CRANFIELD UNIVERSITY

Mehmet Bozdal

A Wavelet-based Intrusion Detection System for Controller Area Network (CAN)

SCHOOL OF AEROSPACE, TRANSPORT AND MANUFACTURING
Integrated Vehicle Health Management

DOCTOR OF PHILOSOPHY
Academic Year: 2017 - 2021

Supervisor: Prof. Ian K Jennions
Associate Supervisor: Dr. Mohammad Samie
May 2021

CRANFIELD UNIVERSITY

SCHOOL OF AEROSPACE, TRANSPORT AND MANUFACTURING
Integrated Vehicle Health Management

DOCTOR OF PHILOSOPHY

Academic Year 2017 - 2021

Mehmet Bozdal

A Wavelet-based Intrusion Detection System for Controller Area Network (CAN)

Supervisor: Prof. Ian K Jennions
Associate Supervisor: Dr. Mohammad Samie
May 2021

It is okay not to be okay…

ABSTRACT

Controller Area Network (CAN), designed in the early 1980s, is the most widely used in-vehicle communication protocol. The CAN protocol has various features to provide highly reliable communication between the nodes. Some of these features are the arbitration process to provide fixed priority scheduling, error confinement mechanism to eliminate faulty nodes, and message form check along with cyclic redundancy checksum to identify transmission faults. It also has differential voltage architecture on twisted two-wire, eliminating electrical and magnetic noise. Although these features make the CAN a perfect solution for the real-time cyber-physical structure of vehicles, the protocol lacks basic security measures like encryption and authentication; therefore, vehicles are vulnerable to cyber-attacks. Due to increased automation and connectivity, the attack surface rises over time. This research aims to detect CAN bus attacks by proposing WINDS, a wavelet-based intrusion detection system. The WINDS analyses the network traffic behaviour by binary classification in the time-scale domain to identify potential attack instances anomalies. As there is no standard testing methodology, a part of this research constitutes a comprehensive testing framework and generation of benchmarking dataset. Finally, WINDS is tested according to the framework and its competitiveness with state-of-the-art solutions is presented.

Keywords:

CAN bus; In-vehicle communication; Automotive security; Automotive attack surface; Encryption

ACKNOWLEDGEMENTS

Foremost, I would like to thank my sponsor, the Republic of Turkey Ministry of National Education, for the financial support. I do not think it was possible to have this opportunity without their support.

I want to thank my supervisors, Prof. Ian Jennions and Dr Mohammad Samie for their guidance and support. They help me in every aspect of my research and academic development.

I also want to thank my lovely friends Cordelia and Sohaib, my office mates Iftikhar and Maulana, my former colleagues Kadir and Emrah, and other Cranfield mates from all over the world. Their support and faith in me brighten my dark mood in Cranfield.

Finally, I want to dedicate this work to my parents and express my gratitude to them. I am aware that they have sacrificed so many things for me. I also thank my girlfriend Serpil for her support to help me survive during thesis writing and the pandemic.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

ACK	Acknowledgement
CA	Collision Avoidance
CAN	Controller Area Network
CAN FD	CAN Flexible Data-rate
CMAC	Cipher-based Message Authentication Code
CRC	Cyclic Redundancy Checksum
CSE	Cryptographic Service Engine
CSMA	Carrier Sense Multiple Access
CSV	Comma-Separated Values
CTS	Clear to Send
DLC	Data Length Code
DNN	Deep Neural Network
DOI	Digital Object Identifier
DoS	Denial-of-Service
ECM	Error Confinement Mechanism
ECU	Electronic Control Unit
EOF	End of Frame
FM	Frequency Modulation
FPGA	Field-Programmable Gate Array
GAN	Generative Adversarial Nets
ID	Identifier
IDE	Identifier Extension Bit
IDS	Intrusion Detection System
IFS	Interframe Space
IT	Information Technology
LIN	Local Interconnect Network
MAC	Message Authentication Code
MOST	Media Oriented Systems Transport
OBD	On-board Diagnostic
OTA	Over-the-air
PGN	Parameter Group Number
PUF	Physical Unclonable Function

SOF	Start of Frame
REC	Received Error Counter
ROC	Receiver Operating Characteristic
RTR	Remote Transmission Request
RTS	Request to Send
TEC	Transmitted Error Counter
TPMS	Tire Pressure Monitoring System
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
VANET	Vehicular Ad Hoc Networks

1 Introduction

Carl Benz applied the first patent application for a vehicle powered by a gas engine in 1886 [1]. The three-wheeled vehicle, in Figure 1, had one cylinder and was able to reach 10 mph. It was a purely mechanical vehicle; the vehicle industry has revolutionised since then. Today's advanced vehicles have extensive automation with a mesh of sensors and computational systems to improve functionality and safety. These sensors are controlled by embedded Electronic Control Units (ECUs), designed for the optimal management of a wide array of functions ranging from engine control to Anti-lock Braking and Advanced Driver-Assistance Systems – ABS and ADAS, respectively. According to [2], [3], a modern automobile is fitted with more than a hundred ECUs, and this number is envisaged to increase in the future. These ECUs are distributed all around the vehicle. They communicate with each other via in-vehicle communication networks, such as Controller Area Network (CAN), FlexRay, Local Interconnect Network (LIN), and Media Oriented Systems Transport (MOST) [4]. The most common in-vehicle communication protocol CAN [5] offers advantages such as cost-effective wiring, immunity to electrical and magnetic interferences, self-diagnosing, and error correction mechanism.

Figure 1-1 The replica of Benz patent motor car

However, despite these functional benefits, the rising inter-vehicle and intra-vehicle communications render CAN vulnerable to cyber-attacks. The existing built-in security features of the CAN bus are primarily designed for ensuring

reliable communication, not for cybersecurity; therefore, it cannot prevent the network from cyberattacks.

The first attack on in-vehicle networks was implemented by Hoppe and Dittman in 2007[6], [7]. Since then, the attack surface of the vehicle has increased drastically in parallel to communication networks. In 2015, security researchers remotely hacked a Jeep Cherokee via a cellular network and were able to control steering and braking while the vehicle was moving [8]. Various physical and remote access attacks [9]–[12] demonstrated that the CAN network is defenceless to any attacks. As a result, the far-reaching implications of cyberattacks on CAN are anticipated. For instance, the attack on airbags [13] or ABS systems can jeopardise the driver and passengers' safety. Eventually, it may affect the car manufacturer's reputation with substantial financial implications, like recalls [14]. Tampering of ECUs, e.g., used-cars' odometer [15], is another example that may result in dire consequences for the consumers and the manufacturers. Overcoming such security shortcomings relies on developing efficient prevention mechanisms, which with the current state of the art, fall into four categories: network segmentation, encryption, authentication, and Intrusion Detection Systems (IDSs).

Network segmentation limits access to the critical ECUs by separating them from the user-accessible network via a gateway ECU. Although the method exists in commercial vehicles, it is not secure enough to stop adversaries. There are successful attacks that pass gateway ECU and intrude to the in-vehicle network [16].

Lack of encryption and authentication is the leading root cause of the CAN vulnerabilities. Although cryptographic techniques are the direct solution, implementing such algorithms is not feasible for CAN in automotive applications because of limited resources (bandwidth, computational power), the need for long service life, and time constraints. Researchers [17] have shown that current cryptographic methods are unsuitable for commercial vehicles due to significant overhead or backward incompatibility.

Hence, the problem's root cause is not feasible to solve by cryptographic techniques; the problem is mitigated by the Intrusion Detection System (IDS). An IDS targets to find malicious messages by analysing network traffic and generates an alert if there is any malicious activity.

1.1 Research Aim and Objectives

1.1.1 Problem Description

Given a vehicular network consisting of N modules allowed to broadcast messages in format M, there are potential threats in which an unknown adversary module broadcasts malicious messages with parameters of pattern *P*, signature *S*, and variation *V* while each parameter is associated with properties of existence *e*, strength *st* and recognised *r* (described in Table 1-1) and Equation (1-1). The problem of locating threats falls in utilising a supervisory module that monitors the messages on the CAN network provisioning to recognise the malicious messages, among others.

$$Malicious\ Message\ =\ M(P(e,st,r),S(e,st,r),V(e,st,r)) \quad \textbf{(1-1)}$$

As widely referred to, the supervisory module is the IDS system, which contributes to evaluating the network traffic against known attacks and unrecognised activity based on characteristics of the Malicious message given in Equation (1-1). Determining Message, M, in Equation (1-1) is not a straightforward task and requires understanding the system's behaviour, the performance of the driver, and the attackers.

Most current practices limit the IDSs to certain types of attacks mainly to reduce complexity, while others require highly intensive resources offering slightly robust performance. Still, none of the existing methods has a vehicle agnostic structure; therefore, they need training and testing for each vehicle make and model. Apart from lacking holistic IDS, there is no accepted standard for a testing methodology which brings many disadvantages.

Table 1-1 Parameters for locating threads

Parameter		
Name	**Symbol**	**Description**
Behaviour	B	Statistical or AI model that represents the data and network traffic and pinpoints anomalies.
Signature	S	Bit of patterns that match a particular attack's bit-sequence found within network packet headers, data, destination or source network, or in specific sequences of data or series of packets.
Variation	V	A shift or change in behaviour (B) or signature (S) for modelling the data traffic more precisely.
Property		
Name	**Symbol**	**Description**
Existence	e	It represents if a particular parameter is applicable for modelling a specific message or not.
Strength	st	Influence of behaviour or signature on a message in the form of strength or sensitivity.
Recognised	r	It is an index to the list of recognised behaviour or signature.

1.1.2 Hypothesis

The vehicular network has rigid timing rules to meet real-time operations. Any disruption in the system can lead to frequency changes. Frequency analysis tools like wavelet transform may detect this deviation. Wavelet transform is a powerful tool to analyse frequency variations over time. Unlike Fourier transform, wavelet analysis gives spectrum analysis on the time domain. Therefore, it can be used to identify malicious activities by analysing network traffic. The network traffic of

a vehicle is a non-stationary signal. Continuous Wavelet Transform (CWT) is an excellent tool to analyse the non-stationary signal.

1.1.3 Aim

This research aims to develop a propensity of vehicle-independent detection capability for IDS by using wavelet analysis. It allows low resource usage and fast detection time, which are the essential requirements for implementing an IDS in vehicular systems.

1.1.4 Objectives

To achieve the aim of this research, the following objectives are set out to be met.

1. A literature review to identify gaps and problems regarding CAN security
2. Design and implementation of an IDS for CAN which has the following criteria:
    a. vehicle agnostic implementation
    b. ability to detect attacks swiftly
    c. low false-positive alarms
3. Data generation to mimic the CAN bus attacks
4. Design and development of a testing framework and attack generation to demonstrate the ability and limitations of the designed IDS.

1.2 Research Methodology

This research is divided into four phases to achieve research objectives and the aim, as shown in Figure 1-2. In order to fulfil objectives and achieve the aim, the literature is surveyed, and research gaps are identified. A Wavelet-based Intrusion Detection System (WINDS) is proposed. A comprehensive benchmarking framework is constructed to evaluate the proposed method. Then WINDS is tested according to this framework. The technique is also compared with other methods by evaluating on the same network traffic. After the verification and validation phase, the work is inspected to improve its capabilities.

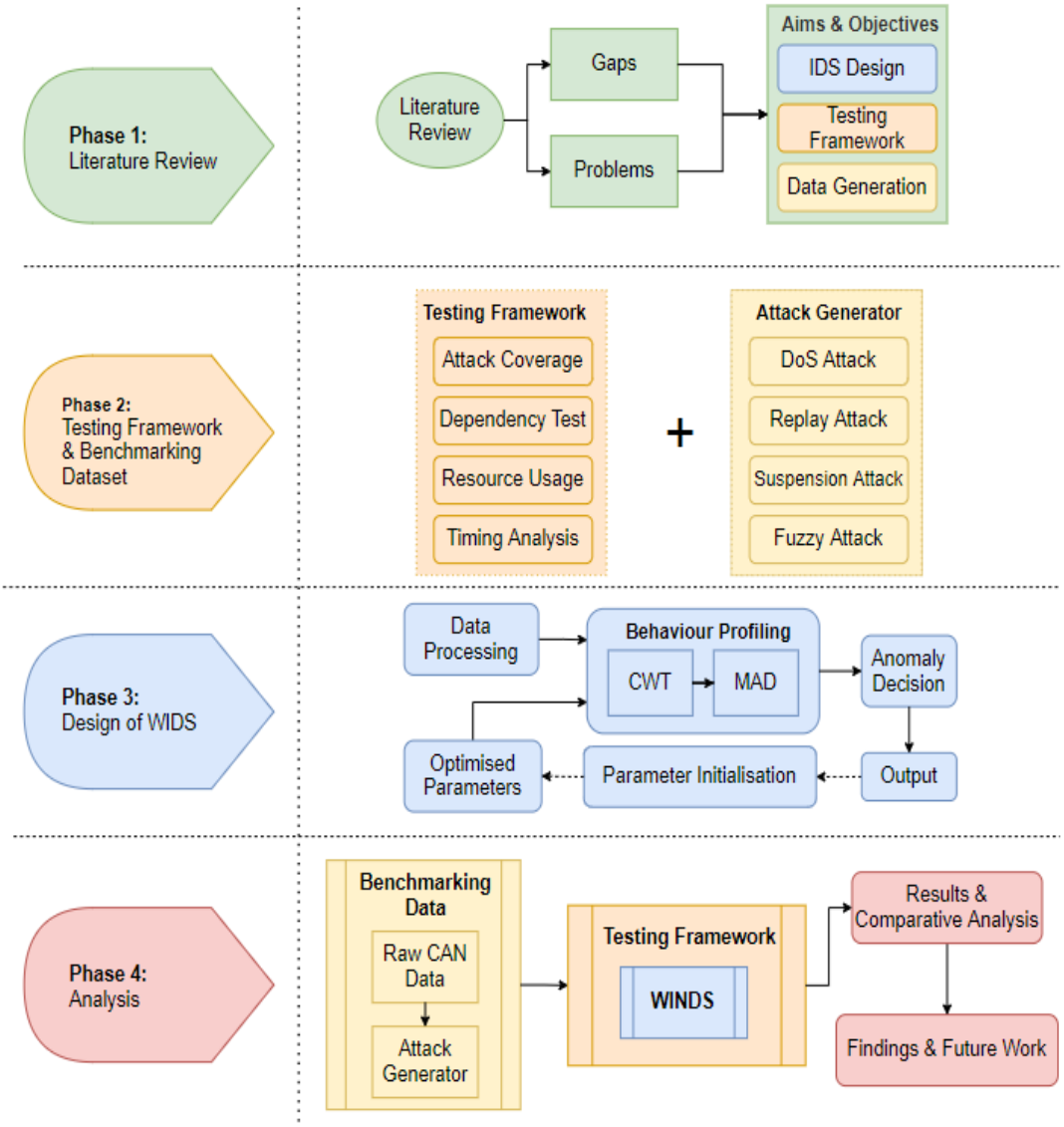

Figure 1-2 Methodology flow diagram

1.2.1 Phase 1: Literature Review

This phase is the beginning of the research process to identify state-of-the-art solutions and gaps in in-vehicle security. Various topics are studied to obtain the knowledge to analyse CAN security comprehensively, understanding existing research, and finding the research gaps. The topics are vulnerabilities of CAN, a survey on security solutions for in-vehicle networks, and a detailed analysis of intrusion detection systems. As a result of the literature review, the following research gaps were identified.

i. There are multiple solutions to secure CAN bus, and an intrusion detection system is the most feasible solution because of the limited resources and timing constraints. However, none of the existing IDSs has a vehicle-independent resolution, and most of them require extensive training.
ii. The lack of available datasets is an obstacle to the development of reliable IDS.
iii. Lack of an accepted benchmarking framework causes improper testing and misleading results.

Phase 1 helped in defining the aim and objectives of this research. As an outcome of the literature review, a journal paper and a conference proceeding were published.

1.2.2 Phase 2: A Framework for Benchmarking Intrusion Detection Systems for CAN Bus

Benchmarking is essential for researchers, security analysts, and customers. A good quality benchmark helps researchers and security analysts understand the strength and limitations of the IDS; therefore, they can focus on improving IDS' weaknesses. Customers also benefit from benchmarking results and decide on suitable IDS methodology for their network.

Although researchers proposed various Intrusion Detection Systems (IDSs) to identify intrusions in the CAN network, there is no accepted testing methodology and enough dataset. The field is still in its infancy, and produced works lacks comprehensive evaluation and comparative analysis. A 2018 survey [18] shows that only 4 out of the 65 surveyed papers compare their works with baseline methods.

In addition to presenting whether the method works or not, proper testing also illustrates how well the method works. Lack of standardised testing methodology causes a variety of problems. First, many IDS solutions are not appropriately tested, which generates misleading results. The second problem is that it is difficult to compare existing solutions which hardens researchers' tasks to

evaluate their methodologies. The issue with comparison also hinders reaching the best IDS.

This phase focused on generating a comprehensive benchmarking dataset and defining benchmarking criteria covering Objective 3 and Objective 4. The proposed benchmarking framework assesses an IDS with quantifiable metrics. Therefore, it is a repeatable and objective testing methodology. The test result is not a binary output, and it will show the strengths and weaknesses of the IDS under the test. It identifies the performance metrics and attacks conditions to be tested. By adopting this approach, IDS solutions can be verified objectively and efficiently compared with other solutions which use the same benchmarking framework.

1.2.3 Phase 3: WINDS: A Wavelet-based Intrusion Detection System for Controller Area Network (CAN)

A novel vehicle agnostic intrusion detection system based on wavelet analysis is proposed to address the gaps mentioned earlier. This is the main contribution of this research. The WINDS analyses the CAN-network traffic behaviour using wavelets, and it can be used as a tool for intrusion detection. To the best of our knowledge, this is the first attempt applied to vehicular applications.

The proposed wavelet-based IDS does not require training phases, and further, it is independent of the driver's driving style.

1.2.4 Phase 4: Verification and Validation

This phase presents the evaluation results of WINDS according to the proposed benchmarking framework in Phase 2. The WINDS is tested on various scenarios using real and synthetic attack data. The chapter demonstrates the competitiveness of WINDS with state-of-the-art methods. It also presents the vehicle agnostic behaviour of WINDS, which is a significant advantage over existing solutions. After showing the competitiveness of WINDS, this phase summarises the current form of the WINDS and presents future works to improve its capabilities and resource usage. Although WINDS has major vehicle agnostic behaviour as an advantage over existing solutions, there are some ways to

improve it. This can be achieved by focusing on comprehensive thresholding, analysis of discrete wavelet transforms, detecting infrequent node attacks, transformation to intrusion prevention system, and optimisation.

1.3 Risks and Mitigations

1.3.1 Dataset

Dataset is a vital element of the research to implement and test the methodology. However, implementing attacks on a running vehicle has a serious safety risk for people onboard and the surrounding environment. Therefore, attacks should be implemented in specialised areas like airports or controlled environments. The implementation of the attacks is also costly, requiring specialised tools, insurance, and an actual vehicle. To mitigate these problems, open-source datasets are used. Although there are a limited number of available datasets and those datasets have limited variations, a comprehensive benchmarking dataset can be obtained by synthetically simulating the attacks on available datasets.

In this research, datasets from two independent research centres are used; “Automotive Controller Area Network (CAN) Bus Intrusion Dataset v2” [19] and “Car-Hacking Dataset” [20]. More data is generated by mimicking the CAN bus attacks virtually.

1.3.2 Evaluation and Comparative Analysis

There is no standardised testing methodology and accepted benchmarking dataset. As a result, it becomes difficult to compare the proposed method with existing solutions. To overcome this issue, testing methodology in Information Technology (IT) is studied, and a comprehensive framework is constructed. The WINDS is tested according to the proposed framework.

To get the comparative analysis, the WINDS is tested on the same dataset, “Car-Hacking Dataset”, with other methods along with the frequency-based baseline method. Although used “Car-Hacking Dataset” has some deficiencies and lacks experimental details, it is a valuable dataset used by many researchers. Hence,

the dataset has a limited number of attack conditions; using this dataset cannot guarantee the judgemental decision, but it can be a comparison.

1.4 The Organisation of the Thesis

The remaining part of this thesis is organised as follows: Chapter 2 presents a comprehensive literature review of CAN bus security focusing on intrusion detection systems. Chapter 3 outlines the details of the wavelet-based intrusion detection system (WINDS) for in-vehicle networks. Then Chapter 4 presents the data generation and the framework for testing vehicle IDS. Additionally, this chapter evaluates the WINDS algorithm and compares it with other state-of-the-art methods. Finally, Chapter 5 concludes the thesis and outlines future directions.

1.5 Publications & Activities

1.5.1 List of Publications

Journal Papers:

I. M. Bozdal, M. Samie, S. Aslam, I. Jennions, “A Wavelet-based Intrusion Detection System for Controller Area Network (CAN)”, IEEE Access, 2021. (DOI: https://doi.org/10.1109/ACCESS.2021.3073057).

II. M. Bozdal, M. Samie, S. Aslam, I. Jennions, “Evaluation of CAN bus security challenges”, Sensors, 2020. (DOI: https://doi.org/10.3390/s20082364).

Conference Papers:

I. M. Bozdal, M. Samie, I. Jennions, “A survey on CAN bus protocol: Attacks, challenges, and potential solutions”, IEEE International Conference on Computing, Electronics and Communications Engineering, 2018. (DOI: https://doi.org/10.1109/iCCECOME.2018.8658720)

II. M. Bozdal, M. Randa, M. Samie, I. Jennions, “Hardware trojan enabled denial of service attack on CAN bus”, 7th International Conference on Through-life Engineering Services, 2018. (DOI: https://doi.org/10.1016/j.promfg.2018.10.158)

1.5.2 Training and Networking Activities

Numerous courses and activities are attended to keep up to date in the field and share ideas with the other researchers. Some of the main ones are summarised below.

IVHM Technical Review Meeting: These meetings are held three times a year, and all the meetings organised during this PhD research were attended. It allows to the presentation of research progress to colleagues and industrial partners. The feedback from the meetings improved the research quality and presentation skills.

AESIN Security Conference: AESIN is a non-profit organisation recognised by the Automotive Council UK. It is a response to the explosion of electronics in-car, which is approaching 50% of vehicle cost. There are specialised conferences covering a variety of automotive electronics. AESIN Security Conference focuses on the cyber resilience of connected automobiles. The following meetings were attended.

- AESIN Security Virtual Conference, 15 July 2020, Online
- AESIN Security Conference, 10 July 2019, Coventry UK

Vector Cybersecurity Symposium: Vector is a leading company that provides tools, software components, and services for automotive and related industries. Vector Cybersecurity Symposium brings together state-of-the-art industry solutions and ongoing academic works. It gives insights into safety and security integration in practice, security standards, and solutions for automotive cybersecurity. The following conference was attended.

- Vector Cybersecurity Symposium, 2 - 4 April 2019, Stuttgart Germany

Europractice Hardware Security Course: It is a five-day hands-on course for designing secure ICs and systems in different application domains. It covers security, encryption and security threats, secure implementations resistant to passive and active attacks, and security building blocks like random number generators and physically unclonable functions. The following course was taken.

- Europractice Hardware Security Course,10-14 December 2018, Leuven, Belgium

References

[1] Daimler AG, “Benz Patent Motor Car: The first automobile (1885–1886).” https://www.daimler.com/company/tradition/company-history/1885-1886.html (accessed Mar. 21, 2021).

[2] E. Hira, “Automotive Electronic Control Unit (ECU) market size share, 2022,” *Allied Market Research*, 2017. https://www.alliedmarketresearch.com/automotive-electronic-control-unit-ecu-market (accessed Jul. 22, 2018).

[3] “‘ECU’ is a three letter answer for all the innovative features in your car: know how the story unfolded,” *Embitel*, 2017. https://www.embitel.com/blog/embedded-blog/automotive-control-units-development-innovations-mechanical-to-electronics (accessed May 23, 2018).

