San Jose State University
SJSU ScholarWorks

Master's Projects | Master's Theses and Graduate Research

Spring 5-24-2021

Defending Vehicles Against Cyberthreats: Challenges and a Detection-Based Solution

Qilin Liu

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

 Part of the Artificial Intelligence and Robotics Commons, and the Information Security Commons

Defending Vehicles Against Cyberthreats: Challenges and a Detection-Based Solution

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

Qilin Liu

May, 2021

© 2021

Qilin Liu

ALL RIGHTS RESERVED

The Designated Project Committee Approves the Project Titled

Defending Vehicles Against Cyberthreats: Challenges and a Detection-Based Solution

by

Qilin Liu

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

San José State University

May 2021

Dr. Robert Chun	Department of Computer Science
Dr. Ben Reed	Department of Computer Science
Dr. Jesse Geneson	Department of Mathematics

ABSTRACT

The lack of concern with security when vehicular network protocols were designed some thirty years ago is about to take its toll as vehicles become more connected and smart. Today as demands for more functionality and connectivity on vehicles continue to grow, a plethora of Electronic Control Units (ECUs) that are able to communicate to external networks are added to the automobile networks. The proliferation of ECU and the increasing autonomy level give drivers more control over their vehicles and make driving easier, but at the same time they expand the attack surface, bringing more vulnerabilities to vehicles that might be exploited by hackers. Possible outcomes of a compromised vehicle range from personal information theft to human life loss, raising the importance of automotive cybersecurity to a whole different level. Therefore, network safety has become a necessary and vital consideration of a vehicle. This project is two-fold: the first half will focus on the background of vehicle cybersecurity, characteristics of vehicular networks that could be leveraged during a hacking process, including ECU, Controller Area Network (CAN bus) and On-Board Diagnostics (OBD). It also discusses and evaluates previous hacking experiments conducted by researchers and their proposed countermeasures. The second half is an evaluation of approaches to design an Intrusion Detection System (IDS). The aim of this project is to find an effective and suitable solution to defend vehicles against various types of cyber threats.

Keywords **- vehicle cybersecurity, Electronic Control Unit (ECU), Controller Area Network (CAN bus), On-Board Diagnostics (OBD)**

TABLE OF CONTENTS

List of Tables

List of Figures

I. INTRODUCTION

After the first Internet-connected car was brought to market in late 1990s, automobiles were no longer local networks isolated from the web. Today vehicles are gradually transforming from purely mechanical entities to digital entities as innovative features are deployed on vehicular networks one after another. From conventional cars to connected cars, then to smart cars and autonomous cars, our vehicles are becoming smarter and more integrated, revolutionizing the way people drive. In 2016, the size of the smart car market around the world was estimated to be approximately 20 billion USD and predicted to grow by 6.7% every year [1]. While people benefit from the convenience of Internet connectivity and driving assistance inside their cars, they are also becoming potential targets of cyber threats. In fact, automobile hacking tests had been conducted in various countries that successfully operated steering wheels and changed engine velocity remotely [1]. It has been a well-established fact that any one of the cars on the road could be attacked anytime without the driver noticing and leave little forensic evidence behind [2]. The standardization of V2V/V2I communication technologies and the increasing reliance on sensors and intercommunications will potentially bring more vulnerabilities to vehicles and further exacerbate the security problem. Therefore, automotive network security has a strong potential to become a serious issue, which concerns and threatens more and more drivers and automakers around the world.

Although the power of cars keeps evolving, their general structures have changed little since the 1980s. A car consists of hundreds of ECUs that control different functionalities and aspects of the car. These ECUs primarily use the Controller Area Network (CAN bus) protocol to communicate with each other as well as outside networks. Thus, once attackers successfully hack

into a car's CAN network, they would have access to the ECUs on the vehicle and be able to hijack the vehicle regardless of the driver's action, resulting in life-threatening consequences. The first half of this project will provide a comprehensive and systematic overview of vehicle structure and vehicular cyber attacks, which is needed to answer the research questions: What are the most possible and viable ways to attack a vehicular network, and how could automotive networks be protected from these attacks? It also provides the necessary background and perspective on the rationale to the second half of the project, which is an implementation of IDS based on One-Class Support Vector Machine (OCSVM).

