Cyberattacks and Countermeasures for In-Vehicle Networks

EMAD ALIWA, OMER RANA, CHARITH PERERA, and PETER BURNAP,
Cardiff University, UK

As connectivity between and within vehicles increases, so does concern about safety and security. Various automotive serial protocols are used inside vehicles such as Controller Area Network (CAN), Local Interconnect Network (LIN), and FlexRay. CAN Bus is the most used in-vehicle network protocol to support exchange of vehicle parameters between Electronic Control Units (ECUs). This protocol lacks security mechanisms by design and is therefore vulnerable to various attacks. Furthermore, connectivity of vehicles has made the CAN Bus vulnerable not only from within the vehicle but also from outside. With the rise of connected cars, more entry points and interfaces have been introduced on board vehicles, thereby also leading to a wider potential attack surface. Existing security mechanisms focus on the use of encryption, authentication, and vehicle Intrusion Detection Systems (IDS), which operate under various constraints such as low bandwidth, small frame size (e.g., in the CAN protocol), limited availability of computational resources, and real-time sensitivity. We survey and classify current cryptographic and IDS approaches and compare these approaches based on criteria such as real-time constraints, types of hardware used, changes in CAN Bus behaviour, types of attack mitigation, and software/ hardware used to validate these approaches. We conclude with mitigation strategies limitations and research challenges for the future.

CCS Concepts: • **Security and privacy** → *Intrusion/anomaly detection and malware mitigation*;

Additional Key Words and Phrases: CAN bus, cybersecurity, intrusion detection systems

ACM Reference format:
Emad Aliwa, Omer Rana, Charith Perera, and Peter Burnap. 2021. Cyberattacks and Countermeasures for In-Vehicle Networks. *ACM Comput. Surv.* 54, 1, Article 21 (March 2021), 37 pages.
https://doi.org/10.1145/3431233

1 INTRODUCTION

In recent years, vehicles have become more connected (to other vehicles – referred to as Vehicle-2-Vehicle (V2V) and external infrastructure—referred to as Vehicle-2-Infrastructure (V2I)) and the cyberattack surface for these vehicles continues to increase. Cyberattacks have also become a real concern for vehicle manufacturers, especially where services need to be supported using networks outside a vehicle. These services can include Global Positioning Systems (GPS), On-Board Diagnostic– (OBD-2) based cellular dongles, and entertainment services. As a result, vehicles are now more vulnerable to attacks not only from inside but also from outside the vehicle. For instance, recent report has indicated that two famous connected cars are vulnerable to cyberattacks [141]. As

Authors' addresses: E. Aliwa, O. Rana, C. Perera, and P. Burnap, Cardiff University, UK, 5 The Parade, Cardiff, UK, CF24 3AA; emails: {aliwaem, ranaof, pererac, burnapp}@cardiff.ac.uk.

0360-0300/2021/03-ART21 $15.00
https://doi.org/10.1145/3431233

Table 1. Related Survey Papers on Vehicular Networks Security

Coverage	Other Papers	This Paper
Focus on In Vehicle Network security (IVN)	[43, 46, 62, 90, 129, 161]	✓
Analysis of existing cryptographic methods for securing IVNs	[62, 90]	✓
Analysis of existing IDS methods for securing IVNs	[43, 62, 90]	✓
Introducing countermeasure & architecture for vehicles security	[90, 129, 161]	
Limitation with current CAN Bus countermeasures		✓
Datasets, software and hardware used		✓

potential cyberattacks on vehicles have widened, more vulnerabilities and entry points have been discovered—generally grouped under: direct interfaces-initiated attacks, infotainment-initiated attacks, telematics-initiated attacks, and sensor-initiated attacks. This raises the need for better security mechanisms. Due to lack of suitable security support in the Controller Area Network (CAN) protocol itself, mechanisms to secure communications between components within a vehicle is also limited. Attacks such as CAN Bus Denial of Service (DoS) and bus injection attacks are common [27]. CAN Bus security limitations have been investigated by various researchers over both laboratory-based environments and in real vehicles. The attacks demonstrate how attackers are able to successfully take control of various parts of a vehicle, such as brakes, lights, steering and gears [23]. Such attacks and malicious data on the CAN Bus was generated from both on-board the vehicle and at remote locations.

Serial protocols are used for in-vehicle networks to exchange parameters between Electronic Control Units (ECUs) and sensors. These protocols lack security mechanisms by design and are thus vulnerable to various attacks. Researchers have also shown how to attack vehicles from within a vehicle using direct interfaces and infotainment systems via the OBD-2, USB and CD player and from outside the vehicle using medium and long distance communication such as Wi-Fi, Bluetooth, mobile (phone) networks, and sensors signals such as keyless fob attacks and tyre pressure monitoring system sensors. These attacks have widened the potential attack entry points within a connected vehicle—suggesting the importance of protecting the CAN Bus.

To ensure focus, this survey does not cover a number of other related topics, such as driving behaviour, unauthorised drivers, software defined/updated ECUs, and Blockchain for vehicular security. Table 1 provides a comparison between this article and other papers covering vehicular security.

This survey provides the following contributions: (i) description of in-vehicle serial bus protocols (particularly the CAN Bus), (ii) evaluation of current cryptographic and Intrusion Detection Systems (IDS) approaches used for protecting vehicular data, (iii) comparison and assessment of current mitigation strategies to protect vehicles against cyberattacks, and (iv) challenges and potential future research directions for in-vehicular cybersecurity. The rest of this article is structured as follows: Section 2 provides an introduction to serial data exchange protocols within a vehicle, outlining key concepts and terminology, and providing the context for the rest of this article. In Section 3, the CAN Bus protocol and architecture along with hardware and software used inside vehicles utilising the CAN Bus is described. In Section 4, we describe the connected car infrastructure, including various ECUs and sensors that can be used inside a vehicle. In Section 5, attacks initiated using data interfaces, telematics, infotainment, and sensor entry points and how such attacks can be generated are described. In Section 6, we review security mechanisms to secure the CAN Bus, which include encryption, message authentication, and vehicular IDS. We evaluate existing approaches based on criteria such as real-time data requirements, the types of network

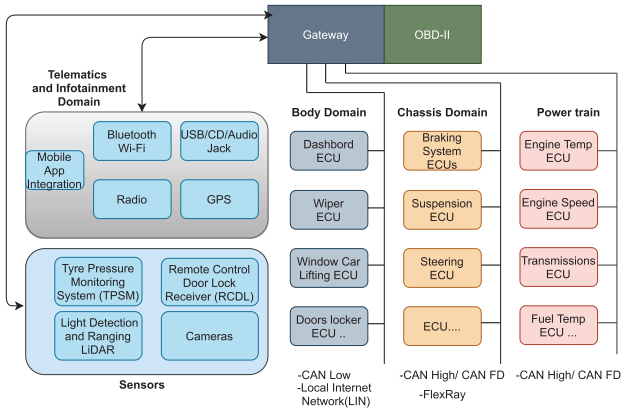

Fig. 1. Serial bus protocols inside a vehicle from Reference [54]. The figure focuses on the three most popular protocols: CAN, LIN, and FlexRay.

infrastructure required, computational resource constraints, modifications needed for vehicular hardware, change in the protocol behaviour, types of attacks mitigated, and software/ hardware used to validate these approaches. Finally, in Section 7 we evaluate mitigation strategies and limitations mentioned in Section 6 and conclude with possible research directions.

2 AUTOMOTIVE SERIAL BUS PROTOCOLS

Three key protocols are used inside vehicles for data communications: CAN, FlexRay, and Local InterConnect Network (LIN) (Figure 1). The CAN Bus protocol is the most widely used to support critical functions such as power train, engine management, anti-brake system, and transmission. A vehicular system is divided into four domains, in terms of the function it performs and the whether it requires real-time data [99]: (i) the *Power train domain*, which includes engine and transmission functions—this domain is critical and needs real-time response; (ii) the *Chassis domain*, which includes the braking system, suspension, and steering, which also provides real-time and safety critical functions inside the vehicle; (iii) the *Body domain* for functions such as the dashboard, wipers, lights, windows, and seats—these functions do not generally require real-time response; and (iv) the *Telematics and infotainment domain*, which manages the various communications, information, and entertainment services inside a vehicle, e.g., in-car navigation, CD/ DVD players, rear seat entertainment systems, driving assistance, and wireless interfaces. These domains differ in terms of the functions they provide and the performance and quality of the network they require. Based on these differences, each domain has different performance and response time requirements.

2.1 Controller Area Network

CAN Bus is serial communication protocol used for real-time, safety critical functions inside road vehicles and other controlled applications [57]. It is a multi-master protocol and most widely used inside vehicles [67]. Below are the main characteristics of the CAN protocol: (i) The maximum bitrate is 1 Mbps in the *classical* CAN Bus with a payload up to 8 bytes, with the high-speed CAN Bus bitrate varying from 125 kbps to 1 Mbps and the low-speed CAN Bus bitrate from 5 kbps to 125 kbps; (ii) in the CAN Flexible Data (FD) rate, the bitrate is up to 8 Mbps with a payload size of up to 64 bytes in CAN FD. CAN XL [21] is the third generation, providing up to 2,048 bytes of data payload and a bitrate of up to 10 Mbps [119]—it is expected to be used with Internet Protocol–based services; (iii) critical sensors can be connected to the high-speed

Table 2. Automotive Serial Communication Protocols

Protocol	Bitrate	Application	Domain	Standard
High CAN	125 Kbps to 1 Mbps	Real time critical applications e.g., engine and braking systems	Powertrain and Chassis train	ISO 11898
Low CAN	5 kbps to 125 kbps	Non-critical such as doors and windows	Body Domain	ISO 11898
CAN FD	Up to 10 Mbps	Critical real time applications	Powertrain and Chassis train	ISO 11898
LIN	1 kbps to 20 kbps	Non-critical applications	Body Domain	ISO 17987
FlexRay	up to 10 Mbps	Critical applications	Powertrain and Chassis	ISO 17458

The table shows Controller Area Network (CAN and CAN FD), LIN, and FlexRay.

CAN Bus while less-critical sensors can be connected to a low-speed CAN Bus; (iv) the CAN Bus protocol provides real-time access for critical sensors via the Carrier Sense Multiple Access with the Collision Avoidance and Arbitration Priority (CSMA/CA-AP) access/arbitration mechanism; (v) the CAN Bus is implemented physically using the twisted pair cable; (vi) the CAN Bus is used in real-time automotive applications such as engine management, transmission, braking system, and steering; and (vii) the CAN Bus provides error detection and correction mechanisms. Since the CAN Bus protocol is the most widely used, this review article will focus on this protocol in terms of attacks, vulnerabilities, and potential countermeasures.

2.2 Local Interconnect Network

The LIN protocol (International Organization for Standardization (ISO) standard 17987) is used for low-cost vehicular applications that have a low-bitrate, asynchronous data requirement. It is used for non-critical applications such as door module and air condition systems [60]. LIN is used where the reliability and robustness of the network is not critical [126]. It is standardised in ISO 17987 series, which has been sub-divided into seven sub-standards describing the protocol functions aligned with the OSI layers such as physical layer, data link frame, and so on [60]. The LIN protocol has the following features: (i) a bitrate ranging from 1 to 20 kbps; (ii) a single master with multiple slave nodes with support for up to 16 nodes; and (iii) a single physical wire to realise the bus.

2.3 FlexRay

This is a protocol used for high-bitrate requirements, with error detection and correction, redundancy, and safety [126]. It is used for high-end applications inside vehicles such as power train and safety functions (adaptive cruise control and active suspension) [97]. The protocol is standardised under ISO 17458 series that describes the physical layer and data link layer characteristics [61]. The main features of the protocol are as follows: (i) a bitrate of up to 10 Mbps with half-duplex bus access; (ii) support for fault tolerant mechanisms; and (iii) designed to work for high-speed and safety-critical applications (e.g., braking-by-wire and steering-by-wire). Vulnerabilities and countermeasures of FlexRay can be found in References [79, 92, 93, 102, 115], and a comparison with CAN Bus in References [33, 42, 114, 116].

3 CONTROLLER AREA NETWORK

Initial use of CAN Bus within a vehicle did not consider security, as vehicles at that time were not connected to an outside networks. The CAN protocol does not have security features and is vulnerable to attacks such as frame injection and denial of service. Nowadays, vehicles are more

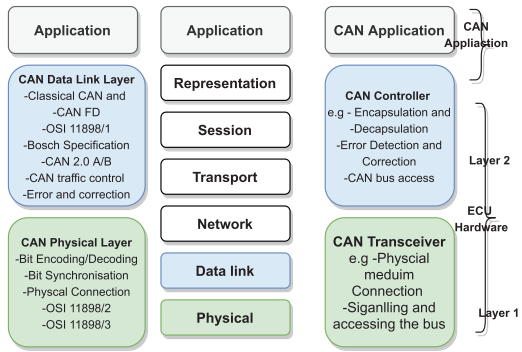

Fig. 2. CAN Bus protocol in OSI Model. CAN Bus Hardware, Software, and Standards in OSI layers.

connected, and they have internal and external interfaces such as Wi-Fi, Bluetooth, and mobile (phone) networks, and thus cybersecurity has become a real concern. CAN is a protocol invented in the early 1980s by Bosch GmbH and used widely inside vehicles to send and receive data between ECUs and sensors [4]. It was standardised in ISO 11898 series and it works as a serial bus, indicating that any node on the network (e.g., an ECU) can use the network bus to send data via a multi-master mechanism using two wires [28]. This reduces the cost of wiring compared to point-to-point wiring and reduces the negative effects of external noise through its CAN-High and CAN-Low (signal differential) transmission [4]. It works on the physical and data link layers, although it does not use Media Access Control (MAC) addresses to send and receive (route) frames. Instead, it uses message identification (ID) (does not have sender or receiver addresses compared to Ethernet) and a broadcast half duplex mechanism to transmit data over the bus. Also, it does not verify and use authentic messages as it sends data based on message id and not source address. CAN Bus controller inside the vehicle connects critical parts of the vehicle such as engine and body control modules, such as gears, speed, brakes, and so on. The protocol consists of two versions: the classical CAN protocol and CAN FD protocol (Flexible Data rate)—both protocols are defined and standardised under ISO 11898 series [12].

3.1 Cybersecurity and Safety Standards for Vehicle Networks

Various efforts already exist to provide security and safety for vehicles, from design to production and operation. Such efforts originate from standards organisations such as ISO, the Society of Automotive Engineering (SAE), and other organisations. Some of the main standards for automotive cybersecurity include the following: (i) *ISO / SAE 21448*: This standard provides a guide from measurement to design to enable the verification and validation of services inside vehicles [58]. (ii) *SAE J3061*: This provides a guide for cyber-physical systems within vehicles and has been provided by the Society of Automotive Engineering [134]. (iii) The *SAE J3101* standard provides the required features to support hardware security protection for vehicular applications [121]. (iv) *SAE J3138* (Diagnostic Link Connector Security): This is a standard used for diagnostics and security purposes for direct interfaces such as the OBD-2 port inside a vehicle [135]. (v) *ISO / SAE 21434:* This standard is used to support road vehicle cybersecurity engineering [72]. It is a shared effort between ISO and SAE covering security for road vehicles, in-vehicle systems, components, software, and communication with external networks and devices. This standard aims to provide guidelines for vehicle manufacturers and suppliers from design to production phases [72].

Other initiatives to secure in-vehicle network infrastructure include the following: (i) *E-safety Vehicle Intrusion Protected Applications (2008–2011)* [125]: This project focuses on securing in-

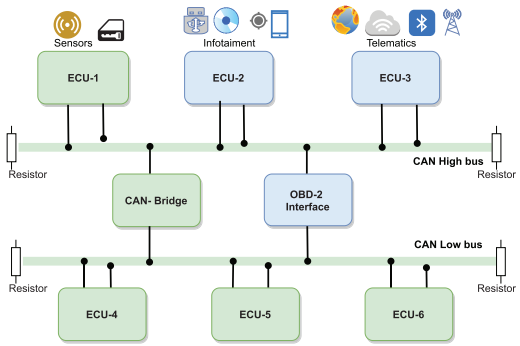

Fig. 3. Two CAN Buses connected to each other through a bridge based on Reference [152].

vehicle network infrastructure from physical and remote attacks. (ii) *Secure Vehicular Communication (2006–2008)* [127]: This has focused on securing V2I networks, with an emphasis on wireless data communications that is used to transmit vehicle parameters to an outside network or device. (iii) *Cooperative Vehicle-Infrastructure Systems (2006–2010)* [24]: This is a framework to provide V2I security and privacy for drivers and connected vehicles.

3.2 Electronic Control Units

Modern cars have around 70 ECUs that control various functions of the car, such as brakes, gears, and engine status [126]. An ECU is primarily a microprocessor that contains a CAN controller used to support data link layer functions and a CAN transceiver used for physical layer functions such as frame delivery, error detection and correction, and other data link layer tasks [86].