[4] T. Kosch, C. Schroth, M. Strassberger, and M. Bechler, *Automotive Internetworking*. Chichester, UK: John Wiley & Sons, Ltd, 2012.

[5] K. Matheus and T. Königseder, *Automotive Ethernet*. Cambridge University Press, 2015.

[6] T. Hoppe and J. Dittman, “Sniffing/Replay Attacks on CAN Buses: A simulated attack on the electric window lift classified using an adapted CERT taxonomy,” in *Proceedings of the 2nd workshop on embedded systems security (WESS)*, 2007, pp. 1–6.

[7] B. Groza and S. Murvay, “Security solutions for the Controller Area Network: Bringing Authentication to In-Vehicle Networks,” *IEEE Vehicular Technology Magazine*, pp. 40–47, 2018.

[8] C. Miller, “Lessons learned from hacking a car,” *IEEE Des. Test*, vol. 36,

no. 6, pp. 7–9, 2019, doi: 10.1109/MDAT.2018.2863106.

[9] S. Fröschle and A. Stühring, “Analysing the capabilities of the CAN Attacker,” in *European Symposium on Research in Computer Security*, Sep. 2017, vol. 10492 LNCS, pp. 464–482, doi: 10.1007/978-3-319-66402-6_27.

[10] C. Miller and C. Valasek, “A survey of remote automotive attack surfaces,” 2014. Accessed: Mar. 31, 2018. [Online]. Available: https://www.ioactive.com/pdfs/IOActive_Remote_Attack_Surfaces.pdf.

[11] P.-S. Murvay and B. Groza, “DoS attacks on Controller Area Networks by fault injections from the software layer,” in *Proceedings of the 12th International Conference on Availability, Reliability and Security - ARES ’17*, 2017, vol. 10, pp. 1–10, doi: 10.1145/3098954.3103174.

[12] A. Palanca, E. Evenchick, F. Maggi, and S. Zanero, “A stealth, selective, link-layer denial-of-service attack against automotive networks,” in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Jul. 2017, vol. 10327 LNCS, pp. 185–206, doi: 10.1007/978-3-319-60876-1_9.

[13] J. Dürrwang, J. Braun, M. Rumez, and R. Kriesten, “Security evaluation of an airbag-ECU by reusing threat modeling artefacts,” in *2017 International Conference on Computational Science and Computational Intelligence, CSCI 2017*, 2018, pp. 37–43, doi: 10.1109/CSCI.2017.7.

[14] C. Matthews, “Jeep Hack: Fiat recalls 1.4 million vehicles for software fix,” *Fortune*, 2015. https://fortune.com/2015/07/24/jeep-cherokee-recall/ (accessed Mar. 27, 2020).

[15] D. Maloney, “Dashboard dongle teardown reveals hardware needed to bust miles,” *Hackaday*, 2019. https://hackaday.com/2019/12/16/dashboard-dongle-teardown-reveals-hardware-needed-to-bust-miles/ (accessed Mar. 21, 2020).

[16] S. Nie, L. Liu, and Y. Du, “Free-Fall : Hacking Tesla from wireless to CAN

bus," in *BlackHat USA 2017*, 2017, pp. 1–16, Accessed: Nov. 30, 2018. [Online]. Available: https://www.blackhat.com/docs/us-17/thursday/us-17-Nie-Free-Fall-Hacking-Tesla-From-Wireless-To-CAN-Bus-wp.pdf.

[17] N. Nowdehi, A. Lautenbach, and T. Olovsson, "In-vehicle CAN message authentication: An evaluation based on industrial criteria," in *IEEE Vehicular Technology Conference*, 2017, vol. 2017-Septe, pp. 1–7, doi: 10.1109/VTCFall.2017.8288327.

[18] G. K. Rajbahadur, A. J. Malton, A. Walenstein, and A. E. Hassan, "A Survey of Anomaly Detection for Connected Vehicle Cybersecurity and Safety," in *IEEE Intelligent Vehicles Symposium*, 2018, vol. 2018-June, pp. 421–426, doi: 10.1109/IVS.2018.8500383.

[19] G. Dupont, A. Lekidis, J. Den Hartog, and S. Etalle, "Automotive Controller Area Network (CAN) Bus Intrusion Dataset v2," *4TU.ResearchData*, 2019. https://data.4tu.nl/repository/uuid:b74b4928-c377-4585-9432-2004dfa20a5d (accessed Dec. 09, 2019).

[20] H. K. Kim, "Car-Hacking Dataset," *Hacking and Countermeasure Research Lab*. https://sites.google.com/a/hksecurity.net/ocslab/Datasets/CAN-intrusion-dataset (accessed Dec. 01, 2020).

2 Evaluation of CAN Bus Security Challenges

The automobile industry no longer relies on pure mechanical systems; instead, it benefits from intelligent features based on advanced embedded electronics. Although the rise in electronics and connectivity has improved comfort, functionality, and safe driving, it has also created new attack surfaces to penetrate the in-vehicle communication network, which was initially designed as a close loop system. Although the Controller Area Network (CAN) is the most widely used communication protocol, it still suffers from various security issues because of the lack of encryption and authentication. As a result, any malicious/hijacked node can cause catastrophic accidents and financial loss. This chapter analyses the CAN bus comprehensively to provide an outlook on security concerns. First, it gives the CAN protocol details, standardised by ISO 11898-1:2015. Then, the protocol's vulnerability is assessed based on confidentiality, integrity, and availability. The chapter continues with existing attacks and presents a state-of-the-art attack surface. It goes through different solutions that assist in attack prevention, mainly based on an intrusion detection system (IDS). The chapter is finalised with a discussion section covering the standardisation of automotive cybersecurity, industrial products, and obstacles preventing CAN security research. The organisation of the chapter is presented in Figure 2-1.

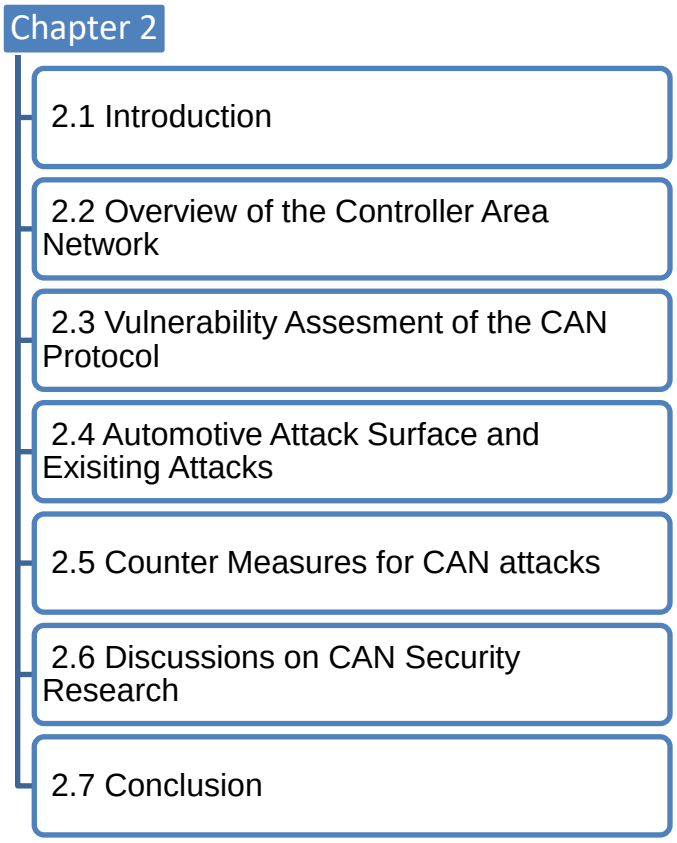

Figure 2-1 Organisation of Chapter 2

2.1 Introduction

Modern vehicles are equipped with dozens of Electronic Control Units (ECUs) to improve driving comfort and safety[1][2]. ECUs control most of the car's functions, including safety-critical ones like engine control, airbag deployment, and anti-lock braking system. To have safe driving, ECUs should have a reliable communication network. One of the main in-vehicle communication protocols is Controller Area Network (CAN). Its well-recognised advantages, such as high immunity to electrical interference, easy wiring, ability to self-diagnose, and mitigating errors, make CAN bus suitable for the automobile industry. Although CAN is resilient to electrical noise and has reliable communication features, it is still vulnerable to attacks.

The first known attack on the CAN bus was implemented on the electric window lift in the simulation environment by Hoppe and Dittman in 2007[3]. Since then, different attack scenarios have been implemented [4]–[7]. While most attacks are

implemented via physical access to the bus, wireless attacks are increasing. The wireless attack surface will continue to grow with the new wireless interfaces like Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I).

Equally alarming is the lack of encryption in CAN, which has a strong bearing on individual data privacy. By design, CAN is a broadcast network that allows nodes to capture messages going through the network. An adversary can acquire the desired data as the broadcasted data is not encrypted. This may lead to an invasion of privacy, mainly when modern cars are capable of acquiring the driver's personal information.

According to the 2019 industry survey [8], safety and security are the highest short-term and mid-term challenges for the automotive industry. Therefore, extensive studies have been carried out to find possible solutions [3], [9] to the vulnerabilities of CAN. Some of these studies have performed successful experimental attacks on passenger cars [4], [5], [10]–[13] and heavy-duty vehicles [14], [15]. At the same time, researchers have also proposed preventative methods for such known attacks. These include network segmentation, encryption, authentication, and intrusion detection systems (IDSs).

In light of the above, this chapter provides a comprehensive literature review with the following main contributions:

a. Identification of the state-of-the-art and the most potential security challenges associated with modern vehicles, covering a number of implemented physical and remote access attacks.

b. Highlighting the attack surfaces of modern vehicles with a critique on possible future attacks.

c. An in-depth analysis of the current research on CAN security issues to facilitate their effective and optimal mitigation.

2.2 Overview of the Controller Area Network (CAN)

The CAN bus is a multi-master broadcast communication protocol developed by Robert Bosch GmbH in the early 1980s. A traditional CAN interface can provide

up to 1 Mbps [16]. In 2012, Bosch released the CAN FD (flexible data-rate), which can achieve 5 Mbps in practice and has a 64-byte payload compared to 8 bytes in the classical CAN [17]. CAN FD is backwards compatible and can coexist with classical CAN nodes. Classical CAN and CAN FD are both standardised under ISO 11898-1:2015.

The single two-wire bus architecture of CAN, as shown in Figure 2-2, reduces cabling. The distributed architecture of the network provides easy maintenance and decreases the overall system cost. Moreover, the protocol uses differential wiring mode, represented by CAN_H and CAN_L, enhancing immunity to noise and electrical interference. From a logic point of view, signals have two states (voltage levels): a dominant logic '0' and a recessive logic '1', meaning that the bus signal remains '0', the dominant logic, as long as one of the nodes releases logic '0' to the bus. As there is no dedicated clock line, synchronisation is provided via signal edges and bit-stuffing. The Bit-stuffing rule limits the number of repeated bits. After five consecutive bits of the same logic level, the next bit must complement the previous logic level; otherwise, it will cause a protocol error. If the data has more than five successive corresponding bits, a complement bit is inserted by the transmitter CAN controller and the receiver ignores it.

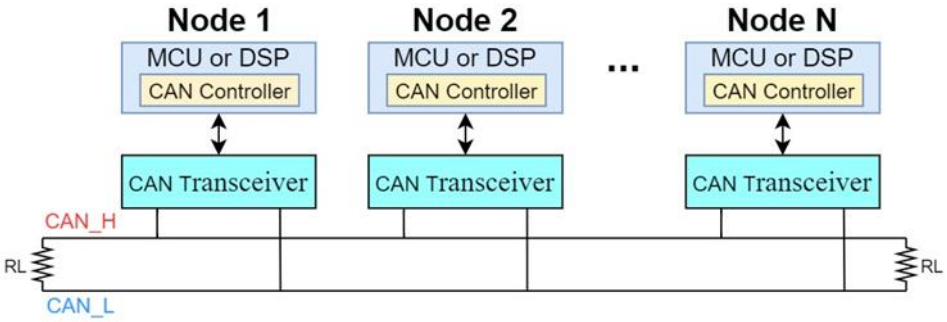

Figure 2-2 An example of a single two-wire Controller Area Network (CAN).

The CAN protocol has message-based communication provided via frames, as shown in Figure 2-3. Each frame has a message identifier field, data field, cyclic redundancy checksum (CRC), and some control bits. Every node listens to each

frame and processes the relevant ones based on the message identifier field, which is also used for the arbitration.

SOF	Message Identifier	RTR	IDE	r0	DLC	Data	CRC	ACK	EOF	IFS
1-bit Dominant	11-bit or 28-bit (Arbitration Field)	1-bit	1-bit	1-bit	4-bit	0 to 8 Bytes	15-bit Checksum 1 - bit Delimeter	1 - bit Acknowledgement 1 - bit Delimeter	7-bit Recessive	-

Figure 2-3 Classical CAN frame structure.

2.2.1 Reliable Communication in CAN

The CAN protocol has a set of built-in features that provide robust communication. Suppose two nodes start transmitting at the same time. In that case, the non-destructive arbitration mechanism resolves the conflict by allowing the highest priority node to continue the transmission without any interruption (e.g., Node 1 wins arbitration in Figure 2-4, without any disruption, as the dominant bit overrides the recessive one). Another feature is carrier sense multiple access with collision avoidance (CSMA/CA), which rules that the nodes have to wait for a certain amount of inactivity before the transmission. This assists in sensing if the bus is idle and ensuring that a collision will not occur.

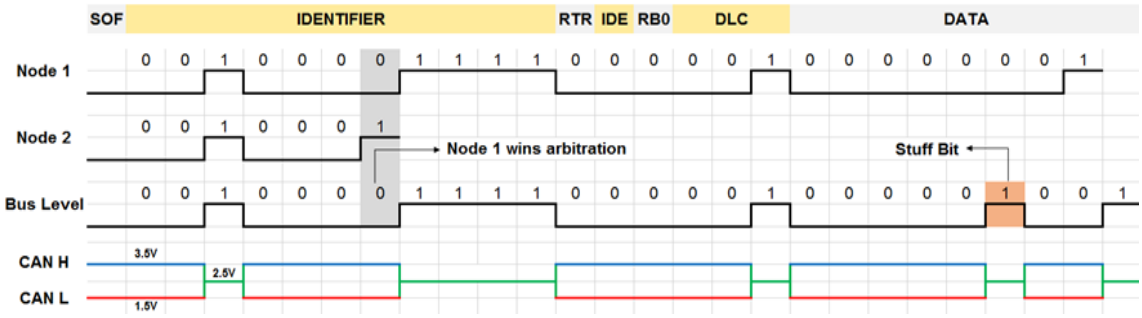

Figure 2-4 Signalling in CAN; Node 1 wins arbitration without any disruption.

The CAN bus has some bit-level and message-level error checking mechanisms. At the bit level, the transmitter node monitors the bus. An error arises if there is a difference between the transmitted bit and the one observed on the bus. On the other hand, the message-level CAN bus error check mechanism includes frame check over acknowledgement (ACK), cyclic redundancy checksum (CRC), and end of frame (EOF) fields. After the transmission of a frame, the transmitter node writes a recessive bit to the ACK field. If a node receives a message correctly, it

overwrites the ACK field with a dominant bit; otherwise, the ACK field stays recessive, which indicates a transmission error. There is up to a 21-bit CRC field in a CAN frame for data integrity. An error flag will be sent if any node calculates a different CRC than the transmitter node. The CRC delimiter, ACK delimiter, and EOF bits have fixed values and must always be recessive. An error is generated if these bits are dominant during the frame form check.

CAN also prevents the physical errors by disabling the faulty nodes from the bus traffic with an error confinement mechanism (ECM), as shown in Figure 2-5. The ECM is facilitated in each node using two error counters known as the received error counter (REC) and transmitted error counter (TEC). The TEC increases by eight if an error occurs during the transmission, and REC increases by one if the error comes during the reception. Every successful transmission or reception of a frame decreases the responsible counter by one. The counters' default values are zero, and nodes start at the error active state. A node will enter the error passive state if the value of the node's counter exceeds 127. In an error passive state, the node can only write recessive error flags, which will not affect the bus traffic. The node turns to the bus off state if the TEC counter exceeds 255, meaning that the affected node will no longer participate in the bus traffic.

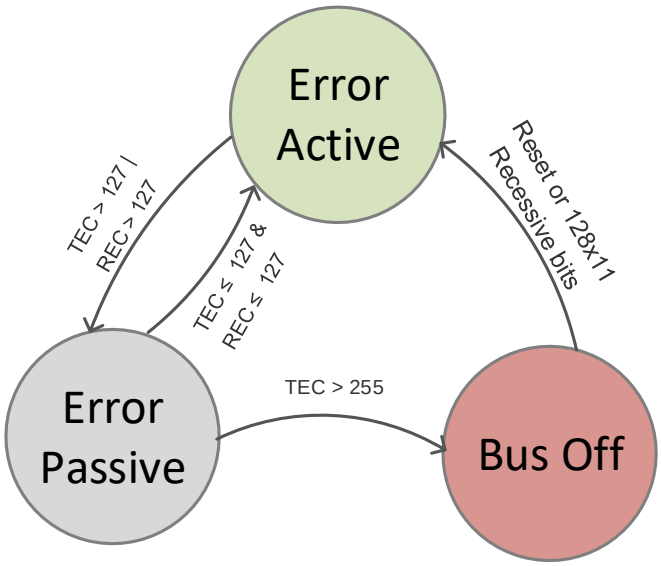

Figure 2-5 The state diagram of the error confinement mechanism (ECM) in the CAN bus.

2.3 Vulnerability Assessment of the CAN Protocol

It is essential to have a vulnerability assessment of a network to highlight security problems. Therefore, the CAN protocol's vulnerability assessment can be carried out based on confidentiality, integrity, and availability.

Confidentiality means providing the data only to authorised people. However, the CAN protocol does not have inherent cryptographic methods to ensure confidentiality. This allows an intruder to access sensitive user data and cause an invasion of privacy.

Integrity is the accuracy, completeness, and validity of the data. The CAN bus has a CRC to verify integrity against transmission errors, but it cannot prevent data injected by malicious parties, which breaks the integrity. The protocol does not have a comprehensive integrity check and fails to sustain integrity.

Availability means that authorised users can use the system at all times. Given the nature of priority-based messaging, if a message with the highest priority is transmitted/inserted, the network will be inaccessible by the lower priority nodes, and availability is violated.

The CAN bus failed to pass all three essential security criteria. Thus, it is clear that the CAN protocol does not have any security measurements against the attacks.

2.4 Automotive Attack Surface and Existent Attacks

In the 1950s, automotive electronics cost only 1% of the total car cost, while it is currently 35% and is expected to rise to 50% in 2030 [18]. Although the rise in electronics has improved comfort, functionality, and driving safety, it has created new attack surfaces, as shown in Figure 2-6. The protocol itself is defenceless to attacks; therefore, any exploit in the current/future telematics unit or infotainment system can disrupt the network, as summarised in Table 2-1.

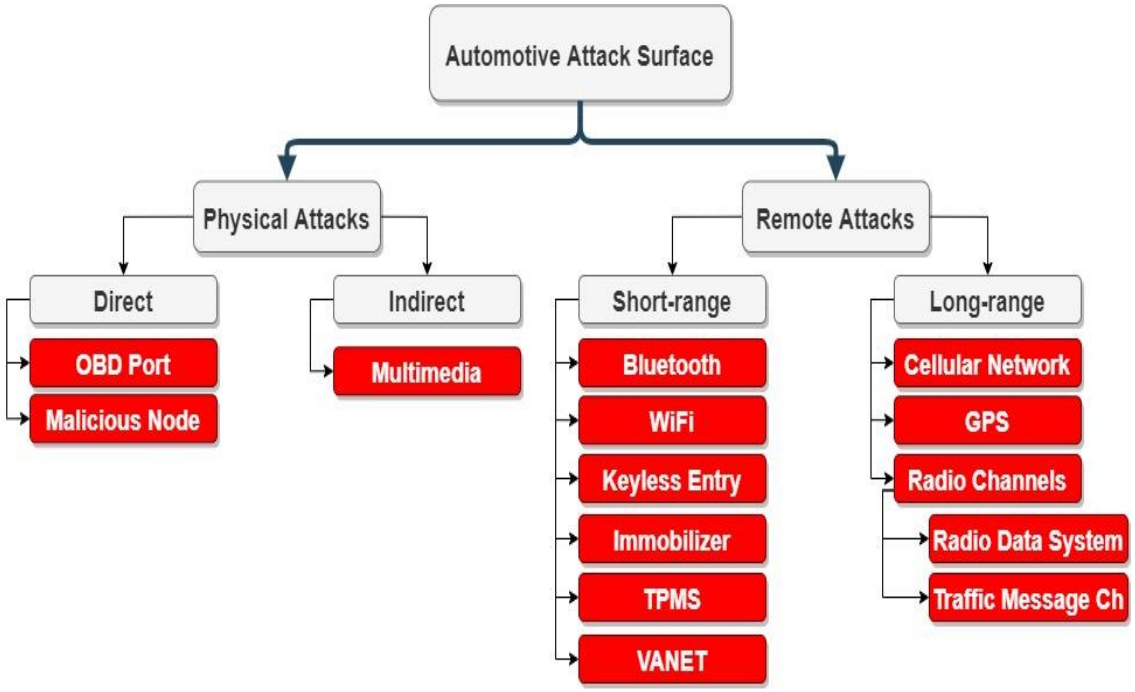

Figure 2-6 The automotive attack surface.

The first CAN bus attack was performed on the power window by Hoppe and Dittman in 2007 [3], [24]. Since then, numerous attacks have been performed. These attacks can be categorised as physical access attacks, where the attacker should access the vehicle physically, or remote attacks, which are implemented via wireless communication interfaces. Although attacks in the literature are mainly physical access, some experts have argued that physical access to the CAN network is not practical [25]. Therefore, current research is primarily focusing on remote access attacks.

Table 2-1 Summary of the Controlled Area Network (CAN) bus attacks.

Ref.	DoS	Modification [1]	Access Type	Notes / Root Cause
[4]	Y	N	OBD II	Does not require full CAN messages
[19]	N	Y	OBD II, CD, Bluetooth, GSM	Systematical experimental attacks. Indirect access via the car service computer
[20]	N	Y	In-direct OBD II	Attack via a smartphone app
[21]	Y	Y	Multiple remote sources	Remote attack analysis of 21 commercial cars
[5]	N	Y	Wi-Fi, GSM	Access CAN network via a browser exploit
[14]	Y	N	OBD II, compromised ECU	SAE J1939 data-link layer exploits
[22]	N	Y	Wi-Fi, GSM	Ransomware attack over the air
[23]	N	Y	TPMS	Remotely sending false TPMS data

[1] The modification includes replay, impersonation, and bogus information attacks.

2.4.1 Physical Access Attacks

Physical access attacks require direct or indirect access to the CAN bus network. Direct access can be obtained by the On-Board Diagnostic (OBD) port or a malicious node. The OBD port is the primary attack surface; hence, it has access to all nodes, even though network segmentation is used.

Koscher et al. [11] manipulated the CAN and controlled various modules, including essential brake control and engine control modules through the On-Board Diagnostics II (OBD-II) port. They released the brake and prevented its activation while the car was running 40 mph by the continuous fuzzing method. The attack also includes manipulating the instrument cluster with false data, changing engine parameters and disabling it.

Due to the CAN architecture, any malicious node can listen or send a message to disrupt the network. The attacks implemented through the OBD port can be replicated using a malicious node. Palanca et al. [4] applied a selective Denial-of-Service (DoS) attack on an unmodified 2012 Alfa Romeo Giulietta. The research showed that any person with physical access to the network could disrupt it, even with a simple tool. This attack does not require a complete message transmission; instead, it overwrites the recessive bits and generates a transmission error. The contribution of this research is that it exploited the vulnerability of the CAN standard. After this research, an alert (ICS-ALERT-17-209-01) [26] was announced by the U.S. government. A similar research analysis was carried out by Murvay and Groza [27] to show the attack's limitations on different bit rates and breach the authentication methods.

