CYBERATTACKS AND COUNTERMEASURES FOR IN-VEHICLE NETWORKS

Emad Aliwa
School of Computer Science and Informatics
Cardiff University, UK
aliwaem@cardiff.ac.uk

Omer Rana
School of Computer Science and Informatics
Cardiff University, UK
RanaOF@cardiff.ac.uk

Charith Perera
School of Computer Science and Informatics
Cardiff University, UK
PereraC@cardiff.ac.uk

Peter Burnap
School of Computer Science and Informatics
Cardiff University, UK
BurnapP@cardiff.ac.uk

April 24, 2020

ABSTRACT

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 not only vulnerable 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 constrains such as low bandwidth, small frame size (e.g. in the CAN protocol), limited availability of computational resources and real-time sensitivity. We survey In-Vehicle Network (IVN) attacks which have been grouped under: direct interfaces-initiated attacks, telematics and infotainment-initiated attacks, and sensor-initiated attacks. We survey and classify current cryptographic and IDS approaches and compare these approaches based on criteria such as real time constrains, types of hardware used, changes in CAN bus behaviour, types of attack mitigation and software/ hardware used to validate these approaches. We conclude with potential mitigation strategies and research challenges for the future.

Keywords CAN bus, Cyberattacks, Cybersecurity , Intrusion Detection System, Cryptography, Connected cars

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 different attacks not only from inside but also from outside the vehicle. For instance, recent report has indicated that two famous connected cars in Europe from Ford and Volkswagen are vulnerable to cyberattacks from infotainment unit [1]. As 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 [2]. 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 [3]. 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 On-Board Diagnostics port (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. This survey provides the following contributions:

- description of in-vehicle serial bus protocols (particularly the CAN bus);
- evaluation of current cryptographic and IDS approaches used for protecting vehicular data;
- comparison and assessment of current mitigation strategies to protect vehicles against cyberattacks;
- challenges and potential future research directions for in-vehicular cybersecurity.

The rest of this paper is structured as follows: an introduction to serial data exchange protocols within a vehicle is provided in Section 2. This material provides the context for the rest of this paper – outlining key concepts and terminology. In Section 3 the CAN bus protocol, bus architecture along with hardware and software used inside vehicles 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 reported in literature to secure the CAN bus, which includes encryption, message authentication and vehicular Intrusion Detection Systems (IDS). We evaluate existing approaches based on criteria such as real time data requirements, the types of network infrastructure required, computational resource constrains, 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 mentioned in section 6 and discuss future research challenges.

2 Automotive Serial Bus Protocols

Three key protocols are used inside vehicles for data commuications: CAN, FlexRay and LIN. Figure 1 shows the serial bus protocols used inside a vehicle. The CAN bus protocol is the most widely used to support critical functions such as Powertrain, engine management, anti-brake system and transmission. A vehicular system is divided into four domains, in terms of the function it performs and whether it requires real time data [4], as outlined below:

- **Power train domain** such as engine and transmission functions. This domain is critical and needs real time response;
- **Chassis domain** such as braking system, suspension and steering which also provides real time and safety critical functions inside the vehicle;
- **Body domain** for functions such as dashboard, wipers, lights, windows and seats. These functions do not generally require real time response;
- **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. For example, a high speed CAN bus is used for real time and safety related domains inside the vehicle such as transmission sensors. LIN and low speed CAN bus are however used for non-critical functions inside a vehicle, as they are less expensive compared with the high speed CAN and FlexRay buses.

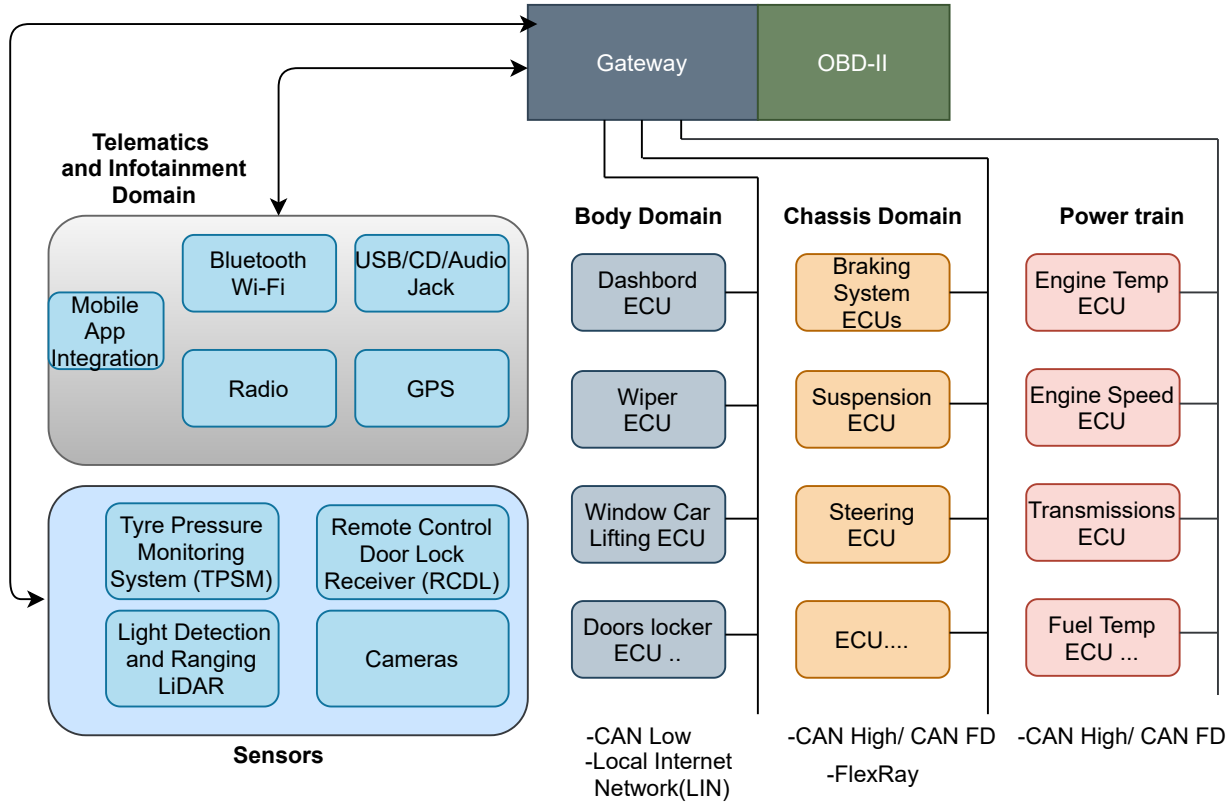


Figure 1: Serial bus protocols inside a vehicle – from [5]. The figure focuses on the three most popular protocols: Controller Area Network (CAN), Local InterConnect Network(LIN) and FlexRay.

2.1 Controller Area Network (CAN)

CAN is serial communication protocol used for real-time, safety critical functions inside road vehicles and other controlled applications [6]. It is a multi-master protocol and most widely used inside vehicles [7]. Below are the main characteristics of the CAN protocol:

- the maximum bitrate is 1Mbps in the *classical* CAN bus;
- high speed CAN bus bitrate can vary from 125kbps to 1Mbps, while low speed CAN bus bitrate from 5kbps to 125kbps;
- critical sensors can be connected to high speed CAN bus while less critical sensors can be connected to a low speed CAN bus;
- in CAN Flexible Data rate (FD), the bitrate is up to 8 Mbps – with a payload size of 8 bytes in Classical CAN and up to 64 bytes in CAN FD;
- CAN bus protocol provides real time access for critical sensors via the Carrier Sense Multiple Access with Collision Avoidance and Arbitration Priority (CSMA/CA-AP) access/arbitration mechanism;
- CAN bus is implemented physically using the twisted pair cable;
- CAN bus is used in real time automotive applications such as engine management, transmission, braking system and steering;
- CAN bus provides error detection and correction mechanisms;
- CAN XL is expected to be announced in 2020 [8], and is the third generation of CAN which provides up to 2048 bytes of data payload and bitrate of up to 10Mbps [9]. This generation of the protocols is expected to be used with Internet Protocol (IP) based services.

As the CAN bus is the most widely used protocol inside a vehicle, a more detailed description of this protocol is provided in Section 3.

2.2 Local Interconnect Network (LIN)

The LIN protocol is used for low cost vehicular applications which have a low bit rate, asynchronous data requirement. It is used for non-critical applications such as door module and air condition systems [10]. LIN is used where the reliability and robustness of the network is not critical [11]. It is standardised in International Organization for Standardization (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 etc [10]. The LIN protocol has the following features:

- a bit rate ranging from 1kbps to 20kbps;
- single master with multiple slave nodes – with support for up to 16 nodes;
- a single physical wire to realise the bus;
- standardised as ISO standard 17987.

2.3 FlexRay

It is a protocol used for high bitrate requirements, with error detection and correction, redundancy and safety [11]. It is used for high end applications inside vehicles such as power train and safety functions (adaptive cruise control and active suspension) [12]. The protocol is standardised under ISO 17458 series which describes the physical layer and data link layer characteristics [13]. The main features of the protocol are:

- A bit rate of up to 10 Mbps with half-duplex bus access;
- standardised as ISO standard 17458;
- support for fault tolerant mechanisms;
- designed to work for high speed and safety critical applications (e.g. braking-by-wire and steering-by-wire)

Since the CAN bus protocol is the most widely used, this review paper will focus on this protocol in terms of attacks, vulnerabilities and potential countermeasures.

Table 1: Automotive serial communication protocols. The table shows Controller Area Network (CAN and CAN FD), Local Internet Network (LIN) and FlexRay

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

3 Controller Area Network (CAN)

Initial use of CAN bus within a vehicle did not consider security, as vehicles at that time were not connected to 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 connected, 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 [14]. 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 2 wires [15]. This reduces the cost of wiring compared to a point to point wiring mechanism and reduces the negative effects of external noise through its CAN-High and CAN-Low (signal differential) transmission [14] [16]. It works on the physical and data link layers, although it does not use Media Access Control addresses (MAC) and MAC tables to send and receive (route) frames. Instead, it uses message 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 does not use a 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 CAN protocol itself does not provide message authentication and so it is vulnerable to cyberattacks such as CAN frame injections. 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 [17].

3.1 Standard CAN bus Frame 2.0A

The classical CAN bus was standardised in 1993 in ISO 11898. It consists of two versions based on the message identifier length. The standard CAN bus 2.0A has an 11-bit identifier while the extended CAN bus 2.0B has a 29-bit identifier.

3.2 Extended CAN bus Frame 2.0B

This extended CAN bus provides a 29-bit identifier which gives more message ids, and hence a greater number of potential nodes that can be supported. The data payload is up to 8bytes. CAN frame structure is illustrated in figure 2

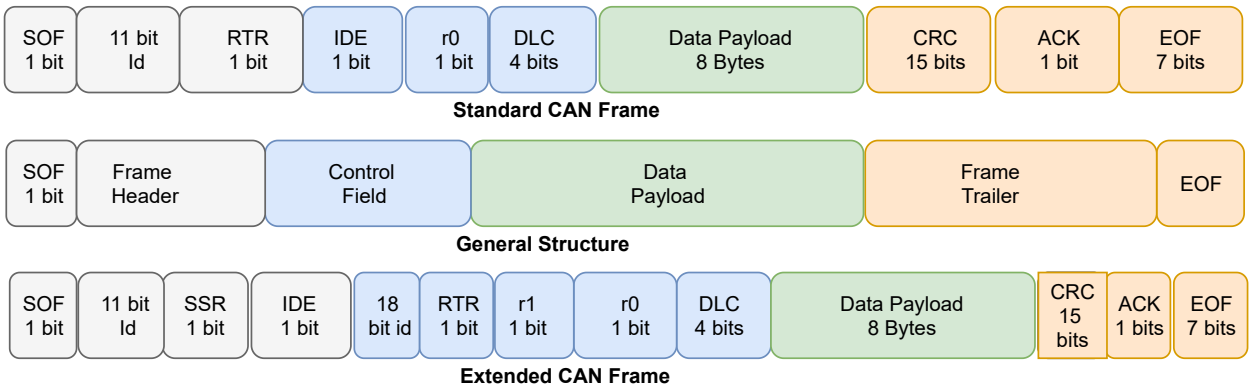


Figure 2: Standard and extended Controller Area Network CAN bus frame – showing the Frame header fields, Control fields, Data payload fields and Frame trailer

The CAN protocol frame consists of the fields shown in figure 2. A classical CAN frame [18], [14] consists of: the arbitration field, the message identifier field and the Remote Transmission Request (RTR) bit.