The CAN protocol uses a broadcast-based mechanism for message exchange [149] and each node can request use of the bus randomly. An arbitration mechanism is used to ensure priority on the bus [35], as ECUs with critical functions such as engine, transmission, and braking systems usually have higher priority to access the bus and require least broadcast frequency [146]. Priority is based on comparing the arbitration id of requesting nodes, and the node with higher priority is granted access to send data on the bus. Inside the vehicle, ECUs with critical functions (e.g., brakes and steering) can be connected to a high-speed CAN Bus while ECUs with low importance (e.g., windows) can be connected to low-speed CAN Bus. Both CAN Buses are connected through a gateway ECU [27].

With this structure, any ECU with connection to the bus, using an OBD-2 port or Bluetooth, can sniff and inject data into both buses. These vulnerabilities have led to the development of IDSs and firewalls to prevent unauthorised access, as well as using cryptography methods to provide confidentiality, integrity, and authentication.

The CAN Bus protocol has been used in many application areas due to its simplicity and offered bitrate. The ease of implementation, low cost, and the small number of physical wires need to realise it makes it suitable for use in many embedded system [35]. It is used inside vehicles, built environments (e.g., controlling elevators in buildings, building energy management systems), railway applications, medical devices, and aircrafts [98].

4 CONNECTED VEHICLE ENVIRONMENT

Connected vehicles can simply mean a vehicle connected to a network and providing services such as vehicle diagnostic parameters and GPS information to the vehicle owner. According to Juniper research, connected cars are expected to increase to 750 million by 2023 [64]. This connectivity will be through telematics or by in-vehicle applications. Vehicles can be connected with either

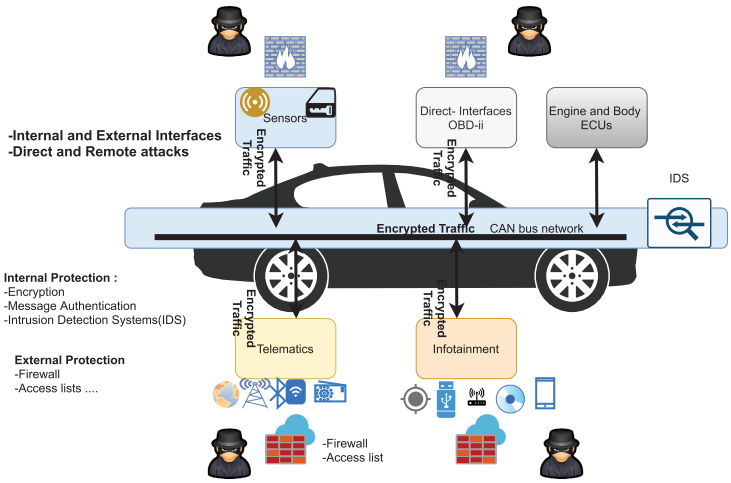

Fig. 4. Connected Car environment with four potential entry points for data injection: Telematics, Infotainment, Direct interfaces, and Sensors. Also, security countermeasures to detect and prevent physical and remote attacks using Cryptography, IDS, Firewalls, and Access Control Lists.

aftermarket tools such as OBD-2 cellular device, GPS device used in fleet management, or hardware and software included from the vehicle Original Equipment Manufacturer (OEM). Components used in connected cars can be classified as follows: (i) *Telematics Unit:* Provides connectivity to the car using WiFi, Bluetooth, GPS, and mobile data interfaces. (ii) *Infotainment Unit:* This provides the information and entertainment to the driver through a head display unit such as CD, DVD player, USB, and mobile applications integration with the head unit. (iii) *Driver assistance Unit:* This provides the driver and the vehicle with driving assistance hardware such as cameras and LiDAR sensors to provide safety on the road. Also, these sensors are used to support autonomous driving and auto park assistance. (iv) *Vehicle-2-X*: Connected cars can also provide communication to cars (V2V) and roadside infrastructure (V2I) using Dedicated Short Range Communications (DSRC) that allow exchange of data such traffic conditions between cars and/or road side unit.

4.1 Connected Vehicle Interfaces and Sensors

Bluetooth provides connectivity with mobile apps hosted on devices operated by passengers. Such a service involves pairing a mobiles phone(s) with head unit inside the vehicle [16]. This can lead to vulnerabilities from the Bluetooth connection as legacy and vulnerable versions are still used—as described in Reference [15]. *Wi-Fi:* Connected cars provide wireless connectivity for various services, such as providing internet through Wi-Fi on board. Wi-Fi has a number of vulnerabilities, e.g., via a Wi-Fi hotspot on a Jeep [88] and a Tesla S [68]. *Cellular/phone network*: Modern cars can also provide mobile/phone/cellular connection that can be used to retrieve data such as weather conditions and traffic [88]. Attacks on such networks have also been identified [16]. For example, Reference [88] has shown how a cellular network interface can be hacked inside a Jeep. *OBD-2*: This is a mandatory port that is used for capturing diagnostic and environmental (e.g., emissions) data. This interface is directly connected to the vehicle's CAN Bus network and by using an aftermarket OBD dongle and attaching it to the OBD port, it is possible to initiate various attacks such as a DoS attack that can affect vehicle operation and driver safety [34]. Various attacks have been demonstrated using direct connection to an OBD port and generating remote attacks using wireless OBD dongles [16]. In 2018, a remote attack on a vehicle was initiated using a custom hardware

that was connected to a CAN Bus over OBD-2 port. This customised board used a SIM card, and the attacker sent malicious SMS messages to inject this data into the CAN Bus [162].

GPS is used to provide driving assistance and positioning for drivers. Further, it is used by fleet management to monitor vehicle location. This interface can provide an entry point for an attacker both for injecting and sniffing data [107]. In Reference [88], GPS information was retrieved from the head unit of a vehicle through unprotected 6667 port. A *Compact Disc (CD)* player is used in the head unit for entertainment purposes. It has been shown that this unit is directly connected to the internal data network of a vehicle and also susceptible to cyberattacks, as described in Reference [16]. *Sensors:* Sensors and actuators are used inside vehicles to support various functions such as sensing engine temperature. Physical availability attack can be initiated using signal jamming [82] to block data between the sensors and the CAN network. Incorrect sensor values can also be injected into the CAN Bus to modify the behaviour of the ECUs that operate on this data. A particular type of sensors used for Tyre Pressure System Monitoring (TPSM) are connected to each tyre to monitor pressure and send real-time data to an ECU [88]. Attack on TPSM is described in Reference [59], where the authors were able to perform eavesdropping attacks from 40 m on a passing car. *LiDAR and Camera:* Cameras and laser signals are used inside vehicles to provide safety and driving assistance. These components can lead to attacks such as signal jamming, e.g., Reference [111] that reports signal jamming attacks on LiDAR and cameras. *Keyless entry:* Miller and Valasek [88] show how Remote Control Door Lock Receiver within a vehicle is directly connected to the internal CAN Bus. It receives the signal from the key fob to lock/unlock doors and trunk of the vehicle. Keyless entry attacks were initiated to steal vehicles and has been shown to be the most used attack between 2010 and 2019 [123]. This attack can be initiated by jamming the signal between the key fob and the vehicle to keep the doors open while the owner of the car thinks it is closed. Also, it can be initiated by capturing the key fob signal and redirecting it to the vehicle. For example, in Reference [29] researchers were able to hack key fob block cipher and perform relay signal attack and were able to lock and unlock doors. The attacker needs to be in the range of the key fob to be able to intercept the signal for this type of attack.

5 VULNERABILITY OF IN-VEHICLE CAN BUS

The intention of using a CAN Bus inside a vehicle was to reduce cost, simplify installation, and improve efficiency for real-time communication. However, as mentioned previously, a CAN Bus has a number of security vulnerabilities [23]:

- The network is not segmented, as all nodes (ECUs) are connected to the same bus. The CAN Bus protocol uses a broadcast mechanism to transfer data, which means all nodes on the network can send and receive the same messages.
- There is no security mechanisms used for authentication and thus the CAN Bus is vulnerable to message poisoning and DoS attacks.
- The traffic on the CAN Bus is not encrypted and can be easily read through a data sniffing attack. Every ECU connected to the bus can therefore sniff CAN frames due to the broadcast mechanism.
- An ECU can make the CAN Bus in domination status using the arbitration scheme (Message ID priority scheme) and thus prevent other ECUs from using the bus—which can lead to DoS attack.
- It is not possible to know whether an ECU has sent or received certain messages (non-repudiation).
- Access to the CAN Bus network via external interfaces and connections such as OBD-2, Wi-Fi and Bluetooth widens the potential attack surface (and entry points) [158].

Table 3. In-Vehicle EntryPoints

CAN Bus initiated attack	Entry Points	Physical remote	Attack Mechanism	Position of the attacker	Result of the attacks
Interfaces Initiated attacks	• OBD-2 • USB • CD	• Physical • Physical • Physical	• OBD-2 direct connection • USB direct connection • CD direct connection	• Inside/Outside • Inside • Inside	• Full access • Sniffing • Injection
Infotainment and telematics initiated attacks	• USB • CD Player • BT • Wi-Fi • Cellular • GPS	• Physical • Physical • Remote • Remote • Remote • Remote	• Direct Connection • Direct connection • Unauthorised access to BT • Wi-Fi unauthorised access • Access cellular interface • Access GPS information	• Inside • Inside • Outside • Outside • Outside • Outside	• Inject CAN • Inject CAN • Inject and sniffing • Inject and sniffing • inject and sniffing • inject and sniffing
Sensor initiated attacks	• TPSM • Key fob • LiDAR	• Remote • Remote • Remote	• Decode and replay • Intercept and relay • Jamming LiDAR signals	• Outside • Outside • Outside	• Attack TPSM sensors • Unlock doors • Block driving assistance

Entry-points for data injection are grouped into Direct interfaces, Telematics and Infotainment attacks and Sensor-based attacks.
OBD-2: On-Board Diagnostics; BT: Bluetooth; USB: Universal Serial Bus; GPS: Global Positioning System; TPSM: Tyre Pressure Monitoring System; LiDAR: Light Detection and Ranging.

There has been an increase in the number of cyberattacks on vehicles, increasing 7 times in 2019 compared to 2010 and doubling in 2019 compared to 2018 [123]. The vulnerable points could be classified as direct, indirect, short-range, and long-range attacks [16]. The CarShark software was used to sniff, analyse, observe, and replay data on the CAN Bus using OBD-2 connector and then control the wheels, brakes, and other ECUs and components of the vehicle [70]. Other entry points to the CAN Bus inside the vehicle were also identified, e.g., audio jack, USB, and Wi-Fi. Similarly, other attacks were performed from outside the car identifying potential vulnerabilities [16]. Table 3 shows potential entry points. Other examples include attacks on Toyota Prius 2010 [87] and Ford Escape 2012 vehicles by physically connecting to the OBD-2 port (and CAN Bus) and controlling vehicle speed, brakes, and steering. Other examples include remote attacks carried out on a Jeep Cherokee [88]. Another attack is the keyless fob attack used to forcibly unlock the doors of a vehicle [29]. A summary of in-vehicle network-based attacks in provided in Table 3.

Table 4 identifies some of the attacks initiated on real vehicles and simulated environments. The table identifies entry points used, how attacks were initiated, the position of the attacker, the outcome of the attacks, and the software/ hardware test environment used.

5.1 Attacks against the CAN Bus

The classical CAN and CAN FD buses are vulnerable to various attacks. Once attackers have access from either inside or outside the vehicle, they can generate various attacks on the CAN Bus network such as CAN sniffing, CAN fuzzing, CAN replay, and DoS attacks. Some of the mechanisms for initiating these attacks include the following.

CAN Bus sniffing: With no authentication mechanisms, encryption and broadcast transmission, it is possible to sniff the data on the CAN Bus [23]. Using off the shelf OBD2 sniffer such as CANdo board [100], it is possible to read and analyse the data on the bus to manipulate and generate similar messages [23]. This attack can be avoided by implementing encryption to prevent exposing CAN frames. This attack is difficult to detect due to the passive nature of sniffing traffic. The next step is to reverse engineer the raw CAN messages so that they can be used to target specific parts of the vehicle. This an important step, since manufacturers tend not to publish their CAN message specification [67].

Table 4. Attacks on In Vehicle Networks

Authors	Initiated Attacks	Entry Points	Position of the Attackers	Attack Result	Test Environment
[29] 2008	Sensors	• Keyfob Keyless entry system	• Outside	• Lock,unlock door • and start the engine	• Real vehicles
[70] 2010	Interfaces Infotainment	• OBD-2 • USB • CD Player	• Inside (Direct) • Inside • Inside	• CAN Bus injection • Full access	• Real vehicles
[51] 2011	Interfaces	• OBD-2	• Inside	• Control Window car lifting • Warning light • and airbag	• Parts of a vehicle • such as • instrument cluster, • window car lifting • and head unit ECUs • CANoe simulator
[16] 2011	Interfaces Infotainment Telematics	• OBD-2 • Cellular • BT • CD Player • Radio	• Inside • Outside • Outside • Inside • Outside	• Get access to CAN Bus • Disable parts of the vehicle	• Real vehicles
[120] 2012	Sensors	• TPSM	• Outside	• Inject with • False TPSM values • and signal jamming	• Real vehicles
[88] 2015	Interfaces	• OBD-2	• Inside • Outside	• Control brakes, • Wheels and • Get access to the CAN Bus	• Real vehicles
[111] 2015	Sensors	• LiDAR • Cameras	• Outside • Outside	• Signal jamming	• LiDAR Hardware • CAN software
[88] 2015	Telematics	• WiFi	• Outside	• Unauthorised access • Inject CAN message • to stop the engine	• Jeep Cherokee
[68] 2016	Telematics	• WiFi	• Outside	• Full access to CAN Bus	• Tesla model S
[162] 2018	Interfaces	• OBD-2 Cellular Dongle	• Outside	• CAN Bus injection	• Real vehicle

Grouped according to data of publication. Entry of attacks, position of attackers, result, and test environment are also provided.

CAN Bus fuzzing attack: CAN Bus protocol lacks authentication and data integrity checking and as a result ECUs accept CAN messages and respond to them. This attack is used to send random CAN data frames, checking the bus and observing changes on the instrument panel of the vehicle. This attack looks at the impact of CAN frames on the ECUs such as observing the change in vehicle speed while injecting CAN frames [69]. It usually happens after sniffing and analysing captured CAN messages. Also, it can be generated using a black-box, where CAN ID and payload values are generated randomly without prior knowledge of the actual CAN ID used. It involves sending randomly captured CAN frames and recording the outcome. Encryption is needed to prevent analysis of the captured data, along with authentication to only accept CAN frames from legitimate ECUs.

CAN Bus frame falsifying attack: This attack is used to modify CAN message payload by inserting incorrect values. For example, the attacker can inject a vehicle with incorrect parameter values. This type of modification attack is used when the CAN ID is known, and the intention is to provide incorrect data payload to disrupt vehicle services. This happens due to the lack of data integrity and authentication support in the CAN Bus protocol. To prevent this attack, CAN Bus should provide authentication to verify the source of the data before acting upon it. Usually

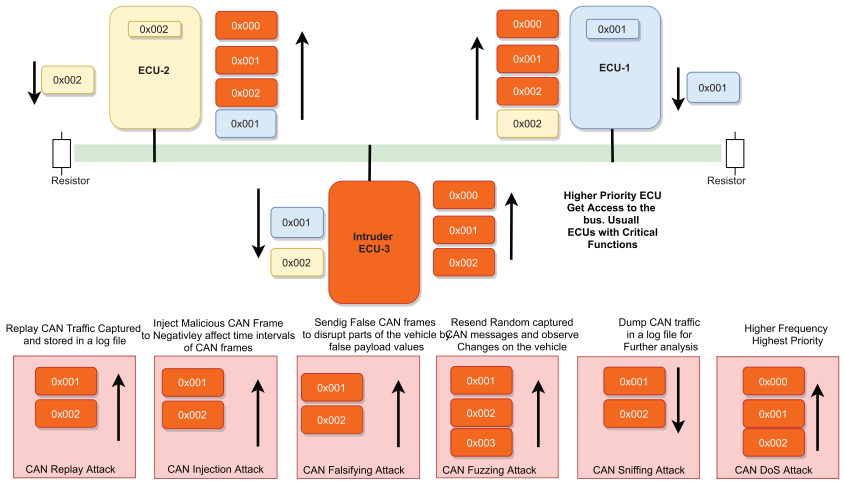

Fig. 5. Overview of some In Vehicle CAN Bus Network (IVN) attacks.

this attack involves a small amount of data, making it difficult to detect and monitor. To detect this attack, a system should consider checking CAN ID and data payload consistency in a time window.

