawar, Pakistan, 2–3 December 2019; pp. 1–6.
53. Kang, M.-J.; Kang, J.-W. Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security. *PLoS ONE* **2016**, *11*, e0155781. [CrossRef]
54. Seo, E.; Song, H.M.; Kim, H.K. GIDS: GAN based Intrusion Detection System for In-Vehicle Network. In Proceedings of the 2018 16th Annual Conference on Privacy, Security and Trust (PST), Belfast, Ireland, 28–30 August 2018; pp. 1–6.
55. Hossain, D.; Inoue, H.; Ochiai, H.; Fall, D.; Kadobayashi, Y. LSTM-Based Intrusion Detection System for In-Vehicle Can Bus Communications. *IEEE Access* **2020**, *8*, 185489–185502. [CrossRef]
56. Hossain, D.; Inoue, H.; Ochiai, H.; Fall, D.; Kadobayashi, Y. Long Short-Term Memory-Based Intrusion Detection System for In-Vehicle Controller Area Network Bus. In Proceedings of the 2020 IEEE 44th Annual Computers Software and Applications Conference (COMPSAC), Madrid, Spain, 13–17 July 2020; pp. 10–17.
57. Lin, Y.; Chen, C.; Xiao, F.; Avatefipour, O.; AlSubhi, K.; Yunianta, A. An Evolutionary Deep Learning Anomaly Detection Framework for In-Vehicle Networks—CAN Bus. *IEEE Trans. Ind. Appl.* **2020**, 1. [CrossRef]
58. Zhu, K.; Chen, Z.; Peng, Y.; Zhang, L. Mobile Edge Assisted Literal Multi-Dimensional Anomaly Detection of In-Vehicle Network Using LSTM. *IEEE Trans. Veh. Technol.* **2019**, *68*, 4275–4284. [CrossRef]
59. Fenzl, F.; Rieke, R.; Chevalier, Y.; Dominik, A.; Kotenko, I. Continuous fields: Enhanced in-vehicle anomaly detection using machine learning models. *Simul. Model. Pr. Theory* **2020**, *105*, 102143. [CrossRef]
60. Tariq, S.; Lee, S.; Kim, H.K.; Woo, S.S. CAN-ADF: The controller area network attack detection framework. *Comput. Secur.* **2020**, *94*, 101857. [CrossRef]
61. Al-Saud, M.; Eltamaly, A.M.; Mohamed, M.A.; Fard, A.K. An Intelligent Data-Driven Model to Secure Intravehicle Communications Based on Machine Learning. *IEEE Trans. Ind. Electron.* **2019**, *67*, 5112–5119. [CrossRef]
62. Hanselmann, M.; Strauss, T.; Dormann, K.; Ulmer, H. CANet: An Unsupervised Intrusion Detection System for High Dimensional CAN Bus Data. *IEEE Access* **2020**, *8*, 58194–58205. [CrossRef]
63. Lokman, S.F.; Othman, A.T.; Musa, S.; Abu Bakar, M.H. Deep Contractive Autoencoder-Based Anomaly Detection for In-Vehicle Controller Area Network (CAN). *Prop. Charact. Mod. Mater.* **2019**, *119*, 195–205. [CrossRef]
64. Baldini, G. On the Application of Entropy Measures with Sliding Window for Intrusion Detection in Automotive In-Vehicle Networks. *Entropy* **2020**, *22*, 1044. [CrossRef] [PubMed]
65. Wu, W.; Huang, Y.; Kurachi, R.; Zeng, G.; Xie, G.; Li, R.; Li, K. Sliding Window Optimized Information Entropy Analysis Method for Intrusion Detection on In-Vehicle Networks. *IEEE Access* **2018**, *6*, 45233–45245. [CrossRef]
66. Barletta, V.S.; Caivano, D.; Nannavecchia, A.; Scalera, M. Intrusion Detection for In-Vehicle Communication Networks: An Unsupervised Kohonen SOM Approach. *Futur. Internet* **2020**, *12*, 119. [CrossRef]

67. Olufowobi, H.; Young, C.; Zambreno, J.; Bloom, G. SAIDuCANT: Specification-Based Automotive Intrusion Detection Using Controller Area Network (CAN) Timing. *IEEE Trans. Veh. Technol.* **2019**, *69*, 1484–1494. [CrossRef]
68. Olufowobi, H.; Ezeobi, U.; Muhati, E.; Robinson, G.; Young, C.; Zambreno, J.; Bloom, G. Anomaly Detection Approach Using Adaptive Cumulative Sum Algorithm for Controller Area Network. In Proceedings of the ACM Workshop on Automotive Cybersecurity, Richardson, TX, USA, 27 March 2019; pp. 25–30.