1. **Frame header**: The frame header contains the message identifier (11-bits or 29-bits) which identifies the priority and the content of the message. It is worth noting that a CAN bus does not provide source and destination addresses like IP networks, instead it uses unique CAN identifier numbers in each message.
   - **Start of the Frame field (SOF)** has a dominant value of 1.
   - **11-bit identifier** shows the message id which determines the priority of the message where the lowest value means the highest priority. Also, it is used to represent the content of the message.
   - **29-bit identifier** – as part of the extended frame has similar structure as the standard frame except additional fields in the frame header and control field such as the following: **The frame header** consists of a base identifier (11-bits) and the extended identifier 18-bits which both provide the priority of the frame and its content.
   - **Remote Transmission Request (RTR)** is used to retrieve information from a node in the network.
2. **Control fields** The control field consists of: Identifier Extension Flag (IDE), rO and Data length Control (DLC) flag:
   - **IDE** Identifier Extension is used to distinguish between standard and extended frame
   - **r0**field is a 1-bit value and reserved and it always has the recessive value of 0.
   - **DLC** field is a 4-bit value and indicates the length of the data in the data field
   - **SSR** Substitute Remote Request
   - **IDE** Identifier Extension Flag is used to identify the frame as standard (dominant bit) and in the extended frame the value is a recessive bit

- **R1 and r0** are reserved bits and they are always dominant.

3. **Data field**:This field contain the payload of up to 8-bytes.
4. **Frame trailer**: These fields are used to detect frame errors using checksum and correction mechanisms:
   - **CRC** (Cyclic Redundancy Code) field consists of 15bits and is used for checksum error detection. [19].
   - **ACK** or acknowledgement field is used to indicate that messages have been sent without any errors.
   - **EOF** indicates the End of the Frame and the message sequence.
   - **IFS** Inter Frame Space is 3bits in size and used to provide frame separation and initiate frame processing.

3.3 Controller Area Network with Flexible Data Rate (CAN FD)

CAN FD is developed to meet the needs of higher speed and more data payload size. It can provide up to 64bytes of data payload along with up to 8Mbps of speed [20]. It comes with a standard and an extended version similar to the classical CAN protocol. It is standardised in ISO 11898 series as well. The frame structure of the CAN FD protocol is illustrated in figure 3.

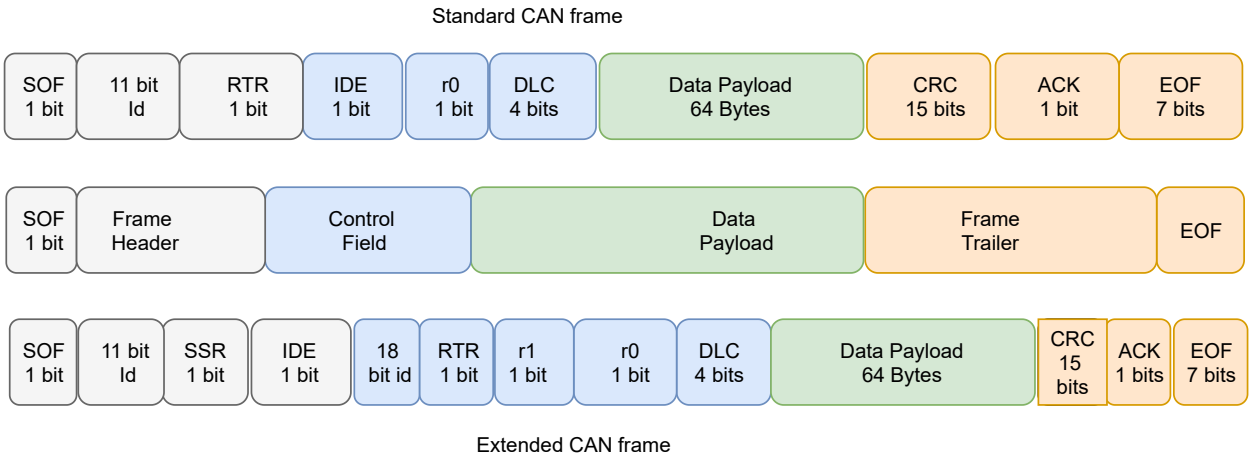


Figure 3: CAN FD 11- and 29-bit identifier frame structure

3.4 Frame Types of CAN bus Protocol

The CAN bus is divided into four types of frames [18]:

1. **Data Frame**: contains the data payload.
2. **Remote Frame**: used to ask for the transmission of data frame with the same identifier from another node on the bus. The difference between this frame and the data frame is that the RTR field inside the arbitration id is put in recessive and there is no data payload.
3. **Error frame** indicates that there is an error in the bus and this frame can be used by any node.
4. **Overload Frame** is used when a node on the bus is too busy to receive data from another node. When a CAN node transmits frames on the bus, it will be received by all other nodes connected to the bus due to its broadcast feature. The CAN controller board on a CAN node is responsible for handling the relevant frames. An error frame can be raised if an error occurs when receiving data and the remote frame is raised by the node to ask for a re-transmission of data.

The CAN standard was updated as ISO11898:2015 to include CAN FD [21]. The 11898 series describes the CAN bus data link and physical layer functions such as described in figure 4.

3.5 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:

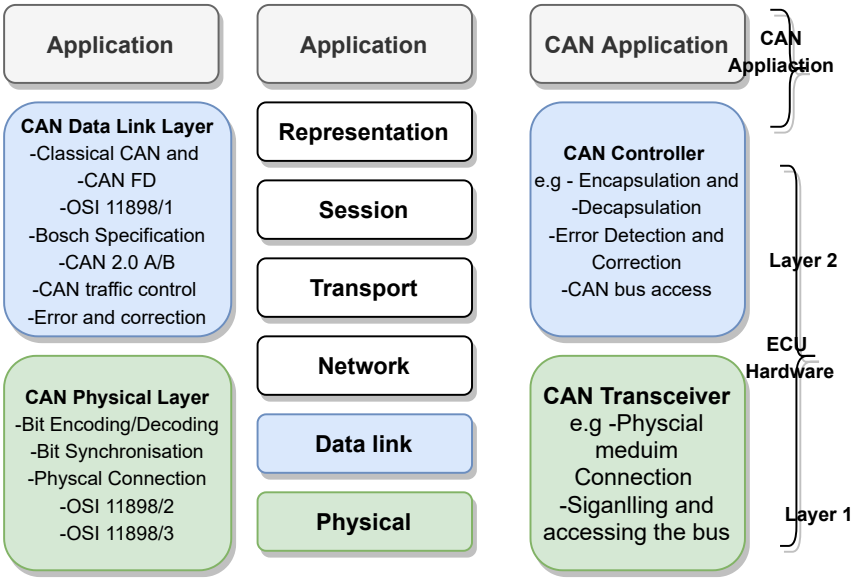


Figure 4: CAN bus protocol in OSI Model. CAN bus Hardware, Software and Standards in OSI layers

1. **ISO / SAE 21448:** This standard provides a guide from measurement to design, to enable the verification and validation of services inside vehicles [22]
2. **SAE J3061:** Provides a guide for cyber-physical systems within vehicles, and has been provided by the Society of Automotive Engineering [23].
3. **SAE J3101** standard provides the required features to support hardware security protection for vehicular applications [24]
4. **SAE J3138** (Diagnostic Link Connector Security): A standard used for diagnostics and security purposes for direct interfaces such as the on-board diagnostics (OBD-2) port inside a vehicle [25].
5. **ISO / SAE 21434:** This standard is used to support road vehicle cybersecurity engineering – expected to be released in 2020 [26]. 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 [26].

Other efforts to secure in-vehicle network infrastructure such as E-safety Vehicle Intrusion protected Applications (EVITA) and vehicle to infrastructure (V2I) such as Secure Vehicular Communication (SEVCOM):

1. **E-safety Vehicle Intrusion Protected Applications (EVITA)(2008-2011)**: [27]: This project focuses on securing in-vehicle network infrastructure from physical and remote attacks.
2. **Secure Vehicular Communication (SEVCOM) (2006-2008)**: [28] 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.
3. **Cooperative Vehicle-Infrastructure Systems (CVIS) (2006-2010)**: [29] a framework to provide V2I security and privacy for drivers and connected vehicles.

3.6 CAN bus Network Infrastructure

The CAN protocol focuses on the physical and data link layers. Extended protocols from industry such as J1939 (for heavy vehicles) and OBD-2 (for vehicular diagnostics) are built on top of the CAN data link and physical layers. The CAN layer functions can be identified [18] as follows and as shown in 5:

CAN Physical Layer (CPL) focuses on transmission of the signal across the CAN bus hardware.

CAN Data Link Layer protocol (CDLL) defines the core protocol (realised using the CAN chipset) such as bit timing, message framing, synchronisation, arbitration logic and error detection (e.g. use of CRC and ACK).

CAN High Layer Protocols (CHLP) (Application layer Protocols) use high speed CAN to provide real time information and diagnostic data exchange between ECUs. There are different high layer protocols in industry such as J1939 by the Society of Automotive Engineers (SAE) and used in heavy duty vehicles [30]. The J1939 protocol uses the 29bit version of the CAN bus protocol. Other protocols such as the protocol used with the On-Board Diagnostic (OBD-2) port are used for vehicle diagnostic and emissions analysis.

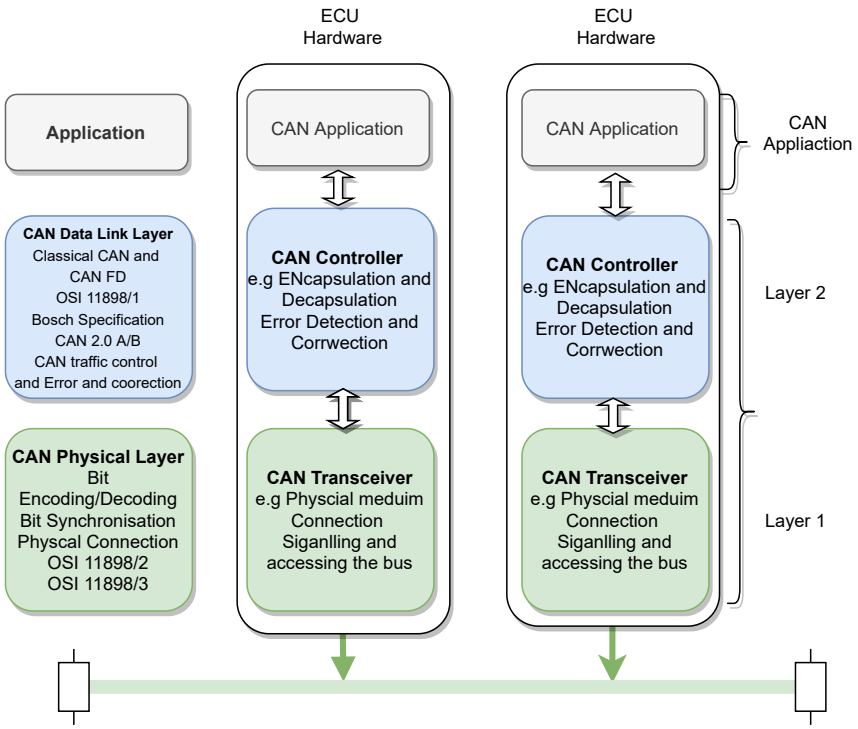


Figure 5: CAN Electronic Control Unit (ECU) hardware and software components, functions and standards

3.7 Automotive Application Layer Protocols

SAE J1939 is an application layer protocol widely used in commercial heavy vehicles such as coaches and agricultural vehicles. It works on top of the CAN 2.0B bus with a 29-bit extended frame providing a bitrate of 250Kbps to 500Kbps [31]. It also provides a standard message format and specification which allows using components from various manufacturers inside vehicles. Figure 6 illustrates CAN bus application layer.

3.8 Electronic Control Units (ECU)

Modern cars have around 70 ECUs which control various functions of the car, such as breaks, gears and engine status [11]. An ECU is primarily a microprocessor which 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 [32]. Figure 5 shows CAN Electronic Control Unit (ECU) hardware and software components, functions and standards in OSI model. Also, table 2 illustrates some ECUs which are used inside vehicular network systems.

3.9 CAN bus communication

The CAN protocol uses a broadcast based mechanism for message exchange [33] and each node can request use of the bus randomly. An arbitration mechanism is used to ensure priority on the bus [18], 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 [34]. 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, 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 [35][2]. Both CAN buses then are connected through a gateway ECU [2].