Mukherjee et al. [14] implemented DoS attacks on the SAE J1939 standard [28], used in heavy-duty commercial vehicles. They performed three separate DoS attacks: (i) sending too many request messages for a supported Parameter Group Number (PGN) to overload the recipient ECU, (ii) sending manipulated false Request to Send (RTS) and causing overflow at the recipient buffer, and (iii) keeping the connections open via Clear to Send (CTS) messages and occupying the whole network. This work was one of the first studies to exploit the SAE J1939 specification. Murvay and Groza [15] implemented impersonation and DoS attacks on SAE J1939. These works showed that SAE J1939 is vulnerable to protocol-specific attacks in addition to all CAN bus attacks.

There can also be indirect physical access attacks. These attacks require a physical object to be inserted into the car, but adversaries do not necessarily have direct access to the network. Checkoway et al. [19] developed an indirect

access attack model, which included hacking the car service's IT system and accessing the CAN via computer. The attack model also included attacking via multimedia devices (CD, USB, or MP3 player). Hoppe et al. [12] implemented an attack with a multimedia disc. Although the attack did not breach the CAN, it may scare the driver by flashing a warning on the screen and playing an alarm signal.

2.4.2 Remote Access Attacks

Nowadays, modern vehicles contain different types of wireless interfaces needed for communicating with systems such as passive anti-theft, Tire Pressure Monitoring System (TPMS), Bluetooth, radio data, telematics, and so on. These wireless interfaces need to communicate with the CAN, usually via a gateway ECU to protect the network. However, some studies have demonstrated a gateway ECU hacking and gaining access to the isolated CAN [12].

Checkoway et al. [19] compromised the TPMS, Bluetooth, FM channel, and a car's cellular network through reverse engineering. They claimed that thieves could steal vehicles easily as doors could be unlocked through CAN messages. Woo et al. [20] proposed a remote attack via a malicious self-diagnostic app. If someone uses a malicious app to monitor/diagnose the vehicle's situation, the adversary takes control of the vehicle remotely and performs its attack from a long distance.

Valasek and Miller [21] carried out a remote attack survey on 12 car brands and 21 commercial cars and identified the remote attack surfaces and their difficulties in compromising each vehicle. The attack was three-staged. The first stage was to compromise the ECU responsible for a wireless interface. The second stage was to inject messages to communicate with the safety-critical ECU. The last stage was to modify the ECU to behave maliciously. While the researchers believed that the increasing number of cyber-physical systems in the cars would increase their vulnerabilities, they could not practically verify this because of the high number of different applications in the vehicles. Furthermore, they also hacked a Jeep Cherokee remotely and disabled the engine in 2014 [10]. After this attack, a public announcement that stated the vulnerability of motor vehicles against remote attacks was published [29].

Savage and his team [30] took control of a Chevrolet Corvette's brakes and windshield wipers via a commercial telematics control unit in 2016. This attack indicates that the CAN's vulnerability can be penetrated by the aftermarket equipment and cannot be entirely addressed by the manufacturer [31].

Nie et al. [5] implemented a remote attack on a Tesla Model S in 2016 via wireless and cellular interfaces. The Keen Security Lab of Tencent [13] discovered multiple attack surfaces on BMW vehicles, which showed that even high-end commercially available cars could suffer from cyber-attacks.

Another wireless attack method is over-the-air (OTA) software updates. OTA is a cost-effective and scalable solution that allows manufacturers to deliver software updates remotely. However, it is another attack surface where hackers can dive into the vehicle's communication network. Beek and Samani [22] implemented a ransomware attack via an OTA update.

The remote attack surface of the modern car is more substantial than the physical one. With the rising connectivity in vehicles, the number of wireless attack surfaces is increasing day by day. In the near future, cars will be equipped with vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications, which build vehicular ad hoc networks (VANETs). VANETs aim for traffic optimisation and collision avoidance. To provide these benefits, VANETs use car sensors and have wireless connectivity. In VANETs, spoofed messages can be received or transmitted, and as a result, the in-vehicle communication network may be disrupted.

2.4.3 Privacy in the CAN

Acquiring CAN network data causes not only safety issues but also the invasion of privacy. The modern vehicle collects data related to the driver, which passes through the vulnerable CAN network. An investigation [32] revealed that it was possible to obtain the car's precise location history and other personal data (log of phone calls, list of contacts, email addresses, and photos) from the connected phone. An adversary can steal personal information only by passively listening to the bus. Furthermore, researchers [33], [34] have shown that it is possible to

identify the driver based on the sensory data travelling through the CAN bus. Therefore, monitoring the in-vehicle network can invade personal privacy.

2.5 Counter Measures for CAN Attacks

The attacks on CAN clearly show that the protocol is very vulnerable and requires cyber defence mechanisms for safe driving. The studies to solve this problem have mainly focused on four defence mechanisms: network segmentation, encryption, authentication, and intrusion detection, summarised in Table 2-2.

Table 2-2 Methods to secure the CAN bus.

Proposed Method	Benefits	Disadvantages
Network Segmentation	Limit access to the end-user	Increased cost, Difficulty in maintenance
Encryption	Hardened attacks, Confidential data transmission	Increased computational power, Increased traffic, Weak encryption due to frame size
Authentication	Secure data transmission	Increased computational power, Increased traffic
Intrusion Detection	Detect anomalies and attacks	Complicated algorithm design, Cannot guarantee the security

2.5.1 Network Segmentation

The most straightforward protection mechanism is separating the CAN network into multiple subnetworks. The segmentation provides control over who can access a particular subnetwork and reduce the attack's damage by limiting its spread. The interconnection between subnetworks is controlled via a gateway ECU. This model currently exists in commercial vehicles. The method is simple to implement, but it is ineffective if the gateway ECU is compromised or manipulated like the hacking exhibited in [12]. Kammerer et al. [35] addressed this issue and proposed a star coupling router with security features. The paper ignored the security inside a subnetwork, but it is possible to implement a replay attack in a subnetwork and attack the other subnetworks bypassing the security check of the router.

Researchers at TU München proposed an automotive service bus architecture [36] whose two-layer architecture was designed to prevent external attacks. The infotainment system and all vital functions were separated from each other. All components could send and receive messages, but by default, they could not send any data as the central ECU allows whom to write to the automotive service bus.

Network segmentation increases the security level, but it is not sufficient to protect the CAN. It also makes the maintenance of the system more complicated, along with the increased cost.

2.5.2 Encryption

The CAN protocol uses a shared broadcast network without a built-in encryption mechanism. This allows an adversary to eavesdrop on all the nodes and understand the communication. To prevent this data breach, a lightweight encryption system should be implemented. Although there are commercial software-based encryption methods (e.g., Trillium [37], CANcrypt [38]) and manufacturers have proprietary encryption techniques implemented in cars, there have been reports claiming that encryption mechanisms in commercially available vehicles can be broken [39],[40].

The limited data field is one of the problems for secure CAN encryption. This problem can be overcome by sending multiple CAN frames for a single message and may solve the problem on low traffic networks. Still, it is not a solution for the currently rising traffic in automobile CAN networks. Another issue is the limited computational power of ECUs. If we consider the lifetime of a vehicle, it is possible to crack a static encryption key. Therefore, dynamic key exchange is required. However, this is harder to implement and is computationally expensive. The dynamic key can also cause latency on resource-constrained ECUs, and it is not acceptable for safety-critical real-time systems.

The different encryption mechanisms proposed are shown in Table 2-3. Doan and Ganesan [41] implemented hardware-based AES-128 encryption on FPGA chips for the CAN system. The hardware implementation of the method decreases

latency and increases throughput. However, the method changes the legacy ECU and is not backwards compatible. Another study used Physical Unclonable Functions (PUFs) [42]. This method can obtain the private key from the physical characteristics of the ECUs; thus, hiding the key is not a problem. Although the method solves the problem of generating encryption keys, it also requires modifying the ECU.

Table 2-3 Encryption methods for the CAN bus.

Ref.	Encryption Method	Traffic Effect	Key
[41]	AES-128 and SHA-1	Increased	Static Symmetric
[43]	XOR	No Change	Dynamically Synchronised
[42]	AES-256 and Elliptic-curve Diffie Hellman	Increased	Symmetric
[44]	XOR	No Change	Static Symmetric
[45]	Tiny Encryption Algorithm	Increased	Static Symmetric
[46]	Triple DES	Increased	Dynamically Synchronised

Encryption hardens attacks and provides privacy; however, it is not sufficient to protect the CAN. Even the unbreakable encryption mechanism cannot prevent replay attacks.

2.5.3 Authentication

It is not possible to identify the sender of a CAN message. If an adversary has access to the network, they can send malicious messages and all the nodes accept them as authentic. This attack can be prevented via authentication.

VeCure [47] authentication, which has an acceptable 50 us processing delay, is based on trust groups where high-trust groups share a symmetric secret key. The method has a significant advantage with fewer key numbers, corresponding to the number of trust groups rather than the ECU number. However, it sends an

authentication message after every transmitted frame, which doubles the network traffic. Another drawback of the method is that it cannot protect the system if a node from the trust group is compromised. LiBrA-CAN [48], proposed by Groza et al., splits the authentication keys between groups of multiple nodes to improve efficiency. Although the method is quite successful, it requires high bandwidth and is not compatible with traditional CAN.

Nowdehi et al. [49] identified five criteria for an authentication method to be implemented commercially: cost-effectiveness, backward compatibility, support for vehicle repair and maintenance, sufficient implementation details, and acceptable overhead. They analysed ten authentication methods in the literature using them. Not surprisingly, none of the methods could pass all five criteria.

There are also off-the-shelf products providing hardware-based authentication like the S32K family from NXP [50]. The S32K family has Cryptographic Service Engine (CSE), a Cipher-based Message Authentication Code (CMAC) to provide secure authentication. It is a hardware-based system that accelerates the process drastically. For instance, public-key authentication can be achieved in less than 100 us [51] with hardware acceleration, while software authentication takes more than 10 ms, depending on the key size. However, the industry is currently concerned with the cost of ECUs. With the enhancement of hardware technology, it is possible to see more hardware-based methods to secure the CAN.

2.5.4 Intrusion Detection System (IDS)

Implementing security features on a safety-critical real-time system is a difficult task. Robust cryptographic methods are not feasible due to the limited resources (memory, bandwidth, and computational power) and time constraints. This leads to emerging research on intrusion detection system (IDS) for CAN. The main advantage of IDS is that it usually does not modify the current CAN controller, and the bus traffic does not increase.

Intrusion detection methods can be categorised as signature-based (misuse) detection and anomaly-based detection [52]. Signature-based detection checks

for known attacks on the database; therefore, it requires regular updates for new attacks. Although it is quite successful in detecting known attacks, it fails to detect unknown attacks. Anomaly-based IDS analyses the behaviour of the network and recognises the deviation from expected behaviour. Accuracy is usually lower than that of the signature-based. In contrast to signature-based detection, anomaly-based IDS may detect unknown attacks.

Table 2-4 Automotive anomaly detection sensors [53].

Sensor	**Description**
Formality	Correct message size, header and field size, field delimiters, checksum, etc.
Location	The message is allowed with respect to the dedicated bus system
Range	Compliance of payload in terms of data range
Frequency	Timing behaviour of messages is approved
Correlation	Correlation of messages on different bus systems adheres to the specification
Protocol	The correct order, start-time, etc. of internal challenge-response protocols
Plausibility	Content of message payload is plausible, no infeasible correlation with previous values
Consistency	Data from redundant sources is consistent

There are different parameters that an IDS system can assess on the CAN. Müter et al. [53] defined eight anomaly detection sensors, as shown in Table 2-4, to identify the anomalies in a structured way. All these detection sensors were inspired by the typical behaviour of the CAN bus. Deviation from these sensors' normal behaviour is the sign of an intrusion, and different IDS solutions use these sensors to detect intrusions. These solutions can be categorised as

time/frequency-based, physical system characteristic, specification-based, and feature-based.

2.5.4.1 Time/Frequency-Based IDS

Automobiles have rigid safety rules, and most of the ECUs transmit periodic signals. Any change in the frequency can be interpreted as abnormal behaviour, in other words, an intrusion. The basic IDS analyses the CAN messages' frequency as presented in [54][55].

Offset ratio and time interval based IDS [56], proposed by Lee et al., analyses the response time of the transmitted remote frame where the simple and effective algorithm can detect attacks and type of attacks; however, the method increases bus traffic by injecting remote frames for analyses.

The time/frequency analysis provides valuable information about the CAN. However, the vehicle's situation (e.g., idle, running) and the priority scheme of the CAN may significantly change the timing information and affect the result of time/frequency-based IDS. The method cannot detect attacks where the frequency is not changed, like a masquerade attack in [57].

2.5.4.2 Physical Characteristic Based IDS

The CAN network's physical characteristic may detect intrusions; hence, each transceiver has a different signal shape even though they transmit the same data. This can be caused by random manufacturing variations, cabling, and ageing.

In [58], Choi et al. proposed VoltageIDS, which uses unique electrical characteristics of the CAN signal like a fingerprint. The different locations of the ECUs with varying lengths of wire results in different resistance [59] and the resistance changes the signal features. They analysed eight signal features like positive and negative slope values and voltage values at a dominant level. The method has zero false-positive rates and can differentiate between attacks and errors; however, it requires an oscilloscope to gather the network signal and has heavy signal processing.

The CAN does not have a shared master clock, and each ECU uses its own quartz crystal. Cho and Shin [57] suggested the use of clock skew to detect intrusions. Although ECUs run the same frequency, they may have random drifting exceeding 2400 ms in a day [60]. They fingerprinted the transmitter ECU via the clock skew and detected the intrusions. Although they could reach 97% of the anomaly detection with a false-positive rate of 0.55%, the method only worked for the periodic messages. However, this method can be tricked by mimicking the clock skew, as shown in [61].

The physical characteristic of the CAN provides substantial information about ECUs. However, environmental factors like temperature and humidity and ageing of the components can change the physical characteristics; therefore, the IDS may fail. They can also not detect the attacks from the software layer because the authenticated ECU will transmit the malicious messages, and the IDS does not find any changes to the signal characteristics. Similarly, the physical characteristic-based IDS requires heavy signal processing. As a result, it may cause latency or require expensive hardware.

2.5.4.3 Specification-Based IDS

Larson et al. [62] suggested specification based attack detection and implemented specification rules based on the CAN Open protocol. This method has limited attack detection capability and requires all the ECUs to have detectors. The method is also not powerful enough to prevent attacks; hence, there are protocol-compliant attacks like [63].

Studnia et al. [64] proposed a language-based intrusion detection and derived the network's language characteristic from the ECUs' specifications and generated the forbidden sequences. If one of these sequences occurs, an intrusion is detected.

2.5.4.4 Feature-Based IDS

Feature-based system analysis examines the network parameters like busload, frequency, number of dropped messages, and other parameters like abnormal messages and payload. This is usually based on artificial intelligence techniques.

Generative Adversarial Nets (GAN) based IDS [65] was proposed by Seo et al., who used the deep-learning model. The method is easy to expand and difficult to manipulate by an attacker thanks to a black-box characteristic of the detection mechanism. Bloom filtering [66], proposed by Groza and Murvay, analysed the periodicity and payload of CAN messages. This method provides a memory-efficient analysis of data. Although both methods require heavy computation, they look promising in terms of tackling the CAN security problem.

Table 2-5 presents the comparison of the IDSs. Each method has a unique feature to suppress other methods but also comes with a cost. For example, physical characteristic-based IDS can easily detect an inauthentic node, but it fails to detect an attack from a software layer. The best IDS system should be a hybrid system that takes advantage of different methods. Although IDS can mitigate a security problem, it cannot provide confidentiality. To have complete security, cryptography is required.

Table 2-5 Comparison of the intrusion detection systems for CAN protocol.

Ref.	Algorithm Analyses	Parameters	Advantages	Downsides
[65]	Generative Adversarial Nets	A pattern of CAN ID	CAN train itself for unknown attacks	Expensive hardware
[67]	Adaptive Network-based Fuzzy Inference System	Busload, message frequency analysis	Detect attack type, simple solution	Works for simple attacks, updated each second, needs a feature database
[68]	Entropy-based	Entropy of IDs, payload	Does not require much information about traffic data	Very vulnerable to some attacks which include random bits
[69]	Long Short-term Memory Networks	Payload	Does not require pre-knowledge	Does not understand the natural change
[62]	Specification-based	Protocol policy	Less dependency	IDS should be placed at every ECU
[70]	Hamming Distance	Payload	Low computation	Low detection
[56]	Offset ratio and time interval	Remote frame timing	Simple efficient algorithm with low-cost hardware	Increased traffic

[71]	Analysis of ID Sequence	Sequence of ID	Low memory and computation requirement, detection of inserted few malicious messages	Very vulnerable to attacks which have a similar sequence of normal traffic
[58]	Support Vector Machine and Boosted Decision Tree	Electrical signal	Robust to some attack types, first IDS to differentiate between an error and an attack	High cost and vulnerability to environmental changes
[57]	Recursive Least Squares	Clock skew	Robust to some attack types,	Only works on periodic signals
[66]	Bloom Filtering	Message identifier, payload	Low memory usage for membership testing	Complex algorithm
[55]	Probability Density Function	Reception cycle period (frequency analysis)	Online learning	Hard to authenticate a non-periodic message
[54]	Flow-based	Message frequency	Simple algorithm	Only works on periodic signals

2.6 Discussions on CAN Security Research

Automotive security is getting more attention, and standardisations are coming to tackle cybersecurity problems. Cybersecurity guidebook for cyber-physical vehicle systems [72] and the fundamental principles of automotive cybersecurity specification (PAS 1885:2018) [73] were published by SAE in 2016 and British Standards Institute in 2018 consecutively. ISO 21434 Automotive Cybersecurity [74] is under development and expected to be released by 2020.

The CAN protocol has also gained attention from the industry to its vulnerabilities, and companies are now manufacturing high-end secure ECUs. The Secure Hardware Extension (SHE) [1] specification developed by the Hersteller Initiative (HIS) becomes an open standard and is used by many companies in their ECUs like NXP MPC5646C [51] microcontroller. Some commercial ECUs have built-in IDS; the NXP TJA115x [75] series can prevent spoofing attacks and be used as an IDS. There are also commercial proprietary intrusion detection systems [76],[77].

Although there have been steps taken to protect the CAN, there is still more to do. The industry does not share some of its research, and academia does not have enough resources. As such, there are not sufficient attack data and benchmarks. Implementing attacks on real vehicles can be unfeasible for safety concerns and costs. To overcome these challenges, there should be more research on modelling CAN bus attacks like in [78] and creating attack databases like in [79], [80]. Sharing datasets as an open-source (e.g., like in [65]) will help researchers; hence working on shared datasets will give a reference point to compare their research.

2.7 Conclusions

The CAN protocol facilitating ECUs in modern vehicles is not geared up and well-protected against the complex and evolving nature of cyberattacks. The existing security features incorporated in vehicles are not fit and adequate to resist and defy them. This is attributable to the lack of encryption and authentication mechanisms, which provide multiple opportunities for several types of attacks to

materialise and as a result, jeopardise the individual data privacy and the safety of the vehicle occupants. These blemish the manufacturers' reputation and downgrade vehicle reliability, followed by substantial financial losses.

It is observed that the existing trend of attacks is mainly physical-access oriented; however, with the growing connectivity in vehicles, it is also noted a considerable increase in wireless attacks. This developing trend indicates wireless attacks outpacing physical access attacks in the near future.

Moreover, an in-depth analysis of the CAN bus vulnerabilities to cyberattacks points to the limitations posed by the protocol. The root cause evaluation of various attacks and the critique of potential solutions have revealed the industry and academia's inadequacies and constraints. They are not driven toward mutual sharing of an attack database, allocating testing and trial resources, and developing benchmarks for an open-source.

There are four main countermeasures for CAN attacks: network segmentation, encryption, authentication, and IDS. They are, however, heavy on overheads with respect to the availability of the existing resources. Further analysis has revealed IDS as the most promising option compared to the rest of the solutions above-mentioned. It is noteworthy that the IDS may not provide complete security, but it can prevent several CAN vulnerabilities with acceptable overhead. It is presumed that future vehicles will have IDS solutions not only to secure the vehicle, but also to provide data to the manufacturer to tackle cyberattacks.

References

[1] P. Mundhenk, "Security for Automotive Electrical / Electronic (E / E) Architectures," Cuvillier Verlag, 2016.

[2] "'ECU' is a three letter answer for all the innovative features in your car: know how the story unfolded," *Embitel*, 2017. https://www.embitel.com/blog/embedded-blog/automotive-control-units-development-innovations-mechanical-to-electronics (accessed May 23, 2018).

[3] B. Groza and S. Murvay, "Security solutions for the Controller Area Network: Bringing Authentication to In-Vehicle Networks," *IEEE Vehicular Technology Magazine*, pp. 40–47, 2018.

[4] A. Palanca, E. Evenchick, F. Maggi, and S. Zanero, "A stealth, selective, link-layer denial-of-service attack against automotive networks," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Jul. 2017, vol. 10327 LNCS, pp. 185–206, doi: 10.1007/978-3-319-60876-1_9.

[5] S. Nie, L. Liu, and Y. Du, "Free-Fall : Hacking Tesla from wireless to CAN bus," in *BlackHat USA 2017*, 2017, pp. 1–16, Accessed: Nov. 30, 2018. [Online]. Available: https://www.blackhat.com/docs/us-17/thursday/us-17-Nie-Free-Fall-Hacking-Tesla-From-Wireless-To-CAN-Bus-wp.pdf.

[6] C. Matthews, "Jeep Hack: Fiat recalls 1.4 million vehicles for software fix," *Fortune*, 2015. https://fortune.com/2015/07/24/jeep-cherokee-recall/ (accessed Mar. 27, 2020).

[7] R. Currie, "Hacking the CAN Bus: Basic Manipulation of a Modern Automobile Through CAN Bus Reverse Engineering GIAC (GCIA) Gold Certification," 2017, Accessed: Jun. 03, 2018. [Online]. Available: https://www.sans.org/reading-room/whitepapers/threats/hacking-bus-basic-manipulation-modern-automobile-through-bus-reverse-engineering-37825.

[8] Vector Informatik, "Industry trends 2019: convergence drives competitiveness and innovation," Stuttgart, 2019. Accessed: Mar. 27, 2019. [Online]. Available: https://assets.vector.com/cms/content/consulting/publications/Ebert_IndustryTrends2019_Whitepaper.pdf.

[9] J. Liu, S. Zhang, W. Sun, and Y. Shi, "In-vehicle network attacks and countermeasures: Challenges and future directions," *IEEE Netw.*, vol. 31, no. 5, pp. 50–58, 2017, doi: 10.1109/MNET.2017.1600257.

[10] A. Greenberg, "Hackers remotely kill a jeep on the highway," *Wired.com*, 2015. https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/ (accessed Sep. 10, 2018).

[11] K. Koscher *et al.*, "Experimental security analysis of a modern automobile," in *Proceedings - IEEE Symposium on Security and Privacy*, 2010, pp. 447–462, doi: 10.1109/SP.2010.34.

[12] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive CAN networksPractical examples and selected short-term countermeasures," *Reliab. Eng. Syst. Saf.*, vol. 96, no. 1, pp. 11–25, Jan. 2011, doi: 10.1016/j.ress.2010.06.026.

[13] Tencent Keen Security Lab, "Experimental security assessment of BMW cars: A summary report," Shenzhen, 2018. Accessed: Nov. 30, 2018. [Online]. Available: https://keenlab.tencent.com/en/Experimental_Security_Assessment_of_BMW_Cars_by_KeenLab.pdf.