This project is based on published papers, journals and articles in the computer science area. It includes the following sections: Section II introduces the background of vehicle cybersecurity and demonstrates its importance and efforts made to address the issue. Section III explains the vehicle structures, components and their vulnerabilities in more detail, including ECU, CAN bus and On-Board Diagnostics (OBD). Section IV discusses possible attack scenarios and builds attack models by analyzing previous car hacking experiments. Section V lists the pros and cons of possible ways to defend against vehicular cyberattacks. Section VI elaborates on the proposed defense method and its advantages over other methods. Section VII provides the details of the experiment to implement and evaluate the proposed method. Lastly, the review is concluded in Section VIII.

II. BACKGROUND

A. Problem analysis

At first blush, one might consider vehicle cybersecurity to be an ordinary security problem. However, vehicular cybersecurity faces many unique challenges that are not well-addressed in typical computer networks, and they could bring severe and irreversible consequences if not properly dealt with. To elaborate:

- Besides information and identity theft, it could even threaten drivers', passengers' and pedestrians' physical well-being.
- Drivers could do little to prevent themselves from being hacked, so the responsibilities to protect them fall onto car makers' shoulders.
- If a vulnerability is found, normal computer systems can easily release a patch to repair it. On the other hand, once manufactured, cars are expected to be operable for decades, and their updates are not as fast and frequent as computers. Therefore vehicle security systems must be designed to be resilient to novel attacks and avoid obsolescence over its long lifecycle.
- Considering the fact that automotive security incidents are not frequently occurring yet, solutions that would greatly increase the overall cost of vehicles should be avoided. The stringent cost requirement makes most sophisticated protection systems infeasible.

B. Hacking cases

In the past decades, the security of vehicles was ensured primarily through an approach called "security by obscurity", which means to keep the implementation details private and

prevent anyone from fully understanding the system. Today, many hardware engineers are still under the preconceived notion that people have neither the ability nor the motive to hack a car. However, this no longer applies in today's automotive industry. Systems and devices that were once too complex for anyone to understand have become hackers' common targets [14]. Corner-cutting designs inside vehicles, such as scripts that run as root on a removable card and protection of remote control with a default password of all-zeros, have also been found and exploited by hackers. On the other hand, recent evolutions of vehicular networks have made vehicle-related crimes more discreet and easier to conduct by allowing indirect and wireless access, thus removing the precondition of physical access to the vehicle. Some conceivable motives for vehicle cyberattacks include theft, electronic tuning, sabotage, intellectual property theft, privacy breach and intellectual challenge [12]. As proof, there have been a few cases related to vehicle hacking. For instance, in 2012 there were reports of missing luxurious BMW vehicles that turn out to be stolen by unauthorized access to on-board computers and programming of blank smart keys. There has also been an incident where a vulnerability in a web-based vehicle immobilization system was exploited, resulting in hundreds of cars in a city becoming unresponsive and their horns blaring wildly [15]. Evidently, the wide barrier of entry around vehicle systems that attempts to provide vehicle cybersecurity cannot defend against the rising vehicle attacks much longer.

C. Related regulations

In Europe, the regulatory bodies within the European Union are very active in the promotion of vehicle cybersecurity. Aside from the existing General Data Protection Regulation,

the European Commission and European Union Agency for Network and Information Security (ENISA) have a specific regulation on cybersecurity across Europe. It includes a certification scheme that issues certifications to vehicles based on the ranking a vehicle receives during an accredited assessment [21]. The US government has also begun to realize the importance of enforcing vehicle cybersecurity through regulations. In March 2016, a warning about the growing vulnerability of vehicles to remote cyberattacks was given out in a public service announcement by the National Highway Traffic Safety Administration, the US Department of Transportation and the Federal Bureau of Investigation [19]. In the following month, two bills that give life sentences to people who hack into in-vehicle networks were proposed by the Michigan state senate [19]. In 2017 the US Congress and Senate passed the Safely Ensuring Lives Future Deployment and Research In Vehicle Evolution (SELF DRIVE) Act which takes an approach for cybersecurity similar to Europe's ranking system.