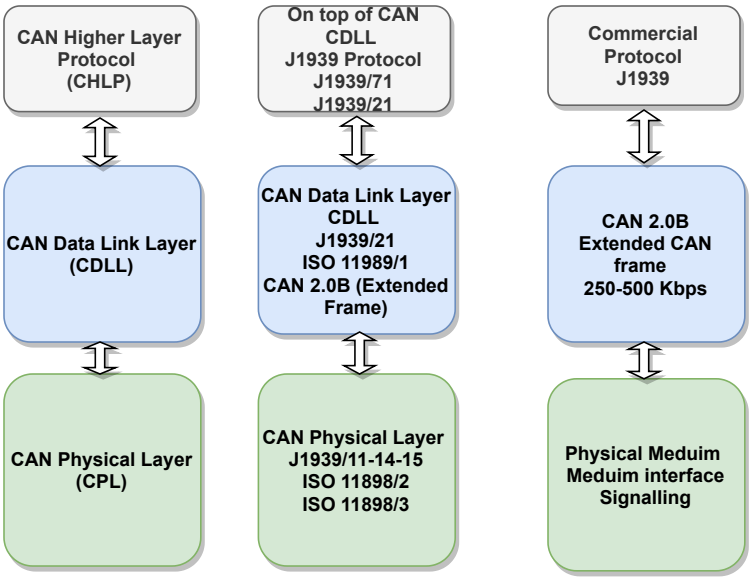


Figure 6: SAE J1939 CAN Application layer protocol

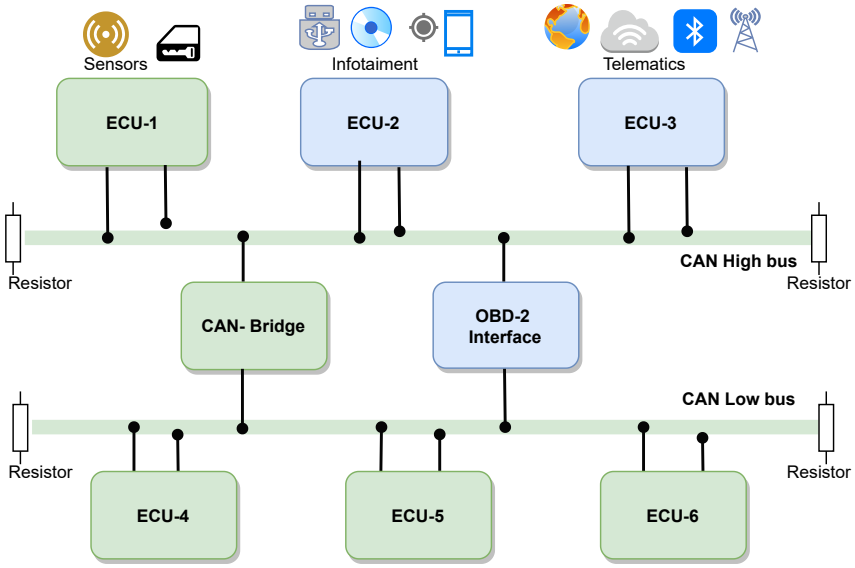


Figure 7: Two CAN buses connected to each other through a bridge – based on [36]

With this structure as outlined in figure 7, 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 Intrusion Detection Systems (IDSs) and firewalls to prevent unauthorised access, as well as using cryptography methods to provide confidentiality, integrity and authentication.

3.10 Protocol usage

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 [18]. It is used inside vehicles, built environments (e.g. controlling elevators in buildings, building energy management systems), railway applications, medical devices and aircrafts [37].

Table 2: Some Electronic Control Units inside CAN bus Network – based on [38]

Electronic Control Unit	CAN bus Connection	Critical
Engine Control Module (ECM)	High CAN bus	✓
Electronic Brake Control Module (EBCM)	High CAN bus	✓
Transmission Control Module (TCM)	High CAN bus	✓
Body Control Module (BCM)	High and Low CAN bus	×
Telematics Module (TM)	High and Low CAN bus	×
Remote Control Door Lock Receiver (RCDLR)	High CAN bus	✓
Heating, Ventilation, Air Conditioning	High CAN bus	×
Sensing and Diagnostic Module (SDM)	High CAN bus	✓
Instrument Panel Cluster/Driver Information Center	High CAN bus	×
Radio	High CAN bus	×
Theft Deterrent Module (TDM)	High CAN bus	✓

4 Connected Car Environment

Connected cars 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 [39]. This connectivity will be through telematics or by in-vehicle applications. Vehicles can be connected with either 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:

Telematics Unit: provides connectivity to the car using WiFi, Bluetooth, GPS and mobile data interfaces.

Infotainment Unit: 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.

Driver assistance Unit: 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. Also, Adaptive Cruise Control and Park Assist are used for measuring parking space and auto park assistance.

Vehicle 2 X: connected cars can also provide communication to cars (V2V) and roadside infrastructure (V2I) using wireless communication called Dedicated Short Range Communications (DSRC) which allow exchange 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 [40]. This can lead to vulnerabilities from the Bluetooth connection as legacy and vulnerable versions are still used – as described in [41][42] [40].

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 [43] and a Tesla S [44].

Cellular/phone network: modern cars can also provide mobile/phone/cellular connection which can be used to retrieve data such as weather conditions and traffic[43]. Attacks on such networks have also been identified [40]. For example, [43] have shown how a cellular network interface can be hacked inside a Jeep.

OBD-2 (On-Board Diagnostic) : It is a mandatory port which 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 attach which can affect vehicle operation and driver safety [45]. Various attacks have been demonstrated using direct connection to an OBD port and generating remote attacks using wireless OBD dongles [40]. 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 in order to inject this data into the CAN bus [46].

Global Position System (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 [47]. In [43] GPS information was retrieved from the head unit of a vehicle through unprotected 6667 port.

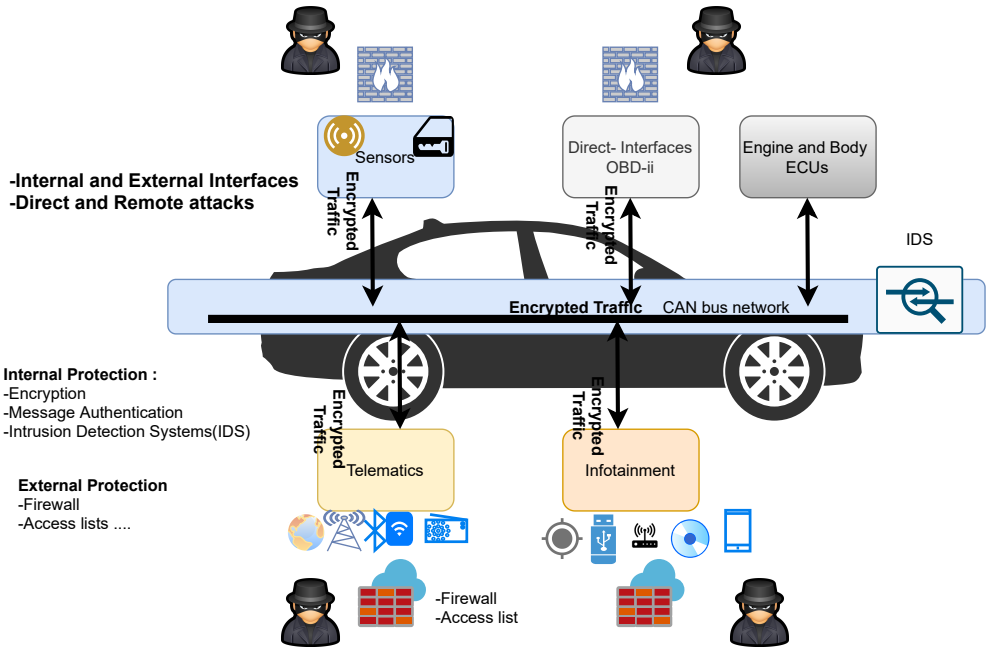


Figure 8: Connected Car environment with four potential entry points for data injection: Telematics, Infotainment, Direct interfaces and Sensors. Also, security countermeasures to detect and prevent physical and remote attacks using Cryptography, Intrusion Detection System (IDS), Firewalls and Access Control Lists.

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 [40].

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 [48] to block data between the sensors and the CAN network. In correct 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 [43]. Attack on TPSM is described in [49], where the authors were able to perform eavesdropping attacks from 40 meters on a passing car.

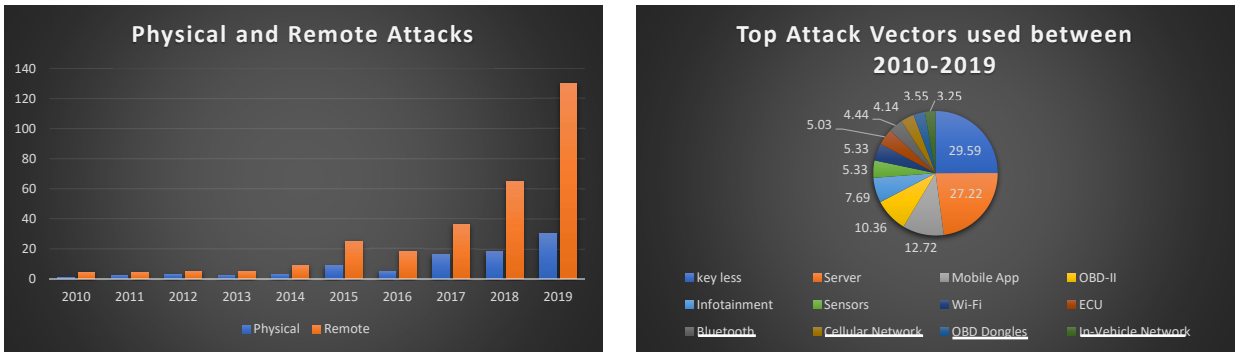
LiDAR and Camera: Cameras and laser signals are used inside vehicles to provide safety and driving assistance. These components can be manipulated by various attacks such as signal jamming. In [50], the authors performed signal jamming attacks on LiDAR and cameras. These components are also widely used inside autonomous vehicles.

Keyless entry : Miller and Valasek [43] show how Remote Control Door Lock Receiver (RCDL) 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 in many occasions, and it has been shown as the most used attack between 2010–2019 [51]. 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 [52] 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 [3]:

- 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.



(a) No. of physical & remote attacks: 2010–2019 [51]

(b) Top attack vectors: 2010–2019 [51]

Figure 9: Vehicle Attacks: 2010–2019

- There is no security mechanisms used for authentication and thus the CAN bus is vulnerable to message poisoning and denial of service (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) [53].

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 [51] as shown in figure 9. The vulnerable points could be classified as direct, indirect, short-range and long-range attacks [40].

The CarShark software was used within a vehicle to sniff, analyse, observe and replay the data on the CAN bus using OBD-2 connector and then control the wheels, brakes and other ECUs and components of the vehicle [38]. This work also reports on other entry points to the CAN bus inside the vehicle such as the audio jack, USB and Wi-Fi, and the use of these to perform various attacks. Similarly, other attacks were performed from outside the car identifying potential vulnerabilities [40]. Figure 9 shows potential entry points for attacking a vehicle.

Other examples include attacks on Toyota Prius 2010 [54] 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 [43]. Another attack is the keyless fob attack used to forcibly unlock the doors of a vehicle [52]. A summary of in-vehicle network based attacks includes: Table 4 identifies some of the attacks initiated

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 • BT • Wi-Fi	• Physical • Remote • Remote	• OBD-2 direct connection • BT vulnerabilities • WiFi on board access	• Inside/Outside • Outside • Outside	• 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

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.

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.