[14] S. Mukherjee, H. Shirazi, I. Ray, J. Daily, and R. Gamble, "Practical DoS attacks on embedded networks in commercial vehicles," 2016, vol. 10063, doi: 10.1007/978-3-319-49806-5.

[15] P. S. Murvay and B. Groza, "Security shortcomings and countermeasures for the SAE J1939 commercial vehicle bus protocol," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4325–4339, 2018, doi: 10.1109/TVT.2018.2795384.

[16] R. Bosch, "CAN specification version 2.0," 1991. Accessed: Jun. 04, 2018. [Online]. Available: http://esd.cs.ucr.edu/webres/can20.pdf.

[17] CSS Electronics, "CAN bus explained," *CCS Electronics*, 2019. https://www.csselectronics.com/screen/page/can-fd-flexible-data-rate-intro/language/en (accessed Jan. 31, 2020).

[18] Statista, "Automotive electronics cost as a percentage of total car cost worldwide from 1950 to 2030," 2018. Accessed: Jul. 22, 2018. [Online].

Available: https://www.statista.com/statistics/277931/automotive-electronics-cost-as-a-share-of-total-car-cost-worldwide/.

[19] S. Checkoway *et al.*, "Comprehensive experimental analyses of automotive attack surfaces," in *SEC'11 Proceedings of the 20th USENIX conference on Security*, 2011, pp. 6–6, doi: 10.1109/TITS.2014.2342271.

[20] S. Woo, H. J. Jo, and D. H. Lee, "A practical wireless attack on the connected car and security protocol for in-vehicle CAN," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 993–1006, 2015, doi: 10.1109/TITS.2014.2351612.

[21] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," 2014. Accessed: Mar. 31, 2018. [Online]. Available: https://www.ioactive.com/pdfs/IOActive_Remote_Attack_Surfaces.pdf.

[22] C. Beek and R. Samani, "DEFCON – Connected sar security," *McAfee*, 2017. https://securingtomorrow.mcafee.com/other-blogs/mcafee-labs/defcon-connected-car-security/ (accessed Aug. 18, 2019).

[23] I. Rouf *et al.*, "Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study.," *Proc. USENIX Secur. Symp.*, vol. 39, no. 4, pp. 11–13, 2010, doi: 10.1177/004057368303900411.

[24] T. Hoppe and J. Dittman, "Sniffing/Replay Attacks on CAN Buses: A simulated attack on the electric window lift classified using an adapted CERT taxonomy," in *Proceedings of the 2nd workshop on embedded systems security (WESS)*, 2007, pp. 1–6.

[25] B. Rebecca, "Proof-of-concept CarShark software hacks car computers, shutting down brakes, engines, and more," *Popular Science*. https://www.popsci.com/cars/article/2010-05/researchers-hack-car-computers-shutting-down-brakes-engine-and-more (accessed May 29, 2018).

[26] The National Cybersecurity and Communications Integration Center

(NCCIC), "CAN bus standard vulnerability | ICS-CERT," 2017. https://ics-cert.us-cert.gov/alerts/ICS-ALERT-17-209-01 (accessed Mar. 11, 2019).

[27] P.-S. Murvay and B. Groza, "DoS attacks on Controller Area Networks by fault injections from the software layer," in *Proceedings of the 12th International Conference on Availability, Reliability and Security - ARES '17*, 2017, vol. 10, pp. 1–10, doi: 10.1145/3098954.3103174.

[28] SAE International, "J1939: Serial control and communications heavy duty vehicle network," 2018. https://www.sae.org/standards/content/j1939_201808/ (accessed Dec. 29, 2019).

[29] Federal Bureau of Investigation, "Motor vehicles increasingly vulnerable to remote exploits," 2016. https://www.ic3.gov/media/2016/160317.aspx (accessed Aug. 05, 2019).

[30] A. Greenberg, "Hackers cut a Corvette's brakes via a common car gadget," *Wired*, 2015. https://www.wired.com/2015/08/hackers-cut-corvettes-brakes-via-common-car-gadget/ (accessed Aug. 05, 2019).

[31] I. Foster, A. Prudhomme, K. Koscher, and S. Savage, "Fast and Vulnerable: A story of telematic failures," 2015, Accessed: Aug. 05, 2019. [Online]. Available: http://cseweb.ucsd.edu/~savage/papers/WOOT15.pdf.

[32] G. Fowler, "Driving surveillance: What does your car know about you? We hacked a 2017 Chevy to find out. - The Washington Post," *Washingtonpost*, 2019. https://www.washingtonpost.com/technology/2019/12/17/what-does-your-car-know-about-you-we-hacked-chevy-find-out/ (accessed Mar. 19, 2020).

[33] M. Enev, A. Takakuwa, K. Koscher, and T. Kohno, "Automobile driver fingerprinting," in *Proceedings on Privacy Enhancing Technologies*, 2016, no. 1, pp. 34–51, doi: 10.1515/popets-2015-0029.

[34] U. Fugiglando, P. Santi, S. Milardo, K. Abida, and C. Ratti, "Characterising the 'driver DNA' through CAN bus data analysis," in *CarSys 2017 -*

Proceedings of the 2nd ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services, co-located with MobiCom 2017, 2017, vol. 17, pp. 37–41, doi: 10.1145/3131944.3133939.

[35] R. Kammerer, B. Frömel, and A. Wasicek, "Enhancing security in CAN systems using a star coupling router," in *7th IEEE International Symposium on Industrial Embedded Systems, SIES 2012 - Conference Proceedings*, 2012, pp. 237–246, doi: 10.1109/SIES.2012.6356590.

[36] Technische Universität München, “The car becomes internet hardware - TUM,” 2015. https://www.tum.de/nc/en/about-tum/news/press-releases/details/32277/ (accessed Nov. 21, 2019).

[37] J. Yoshida, "CAN Bus Can Be Encrypted , Says Trillium," *EEtimes*, 2015. https://www.eetimes.com/document.asp?doc_id=1328081&page_number=2 (accessed May 29, 2018).

[38] "CANcrypt - Home." https://www.cancrypt.eu/en/#Basics (accessed May 29, 2018).

[39] "2015 BMW F80 M3 / F82 M4 S55 inline-6 ecu flash dyno results from Jailbreak Tuning," *BimmerBoost*, 2014. https://www.bimmerboost.com/content.php?5101-2015-BMW-F80-M3-F82-M4-S55-inline-6-ecu-flash-dyno-results-from-Jailbreak-Tuning (accessed Aug. 05, 2019).

[40] R. Jurnecka, "Cobb Tuning cracks Nissan GT-R's encrypted ECU - MotorTrend," *Motortrend*, 2008. https://www.motortrend.com/news/cobb-tuning-cracks-nissan-gtrs-encrypted-ecu-308/ (accessed Jul. 01, 2019).

[41] T. P. Doan and S. Ganesan, "CAN crypto FPGA chip to secure data transmitted through CAN FD bus using AES-128 and SHA-1 algorithms with a symmetric key," 2017, doi: 10.4271/2017-01-1612.Copyright.

[42] A. S. Siddiqui, Y. G. J. Plusquellic, and F. Saqib, "Secure communication over CANBus," in *Midwest Symposium on Circuits and Systems*, 2017, vol.

2017-Augus, pp. 1264–1267, doi: 10.1109/MWSCAS.2017.8053160.

[43] A. Harel and A. Hezberg, "Optimizing CAN bus security with in-place cryptography," in *SAE Connected and Automated Vehicle Conference Israel*, 2019, pp. 1–12, doi: 10.4271/2019-01-0098.

[44] W. A. Farag, "CANTrack: Enhancing automotive CAN bus security using intuitive encryption algorithms," 2017, doi: 10.1109/ICMSAO.2017.7934878.

[45] M. Jukl and J. Čupera, "Using of tiny encryption algorithm in CAN-Bus communication," *Res. Agric. Eng.*, vol. 62, no. 2, pp. 50–55, 2016, doi: 10.17221/12/2015-RAE.

[46] A. Hanacek and M. Sysel, "Design and implementation of an integrated system with secure encrypted data transmission," in *Advances in Intelligent Systems and Computing*, Apr. 2016, vol. 466, pp. 217–224, doi: 10.1007/978-3-319-33389-2_21.

[47] Q. Wang and S. Sawhney, "VeCure: A practical security framework to protect the CAN bus of vehicles," in *2014 International Conference on the Internet of Things, IOT 2014*, 2014, pp. 13–18, doi: 10.1109/IOT.2014.7030108.

[48] B. Groza, S. Murvay, A. Van Herrewege, and I. 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)*, Dec. 2012, vol. 7712 LNCS, pp. 185–200, doi: 10.1007/978-3-642-35404-5_15.

[49] N. Nowdehi, A. Lautenbach, and T. Olovsson, "In-vehicle CAN message authentication: An evaluation based on industrial criteria," in *IEEE Vehicular Technology Conference*, 2017, vol. 2017-Septe, pp. 1–7, doi: 10.1109/VTCFall.2017.8288327.

[50] NXP, "32-bit automotive general purpose MCUs," *NXP*.

https://www.nxp.com/products/processors-and-microcontrollers/arm-processors/s32-automotive-platform/32-bit-automotive-general-purpose-microcontrollers:S32K (accessed Aug. 14, 2019).

[51] R. Soja, "Automotive Security: From standards to implementation," 2014. Accessed: Aug. 14, 2019. [Online]. Available: https://www.nxp.com/docs/en/white-paper/AUTOSECURITYWP.pdf.

[52] K. Scarfone and P. Mell, *Guide to Intrusion Detection and Prevention Systems (IDPS)*, vol. 800–94, no. July. 2012.

[53] M. Müter, A. Groll, and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in *6th International Conference on Information Assurance and Security, IAS 2010*, 2010, pp. 92–98, doi: 10.1109/ISIAS.2010.5604050.

[54] A. Taylor, N. Japkowicz, and S. Leblanc, "Frequency-based anomaly detection for the automotive CAN bus," in *World Congress on Industrial Control Systems Security (WCICSS)*, 2015, pp. 45–49, doi: 10.1109/WCICSS.2015.7420322.

[55] Y. Hamada, Y. Miyashita, Y. Hata, M. Inoue, and H. Ueda, "Anomaly-based intrusion detection using the density estimation of reception cycle periods for in-vehicle networks," *SAE Int. J. Transp. Cybersecurity Priv.*, vol. 1, no. 1, pp. 39–56, 2018, doi: 10.4271/11-01-01-0003.

[56] H. Lee, S. H. Jeong, and H. K. Kim, "OTIDS : A novel intrusion detection system for in-vehicle network by using remote frame," 2017, [Online]. Available: https://www.ucalgary.ca/pst2017/files/pst2017/paper-67.pdf%0Ahttp://ocslab.hksecurity.net/Dataset/CAN-intrusion-dataset.

[57] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *25th USENIX Security Symposium*, 2016, pp. 911–927, Accessed: May 31, 2018. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/cho.

[58] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "VoltageIDS: Low-level communication characteristics for automotive intrusion detection system," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, 2018.

[59] K.-D. Kang, Y. Baek, S. Lee, and S. H. Son, "An analysis of voltage drop as a security feature in Controller Area Network," 216AD, Accessed: Mar. 25, 2019. [Online]. Available: https://pdfs.semanticscholar.org/26c7/ae0d12e7f4dd3a6b8fe6a049e7883608d004.pdf.

[60] S. Mohalik *et al.*, "Model checking based analysis of end-to-end latency in embedded, real-time systems with clock drifts," in *Proceedings - Design Automation Conference*, 2008, pp. 296–299, doi: 10.1109/DAC.2008.4555826.

[61] S. U. Sagong, X. Ying, A. Clark, L. Bushnell, and R. Poovendran, "Cloaking the Clock: emulating clock skew in Controller Area Networks," 2017, doi: 10.1109/ICCPS.2018.00012.

[62] U. E. Larson, D. K. Nilsson, and E. Jonsson, "An approach to specification-based attack detection for in-vehicle networks," in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2008, pp. 220–225, doi: 10.1109/IVS.2008.4621263.

[63] W. Si, D. Starobinski, and M. Laifenfeld, "Protocol-compliant DoS attacks on can: Demonstration and mitigation," 2016, doi: 10.1109/VTCFall.2016.7881182.

[64] I. Studnia, E. Alata, V. Nicomette, M. Kaâniche, and Y. Laarouchi, "A language-based intrusion detection approach for automotive embedded networks," *Int. J. Embed. Syst.*, vol. 10, no. 1, pp. 1–12, 2018, doi: 10.1504/IJES.2018.089430.

[65] E. Seo, H. M. Song, and H. K. Kim, "GIDS: GAN based Intrusion Detection System for In-Vehicle Network," in *2018 16th Annual Conference on*

Privacy, Security and Trust (PST), Aug. 2018, pp. 1–6, doi: 10.1109/PST.2018.8514157.

[66] B. Groza and P. Murvay, "Efficient intrusion detection with bloom filtering in Controller Area Networks (CAN)," *IEEE Trans. Inf. Forensics Secur.*, vol. PP, no. c, p. 1, 2018, doi: 10.1109/TIFS.2018.2869351.

[67] F. Li, L. Wang, and Y. Wu, "Research on CAN network security aspects and intrusion detection design," SAE, Sep. 2017. doi: 10.4271/2017-01-2007.

[68] M. Muter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, no. IV, pp. 1110–1115, doi: 10.1109/IVS.2011.5940552.

[69] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in *Proceedings - 3rd IEEE International Conference on Data Science and Advanced Analytics, DSAA 2016*, 2016, pp. 130–139, doi: 10.1109/DSAA.2016.20.

[70] D. Stabili, M. Marchetti, and M. Colajanni, "Detecting attacks to internal vehicle networks through Hamming distance," in *2017 AEIT International Annual Conference*, Sep. 2017, pp. 1–6, doi: 10.23919/AEIT.2017.8240550.

[71] M. Marchetti and D. Stabili, "Anomaly detection of CAN bus messages through analysis of ID sequences," in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2017, pp. 1577–1583, doi: 10.1109/IVS.2017.7995934.

[72] Vehicle Cybersecurity Systems Engineering Committee, "J3061 - Cybersecurity guidebook for cyber-physical vehicle systems." p. 128, 2016, doi: 10.4271/J3061_201601.

[73] BSI, *PAS 1885:2018 The fundamental principles of automotive cyber security - Specification*. BSI Standards Limited, 2018.

[74] ISO, *ISO/SAE CD 21434 - Road Vehicles -- Cybersecurity engineering*. 2019.

[75] N. Semiconductors, "TJA115x Secure CAN communication without cryptography," 2019. Accessed: Jan. 31, 2020. [Online]. Available: www.nxp.com/CAN.

[76] "ECUSHIELD - The only Proven Ready for Integration Automative Cyber Security Solution." http://tower-sec.com/ecushield/ (accessed Mar. 30, 2018).

[77] "Argus Cyber Security - automotive cyber security." https://argus-sec.com/ (accessed Mar. 30, 2018).

[78] S. Fröschle and A. Stühring, "Analysing the capabilities of the CAN Attacker," in *European Symposium on Research in Computer Security*, Sep. 2017, vol. 10492 LNCS, pp. 464–482, doi: 10.1007/978-3-319-66402-6_27.

[79] M. Ring, J. Dürrwang, F. Sommer, and R. Kriesten, "Survey on vehicular attacks - Building a vulnerability database," in *2015 IEEE International Conference on Vehicular Electronics and Safety, ICVES 2015*, 2015, pp. 208–212, doi: 10.1109/ICVES.2015.7396919.

[80] T. Huang, J. Zhou, and A. Bytes, "ATG: An attack traffic generation tool for security testing of in-vehicle CAN bus," in *ACM International Conference Proceeding Series*, 2018, vol. 6, doi: 10.1145/3230833.3230843.

3 A framework and Comprehensive Benchmarking Dataset for CAN Bus Intrusion Detection Systems

This chapter proposes a framework to assess an IDS and the generation of benchmarking dataset. IDS is becoming the primary choice to address CAN's vulnerabilities; however, a lack of testing methodology prevents assessing IDS properly. The testing framework proposed in this chapter presents performance evaluation metrics for quantitative evaluation with required test conditions, including various attack types and dependency tests. As vehicles are resource-constrained cyber-physical systems, resource usage is also considered in the assessment. The second part of the chapter focuses on attack generation for benchmarking dataset, which is an essential part of successful testing. Various attack scenarios are implemented according to the proposed testing framework. The organisation of the chapter is presented in Figure 3-1.

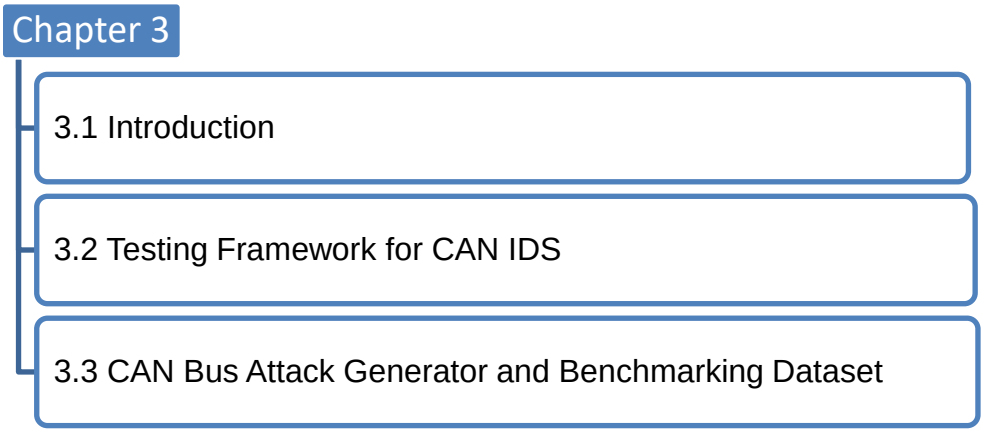

Figure 3-1 Organisation of Chapter 3

3.1 Introduction

Security of the connected vehicle is a significant concern and researchers are looking for potential solutions. The most widely used in-vehicle communication protocol, Controller Area Network (CAN), lacks intrinsic security features like encryption and authentication; thus, vehicles are vulnerable to cyberattacks. Researchers proposed various Intrusion Detection Systems (IDSs) to identify

intrusions and secure the CAN network. Although the research on IDS for CAN bus is rising exponentially, there is no accepted testing methodology and enough dataset. The field is still in its infancy and produced works lacks comprehensive evaluation and comparative analysis. Blevins et al. [1] compared four time-based IDS by implementing each method on the same dataset. The work presents the difficulty of comparing methods that use the same parameter. Berger et al. [2] evaluated the machine learning algorithms to detect anomalies in CAN. However, the evaluation was not comprehensive as the work only relies on accuracy, which alone is not reliable. A 2018 survey [3] shows that only four out of the 65 surveyed papers compare their works with baseline methods. Consequently, it becomes difficult to verify and compare existing IDS solutions.

The testing should be objective and repeatable with quantifiable metrics to verify and validate an IDS. Therefore, the framework in this chapter starts with the quantitative performance metrics. This allows easy comparison between the various methods. After that, comprehensive test conditions are presented to assess dependency and attack coverage. Lastly, the framework focuses on two critical parameters resource usage and timing behaviour.

The viable IDS assessment should be carried out on a comprehensive dataset with all the variations, including attack types, vehicle models, driving styles. However, such a data set does not exist. The lack of a publicly available benchmarking dataset costs time, effort, and money. There are a few reasons why there is no comprehensive dataset. First, the implementation of attacks on a vehicle requires a carefully designed testing environment. Some of the attacks [4] can only be carried out on closed roads (like de-commissioned airport runway) under high safety measures that still lack the real traffic environment. Second, it is not always possible to find these datasets because of Intellectual Property (IP) rights, commercial issues for brands. Although the researchers share very few datasets, these datasets have numerous problems [5]. This pushes researchers to use a proprietary dataset that prevents comparison between methods because generally, attack implementation is not parameterised and each vehicle has different network characteristics.

This chapter has two main objectives: comprehensive testing methodology and CAN bus attack generation to create benchmarking dataset. At first, the chapter presents performance evaluation metrics to have a quantitative testing methodology. Then it shows various test conditions to have a reliable result. These conditions measure the attack coverage, dependency (to vehicle, driver, and ECU), resource usage (memory and computational power), and timing behaviour. Later, the chapter then focuses on the generation of CAN bus attacks aligned with the testing methodology. The dataset is an essential part of the testing. If it is not high quality, the testing will fail, and the result will be unreliable.

3.2 Testing Framework for CAN IDS

It is vital to test an IDS to see its capabilities and limitation. Proper testing not only presents the method works or not, but it also presents how well the method works. However, there is no standardised or commonly used testing methodology for automobile IDS solutions. This causes two main problems. First, many IDS solutions are not appropriately tested, which generates misleading results. The second problem is that it is difficult to compare existing solutions which hardens researchers' tasks to evaluate their methodologies. It also hinders reaching the best IDS.

This chapter presents a repeatable and objective testing methodology to solve these problems. Before testing an IDS, it is essential to have quantitative evaluation metrics so that various methods can be objectively compared. As vehicles are real-time cyber-physical systems with limited resources, the performance evaluation alone is not sufficient. Each task should be completed by the deadline to meet real-time constraints by budget microcontrollers, some of which have only an 8-bit controller and a few kbytes of memory. In this condition, it is essential to analyse the timing behaviour and resource usage of the IDS in addition to the correctness of the IDS decision.

After identifying evaluation metrics, an IDS should be tested on datasets, covering all known attacks with various implementation settings. This will increase the reliance on the IDS.

3.2.1 Performance Evaluation Metrics

The main aim of the IDS is to alert on intrusion; that is what they are designed for. At the same time, the IDS should not generate an alarm for authentic messages. Traditionally, accuracy is used to measure performance; however, accuracy alone may lead to misinterpretations, especially when the data is not symmetrical. Therefore, we need a more comprehensive assessment. If we have a binary classifier for the IDS, there are four possible outcomes of the system as presented in Table 3-1 and explained below:

True Positive (TP): A malicious message detected correctly.

False Positive (FP): An authentic message is detected as a malicious message.

True Negative (TN): An authentic message is detected correctly.

False Negative (FN): A malicious message is regarded as an authentic message.

Table 3-1 Confusion matrix for binary IDS decision

		Actual Situation	
		Attack	No Attack
IDS Decision	Attack	TP	FP
	No Attack	FN	TN

Using this terminology, the sensitivity (recall) of the IDS, which presents the detection rate of the attacks, can be defined as in Equation(3-2). False Positive Rate (FPR) in Equation (3-3) indicates the likelihood of false alarms. Precision in Equation (3-4) presents how reliable the true positive result of the IDS is.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad \textbf{(3-1)}$$

$$\text{Sensitivity} = \frac{TP}{TP + FN} \quad \textbf{(3-2)}$$

$$\text{FPR} = \frac{FP}{TN + FP} \quad \textbf{(3-3)}$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad \textbf{(3-4)}$$

3.2.2 Attack Coverage

There are multiple attack types applied to the in-vehicle networks. Suppose an attacker has direct access to the CAN bus. In that case, they can read and write to the CAN network, proceed with overwriting and invalidating legitimate messages, and further disable a CAN node. In this circumstance, the following attacks are possible to implement:

Denial of Service (DoS): DoS attack can be carried out by taking advantage of the CAN arbitration scheme (as described in Section 2.2). Inserting high priority messages will hold the bus in busy condition and bar the lower priority nodes from message transmission; therefore, other low-priority nodes cannot access the network. Murvay presented an example of this attack in [6].

Drop/suspension: The suspension attack prevents legitimate message transmission by attacking the targeted ECU on the physical or software layers. It is a subcategory of the DoS attack; hence, it is not possible to get service from the suspended node. The attack can also be implemented by causing protocol error which takes advantage of the Error Confinement Mechanism as shown by Palanca et al. [7]. They disabled an ECU by transmitting dominant bits over the recessive ones, which infracts the bit-stuffing rule explained in Section 2.2.1.