CAN Bus injection attack: Injecting data into a CAN Bus can be used to send messages at an abnormal rate [83]. The purpose of this attack is to change frequency and amount of CAN frames on the bus, and change the sequence of legitimate CAN frames and data payload. Since CAN Bus does not provide authentication to check if the sender is legitimate, this attack will inject the bus with abnormal CAN traffic targeting the vehicle speed. Lack of encryption also enables arbitrary nodes to connect to the bus. The data on the bus can then be monitored to obtain the arbitration and data field and generate messages to simulate events [22]. This could lead to generation of fake events that cause parts of the vehicle to behave as required by the attacker. This attack can be prevented using authentication and integrity mechanisms. The result of the attack can increase the broadcast frequency of certain CAN ID that can be detected through abnormal broadcast behaviour.

CAN Bus DoS attack: Classical CAN and CAN FD use the same mechanism to access the medium with multi access using the CAN ID priority [54]. The nodes on the CAN Bus use the arbitration field to determine the priority of the message and which node can occupy the bus and send data. In this case, a DoS attacked can be lunched using highest attribution id such as 0x000 to occupy the bus and make it busy by using CAN frame priority arbitration scheme and send too many highest priority frames so that other nodes cannot use the bus [88]. Also, it can use the same CAN message id of an existed ECU and by knowing its transmission rate, a DoS can be performed by incrementing the frequency time. For example, if an ECU sends a message every 200 ms, then the attacker can increase the frequency by injecting the same message with higher frequency that can lead to disruption of the sensor part.

ECU impersonation: Once an attacker has access to the CAN Bus network, the attacker can receive all the traffic broadcast on the bus. With a focused analysis of the traffic, attackers can learn the behaviour of each ECU such as its CAN ID, payload range, and transmission rate. In this way, they can simulate ECU behaviour by sending the same data with the same frequency. An increase in the CAN messages rate will occur that generates an attack. However, if the attack was more focused, then they could initiate an attack to disable particular ECUs. For example, Iehira

et al. [55] introduced a sophisticated spoofing ECU attack by first performing an attack on an ECU by taking advantage of the error handling mechanism of the CAN Bus protocol. This attack works by mimicking the target ECU behaviour, CAN ID, and frequency. Then, the attacker ECU contradicts the target ECU by sending a dominant bit while the original ECU sends a recessive bit. This would raise an error in the ECU controller that leads, at a certain point, to disconnecting the ECU from the bus and dropping all the CAN Bus communication. This enables an attacker to perform various attacks, such as an ECU impersonation attack, which is difficult to detect.

6 CAN BUS SECURITY MECHANISMS

Implementing and testing security of CAN Bus traffic has been conducted by many researchers. In this section, we identify current countermeasures used and divide them based on the mechanisms they use, and whether these are used from within or outside the vehicle. We also consider factors such as the test environment, security metric being considered, countermeasure used, the type of mitigated attacks, overhead of supporting the countermeasure, and utilization.

6.1 In-Vehicle Network Cybersecurity

Given the limited capacity of the CAN Bus, any countermeasures used to address its vulnerabilities should consider this limitation and not overload the bus. Security solutions for a CAN Bus can be divided into encryption, authentication, and redesign of the protocol stack by replacing fields in the frame, splitting the message to multiple frames, or by adding nodes and components to the bus to realise additional capability. These approaches can be costly to deploy. Cryptography-based methods have focused on securing the CAN Bus from malicious messages, while IDS focus on the detection of malicious messages. Firewall and Intrusion Prevention Systems can be used in external interfaces to block access to the bus. Implementing a dedicated node to realise the IDS and firewall may be required.

6.2 Using Cryptography

Implementing cryptography in the CAN Bus requires additional computational resources in the ECUs and the CAN Bus controller. Cryptography can be used to provide authentication and data integrity through MAC and confidentiality through symmetric and asymmetric cryptosystems. For in-vehicle networks, a key challenge is to create a secure method that does not alter the payload size (e.g., splitting the message can lead to more load on the bus) and response time latency that would affect vehicle safety [117].

Lightweight encryption is needed in such embedded systems, due to limited computational capacity within ECUs inside the vehicle. The approach used in the classical CAN Bus protocol involves creating a small MAC tag size, less than 8 bytes, and inserting it along with the actual data payload. This tag provides integrity and authentication as it is encrypted by a shared secret key. Session keys are used for authentication and to prevent subsequent re-play attacks. Key distribution is a concern in CAN Bus broadcast environments and therefore a pre-loaded key in each ECU can be used to establish key exchange and freshness to tackle data broadcast and to avoid bus loading due to key exchange. Also, to tackle the issue of low computing resources, Hardware Security Module (HSM) can be used in resource-constrained ECUs to provide better encryption/ decryption time. However, these approaches can still be costly to realise within existing vehicles. In summary, the adopted approaches involve the following: (i) Using lightweight MAC to overcome resource constraints in ECUs and making use of a small key size and MAC signature; (ii) implementing changes in the CAN protocol standard by replacing fields such as CRC with MAC signature or by extending the protocol data field (called CAN+) and extending the data payload to 16 bytes to give more space for the MAC signature (however, this approach leads to compatibility issues);

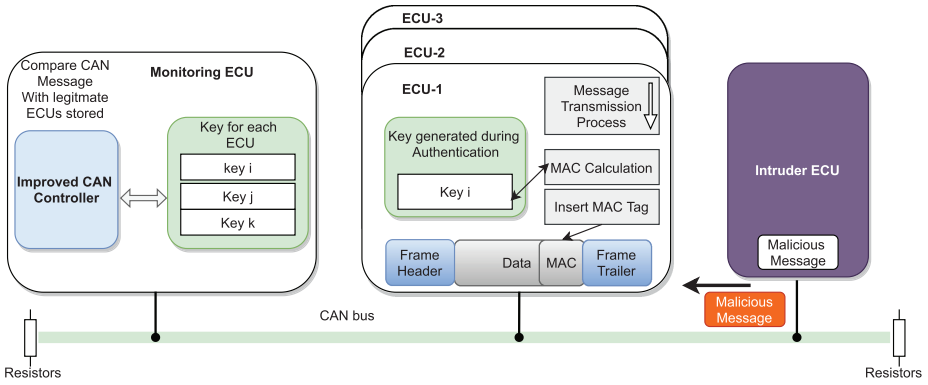

Fig. 6. CAN message authentication and dedicated monitoring ECU. Derived from Reference [147].

and (iii) an HSM [154] can be used (as additional hardware) to overcome computational resource limitations (however, costs of these HSMs can vary depending on security requirements and cost constraints) [109].

6.2.1 CAN Frame Authentication. Authentication mechanisms can be used to provides authentication and integrity of the data for an in-vehicle network. This mechanism is called MAC. However, this approach does not provide confidentiality, which means CAN traffic is still exposed to sniffing and reverse engineering attacks. Thus, a combination of MAC and encryption is needed to provide better security. The following approaches can provide authentication and integrity of CAN Bus data but change the behaviour of the protocol by splitting CAN frames, replacing fields, increasing CAN frame size, or increasing bus payload and response time. Some approaches also require additional hardware that can increase the cost of implementation and lead to incompatibility with current vehicles.

Nilsson et al. [101] introduced an approach based on a shared 128-bit key between ECUs and a Cipher-Block Chaining Message Authentication Code (CBC-MAC) using KASUMI [30] encryption algorithm. Their approach provides integrity and authentication through a 64-bit MAC tag. However, it splits a single CAN ID message into multiple messages to insert 16-bits inside the CRC field. This means that four messages are needed to send the 64-bit tag. As a result, their approach increases the bus load [44] by increasing the number of CAN Bus messages and it changes the CAN protocol behaviour by replacing the CRC field with MAC tags. Furthermore, it causes more latency through CAN message splitting.

Wang and Sawhney [152] proposed a trusted group-based technique to enforce access control while minimizing the distribution of keys through a pre-calculated cryptographic function. Their approach was able to successfully prevent message injection attacks while message processing delay was approximately 50 μs. They used Freescale's automotive boards [104] to test their solutions. Trusted group ECUs share a secret symmetric key (K h), and ECUs in the trusted group can hold keys of other groups if needed. This approach is effective, since it separates telematics and OBD-2 ports, which are the main entry points for attackers. However, their authentication approach is achieved by sending data and authentication messages for each CAN ID, which doubles the bus load. This can increase the CPU overhead by 2,000 additional clock cycles [3]. Further, authors have not provided details about bus overhead [103]. Another approach, CANAuth, used lightweight authentication and counters to mitigate injection and reply attacks [49]. The authors considered the limitations of the CAN Bus protocol and used lightweight encryption mechanisms to mitigate attacks, but DoS attacks were not investigated and a CAN+ protocol (16 bits) was used

that is incompatible with standard CAN Bus specifications. The proposed approach uses HMAC function with pre-shared symmetric and group keys for key distribution. It uses 15 bytes for HMAC flag and key transmission while 1 byte is used for the actual data. LCAP in Reference [47] used a one-way function to provide a 2-byte magic number. The authors proposed using their approach in the data field (2 bytes of 8 bytes) in the standard CAN frames 2.0A. In the extended CAN, they used magic number of 16 bits in the header field of the extended header 29-bit CAN frame 2.0B. Their approach provides authentication and integrity through symmetric keys and HMAC magic number. LCAP provides protection against re-play attacks due to the changeable magic number and session keys. A drawback is that it is based on the CAN+ (16 bytes of data) that raises compatibility issues as the CAN transceiver hardware needs modification to handle the 16 bytes of data payload. LibrA-CAN is a broadcast authentication protocol using the MD5 Message Digest, compatible with CAN+ specification introduced in Reference [38]. Multiple receivers can hold keys and provide authentication roles based on monitoring their own message ID usage inside the CAN Bus. This approach splits CAN messages into normal CAN messages and authentication tag messages, which increases bus traffic [7]. For improvement, they suggest using CAN+, but the CAN transceiver hardware should be changed to be able to handle the new CAN+ frame.

In Reference [9], the authors used a combination of SHA3 and HMAC function along with session keys to avoid re-play attacks. This method used the length in the CRC fields to insert a cryptographic checksum. The processing time of sending and receiving a message was not provided in this approach. Also, there is a change in compatibility of the CAN frame specification by replacing CRC field with MAC tag field. CaCAN in Reference [73] is used to carry out authentication and validate integrity of CAN messages. This is achieved by using a main "Monitor" ECU that shares keys with other ECUs. Using the broadcast behaviour of the CAN us, it receives all messages and can detect and overwrite unauthorised messages. This approach does not provide confidentiality, and additional hardware is needed as a monitoring ECU node. LeiA was introduced in Reference [117], which used 128-bit key, MAC, and counter-based algorithms to authenticate data and generate counters to mitigate re-play attacks. This algorithm does not require any changes to the hardware and topology of the CAN Bus. However, it is compatible only with CAN 2.0B 29-bit extended frame and changes the CAN header by replacing the 18-bit identifier with a counter—potentially leading to incompatibility issues with current vehicle networks. In Reference [65], the authors introduced a one-way hash chain using HMAC-MD5 and Advanced Encryption Standard 128 (AES-128) to provide authentication. They tested their approach on simulated ECUs using a CANoe Vector [148] tool and Freescale S12XF as CAN hardware. Re-play and spoofing attacks were considered in this approach. They used the symmetric key and Authentication Key Exchange Protocol2 (AKEP2) and assumed that the symmetric key and the ID of the sender are stored during manufacture. They have demonstrated only a limited overhead (bus load and latency) when using their approach. In Reference [147], the authors have used HMAC-SHA256 to provide ECU authentication and data integrity. Their MAC tag size is 1 byte and it is inserted along with the actual data payload and a counter size of 4-bit to prevent re-play attacks. They include a "Monitor" ECU that receives all the messages on the bus and checks if they are legitimate by holding all the keys of the ECUs. In case of an illegitimate message, they send a remote frame to overwrite the malicious message.

While the above approaches provide authentication and integrity for the CAN Bus protocol, they suffer from other limitations such as backward incompatibility, real-time constraints (delayed authentication), or cost of implementations by using dedicated hardware. Therefore, a software-based approach should focus on providing authentication and integrity without failing in these shortcomings. The approach proposed by Fassak et al. [32] made use of an asymmetric key. HMAC is then used with changeable session keys. The authors assume that both public and private keys

Table 5. Frames Authentication for Controller Area Network

Authors	Method	Attacks mitigated	Hardware Security Module	Real time	Bus load	Change CAN Bus	Test bed environment	C,I,A
[101] 2008	CBC-MAC	Injection spoofing	No	Delayed authentication	Multiple CAN frames	Splitting CAN frame for authentication purposes	Theoretical	No, Yes,No
[49] 2011	HMAC Symmetric key Counters	Spoofing Injection Replay	No	Yes	16 bytes of data payload	CAN+ 16 Bytes All nodes must Know pre-shared key	Theoretical	No,Yes,No
[47] 2012	One way function Magic number of 2 bytes Session keys	Replay Injection	No	Yes, but Consume time during key distribution	Add extra~ 2 Bytes in~ the payload	HMAC tag in the 2.0B CAN frame~header	Starter-TRAK TRK-MPC5604B board	Yes,Yes,No
[38] 2012	MD5 LM-MAC	Replay Injection	No	No	Yes	Split CAN messages Using CAN+	InfineonTriCore controllers Freescale S12X cores	No,Yes,No
[152] 2014	Trusted Group HMAC Symmetric key	Spoofing Injection	Pre-load keys During manufacturing	Yes	Message Splitting	Split CAN frames for authentication purposes	Freescale's automotive boards EVB9S12XEP100	No,Yes,No
[73] 2014	HMAC Symmetric keys Counter	Replay Spoofing Injection	ECU server	Yes	No	Special ECU server hardware	Altera FPGA board CAN transceiver board	No,Yes,No
[9] 2014	SAH3 HMAC	Replay Injection	No	Yes	No	Replace CRC field	Theoretical	No,Yes,No
[147] 2015	HMAC SHA 256	Replay Spoofing	Dedicated ECU-FPGA board with built-in HMAC	Yes	No	Monitoring ECU Node	Altera FPGA development board and CAN transceiver board	No,Yes,No
[117] 2016	128-bit key MAC Counter	Replay Injection Spoofing	No	Yes	No	16-bit counter in the CAN 2.0B frame header	FreescaleS12X and Infineon TriCore	No,Yes,No
[65] 2017	HMAC MD5 AES-128	Spoofing Replay	No	Yes	No	Insert 18-bit MAC tag in CAN 2.0B header	CANoeVector tool Freescale S12XF board	No,Yes,No

Works are ordered according to date of publication. Also, Methods, Attacks, Real time, Change CAN Bus, Test beds, and CIA triangle met by each work.

are pre-installed in the ECUs during manufacture. Also, the performance of their approach was validated analytically using a commercial bus load calculator by OptimumG [106]. The security of the algorithm was validated using the AVISPA software [6]. However, their approach was not tested in a realistic test environment, and it is not compatible with current vehicles—as their assumption is to embed the key during manufacture. Adding keys to an ECU can lead to additional complexity during manufacturing and can raise compatibility issues when an ECU from one manufacturer is replaced by another. This is particularly relevant when different OEMs can be installed inside a vehicle. If an attacker is able to break into an ECU and extract key materials, then they can inject malicious data inside a CAN bus network and work as legitimate ECU inside a vehicular network. Groza and Murvay [37] provide a secure broadcast protocol for the CAN Bus. It uses a central ECU to manage and distribute the keys between the sender and the receiver. They validated their approach using Freescale and S12X (16-bit) and InfineonTriCore (32bit) microcontrollers [56]. However, their approach is based on a delayed authentication approach that is difficult to support in real time for a CAN Bus [156]. In Reference [7], the authors introduced "TOUCAN," which provides authentication, integrity, and encryption for a CAN Bus. They use AES 128-bit and

Table 6. Message Authentication for CAN Frames

Authors	Method	Attacks mitigated	Hardware Security Module	Real time	Bus load	Change CAN Bus	Test bed environment	C,I,A
[77] 2012	MAC tables Pairwise key Symmetric key	Replay Spoofing	No	Yes	No	No	Theoritical	No,Yes,No
[159] 2016	AES-128 HMAC Compression algorithm	Replay Injection	No	Yes	No	Split CAN frame to encrypted and authentic frames	CANoe simulator	Yes,Yes,No
[32] 2017	Asymmetric key HMAC Changeable keys	Replay Spoofing	No	Yes	Load during key exchange	Assume pre-installed keys	AVISPA software	No,Yes,No
[39] 2017	L-MAC Splitting keys M-MAC	Replay Spoofing	No	Yes	No	Based on CAN+	S12 equipped with an XGATE coprocessor and Infineon TriCore	No,Yes,No
[7] 2019	AES-128 Chasekey MAC	Spoofing Replay	No	Yes	No	24-bit MAC tag and 40-bits for the actual data	STM32F407 CAN boards	Yes,Yes,No

Date of Publication is used to order these works.