69. Islam, R.; Refat, R.U.D.; Yerram, S.M.; Malik, H. Graph-Based Intrusion Detection System for Controller Area Networks. *IEEE Trans. Intell. Transp. Syst.* **2020**, 1–10. [CrossRef]
70. Han, M.L.; Kwak, B.I.; Kim, H.K. Anomaly intrusion detection method for vehicular networks based on survival analysis. *Veh. Commun.* **2018**, *14*, 52–63. [CrossRef]
71. Zhang, L.; Shi, L.; Kaja, N.; Ma, D. A two-stage deep learning approach for can intrusion detection. In Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium, Novi, Michigan, 7–9 August 2018.
72. Katragadda, S.; Darby, P.J.; Roche, A.; Gottumukkala, R. Detecting Low-Rate Replay-Based Injection Attacks on In-Vehicle Networks. *IEEE Access* **2020**, *8*, 54979–54993. [CrossRef]
73. Zhang, J.; Li, F.; Zhang, H.; Li, R.; Li, Y.; Jiayan, Z.; Fei, L.; Haoxi, Z.; Ruxiang, L.; Yalin, L. Intrusion detection system using deep learning for in-vehicle security. *Ad Hoc Netw.* **2019**, *95*, 101974. [CrossRef]
74. Jichici, C.; Groza, B.; Murvay, P.-S. Integrating Adversary Models and Intrusion Detection Systems for In-vehicle Networks in CANoe. In Proceedings of the Lecture Notes in Computer Science; J.B. Metzler: Stuttgart, Germany, 2020; pp. 241–256.
75. Jichici, C.; Groza, B.; Murvay, P.-S. Examining the Use of Neural Networks for Intrusion Detection in Controller Area Networks. In Proceedings of the Lecture Notes in Computer Science; J.B. Metzler: Stuttgart, Germany, 2019; pp. 109–125.
76. Yang, Y.; Duan, Z.; Tehranipoor, M. Identify a Spoofing Attack on an In-Vehicle CAN Bus Based on the Deep Features of an ECU Fingerprint Signal. *Smart Cities* **2020**, *3*, 2. [CrossRef]
77. Gupta, A. Security Risk Analysis of Automotive Ethernet Networks. 2017. Available online: https://essay.utwente.nl/73894/1/Gupta_MS_EEMCS.pdf (accessed on 10 February 2021).
78. Lee, T.-Y.; Lin, I.-A.; Liao, R.-H. Design of a FlexRay/Ethernet Gateway and Security Mechanism for In-Vehicle Networks. *Sensors* **2020**, *20*, 641. [CrossRef]
79. Grimm, D.; Weber, M.; Sax, E. An Extended Hybrid Anomaly Detection System for Automotive Electronic Control Units Communicating via Ethernet—Efficient and Effective Analysis using a Specification- and Machine Learning-based Approach. In Proceedings of the 4th International Conference on Vehicle Technology and Intelligent Transport Systems, Funchal, Madeira, Portugal, 16–18 March 2018; pp. 462–473.
80. Yang, H.; Liu, M.-Z.; Xu, Y.-H.; Wu, Y.-J.; Xu, Y.-N. Research of Automotive Ethernet Security Based on Encryption and Authentication Method. *Int. J. Comput. Theory Eng.* **2019**, *11*, 1–5. [CrossRef]
81. Kishikawa, T.; Hirano, R.; Ujiie, Y.; Haga, T.; Matsushima, H.; Fujimura, K.; Anzai, J. Vulnerability of FlexRay and Countermeasures. *SAE Int. J. Transp. Cybersecur. Priv.* **2019**, *2*, 21–33. [CrossRef]
82. Kishikawa, T.; Hirano, R.; Ujiie, Y.; Haga, T.; Matsushima, H.; Fujimura, K.; Anzai, J. Intrusion detection and prevention system for flexray against spoofed frame injection. In Proceedings of the 17th Escar Europe: Embedded Security in Cars Conference (Konferenzveröffentlichung), Detroit, MI, USA, 19–20 November 2019; pp. 59–73. [CrossRef]
83. Mousa, A.R.; Noureldeen, P.; Azer, M.; Allam, M. Lightweight Authentication Protocol Deployment over FlexRay. In Proceedings of the 10th International Conference on Predictive Models in Software Engineering, Turin, Italy, 17 September 2014; pp. 233–239.