Fuzzing: An attacker can send random values without any in-depth knowledge and confuse the network. This attack does not require reverse engineering to understand the function of each ID; therefore, it is easy to implement. However,

it may not cause a functional problem apart from busload; hence inserting fuzzing messages will increase the frequency of the CAN message like the one in [8].

Replay: An attacker can monitor the CAN messages and send them back to the network later on, such as [9]. Hence, there is no freshness check on the protocol; other nodes will accept the replayed message. This attack type generally targets certain ID/s; therefore, it is also called a targeted ID attack.

The replay attack does not require modification of authentic ECU, so it can be implemented by any malicious node that uses authentic ECU ID. This will lead to concurrent message transmission from both authentic and malicious ECUs. As ECUs accepts the latest messages, the attack will cause unstable system behaviour. If an attacker wants to transmit certain data to be processed, it should have an equal or higher frequency value than the authentic messages. The stealthiest version of replay attack requires malicious message transmission soon after the legitimate message.

The coverage test determines which attacks can be detected by the IDS under test. A successful IDS ideally should detect all the attacks mentioned above. Each attack has different behaviour; therefore, IDS should be tested in each of them to assess IDS's coverage.

3.2.3 Dependency Test

The characteristic of the in-vehicle network depends on various parameters like the driver, ECU/ID, vehicle make and model. Each parameter should be assessed to present the viability of the tested methodology.

3.2.3.1 Vehicle Dependency

Each vehicle make and model has different behaviour by design. For instance, there can be a few dozen ECUs for low-end vehicles. However, the number can go further than one hundred for high-end ones. The number of ECUs significantly change the network traffic. Apart from the number, ECU types and their configuration also affect traffic. This causes various network traffic patterns for different vehicles.

3.2.3.2 Driver Dependency

When people drive the same vehicle model, each may generate different traffic because of their driving style, affecting the ECU communication. This effect is so significant that the driver can be recognised from the network traffic [10], [11]. As this significant change may affect the IDS result, IDS should be tested for driver dependency. The test can be carried out using the exact vehicle with different drivers and implementing various attack conditions.

3.2.3.3 ECU/ID Dependency

ECUs have various configurations to transmit messages. Some transmit periodically, some are event-driven, and others are sending based on other CAN frames. The configuration may change based on the vehicle situation (running or idle), affecting the bus traffic. Each ECU also has different prioritisation represented by ID number. If a driver activates an ECU with high priority, this may delay the transmission of the low priority IDs. Therefore, this test is essential for the time/frequency-based IDS.

3.2.4 Timing Analysis

There are two main parameters for timing analysis. These are time-to-detection and processing time. The Time-To-Detection (TTD), the time difference between the attack start and the time that the algorithm detects an attack [12] calculated by the following formula:

$$t_{TTD} = t_D - t_s \tag{3-5}$$

where t_{TTD} is TTD, t_s is the time attack started, and t_D is the time the algorithm detected the attack. The average of TTDs results in a key performance indicator Mean Time-To-Detect (MTTD) calculated by the following formula:

$$MTTD = \frac{1}{n}\sum_{k=1}^{n} t_{TTD}(k) \tag{3-6}$$

It is vital to prevent misinformation from spreading and causing disruptions in the vehicle; therefore, a successful IDS should have low MTTD.

The processing time is also an important parameter to evaluate timing behaviour. Contrary to TTD, which is related to the algorithm and varies by threshold and other parameters of the IDS, the processing time depends on the hardware and can be decreased by optimising computational logic. As ECUs have quite different processors/controllers than personal computers, actual processing time requires implementing the IDS algorithm on an ECU.

3.2.5 Resource Usage

Vehicles are resource-constrained cyber-physical systems. Distributed ECUs have limited memory, computational power, and bandwidth. Therefore, optimum IDS should have low resource usage.

3.2.5.1 Traffic Effect (Bus Load)

Electric vehicles and autonomous vehicles are the future of the vehicle industry, and those vehicles require higher bandwidth. However, the CAN bus standard is limited to 1 Mbit/s, and the speed gets significantly slower for low-speed CAN (ISO 11898-3). With the increased number of ECUs and sensors, the busload becomes a problem to have reliable communication. CAN-FD[13], which has up to eight times payload speed improvement, is proposed to overcome this bottleneck. However, it comes with an increased hardware cost, various signalling and network topology problems [14]. The arbitration process and other control bits are still limited to standard CAN because of the backwards compatibility, which curbs the effective bus speed. Therefore, the bandwidth is precious, and a successful IDS should not increase the traffic, which causes too much busload. If an IDS transmits CAN frames to function like in [15], it can increase traffic, delaying the message transmission from lower IDs.

3.2.5.2 Memory Usage and Computational Power

The ECUs are responsible for various applications; therefore, there is a wide range of microcontrollers from 8-bit single-core to 32-bit multicore architecture. Although there are powerful ECUs, most of the ECUs in the vehicle have low-end processors/microcontrollers with limited memory and computational power because of the cost (e.g. 9S08SG4[16] which has 4kb memory). Therefore, it is

not feasible to design an IDS that demand high memory and computational power. Resource usage is especially significant for node-based IDS as it requires each node to have the algorithm running, whereas network-based IDS requires only the gateway to run the algorithm.

3.3 CAN Bus Attack Generator and Benchmarking Dataset

An essential step for developing an IDS is to test it on comprehensive datasets on vehicles considering various working conditions, attack models, vehicles made, and driving styles. The collection of comprehensive datasets requires running a testing vehicle equipped with measurement instruments on dedicated roads where safety measures are taken. It is a challenging task and no publicly available dataset presents all the variations. As a result, there is no standardised benchmarking data set and the issue is raised by researchers [17]. There are very few datasets available as open-source, as summarised in Table 3-2. Although these datasets are very valuable, they have limitations and artefacts [5].

Table 3-2 Summarisation of open access CAN bus dataset for IDS

Dataset	Implementation Type	Advantages	Disadvantages
OTIDS [15]	Real vehicle	Stealthy attacks	Additional remote frames, no label
Survival [18]	Real vehicle	Labelled attacks on multiple vehicles	Inadequate data samples, abrupt change
CANET [17]	Simulation	Signal attack, stealthy attacks	Suitable only for signal-based IDS
Car-hacking [19]	Real vehicle	Labelled extended dataset with comprehensive attack coverage	Gaps and artefacts in the data
Intrusion dataset [20]	Simulation	Clear definition of attack methodology with comprehensive attack coverage, multiple vehicles	Only one attacks instance for each attack type

The most widely used datasets are published by Hacking & Countermeasures Research Lab (HCRL). CAN Dataset for intrusion detection (OTIDS) [15] has stealthy attacks, including DoS, fuzzy, and impersonation attacks. The dataset has a good amount of data with 250 seconds of attack free then attacks are implemented. Unfortunately, the dataset does not have a label for malicious frames. There are also a significant number of remote frames inserted to detect anomalies; therefore, this dataset may behave unexpectedly for some algorithms. Survival analysis dataset for automobile IDS has five seconds of injected malicious CAN frames every 20 seconds on three different commercial vehicles. Therefore, this dataset is suitable for the vehicle dependency test. However, the implementation of the attacks is causing abrupt changes, and it has a short duration of attack-free data, which limits the training. Car-hacking dataset [19] includes the well-recognised attack models, including Denial of Service (DoS), spoofing, and fuzzy attacks. It is a very long dataset that provides sufficient attack-free data and many attack instances that last 3-5 seconds. Although the dataset has artefacts of gaps [2] in the data and attacks are implemented during vehicle was stationary [5], the research community well deserves this dataset. It has already been cited for many different research pieces.

3.3.1 Attack Generation

One of the downsides of datasets is the limited number of attacks instances (usually only one) for each attack model, which is insufficient for testing IDS' capabilities. The limited test data also causes an overtraining problem, resulting in significantly poor results on different datasets. To have a reliable dataset, we studied the CAN traffic for vehicular applications and explored realistic traffics under various driving scenarios and attack models to get a view for intentionally extending the initial datasets to the one that covers more comprehensive attack datasets. The methodology to implement each attack type is as follow:

DoS Attack: The DoS attack can be implemented by inserting high priority CAN frames. The available datasets implement the DoS attack by inserting messages that belong to the most priority ID, "0000". However, this can be detected easily by checking the message IDs. Another downside of these datasets is they have

an obvious implementation of DoS attack with a long attack duration. The obvious attack implementation causes improper testing of IDSs.

We implemented comprehensive DoS attacks. The attacks include traditional implementation with varying attack duration as well as a stealthier performance with existing IDs in the CAN network. The attacks were implemented for a period of 0.25s, 0.5s, and from 1s to 5s with a step size of 1s.

Replay Attack: We implemented replay attacks with various attack strengths and durations. The inserted malicious message' frequency increases step by step from the attacked node's base frequency to multiple times faster than the base frequency for the replay attack.

While implementing replay attacks, we also consider the CAN arbitration scheme and message timing to get as close as possible to real implementation. For instance, the algorithm checks the time difference between the targeted ID and the following message. If the gap between these two messages is lower than the inserted message transmission time, it will consider an arbitration scheme. The highest priority ID will transmit, and the lower one will be shifted.

Fuzzy Attack: Two methodologies can be used to implement fuzzy attacks. One is implementing random CAN messages from random IDs, which is a simple attack to detect with an ID detector. The second attack is executed with random messages from only existing IDs. This version is stealthier than the previous one. Like DoS attack, fuzzy attacks are implemented for a duration of 0.25s, 0.5s, and from 1s to 5s with a step size of 1s.

Suspension Attack: The suspension attack can be carried out by deleting the messages related to a particular ID. This action will be like when a node is silenced; however, it ignores the relationship between the ECUs. The attack can be implemented with various IDs to test ID/ECU dependency. The suspension attack is executed for multiple attack durations, similar to other attack types.

3.3.2 Implementation

The attack generation method presented in Section 3.3.1 is a framework to generate a structured benchmarking dataset. It can be used to implement attacks on existing datasets to create synthetic attacks or a methodology to follow while injecting the attacks into a running vehicle. As there are difficulties in executing an attack on an actual vehicle, the benchmarking dataset is generated in a simulation environment on Matlab.

The open-source datasets from two independent research centres [19], [20], which are well deserved by the research community and have already been cited for many different research pieces, are used in the experiment. The Car-hacking dataset [19] is used without any changes, and the other is used to generate multiple synthetic attacks. Having multiple datasets allows us to develop a benchmarking dataset consisting of numerous vehicle models and driving styles while avoiding manipulation for any dataset.

The open-source datasets contain CAN frames with a timestamp in CSV file format. The raw data is imported to Matlab, and the code in Appendix A.1 inserts malicious messages for DoS, replay, and fuzzy attacks according to the methodology presented in Section 3.3.1. The Matlab code also deletes messages belonging to the attacked node for suspension attack.

3.3.3 Benchmarking Dataset

By applying the technique to the Automotive Controller Area Network (CAN) bus intrusion dataset v2 [20], consisting of real CAN traffic from two commercially available vehicles, comprehensive synthetic attacks are generated, as shown in Table 3-3. The resulted synthetic attacks allow us to see the capabilities and limitations of the IDS by testing it on different attack scenarios. The synthetic attacks summarised in Table 3-3 are more challenging to detect than the original attacks because attack durations are shorter, and the traffic effect is minimal. It also provides more attack episodes to have a statistical result.

Table 3-3 Generated synthetic attacks based on Automotive Can Bus Intrusion Dataset v2

Data Source	Attack Type	# of Messages	Malicious Messages	Attack Duration
Vehicle 1	No attack	2690069	-	-
Vehicle 2		386567	-	-
Vehicle 1	Denial of Service	806999	4000 - 40000	1s to 10 s
Vehicle 2		115971	4000 - 40000	1s to 10 s
Vehicle 1	Suspension	806999	50 – 500 [a]	1s to 10 s
Vehicle 2		115971	40 – 400 [a]	1s to 10 s
Vehicle 1	Replay	806999	8-30	75 ms
Vehicle 2		115971	8-30	66 ms

[a] Number of maliciously deleted messages.

Apart from the synthetic attacks, it is still important to use a real attack dataset to test the algorithm in a real-world scenario where the targeted vehicle is running. Although synthetic attacks mimic the real ones, they cannot mimic the knock-on effect, which may slightly affect the results. The Car-hacking dataset [19], one of the most widely used dataset, can be used. As the dataset is used in many IDS research, it also helps to have comparative analysis with other research. As summarised in Table 3-4, the dataset has data from an actual vehicle while message injection attacks were performed. DoS attack was implemented by injecting the highest priority CAN messages, while fuzzing attack was executed by random CAN ID and payload values. A spoofing attack was implemented by inserting malicious messages on relevant CAN IDs for Gear and RPM.

Table 3-4 Car-Hacking dataset from real vehicle attack.

Attack Type	# of Messages	Malicious Messages	Attack Duration
DoS Attack	3665771	587521	3s to 5s
Gear Spoofing	4443142	597252	3s to 5s
RPM Spoofing	4621702	654897	3s to 5s
Fuzzing Attack	3838860	491847	3s to 5s

References

[1] D. H. Blevins, P. Moriano, R. A. Bridges, M. E. Verma, M. D. Iannacone, and S. C. Hollifield, “Time-Based CAN Intrusion Detection Benchmark,” 2021, doi: 10.14722/autosec.2021.23013.

[2] I. Berger, R. Rieke, M. Kolomeets, A. Chechulin, and I. Kotenko, “Comparative study of machine learning methods for in-vehicle intrusion detection,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11387 LNCS, Springer Verlag, 2019, pp. 85–101.

[3] G. K. Rajbahadur, A. J. Malton, A. Walenstein, and A. E. Hassan, “A Survey of Anomaly Detection for Connected Vehicle Cybersecurity and Safety,” in *IEEE Intelligent Vehicles Symposium*, 2018, vol. 2018-June, pp. 421–426, doi: 10.1109/IVS.2018.8500383.

[4] K. Koscher *et al.*, “Experimental security analysis of a modern automobile,” in *Proceedings - IEEE Symposium on Security and Privacy*, 2010, pp. 447–462, doi: 10.1109/SP.2010.34.

[5] M. E. Verma, M. D. Iannacone, R. A. Bridges, S. C. Hollifield, B. Kay, and F. L. Combs, “ROAD: The Real ORNL Automotive Dynamometer Controller Area Network Intrusion Detection Dataset (with a comprehensive CAN IDS dataset survey & guide),” 2020. Accessed: Apr. 17, 2021. [Online].

Available: http://energy.gov/downloads/doe-public-access-plan.

[6] P.-S. Murvay and B. Groza, “DoS attacks on Controller Area Networks by fault injections from the software layer,” in *Proceedings of the 12th International Conference on Availability, Reliability and Security - ARES ’17*, 2017, vol. 10, pp. 1–10, doi: 10.1145/3098954.3103174.

[7] A. Palanca, E. Evenchick, F. Maggi, and S. Zanero, “A stealth, selective, link-layer denial-of-service attack against automotive networks,” in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Jul. 2017, vol. 10327 LNCS, pp. 185–206, doi: 10.1007/978-3-319-60876-1_9.

[8] H. Lee, K. Choi, K. Chung, J. Kim, and K. Yim, “Fuzzing CAN packets into automobiles,” in *International Conference on Advanced Information Networking and Applications, AINA*, 2015, vol. 2015-April, pp. 817–821, doi: 10.1109/AINA.2015.274.

[9] T. Hoppe, S. Kiltz, A. Lang, and J. Dittmann, “Exemplary Automotive Attack Scenarios: Trojan Horses for Electronic Throttle Control System (ETC) and Replay Attacks on the Power Window System,” in *Proc. Autom. Security VDI-Berichte Nr. VDI/VW Gemeinschaftstagung Autom. Security*, 2007, pp. 165–183, Accessed: Jun. 21, 2019. [Online]. Available: https://pdfs.semanticscholar.org/d10a/558b9caa8bd41c0112434f3b19eb8aab2b5c.pdf.

[10] B. Il Kwak, J. Y. Woo, and H. K. Kim, “Know your master: Driver profiling-based anti-theft method,” in *14th Annual Conference on Privacy, Security and Trust*, 2016, pp. 211–218, doi: 10.1109/PST.2016.7906929.

[11] U. Fugiglando, P. Santi, S. Milardo, K. Abida, and C. Ratti, “Characterizing the ‘driver DNA’ through CAN bus data analysis,” in *CarSys 2017 - Proceedings of the 2nd ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services, co-located with MobiCom 2017*, 2017, vol. 17, pp. 37–41, doi: 10.1145/3131944.3133939.

[12] Q. Lin, S. Verwer, R. Kooij, and A. Mathur, “Using datasets from industrial control systems for cyber security research and education,” in *International Conference on Critical Information Infrastructures Security*, 2019, no. October, pp. 122–133, doi: 10.1007/978-3-030-37670-3_10.

[13] Bosh GmbH, “CAN FD | Bosch Semiconductors.” http://www.bosch-semiconductors.com/ip-modules/can-ip-modules/can-fd/ (accessed Mar. 21, 2019).

[14] NXP Semiconductors, “TJA1462 CAN Signal Improvement,” 2020. Accessed: Apr. 23, 2021. [Online]. Available: https://www.nxp.com/docs/en/white-paper/CANSIGIMP-WP.pdf.

[15] H. Lee, S. H. Jeong, and H. K. Kim, “OTIDS : A novel intrusion detection system for in-vehicle network by using remote frame,” 2017, [Online]. Available: https://www.ucalgary.ca/pst2017/files/pst2017/paper-67.pdf%0Ahttp://ocslab.hksecurity.net/Dataset/CAN-intrusion-dataset.

[16] NXP Electronics, “S08SG Family of 8-bit Microcontrollers,” 2009. Accessed: Apr. 24, 2021. [Online]. Available: https://www.nxp.com/docs/en/fact-sheet/S08SG8BITMCFS.pdf.

[17] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, “CANet: An Unsupervised Intrusion Detection System for High Dimensional CAN Bus Data,” *IEEE Access*, vol. 8, pp. 58194–58205, 2020, doi: 10.1109/ACCESS.2020.2982544.

[18] M. L. Han, B. Il Kwak, and H. K. Kim, “Anomaly intrusion detection method for vehicular networks based on survival analysis,” *Veh. Commun.*, vol. 14, pp. 52–63, Oct. 2018, doi: 10.1016/j.vehcom.2018.09.004.

[19] H. K. Kim, “Car-Hacking Dataset,” *Hacking and Countermeasure Research Lab*. https://sites.google.com/a/hksecurity.net/ocslab/Datasets/CAN-intrusion-dataset (accessed Dec. 01, 2020).

[20] G. Dupont, A. Lekidis, J. Den Hartog, and S. Etalle, “Automotive Controller Area Network (CAN) Bus Intrusion Dataset v2,” *4TU.ResearchData*, 2019.

https://data.4tu.nl/repository/uuid:b74b4928-c377-4585-9432-2004dfa20a5d (accessed Dec. 09, 2019).

4 WINDS: A Wavelet-based Intrusion Detection System for Controller Area Network (CAN)

Vehicles are equipped with Electronic Control Units (ECUs) to increase the overall vehicular system's functionality and connectivity. However, the rising connectivity exposes defenceless internal Controller Area Network (CAN) to cyberattacks. An Intrusion Detection System (IDS) is a supervisory module proposed for identifying the CAN network's malicious messages without modifying legacy ECUs and causing the traffic overhead. The traditional IDS approaches rely on time and frequency thresholding, leading to high false alarms, whereas state-of-the-art solutions may suffer from vehicle dependency. This chapter presents a wavelet-based approach to locating the CAN traffic behaviour change by analysing the CAN network's transmission pattern. The proposed Wavelet-based Intrusion Detection System (WINDS) is tested on various attack scenarios using real vehicle traffic from two independent research centres while expanding toward more comprehensive attack scenarios using synthetic attacks. The technique is evaluated and compared against the state-of-the-art solutions, along with the baseline frequency method. Experimental results show that WINDS offers a vehicle-independent solution applicable for various vehicles through a unique approach while satisfactorily generating low false alarms. The organisation of the chapter is presented in Figure 4-1.

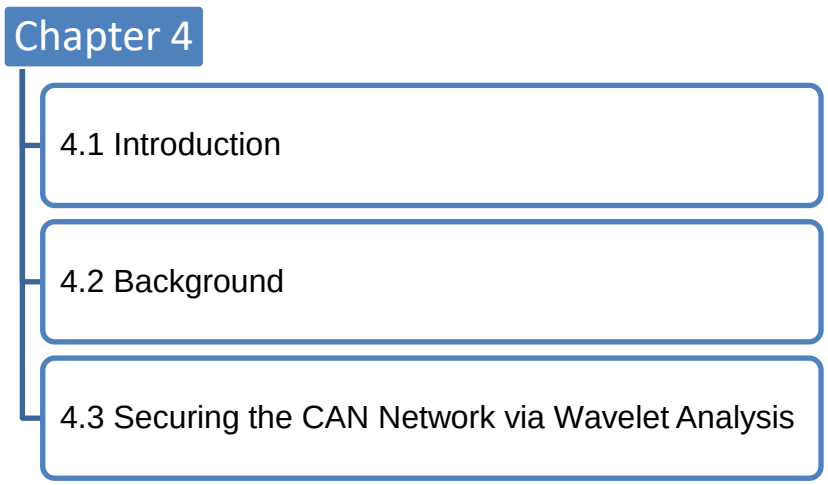

Figure 4-1 Organisation of Chapter 4

4.1 Introduction

The vehicles are getting more connected and autonomous year by year thanks to communication between Electronic Control Units (ECUs), which controls one or more vehicle functions such as engine control, telematics control, and airbag deployment. There are various established in-vehicle communication standards such as Controller Area Network (CAN), FlexRay, Local Interconnect Network (LIN), and Media Oriented Systems Transport (MOST) [1]. Among these, CAN is the most widely used in-vehicle communication protocol [2] because of its recognised advantages in robustness, suitability for real-time networks, easy maintenance, and low-cost implementation. However, it does not have any intrinsic security features to protect against cyberattacks. The vulnerabilities of the CAN were presented for the first time by Hoppe et al. in 2007 [3], [4]; since then, researchers have demonstrated a variety of physical and remote access attacks [5]–[7]. The increasing number of attacks shows that the protocol is defenceless to cyber attacks.

Although the problem's root cause is a lack of encryption and authentication, cryptographic methods are not feasible. The problem is mitigated with an Intrusion Detection System (IDS). IDS can provide adaptable protection by monitoring the CAN network and labelling the malicious messages without modifying the legacy ECUs.

Different IDS approaches are applied to mitigate the security problem of the CAN network. Some of these solutions are developed based on promising machine learning techniques like Hierarchical Temporal Memory (HTM) [8], Generative Adversarial Nets (GAN) [9], Long Short-term Memory (LSTM) [10], and other deep neural networks [11], [12]; however, such methods initially suffer from high computational power. Additionally, these methods are heavily vehicle dependent and require specific training for different vehicle makes and models. Similarly, entropy-based IDSs [13]–[15] need training to detect anomalies. They are also highly vulnerable to attacks that do not change the entropy, for instance, replay attacks. Others applied specification-based IDS solutions [16], [17] by creating rules based on the protocol specification. However, these solutions are protocol-

dependent and can fail if an attacker mimics the sequence of messages. Although IDS promises to address the CAN's vulnerabilities by labelling malicious messages despite the limited resources, available IDSs have major weaknesses [18], such as high false-positive rate, vulnerability to certain attack types, and vehicle dependency. Many IDS solutions do not even consider the detection time, which has an enormous impact on real-time systems.