Chaskey hashing [91] for MAC authentication without the need for ECU upgrade or additional hardware on the bus. They tested their approach on STM32F407 CAN boards. The actual data in the payload is 40 bits while the remaining 24 bits are used for the hashing value. In their approach (using AES 128 and Chaskey hashing), the overall execution times are approximately 12 ms. In Reference [159], the authors used AES 128 encryption and HMAC for authentication. They also use a compression algorithm to improve the efficiency of their approach by reducing the delay time and bus load. They have used Vector CANoe software to validate their approach, showing that the average message delay is 0.13 ms. In Reference [77], the authors focus on preventing re-play and spoofing attacks by using various approaches such as message counters, CAN ID tables to look up the ID of each ECU, and which MAC to use for each ECU ID. Pairwise symmetric key is used as each ECU stores the shared key with other ECUs. Also, the MAC tag is previously generated and stored in the lookup ID table where the ECU uses this ID table to link the receiving ECU with the correspondent MAC tag. Their approach does not need any hardware modification and has a low message latency and bus load. However, it does not provide confidentiality.

6.2.2 CAN Frame Encryption. In Reference [26], the authors used a combination of encryption and authentication mechanisms to provide data confidentiality, integrity, and authenticity. This approach provides prevention against sniffing and injection attacks. However, it sends more than one frame for a single CAN ID message, which can lead to latency and increased bus load [7]. In Reference [130], the authors used a hardware-based approach to provide authentication and encryption for a CAN Bus. A dedicated hardware (ECU Server) was used to manage all the ECUs in the CAN Bus to authenticate ECUs and distribute keys. A Xilinx Kintex KC705 FPGA Evaluation board and an embedded Physical Unlockable Function (PUF) was used in the testbed. This approach assumes that keys are registered for ECUs during manufacture, and assembling and CAN controller boards need to support the physical PUF function. This is likely to lead to less overhead, but it is infeasible to implement in current vehicle network due to the hardware modifications required [11]. CANTrack algorithm by Farag et al. [31] uses a dynamic symmetric key to encrypt the 8-byte data payload but does not modify the Msg ID as it is used to access the bus during the arbitration mechanism. They have tested their approach with CANoe software, and it has shown to prevent sniffing, replay, and spoofing attacks.

Table 7. CAN Frame Encryption Methods

Authors	Method	Attacks mitigated	Hardware Security Module	Real time	Bus load	Change CAN Bus	Test bed environment	C,I,A
[26] 2017	HMAC SHA1 AES DES	Sniffing Replay Spoofing	No	Yes	Yes	Split CAN Frames	Theoritical	Yes,Yes,No
[130] 2017	AES-128 Asymmetric key	Sniffing Replay spoofing	Hardware PUF ECU server	Yes	During initialisation and session	Change CAN transceiver	Xilinx Kintex KC705 FPGA hardware embedded PUF	Yes,Yes,No
[31] 2017	Dynamic symmetric key Key generator	Spoofing Replay Sniffing	No	Yes	No	No	CANoe software	Yes,Yes,No

Ordered by date of publication.

Table 8. CAN FD Encryption and Authentication

Authors	Method	Attacks mitigated	Hardware Security Module	Real time	Bus load	Change CAN Bus	Test bed environment	C,I,A
[155] 2016	AES-128 HMAC SHA256	Sniffing Replay Spoofing	No	Yes	No	No	Threetypes of CAN-FD boards and CANoe software	Yes,Yes,No
[39] 2017	ECU Group Sharing keys	Spoofing Replay Sniffing	No	Yes	NO	Group-based key sharing	InfineonTriCore controllers contrasted with low-end Freescale S12X cores	Yes,Yes,No
[13] 2018	ChaskeyMAC Pre-shared 128 bit key	Replay Spoofing	No	Yes	No	4 bytes counters 16 bytes MAC tag 43 bytes actual data	ArduinoUno Rev3 Arduino MPro	Yes,Yes,No
[1] 2019	Gateway ECU Public and Private keys	Sniffing Replay spoofing	No	Yes	No	Gateway ECU needed	CANoeVector and LPC54618 micro-controller	Yes,Yes,No

Improved data payload of 64 bytes allows more space for MAC signature along with actual data. Approaches ordered by date of publication.

CAN FD was introduced to tackle the needs of higher speed and larger data payload size. In Reference [155], the authors introduced an architecture supporting key management, encryption, and authentication for a CAN FD bus. They used symmetric key and AKEP2 to ensure distribution of keys and key freshness. They provided 16 bytes of HMAC-SHA256 tags and AES-128 to encrypt the rest of the data (47 bytes). Also, they provided an access control gateway ECU to limit the number of nodes that can access the bus. They validated their approach using three types of CAN-FD boards and CANoe software. Agrawal et al. [1] introduced a secure CAN FD bus that uses public, private keys, and groups of ECUs connected through a Gateway ECU (GECU). The GECU is used to verify session keys and key freshness for each ECU and to forward frames between different CAN sub-buses, e.g., the high- and low-speed CAN Buses. This approach uses 36 bytes for the data payload and 28 bytes for the cryptographic tag. They used CANoe software and the LPC54618 microcontroller to validate their approach. Groza et al. [39] introduced an approach for supporting CAN FD authentication. They used CANoe software to validate their approach. Their approach makes use of a group-based key sharing and generation key algorithm, a MAC algorithm to produce tags, and a verification algorithm to validate received messages. Carel et al. [13] used the lightweight Chaskey MAC algorithm [91] over limited-capacity computational resources such as a 32-bit microcontroller. They used this algorithm with a 128-bit key and compared it with the HMAC-SHA1 algorithm. They found Chaskey has a lower latency compared with HMAC-SHA1.

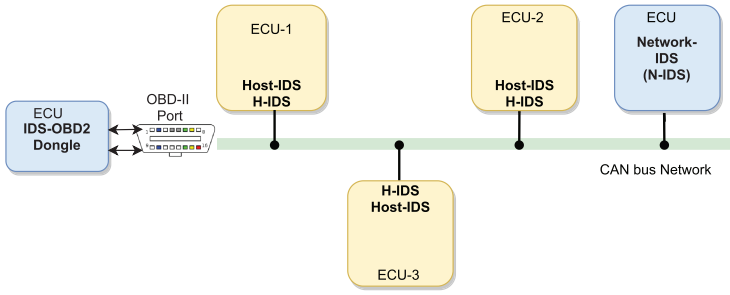

Fig. 7. Positions of IDS inside automotive CAN network.

They used 4 bytes as message counters, 16 bytes for Chaskey MAC tag, and 43 bytes of actual data payload. Their approach has focused on message authentication and ignores issues of data confidentiality.

6.3 In-Vehicle Intrusion Detection Systems

Using IDS to detect malicious attacks is a key approach implemented inside vehicle networks. IDS can be signature-based or anomaly-based systems [50]. The location of the IDS is also a key decision: Host-IDS (HIDS) based and Network-IDS (NIDS) based [158]. HIDS to detect attacks [74] may not be applicable for current vehicle networks and not cost effective, as this would require a change in ECUs. Therefore, installing a NIDS, as an additional node on the CAN Bus, such as an OBD-2 dongle, can be more feasible and practical and does not need CAN Bus modification [160].

An IDS can be passive, i.e., only reporting attacks, or active, i.e., performing actions to prevent attacks. ECUs inside the vehicle have a fixed interval to generate CAN messages even if no change occurs [89]. Thus the implementation of an IDS relies on deviations from a constant CAN traffic behaviour. Another approach uses the characteristics of the physical layer of each ECU, such as its signal and voltage profile, and compares the changes in these characteristics to detect anomalies. Müter et al. [95] have categorised the features that an IDS can use to detect attacks on the bus using the following sensors: *Format sensor* looks at different fields in the CAN frame such the correct size of the CAN message and the value of the check sum field; *Location sensor* indicates whether the message comes from the right CAN subsystems; *Payload range sensor* checks for legitimate range of values (data payload) inside the payload field; *Frequency sensor* considers the timing of the CAN message, as ECUs have a fixed frequency of data exchange/ operation; and *Correlation sensor* considers messages exchanged between multiple sub-domains within a vehicular networks. The gateway sensor is used to connect different sub-networks such as a high and low CAN domain. Thus, this sensor can use this feature to verify the legitimacy of the message that is transferred from one domain to another. *Protocol sensor* is used to monitor CAN traffic and detect changes in the protocol specification, such as the order of the messages and validity of the start and end time; *Plausibility sensor* checks if payload values are in the pre-defined range and if there is no sudden, anomalous increase in the payload; and *Consistency sensor* looks at the consistency of the values in the payload field. This sensor operates in contrast to the Plausibility, looking at additional sensors to verify the consistency of the messages transferred on the CAN Bus. For instance, the rotation of a tyre would indicate that the vehicle is stopped, while the GPS sensor indicates that the vehicle is moving. This approach therefore checks for consistency across multiple sensor values.

Figure 7 illustrates the position of an IDS inside a CAN network. The IDS can use different CAN frame features as follows:

Table 9. IDS Using Signatures and Rules

Authors	Type	Layer	CAN ID / Data Payload	Detection mechanism	Attacks Detected	Prevention
[74] 2008	**Specification**	DataLink	Extract signature from CAN Open protocol specifications	Detect attack based on rules	Specification based attacks	No
[25] 2016	**Access list**	Data link	CAN ID	HIDS in each ECU	Malicious CAN ID	No
[136] 2018	**Signature based**	DataLink Layer (Controller layer)	CAN frame ID and dataflow	Derive signature and rules match	Malicious CAN ID and false payload	No

Works are ordered by date of publication.

- **CAN identifier:** This is the 11-bit or 29-bit value that determines the priority of the message on the bus and the content of the message. For example, CAN ID 0x000 is a malicious message, since it can be used to occupy the bus and perform DoS attacks. Also, monitoring the broadcast intervals can be through the CAN ID frequency, as it is unique across the network.
- **Data Length Code:** This is a 4-bit field used to identify the length of the data payload. This also has a fixed value and range, as each ECU uses fixed byte size in the data payload.
- **Data field:** This is 8 bytes maximum with a fixed length and range that should not be exceeded. Anomalies can be detected if abnormal values and changes occur in the data field.
- **Timestamp:** Each CAN frame has a timestamp; using this, an IDS can monitor the time intervals of CAN messages and observe any unusual behaviour. This approach is based on the observation that ECUs have fixed broadcast intervals, and thus an anomaly can be detected.

6.3.1 IDS Based on Signature. This IDS is based on detecting a pre-defined list of attack signatures. Although it has low false positive in the detection process, it needs to update its database signatures when new attacks emerge [138]. Also, the signature-based IDS needs to maintain a potentially large database of known attacks on in-vehicle networks (including potential variants of these) [133]. Extracting attack signatures in real time for a moving can also be a challenge and suffer from high latency.

Studnia et al. [136] introduced a signature-based IDS that uses a list of signatures derived from a CAN dataset. However, this approach has limited benefit as the length of CAN Bus words may not be known *a priori*. Furthermore, this approach may fail to detect an attack if it does not sense the first part of the data exchanged as malicious packets [5]. Larson et al. [74] introduced a HIDS installed on each ECU and compares messages on the bus based on the CAN Bus specification. This IDS monitors all incoming and outgoing traffic and compares them against the protocol specification. This approach requires changing the network topology and is not usable for real-time applications. In Reference [25], the authors introduced an anti-spoofing system that detects malicious messages using each ECU by detecting CAN message ID that were not sent by the ECU itself. The ECU informs the IDS, and an interrupt pulse is sent to the CAN Bus to overwrite the spoofed message.

6.3.2 IDS Based on Anomaly Detection. This method is implemented using statistical, machine learning, rule-based, and physical fingerprint methods. It builds a learning model able to identify *abnormal* traffic, identify new patterns, and predict attacks that have not been observed before.

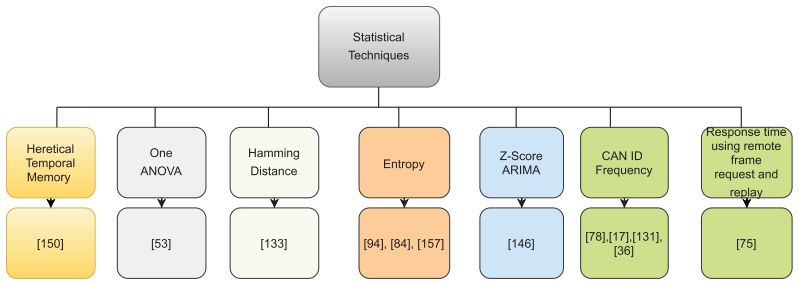

Fig. 8. IDS based on Statistical Techniques.

Using statistical approaches: This IDS learns *normal* behaviour of the system based on conditional statistical relationship analysis, as described in Figure 8. A baseline pattern is then developed as a threshold, in case changes are detected from the norm. In CAN Bus networks, statistical analysis uses CAN features such as CAN ID frequency and payload consistency. In general, ECUs have fixed intervals of time to send CAN frames. These CAN messages have a unique CAN identifier and used as a feature along with the time interval between frames, and the number of frames in each time unit [144]. Furthermore, the payloads inside CAN frames usually have consistent sequential values. A broader approach involves linking relationships between vehicular parameters such as the speed and RPM signals (under normal operation, there is a statistical correlation between RPM and speed). Finally, transmission frequency of messages, ID of messages, the number of packets received over a pre-determined time frame, message received sequence, and semantics of data fields can be used [63]. Hence, this IDS can detect manipulated and incorrect payload values along with inconsistent use of CAN ID. The following statistical approaches have been reported in the literature: (i) number of packets exchanged within a particular time period and time interval between CAN frames; (ii) frequency of transmission using a particular CAN ID; (iii) throughput observed; (iv) response time using remote frame requests and message reply; and (v) analysis approach used, e.g., Entropy, Anova, Z-score, ARIMA (time window-based moving average), Heretical Temporal Memory, and so on. Each of these approaches is investigated further below.

CAN ID Frequency: Ling and Feng [78] measure anomalies observed in traffic frequency to detect DoS and message injection attacks. However, using their approach, it is difficult to detect small-volume attacks, payload manipulation attacks, and impersonated ECU attacks where legitimate CAN ID messages are generated from the attacker ECU. **Intervals between CAN frames**: Cho and Shin [17] introduced a clock-based IDS to fingerprint each ECU based on its message exchange interval. Their approach uses a least square cost function and sequential analysis technique called Cumulative Sum algorithm to detect anomalies. Similarly to the previous approach, it is difficult to detect low-volume injected messages and impersonated disabled ECUs. The testbed consists of an Arduino UNO board and a SeedStudio CAN shield. Other approaches identified in Reference [131] have used an algorithm to analyse and detect unusual time interval for specific CAN ID message transmissions. Their approach focused on CAN injection attacks. **CAN ID frequency**: In Reference [36], the authors have introduced an IDS that can monitor CAN message frequency for each CAN message ID used by ECUs. This approach can also detect CAN injection and DoS attacks, but small forged messages are difficult to detect, since they might not alter the broadcast frequency of CAN ID. Furthermore, their approach does not consider data payload manipulation attack. **CAN traffic behaviour over a time window**: In Reference [139], the author developed an IDS based on an analysis of anomalies in data flow. This work involves comparing statistical values of current CAN traffic, over a 1-s time window, with historical values. However,

this anomaly detection over a time window cannot precisely detect small-sized malicious messages [133]. **Remote Frame Request and Reply intervals**: Lee et al. [75] used remote frames to detect anomalies based on the request and response intervals between frames. Since each ECU replies to a remote frame that has its CAN message id (and where the data payload field is empty), the authors calculated the average time between the request and reply to each ECU and were able to detect anomalies based on time interval variation against a calculated average response time. They were able to detect CAN Bus injection and ECU impersonation, as this would change the average response to a remote frame, and in the case of an impersonated ECU in the network, this would get response from both the legitimate and illegitimate ECU. **Entropy of CAN ID and data payload behaviour**: In Reference [94], the authors consider CAN ID and payload as features. They measured the entropy associated with changes in CAN traffic compared to a baseline normal CAN traffic. They tested their approach against frame injection attacks, and found that their approach cannot detect a small number of injected CAN messages. In Reference [84], the authors evaluated an entropy-based anomaly detection IDS for in-vehicle networks and found that dividing CAN messages into classes and passing them to an entropy-based anomaly detection algorithm provides more accurate detection. Their approach calculates entropy of all CAN Bus traffic (message ids) over a time window compared to a baseline (normal) traffic over the same time window. They found that measuring the entropy for each message id gives better performance in detecting smaller forged attacks, whereas considering all CAN traffic together would detect only larger sized attacks. Wu et al. [157] used an entropy-based IDS, with a fixed number of CAN frames over a sliding window as a baseline for their IDS. They improved the detection accuracy of the IDS based on the use of entropy calculation by using the optimal sliding window size with a fixed number of messages. They were able to achieve a better accuracy compared to previous entropy-based IDS. **One-way ANOVA Function**: In Reference [53], the authors used a one-way ANOVA function to statistically determine the pattern within a dataset and created a set of *normal* patterns to detect anomalies. They grouped CAN dataset using vehicle parameters, such as fuel usage, gear ratio, engine parameters, and so on, to detect abnormal events for each group. **Hamming Distance**: Anomaly detection based on Hamming distance algorithms have also been considered by other authors, e.g., in Reference [133], the authors analysed CAN payload and recorded each bit in the data field. They calculate Hamming distance for each payload to each message id, and attacks were identified based on significant deviation from the calculated Hamming distance function. **Quantized interval and the absolute Differences**: In Reference [71], the authors used an anomaly detection system based on quantized intervals for periodic CAN ID and determined the absolute difference of the CAN payload values. Their approach was validated against message injection attacks, and it showed positive results (using metrics such as True Positive/Negative Rates and False Positive/Negative Rates). However, they acknowledged that low-volume injection attacks were difficult to detect using their approach. **Cumulative Sum algorithm**: In Reference [105], the authors used an anomaly detection system based on the statistical cumulative sum algorithm. This is a sequential analysis technique used to support change detection. **ARIMA and Z-SCORE in Defined Time Window**: In Reference [146], the authors used an average value for the number of times a CAN ID was broadcast mean over a time window. This was used to determine changes in the CAN ID broadcast intervals over different time windows. The authors used a Z-Score and ARIMA, along with a supervised method, to compare the mean broadcast intervals of CAN ID. They were able to detect CAN injections and dropped packet attacks. **Heretical Temporal Memory (HTM):** In Reference [150], the authors used a distributed IDS based on HTM, a technique (similar to recurrent neural networks) widely used in time-series forecasting and analysis.