Table 4: Attacks on In Vehicle Networks

Authors	Initiated Attacks	Entry Points	Position of the Attackers	Attack Result	Test Environment
[38]	Interfaces Infotainment	• OBD-2 • USB • CD Player	• Inside (Direct) • Inside • Inside	• CAN bus injection • Full access	• Real vehicles
[43]	Interfaces	• OBD-2	• Inside • Outside	• Control brakes, • Wheels and • Get access to the CAN bus	• Real vehicles
[55]	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
[40]	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
[50]	Sensors	• LiDAR • Cameras	• Outside • Outside	• Signal jamming	• LiDAR Hardware • CAN software
[56]	Sensors	• TPSM	• Outside	• Inject with • False TPSM values • and signal jamming	• Real vehicles
[52]	Sensors	• Keyfob Keyless entry system	• Outside	• Lock,unlock door • and start the engine	• Real vehicles
[43]	Telematics	• WiFi	• Outside	• Unauthorised access • Inject CAN message • to stop the engine	• Jeep Cherokee
[44]	Telematics	• WiFi	• Outside	• Full access to CAN bus	• Tesla model S
[46]	Interfaces	• OBD-2 Cellular Dongle	• Outside	• CAN bus injection	• Real vehicle

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: **CAN bus sniffing**: With

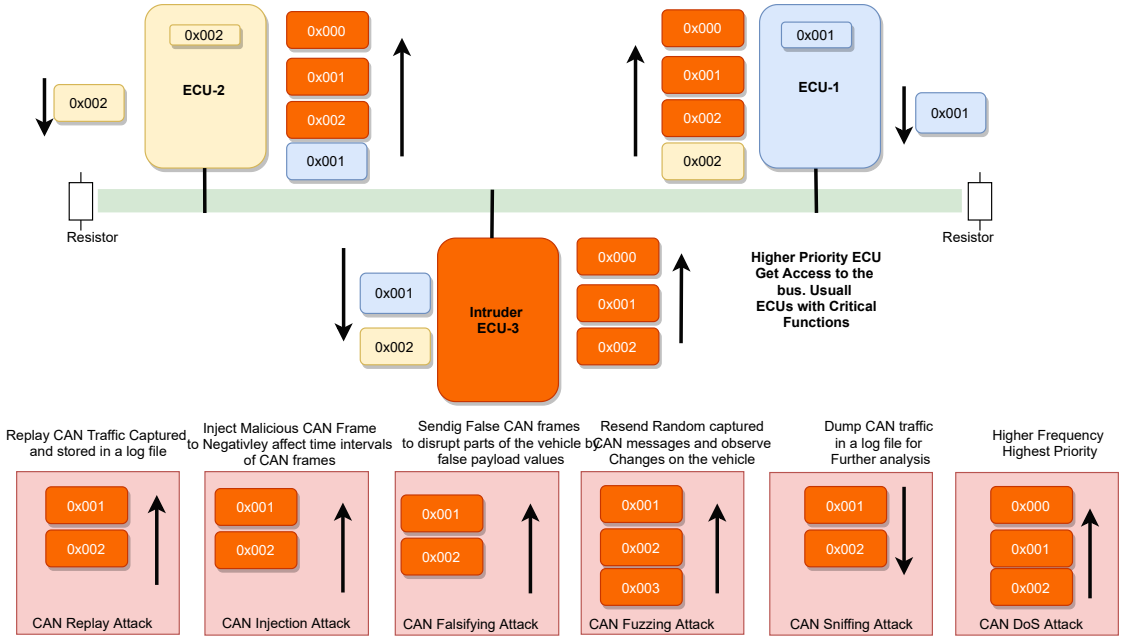


Figure 10: Overview of some In Vehicle CAN bus Network (IVN) attacks

no authentication mechanisms, encryption and broadcast transmission, it is possible to sniff the data on the CAN bus [3]. Using off the shelf OBD2 sniffer such as CANdo board, it is possible to read and analyse the data on the bus to manipulate and generate similar messages [43]. 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 [7].

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 [57]. 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. In order to prevent this attack, CAN bus should provide authentication to verify the source of the data before acting upon it. Usually 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 [58]. 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 and generate messages to simulate events [59]. 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 which 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 [5]. 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 [43]. 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, the attacker can increase the frequency by injecting the same message with higher frequency which 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 it's 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 which generates an attack. However, if the attack was more focused, they could initiate an attack to disable particular ECUs. For example, Iehira et al. [60] 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 which 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 Intrusion Detection Systems (IDS) focus on the detection of malicious messages. Firewall and Intrusion Prevention Systems (IPS) 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 Message Authentication Code (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 which would affect vehicle safety. CAN bus also provides a checksum calculation using Cyclic Redundancy Code (CRC) to check if there is a change in the frame during transmission, but it only provides error detection not the integrity and authentication of the frame. An ACK field is used for error detection and correction purposes. Implementing cryptography in the CAN bus should consider the following [61], [62], [63]:

- Limited frame size and capacity of the bus;
- limited speed of response and high latency;
- broadcast nature of the bus – and lack of support for confidentiality, integrity and authentication by design;
- no backward compatibility;
- limited computational capacity within ECUs.

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 8bytes, 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:

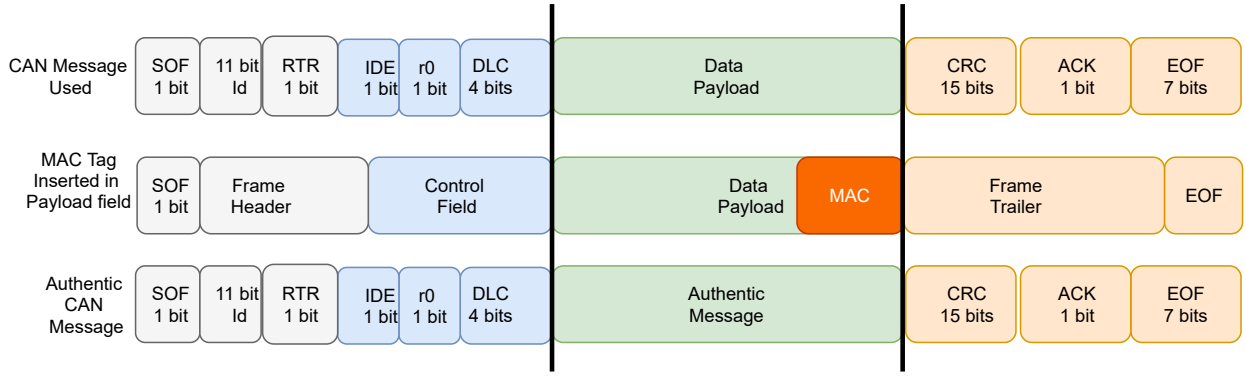


Figure 11: MAC signature inside CAN frame [64]

- Using lightweight Message Authentication Code (MAC) to overcome resource constraints in ECUs, and making use of a small key size and MAC signature.
- 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.

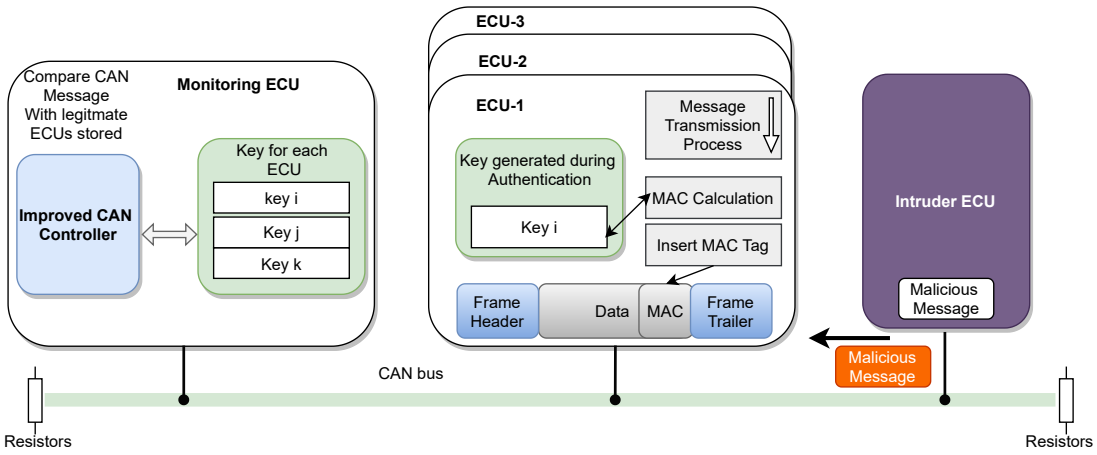


Figure 12: CAN message authentication and dedicated monitoring ECU. Derived from [64]

- A Hardware Security Accelerator can be used (as an additional hardware) to overcome computational resource limitations – however, it may not be a cost-effective approach.

According to the National Institute of Standard and Technology (NIST), HMAC [65] provides authentication and integrity of the data through a hash function and a shared secret key between sender and receiver. Hash algorithms such as SHA-1, SHA-224 and SHA-256 produce message digest or MacTag of 160, 224 and 256 bits according to [66]. The size of this tag is exceeds the maximum data payload of CAN frame (of 64 bits) and thus a truncated tag is used by deriving a smaller size from the MAC computation.

6.3 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 Message Authentication Code (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 either change the behaviour of the protocol by splitting CAN frames, replace fields, increase CAN frame size or increase bus payload and response time. Some approaches also require additional hardware which can increase the cost of implementation and lead to incompatibility with current vehicles.

-Nilsson et al [67] introduced an approach based on a shared 128-bit key between ECUs and a Cipher-Block Chaining Message Authentication Code (CBC-MAC) using KASUMI 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 in order to insert 16-bits inside the CRC field. This means that 4 messages are needed in order to send the 64-bit tag. As a result, their approach increases the bus load 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 [36] 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 approx. 50μs. They used Freescale's automotive boards 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.

-Another approach CANAuth used light weight encryption to mitigate sniffing and poisoning attacks [61]. 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 (16bits) was used which 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 [68] used a one way function to provide a 2byte magic number. The authors proposed using their approach in the data field (2 bytes out of 8 bytes) in the standard CAN frames 2.0A. In the extended CAN, they used magic number of 16 bit 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) which 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 [69]. 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 [70]. For improvement, they suggest using CAN+, but the CAN transceiver hardware should be changed to be able to handle the new CAN+ frame.

-In[71], 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 [72] 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 [63] 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 [73], the authors introduced a one-way hash chain using HMAC-MD5 and AES-128 to provide authentication. they tested their approach on simulated ECUs using CANoe Vector 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 assume 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 [64], the authors have used HMAC-SHA256 to provide ECU authentication and data integrity. Their MAC tag size is 1byte 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 constrains (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. [74] made use of an asymmetric key. HMAC is then used with changeable session keys. The authors assume that both public and private keys 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. The security of the algorithm was validated using the AVISPA software. 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.

- Groza and Murvay [75] 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 TriCore (32bit) microcontrollers. However, their approach is based on a delayed authentication approach which is difficult to support in real time for a CAN bus [76].

- In [70], the authors introduced “TOUCAN” which provides authentication, integrity and encryption for a CAN bus. They use AES 128bit and Chaskey hashing 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 40bits while the remaining 24bits are used for the hashing value. In their approach, (using AES 128 and Chaskey hashing) the overall execution times are approx. 12ms.

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
[67]	CBC-MAC	• Injection • spoofing	No	Delayed authentication	Multiple CAN frames	Splitting CAN frame for authentication purposes	Theoretical
[36]	Trusted Group HMAC Symmetric key	• Sniffing, • Spoofing • Injection	Pre-load keys During manufacturing	Yes	Message Splitting	Split CAN frames for authentication purposes	Freescale's automotive boards
[61]	HMAC Symmetric key Counters	• Sniffing, • Spoofing • Injection • Replay	No	Yes	16 bytes of data payload	CAN+ 16 Bytes All nodes must Know pre-shared key	Theoretical
[68]	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
[72]	HMAC Symmetric keys Counter	• Replay • Spoofing • Injection	ECU server	Yes	No	Special ECU server hardware	Altera FPGA board CAN transceiver board
[71]	SAH3 HMAC	• Replay • Injection	No	Yes	No	Replace CRC field	Theoretical
[69]	MD5 LMAC	• Replay • Injection	No	No	Yes	Split CAN messages Using CAN+	Theoretical
[63]	128-bit key MAC Counter	• Replay • Injection • Spoofing	No	No	No	16-bit counter in the CAN 2.0b frame header	FreescaleS12X and Infineon TriCore
[73]	HMAC MD5 AES-128	• Spoofing • Replay	No	Yes	No	Insert MAC tag in 18-bit filed in CAN 2.0B header	CANoeVector tool Freescale S12XF board
[64]	HMAC SHA 256	• Replay • Spoofing	Dedicated ECU-FPGA board with built-in HMAC	Yes	No	Insert 8-bit MAC tag 4-bit counter	Altera FPGA development board and CAN transceiver board

HMAC, Hash Message Authentication Code; **CBC-MAC**, Cipher Block Chaining Message Authentication Code; **LM-MAC**, Linearly Mixed MAC; **MD5**, Message Digest; **SHA3**, Secure Hash Algorithm3.

-In [77], 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 0.13ms.

-In [78], 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. Pair-wise 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 look up 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.

Table 6: Message Authentication for CAN frames

Authors	Method	Attacks mitigated	Hardware Security Module	Real time	Bus load	Change CAN bus	Test bed environment
[74]	• Asymmetric key • HMAC • Changeable keys	• Replay • Spoofing	No	Yes	Load during key exchange	Assume pre-installed keys	Analytical evaluation
[79]	• Symmetric key • HMAC	• Replay • Spoofing	No	Delayed authentication	No	Delayed authentication	S12 equipped with an XGATE coprocessor and Infineon TriCore
[70]	• AES-128 • Chasekey HMAC	• Spoofing • Replay	No	Yes	No	24-bit MAC tag and 40-bits for the actual data	STM32F407 CAN boards
[77]	• AES-128 • HMAC • Compression algorithm	• Replay • Injection	No	Yes	No	No	CANoe simulator
[78]	• MAC tables • Pairwise key • Symmetric key	• Replay • Spoofing	ECU server	No	Yes	No	No

6.4 CAN Frame Encryption

-In [80] 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 [70] .

-In [81] 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.

-CANTrack algorithm by Farag et al. [82] uses a dynamic symmetric key to encrypt the 8byte 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
[80]	• HMAC • SHA1 • AES • DES	• Sniffing • Replay • Spoofing	No	Yes	Yes	Split CAN Frames	Simulator
[81]	• 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
[82]	• Dynamic symmetric key • Key generator	• Spoofing • Replay • Sniffing	No	Yes	No	No	CANoe software

CAN FD was introduced to tackle the needs of higher speed and larger data payload size. The following approaches are focused on CAN FD.