Our vision to overcome the problem is to explore techniques that speed up attack detection time and reduce the IDS' decision-maker unit's dependency on prior knowledge, with an aim to reduce the rate of false alarms, which ultimately increases attack detection accuracy. In this regard, the chapter contributes to identifying malicious messages by analysing network traffic behaviour using wavelet analysis rather than its frequency value. The main contributions of the chapter are the followings:

- A novel fast detection wavelet-based IDS for in-vehicle networks
- A vehicle independent IDS approach for attack detection without prior knowledge

4.2 Background

4.2.1 Wavelet Transform

Wavelet analysis provides a frequency analysis of the signal and gives information about breakpoints, trends, and self-similarity. It is used in various fields, including information security, oceanography, medicine, and finance. Unlike the Fourier Transform, it gives frequency analysis on the time domain. Continuous Wavelet Transform (CWT) converts signal $f(t)$ into wavelet coefficients $F(a,b)$ which is a function of scale a and position b as defined below:

$$F(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} f(t)\ \psi^*\left(\frac{t-b}{a}\right) dt \quad [19] \qquad \textbf{(4-1)}$$

where ψ is called mother wavelet, which is any function that satisfies:

$$\int_{-\infty}^{\infty} \psi\,(\mathrm{t})\, dt = 0 \quad [19] \qquad \textbf{(4-2)}$$

$$\int_{-\infty}^{\infty} \psi^2\,(\mathrm{t})\,dt = 1 \quad [19] \qquad \textbf{(4-3)}$$

Scaling means compressing or stretching the mother wavelet. While the compressed wavelet provides rapidly changing high-frequency information, the stretched one gives details of slow changes. The scaling feature offers local and global details of the signal. Unlike Discrete Wavelet Transform (DWT), which has a decreasing number of coefficients with increasing scaling factor, CWT has the same number of coefficients at each scale. This redundancy (i.e. has the exact time resolution as the original data) of CWT provides a more accurate time-frequency spectrum.

4.2.2 Intrusion Detection and Related Work

An IDS can be categorised as signature-based and anomaly-based. As shown in Figure 4-2.a, Signature-based IDS has an attack (signature) database and works like anti-virus software. If an attack from the database occurs, it can identify the attack. On the other side, anomaly-based IDS, as shown in Figure 4-2.b, characterises the system's behaviour and compares it with baseline and alerts if the deviation from the baseline exceeds a certain threshold. Although signature-based IDS is quite successful for known attacks, it cannot detect unknown attacks and requires a regular database update. Hence, it is impossible to know that all the attacks and regular updates can be a hassle; anomaly-based IDS solutions have advantages over signature-based ones.

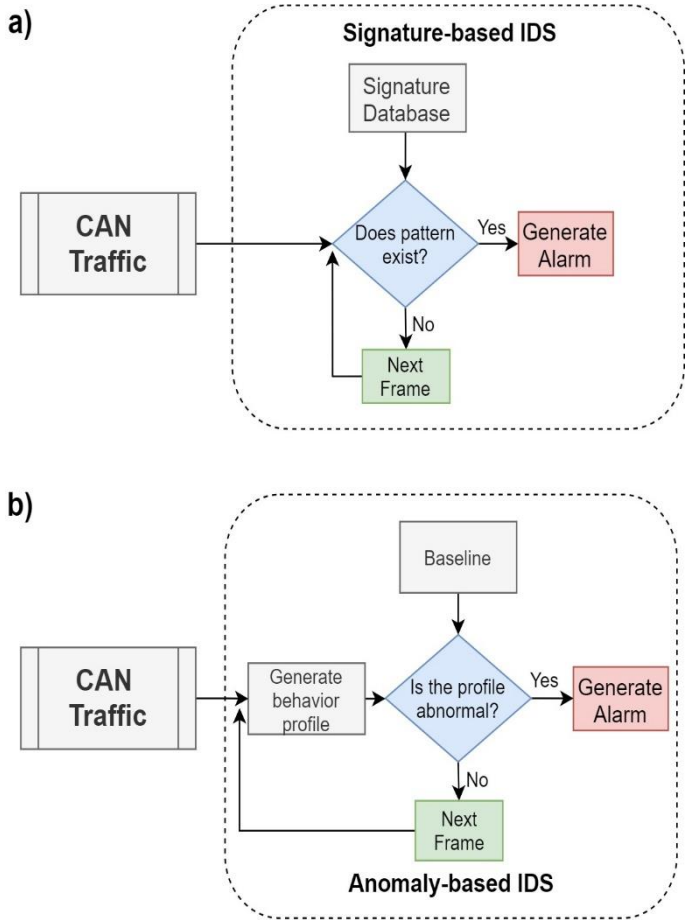

Figure 4-2 a) Flowchart of signature-based and b) anomaly-based intrusion detection systems.

As summarised in Table 4-1, the current IDS solutions use various parameters to analyse the CAN network. Research in [20]–[22] takes advantage of the network's physical characteristics. Thanks to random manufacturing variations, cabling, and ageing, each transceiver has a slightly different signature on the signal even though they transmit the same data. Analysing these signatures gives the means to identify authentic messages. Although these methods are highly reliable in a controlled environment, their performance changes significantly based on environmental changes like temperature. They are also vulnerable to detect malicious messages from the software layer, as explained in [5].

Table 4-1 Summary of recent intrusion detection systems for CAN bus

Ref.	Parameter	Algorithm / Method	Advantages	Downsides
[20]	Electrical signal	Support vector machine and boosted decision tree	Robust to some attack types, differentiate between an error and an attack	High cost and vulnerability to environmental conditions
[21]	Electrical signal	Multilayer perceptron	Robust detection of malicious nodes	High cost and vulnerability to environmental conditions
[22]	Time intervals	Recursive least squares and Cumulative Sum	Identification of attacked ECU	Works only for periodic signals, susceptible to environmental conditions
[9]	A pattern of CAN ID	Generative adversarial nets	Robust to attacker manipulation	Heavy resource usage, vehicle dependency
[12]	Traffic pattern	Deep convolutional neural network	Better performance than other machine learning methods	Heavy resource usage
[23]	Timing analysis	Specification-based	Low computational requirement	Defining the specifications
[24]	Remote frame Timing	Offset ratio	Efficient and straightforward algorithm with low-cost hardware	Increased traffic

[25]	Period and payload	Bloom filtering	Low memory usage	High computational power
[26]	Time intervals	Z-score and ARIMA	Minimal training	High time-to-detect

Müter et al. [27] identify eight anomaly detection sensors that provide the essential input to structure an in-vehicle network. These are frequency, formality, location, range, correlation, protocol, plausibility, and consistency. These are not necessarily physical sensors but are signal processing boxes/tools that process the CAN bus's network traffic to observe and monitor changes for such parameters. Any IDS solutions use one or multiple of these sensors. As many ECUs broadcast CAN frames regularly, frequency is one of the most critical anomaly detection sensors to characterise the automotive network, if not the best. An intrusion into the CAN network will disrupt the regularity of the transmissions and the system's frequency. Although time thresholding is a simple technique to detect attacks, it can generate a high false-positive rate. On the contrary, frequency analysis gives more stable information [28]. Therefore, the CAN network's frequency analysis is a simple but effective IDS solution for resource-constrained vehicles.

There are multiple pieces of research to assess the time interval and frequency of the CAN messages. Some of these use basic statistical analysis [29], [30], but they are highly vehicle-dependent. Machine learning algorithms like One-Class Support Vector Machine (OCSVM) [31], Gaussian mixture model [32] also proposed to detect anomalies via frame timing analysis; however, they require a comprehensive training data set for each vehicle model. ARIMA and Z-score were proposed [26] to minimise the training phase and vehicle dependency, but a successful result requires a long window size, which will increase the detection time. Lee et al. [24] analysed the response time of the ECUs by sending them remote frames. Their method requires low computational power and is successful in detecting attacks. The downside of the technique is that it increases bus traffic by sending remote frames.

On the other hand, wavelet analysis has outstanding performance, mainly due to its simple procedure, easy computation, and reconstructable decomposition. This motivated researchers from the IT security domain to benefit [33]–[35]. Spicer [21] proposed wavelet analysis for CAN bus to complement his noise-content-based multilayer perceptron IDS with frequency analysis. His implementation was

limited to the signal level and analysed the electrical characteristic to identify different signatures. By fingerprinting ECUs, it is possible to identify the sender ECU; therefore, the work can also be regarded as an authentication method. The work presented in this chapter moves beyond Spicer's research and intends to develop the entire IDS based on wavelet analysis. The WINDS is applied to message level and analyses behaviour of message frequency, facilitating low-latency frequency analysis for the CAN network without increasing the network traffic and training data requirement

4.3 Securing the CAN Network via Wavelet Analysis

The frequency profile contains essential information about CAN messages' authenticity obtained by the Continuous Wavelet Transform (CWT). CWT is a powerful tool for the precise localisation of frequency components on the time axis, useful for identifying irregularities in the CAN network's traffic pattern. In order to find the signal's behaviour change, WINDS benefits from CWT for dividing the network pattern, which is a continuous time-series signal, into different scale components. Then the analysis is further carried out on the scale domain. Figure 4-3 visualises the CAN traffic and its wavelet representation. The figure depicts a set of large CWT coefficients located vertically around $t = 6.318\ (s)$ where the change (attack) occurs in the signal. The area of large coefficient values, called the cone of influence, spreads with rising scale but still centred at $t = 6.318\ s$. It presents us which CWT coefficients are affected by the signal at that point. Therefore, the proposed WINDS algorithm can detect both long-time and sudden short-time duration attacks by analysing scales.

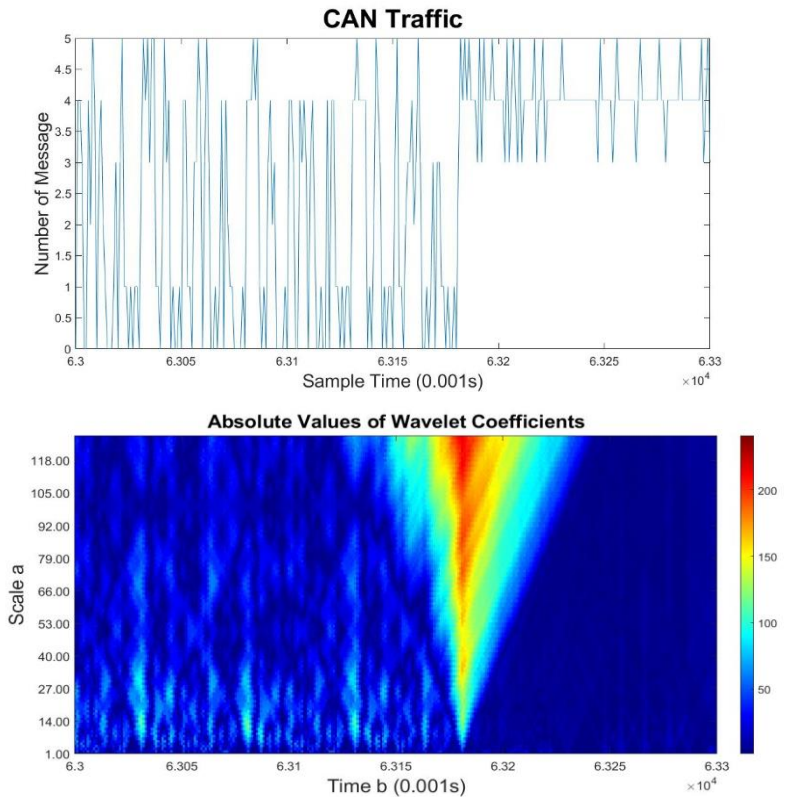

Figure 4-3 Message count of the CAN traffic (top) during a DoS attack and its wavelet analysis (bottom).

The WINDS algorithm can be split into four stages, as shown in Figure 4-4: data collection and preprocessing, behaviour profiling with CWT, anomaly decision, and parameter initialisation.

Data collection and preprocessing: The first stage is to monitor the CAN traffic under various no-attack and attack scenarios. This is a time-consuming data creation task and requires multiple resources and tools. Several research centres lead such experiments and data collection steps, providing researchers with valuable datasets. Although open-access datasets might be limited to specific cases, they can be well extended to comprehensive data by considering various attack models and understanding the CAN bus system's technical details and the vehicle's performance. This usually turns in populating the initial experimentally collected dataset with several synthetic attacks that mimic the real attacks.

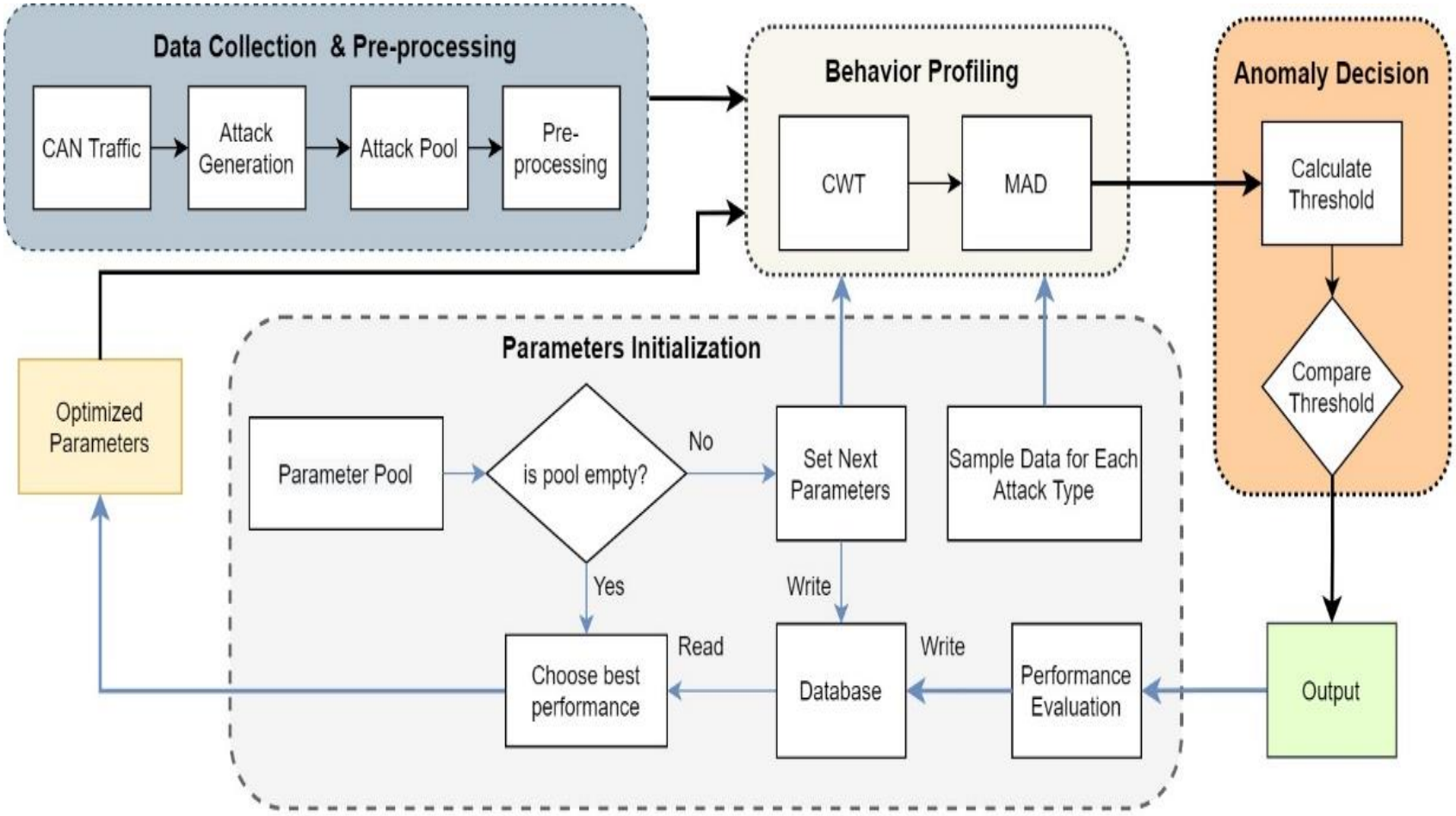

Figure 4-4 The flowchart of wavelet-based intrusion detection system for in-vehicle communication.

The preprocessing step starts with windowing the dataset, proceeded with a feature extraction step, which is usually conducted by the signal-processing tool. Assuming a windowed data as *w(t)*, it is a collection of messages, *M,* while each has a time interval of sampling time t_s as in (4-4), representing traces of the message counted over the previous n samples:

$$w(t) = \{ M_{t-(n-1)*t_s)}, M_{t-(n-2)*t_s)}, \dots, M_t\} \quad \textbf{(4-4)}$$

The WINDS benefits from message count N_w in the CAN traffic in window *w* within a specified time interval between t and $t - t_s$, where t_s is the period of the interval. Hence, we specify a message frequency, S_f, with the following equation, applied on i[th] window w_i, to account the frequency of message in that window:

$$N_i = S_f(M_{w_i}) = \sum_{k=1}^{n_{max}} M_k \quad \textbf{(4-5)}$$

where n_{max} is the maximum number of messages that a window may have, and M_k represents the existence of the k^{th} message within the i^{th} window (w_i) that is one if a message exists; otherwise, it is zero. The window is stretched from the current time to the past, and the analysis is processed frequently. This results in the featured i_{th} window by the frequency conversion S_f, represented by w_i^S, as in the following equation:

$$w_i^S = \{N \in Z: \exists N_1, \dots, N_{n-max} \in w \; with \; N = S_f(M_{w_i})\} \quad \textbf{(4-6)}$$

Behaviour profiling: The second stage generates the behaviour profile from the preprocessed traffic signal using the wavelet transform in (4-1). It transforms w_i^S to a set of wavelet coefficients $W(a, b)$, which is a two-dimension matrix of $n \; x \; k$ where k is the highest wavelet scale and *n* is the window size. To decrease the complexity and get meaningful data out of all wavelet scales, Mean Absolute Deviation (MAD) is used, as in (4-7), where L is the length of scale for the chosen w_i^S after wavelet transformation, j and q denotes a specific component of the scale as an index, and A_i^{MAD} projects the results after applying MAD function on the scaled component for the i[th] w_i^S. Therefore, MAD provides the absolute

deviations from the mean point and gives information about the wavelet scale changes in each sample. Figure 4-5 demonstrates an example of MAD transformation from the wavelet coefficients during a replay attack.

$$A_i^{MAD} = \frac{1}{L}\sum_{j=1}^{L}\left|a_j - \frac{1}{L}\sum_{q=1}^{L} a_q\right| \quad \textbf{(4-7)}$$

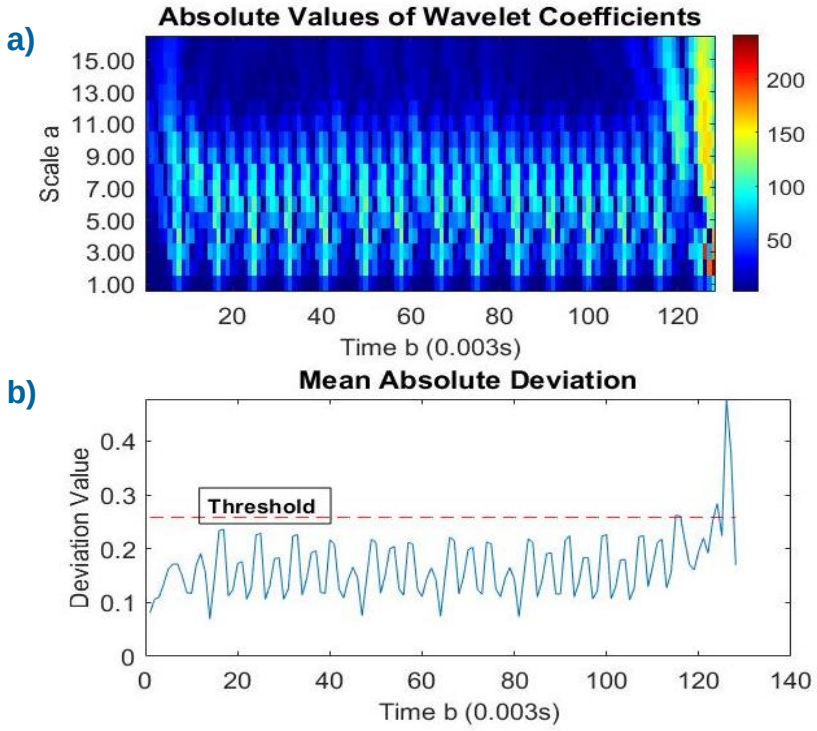

Figure 4-5 The wavelet transform of the windowed signal w(t) for single ID during replay attack (top) and median absolute deviation of W(a,b) (bottom).

Anomaly decision: This is the step for interpreting the wavelet coefficients, which leads to change point detection, needed for exploring anomalies' symptoms caused by an attack. The core of anomaly detection is assessing each window to find behaviour deviations using a thresholding technique. Donoho and Johnstone [36] proposed a universal threshold λ_u defined as:

$$\lambda_u = \sigma \,.\sqrt{2\log(N)} \quad \textbf{(4-8)}$$

where σ is the standard deviation and *N* is the number of samples. Donoho and Johnstone's threshold technique's advantages are slightly limited to denoising the White Gaussian noise affected signals by finding substantial change. Mozzaquatro et al. [37] presented that the universal threshold λ_u should be updated by a constant correction factor ρ to get a better results, (4-9). The constant correction factor ρ depends to specific applications of interests like anomaly detection for web attacks, boundary conditions, etc.

$$\lambda = \rho \,.\, \lambda_u \qquad \textbf{(4-9)}$$

It is known so far that thresholding is the crucial element of an IDS so that low and high thresholds lead to false positives and false negatives results, respectively. WINDS involves an adaptive thresholding technique for increasing the accuracy of decisions and calculating a new threshold for each window by updating the ρ parameter based on each window's MAD value. Finally, the updated λ is applied to the anomaly decision process for denoting the values higher than the threshold as anomalies and so detection of threat. Figure 4-5 visualises the WINDS' thresholding mechanism via an example, demonstrating the results of converting the wavelet coefficients in Figure 4-5.a into MAD values in Figure 4-5.b. If any of the MAD values within a window exceeds the threshold, that window is regarded as malicious.

There are conditions in which MAD produces results equal to zero based on the specific attack types. An example is when a flooding attack causes suspension of the messages from lower priority ECUs in the presence of the CAN network's arbitration mechanism. In such cases, the window spans inside the attack duration, which causes all the wavelet coefficients to turn to zero, and as a result, MAD generates zero.

Parameter initialisation: The proposed IDS involves a multi-parameter optimisation problem that requires extensive time and works to find the best performance for the WINDS. Instead, the effort is put forward to initialise the WINDS with the best possible parameters experimentally founded by looking into the performance when feeding WINDS with various datasets. This stage is run

only once for gathering the values of the parameters. At first, the ranges of each parameter value are chosen and inserted into the parameter pool. These parameters are wavelet type (Haar and Daubechies), wavelet scale (from 4 to 32), window size (from 32 to 256), window-type (discrete and continuous), sample time (from 0.5 ms to 3 ms), and threshold (from 1.1 x MAD to 2.2 x MAD of the current window). The algorithm is then tested for one attack data for each attack type for all the parameters inserted in the pool. The parameter setting which provides better performance on average is chosen as experimental parameters.

4.4 Results and Discussions

4.4.1 Experimental Setup

The experiment is carried out by using Matlab software. The continuous wavelet transform is implemented by using *cwt* command. Resulting from the parameter initialisation step mentioned in Section 4.3, we set the parameters with the specifications given in Table 4-2. The window *w(t)* includes 128 samples, consisting of 384 ms network traffic, collected by a sampling time of 3 ms for the ID-based data segments. The setup is tied for short sampling times, ensuring the window would be populated with sufficient active data while avoiding information misses. Short sampling time also results in earlier attack detection, consequently. The threshold is set to 1.8 x MAD value of the current window. The Haar wavelet, consisting of shifted and scaled square wave functions, is used as a mother wavelet in the analysis. The Haar function in (4-10) [38] has the potential for looking at differences of averages, essentially. The initial scale for Haar wavelet in this experiment is 16. The source code for WINDS can be found in Appendix A.2.