Machine Learning-based Approaches: IDS based on Machine Learning (ML) can be a good choice in extracting and learning normal vs. anomalous behaviour and then providing a model

Table 10. IDS Based on Statistical Methods

Authors	Layer	CANID / Data Payload	Detection mechanism	Attacks Detected	Dataset Available Online
CAN ID Frequency [78] 2012	Data Link (Controller layer)	CAN ID behaviour	Detect malicious CAN ID Detect Unusual CAN frequency	• Injection • DoS	NAO
One-Way ANOVA [53] 2015	Data link	Data payload consistency across multiple CAN signals	Compare the mean of related CAN frames according to the normal statistical observation e.g., speed and engine	• Data payload • manipulation	NAO
Entropy-based anomaly Detection [84] 2016	Data Link	CAN ID frequency changed	Provide independent variables for entropy-based anomaly detector for each group or class of CAN messages	• Malicious CAN ID • Manipulated payload	NAO
Detecting attacks through Hamming distance [133] 2017	Data link	Consecutive data payloads in certain CAN message ids	Compare the changes in Hamming distance values in sequential data payloads of CAN message ID	• Injection • Spoofing	NAO
Anomaly detection based on ID sequence [83] 2017	Data link	Sequence between CAN ID messages	Compare the sequence of CAN ID with the knowledge acquired from real-time model	• Replay • injection	NAO
Detecting attacks based on identifying Packet timing Anomalies in Time Windows [146] 2018	Data link	CAN ID broadcast mean in defined time window	Detect attack based on specification rules	• Injections • DoS attacks	NAO
Time-series algorithm. ARIMA and Z-Score [146] 2018	Data link	Broadcast intervals in time window	Check the change in broadcast intervals of CAN ID	• Drop • injection	NAO
offset ratio and remote frame IDS [75] 2018	Data link	CAN request and response intervals using remote frame	Time interval changes and the derived change in response to a remote frame	• Injection • ECU impersonation	[40]
Entropy IDS based on CAN ID [151] 2019	Data link	Entropy of each CAN ID	Detect the changes on each bit of the CAN ID	• Flooding • injection	NAO
Cumulative Sum algorithm in defined time window [105] 2019	Data link	CAN ID sequence	CAN ID sequence behaviour	• Injection • DoS attack • Frame Fuzz attack	NAO

Works are ordered by date of publication. Diffrent features are provided inside the table while very limited datasets are published online.

to detect and predict attacks. ML-IDS is widely used to handle large data volumes of CAN traffic with multiple features. It is useful to have a method to extract raw CAN data and pre-process it. This is particularly important, as vehicle manufacturers tend not to publish detailed specification and provide guidance on how to decode raw data features. Supervised ML algorithms can be time consuming, as raw CAN data need to be labelled, CAN attacks need to be identified, and then the data needs to be labelled and classified as well. Whereas unsupervised ML approaches do not require labelled datasets, and the algorithms can find common patterns directly from data and can use these patterns to classify traffic and identify anomalous behaviour. **Hidden Markov Models:** This approach works on time-series data to detect anomalous behaviour. Narayanan et al. [96] used an IDS based on a Hidden Markov Model to build a model able to detect anomalies and raise alarms. They investigated the use of each ID separately and using multiple vehicle variables together, such as vehicle speed and RPM CAN ID messages. They evaluated their model using instant observations against sudden changes in speed and RPM by injecting malicious message

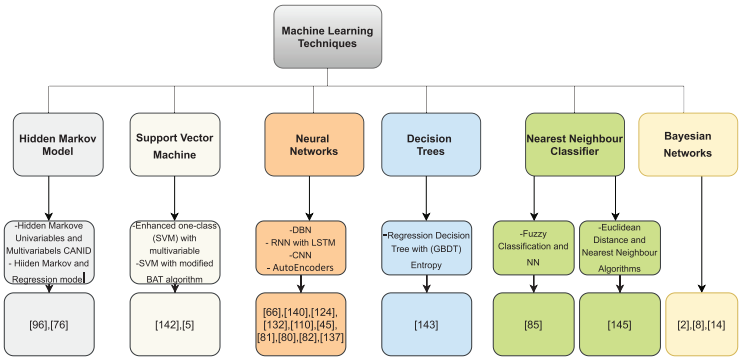

Fig. 9. Machine learning–based IDS.

for the parameters separately. They evaluated multiple attacks by injecting malicious speed and RPM messages together. Similarly, in Reference [76], the authors used a Hidden Markov Model to learn normal vehicle behaviour and used a regression model to build a threshold for the probability of occurrence of events to identify anomalies. This is a hybrid IDS approach that the authors of Reference [76] trained online during driving and stationary vehicle behaviour through captured data from the CAN Bus. They tested the model with noise attacks to mimic a real environment.

Support Vector Machines: In Reference [142], the authors enhanced one-class Support Vector Machines (SVM) to work with multiple variables to classify CAN data using an unsupervised ML technique. Their technique used unlabelled time-series data from a vehicle to learn normal behaviour and detect anomalies based on deviations. Their approach used a training set from real vehicles with error free logs. They then used a model with noisy data to detect errors and anomalies in the recorded data. In Reference [5], the authors used one-class SVM, comparing their approach with a Random Forest and classical One-class SVM (leading to better detection accuracy using the True Positive Rate metric).

Neural Networks (NN)/Deep Learning: In Reference [66], the authors used deep neural networks to learn normal patterns using unsupervised datasets and to detect deviation from normal as anomalies. They have used an unsupervised Deep Believe Network to pre-process the data and identify a normal pattern. To validate their approach, they inserted noise to their test dataset to mimic real vehicle data. They simulated and generated CAN frames using a real-world vehicle test bed and network experiments software [10]. In Reference [140], the authors used a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) to detect attacks on the CAN Bus. Their approach works with raw CAN Bus data without the need to reduce and abstract data during the pre-processing phase of analysis. In Reference [124], the authors used Generative Adversarial Nets to identify patterns of CAN data without classification. They tested their approach against DoS, frame fuzzification, and Spoofing attacks. In Reference [132], the authors have used Convolutional Neural Networks to build an IDS able to detect sequential patterns of vehicle traffic to detect Spoofing and DoS attacks. Their approach is based on the idea that CAN traffic is fed directly to their model without the need for pre-processing. They tested their approach offline, and they acknowledged that it is difficult to use it online in current vehicles. In Reference [110], the authors used an RNN with three LSTM layers: a dropout layer and two dense layers. The former layer is used to prevent over fitting, and the latter dense layer consists of 64 nodes to predict data payload for each CAN ID. Their approach uses an unsupervised technique where it does not need labelled free attack data and trained on labelled attack dataset. They argue that an ideal IDS should be able to plug inside existing vehicles to detect anomalies without the need for either reverse en-

gineering CAN traffic or contacting the vehicle manufacturer to get CAN messages specification. In Reference [45], the authors used a neural network model that consists of LSTM for CAN Bus time-series behaviour and auto-encoder to learn the normal behaviour of unlabelled data in unsupervised manner. Also, Exponential Linear Unit is used for better classification. This is suitable for CAN Bus data as the CAN Bus data representation is not published and considered as confidential and private for car makers.

In Reference [81], the authors used unsupervised deep learning method known as Deep Contractive Autoencoders. Furthermore, they have evaluated their approach against DoS and frame fuzzification to impersonate attacks, while they have used metrics such as Mean Square Error and Mean Absolute Error to compare between actual and predicted data. In Reference [80], the authors also used an unsupervised deep learning method using multiple layers of Stacked Sparse Autoencoders (SSAEs). This SSAE finds meaningful data representation of CAN, which enables their model to classify attacks from normal CAN data points. Finally, their approach has shown better performance compared to basic Sparsed and Stacked Autoencoders.

Using a different approach, the authors of Reference [82] used a deep learning IDS models using cloud computing to detect cyber-attacks on the CAN Bus. This approach can benefit from the large number of computational resources in the cloud, while it can be limited to provide offline detection. In Reference [128], Sharma and Moller have introduced an architecture for using IDS based on neural networks to detect attacks for in-vehicle networks, alert a manufacturer, surrounding connected cars, and push updates to mitigate the attacks. However, this is a theoretical framework that the authors have not validated on real scenarios. In Reference [137], the authors have used CAN ID, data payload, and intervals between CAN messages using an RNN algorithm. They tested their approach in a simulated environment using CAN data extracted from a real vehicle. They considered malfunction attack (false CAN ID and data payload) and flooding attack. Another hybrid IDS is introduced in Reference [153], based on a specification-based IDS used to detect data payload consistency as a first stage. The authors then use an ML algorithms such as RNN, SVM, and Lightweight online Detector to detect anomalies.

Decision trees: Decision tree– (DT) based approaches classify CAN data into two classes (normal, anomalous). Decision tree–based approaches need a supervised labelled dataset during the training stage to be able to make decisions. In Reference [143], authors have used a regression Decision Tree with Gradient Boosting (GBDT) technique to make a better classifier. They have used entropy to construct the decision algorithm in which they calculated the entropy of the CAN ID and the data payload time. Also, Gradient Boosting is a technique of using multiple trees and training them to get the optimal DT model. They validated their approach using real captured CAN data containing 750,000 messages. They changed the test dataset by inserting random abnormal values as anomalies.

Nearest Neighbour Classifier: In Reference [85], the authors used fuzzy classification algorithms based on Nearest Neighbour classifiers to discriminate attacks targeting the CAN Bus. They used a dataset available online that contains CAN attacks to validate their approach. They used the data payload—actual data (8 bytes)—as features to classify CAN traffic. They tested their approach on different attacks such as DoS, frame injection, and frame fuzzification provided in the dataset. They achieved a precision value between 0.85 to 1 using a neural network algorithm. However, this detector may fail to detect small forged injected messages and impersonated ECU attacks. In Reference [145], the authors used a combination of Euclidean distance and nearest neighbour algorithms. They improved the method of distance-based nearest neighbour technique by categorising CAN data into four domains, improving potential prediction accuracy by limiting to these four domains. They tested their approach against frame fuzzification attacks where they randomly change

the data payload of the logged CAN messages. They considered CAN ID frequency, time between packets, and data payload values as features.

Bayesian Networks: Bayesian networks in IDS can be used to (i) predict the sequence (time evolution) of an event, (ii) integrate previous knowledge with probabilistic techniques, and (iii) to handle missing data by encoding inter-dependencies between variables [108]. In Reference [2], the authors used a collection of sensor data, e.g., speed, geo-location, and routes from a connected car. Their approach then makes use of a detection system using probabilistic Recursive Bayesian Estimation IDS. They have used three models in their approach: filtering (estimating the current event value), smoothing (estimating past event value), and prediction (estimation the likelihood of a future event). In Reference [8], the authors looked at various attack vectors on autonomous vehicles using a Bayesian network to detect and classify the type and source of the attack, e.g., cyber-physical attacks using sensor data from autonomous vehicle. They have used a Hill-Climbing algorithm to construct a Direct Acyclic Graph to learn the behaviour of all data sources, classify these sources, and detect cyber-attacks (remote) and physical attacks based on the source of the data. In Reference [14], the authors used a series of probabilistic approaches based on a Bayesian network to detect attacks. They implemented a test bed based on the CARLA simulator along with support for accelerator, steer, brake sensors, and IDS connected to the simulator as ECUs. Furthermore, they evaluated their IDS based on various metrics such as true-positive and true-negative rates, precision, recall, and F1 score.

IDS Based on Physical Characteristics: This approach works at the physical layer of the CAN Bus, as it builds a profile of signals and voltage signature for each ECU. It then compares the traffic with the profile for abnormal traffic. In Reference [122], the authors introduced a hardware-based Intrusion Response System. This is a signal- and voltage-based physical layer (transceiver layer) IDS that detects attacks based on changes in characteristics for each ECU. This approach can be used to detect unusual signals at the physical layer to overcome attacks such as over current, DoS, and error frame re-transmission. In Reference [18], the authors proposed a clock-based IDS to detect anomalies. They build a fingerprint for each ECU based on measuring and extracting the periodic frequency of messages sent by ECUs. They have used the fingerprint of each ECU to build a baseline behaviour of the ECU clock using Recursive Least Square algorithm. Their approach uses a Cumulative Sum to detect any significant deviation from the normal fingerprint baseline. In Reference [20], the authors introduced a Voltage-IDS that is based on the use of electrical CAN signals as a fingerprint for ECUs. Their approach was also used to detect an off-bus attack where an ECU is blocked and the attacker mimics a disabled ECU.

Comparing IDS: As mentioned previously, an IDS is used to detect malicious CAN attacks. Signature-based IDS has been shown to detect attacks with low false positives; however, an attack signature needs to be extracted from the CAN Bus. Therefore, new CAN attacks can be difficult to identify. It is also difficult to detect attacks on a moving car to extract attack signature and CAN messages. Anomaly- or behaviour-based IDS has the advantage that it can detect and predict attacks based on the training and learning process and can in some cases be used without the need for more training. IDS based on machine learning can benefit from raw data directly extracted from vehicles. Machine learning approaches can also handle multiple variable instances as vehicles generate large amounts of data that need to be pre-processed to be meaningful. This problem can be overcome using unsupervised ML that can classify patterns and detect anomalies in unlabelled raw data.

7 LIMITATIONS WITH CURRENT APPROACHES TO CAN BUS SECURITY

Based on the survey in previous sections, we now describe limitations with current approaches for securing in-vehicle systems (particularly focusing on the CAN Bus). The implementation of

Table 11. Machine Learning–based IDS

Authors	Type	Detecting threshold	Detection mechanism	Attacks Detected	Dataset Available Online
[142] 2014	**Enhaced SVM**	CAN ID and data payload Multivarinete CAN signals	Deviation from ESVM Enhanced one-class Support Vector Machine	Error and Signal faults	NAO
[96] 2016	**Hidden Markov Model**	Univariante CAN signal Multivariant CAN signals e.g., RPM and speed	Deviation from the sequence behaviour	Single injection Multiple injection	NAO
[140] 2016	**RNN with long short-term memory**	CAN ID and data payload behaviour	Devaition from the RRN model and observations learned in LSTM mechanism	Injection DoS	NAO
[66] 2016	**Deep Believe Neural Network with Probability feature**	CAN ID and data payload behaviour	Change from the NN model pattern	Injection	NAO
[19] 2017	**O-SVM**	CAN message intervals and frequencies One Class support Vector based Anomaly IDS	Anomaly based on One SVM class detection	Fuzzing	NAO
[85] 2017	**Fuzzy classification Nearest Neighbor Classification**	CAN ID and data payload	Checking each byte of the data payload as features to detect anomalies	DoS Injection Fuzzy	[48]
[76] 2018	**Hidden Markov Model Regression Model**	HMM and regression model to build a threshold for the log probabilities	Offline learning from dataset and online learning	Noise Attack	NAO
[145] 2018	**Euclidean distance and nearest neighbor algorithms**	CAN ID frequency in time window	Change in CAN ID broadcast data payload	Fuzzy	NAO
[143] 2018	**Regression GBDT Entropy**	CAN ID and data payload entropy change	Entropy change of CAN traffic	Injection DoS	NAO
[153] 2018	**MLHybrid-IDS**	CAN messages payload sequence in static check module. RNN-based IDS OCSVM and Online Algorithm LODA	Detection in time window and payload consistency	Injection DoS	NAO
[150] 2018	**Multiple Anomaly IDS based on HMS**	Data sequence anomaly based on HMS	Multiple HMS-IDS for each CAN signal learn from online stream	Injection DoS	NAO
[2] 2018	**Bayesian Network**	Estimation the likelihood of events based on BN networks	future state predictions based on the previous behaviour	Abnormal behaviour Malicious activities	NAO
[5] 2019	**O-SVM with modified BAT algorithm**	Deviation from recurring patterns in the message IDs	One-class SVM algorithm	Injection DoS	NAO
[132] 2020	**Deep Conventional Neural Network**	Learning sequential patterns of CAN Bus traffic and detects message injection	Conventional Neural Network	Message Injection	[41]

Limited datasets are published online. Works are ordered by date.

a CAN cryptographic algorithm should consider the unique nature of the protocol, the limited network infrastructure (with support for limited data payload), and computationally constrained ECU specification. The algorithms should also consider the broadcast nature of the CAN Bus, key distribution, and freshness to avoid replay attacks. Real-time sensitivity is a concern inside vehicles, since critical services and functions are sensitive to latency. Therefore, any countermeasure should consider these criteria.