- In [83], the authors introduced an architecture supporting key management, encryption and authentication for a CAN FD bus. They used symmetric key and Authenticated Key Exchange Protocol 2 (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. [84] introduced a secure CAN FD bus which 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 36bytes for the data payload and 28bytes for the cryptographic tag. They used CANoe software and the LPC54618 microcontroller to validate their approach.

-Groza et al. [79] 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. [85] used the lightweight Chaskey MAC algorithm over limited capacity computational resources such as a 32-bit microcontroller. They used this algorithm with a 128bit key and compared it with the HMAC-SHA1 algorithm. They found Chaskey has a lower latency comparing with HMAC-SHA1. They used 4bytes 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.

6.5 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 [86]. The location of the IDS is also a key decision: Host-IDS (HIDS) based and Network-IDS (NIDS) based [53]. HIDS to detect attacks [87] 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 [88].

Table 8: CAN FD encryption and authentication. Improved datapayload 64-bytes, allow more space for MAC signature along with Actual data

Authors	Method	Attacks mitigated	Hardware Security Module	Real time	Bus load	Change CAN bus	Test bed environment
[83]	• AES-128 • HMAC • SHA256	• Sniffing • Replay • Spoofing	No	Yes	No	No	Threetypes of CAN-FD boards and CANoe software
[84]	• Gateway ECU • Public and • Private keys	• Sniffing • Replay • spoofing	No	Yes	No	Gateway ECU needed	CANoeVector and LPC54618 micro-controller
[79]	• ECU Group • Sharing keys	• Spoofing • Replay • Sniffing	No	Yes	NO	Group based key sharing	neonTriCore controllers contrasted with low-end Freescale S12X cores
[85]	• 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

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. [90] 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**: It looks at the 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.
- **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.
- **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.

Below is a figure 13 illustrates the position of an IDS inside a CAN network. The IDS can use different features in data link layer CAN frames such as:

- **CAN identifier**: 11-bit or 29-bit value which 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.
- **DLC**: Data Length Code is a 4-bit field and 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**: It is 8 bytes maximum and it also has a fixed length and range which should not be exceeded. Anomalies can be detected if abnormal values and changes occur in the data field. The authorised identifier, the payload range and consistency, fixed DLC length and fixed rate are features that can be used to detect malicious and anomalous traffic.
- **Timestamp**: Each CAN frame has a timestamp which can be either hardware or software based. Through this timestamp, an IDS can monitor the time intervals of CAN messages and observe any unusual behaviour. This approach is based on the observation that that ECUs have fixed broadcast intervals and thus an anomaly can be detected.

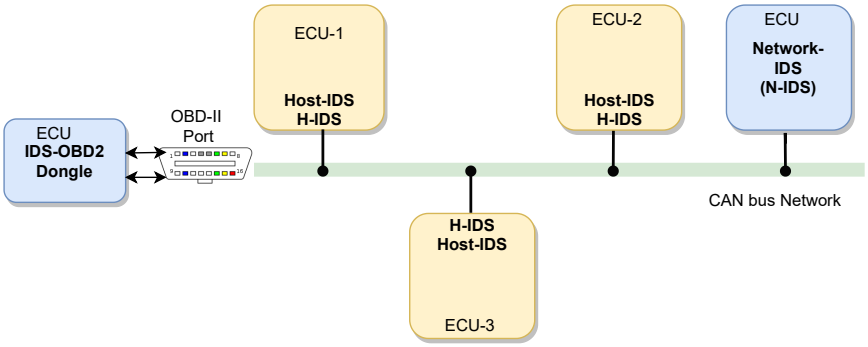


Figure 13: Positions of IDS inside automotive CAN network

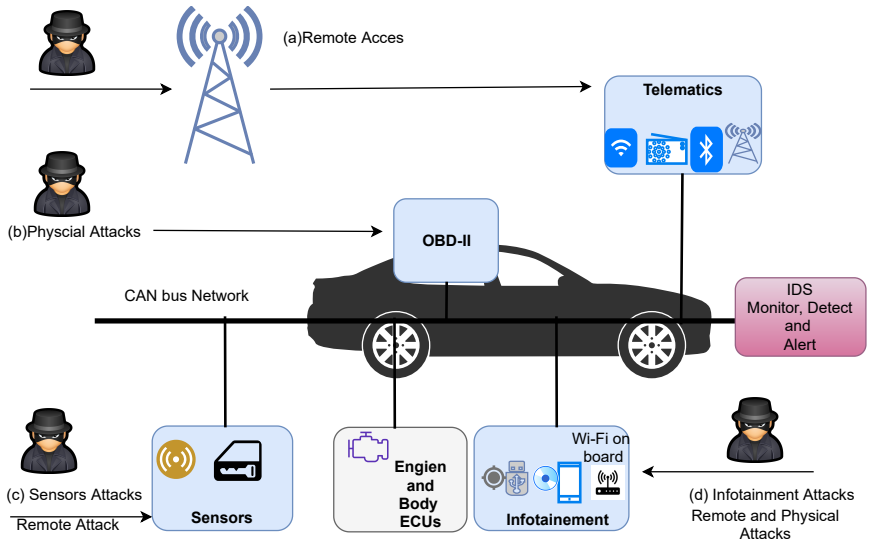


Figure 14: IDS as ECU inside a vehicle – based on [91]

6.6 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 [92]. Also, the signature-based IDS needs to maintain a potentially large database of known attacks on in-vehicle networks (including potential variants of these) [93]. Extracting attack signatures in real time for a moving can also be a challenge and suffer from high latency.

Studnia et al. [94] introduced a signature-based IDS which uses a list of signature derived from a CAN data set. However, this approach has limited benefit as the length of CAN bus words may not be known apriori. Furthermore, this approach may fail to detect an attack if it does not sense the first part of the data exchanged as malicious packets [95]. Larson et al. [87] introduced a Host IDS (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 [96], 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.

6.7 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.

Table 9: IDS using signatures & rules

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

6.8 IDS using statistical approaches

This IDS learns *normal* behaviour of the system based on conditional statistical relationship analysis – as outlined in figure 15. 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 [97]. 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, identification (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 [98]. Hence, this IDS can detect manipulated and incorrect payload values along with

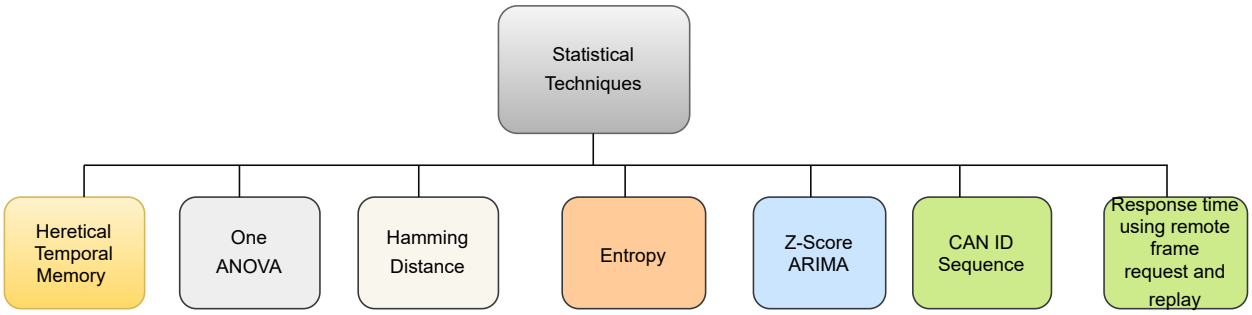


Figure 15: IDS based on Statistical Techniques

inconsistent use of CAN ID. The following statistical approaches have been reported in literature:

- number of packets exchanged within a particular time period;
- time interval between CAN frames;
- frequency of transmission using a particular CAN ID;
- throughput observed;
- response time using remote frame requests and message replay;
- Hamming distance to compare messages;
- analysis approach used, e.g. Entropy, Anova, Z-score, ARIMA (time wondow based moving average), Heretical Temporal Memory etc.

CAN ID Frequency: -Ling and Feng [99] 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 [100] 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. Similar 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 [101] 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 [102], 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 [103], 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 one second time window, with historical values. However, this anomaly detection over a time window cannot precisely detect small sized malicious messages.[93].

Remote Frame Request and Reply intervals: Lee et al [104] used remote frames to detect anomalies based on the request and response intervals between frames. Since each ECU replies to a remote frame which 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 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 [105], the authors consider the CAN id and the payload as features for their approach. They measured the entropy associated with variation in CAN traffic compared to a baseline of normal CAN traffic. They have tested their approach against frame injection attacks, and they found that their approach cannot detect a small number of injected CAN messages.

Entropy of CAN ID and data payload behaviour: In [106], the authors evaluated an entropy-based anomaly detection IDS for in-vehicle networks, and they found that dividing CAN messages into classes and feeding them to an entropy-based anomaly detection algorithm would lead to a more accurate detection (compared to considering one class). Their anomaly detection 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. Also, they calculated the entropy for each message id separately over a time window with a fixed number of messages. 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.

Entropy of CAN ID and data payload behaviour: Wu et al. [107] 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 [108], the authors used a one-way ANOVA function to statistically determine the pattern within a data set and created a set of *normal* patterns to detect anomalies. They grouped CAN data set using vehicle parameters such as: fuel usage, gear ratio, engine parameters, etc 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 [93] 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 [109], 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 [110], 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 [34], 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 [111], the authors used a distributed IDS based on Heretical Temporal Memory (HTM), a technique (similar to recurrent neural networks) widely used in time series forecasting and analysis.

Table 10: IDS based on Statistical methods

Authors	Type	Layer	CANID / Data Payload	Detection mechanism	Attacks Detected
CAN ID Frequency [99]	Statistics	Data Link (Controller layer)	CAN ID behaviour	Detect malicious CAN ID Detect Unusual CAN frequency	• Injection • DoS
Entropy based anomaly Detection [106]	Statistics	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
Detecting attacks based on identifying Packet timing Anomalies in Time Windows [34]	Statistics	Data link	CAN ID broadcast mean in defined time window	Detect attack based on specification rules	• Injections • DoS attacks
Detecting attacks through Hamming distance [93]	Statistics	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
Anomaly detection based on ID sequence [58]	Statistics	Data link	Sequence between CAN ID messages	Compare the sequence of CAN ID with the knowledge acquired from real time model	• Replay • injection
Entropy IDS based on CAN ID [112]	Statistics	Data link	Entropy of each CAN ID	Detect the changes on each bit of the CAN ID	• Flooding • injection
Time series algorithm. ARIMA and Z-Score [34]	Statistics	Data link	Broadcast intervals in time window	Check the change in broadcast intervals of CAN ID	• Drop • injection
offset ratio and remote frame IDS [104]	Statistics	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
One-Way ANOVA [108]	Statistics	Data link	Data payload consistency across multiple CAN signals	Compare the mean of related CAN frames according to the normal statistical observation eg. speed and engine	• Data payload • manipulation
Cumulative Sum algorithm in defined time window [110]	Statistics	Data link	CAN ID sequence	CAN ID sequence behaviour	• Injection • DoS attack • Frame Fuzz. Attack

6.9 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 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 needs 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 data sets, 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. [113] 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 for the parameters separately. They evaluated multiple attacks by injecting malicious speed and RPM messages together. Similarly, in [114] 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 which the authors in [114] 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 (SVM): In [115], 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

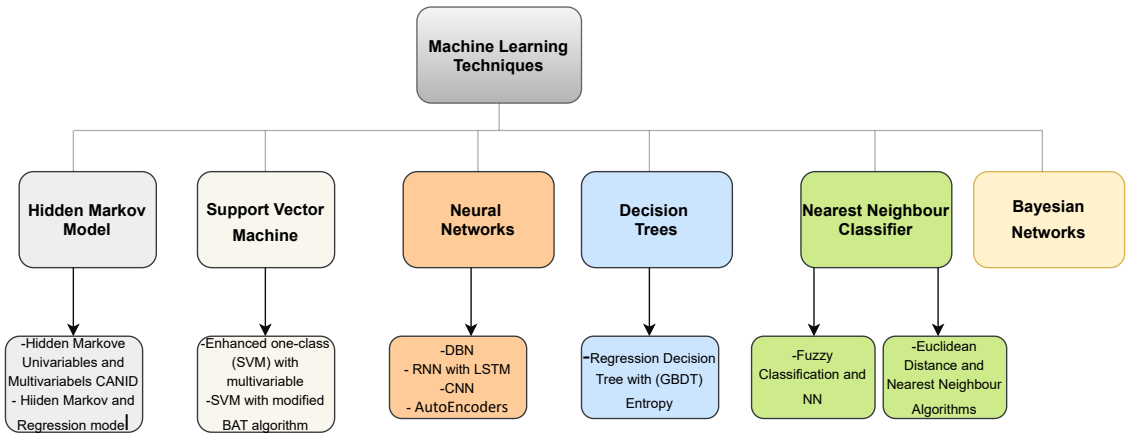


Figure 16: Machine learning based IDS

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 [95], 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): In [116] the authors used deep neural networks to learn normal patterns using unsupervised data sets, and to detect deviation from normal as anomalies. They have used an unsupervised Deep Believe Network (DBN) to pre-process the data and identify a normal pattern. To validate their approach, they inserted noise to their test data set to mimic real vehicle data. They simulated and generated CAN frames using a real world vehicle test bed and network experiments software [117].