84. Han, G.; Zeng, H.; Li, Y.; Dou, W. SAFE: Security-aware flexray scheduling engine. In Proceedings of the 2014 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 24–28 March 2014.
85. Zheng, Z.; Xie, S.; Dai, H.-N.; Chen, X.; Wang, H. Blockchain challenges and opportunitiees: A survey. *Int. J. Web Grid Serv.* **2018**, *14*, 352–375. [CrossRef]
86. Qadri, Y.A.; Nauman, A.; Bin Zikria, Y.; Vasilakos, A.V.; Kim, S.W. The Future of Healthcare Internet of Things: A Survey of Emerging Technologies. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1121–1167. [CrossRef]
87. Jiang, T.; Fang, H.; Wang, H. Blockchain-Based Internet of Vehicles: Distributed Network Architecture and Performance Analysis. *IEEE Internet Things J.* **2019**, *6*, 4640–4649. [CrossRef]
88. Kim, S.; Shrestha, R. Internet of Vehicles, Vehicular Social Networks, and Cybersecurity. In *Automotive Cyber Security*; J.B. Metzler: Stuttgart, Germany, 2020; pp. 149–181.
89. Cebe, M.; Erdin, E.; Akkaya, K.; Aksu, H.; Uluagac, S. Block4Forensic: An Integrated Lightweight Blockchain Framework for Forensics Applications of Connected Vehicles. *IEEE Commun. Mag.* **2018**, *56*, 50–57. [CrossRef]
90. Liu, X.; Zhao, G.; Wang, X.; Lin, Y.; Zhou, Z.; Tang, H.; Chen, B. MDP-Based Quantitative Analysis Framework for Proof of Authority. In Proceedings of the 2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), Guilin, China, 17–19 October 2019.
91. Salem, M.; Mohammed, M.; Ali, R. Security Approach for In-Vehicle Networking Using Blockchain Technology BT—Advances in Internet, Data and Web Technologies. In Proceedings of the International Conference on Emerging Internetworking Data & Web Technologies, United Arab Emirates, 26–28 February 2019.
92. Shrestha, R.; Bajracharya, R.; Shrestha, A.P.; Nam, S.Y. A new type of blockchain for secure message exchange in VANET. *Digit. Commun. Netw.* **2020**, *6*, 177–186. [CrossRef]

93. Shrestha, R.; Bajracharya, R.; Nam, S.Y. Blockchain-based Message Dissemination in VANET. In Proceedings of the 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), Kathmandu, Nepal, 25–27 October 2018; pp. 161–166.
94. Xiao, Y.; Zhang, N.; Lou, W.; Hou, Y.T. A Survey of Distributed Consensus Protocols for Blockchain Networks. *IEEE Commun. Surv. Tutorials* **2020**, *22*, 1432–1465. [CrossRef]
95. Steger, M.; Dorri, A.; Kanhere, S.S.; Römer, K.; Jurdak, R.; Karner, M. Secure Wireless Automotive Software Updates Using Blockchains: A Proof of Concept. In *Road Vehicle Automation 3*; J.B. Metzler: Stuttgart, Germany, 2017; pp. 137–149.
96. Hasan, S.S.; Sultan, N.H.; Barbhuiya, F.A. Cloud Data Provenance using IPFS and Blockchain Technology. In Proceedings of the SCC '19: Proceedings of the Seventh International Workshop on Security in Cloud Computing, Auckland, New Zealand, 7–12 July 2019.
97. Rouf, I.; Miller, R.; Mustafa, H.; Taylor, T.; Oh, S.; Xu, W.; Gruteser, M.; Trappe, W.; Seskar, I. Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In Proceedings of the USENIX Security Symposium, Washington, DC, USA, 11–13 August 2010.