Table 12. IDS Based on Physical Characteristics

Authors	Type	Layer	Detecting threshold	Detection mechanism	Attacks Detected	Prevention	Online Dataset
[18] 2017	**Voltage Profile for each ECU**	Physical layer	The changes of voltage on the line	each ECU has its own unique voltage profile	Any data generated from unfamiliar ECU voltage will be detected	Yes	NAO
[20] 2018	**Electrical CAN signals as a fingerprint for ECUs**	Physical layer	Change in the electric signal of each ECU	Change in the electrical signals on the bus and comparing the fingerprint of the ECU	Off bus attack	Yes	NAO
[122] 2019	**Signal and Voltage based**	Physical layer (Transceiver layer)	Detect attacks based on changes on (signal and voltage) characteristics	measure the unique signal for each ECU and detect unusual behaviour	physical layer attacks such as over-current DoS bus idle and error frame re-transmission. Will not work in ECU impersonation attack	Yes	NAO

Ordered by date. Works are based on Layer 1 physical, e.g., signal and voltage features.

Table 13. Comparison of Statistical, Machines Learning–, and Physical Characteristics-based IDS

IDS Type	Layer	Data Collection	Detection/ Prevention	Cost of Implementation
Statistical IDS	Layer2	CAN ID Frame/ Frame Flow behaviour	Yes/No	Medium and required labelled data
ML IDS	Layer2	CAN ID Frame/ Frame Flow behaviour	Yes/No	Medium. Can work with labelled and raw data
Physcial IDS	Layer1	Physical statistical features, e.g., Skew, clock offset and clock frequency	Yes/Yes	High. Extract layer 1 features such as signal and voltage of each ECU.

7.1 Cryptography

Hardware-based cryptography: This can be used to speed up the process of generating cryptographic functions. Depending on the number of ECUs (typically 70), each ECU would need to be updated. While this approach can increase and speed up the process of cryptographic mechanisms to meet real-time needs, it is not compatible with current vehicles and the cost of implementation can be significant. Future CAN FD boards are expected to be embedded with hardware supported security mechanisms such as AES and embedded authentication [112]. Also, better computational resources are expected in the next generation of CAN FD ECUs to handle the higher bitrate and data payload size needed. **Software-based cryptography:** This does not require additional hardware or modification to existing ECUs. Authentication approaches are less computationally expensive, as they add additional information within an existing data payload. In Reference [103], the authors evaluated CAN MAC approaches and indicate that some of these approaches can be applicable to a subset of the network with limited number of critical ECUs. This is due to the issue of overhead, as suggested by the authors of Reference [103]. The main obstacle with these approaches is to implement them within computationally constrained ECUs. The authors suggest that some of these MAC mechanisms can be used inside "a small subset of safety-critical ECUs."

Message latency: Vehicles utilise several real-time functions for which latency can threaten safety on the road. Therefore, encryption mechanisms should provide security and reduce message latency to a minimum. This process overcomes the limited computational resources inside current vehicle ECUs. Further, the payload size of a classical CAN Bus makes it difficult to add secure

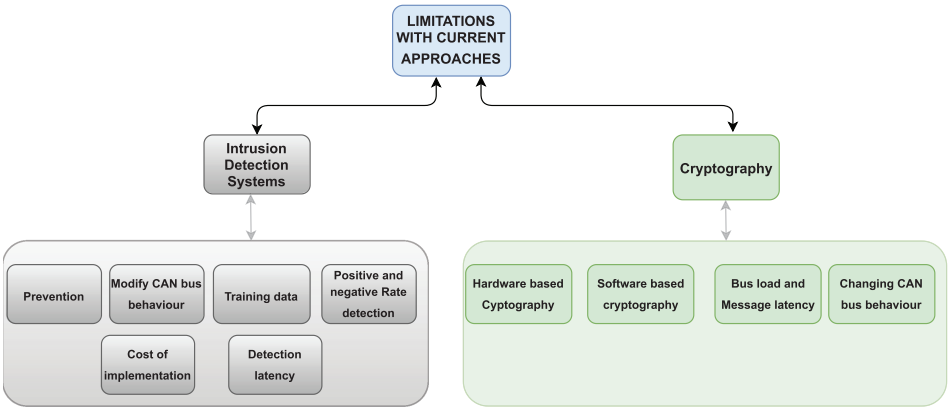

Fig. 10. Research challenges: Cryptography and IDS-based approaches.

tags and signatures along with actual data. Therefore, authentication and encryption focus on lightweight mechanisms using MAC tags without encrypting the whole payload. The CAN Bus should not be loaded with extra security related messages, e.g., by splitting CAN Bus messages, one message for the actual data and the other for authentication of the message. This can increase bus load twofold and therefore affect the quality of the network.

Changing CAN Bus behaviour: Some approaches change the CAN protocol by either changing the payload size or splitting a message into data and authentication messages. Other approaches have changed CAN frame structure by replacing and inserting MAC tags and signatures inside CRC fields and CAN identifier fields. This can lead to incompatibility issues with ECUs and add additional complexity to the current CAN Bus.

7.2 Intrusion Detection System

As there is no global attack signature database, an IDS needs to collect and analyse CAN network data to build an attacks signature database. While a global database for CAN Bus malware can help to provide better protection and countermeasures, it can be difficult to implement, as ECUs from different manufacturers may have different properties. Additionally, responsibility for updating and maintaining such a database is also unclear. A suggestion would be for each OEM, for example, Mercedes/Toyota/BMW, and so on, to build such a global database for their vehicles (e.g., in an OEM-specific cloud) and to update and maintain their database. Machine Learning–based IDS may have high computational resource requirements; however, ECU resources that exist inside vehicles may not be able to handle this workload. It is worth noting that an IDS can be installed in each ECU, as a Host-IDS, to detect attacks [74]; however, this can lead to incompatibility and high deployment cost. Therefore, installing an IDS as a network node, such as an IDS-OBD-2 dongle, can be more feasible and practical and does not need CAN Bus modification [160]. As considered in this survey, high-volume attacks (e.g., DoS) were easily detected by all named IDS, whereas for low-volume attacks (e.g., small number of spoofed CAN messages) information entropy and throughput performed poorly, with attacks having little effect on CAN traffic behaviour. Impersonated ECU can be detected by physical IDS, as it profiles unique signal and voltage for each legitimate ECU. Control-oriented techniques are not widely covered in the area of vehicular security; therefore, further investigation and work on these techniques are needed to ensure resilience and recovery, especially for connected and autonomous vehicles.

8 CONCLUSION

Cryptographic mechanisms have been used to secure the CAN Bus from attacks that originate from inside the vehicle or when an external attacker can get access to the CAN Bus. However, it may be difficult to use encryption because of the lack of computational resources in current ECUs and the small data payload size and low data bitrate of the CAN Bus network. Additionally, decision making within vehicles requires real-time data analysis, and any delay due to data encryption can lead to safety issues on the road. In contrast, IDS operates as a countermeasure inside a vehicles and works in a passive manner. An IDS does not require a change in the network and protocol specifications compared to some cryptographic methods. However, some IDS based on deep learning, for instance, requires significant computational resources not available within a vehicle.

For the current classical CAN Bus vehicle networks, edge ECU devices can be used for monitoring and management of message authentication and encryption mechanisms and IDS approaches. These edge devices can be used as a plugin device, e.g., inside OBD-2, telematics, and infotainment interfaces to support IDS mechanisms, detect attacks, process vehicular data, and push it for further analysis (e.g., OEM and fleet management clouds) such as diagnostics and attack analysis. Edge devices can also provide suitable resources to support countermeasures and be used in current vehicles with limited CAN Bus modifications. If ECU modification are needed, for example to support authentication and contact with edge devices, then OEMs should consider update ECU capabilities, e.g., over-the-air updates. It is worth mentioning that there are efforts between Mercedes and Nvidia to introduce Software Defined ECUs [113] to provide better management and security along with overcoming the current issues with legacy ECUs. As CAN FD is the improved version of the classical CAN Bus, it is already implemented inside many new vehicles. Many OEMs are expected to use CAN FD by 2022 in the US and Europe [118]. The next-generation CAN FD is expected to provide better resources, e.g., higher data bitrate, payload size, and support for embedded encryption methods. As a result, vehicles based on CAN FD can overcome current ECU shortcomings. Other additional capabilities include (1) better ECUs to handle higher data payload and (2) embedded cryptographic algorithms such as AES and better data bitrate. Therefore, suppliers and OEMs can embed countermeasures, e.g., IDS, encryption, and message authentication, along with firewall and access control lists for external CAN Bus connections.

Communication outside CAN Bus such as telematics, infotainment, and wireless sensor interfaces along with DSRC for V2V and V2I should be protected, as they are entry points for data injection to the internal vehicle CAN Bus network.Existing countermeasures have mainly focused on securing vehicular networks from inside out. However, control-oriented approaches (to control and recover from attacks especially in real time) are not widely covered. These approaches should be further investigated.

Also, overhead calculation and comparison between approaches that we have reviewed are rather complex, as limited information and various metrics are considered among related works.

We found limited information available in the literature on how attack data are collected and validated. It is therefore important that the research community identify mechanisms to develop and publish realistic datasets. This is an important requirement to improve the maturity of research in this area. This could also enable researchers to ask the question whether these datasets are sufficient and to identify limitations in their use for emerging machine learning algorithms (e.g., deep learning approaches) and could inspire additional work to build better datasets for vehicular security.

Although a number of bus architectures have been introduced, e.g., FlexRay and LIN, it is important to highlight that the CAN Bus remains the most widely used standard in the automotive

industry. As outlined in this article, a number of improvements have been made by vehicle manufacturers to the CAN Bus over the years, e.g., to support higher data rates for connected and autonomous cars, such as in CAN FD and CAN XL. Also, since the CAN Bus protocol is used inside both electric and autonomous cars [52], and due to the significant interest in these types of vehicles, interest in cybersecurity of the CAN Bus protocol will continue to grow.

REFERENCES

[1] Megha Agrawal, Tianxiang Huang, Jianying Zhou, and Donghoon Chang. 2019. CAN-FD-Sec: Improving security of CAN-FD protocol. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 11552. Springer Verlag, 77–93.

[2] Haider Al-Khateeb, Gregory Epiphaniou, Adam Reviczky, Petros Karadimas, and Hadi Heidari. 2018. Proactive threat detection for connected cars using recursive bayesian estimation. *IEEE Sens. J.* 18, 12 (Jun. 2018), 4822–4831.

[3] Omid Avatefipour, Azeem Hafeez, Muhammad Tayyab, and Hafiz Malik. 2017. Linking received packet to the transmitter through physical-fingerprinting of controller area network. In *Proceedings of the 2017 IEEE Workshop on Information Forensics and Security (WIFS'17)*, Vol. 2018. Institute of Electrical and Electronics Engineers Inc., 1–6.

[4] Omid Avatefipour and Hafiz Malik. 2018. State-of-the-art survey on in-vehicle network communication (CAN-Bus) security and vulnerabilities. arxiv:1802.01725. Retrieved from https://arxiv.org/abs/1802.01725.

[5] Omid Avatefipour, Ameena Saad Al-Sumaiti, Ahmed M. El-Sherbeeny, Emad Mahrous Awwad, Mohammed A. Elmeligy, Mohamed A. Mohamed, and Hafiz Malik. 2019. An intelligent secured framework for cyberattack detection in electric vehicles' can bus using machine learning. *IEEE Access* 7 (2019), 127580–127592.

[6] AVISPA. 2018. Retrieved August 2020 from http://www.avispa-project.org/.

[7] Giampaolo Bella, Pietro Biondi, Gianpiero Costantino, and Ilaria Matteucci. 2019. TOUCAN: A proTocol to secUre Controller Area Network. In *Proceedings of the ACM Workshop on Automotive Cybersecurity, co-located with CODASPY 2019 (AutoSec'19)* (2019), 3–8.

[8] Anatolij Bezemskij, George Loukas, Diane Gan, and Richard J. Anthony. 2018. Detecting cyber-physical threats in an autonomous robotic vehicle using Bayesian networks. In *Proceedings of the 2017 IEEE International Conference on Internet of Things, IEEE Green Computing and Communications, IEEE Cyber, Physical and Social Computing (IEEE Smart Data, iThings-GreenCom-CPSCom-SmartData'17)*, Vol. 2018. Institute of Electrical and Electronics Engineers Inc., 98–103.

[9] Sebastian Bittl. 2014. Attack potential and efficient security enhancement of automotive bus networks using short MACs with rapid key change. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 8435, 113–125.

[10] Parnian Najafi Borazjani, Christopher E. Everett, and Damon Mccoy. 2014. OCTANE: An extensible open source car security testbed. In *Proceedings of the Embedded Security in Cars Conference (ESCAR'14)*, 1–10.

[11] Mehmet Bozdal, Mohammad Samie, and Ian Jennions. 2019. A survey on CAN bus protocol: Attacks, challenges, and potential solutions. In *Proceedings of the 2018 International Conference on Computing, Electronics and Communications Engineering (iCCECE'18)*. Institute of Electrical and Electronics Engineers Inc., 201–205.

[12] CAN in Automation. 2013. CAN in Automation. Retrieved August 2020 from https://www.can-cia.org/can-knowledge/.

[13] Guillaume Carel, Ryunosuke Isshiki, Takuya Kusaka, Yasuyuki Nogami, and Shunsuke Araki. 2018. Design of a message authentication protocol for CAN FD based on chaskey lightweight MAC. In *Proceedings of the 2018 6th International Symposium on Computing and Networking Workshops (CANDARW'18)*, 267–271.

[14] Mario Casillo, Simone Coppola, Massimo De Santo, Francesco Pascale, and Emanuele Santonicola. 2019. Embedded intrusion detection system for detecting attacks over CAN-BUS. In *Proceedings of the 2019 4th International Conference on System Reliability and Safety (ICSRS'19)*. Institute of Electrical and Electronics Engineers Inc., 136–141.

[15] Madeline Cheah, Jeremy Bryans, Daniel S. Fowler, and Siraj Ahmed Shaikh. 2017. Threat intelligence for Bluetooth-enabled systems with automotive applications: An empirical study. *Proceedings of the 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W'17)* (2017), 36–43.

[16] Stephen Checkoway and D. McCoy. 2011. Comprehensive experimental analyses of automotive attack surfaces. In *Proceedings of the 20th USENIX Conference on Security* (2011), 6.

[17] Kyong-Tak Cho and Kang G. Shin. 2016. Fingerprinting electronic control units for vehicle intrusion detection. In *25th Usenix Security Symposium (Usenix Security'16)*. 911–927.

[18] Kyong Tak Cho and Kang G. Shin. 2017. Viden: Attacker identification on in-vehicle networks. In *Proceedings of the ACM Conference on Computer and Communications Security*, 1109–1123.

[19] Valliappa Chockalingam, Ian Larson, Daniel Lin, and Spencer Nofzinger. 2017. Detecting Attacks on the CAN Protocol With Machine Learning. Retrieved August 2020 from http://www-personal.umich.edu/ valli/assets/files/CAN_AD.pdf.

[20] Wonsuk Choi, Kyungho Joo, Hyo Jin Jo, Moon Chan Park, and Dong Hoon Lee. 2018. VoltageIDS: Low-level communication characteristics for automotive intrusion detection system. *IEEE Trans. Inf. Forens. Secur.* 13, 8 (2018), 2114–2129.

[21] CiA. 2020. CAN in Automation (CiA): CAN XL Is Knocking at the Door. Retrieved August 2020 from https://www.can-cia.org/news/cia-in-action/view/can-xl-is-knocking-at-the-door/.

[22] Roderick Currie. 2017. Hacking the CAN bus: basic manipulation of a modern automobile through CAN bus reverse engineering. *SANS Institute* (2017). Retrieved August 2020 from https://www.sans.org/reading-room/whitepapers/threats/hacking-bus-basic-manipulation-modern-automobile-through-bus-reverse-engineering-37825.