In [35], 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 [118], the authors used Generative Advertorial Nets (GANs) to identify patterns of CAN data without classification. Their approach was able to detect anomalies based on the normal data provided. They tested their approach against Denial of Service (DoS), frame fuzzification and Spoofing attacks. Their work demonstrates that this technique is able to detect attacks with high accuracy.

In [91], the authors have used Convolutional Neural Networks (CNNs) 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 [119], the authors have used a Recurrent Neural Network (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 data set. They argue that an ideal IDS should be able to plug inside existing vehicles to detect anomalies without the need for either reverse engineering CAN traffic or contacting the vehicle manufacturer to get CAN messages specification.

In [120], the authors used a neural network model which consists of LSTM for CAN bus time series behaviour, auto-encoder to learn the normal behaviour of unlabelled data in unsupervised manner. Also, Exponential Linear Unit (ELU) is used for better classification as part of their neural network model. This approach benefit from the LSTM ability to learn from previous events of CAN bus traffic as it is it is suitable for time series data and the auto-encoder ability to extract normal behaviour from unsupervised datasets. 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 [121], the authors used unsupervised deep learning method known as Deep Contractive Autoencoders (DCAE). Furthermore, they have evaluated their approach against DoS, 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 [122], the authors also used 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.

In a different approach, authors in [48] have 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 [123], Sharma and Moller have introduced an architecture for using IDS based on neural networks to detect attacks for in-vehicle networks, alert a manufacturer, surrounded 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 [124], 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 [125], 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 (DT): Decision tree-based approaches classify CAN data into two classes (normal, anomalous). DT needs a supervised labelled data set during the training stage to be able to make decisions. In [126], 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 data set by inserting random abnormal values as anomalies.

Nearest Neighbour Classifier: In [127], the authors used fuzzy classification algorithms based on Nearest Neighbour classifiers (NNC) to discriminate attacks targeting the CAN bus. They used a data set available online which contains CAN attacks to validate their approach. They used the data payload, actual data (8bytes), as features to classify CAN traffic. They tested their approach on different attacks such as DoS, frame injection and frame fuzzification provided in the data set. 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 [128], 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 is a graphical model that uses probabilistic relations between related variables. IDS based on this approach can utilise a variety of features such as the ability of Bayesian networks, e.g.: (1) to predict the sequence (time evolution) of an event; (2) to integrate previous knowledge with probabilistic techniques, and (3) to handle missing data by encoding inter-dependencies between variables[129]

In [130], authors have 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 [131], 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 Hill-Climbing algorithm to construct a Direct Acyclic Graph (DAG) 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 [132], 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.

6.10 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 [134], the authors introduced a hardware-based Intrusion Response System (IRS). This is a signal and voltage based physical layer (transceiver layer) IDS which

Table 11: Machine Learning based IDS

Authors	Type	Detecting threshold	Detection mechanism	Attacks Detected
[113]	**Hidden Markov Model**	Univariante CAN signal Multivariant CAN signals e.g RPM and speed	Deviation from the sequence behaviour	• Single injection • Multiple injection
[114]	**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
[115]	**Enhaced SVM**	CAN ID and data payload Multivarinete CAN signals	Deviation from ESVM Enhanced one-class Support Vector Machine	• Error and • Signal faults
[133]	**O-SVM**	CAN message intervals and frequencies One Class support Vector based Anomaly IDS	Anomaly based on One SVM class detection	• Fuzzing
[95]	**O-SVM with modified BAT alogrithm**		One-class SVM algorithm	• Injection • DoS
[35]	**Recurrent Neural Network (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
[116]	**Deep Believe Neural Network with Probability feature**	CAN ID and data payload behaviour	Change from the NN model patteren	• Injection
[127]	**Fuzzy classification Nearest Neighbor Classification**	CAN ID and data payload	Checking each byte of the datapayload as features to detect anomalies	• DoS • Injection • Fuzzy
[128]	**Euclidean distance and nearest neighbor algorithms**	CAN ID frequency in time window	Change in CAN ID broadcast data payload	• Fuzzy
[126]	**Regression Decision Tree with Gradient Boosting (GBDT) Entropy**	CAN ID and data payload entropy change	Entropy change of CAN traffic	• Injection • DoS
[125]	**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
[111]	**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

Table 12: IDS based on Physical Characteristics

Authors	Type	Layer	Detecting threshold	Detection mechanism	Attacks Detected	Prevention
[134]	**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
[135]	**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
[136]	**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

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 [135], 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 (RLS) algorithm. Their approach uses a Cumulative Sum (CUSUM) to detect any significant deviation from the normal fingerprint baseline. In [136], the authors introduced a Voltage-IDS which 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 behavior-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 (ML) approaches can also handle multiple variable instances as vehicles generate large amounts of data which needs to

be pre-processed in order to be meaningful. This problem can be overcome using unsupervised ML which can classify patterns and detect anomalies in unlabeled raw data.

7 Limitations with Current Approaches

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 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.

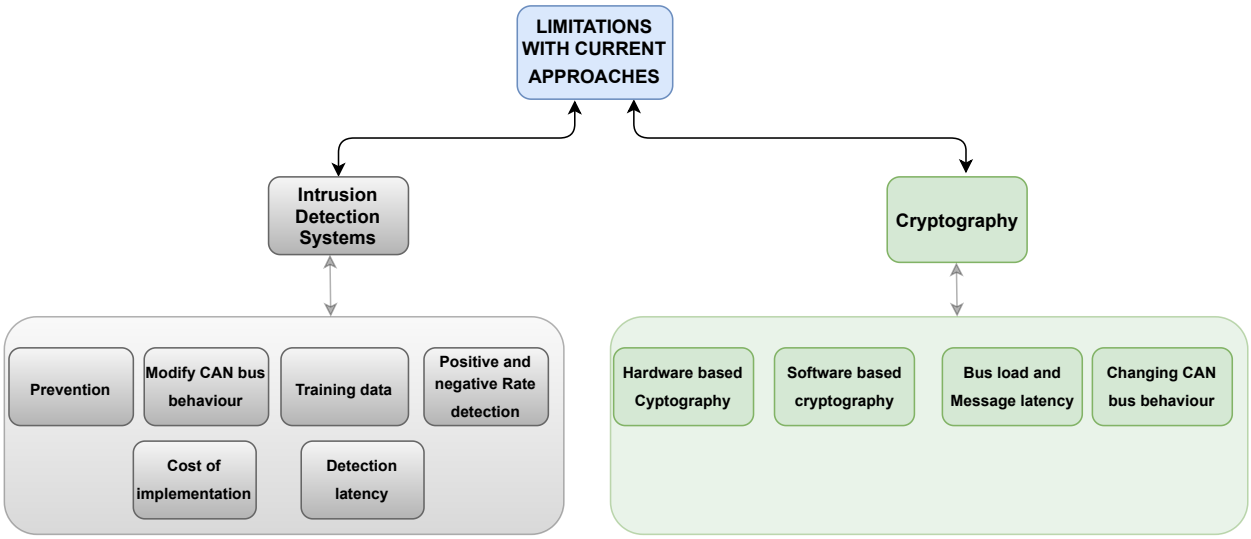


Figure 17: Research Challenges: Cryptography and IDS based approaches

7.1 Cryptography

Hardware based cryptography: hardware mechanisms can be used to speed up the process of generating cryptographic functions for in-vehicle networks. Depending on the number of ECUs (typically 70), each ECU would need to be updated to include hardware-based cryptography. 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 [137]. Also, better computational resources are expected in the next generation of CAN FD ECUs to handle the higher bit rate and data payload size needed.

Software based cryptography: the main approaches to provide security for a CAN bus is using encryption and authentication mechanisms without the need for additional hardware or modification to existing ECUs. Authentication approaches are less computationally expensive, as they add additional information within an existing data payload. However, with the large number of ECUs inside vehicles, current approaches have not validated their approach using this large number. In [138], the authors evaluated 10 CAN MAC approaches against industrial criteria and they indicated that some of these approaches are applicable to a subset of the network with a very limited number of 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 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 two folds, 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. Furthermore, 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. Other approaches extended CAN payload size to 16 byte CAN+ which also causes incompatibility, as ECU controller and transceiver needs to be changed in order to support this extended frame size.

7.2 Intrusion Detection System

Complexity: since there is no global attack signature database, an IDS needs to collect and analyse CAN network data in order to build an attacks signature database. Furthermore, it is dangerous to perform attacks on moving vehicle to extract attack signatures and maintain them.

Computational resources: very few approaches have validated their approach in a testbed that contains resources with representative computational capabilities to a vehicular network. ML based IDS may have high computational resource requirements, however ECU resources which exist inside vehicles may not be able to handle this workload. In [107] and [109] the authors suggest that dedicated hardware is needed to deploy such approaches. Furthermore, their idea is that a statistical based IDS can be a lighter approach that can be applicable in current vehicle networks. Similarly, in [139] authors have examined the use of IDS based on neural networks and they recommend that it is difficult to use them in current vehicle networks due to the large memory and computational time needed and they have suggested a dedicated hardware. In other approach [113], authors have inserted an IDS in OBD-2 port using raspberry pi board and they said that this approach can be used in current vehicles and embed in future vehicles. This can be a good approach to avoid the constrained power in current vehicles.

Modify CAN bus behaviour: IDS approaches work in a passive manner where they don't require to change the CAN bus protocol behaviour. They only monitor, detect malicious attacks and report them, for example, to the driver and fleet management centre.

Detection latency: In current vehicle networks, ECUs have low computational power, thereby limiting the potential to implement IDS based on deep learning. In [48], authors indicate that an IDS based on deep learning incurs a high latency due to increased processing time. They therefore located their deep learning model in the cloud and provide offline detection for vehicles from a central point. However, they acknowledged that offloading data from a vehicle to a cloud platform requires a stable network connection. Also, real time detection is needed for passenger safety, for which an IDS based on the cloud may not be able to offer. Alternatively, authors in [140] have suggested to put Edge E-IDS in OBD-2 port as a plug in dongle to detect attacks and process data at the network edge before utilising a cloud platform.

Cost of implementation: It is worth noting that an IDS can be installed in each ECU, as a Host-IDS, to detect attacks [87] – 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 [88].

Positive and Negative Rate detection: IDS should work and detect attacks at runtime with the intention to minimise false positives. According to [97], a false positive rate of 0.0001 percent can cause 5 false positives every 1 hour in a CAN network broadcasts 1500 frames per second. Therefore, an IDS should carefully verify their decisions. In [98], authors evaluated four types of IDS, information entropy, CAN ID sequence, message frequency and throughput based IDSs. They evaluated these IDSs based on the positive and negative detection rate. They tested them offline against four known attacks, packet dropping, spoofing, replay and flooding attacks. They found different negative and positive rates for each attack in each type of IDS they evaluated.

Training data: IDS need either a database of CAN attacks (signatures) to be able to detect malicious attacks or by analysing a CAN data set offline to extract normal behaviour. While there is no global signature database of attacks, a signature database should be built by analysing normal CAN traffic along with generating various CAN attacks. However, attacks on vehicles continue to emerge and signatures of all known attacks are difficult to be maintained and updated to detect new attacks.

Prevention: Some approaches focus on preventing attacks rather than passively detecting them. A combination of hardware and software-based IDS techniques are needed to be more effective to prevent attacks.

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 bit rate 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 to manage countermeasures e.g (1) monitoring and management of message authentication and encryption mechanisms, while in (2) IDS approaches, edge ECU devices can be used as a plug in edge device e.g inside OBD-2, telematics and infotainment interfaces to support edge 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. These edge ECUs devices can provide suitable resources to support countermeasures and can be used in current vehicles with limited CAN bus modifications. If ECU modification are needed, for example to support authentication and contact with edge ECU monitoring devices, OEMs and car makers should consider update ECU capabilities, e.g. over-the-air updates.