III. VEHICULAR NETWORK

A. Connected car

In the US and Europe, a connected car or a smart car is defined as a vehicle that makes use of information gathered from internal systems and the external environment to provide useful driving assistance and improve convenience and safety. It is considered a step toward the eventual automated self-driving car [1]. As a major addition to the world of the Internet of Things (IoT) and a fundamental part of the smart city vision, the need for modern vehicles' connectivity and automation grows rapidly. By 2016 a typical connected car already had more than 75 sensors, 150 actuators and 4000 signal data, while generating 25 gigabytes of data per hour which are analyzed by 70 on-board computers (OBCs) [1]. There have also been mobile applications and devices that enable remote monitoring and diagnosing of vehicles. They are designed to enhance users' knowledge about their cars, but on the other hand, they bring security challenges to the in-vehicle networks by introducing remote access to the in-vehicle network. Some researchers argue that vehicles are not ready for the potential attacks it attracts. Thus, it is important to evaluate the tradeoffs this technology has brought about.

As shown in Figure 1, the vehicles are now at the intersection of myriads of connections, each of which is a potential entry point to the car [8]. The connections on a vehicle include, but are not limited to:

- OBD-II port used to access embedded microcontrollers,
- Vehicle-to-Vehicle (V2V) communications formed with the VANET network that allows for aid in effortless driving and automatic response to imminent danger,

- Vehicle-to-Infrastructure (V2I) communications allowed by interacting with Roadside Units, which are installed at critical points on the roads to enhance driving reliability,
- Interconnections with smartphones and tablets through auto platforms such as Apple's CarPlay and Google's Android Auto to enable hands-free access to mobile services,
- Connection with 4G and 5G mobile networks to enable over-the-air firmware updates, satellites and Global Positioning System (GPS) services,
- And finally, physical connections via CD player and USB sockets.

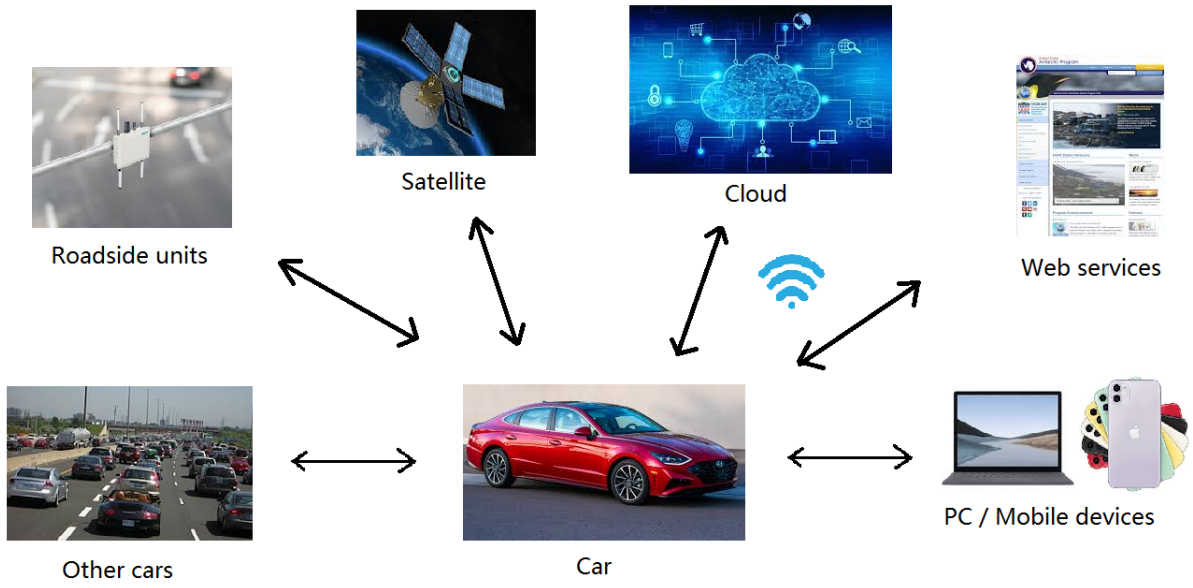

Fig. 1. Interconnections of a connected car

B. Electronic control units

ECU is an embedded system that controls one or more of the modules and subnetworks on a vehicle. Modern vehicles usually comprise more than fifty ECUs, including telematics, airbag control unit, engine control unit, transmission control unit and antilock braking system. These ECUs are responsible for runtime monitoring and accurate control over the vehicle