[23] Roderick Currie. 2015. Information security reading room developments in car hacking. Retrieved August 2020 from https://www.sans.org/reading-room/whitepapers/ICS/developments-car-hacking-36607.

[24] CVIS Cooperative Vehicle-Infrastructure Systems. 2010. CVIS Cooperative Vehicle-Infrastructure Systems. Technical Report. Retrieved August 2020 from http://www.cvisproject.org/.

[25] Tsvika Dagan and Avishai Wool. 2016. Parrot, a software-only anti-spoofing defense system for the CAN bus. In *Proceedings of the 14th Embedded Security in Cars (ESCAR'16)*, 10.

[26] Luca Dariz, Michele Selvatici, Massimiliano Ruggeri, Gianpiero Costantino, and Fabio Martinelli. 2017. Trade-off analysis of safety and security in CAN bus communication. In *Proceedings of the 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS 2017)*, 226–231.

[27] Juan Deng, Lu Yu, Yu Fu, Oluwakemi Hambolu, and Richard R. Brooks. 2017. *Security and Data Privacy of Modern Automobiles*. 131–163.

[28] Wilrid Dubitzky and Turgut Karacay. 2013. CAN—From its early days to CAN FD. *CAN Newslett.* (2013), 8–11. Retrieved Aug. 2020 from http://www.can-newsletter.org/uploads/media/raw/6b2563046de889524638725c61627661.pdf.

[29] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh, and Mohammad T.Manzuri Shalmani. 2008. On the power of power analysis in the real world: A complete break of the KeeLoq code hopping scheme. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 5157. Springer, Berlin, 203–220.

[30] ETSI. 2001. Universal Mobile Telecommunications System (UMTS); specification of the 3GPP confidentiality and integrity algorithms; Document 1: f8 and f9 specifications (3GPP TS 35.201 version 4.1.0 Release 4).

[31] Wael A. Farag. 2017. CANTrack: Enhancing automotive CAN bus security using intuitive encryption algorithms. In *Proceedings of the 2017 7th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO'17)*, 1–5.

[32] Samir Fassak, Younes El Hajjaji El Idrissi, Noureddine Zahid, and Mohamed Jedra. 2017. A secure protocol for session keys establishment between ECUs in the CAN bus. In *Proceedings of the 2017 International Conference on Wireless Networks and Mobile Communications (WINCOM'17)*.

[33] Andreas Forsberg and Johan Hedberg. [n.d.]. *Comparison of FlexRay and CAN-bus for Real-Time Communication*. Technical Report.

[34] Daniel S. Fowler, Jeremy Bryans, Siraj Ahmed Shaikh, and Paul Wooderson. 2018. Fuzz testing for automotive cybersecurity. In *Proceedings of the 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W'18)*, 239–246.

[35] Robert Bosch Gmbh. 1991. Robert Bosch Gmbh: CAN specification version 2.0. Retrieved August 2020 from http://esd.cs.ucr.edu/webres/can20.pdf.

[36] Mabrouka Gmiden, Mohamed Hedi Gmiden, and Hafedh Trabelsi. 2017. 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'16)*, 176–180.

[37] Bogdan Groza and Stefan Murvay. 2013. Efficient protocols for secure broadcast in controller area networks. *IEEE Trans. Industr. Inf.* 9, 4 (2013), 2034–2042.

[38] Bogdan Groza, Stefan Murvay, Anthony Van Herrewege, and Ingrid Verbauwhede. 2012. LiBrA-CAN: A lightweight broadcast authentication protocol for controller area networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 7712. 185–200.

[39] Bogdan Groza, Stefan Murvay, Anthony Van Herrewege, and Ingrid Verbauwhede. 2017. LiBrA-CAN: Lightweight broadcast authentication for controller area networks. *ACM Trans. Embed. Comput. Syst.* 16, 3 (2017).

[40] Hacking and Countermeasure Research Lab. 2020. CAN-intrusion-dataset (OTIDS)—Hacking and Countermeasure Research Lab. Retrieved August 2020 from http://ocslab.hksecurity.net/Dataset/CAN-intrusion-dataset.

[41] Hacking and Countermeasure Research Lab. 2020. Car-Hacking Dataset—Hacking and Countermeasure Research Lab. Retrieved August 2020 from http://ocslab.hksecurity.net/Datasets/CAN-intrusion-dataset.

[42] Azeem Hafeez, Hafiz Malik, Omid Avatefipour, Prudhvi Raj Rongali, and Shan Zehra. 2017. Comparative study of CAN-bus and FlexRay protocols for in-vehicle communication. In *SAE Technical Papers*, Vol. 2017, March, 6–11.

[43] Azeem Hafeez, Khurram Rehman, and Hafiz Malik. 2020. State of the art survey on comparison of physical fingerprinting-based intrusion detection techniques for in-vehicle security. In *SAE Technical Papers*, Vol. 2020, April.
[44] Kyusuk Han, Swapna Divya Potluri, and Kang G. Shin. 2013. On authentication in a connected vehicle: Secure integration of mobile devices with vehicular networks. In *Proceedings of the 2013 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS'13)*. ACM, 160–169.
[45] Markus Hanselmann, Thilo Strauss, Katharina Dormann, and Holger Ulmer. 2020. CANet: An unsupervised intrusion detection system for high dimensional CAN bus data. arxiv:1906.02492. Retrieved from https://arxiv.org/abs/1906.02492.
[46] Shawn Hartzell, Christopher Stubel, and Tamara Bonaci. 2020. Area network bus. *IEEE Potentials* 39, 3 (2020), 19–24.
[47] Ahmed Hazem and Hossam A. H. Fahmy. 2012. LCAP - A lightweight CAN authentication protocol for securing in-vehicle networks. In *Proceedings of the 10th Embedded Security in Cars Europe Conference (ESCAR'12)*.
[48] HCRL. 2019. Hacking and Countermeasure Research Lab. Retrieved August 2020 from https://sites.google.com/a/hksecurity.net/ocslab/Datasets/car-hacking-dataset.
[49] Anthony Van Herrewege, Dave Singelee, and Ingrid Verbauwhede. 2011. CANAuth - A simple, backward compatible broadcast authentication protocol for CAN bus. In *Proceedings of the ECRYPT Workshop on Lightweight Cryptography*. 299–235.
[50] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. 2009. Applying intrusion detection to automotive it – Early insights and remaining challenges. *J. Inf. Assur. Secur.* 4, January 2009 (2009), 226–235.
[51] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. 2011. Security threats to automotive CAN networksPractical examples and selected short-term countermeasures. *Reliabil. Eng. Syst. Safety* 96, 1 (2011), 11–25.
[52] Mike Horton. 2019. What can a CANbus IMU do to make an autonomous vehicle safer? Retrieved August 2020 from https://www.autonomousvehicleinternational.com/opinion/what-can-a-canbus-imu-do-to-make-an-autonomous-vehicle-safer.html.
[53] Ching Hsien Hsu, Feng Xia, Xingang Liu, and Shangguang Wang. 2015. Internet of vehicles—Safe and intelligent mobility. In *Proceedings of the2nd International Conferenceon Internet of Vehicles (*IOV'15*). Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9502 (2015), 89–97.
[54] Tianxiang Huang, Jianying Zhou, Yi Wang, and Anyu Cheng. 2017. On the security of in-vehicle hybrid network: Status and challenges. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 10701. Springer, Cham, 621–637.
[55] Kazuki Iehira, Hiroyuki Inoue, and Kenji Ishida. 2018. Spoofing attack using bus-off attacks against a specific ECU of the CAN bus. In *Proceedings of the 2018 15th IEEE Annual Consumer Communications and Networking Conference (CCNC'18)*, 1–4.
[56] Infineon Technologies AG. 2020. Microcontroller based on TriCore™. Retrieved August 2020 from https://www.infineon.com/cms/en/product/microcontroller/32-bit-tricore-microcontroller/.
[57] International Organization for Standardization. 2015. ISO 11898-1:2015–Road Vehicles–Controller Area Network (CAN)–Part 1: Data Link Layer and Physical Signalling. Retrieved August 2020 from https://www.iso.org/standard/63648.html.
[58] International Organization for Standardization. 2019. Road Vehicles—Safety of the Intended Functionality Retrieved August 2020 from https://www.iso.org/standard/70939.html.
[59] Ishtiaq Rouf, Robert D. Miller, Hossen A. Mustafa, Travis Taylor, Sangho Oh, Wenyuan Xu, Marco Gruteser, Wade Trappe, and Ivan Seskar. 2010. Security and privacy vulnerabilities of In-Car wireless networks: A tire pressure monitoring system case study. In *USENIX Security Symposium, vol. 10*.
[60] ISO. 2016. *ISO 17987: Road Vehicles—Local Interconnect Network (LIN) Part 1: General Information and Use Case Definition*. Technical Report. Retrieved August 2020 from https://www.iso.org/standard/61222.html.
[61] ISO 10681-1:2010. 2010. *Road Vehicles—Communication on FlexRay—Part 1: General Information and Use Case Definition*. Retrieved August 2020 from https://www.iso.org/obp/ui/#iso:std:iso:10681:-1:ed-1:v1:en.
[62] Shriram Jadhav and Deepak Kshirsagar. 2018. A survey on security in automotive networks. In *Proceedings of the 2018 4th International Conference on Computing, Communication Control and Automation (ICCUBEA'18)*.
[63] Haojie Ji, Yunpeng Wang, Hongmao Qin, Yongjian Wang, and Honggang Li. 2018. Comparative performance evaluation of intrusion detection methods for In-Vehicle networks. *IEEE Access* 6 (Jun. 2018), 37523–37532.
[64] Juniper Research. 2018. In-Vehicle Commerce Connected Cars to Exceed 775 Million by 2023. Retrieved August 2020 from https://www.juniperresearch.com/press/press-releases/in-vehicle-commerce-opportunities-exceed-775mn.
[65] Ki Dong Kang, Youngmi Baek, Seonghun Lee, and Sang Hyuk Son. 2017. An attack-resilient source authentication protocol in controller area network. In *Proceedings of the 2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'17)*, 109–118.
[66] Min Joo Kang and Je Won Kang. 2016. Intrusion detection system using deep neural network for in-vehicle network security. *PLoS One* 11, 6 (2016), 1–17.

[67] Tae Un Kang, Hyun Min Song, Seonghoon Jeong, and Huy Kang Kim. 2018. Automated reverse engineering and attack for CAN using OBD-II. In *Proceedings of the IEEE Vehicular Technology Conference*, 1–7.

[68] Sen Nie, Ling Liu, and Yuefeng Du. 2017. Free-fall: hacking tesla from wireless to can bus. *Defcon* (2017), 1–16. Retrieved on August 2020 from https://www.blackhat.com/docs/us-17/thursday/us-17-Nie-Free-Fall-Hacking-Tesla-From-Wireless-To-CAN-Bus-wp.pdf.

[69] Jihas Khan. 2017. Vehicle network security testing. In *Proceedings of the 2017 3rd IEEE International Conference on Sensing, Signal Processing and Security (ICSSS'17)*, 119–123.

[70] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon Mccoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Hovav Snachám, and Stefan Savage. 2010. Experimental security analysis of a modern automobile. In *Proceedings of the IEEE Symposium on Security and Privacy*, 447–462.

[71] Takuma Koyama, Toshiki Shibahara, Keita Hasegawa, Yasushi Okano, Masashi Tanaka, and Yoshihito Oshima. 2019. Anomaly detection for mixed transmission CAN messages using quantized intervals and absolute difference of payloads. In *Proceedings of the ACM Workshop on Automotive Cybersecurity, co-located with CODASPY 2019 (AutoSec'19)* - (2019), 19–24.

[72] John T. Krzeszewski. 2019. *Vector Cybersecurity Symposium 2019–ISO 21434–Current Status*. Technical Report. Retrieved August 2020 from https://assets.vector.com/cms/content/events/2019/vSES19/vSES19_03_Krzeszewski_Aptiv.pdf.

[73] Ryo Kurachi, Yutaka Matsubara, Hiroaki Takada, Naoki Adachi, Yukihiro Miyashita, and Satoshi Horihata. 2014. CaCAN—Centralized authentication system in CAN. In *Proceedings of the 12th Embedded Security in Cars Europe* .

[74] Ulf E. Larson, Dennis K. Nilsson, and Erland Jonsson. 2008. An approach to specification-based attack detection for in-vehicle networks. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, 220–225.

[75] Hyunsung Lee, Seong Hoon Jeong, and Huy Kang Kim. 2018. 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'17)*, 57–66.

[76] Matan Levi, Yair Allouche, and Aryeh Kontorovich. 2018. Advanced analytics for connected car cybersecurity. In *Proceedings of the IEEE Vehicular Technology Conference*, 1–7.

[77] Chung Wei Lin and Alberto Sangiovanni-Vincentelli. 2012. Cyber-security for the controller area network (CAN) communication protocol. In *Proceedings of the 2012 ASE International Conference on Cyber Security (CyberSecurity'12)*, 1–7.

[78] Congli Ling and Dongqin Feng. 2012. An algorithm for detection of malicious messages on CAN buses. In *Proceedings of the 2012 National Conference on Information Technology and Computer Science (CITCS'12)*.

[79] Meng-Zhuo Liu, Yi-Hu Xu, Yu-Jing Wu, and Yi-Nan Xu. 2018. Research of authenticated encryption security protocol for FlexRay in-vehicle network. *Int. J. Comput. Theory Eng.* 10, 5 (2018), 175–179.

[80] Siti Farhana Lokman. 2019. Stacked sparse autoencoders-based outlier discovery for in-vehicle stacked sparse autoencoders-based outlier discovery for in-vehicle controller area network (CAN). *Int. J. Eng. Technol.* 7, August (2019), 375–380.

[81] Siti Farhana Lokman, Abu Talib Othman, Shahrulniza Musa, and Muhamad Husaini Abu Bakar. 2019. Deep contractive autoencoder-based anomaly detection for in-vehicle controller area network (CAN). In *Advanced Structured Materials*. Vol. 119. Springer Verlag, 195–205.

[82] George Loukas, Tuan Vuong, Ryan Heartfield, Georgia Sakellari, Yongpil Yoon, and Diane Gan. 2017. Cloud-based cyber-physical intrusion detection for vehicles using deep learning. *IEEE Access* 6 (2017), 3491–3508.

[83] Mirco Marchetti and Dario Stabili. 2017. Anomaly detection of CAN bus messages through analysis of ID sequences. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, 1577–1583.

[84] Mirco Marchetti, Dario Stabili, Alessandro Guido, and Michele Colajanni. 2016. Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms. In *Proceedings of the 2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a Better Tomorrow (RTSI'16)*, 1–6.

[85] Fabio Martinelli, Francesco Mercaldo, Vittoria Nardone, and Antonella Santone. 2017. Car hacking identification through fuzzy logic algorithms. In *Proceedings of the IEEE International Conference on Fuzzy Systems*.

[86] Microchip. 2003. Mcp2515 notes. 94. Retrieved August 2020 from http://ww1.microchip.com/downloads/en/DeviceDoc/MCP2515-Stand-Alone-CAN-Controller-with-SPI-20001801J.pdf.

[87] Charlie Miller and Chris Valasek. 2014. A survey of remote automotive attack surfaces. *Technical White Paper* (2014), 1–90.

[88] Charlie Miller and Chris Valasek. 2015. Remote exploitation of an unaltered passenger vehicle. *Defcon* 23, 2015 (2015), 1–91. Retrieved August 2020 https://www.academia.edu/download/53311546/Remote_Car_Hacking.pdf.

[89] Charlie Miller and Chris Valasek. 2016. OG Dynamite Edition. Retrieved August 2020 from http://illmatics.com/can%20message%20injection.pdf.

[90] Nazeeruddin Mohammad, Shahabuddin Muhammad, and Eman Shaikh. 2019. Analysis of in-vehicle security system of smart vehicles. In *Communications in Computer and Information Science*, Vol. 1113 CCIS. Springer, 198–211.

[91] Nicky Mouha, Bart Mennink, Anthony Van Herrewege, Dai Watanabe, Bart Preneel, and Ingrid Verbauwhede. 2014. Chaskey: An efficient MAC algorithm for 32-bit microcontrollers. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 8781. Springer Verlag, 306–323.

[92] Ahmed Refaat Mousa, Pakinam NourElDeen, Marianne Azer, and Mahmoud Allam. 2016. Lightweight authentication protocol deployment over FlexRay. In *ACM International Conference Proceeding Series*, Vol. 09. 233–239.

[93] Pal-Stefan Murvay and Bogdan Groza. 2020. Efficient physical layer key agreement for FlexRay networks. *IEEE Trans. Vehic. Technol.* 9545, c (2020), 1–1.

[94] Michael Müter and Naim Asaj. 2011. Entropy-based anomaly detection for in-vehicle networks. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, 1110–1115.

[95] Michael Müter, André Groll, and Felix C. Freiling. 2010. A structured approach to anomaly detection for in-vehicle networks. In *Proceedings of the 2010 6th International Conference on Information Assurance and Security (IAS'10)*, 92–98.