Table 4-2 Experimental setup specifications

Parameter	**Value**
Number of samples in w(t)	128
Number of messages in a window, N	Variable*
Network traffic	384 ms
Sampling time	3 ms
Type of wavelet	Haar
Number of scale	16
Constant correlation factor, ρ	∝ to MAD
Threshold	1.8 x MAD

* Number of the messages depends on the traffic and ID of ECU.

$$\psi^{(H)}(t) = \begin{cases} 1, & 0 \le t < \frac{1}{2}; \\ -1, & \frac{1}{2} \le t < 1; \\ 0, & otherwise. \end{cases} \quad \textbf{(4-10)} \text{ [38]}$$

4.4.2 Results

The WINDS was tested on two groups of the dataset collected from three commercial vehicles using the framework mentioned in Section 3. The first experiment evaluates WINDS capabilities on a broad range of synthetic attacks with varying attack strength (testing code for WINDS can be found in Appendix A.3). Then, the second experiment assesses the performance of WINDS on a real vehicle attack dataset and compares it with existing solutions.

4.4.2.1 Synthetic Attacks – Attack Types

The synthetic attacks allow us to safely implement various attack scenarios, facilitating the observation of the IDS' performance and limitation on different

attacks by tuning the attack strength and duration. Using such techniques, we tested WINDS on various attack scenarios, including DoS, suspension, and replay attacks, along with an attack-free dataset.

The experiment is constructed on splitting the entire network data into segments based on the ID numbers, then proceed with investigating each ID-based data separately to get satisfactory results, shown in Table 4-3.

Table 4-3 The performance of WINDS for the synthetically generated data

Data Source	Attack Type	Accuracy	Sensitivity	FPR	Precision	MTTD (s)
Vehicle 1	No attack	0.9997	-	0.0003	-	-
Vehicle 2		0.9999	-	0.0001	-	-
Vehicle 1	Denial of Service	0.9995	0.9982	0.0004	0.9920	0.003
Vehicle 2		1.0000	0.9979	0.0004	0.9985	0.006
Vehicle 1	Suspension	0.9999	0.9980	0.0001	0.9879	0.003
Vehicle 2		0.9997	0.9981	0.0003	0.9993	0.006
Vehicle 1	Replay	0.9998	0.9893	0.0001	0.9986	0.003
Vehicle 2		0.9997	0.9974	0.0003	0.9863	0.003

Mean value of results for the ten datasets.

The attack-free dataset gives information about how IDS will perform and react in the normal traffic mode by looking into evaluation metrics such as the FPR rate. It is essential to keep FPR low; otherwise, higher rates generate many false alarms, which drivers may ignore. Moreover, a higher FPR rate hardens the tasks of the security team. The WINDS' FPR rate is kept for less than 0.0004.

Denial of Service (DoS) attack by flooding high-priority messages can significantly affect network behaviour. Although the attack is implemented by sending messages with the highest priority (CAN ID '000'), it can be detected by monitoring any ID in the network. The WINDS algorithm successfully detected

DoS attacks, with an average attack detection rate of 99.82% and 99.79% for Vehicle 1 and Vehicle 2, respectively. The detection rate can reach as high as 99.94% for more prolonged attack durations. The attacks were also swiftly detected in less than 6 ms.

Suspension attack has similar results to DoS attacks, as shown in Figure 4-6. It could be anticipated the same because the arbitration scheme does not allow lower priority nodes to transmit when the DoS attack is implemented. Therefore, the suspension attack mimics the DoS attack. The average sensitivity values for Vehicle 1 and Vehicle 2 were 99.8% and 99.81%, consecutively.

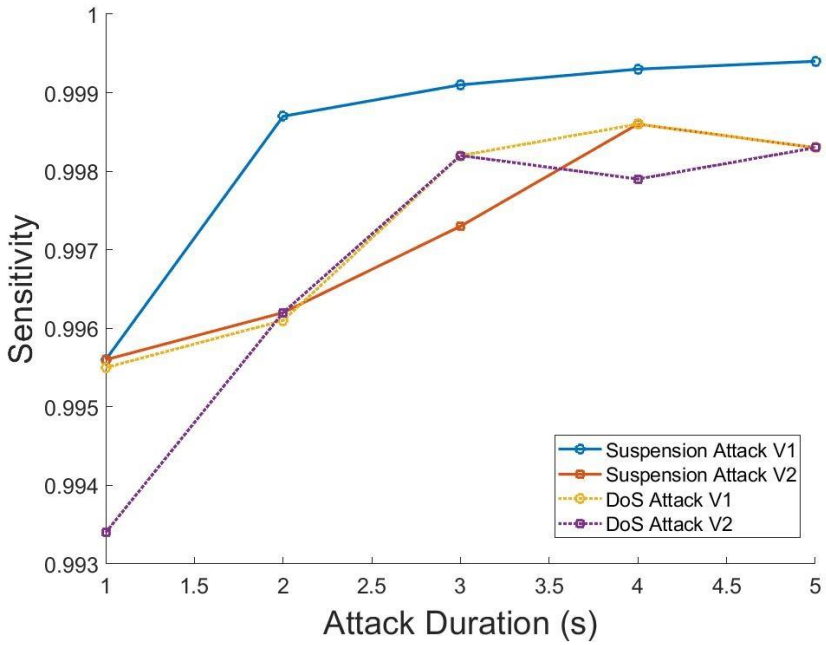

Figure 4-6 The sensitivity of the WINDS algorithm during various suspension and DoS attacks. The sensitivity of the algorithm gets better with the rising attack duration.

Replay attacks were implemented for a short duration of time (75 ms and 66 ms) with low message insertion rates (from 8 to 30 frames). The results shown in Figure 4-7 depict that the WINDS algorithm can respond in milliseconds and successfully detect over 96% of the attacks with almost zero false-positive rate. The observation is that the algorithm's sensitivity rises with the increased rate of malicious messages while the TTD decreases.

The experimental results show that the sensitivity of the WINDS is correlated with the attack strength. In general, the sensitivity increases for the longer duration and more frequent attacks, as seen in Figure 4-7.

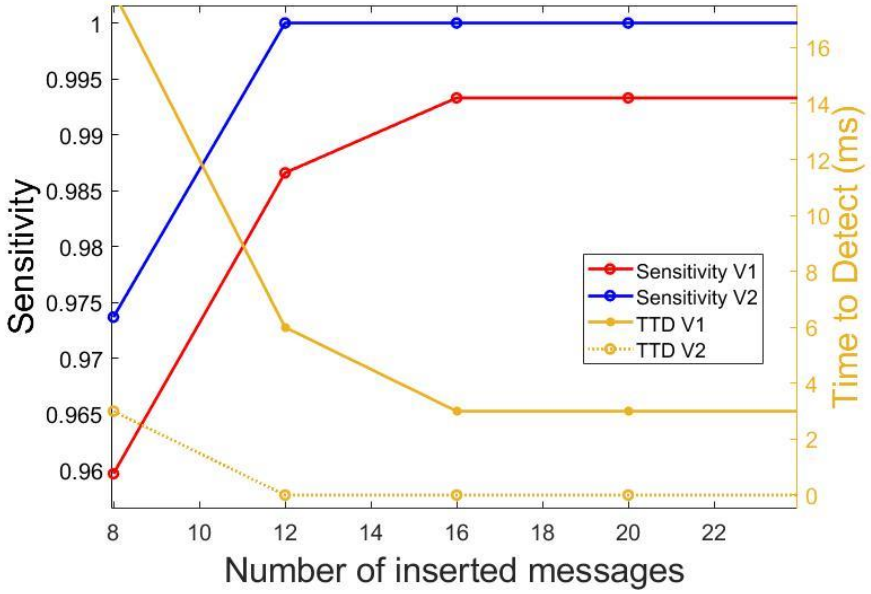

Figure 4-7 The sensitivity of the WINDS algorithm during different replay attacks. The increased message insertion rate increases the sensitivity while decreasing the time to detect.

4.4.2.2 Comparative Analysis of the WINDS on Real Vehicle Attacks

In the second experiment, WINDS is tested on real-world vehicle attacks and compared with baseline frequency-based IDS and other existing methods. As benchmarking dataset has significant importance for reliable testing, WINDS is tested on the most widely accepted dataset, the Car-hacking dataset [39]; and compared with the state-of-the-art methods which use the same dataset. These are GIDS[9], DCNN[12], and SAIDuCANT[23]. GIDS method converts CAN data into an image and applies generative adversarial nets. The method only uses CAN ID to speed up image generation. DCNN method applies a deep convolutional neural network to a two-dimensional binary matrix generated from the CAN traffic. The third method, SAIDuCANT, is a specification-based IDS as compared to machine learning-based methods mentioned above. SAIDuCANT

Table 4-4 Comparison of the WINDS with existing methods using real vehicle attack data

Attack Type	IDS	Accuracy	Sensitivity (Recall)	Precision
Gear Spoofing	WINDS	0.9883	0.9845	0.9958
	SAIDuCANT	0.8262	0.9702	0.8245
	GIDS	0.9620	0.9650	0.9810
	DCNN	0.9995	0.9989	0.9999
	Frequency-based	0.9273	0.8770	0.9886
RPM Spoofing	WINDS	0.9926	0.9890	0.9986
	SAIDuCANT	0.8033	0.9636	0.8010
	GIDS	0.9800	0.9900	0.9830
	DCNN	0.9997	0.9994	0.9999
	Frequency-based	0.9472	0.9211	0.9815
Fuzzy Attack	WINDS	0.8778	0.8339	0.9816
	SAIDuCANT	0.8782	0.9958	0.8639
	GIDS	0.9800	0.995	0.9730
	DCNN	0.9982	0.9965	0.9995
	Frequency-based	0.8170	0.7556	0.9599
DoS Attack	WINDS	0.9497	0.9415	0.9797
	SAIDuCANT	0.9808	1.0000	0.9771
	GIDS	0.9790	0.9960	0.9680
	DCNN	0.9997	0.9989	1.0000
	Frequency-based	0.8711	0.8316	0.9617

monitors the network's timing behaviour according to predefined timing specifications.

The comparison result of WINDS with other state-of-the-art methods for the real-vehicle attacks is summarised in Table 4-4. Gear spoofing and RPM attacks directly target certain IDs, and the WINDS can detect 98.45% and 98.90% of these attacks accordingly and provides over 99% precision for both cases.

The attack detection rate and accuracy decrease for Fuzzy and DoS attacks. This is partly because these attacks do not target any particular IDs; therefore, they are not as disruptive as direct attacks like gear or RPM spoofing. While the WINDS' sensitivity for the DoS attack is 94.15%, it can decrease to 83.39% for the fuzzy attack. This is an expected result; hence DoS attack was implemented using the highest ID number while the fuzzy attack transmits random ID numbers. Some of these IDs have low priority and have no disruption in transmitting authentic messages because of the arbitration process.

WINDS is also compared with some alternative methods including frequency-based IDS, which measures the frequency of attacked ID and generates an alarm if the threshold exceeds lower or higher threshold bonds. The WINDS outperforms the frequency-based IDS in all metrics for all attack types. Our method also generates better results than GIDS and SAIDuCANT methods for gear and RPM spoofing. On the other hand, these methods have better performance than WINDS for DoS and Fuzzy attacks. Although DCNN has the best performance for this dataset, it requires extensive training with attack data; because it involves a supervised learning method. It is also computationally expensive and requires GPU acceleration; therefore, it is not feasible to deploy in a resource-constrained environment.

4.4.2.3 Discussions

An IDS can be implemented as host-based (also known as node-based) or network-based. In the host-based IDS, each ECU has an integrated IDS and may dismiss the message according to the IDS decision. However, this requires additional resources in each ECU. On the other hand, the network-based

approach has only one IDS implemented on the gateway ECU. The WINDS is independent of implementation perspectives, suitable for implementation through host-based or network-based approaches with the same performance; however, the required resources would differ. This allows the method to be implemented on various applications, from low-end resource constraint vehicles as a network-based IDS to high-end vehicles as an advanced sensor for intrusion prevention systems in each ECU.

An IDS should satisfy specific requirements for vehicles, which are real-time safety-critical cyber-physical systems. In short, it should detect attacks correctly in an acceptable time frame while using limited resources and without causing false alarms. Therefore, WINDS is assessed based on three criteria: timing behaviour, success rate, and resource usage.

4.4.2.3.1 Timing Analysis

Successful IDS must detect attacks as soon as possible to prevent propagating misinformation and causing system misbehaviour. A metric suitable for measuring the algorithm's behaviour is TTD, which varies by the parameters like the sampling time and the threshold. Assessing WINDS by TTD demonstrated that an increase in the attack strength decreases the detection time, as presented in Figure 4-7.

Another key parameter for time analysis is the processing time. Actual processing time requires implementing the WINDS algorithm on an ECU, which is not covered in this research. As WINDS can be implemented as a network-based IDS, this can be ignored even on low-end vehicles using only one high-end automobile processor on the gateway ECU.

The WINDS algorithm can detect an anomaly in milliseconds. Considering the delay times in the CAN network [40], the algorithm should be suitable for the real-time analysis of most ECUs.

4.4.2.3.2 Success Rate

As the main parameter, changes in the message frequency should be observed by WINDS to detect attacks. The method cannot locate, for instance, impersonate attacks, where a node is suspended, and a malicious node transmits on behalf of the suspended one by causing the protocol error. However, this can be easily detected by counting the error frames. In contrast, the proposed algorithm successfully detects time variations, which enable WINDS to locate all the flooding attacks by analysing only a single ID even though the attacker targets different IDs.

The threshold is the most critical parameter that affects the success rate. The lower threshold value will increase the detection rate, but it will also raise false alarms. Additionally, the threshold can be adjusted based on IDs and adapted to the arbitration process of the CAN for increasing the overall performance. This adaption will decrease the false alarms because lower priority IDs are not as punctual as the higher priority IDs due to the arbitration mechanism in CAN. The Receiver Operating Characteristic (ROC) curves in Figure 4-8 depict WINDS' behaviour for three different attack models: replay attack, gear and RPM spoofing

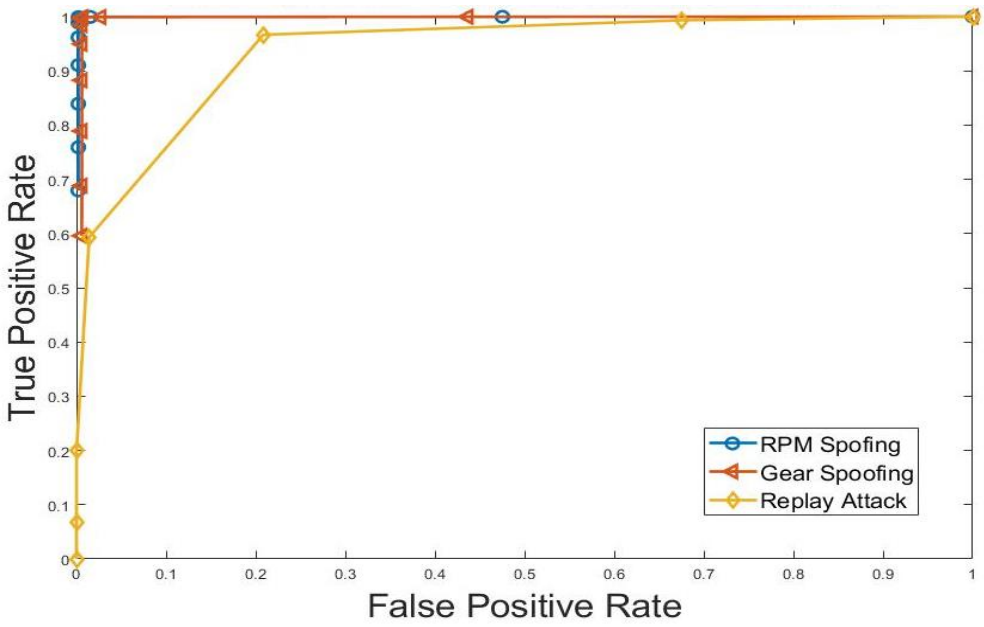

Figure 4-8 The Receiver Operating Characteristic (ROC) curves for varying threshold values for RPM spoofing, gear spoofing, and replay attack.

attacks. The result shows that WINDS provides a good performance characteristic.

An alternative way to increase the system's performance comes from understanding the driving mode; hence, some ECUs are linked to different driving modes. Theoretically, the wavelet can detect this change and may give a false alarm during the transition. After a window passes the transition period, it does not provide a warning. This requires further investigation and testing on data from different driving modes.

4.4.2.3.3 Resource Usage

The vehicles have limited bandwidth, memory, and computational power. Therefore, a feasible IDS should demand low resources. The WINDS does not transmit any messages, so it does not affect the bandwidth.

The memory usage of WINDS is directly proportional to the window size. It analyses the timing of the messages and does not need to store data bits. It only requires a single bit of memory storage as a flag identifying the message that exists in the given sample time. Therefore, each ID requires n-bit memory equal to the window size, which is 16 bytes in this experiment. This is a very reasonable amount, even for low-end ECUs.

The CWT mainly drains computational power. If the scale is increased, the required power will increase, too. Efficient CWT algorithms are essential for making the IDS affordable for all ECUs. A way to reduce the algorithm's computational cost is to sacrifice some memories when ECU has limited computational power available. For instance, the n-bit window is not necessarily required to be transformed to wavelet coefficients as a whole each time. Instead, updating only some bits from the previous transform is sufficient while keeping the rest unchanged. A partial updating of the last window results in the new window, which was expected to be transformed.

4.5 Future Directions

Although the research results demonstrated so far in this chapter through various tests, analysis, and evaluation are promising, further improvement is achievable

by analysing each wavelet scale individually. This additional improvement would be at the cost of higher complexity and computational needs. It is worth investigating alternative wavelet-based IDS systems such as Discrete Wavelet Transforms (DWT) and Maximal Overlap Discrete Wavelet Transform (MODWT), mainly to reduce computational cost and conduct further assessments and comparisons with other techniques.

WINDS is limited to analysing system behaviour based on message frequency, and it is not extended toward nodes transmitting infrequent messages. The current implementation is not detecting attacks that do not affect the message frequency, requiring further investigations.

There is still a need for experimenting with real cars, considering various attack scenarios followed with suitable data collection to generate comprehensive and efficient datasets. Existing datasets available from open-access research centres are limited to specific cases, and yet, they do not provide essential system specifications under test and technical details of testing scenarios. This chapter successfully demonstrated methods for generating synthetic attacks to overcome weaknesses from open-access datasets. This is limited to simple cases and leaves generation complex synthetic attacks, which are needed for sophisticated attacking scenarios, for future research. Furthermore, it is also crucial to test IDS on various driver and journey types, meaning that more datasets are needed for achieving efficient analysis.

The lack of available datasets for various vehicle models also prevents us from implementing an optimisation process for the parameter decision. Optimisation on limited datasets will cause overtraining. Therefore, it will be worth investigating optimisation techniques when we have enough independent datasets to improve the performance of WINDS.

It is essential to prevent attacks that cause system misbehaviour for safe driving. The current implementation of WINDS is designed as an intrusion detection system. To implement a real-time intrusion prevention system, each ID should be analysed separately to gather its deadline. Then WINDS should be adapted to respond to the deadline. The prevention mechanisms to invalidate messages

also need to be assessed and combined with WINDS. It is also worth mentioning that the WINDS is implemented on a personal computer. Although the TTD will be the same, the processing time will vary. Therefore, we aim to apply the WINDS on an ECU and gather processing time.

References

[1] T. Kosch, C. Schroth, M. Strassberger, and M. Bechler, *Automotive Internetworking*. Chichester, UK: John Wiley & Sons, Ltd, 2012.

[2] K. Matheus and T. Königseder, *Automotive Ethernet*. Cambridge University Press, 2015.

[3] B. Groza and S. Murvay, “Security solutions for the Controller Area Network: Bringing Authentication to In-Vehicle Networks,” *IEEE Vehicular Technology Magazine*, pp. 40–47, 2018.

[4] T. Hoppe and J. Dittman, “Sniffing/Replay Attacks on CAN Buses: A simulated attack on the electric window lift classified using an adapted CERT taxonomy,” in *Proceedings of the 2nd workshop on embedded systems security (WESS)*, 2007, pp. 1–6.

[5] S. Fröschle and A. Stühring, “Analysing the capabilities of the CAN Attacker,” in *European Symposium on Research in Computer Security*, Sep. 2017, vol. 10492 LNCS, pp. 464–482, doi: 10.1007/978-3-319-66402-6_27.

[6] C. Miller and C. Valasek, “A survey of remote automotive attack surfaces,” 2014. Accessed: Mar. 31, 2018. [Online]. Available: https://www.ioactive.com/pdfs/IOActive_Remote_Attack_Surfaces.pdf.

[7] P.-S. Murvay and B. Groza, “DoS attacks on Controller Area Networks by fault injections from the software layer,” in *Proceedings of the 12th International Conference on Availability, Reliability and Security - ARES ’17*, 2017, vol. 10, pp. 1–10, doi: 10.1145/3098954.3103174.

[8] C. Wang, Z. Zhao, L. Gong, L. Zhu, Z. Liu, and X. Cheng, “A Distributed Anomaly Detection System for In-Vehicle Network Using HTM,” *IEEE Access*, vol. 6, pp. 9091–9098, 2018, doi: 10.1109/ACCESS.2018.2799210.

[9] E. Seo, H. M. Song, and H. K. Kim, “GIDS: GAN based Intrusion Detection System for In-Vehicle Network,” in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, Aug. 2018, pp. 1–6, doi: 10.1109/PST.2018.8514157.

[10] A. Taylor, S. Leblanc, and N. Japkowicz, “Anomaly detection in automobile control network data with long short-term memory networks,” in *Proceedings - 3rd IEEE International Conference on Data Science and Advanced Analytics, DSAA 2016*, 2016, pp. 130–139, doi: 10.1109/DSAA.2016.20.

[11] M. J. Kang and J. W. Kang, “Intrusion detection system using deep neural network for in-vehicle network security,” *PLoS One*, vol. 11, no. 6, Jun. 2016, doi: 10.1371/journal.pone.0155781.

[12] H. M. Song, J. Woo, and H. K. Kim, “In-vehicle network intrusion detection using deep convolutional neural network,” *Veh. Commun.*, vol. 21, p. 100198, Jan. 2020, doi: 10.1016/j.vehcom.2019.100198.

[13] M. Muter and N. Asaj, “Entropy-based anomaly detection for in-vehicle networks,” in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, no. IV, pp. 1110–1115, doi: 10.1109/IVS.2011.5940552.

[14] M. Marchetti, D. Stabili, A. Guido, and M. Colajanni, “Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms,” 2016, doi: 10.1109/RTSI.2016.7740627.

[15] W. Wu *et al.*, “Sliding Window Optimized Information Entropy Analysis Method for Intrusion Detection on In-Vehicle Networks,” *IEEE Access*, vol. 6, pp. 45233–45245, 2018, doi: 10.1109/ACCESS.2018.2865169.

[16] U. E. Larson, D. K. Nilsson, and E. Jonsson, “An approach to specification-

based attack detection for in-vehicle networks," in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2008, pp. 220–225, doi: 10.1109/IVS.2008.4621263.

[17] I. Studnia, E. Alata, V. Nicomette, M. Kaâniche, and Y. Laarouchi, "A language-based intrusion detection approach for automotive embedded networks," *Int. J. Embed. Syst.*, vol. 10, no. 1, pp. 1–12, 2018, doi: 10.1504/IJES.2018.089430.

[18] G. Dupont, J. Den Hartog, S. Etalle, and A. Lekidis, "Evaluation framework for network intrusion detection systems for in-vehicle CAN," in *8th IEEE International Conference on Connected Vehicles and Expo, ICCVE*, 2019, no. 1, doi: 10.1109/ICCVE45908.2019.8965028.