systems such as fuel injection, air intake, user interface, infotainment, engine control and cruise control systems [4]. These functions are facilitated by reading data from sensors, performing calculations and then setting actuator variables, and thus, fast and safe communications amongst the ECUs are of vital importance. Engine Control Module is a special type of ECU, which specifies and controls the real-time parameters of the engine unit. An ECU's parameters are carefully tuned by engineers, and its undisturbed operation is essential to a vehicle's normal behavior on the road [3][5]. Nowadays, the count and complexity of ECUs on a vehicle continue to increase, bringing more functionalities and control over the vehicle [1]. However, this also increases the possibility of being hacked and increases the severity of the outcomes, as a flaw of any ECU could be leveraged by hackers, causing the whole vehicular security system to crumble and possibly lead to an accident. Most ECUs only have a simple challenge-response authentication scheme to prevent unauthorized access, which could either be cracked in a brute-force manner or bypassed under an integrity attack [14]. The attacker could then masquerade as the compromised ECU and launch attacks on other ECUs. Although the in-vehicle network is segregated into subnetworks based on the ECUs' functionalities, the gateways between subnetworks are basically filters that allow only predefined message types to pass through, so they will relay any message that possibly interests any ECU in the same subnetwork. Thus, gateways can do little to prevent the attacker from gaining full control over the vehicle's functionality. Furthermore, the attacker could even launch attacks on the vehicle's surrounding environment and cause greater damage if V2V/V2I communications are supported.

C. Controller area network

The major in-vehicle subnetwork protocols are CAN, FlexRay, Media Oriented System Transport (MOST) and Local Interconnect Networks (LIN). Their utilities and characteristics are listed in Table I [8], and an example structure of a typical in-vehicle network is shown in Figure 2. Among them, FlexRay has not been widely adopted due to its high cost, and MOST and LIN are designated for non-critical subnetworks. CAN is the most ubiquitous and also the most defenseless protocol in vehicles, and thus, is focused on in this paper.

TABLE I
SUBNETWORK PROTOCOLS IN VEHICLES

Name	Description	Bandwidth
CAN	The predominant network protocol in vehicular networks, divided in high and low speed buses	CAN high: 500 Kbit/s CAN low: 125 Kbit/s
FlexRay	Designed as a replacement of CAN with higher speed, but its higher cost hinders its use	10 Mbit/s
MOST	Offers synchronous and asynchronous data channels, used by infotainment systems	24 Mbit/s for synchronous transmission 14 Mbit/s for asynchronous transmission
LIN	Use a master-slave model, provide low-cost communication between sensors and actuators	20 Kbit/s

The CAN bus protocol is a network protocol developed by Robert Bosch GmbH to replace the previously hard-wired communication systems. It was released in 1986, and despite its age, it is still the de facto standard of all vehicles in the US [4]. Due to its comparably low cost and optimized power to deliver short and periodic messages, its usage even extends to telecommunication backup power systems and robots [7]. CAN bus connects all the ECUs on a vehicle in a fashion similar to how a household electric circuit connects home appliances,