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 [141]. The next generation CAN FD is expected to provide better resources e.g. higher data bit rate, payload size and support for embedded encryption methods. As a result, vehicles based on CAN FD can overcome current ECU shortcomings. Other additional capabilities incude: (1) Better ECUs to handle higher data payload, (2) embedded cryptographic algorithms such as Advanced Encryption Standard (AES) and better data bit rate. Therefore, suppliers, car makers 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 Dedicated Short-Range Communications (DSRC) for Vehicle-2-Vehicle (V2V) and Vehicle-2-Infrastructure (V2I) should be protected, as they are entry points for data injection to the internal vehicle CAN bus network.

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 paper, 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 CAN bus protocol is used inside both electric and autonomous cars [142], 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] The Institution of Engineering and Technology. Serious cyber-security flaws uncovered in Ford and Volkswagen cars | E&T Magazine, 2020.

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

[3] Roderick Currie. Information Security Reading Room Developments in Car Hacking. 2015.

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

[5] Tianxiang Huang, Jianying Zhou, Yi Wang, and Anyu Cheng. 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)*, volume 10701 LNCS, pages 621–637. Springer, Cham, dec 2017.

[6] International Organization for Standardization. ISO 11898-1:2015 - Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signalling, 2015.

[7] Tae Un Kang, Hyun Min Song, Seonghoon Jeong, and Huy Kang Kim. Automated Reverse Engineering and Attack for CAN Using OBD-II. *IEEE Vehicular Technology Conference*, 2018-Augus:1–7, 2018.

[8] CiA. CAN in Automation (CiA): CAN XL is knocking at the door, 2020.

[9] Robert Bosch GmbH. CAN XL News text | Bosch Semiconductors, 2019.

[10] ISO. ISO 17987: Road vehicles — Local Interconnect Network (LIN) Part 1 : General information and use case definition. Technical report, 2016.

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

[12] National Instruments. FlexRay Automotive Communication Bus Overview, 2019.

[13] ISO 10681-1:2010. *Road vehicles – Communication on FlexRay – Part 1: General information and use case definition*. 2010.

[14] Omid Avatefipour and Hafiz Malik. State-of-the-Art Survey on In-Vehicle Network Communication (CAN-Bus) Security and Vulnerabilities. 2018.

[15] Wilrid Dubitzky and Turgut Karacay. CAN – From its early days to CAN FD. *CAN Newsletter*, pages 8–11, 2013.

[16] Sasan Jafarnejad, Lara Codeca, Walter Bronzi, Raphael Frank, and Thomas Engel. A car hacking experiment: When connectivity meets vulnerability. *2015 IEEE Globecom Workshops, GC Wkshps 2015 - Proceedings*, (May 2016), 2015.

[17] CAN in Automation. CAN in Automation, 2013.

[18] Robert Bosch Gmbh. Robert Bosch Gmbh: CAN Specification Version 2.0. 1991.

[19] KVASER. KVASER.

[20] Bomu Cheon and Jae Wook Jeon. The CAN FD network performance analysis using the CANoe. In *2013 44th International Symposium on Robotics, ISR 2013*, 2013.

[21] CAN in Automation. CAN in Automation (CiA): CAN FD - The basic idea. *www.can-cia.org*.

[22] International Organization for Standardization. Road vehicles - Safety of the intended functionality, 2019.

[23] Standard of Automotive Engineering. J3061A (WIP) Cybersecurity Guidebook for Cyber-Physical Vehicle Systems - SAE International, 2016.

[24] SAE International. Requirements for Hardware-Protected Security for Ground Vehicle Applications - J3101. *SAE*, 2012.

[25] Standard of Automotive Engineering. J3138: Diagnostic Link Connector Security - SAE International, 2018.

[26] Angela Barber. Status of Work in Process on ISO/SAE 21434 Automotive Cybersecurity Standard. pages 1–25, 2018.

[27] Hervé Seudié. Vehicular On-board Security : EVITA Project Project. In *Forum American Bar Association*, number November, 2009.

[28] SeVeCom (Secure Vehicular Communication). Sevecom, 2008.

[29] CVIS Cooperative Vehicle-Infrastructure Systems. CVIS Cooperative Vehicle-Infrastructure Systems. Technical report, 2010.

[30] Society of Automotive Engineers. SAE J1939 Standards Collection, 2007.

[31] VECTOR. SAE J1939 Know-how | Vector.

[32] Microchip. Mcp2515 Notes. page 94, 2003.

[33] David Wampler, Huirong Fu, and Ye Zhu. Security threats and countermeasures for intra-vehicle networks. *5th International Conference on Information Assurance and Security, IAS 2009*, 2:153–157, 2009.

[34] Andrew Tomlinson, Jeremy Bryans, Siraj Ahmed Shaikh, and Harsha Kumara Kalutarage. Detection of Automotive CAN Cyber-Attacks by Identifying Packet Timing Anomalies in Time Windows. *Proceedings - 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN-W 2018*, pages 231–238, 2018.

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

[36] Qiyan Wang and Sanjay Sawhney. VeCure: A practical security framework to protect the CAN bus of vehicles. In *2014 International Conference on the Internet of Things, IOT 2014*, pages 13–18. IEEE, oct 2014.

[37] Anon. National Instruments. *Scientific Computing and Instrumentation*, 17(1):48, 1999.

[38] 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. Experimental security analysis of a modern automobile. *Proceedings - IEEE Symposium on Security and Privacy*, pages 447–462, 2010.

[39] Juniper Research. In-Vehicle Commerce Connected Cars to Exceed 775 Million by 2023, 2018.

[40] Stephen Checkoway and D McCoy. Comprehensive experimental analyses of automotive attack surfaces. *Proceedings of the 20th USENIX conference on Security*, page 6, 2011.

[41] Madeline Cheah, Jeremy Bryans, Daniel S. Fowler, and Siraj Ahmed Shaikh. Threat Intelligence for Bluetooth-Enabled Systems with Automotive Applications: An Empirical Study. *Proceedings - 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN-W 2017*, pages 36–43, 2017.

[42] Dennis Kengo Oka, Takahiro Furue, Lennart Langenhop, and Tomohiro Nishimura. Survey of vehicle IoT bluetooth devices. In *Proceedings - IEEE 7th International Conference on Service-Oriented Computing and Applications, SOCA 2014*, pages 260–264. Institute of Electrical and Electronics Engineers Inc., dec 2014.

[43] Charlie Miller and Chris Valasek. Remote Exploitation of an Unaltered Passenger Vehicle. *Defcon 23*, 2015:1–91, 2015.

[44] Keen Security Lab. FREE-FALL : TESLA HACKING 2016 Who we are && What we did. 2016.

[45] Daniel S. Fowler, Jeremy Bryans, Siraj Ahmed Shaikh, and Paul Wooderson. Fuzz Testing for Automotive Cyber-Security. *Proceedings - 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN-W 2018*, pages 239–246, 2018.

[46] Zeljka Zorz. Backdooring connected cars for covert remote control - Help Net Security, 2018.

[47] Pradeep Sharma Oruganti, Matt Appel, and Qadeer Ahmed. Hardware-in-loop based Automotive Embedded Systems Cybersecurity Evaluation Testbed. *AutoSec 2019 - Proceedings of the ACM Workshop on Automotive Cybersecurity, co-located with CODASPY 2019*, pages 41–44, 2019.

[48] George Loukas, Tuan Vuong, Ryan Heartfield, Georgia Sakellari, Yongpil Yoon, and Diane Gan. Cloud-Based Cyber-Physical Intrusion Detection for Vehicles Using Deep Learning. *IEEE Access*, 6:3491–3508, 2017.

[49] Sangho Ohb Ishtiaq Roufa, Rob Millerb, Hossen Mustafaa, Travis Taylora and Ivan Seskarb Wenyuan Xua, Marco Gruteserb, Wade Trappeb. Security and privacy vulnerabilities of in-car wireless networks. 2012.

[50] Jonathan; Petit, Bas Stottelaar, Michael Feiri, and Frank Kargl. Remote Attacks on Automated Vehicles Sensors: Experiments on Camera and LiDAR. *Blackhat.com*, pages 1–13, 2015.

[51] Upstream Security and Global Automotive. UPSTREAM SECURITY's Global Automotive Cybersecurity Report. Technical report, 2019.

[52] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh, and Mohammad T.Manzuri Shalmani. 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)*, volume 5157 LNCS, pages 203–220, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[53] Wufei Wu, Renfa Li, Guoqi Xie, Jiyao An, Yang Bai, Jia Zhou, and Keqin Li. A Survey of Intrusion Detection for In-Vehicle Networks. *IEEE Transactions on Intelligent Transportation Systems*, PP(1):1–15, 2019.

[54] Charlie Miller and Chris Valasek. A Survey of Remote Automotive Attack Surfaces. *Technical White Paper*, pages 1–90, 2014.

[55] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. Security threats to automotive CAN networksPractical examples and selected short-term countermeasures. *Reliability Engineering and System Safety*, 96(1):11–25, 2011.

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

[57] Jihas Khan. Vehicle network security testing. *Proceedings of 2017 3rd IEEE International Conference on Sensing, Signal Processing and Security, ICSSS 2017*, pages 119–123, 2017.

[58] Mirco Marchetti and Dario Stabili. Anomaly detection of CAN bus messages through analysis of ID sequences. *IEEE Intelligent Vehicles Symposium, Proceedings*, (Iv):1577–1583, 2017.

[59] Chris Culling. Information Security Reading Room Which YARA Rules Rule : Basic or Advanced ? Th e In st itu te , A ho r R et ai ns ll Ri gh ts. (Security 503), 2017.

[60] Kazuki Iehira, Hiroyuki Inoue, and Kenji Ishida. Spoofing attack using bus-off attacks against a specific ECU of the CAN bus. *CCNC 2018 - 2018 15th IEEE Annual Consumer Communications and Networking Conference*, 2018-Janua:1–4, 2018.

[61] Anthony Van Herrewege, Dave Singelee, and Ingrid Verbauwhede. CANAuth - A Simple, Backward Compatible Broadcast Authentication Protocol for CAN bus. In *ECRYPT Workshop on Lightweight Cryptography*, number November 2011, pages 299–235, 2011.

[62] Stefan Nürnberger and Christian Rossow. vatiCAN: Vetted, authenticated CAN bus. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9813 LNCS, pages 106–124. Springer Verlag, 2016.

[63] Andreea Ina Radu and Flavio D Garcia. 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)*, volume 9879 LNCS, pages 283–300, 2016.

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

[65] NIST. The Keyed-Hash Message Authentication Code. *Federal Information Processing Standard Publication*, 198(July):1–20, 2008.

[66] Quynh Dang, Rebecca M Blank, and Patrick Gallagher. NIST Special Publication 800-107 Revision 1 Recommendation for Applications Using Approved Hash Algorithms. Technical report, 2012.

[67] Dennis K. Nilsson, Ulf E. Larson, and Erland Jonsson. Efficient in-vehicle delayed data authentication based on compound message authentication codes. *IEEE Vehicular Technology Conference*, pages 1–5, 2008.

[68] Ahmed Hazem and Hossam A H Fahmy. LCAP - A Lightweight CAN Authentication Protocol for Securing In-Vehicle Networks. *10th escar Europe 2012*, 2012.

[69] Bogdan Groza, Stefan Murvay, Anthony Van Herrewege, and Ingrid Verbauwhede. 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)*, volume 7712 LNCS, pages 185–200, 2012.

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

[71] Sebastian Bittl. Attack potential and efficient security enhancement of automotive bus networks using short MACs with rapid key change. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8435 LNCS:113–125, 2014.

[72] Ryo Kurachi, Yutaka Matsubara, Hiroaki Takada, Naoki Adachi, Yukihiro Miyashita, and Satoshi Horihata. CaCAN - Centralized Authentication System in CAN. *12th Embedded Security in Cars Europe*, (November 2014), 2014.

[73] Ki Dong Kang, Youngmi Baek, Seonghun Lee, and Sang Hyuk Son. An Attack-Resilient Source Authentication Protocol in Controller Area Network. *Proceedings - 2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS 2017*, pages 109–118, 2017.

[74] Samir Fassak, Younes El Hajjaji El Idrissi, Noureddine Zahid, and Mohamed Jedra. A secure protocol for session keys establishment between ECUs in the CAN bus. *Proceedings - 2017 International Conference on Wireless Networks and Mobile Communications, WINCOM 2017*, (June 2018), 2017.

[75] Bogdan Groza and Stefan Murvay. Efficient protocols for secure broadcast in controller area networks. *IEEE Transactions on Industrial Informatics*, 9(4):2034–2042, 2013.

[76] Samuel Woo, Hyo Jin Jo, and Dong Hoon Lee. A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):993–1006, 2015.

[77] Yujing Wu, Yeon Jin Kim, Zheyan Piao, Jin Gyun Chung, and Yong En Kim. Security protocol for controller area network using ECANDC compression algorithm. *ICSPCC 2016 - IEEE International Conference on Signal Processing, Communications and Computing, Conference Proceedings*, pages 1–4, 2016.