[19] D. B. Percival and A. T. Walden, *Wavelet Methods for Time SeriesAnalysis*. Cambridge University Press, 2013.

[20] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "VoltageIDS: Low-level communication characteristics for automotive intrusion detection system," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, 2018.

[21] M. Spicer, A. L. Wicks, A. L. Abbott, S. C. Southward, and M. Spicer, "Intrusion Detection System for Electronic Communication Buses : A New Approach," Virginia Polytechnic Institute and State University, 2017.

[22] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *25th USENIX Security Symposium*, 2016, pp. 911–927, Accessed: May 31, 2018. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/cho.

[23] H. Olufowobi, C. Young, J. Zambreno, and G. Bloom, "SAIDuCANT: Specification-Based Automotive Intrusion Detection Using Controller Area Network (CAN) Timing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1484–1494, 2019, doi: 10.1109/tvt.2019.2961344.

[24] H. Lee, S. H. Jeong, and H. K. Kim, “OTIDS : A novel intrusion detection system for in-vehicle network by using remote frame,” 2017, [Online]. Available: https://www.ucalgary.ca/pst2017/files/pst2017/paper-67.pdf%0Ahttp://ocslab.hksecurity.net/Dataset/CAN-intrusion-dataset.

[25] B. Groza and P. Murvay, “Efficient Intrusion Detection With Bloom Filtering in Controller Area Networks,” *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 4, pp. 1037–1051, Apr. 2019, doi: 10.1109/TIFS.2018.2869351.

[26] A. Tomlinson, J. Bryans, S. A. Shaikh, and H. K. Kalutarage, “Detection of Automotive CAN Cyber-Attacks by Identifying Packet Timing Anomalies in Time Windows,” in *Proceedings - 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN-W 2018*, 2018, pp. 231–238, doi: 10.1109/DSN-W.2018.00069.

[27] M. Müter, A. Groll, and F. C. Freiling, “A structured approach to anomaly detection for in-vehicle networks,” in *6th International Conference on Information Assurance and Security, IAS 2010*, 2010, pp. 92–98, doi: 10.1109/ISIAS.2010.5604050.

[28] C. Young, H. Olufowobi, G. Bloom, and J. Zambreno, “Automotive Intrusion Detection Based on Constant CAN Message Frequencies Across Vehicle Driving Modes,” in *AutoSec 2019 - Proceedings of the ACM Workshop on Automotive Cybersecurity, co-located with CODASPY 2019*, 2019, pp. 9–14, doi: 10.1145/3309171.3309179.

[29] H. M. Song, H. R. Kim, and H. K. Kim, “Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network,” in *International Conference on Information Networking*, 2016, vol. 2016-March, pp. 63–68, doi: 10.1109/ICOIN.2016.7427089.

[30] S. Otsuka, T. Ishigooka, Y. Oishi, and K. Sasazawa, *Design and the Reliability Factor*. Warrendale, PA: SAE International, 2015.

[31] A. Taylor, N. Japkowicz, and S. Leblanc, “Frequency-based anomaly detection for the automotive CAN bus,” in *World Congress on Industrial*

Control Systems Security (WCICSS), 2015, pp. 45–49, doi: 10.1109/WCICSS.2015.7420322.

[32] Y. Hamada, M. Inoue, H. Ueda, Y. Miyashita, and Y. Hata, “Anomaly-Based Intrusion Detection Using the Density Estimation of Reception Cycle Periods for In-Vehicle Networks,” *SAE Int. J. Transp. Cybersecurity Priv.*, vol. 1, no. 1, pp. 39–56, May 2018, doi: 10.4271/11-01-01-0003.

[33] M. Hamdi and N. Boudriga, “Detecting denial-of-service attacks using the wavelet transform,” *Comput. Commun.*, vol. 30, no. 16, pp. 3203–3213, 2007, doi: 10.1016/j.comcom.2007.05.061.

[34] S.-Y. Ji, B.-K. Jeong, C. Kamhoua, N. Leslie, and D. H. Jeong, “Estimating Attack Risk of Network Activities in Temporal Domain: A Wavelet Transform Approach,” in *020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2020, pp. 0826–0832, doi: 10.1109/uemcon51285.2020.9298153.

[35] P. Zuraniewski and D. Rincón, “Wavelet Transforms and Change-point Detection Algorithms for Tracking Network Traffic Fractality,” 2006. Accessed: Jun. 25, 2019. [Online]. Available: https://ieeexplore.ieee.org/ielx5/11058/34932/01678244.pdf?tp=&arnumber=1678244&isnumber=34932&ref=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8=.

[36] D. L. Donoho and J. M. Johnstone, “Ideal spatial adaptation by wavelet shrinkage,” *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994, doi: 10.1093/biomet/81.3.425.

[37] B. A. Mozzaquatro, R. P. De Azevedo, R. C. Nunes, A. D. J. Kozakevicius, C. Cappo, and C. Schaerer, “Anomaly-based techniques for web attacks detection,” *J. Appl. Comput. Res.*, vol. 1, no. 2, pp. 111–120, 2012, doi: 10.4013/jacr.2011.12.06.

[38] G. Peyr, “Mathematical Foundations of Data Sciences,” 2018.

[39] H. K. Kim, “Car-Hacking Dataset,” *Hacking and Countermeasure Research*

Lab. https://sites.google.com/a/hksecurity.net/ocslab/Datasets/CAN-intrusion-dataset (accessed Dec. 01, 2020).

[40] U. Klehmet, T. Herpel, K. S. Hielscher, and R. German, “Delay bounds for CAN communication in automotive applications,” in *14th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems*, 2008, pp. 1–15.

5 Conclusions and Future Work

This PhD research focused on the security of the in-vehicle CAN network. The thesis has three main contributions. First, it has summarised the literature and presented vulnerabilities of CAN communication and state-of-the-art research targeting those vulnerabilities. This will help researchers in the field to grasp the current state-of-the-art. The second contribution is the testing framework that assesses performance considering resource usage and timing behaviour. By creating a framework, this thesis achieves a standard evaluation methodology for in-vehicle IDS. The final contribution of the thesis is the wavelet-based intrusion detection system WINDS.

This chapter summarises the thesis and shows the contributions out of this PhD research. Section 5.1 presents how the research aim and objectives are achieved. After that, the chapter is finalised with the future research direction for WINDS and CAN bus security for vehicular applications.

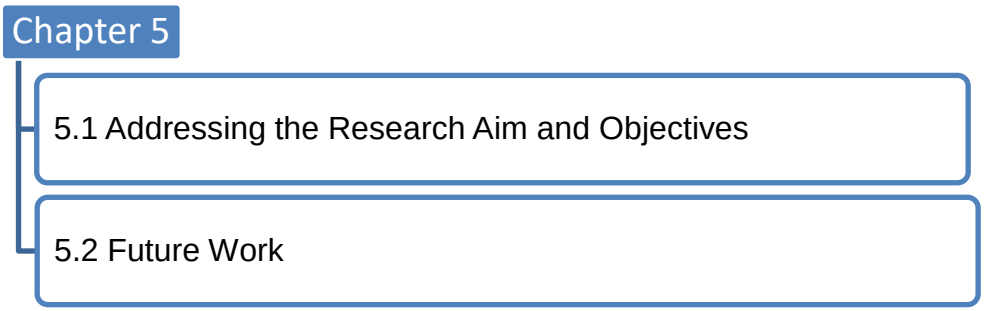

Figure 5-1 Organisation of Chapter 5

5.1 Addressing the Research Aim and Objectives

This research aims to develop a vehicle-independent intrusion detection for Controller Area Network. The research achieves this aim by fulfilling several objectives that are given below.

Objective 1: The literature review is the beginning of the research. A comprehensive literature review was carried out to identify gaps and problems regarding CAN security. The vulnerabilities of the CAN bus are shown and some of the attacks are presented in Section 2.4. A case study of the DoS attack via

hardware trojan is demonstrated in our paper [1]. Section 2 also presents state-of-the-art solutions with their limitations. The findings are presented in a conference [2] and an extended version of it [3] is issued in a journal.

Objective 2: As presented in Section 2.5.4 and Section 4.2.2, the current in-vehicle IDSs have various problems and almost all of them are vehicle dependent. Therefore, designed IDS should be trained for each vehicle make and model. Training IDS for every different vehicle requires an extensive amount of work. After training and modification, each subsequent IDS should be tested thoroughly again. A vehicle agnostic wavelet-based solution (WINDS) is proposed to overcome this issue. The WINDS analyses the short-time history of CAN traffic by applying wavelet analysis and indicates the change points in the network traffic. The algorithm can be applied to any vehicle without any modification. It is tested on multiple datasets with various vehicles to show WINDS' effectiveness and vehicle independent behaviour. The results show that WINDS has competitive performance with state-of-the-art solutions. Apart from the performance, WINDS is also assessed for resource usage and timing behaviour according to the testing framework presented in Section 3.2.

Objective 3: The reliability of the testing depends on a comprehensive dataset consisting of different attack models and strengths. If the dataset is not complete, it can produce misleading results. Although the existing datasets are valuable, they are not enough to test an IDS and generate statistical results. Section 3.3 address the lack of data problem with synthetic data generation. Existing attack models are explained and presented with their implementation. The implementations of some attacks are stealthier than the existing ones, which allows more reliable testing. The timing of the attacks are also considered and attacks are implemented at various durations. Applying synthetic attacks to open-source datasets produce a more comprehensive benchmarking dataset as presented in Section 3.3.2 and tested in Section 4.4.2.

Objective 4: The number of IDS for CAN bus increases exponentially; however, there is no standardised testing methodology. As a consequence, many IDSs are not tested properly. It also makes it difficult to compare results with other existing

solutions. Section 3 presents a framework to test an IDS for in-vehicle communication thoroughly. It aims to solve the comparison problem for CAN bus IDSs. As a case study, the proposed framework is applied to WINDS and results are shown in Section 4.

5.2 Future Work

The time and resource restriction of this research limits to further improve the obtained results and secure in-vehicle communication. Keeping this research work in perspective, we recommend the following as significantly important future works.

5.2.1 Integration of Encryption

Although IDS can alert the malicious messages, it cannot prevent eavesdropping; therefore, it cannot provide confidentiality. To overcome eavesdropping, data should be encrypted. Many secure encryption techniques can solve eavesdropping in the IT domain, but none is feasible to be implemented on a resource-constrained CAN network. A light-weighted encryption technique can solve the confidentiality problem in CAN.

5.2.2 Application to Other In-vehicle Network Protocols

The WINDS is designed mainly for the CAN network. However, other in-vehicle communication protocols are also posing a threat to the security of the in-vehicle network. To have holistic security, these network protocols should be protected too. This requires further investigation to adapt WINDS to those protocols.

5.2.3 Machine Learning Implementation

The vehicle agnostic behaviour of the WINDS can be enhanced with machine learning that allows WINDS to interpret data and use it to learn for itself. Implementing lightweight machine learning techniques will enable WINDS to automatically learn and improve from experience without being explicitly programmed.

5.2.4 Intrusion Prevention System (IPS)

The WINDS is designed as Intrusion Detection System (IDS), which only alerts the problems. However, identifying a problem is half the battle; knowing how to respond appropriately and having the resources in place to do so is equally important. As a vehicle is a cyber-physical system, implementing IPS will be possible if the false positive rate decreases to zero. When WINDS is capable of zero FPR, it should take action and invalidate any unauthentic messages. Invalidation of messages also requires further investigation.

References

[1] M. Bozdal, M. Randa, M. Samie, and I. Jennions, “Hardware Trojan Enabled Denial of Service Attack on CAN Bus,” 2018, vol. 16, pp. 47–52, doi: 10.1016/j.promfg.2018.10.158.

[2] M. Bozdal, M. Samie, and I. Jennions, “A Survey on CAN Bus Protocol : Attacks , Challenges , and Potential Solutions,” in *IEEE International Conference on Computing, Electronics & Communications Engineering (iCCECE ’18)*, Aug. 2018, pp. 201–205, doi: 10.1109/iCCECOME.2018.8658720.

[3] M. Bozdal, M. Samie, S. Aslam, and I. Jennions, *Evaluation of can bus security challenges*, vol. 20, no. 8. MDPI AG, 2020, p. 2364.

Appendices

Appendix A Source Codes for WINDS Implementation and Testing

A.1 Source Code for Generating Attacks

The **attackGenerator** function generates DoS, replay, and suspension attacks. It has four parameters; canData, attackType, vehicleM, attackDuration.

canData: The raw CAN traffic

attackType: The choice of attack type from DoS, replay, suspension

vehicleM: The vehicle model from the existing dataset, which includes Opel and Renault

attackDuration: The implement4ed attack duration.

The example usage of the function is presented below:

attackData= attackGenerator(testing,'dos','r',5);

The code line above implements five seconds of DoS attack on Renault on the testing dataset.

Table - A1: Source code for attackGenerator function

```
function attackData =
attackGenerator(canData,attackType,vehicleM,attackDuration)

    attackMultiplier = attackDuration;

    switch attackType
        case 'dos'
            attackData = dosAttack(canData,attackDuration,vehicleM);
        case 'replay'
            attackData = replay(canData,vehicleM,attackMultiplier);
        case 'suspension'
            attackData = suspension(canData,attackDuration,vehicleM);
    end

end

%% DoS Attack Generation %%
function outputDoS = dosAttack (canData, attackDuration,vehicleM)
```

```
    rawData = canData;

    if vehicleM== 'r'%renault
        attackStart = 1508687506.000236;
    elseif vehicleM == 'o' %opel
        attackStart = 1536574995.000091;
    end

    attackEnd = attackStart + attackDuration;%1508687515.999845;
    attackPeriod = 0.00025; %unit is second - attack data period

    outputBefore= rawData(rawData.time<attackStart,:);
    outputAfter= rawData(rawData.time>attackEnd,:);

    % attack creation
    time = attackStart:attackPeriod: attackEnd;
    id = zeros(size(time));
    data = zeros(size(time));

    outputAttack = table( time',  id' , data');

    outputAttack.Properties.VariableNames{1} = 'time';
    outputAttack.Properties.VariableNames{2} = 'id';
    outputAttack.Properties.VariableNames{3} = 'data';

    outputAttack.data = string(outputAttack.data ); %make same type

    %combining attack with attack-free data
    outputDoS = [outputBefore;outputAttack;outputAfter];

end

%% replay Attack Generation %%
function outputReplay = replay(canData,vehicleM,attackMultiplier)

    rawData = canData;

    if vehicleM== 'r'%renault
        CANid = '2C6';
        attackStart = 1508687499.839714;
        attackEnd = 1508687499.905626;
        normalPeriod = 0.02;
    elseif vehicleM== 'o' %opel
        CANid = '1A1';
        attackStart = 1536575013.172200;
        attackEnd = 1536575013.247372;
        normalPeriod = 0.025;
    end


    outputBefore= rawData(rawData.time<attackStart,:);
    attackRaw = rawData(rawData.time>=attackStart &
rawData.time<=attackEnd ,:);
    outputAfter= rawData(rawData.time>attackEnd,:);
```

```
    attackPeriod = normalPeriod/attackMultiplier;
    attack = attackRaw(attackRaw.id==CANid ,:);

    for i=1:size(attack,1)
        attackManipulation((i-1)*attackMultiplier + 1,:) =
attack(i,:);

        for k = 1:attackMultiplier-1
            rowAttack = attack(i,:);
            rowAttack.time =  rowAttack.time + attackPeriod;
            attackManipulation((i-1)*attackMultiplier + 1 + k,:) =
rowAttack;
        end

    end

    noAttack = attackRaw(attackRaw.id~=CANid ,:);

    outputAttack = [attackManipulation; noAttack];
    outputAttack = sortrows(outputAttack,'time','ascend');

    outputReplay = [outputBefore;outputAttack;outputAfter];

end

%% suspension Attack Generation %%
function outputSuspension =
suspension(canData,attackDuration,vehicleM)

    if vehicleM== 'r'%renault
        CANid = '2C6';
        attackStart = 1508687499.999696;
    elseif vehicleM== 'o' %opel
        CANid = '1A1';
        attackStart = 1536575000.000097;
    end

    rawData = canData;
    attackEnd = attackStart + attackDuration;%1508687510.000100;

    outputBefore= rawData(rawData.time<attackStart,:);
    attackRaw = rawData(rawData.time>=attackStart &
rawData.time<=attackEnd ,:);
    outputAfter= rawData(rawData.time>attackEnd,:);

    outputAttack = attackRaw(attackRaw.id~=CANid ,:);

    outputSuspension = [outputBefore;outputAttack;outputAfter];

end
```

A.2 Source Code for WINDS

The code to implement WINDS can be found in Table A-2.

Table - A2: Source code for WINDS implementation

```
load('rawRenaultClio.mat')

global sampleTime
global waveletModel
global coefficient
global waveletLevel

global attackType

sampleTime = 0.003;
waveletModel = 'haar';
coefficient = 1.8;
waveletLevel = 16;


for k = attackType:attackType

    if k==1 %dos
        rawData = attackData;
        attackDuration
        aStart = 1508687506.000236+ 0.022;
        aEnd = 1508687506.000236 + attackDuration - 0.022;
    elseif k==2 %replay
        rawData = attackData;
        aStart = 1508687499.839714;
        aEnd = 1508687499.905626;
    else %suspension
        attackDuration
        rawData = attackData;
        aStart = 1508687499.999696 + 0.03;
        aEnd = aStart + attackDuration-0.05 ;
   end

    rawData = rawData(rawData.id == '2C6',:);

    sampledSignal = frequencyConvertion(rawData,sampleTime);
    ofsetTime = 1508687476.43810;
    attackStart = ceil ((aStart - ofsetTime) / sampleTime ) ; %
    attackEnd   = floor ((aEnd - ofsetTime ) / sampleTime ) ;

[accuracy,sensitivity,specificity,TP,TN,FP,FN,TPLocTemp,TNLocTemp,FPLo
cTemp,FNLocTemp ] = realTimeFunction
(sampledSignal,attackStart,attackEnd );
    TPLoc(k,1:length(TPLocTemp))=TPLocTemp;
    TNLoc(k,1:length(TNLocTemp))=TNLocTemp;
    FPLoc(k,1:length(FPLocTemp))=FPLocTemp;
    FNLoc(k,1:length(FNLocTemp))=FNLocTemp;
end
```

```
precision = TP/(TP+FP);
FPR = 1 - specificity;
```

The **realTimeFunction** code applies the WINDS algorithm to CAN traffic in real-time. The function has three parameters as below:

sampledSignal: The raw CAN traffic to be processed

attackStart: The start time of the attack

attackEnd: The end time of the attack

As a result, the function generates the following outputs:

Accuracy: Accuracy of the test

Sensitivity: Sensitivity of the test

Specificity: Specificity of the test

TP: True positive number of the test

TN: True negative number of the test

FP: False positive number of the test

FN: False negative number of the test

TPLoc: True positive location

TNLoc: True negative location

FPLoc: False positive location

FNLoc: False negative location

Table - A3: Subfunction of WINDS main file "realTimeFunction"

```
function
[accuracy,sensitivity,specificity,TP,TN,FP,FN,TPLoc,TNLoc,FPLoc,FNLoc]
= realTimeFunction (sampledSignal,attackStart,attackEnd)

global sampleTime
global waveletModel
global coefficient
global waveletLevel

    shiftSize = 1;
    windowSize = 127;

    %counters
    attackDetectionNumber = 0;
    TP = 0;
    TN = 0;
    FP = 0;
    FN = 0;

    %values
    attackLoc = 0;
    TPLoc = 0;
    TNLoc = 0;
    FPLoc = 0;
    FNLoc = 0;

    delayCalDetector=1;

    for  x = (windowSize+1) : shiftSize: (length(sampledSignal) -
windowSize)

        windowedSignal = sampledSignal ( (x- windowSize): x );
        wtSignal=
abs(cwt(windowedSignal,1:waveletLevel,waveletModel));

        wtSignal = mad(wtSignal);
        thresholdUp = mean(wtSignal)*coefficient;
        lctUp = find(wtSignal(1:end-1)>thresholdUp);
        lctDown = find(wtSignal(1:end-1)==0);

        attackDetection = ~isempty(lctUp) ||~isempty(lctDown);

        if attackDetection
            attackDetectionNumber = attackDetectionNumber +1;
            attackWindow(attackDetectionNumber) = x;
        end

        if (attackStart) > x  || (x-windowSize )>attackEnd   % before
any attacks start or after all attacks
            if ~attackDetection %no attack detection
                TN = TN + 1; % True Negative
                TNLoc(TN) = x;
            else  %attack detection
```

```
                FP = FP + 1; %False Positive
                FPLoc(FP)= x;
            end

        else
            if attackDetection %attack detection
                TP = TP + 1; %True Positive
                TPLoc(TP) = x;
                if delayCalDetector
                    disp(x)
                    disp(attackStart)
                    disp(['Time Delay = ',num2str(x-attackStart)])
                    delayCalDetector = 0;
                end
            else     %no attack detection
                FN = FN + 1;  % False Negative
                FNLoc(FN)= x;
            end
        end

    end

    accuracy = (TP+TN)/(TP+FP+TN+FN) ;

    sensitivity = TP / (TP+FN);

    specificity = TN / (FP+TN);
end
```

---

frequencyConvertion function takes raw CAN data as a table and calculates the number of messages at the given time duration. The following example code will convert rawCAN data into message count in every 0.003 seconds.

f = frequencyConvertion(rawCan,0.003);

Table - A4: Subfunction of WINDS main file "frequencyConvertion"

```
function messageInTime = frequencyConvertion(tableIn,timeDuration)

    temp = tableIn;
    temp = sortrows(temp,'time','ascend');
    temp.time = temp.time - temp.time(1); %initial time become zero

    vectorSize = floor (temp.time(end) ./ timeDuration) + 1; % size of
matrix
    messageInTime = zeros(1,vectorSize);
```

```
    % scan matrix from the beginning to the end
    pointerTime = timeDuration;
    pointer = 1; % point the current position

    for i=1:vectorSize -1

       while temp.time(pointer) < pointerTime
           messageInTime(1,i) = messageInTime(1,i) + 1;
           pointer = pointer + 1;

           if pointer == size(temp,1)
               break
           end
       end

       pointerTime = pointerTime + timeDuration;

    end

end
```

A.3 Source Code for Testing WINDS

This code generates multiple attacks automatically and applies the WINDS algorithm to the attacked dataset. Then it produces statistical results for the tests.

Table - A5: Source code for Testing WINDS on multiple datasets

```
global attackType

load('rawRenaultClio.mat')

attackType =1; %dos
for i=1:10
    attackDuration =i;
    attackData= attackGenerator(testing,'dos','r',attackDuration);
    testRenault
    accD(i)=accuracy;
    senD(i)=sensitivity;
    speD(i)=specificity;
    preD(i)=precision;
    falPoRaD(i) = FPR;
end

accD(i),senD(i),speD(i),preD(i),falPoRaD(i)

%% replay
```

```
attackType =2;

for i=3:12
    attackDuration =i;
    attackData= attackGenerator(testing,'replay','r',attackDuration);
    testRenault
    accR(i-2)=accuracy;
    senR(i-2)=sensitivity;
    speR(i-2)=specificity;
    preR(i-2)=precision;
    falPoRaR(i-1) = FPR;
end

accR(i),senR(i),speR(i),preR(i),falPoRaR(i)

%% Suspension

attackType =3;

for i=1:10
    attackDuration =i;
    attackData=
attackGenerator(testing,'suspension','r',attackDuration);
    testRenault
    accS(i)=accuracy;
    senS(i)=sensitivity;
    speS(i)=specificity;
    preS(i)=precision;
    falPoRaS(i) = FPR;
end

accS(i),senS(i),speS(i),preR(i),falPoRaS(i)
```