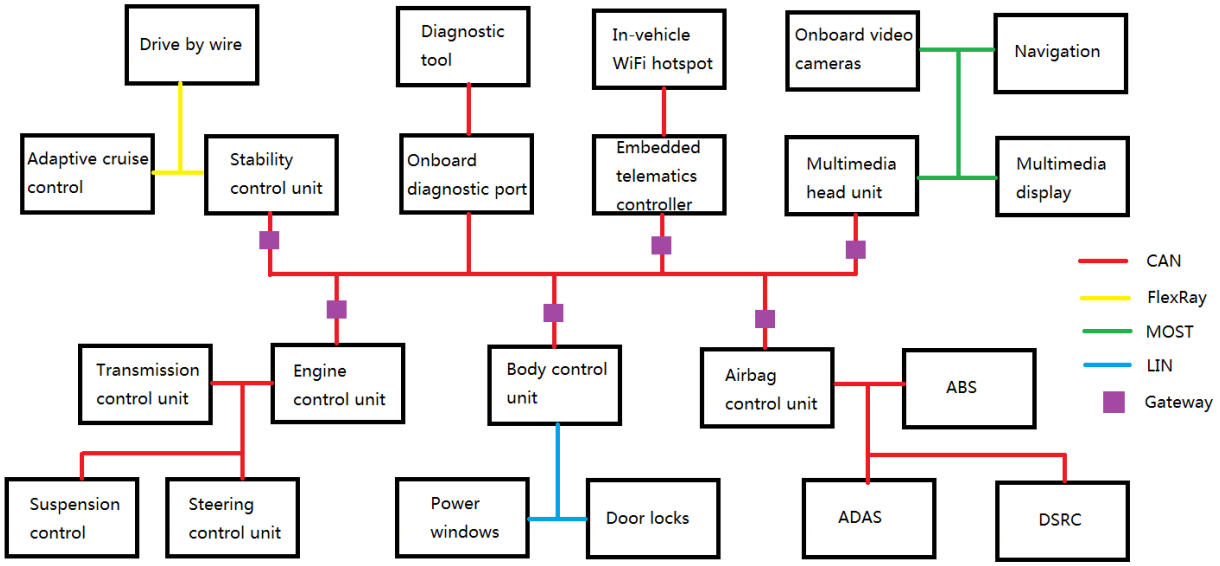

Fig. 2. Example structure of in-vehicle network

allowing them to be easily added to or removed from the network. The advantage of the CAN bus protocol is that it greatly reduced the cost and complexity of interconnection amongst different vehicle systems and made the configuration of ECUs easier [4]. However, it is old, and in the 1980's the Internet was not yet as prominent, let alone cyberattacks. As a protocol designed thirty years ago, CAN bus protocol naturally does not have any intrinsic encryption or security support, delegating this responsibility to the higher-level protocols. On the other hand, since CAN bus protocol is a real-time control protocol with stringent cost constraints, most of the existing security mechanisms for general applications are incompatible with vehicular systems, and must adapt to CAN bus protocol's specific constraints before applying [7]. The security levels of vehicles also differ dramatically between one model and another. Mid-priced cars are the most prone to cyber attacks, because they depend more on the CAN bus network than low-priced cars, but they do not have a strong security gateway like high-priced cars [3]. This could

be an implication of the fact that current security solutions are so expensive to deploy on a car that most car makers do not consider them practical and worthwhile.

TABLE II
STRUCTURE OF BASIC CAN FRAME

Name	Description	Length in bits
Start-of-frame (SOF)	Indicate the start of a frame	1
Identifier	The CAN ID of the sender ECU	11
Remote transmission request (RTR)	Indicate whether this message is a data frame or remote request frame	1
Identifier extension bit (IDE)	Indicate whether this message is in basic or extended frame format	1
Reserved bit	Must be dominant (0)	1
Data length code (DLC)	The number of bytes used in the data field	4
Data	The data transmitted	64
Cyclic redundancy check (CRC)	Detect hardware bit corruptions	15
CRC delimiter	Must be recessive (1)	1
ACK slot	Must be recessive	1
ACK delimiter	Must be recessive	1
End-of-frame (EOF)	Denote the end of a frame, must be recessive	7

There are two types of CAN frame format: basic frame format (CAN 2.0A) and extended frame format (CAN 2.0B). The format of the basic CAN frame is listed in Table II. The extended frame format is similar but with 29 identifier bits. Depending on the purpose, CAN messages can be categorized in four different types, which are explained in Table III [8]. Moreover, there is also a special type of message called diagnostic message, which always has a value greater than 0x700 as its CAN ID. Normally, diagnostic messages occur only when the vehicle is under the

diagnosis of vehicle mechanics, and they could be very powerful depending on the design of ECUs.

TABLE III
TYPES OF CAN MESSAGES

Message type name	Usage
Data frame	The most common message type, which broadcasts a data payload between nodes.
Remote frame	Requests the transmission of data from a particular node.
Error frame	Sent by a node that has detected an error in a message to ask the sender to retransmit that message.
Overload frame	Sent by a busy node to request an extra delay between messages.

CAN bus protocol has two major characteristics that pose inherent security issues. First, to maximize the number of messages transmitted in a time unit, which determines the performance of a car to a large extent, CAN messages are made to be as short as possible [7]. As a result, the receiver's field is omitted in CAN messages, and every message is broadcasted in the network for every ECU's acceptance filter to decide whether the message is intended for itself [3][8]. Second, a unique CAN identifier is assigned to every ECU on a vehicle, which also serves as its priority value. A CAN message contains the CAN ID of the sender, and if it collides with another CAN message, then the message with the higher CAN ID has to yield and resend at a later time. Because of these characteristics, CAN bus protocol cannot guarantee the following security properties, which could potentially be exploited by hackers [12]:

- Confidentiality: a faulty node can easily tap the CAN network and decrypt the meaning of each message.