[78] Chung Wei Lin and Alberto Sangiovanni-Vincentelli. Cyber-security for the Controller Area Network (CAN) communication protocol. *Proceedings of the 2012 ASE International Conference on Cyber Security, CyberSecurity 2012*, (SocialInformatics):1–7, 2012.

[79] Bogdan Groza, Stefan Murvay, Anthony Van Herrewege, and Ingrid Verbauwhede. LiBrA-CAN: Lightweight broadcast authentication for controller area networks. *ACM Transactions on Embedded Computing Systems*, 16(3), 2017.

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

[81] AS Siddiqui, Y Gui, J Plusquellic 2017 IEEE 60th ..., and Undefined 2017. Secure communication over CANBus. *ieeexplore.ieee.org*, 2017.

[82] Wael A. Farag. CANTrack: Enhancing automotive CAN bus security using intuitive encryption algorithms. *2017 7th International Conference on Modeling, Simulation, and Applied Optimization, ICMSAO 2017*, pages 1–5, 2017.

[83] Samuel Woo, Hyo Jin Jo, In Seok Kim, and Dong Hoon Lee. A practical security architecture for in-vehicle CAN-FD. *IEEE Transactions on Intelligent Transportation Systems*, 17(8):2248–2261, 2016.

[84] Megha Agrawal, Tianxiang Huang, Jianying Zhou, and Donghoon Chang. 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)*, volume 11552 LNCS, pages 77–93. Springer Verlag, 2019.

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

[86] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. Applying Intrusion Detection to Automotive IT – Early Insights and Remaining Challenges. *Journal of Information Assurance and Security*, 4(January 2009):226–235, 2009.

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

[88] Clinton Young, Habeeb Olufowobi, Gedare Bloom, and Joseph Zambreno. Automotive Intrusion Detection Based on Constant CAN Message Frequencies Across Vehicle Driving Modes. 2019.

[89] Charlie Miller and Chris Valasek. OG Dynamite Edition. 2016.

[90] Michael Müter, André Groll, and Felix C. Freiling. A structured approach to anomaly detection for in-vehicle networks. *2010 6th International Conference on Information Assurance and Security, IAS 2010*, pages 92–98, 2010.

[91] Hyun Min Song, Jiyoung Woo, and Huy Kang Kim. In-vehicle network intrusion detection using deep convolutional neural network. *Vehicular Communications*, 21:100198, 2020.

[92] A S Syed Navaz, V. Sangeetha, C. Prabhadevi, A. S.SyedNavaz, V. Sangeetha, and C. Prabhadevi. Entropy based Anomaly Detection System to Prevent DDoS Attacks in Cloud. *International Journal of Computer Applications*, 62(15):42–47, 2013.

[93] Dario Stabili, Mirco Marchetti, and Michele Colajanni. Detecting attacks to internal vehicle networks through Hamming distance. *2017 AEIT International Annual Conference: Infrastructures for Energy and ICT: Opportunities for Fostering Innovation, AEIT 2017*, 2017-Janua:1–6, 2017.

[94] Ivan Studnia, Eric Alata, Vincent Nicomette, Mohamed Kaâniche, and Youssef Laarouchi. A language-based intrusion detection approach for automotive embedded networks. *International Journal of Embedded Systems*, 10(1):1–12, 2018.

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

[96] Tsvika Dagan and Avishai Wool. Parrot, a software-only anti-spoofing defense system for the CAN bus. *14th Embedded Security in Cars (escar)*, page 10, 2016.

[97] Andrew Tomlinson, Jeremy Bryans, and Siraj Ahmed Shaikh. Towards Viable Intrusion Detection Methods For The Automotive Controller Area Network. *Cscs*, 2018.

[98] Haojie Ji, Yunpeng Wang, Hongmao Qin, Yongjian Wang, and Honggang Li. Comparative performance evaluation of intrusion detection methods for In-Vehicle networks. *IEEE Access*, 6:37523–37532, jun 2018.

[99] Congli Ling and Dongqin Feng. An algorithm for detection of malicious messages on CAN buses. *Proceedings of the 2012 National Conference on Information Technology and Computer Science, CITCS 2012*, 2012.

[100] Kyong-Tak Cho and Kang G Shin. Fingerprinting Electronic Control Units for Vehicle Intrusion Detection. pages 911–927, 2016.

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

[102] Mabrouka Gmiden, Mohamed Hedi Gmiden, and Hafedh Trabelsi. An intrusion detection method for securing in-vehicle CAN bus. *2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering, STA 2016 - Proceedings*, pages 176–180, 2017.

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

[104] Hyunsung Lee, Seong Hoon Jeong, and Huy Kang Kim. OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame. *Proceedings - 2017 15th Annual Conference on Privacy, Security and Trust, PST 2017*, pages 57–66, 2018.

[105] Michael Müter and Naim Asaj. Entropy-based anomaly detection for in-vehicle networks. *IEEE Intelligent Vehicles Symposium, Proceedings*, (Iv):1110–1115, 2011.

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

[107] Wufei Wu, Yizhi Huang, Ryo Kurachi, Gang Zeng, Guoqi Xie, Renfa Li, and Keqin Li. Sliding Window Optimized Information Entropy Analysis Method for Intrusion Detection on In-Vehicle Networks. *IEEE Access*, 6:45233–45245, 2018.

[108] Ching Hsien Hsu, Feng Xia, Xingang Liu, and Shangguang Wang. Internet of Vehicles - Safe and Intelligent Mobility: Second International Conference, IOV 2015 Chengdu, China, December 19-21, 2015 Proceedings. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9502:89–97, 2015.

[109] Takuma Koyama, Toshiki Shibahara, Keita Hasegawa, Yasushi Okano, Masashi Tanaka, and Yoshihito Oshima. Anomaly Detection for Mixed Transmission CAN Messages Using Quantized Intervals and Absolute Difference of Payloads. *AutoSec 2019 - Proceedings of the ACM Workshop on Automotive Cybersecurity, co-located with CODASPY 2019*, pages 19–24, 2019.

[110] Habeeb Olufowobi, Uchenna Ezeobi, Eric Muhati, Gaylon Robinson, Clinton Young, Joseph Zambreno, and Gedare Bloom. Anomaly Detection Approach Using Adaptive Cumulative Sum Algorithm for Controller Area Network. In *AutoSec 2019 - Proceedings of the ACM Workshop on Automotive Cybersecurity, co-located with CODASPY 2019*, pages 25–30. ACM, 2019.

[111] Chundong Wang, Zhentang Zhao, Liangyi Gong, Likun Zhu, Zheli Liu, and Xiaochun Cheng. A Distributed Anomaly Detection System for In-Vehicle Network Using HTM. *IEEE Access*, 6:9091–9098, 2018.

[112] Qian Wang, Zhaojun Lu, and Gang Qu. An Entropy Analysis Based Intrusion Detection System for Controller Area Network in Vehicles. *International System on Chip Conference*, 2018-Septe:174–179, 2019.

[113] Sandeep Nair Narayanan, Sudip Mittal, and Anupam Joshi. OBD_SecureAlert: An Anomaly Detection System for Vehicles. In *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–6. IEEE, may 2016.

[114] Matan Levi, Yair Allouche, and Aryeh Kontorovich. Advanced Analytics for Connected Car Cybersecurity. *IEEE Vehicular Technology Conference*, 2018-June:1–7, 2018.

[115] Andreas Theissler. Anomaly detection in recordings from in-vehicle networks. *Big Data Applications and Principles*, (September 2014):12, 2014.

[116] Min Joo Kang and Je Won Kang. Intrusion detection system using deep neural network for in-vehicle network security. *PLoS ONE*, 11(6):1–17, 2016.

[117] Parnian Najafi Borazjani, Christopher E Everett, and Damon Mccoy. OCTANE: An Extensible Open Source Car Security Testbed. *Embedded Security In Cars (ESCAR)*, pages 1–10, 2014.

[118] Eunbi Seo, Hyun Min Song, and Huy Kang Kim. GIDS: GAN based Intrusion Detection System for In-Vehicle Network. *2018 16th Annual Conference on Privacy, Security and Trust, PST 2018*, 2018.

[119] Krzysztof Pawelec, Robert A Bridges, and Frank L Combs. Towards a CAN IDS Based on a Neural Network Data Field Predictor. In *AutoSec 2019 - Proceedings of the ACM Workshop on Automotive Cybersecurity, co-located with CODASPY 2019*, pages 31–34, 2019.

[120] Markus Hanselmann, Thilo Strauss, Katharina Dormann, and Holger Ulmer. CANet: An Unsupervised Intrusion Detection System for High Dimensional CAN Bus Data. *IEEE Access*, pages 1–1, 2020.

[121] Siti Farhana Lokman, Abu Talib Othman, Shahrulniza Musa, and Muhamad Husaini Abu Bakar. Deep Contractive Autoencoder-Based Anomaly Detection for In-Vehicle Controller Area Network (CAN). In *Advanced Structured Materials*, volume 119, pages 195–205. Springer Verlag, 2019.

[122] Siti Farhana Lokman. Stacked Sparse Autoencoders-Based Outlier Discovery for In-Vehicle Stacked Sparse Autoencoders-Based Outlier Discovery for In-Vehicle Controller Area Network (CAN). *Article in International Journal of Engineering and Technology*, 7(August):375–380, 2019.

[123] Priyanka Sharma and Dietmar P.F. Moller. Protecting ECUs and Vehicles Internal Networks. *IEEE International Conference on Electro Information Technology*, 2018-May(1):465–470, 2018.

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

[125] 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. CAN Communication To cite this version : HAL Id : hal-01716805 Embedded Hybrid Anomaly Detection for Automotive CAN Communication. *Embedded Real Time Software and Systems ERTS2*, 2018.

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

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

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

[129] Animesh Patcha and Jung Min Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 51(12):3448–3470, aug 2007.

[130] Haider Al-Khateeb, Gregory Epiphaniou, Adam Reviczky, Petros Karadimas, and Hadi Heidari. Proactive Threat Detection for Connected Cars Using Recursive Bayesian Estimation. *IEEE Sensors Journal*, 18(12):4822–4831, jun 2018.

[131] Anatolij Bezemskij, George Loukas, Diane Gan, and Richard J. Anthony. Detecting Cyber-Physical Threats in an Autonomous Robotic Vehicle Using Bayesian Networks. In *Proceedings - 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 2017*, volume 2018-Janua, pages 98–103. Institute of Electrical and Electronics Engineers Inc., jan 2018.

[132] Mario Casillo, Simone Coppola, Massimo De Santo, Francesco Pascale, and Emanuele Santonicola. Embedded Intrusion Detection System for Detecting Attacks over CAN-BUS. In *2019 4th International Conference on System Reliability and Safety, ICSRS 2019*, pages 136–141. Institute of Electrical and Electronics Engineers Inc., nov 2019.

[133] Valliappa Chockalingam, Ian Larson, Daniel Lin, and Spencer Nofzinger. Detecting Attacks on the CAN Protocol With Machine Learning. 2017.

[134] Sang Uk Sagong, Radha Poovendran, and Linda Bushnell. Mitigating Vulnerabilities of Voltage-based Intrusion Detection Systems in Controller Area Networks. *arXiv preprint arXiv:1907.10783*, 2019.

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

[136] Wonsuk Choi, Kyungho Joo, Hyo Jin Jo, Moon Chan Park, and Dong Hoon Lee. VoltageIDS: Low-level communication characteristics for automotive intrusion detection system. *IEEE Transactions on Information Forensics and Security*, 13(8):2114–2129, 2018.

[137] Olaf Pfeiffer and Christian Keyde. System design Security expectations vs . limitations PC / CAN Interfaces. pages 22–25, 2018.

[138] Nasser Nowdehi, Aljoscha Lautenbach, and Tomas Olovsson. In-vehicle CAN message authentication: An evaluation based on industrial criteria. *IEEE Vehicular Technology Conference*, 2017-Septe:1–7, 2018.

[139] Camil Jichici, Bogdan Groza, and Pal Stefan Murvay. Examining the use of neural networks for intrusion detection in controller area networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11359 LNCS, pages 109–125. Springer Verlag, 2019.

[140] Fei Guo, Zichang Wang, Suguo Du, Huaxin Li, Haojin Zhu, Qingqi Pei, Zhenfu Cao, and Jianhong Zhao. Detecting Vehicle Anomaly in the Edge via Sensor Consistency and Frequency Characteristic. In *IEEE Transactions on Vehicular Technology*, volume 68, pages 5618–5628. Institute of Electrical and Electronics Engineers Inc., jun 2019.

[141] Reiner Zitzmann. CiA CANopenFD Integration Workshop. Technical report, CAN in Automation, 2020.

[142] Mike Horton. What can a CANbus IMU do to make an autonomous vehicle safer? | Autonomous Vehicle International, 2019.