[96] Sandeep Nair Narayanan, Sudip Mittal, and Anupam Joshi. 2016. OBD_SecureAlert: An anomaly detection system for vehicles. In *Proceedings of the 2016 IEEE International Conference on Smart Computing (SMARTCOMP'16)*. IEEE, 1–6.

[97] National Instruments. 2019. FlexRay Automotive Communication Bus Overview. Retrieved August 2020 from http://www.ni.com/white-paper/3352/en/.

[98] National Instruments. 2020. Controller Area Network. Retrieved August 2020 from https://www.ni.com/en-lb/innovations/white-papers/06/controller-area-network--can--overview.html.

[99] Nicolas Navet, Yeqiong Song, Françoise Simonot-Lion, and Cédric Wilwert. 2005. Trends in automotive communication systems. *Proc. IEEE* 93, 6 (2005), 1204–1222.

[100] Netronics Ltd. 2020. CANdo—CAN Bus Analyser. Retrieved August 2020 from http://www.cananalyser.co.uk/index.html.

[101] Dennis K. Nilsson, Ulf E. Larson, and Erland Jonsson. 2008. Efficient in-vehicle delayed data authentication based on compound message authentication codes. In *Proceedings of the IEEE Vehicular Technology Conference* (2008), 1–5.

[102] Dennis K. Nilsson, Ulf E. Larson, Francesco Picasso, and Erland Jonsson. 2009. A first simulation of attacks in the automotive network communications protocol flexRay. *Advances in Soft Computing* 53 (2009), 84–91

[103] Nasser Nowdehi, Aljoscha Lautenbach, and Tomas Olovsson. 2018. In-vehicle CAN message authentication: An evaluation based on industrial criteria. In *Proceedings of the IEEE Vehicular Technology Conference*, 1–7.

[104] NXP. 2020. Evaluation Board for the 16-bit MC9S12XE and XS-Families | NXP. Retrieved August 2020 from https://www.nxp.com/products/no-longer-manufactured/evaluation-board-for-the-16-bit-mc9s12xe-and-xs-families:EVB9S12XEP100.

[105] Habeeb Olufowobi, Uchenna Ezeobi, Eric Muhati, Gaylon Robinson, Clinton Young, Joseph Zambreno, and Gedare Bloom. 2019. Anomaly detection approach using adaptive cumulative sum algorithm for controller area network. In *Proceedings of the ACM Workshop on Automotive Cybersecurity, co-located with CODASPY 2019 (AutoSec'19)*. ACM, 25–30.

[106] OptimumG. 2019. OptimumG | Vehicle Dynamics Solutions. Retrieved August 2020 from https://optimumg.com/.

[107] Pradeep Sharma Oruganti, Matt Appel, and Qadeer Ahmed. 2019. Hardware-in-loop based automotive embedded systems cybersecurity evaluation testbed. In *Proceedings of the ACM Workshop on Automotive Cybersecurity, co-located with CODASPY 2019 (AutoSec'19)* (2019), 41–44.

[108] Animesh Patcha and Jung Min Park. 2007. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Comput. Netw.* 51, 12 (Aug. 2007), 3448–3470.

[109] Constantinos Patsakis, Kleanthis Dellios, and Mélanie Bouroche. 2014. Towards a distributed secure in-vehicle communication architecture for modern vehicles. *Comput. Secur.* 40 (2014), 60–74.

[110] Krzysztof Pawelec, Robert A. Bridges, and Frank L. Combs. 2019. Towards a CAN IDS based on a neural network data field predictor. In *Proceedings of the ACM Workshop on Automotive Cybersecurity, co-located with CODASPY 2019 (AutoSec'19)*. 31–34.

[111] Jonathan; Petit, Bas Stottelaar, Michael Feiri, and Frank Kargl. 2015. Remote Attacks on Automated Vehicles Sensors: Experiments on Camera and LiDAR. *Blackhat.com* (2015), 1–13. Retrieved January 2021 from https://www.blackhat.com/docs/eu-15/materials/eu-15-Petit-Self-Driving-And-Connected-Cars-Fooling-Sensors-And-Tracking-Drivers-wp1.pdf.

[112] Olaf Pfeiffer and Christian Keyde. 2018. Security expectations vs. limitations. 22–25. Retrieved January 2021 from https://can-newsletter.org/uploads/media/raw/8a34f7f0d457d109ac17e6a791c4e0dc.pdf.

[113] Philip E. Ross. 2020. Mercedes and Nvidia Announce the Advent of the Software-Defined Car—IEEE Spectrum. Retrieved August 2020 from https://spectrum.ieee.org/cars-that-think/transportation/self-driving/mercedes-and-nvidia-announce-the-advent-of-the-softwaredefined-car.

[114] J. Pradeep, S. Richerd Sebasteen, and R. Dineshkrishna. 2018. *Comparison of CAN and Flexray Protocol for Automotive Application View project Source Protected Distribution and Scrutinize Action for Public Auditing Protocol in Cloud Data View project Pr*. Technical Report.

[115] Dominik Püllen, Nikolaos Athanasios Anagnostopoulos, Tolga Arul, and Stefan Katzenbeisser. 2019. Security and safety co-engineering of the flexray bus in vehicular networks. In *Proceedings of theACM International Conference Proceeding Series* Part F1481 (2019), 31–37.

[116] C. Quigley, A. Williams, and R. McLaughlin. 2013. *iCC 2013 CAN in Automation the Potential of CAN FD Technology to Impact upon FlexRay*. Technical Report.

[117] Andreea Ina Radu and Flavio D. Garcia. 2016. LeiA: A lightweight authentication protocol for CAN. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 9879. 283–300.

[118] Reiner Zitzmann. 2019. *CiA CANopenFD Integration Workshop*. Technical Report. CAN in Automation. Retrieved August 2020 from https://www.automa.cz/downloads/streda15-00.pdf.

[119] Robert Bosch GmbH. 2020. CAN XL News text | Bosch Semiconductors. Retrieved August 2020 from https://www.bosch-semiconductors.com/news/t-newsdetailpage-4.html.

[120] Ishtiaq Rouf, Rob Miller, Hossen Mustafa, Travis Taylor, Sangho Oh, Wenyuan Xu, Marco Gruteser, Wade Trappe, and Ivan Seskar. 2010. Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In *Proceedings of the 19th USENIX Security Symposium*. 323–338.

[121] SAE International. 2020. Requirements for hardware-protected security for ground vehicle applications—J3101. *SAE* (2020). Retrieved August 2020 from https://www.sae.org/standards/content/j3101_202002/.

[122] Sang Uk Sagong, Radha Poovendran, and Linda Bushnell. 2019. Mitigating vulnerabilities of voltage-based intrusion detection systems in controller area networks. *arXiv:1907.10783*. Retrieved from https://arxiv.org/abs/1907.10783.

[123] Upstream Security and Global Automotive. 2020. *Upstream Security's Global Automotive Cybersecurity Report*. Technical Report. Retrieved August 2020 from https://www.upstream.auto/upstream-security-global-automotive-cybersecurity-report-2020/.

[124] Eunbi Seo, Hyun Min Song, and Huy Kang Kim. 2018. GIDS: GAN based intrusion detection system for in-vehicle network. In *Proceedings of the 2018 16th Annual Conference on Privacy, Security and Trust (PST'18)*.

[125] Hervé Seudié. 2009. Vehicular on-board security: EVITA project project. In *Forum American Bar Association*.

[126] Kim Seung-Han, Seo Suk-Hyun, Kim Jin-Ho, Moon Tae-Yoon, Son Chang-Wan, Hwang Sung-Ho, and Jeon Jae Wook. 2008. A gateway system for an automotive system: LIN, CAN, and flexray. In *Proceedings of the IEEE International Conference on Industrial Informatics (INDIN'08)*, 967–972.

[127] SeVeCom (Secure Vehicular Communication). 2008. Sevecom. Retrieved August 2020 from https://sevecom.eu/.

[128] Priyanka Sharma and Dietmar P. F. Moller. 2018. Protecting ECUs and vehicles internal networks. In *Proceedings of the IEEE International Conference on Electro Information Technology*, 465–470.

[129] Barry Sheehan, Finbarr Murphy, Martin Mullins, and Cian Ryan. 2019. Connected and autonomous vehicles: A cyber-risk classification framework. *Transport. Res. A: Policy Pract.* 124, (Nov. 2019), 523–536.

[130] Ali Shuja Siddiqui, Yutian Gui, Jim Plusquellic, and Fareena Saqib. 2017. Secure communication over CANBus. In *IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS'17)*. 1264–1267.

[131] Hyun Min Song, Ha Rang Kim, and Huy Kang Kim. 2016. Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. In *Proceedings of the International Conference on Information Networking*, 63–68.

[132] Hyun Min Song, Jiyoung Woo, and Huy Kang Kim. 2020. In-vehicle network intrusion detection using deep convolutional neural network. *Vehic. Commun.* 21 (2020), 100198.

[133] Dario Stabili, Mirco Marchetti, and Michele Colajanni. 2017. Detecting attacks to internal vehicle networks through Hamming distance. In *Proceedings of the 2017 AEIT International Annual Conference: Infrastructures for Energy and ICT: Opportunities for Fostering Innovation (AEIT'17)*, 1–6.

[134] Standard of Automotive Engineering. 2016. J3061A (WIP) Cybersecurity Guidebook for Cyber-Physical Vehicle Systems—SAE International. Retrieved August 2020 from https://www.sae.org/standards/content/j3061/.

[135] Standard of Automotive Engineering. 2018. J3138: Diagnostic Link Connector Security—SAE International. Retrieved August 2020 from https://www.sae.org/standards/content/j3138_201806/.

[136] Ivan Studnia, Eric Alata, Vincent Nicomette, Mohamed Kaâniche, and Youssef Laarouchi. 2018. A language-based intrusion detection approach for automotive embedded networks. *Int. J. Embed. Syst.* 10, 1 (2018), 1–12.

[137] Hiroki Suda, Masanori Natsui, and Takahiro Hanyu. 2018. Systematic intrusion detection technique for an in-vehicle network based on time-series feature extraction. In *Proceedings of the International Symposium on Multiple-Valued Logic*, 56–61.

[138] A. S. Syed Navaz, V. Sangeetha, C. Prabhadevi, A. S. Syed Navaz, V. Sangeetha, and C. Prabhadevi. 2013. Entropy based anomaly detection system to prevent DDoS attacks in cloud. *Int. J. Comput. Appl.* 62, 15 (2013), 42–47.

[139] Adrian Taylor, Nathalie Japkowicz, and Sylvain Leblanc. 2015. Frequency-based anomaly detection for the automotive CAN bus. In *Proceedings of the 2015 World Congress on Industrial Control Systems Security (WCICSS'15)*. Infonomics Society, 45–49.

[140] Adrian Taylor, Sylvain Leblanc, and Nathalie Japkowicz. 2016. Anomaly detection in automobile control network data with long short-term memory networks. In *Proceedings of the 3rd IEEE International Conference on Data Science and Advanced Analytics (DSAA'16)*, 130–139.

[141] The Institution of Engineering and Technology. 2020. Serious Cyber-security Flaws Uncovered in Ford and Volkswagen Cars. Retrieved August 2020 from https://eandt.theiet.org/content/articles/2020/04/serious-cyber-security-flaws-uncovered-in-ford-and-volkswagen-cars-that-could-endanger-drivers/?utm_source=Adestra&utm_campaign=New EandTNews-AutomationFINAL-MEMBER&utm_medium=Newsletters-E%26TNew.

[142] Andreas Theissler. 2014. Anomaly detection in recordings from in-vehicle networks. In *First Int. Workshop on Big Data Applications and Principles (BIGDAP'14)*.

[143] Daxin Tian, Yuzhou Li, Yunpeng Wang, Xuting Duan, Congyu Wang, Wenyang Wang, Rong Hui, and Peng Guo. 2018. An intrusion detection system based on machine learning for CAN-Bus. In *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, Vol. 221, 285–294.

[144] Andrew Tomlinson, Jeremy Bryans, and Siraj Ahmed Shaikh. 2018. Towards viable intrusion detection methods for the automotive controller area network. In *Proceedings of the Computer Science in Cars Conference (CSCS'18)*.

[145] Andrew Tomlinson, Jeremy Bryans, and Siraj Ahmed Shaikh. 2018. Using a one-class compound classifier to detect in-vehicle network attacks. In *Proceedings of the 2018 Genetic and Evolutionary Computation Conference Companion (GECCO'18)*, 1926–1929.

[146] Andrew Tomlinson, Jeremy Bryans, Siraj Ahmed Shaikh, and Harsha Kumara Kalutarage. 2018. Detection of automotive CAN cyber-attacks by identifying packet timing anomalies in time windows. In *Proceedings of the 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W'18)*, 231–238.

[147] Hiroshi Ueda, Ryo Kurachi, Hiroaki Takada, Tomohiro Mizutani, Masayuki Inoue, and Satoshi Horihata. 2015. Security authentication system for in-vehicle network. *SEI Techn. Rev.*81 (2015), 5–9.

[148] Vector Informatik GmbH. [n.d.]. Testing ECUs and networks with CANoe. Retrieved Retrieved August 2020 from https://www.vector.com/int/en/products/products-a-z/software/canoe/.

[149] David Wampler, Huirong Fu, and Ye Zhu. 2009. Security threats and countermeasures for intra-vehicle networks. *Proceedings of the 5th International Conference on Information Assurance and Security (IAS'09)*, 153–157.

[150] Chundong Wang, Zhentang Zhao, Liangyi Gong, Likun Zhu, Zheli Liu, and Xiaochun Cheng. 2018. A distributed anomaly detection system for in-vehicle network using HTM. *IEEE Access* 6 (2018), 9091–9098.

[151] Qian Wang, Zhaojun Lu, and Gang Qu. 2019. An entropy analysis based intrusion detection system for controller area network in vehicles. *Int. Syst. Chip Conf.* 174–179.

[152] Qiyan Wang and Sanjay Sawhney. 2014. VeCure: A practical security framework to protect the CAN bus of vehicles. In *Proceedings of the 2014 International Conference on the Internet of Things (IOT'14)*. IEEE, 13–18.

[153] Marc Weber, Simon Klug, Eric Sax, Bastian Zimmer, Marc Weber, Simon Klug, Eric Sax, Bastian Zimmer, Embedded Hybrid, Anomaly Detection, Marc Weber, Simon Klug, Eric Sax, and Bastian Zimmer. 2018. Embedded hybrid anomaly detection for automotive CAN communication. In *Proceedings of the Embedded Real Time Software and Systems (ERTS2'18)*.

[154] Marko Wolf and Timo Gendrullis. 2012. Design, implementation, and evaluation of a vehicular hardware security module. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 7259, 302–318.

[155] Samuel Woo, Hyo Jin Jo, In Seok Kim, and Dong Hoon Lee. 2016. A practical security architecture for in-vehicle CAN-FD. *IEEE Trans. Intell. Transport. Syst.* 17, 8 (2016), 2248–2261.

[156] Samuel Woo, Hyo Jin Jo, and Dong Hoon Lee. 2015. A practical wireless attack on the connected car and security protocol for in-vehicle CAN. *IEEE Trans. Intell. Transport. Syst.* 16, 2 (2015), 993–1006.

[157] Wufei Wu, Yizhi Huang, Ryo Kurachi, Gang Zeng, Guoqi Xie, Renfa Li, and Keqin Li. 2018. Sliding window optimized information entropy analysis method for intrusion detection on in-vehicle networks. *IEEE Access* 6 (2018), 45233–45245.

[158] Wufei Wu, Renfa Li, Guoqi Xie, Jiyao An, Yang Bai, Jia Zhou, and Keqin Li. 2019. A survey of intrusion detection for in-vehicle networks. *IEEE Trans. Intell. Transport. Syst.* 21, 3 (2019), 919–933.

[159] Yujing Wu, Yeon Jin Kim, Zheyan Piao, Jin Gyun Chung, and Yong En Kim. 2016. Security protocol for controller area network using ECANDC compression algorithm. In *Proceedings of the IEEE International Conference on Signal Processing, Communications and Computing, Conference Proceedings (ICSPCC'16)* (2016), 1–4.

[160] Clinton Young, Habeeb Olufowobi, Gedare Bloom, and Joseph Zambreno. 2019. Automotive intrusion detection based on constant CAN message frequencies across vehicle driving modes. In *Proceedings of the ACM Workshop on Automotive Cybersecurity*. 9–14.

[161] Jiayan Zhang, Fei Li, H. Zhang, Ruxiang Li, and Y. Li. 2019. Intrusion detection system using deep learning for in-vehicle security. *Ad Hoc Netw.* 95 (Dec. 2019), 101974.

[162] Zeljka Zorz. 2018. Backdooring connected cars for covert remote control—Help Net Security. Retrieved August 2020 from https://www.helpnetsecurity.com/2018/03/05/backdooring-connected-cars/.

Received April 2020; revised August 2020; accepted October 2020