- Authenticity: a node could send packets with any CAN ID, making it impossible to verify the sender of a packet.
- Integrity: likewise, a node could also send packets with falsified data contents.
- Availability: a Denial-of-Service (DoS) attack is plausible by bombarding the CAN bus with messages with low CAN IDs, or high priority values, forcing all other messages to stall and disabling the CAN bus [3]
- Non-repudiation: there is no way for an authentic ECU to prove whether or not it has sent or received a certain message.

D. On-board diagnostics

The term OBD refers to the self-diagnosing capabilities of vehicles. Since 1996, it has been mandatory for every car sold in the US to provide an OBD-II diagnostic connector. An OBD socket is usually located below the steering wheel and above the brake, and can be connected to various diagnosis tools like OBD dongles and scanners to listen to the CAN bus and inspect ECUs by means of the ELM327 functions. Figure 3 shows the possible usages of an OBD port. While OBD connectors allow repair technicians to easily diagnose cars, the wide range of ECU control commands stored in the tools also make them exploitable to hackers who could access the CAN bus and manipulate ECUs in unethical ways. A host of tools that are publicly purchasable in automotive stores and online allow the user to extract and decipher CAN bus traffic or inject arbitrary messages into the CAN bus to manipulate target units [3]. These tools once required a wired connection to a computer terminal, but now, most of them provide remote access via Bluetooth, 3G/4G or Wi-Fi connections, and are easy to be carried and

concealed [3][6]. It has been reported that about half of the investigated OBD dongles have exposed, weak or no encryption key, and thus can be utilized for OBD injection attacks. These tools greatly lower the threshold and cost of vehicle hacking [6].

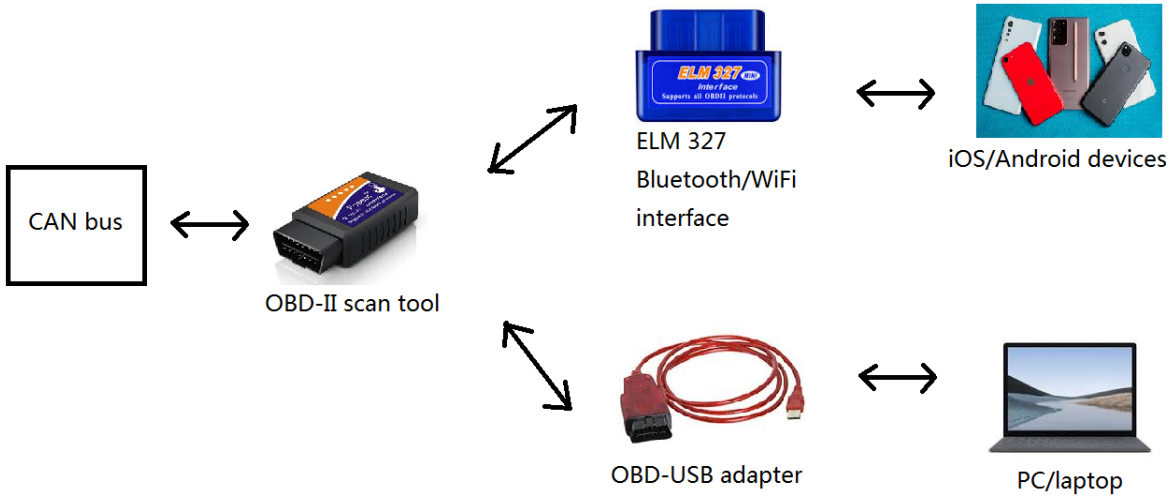

Fig. 3. Usages of an OBD port

IV. ATTACK MODEL

A. Hacking experiments

Establishing the attack model is crucial to learn the attacker's capabilities and effectively design defenses. In the last decade, many car hacking experiments have been mounted and documented to showcase the vulnerabilities of automotive networks. To give a flavor of the wide spectrum of hacking methods, some representative hacking experiments conducted in the past are summarized below:

- In 2015 Miller and Valasek found a vulnerability in the head unit of a 2014 Jeep Cherokee [2]. They demonstrated and published in detail how to exploit this vulnerability to remotely compromise the vehicle. Malfunctions were caused to a 2014 Jeep Cherokee by reprogramming a gateway chip inside the car's head unit and making it send random CAN messages. This hack eventually caused the Jeep company to recall millions of their Jeeps.
- In [3], the control over lights, horn, dashboard, lock and steering of different models of cars was gained by injecting CAN messages through the OBD interface.
- In [5], the throttle control, gear state and brake of a Renault Twizy 80 were remotely controlled by tampering with the vehicle's main ECU, the Sevcon Gen4 controller. It was achieved by finding the passcode of the Sevcon Gen4 with brute force, which only took a matter of hours because the passcode is only two bytes long.
- In [9], the Advanced Driver-Assistance Systems (ADAS) of a car produced in 2018 was successfully hacked and manipulated to remotely accelerate, brake or steer the car.
- In [17], a man-in-the-middle attack was conducted on a simulated CAN bus formed by a BMW E90 (3 Series) instrument cluster, an Arduino MEGA 2560 board and a CAN bus

shield. A rogue device was connected to the CAN bus and was able to intercept CAN packets, modify its content and send them to the rest of the CAN bus without being detected.

- In [20], a live demonstration of an attack experiment is reported in a television news program. The experimenters used a malicious mobile app written by themselves to remotely control a mid-size car. They were able to change the dashboard, steering and acceleration of the car. The video clip of the live experiment is still available at an URL provided in their paper.

The successes of these experiments revealed that car hacking is not a mere academic theory, but an activity that anyone with some knowledge about cars might be able to do. Despite that, little research has been done in recent years due to a large knowledge gap between automotive cybersecurity and other computer science fields [2]. On the other hand, automakers are seeking profit in developing higher-level smart and autonomous cars, while the development of security systems is being outpaced or even neglected. If cybersecurity requirements and standards are not established early in the system design lifecycle and maintained throughout, then it will be even harder to fix security vulnerabilities in later design phases. Therefore, to address the automotive security challenges that are in the way of the deployment of fully autonomous vehicles, academic researchers and industrial engineers need to work together to reach an effective and efficient solution promptly.

B. Attack surface

According to previous hacking experiments, an attack usually needs to first break into the internal CAN network through a wired or wireless entry point. Modern vehicles have a broad

range of access points where unauthorized access could enter through. Some access points that are likely to be targeted are discussed in table IV [12].

TABLE IV
POSSIBLE ENTRY POINTS OF IN-VEHICLE NETWORK

Category	Access point	Description
Indirect Access	OBD port	An OBD adapter used for diagnosis could be left plugged in the OBD port, and any device nearby could pick up its WiFi or Bluetooth signal and connect to the in-vehicle networks (in fact “fit and forget” is a feature OBD adapter sellers boast about). An unprotected OBD dongle could also be rewritten to emit malicious CAN packets when plugged in an OBD port.
	CD player	A malicious music file could be either written on a CD/USB or downloaded from the Internet. The decoding of Windows Media Audio (WMA) files might be exploited to emit malicious packets while the music is played.
Short range attacks	Mobile device connection	A mobile device paired with a vehicle through an auto platform could gain access to data stored in the communication unit, leading to database leakage and even ECU compromises [6].
	V2V/V2I communication	Bogus data packets could be sent through V2V/V2I communication channels and trick the vehicle into taking inappropriate emergency actions.
	Remote keyless system	Experimental attacks have been documented that successfully cracked or bypassed keyless entry systems and unlocked cars.
Long range attacks	Web browsing	Vehicles embedded with a web browser should keep an eye on browser targeted attacks such as buffer overflow and code injection.
	App store	Similar to iOS Appstore and the Play Store, some automakers already released their online App Stores providing auto application downloads. However, a malicious app could disguise as a car diagnostic app and fly under the radar, which could have large-scale consequences.

After the attacker gains the ability to send and receive packets to or from the vehicular networks, they could then start a reconnaissance phase by eavesdropping on the CAN traffic and infer information such as the ECUs in the network, the messages they send, the frequency of their packets, etc. They could also send arbitrary CAN messages to see the reactions of the

ECUs. Once enough information about the target vehicle's CAN network structure has been gathered, the real attack phase will be initiated.

C. Attack scenarios

After an attacker breaks through the defense system and gains a foothold in the CAN bus, the real attack could be launched in limitless ways. The most common types of attack include:

- Denial of Service (DoS) attack: a straightforward way to attack a vehicle is to bombard its CAN bus with injected packets. Because in case of a packet collision, the packet with a higher ID should always yield, flooding the CAN bus using packets with the lowest ID could easily disable the function of the network and bring devastating outcomes.
- Diagnosis attack: manipulate individual ECUs by forging and sending diagnostic messages. The severity of diagnosis attacks have been demonstrated in many hacking experiments. However, diagnosis messages are not supposed to appear in CAN bus during normal operation of a vehicle, so it is not hard to detect and defend against this type of attack.
- Replay attack: an attacker could perform some action on the car such as unlocking the car or turning on the radio, then record the CAN packets while this action is happening. This recording could be saved and replayed later to produce the same effect on any vehicle with the same model. By repeatedly trimming and replaying the recording, the recording's size could be reduced to as small as needed.
- Spoof attack: masquerade as other legitimate ECUs and send erroneous data packets to the CAN bus. This requires more prior knowledge about the vehicle's CAN implementation than a replay attack, but is also more powerful.

- Fuzzy attack: send random CAN messages to disrupt the vehicle's normal operation. Because precise control over the result cannot be achieved, fuzzy attacks are less severe compared to other types of attacks. Nonetheless, fuzzy attacks could temper with the safety constraints of the vehicular systems without any knowledge of the underlying structures, making them easy to carry out and more difficult to counter.
- Suspension attack: intercept or stop the transmission of one or more ECU's authentic messages, resulting in obliteration of packets in the vehicle CAN. This type of attack is rare but is relatively hard to defend.
- Malware: aside from direct attacks, malware may also crawl its way to vehicle systems, which could take the form of a virus, worm, Trojan horse, spyware, ransomware or rootkit [13]. If left untreated, malware could create an adversarial driving environment and greatly impact the vehicle's normal functionality.

V. DEFENSE MECHANISMS

Given the vast number of attack methods and the possible emergence of novel cyber-attacks, it is unrealistic to try to cope with every single type of attack. Securing wired and wireless access points is also important, but it is inefficient to make the broad communication interfaces perfectly hack-proof. To make the investment in vehicle security worthwhile and avoid raising the overall cost too much, a centralized solution that can thwart all types of attacks should be in our best interest. Most defense mechanisms proposed so far can be grouped into three categories: cryptography, IDS and cloud-based solutions. We will discuss the pros and cons of each category in detail.

A. Cryptography

Cryptographic solutions are a natural first thought to protect data from being sniffed or tampered. Cryptographic schemes such as encryption and Message Authentication Code (MAC) are devised to ensure data integrity and authenticity [7]. Some cryptographic schemes have been proposed in [11] and [20] where their ability to increase the difficulty to sniff and spoof the CAN bus traffic have been theoretically proved. However, cryptographic solutions also have their drawbacks. First, constant encoding and decoding will bring a lot of computation overhead to the resource-constrained CAN network, and the extra bits of the MAC tags will also decrease the data throughput. Second, normal encryption could do little against fuzzy attacks, because there will always be some random ciphertexts that can decrypt to a meaningful message and get past the error detection checks [7]. Although MAC is able to defend against fuzzing attacks, it has its key distribution problem, because it cannot make sure that the various components have the

secret shared key which is used to verify the MACs. Third, cryptographic solutions also necessitate a heavy modification to the structure of the CAN bus protocol, which will greatly increase the vehicle system complexity and cost time and resources for manufacturers and agencies to respond to this change. Therefore it should not be expected to happen in the near future and could only be saved as a candidate for a long-term solution.

B. Intrusion detection system

No matter how sophisticated an authentication system is, if it cannot stop attackers from trying repeatedly, then the system will be much more likely to be cracked given enough time. Thus, it is imperative to adopt an Intrusion Detection System (IDS) that is able to mitigate security risks by the means of detecting attacks and taking corresponding countermeasures. Unlike cryptography, an IDS only needs a monitoring controller that scans CAN traffic and distinguishes malicious from benign payloads. This does not require any change to the current CAN bus structure. Once an attack is detected, the driver should be notified and some predefined emergency methods should be provided to protect the driver and passengers. If the IDS is confident enough, it can also take safety measures on its own, such as ignoring packets from offending CAN ID, reconfiguring compromised ECUs or disabling inessential subnetworks.

There are two main challenges presented to the implementation of an IDS. First, false positives have a much greater impact on the feasibility of IDSs than other applications. Thanks to the high speed and heavy traffic in the CAN bus, even a 0.001% False Positive Rate (FPR) could cause the alarm to be triggered once every few minutes, rendering the IDS completely useless. On the other hand, if the sensitivity of the IDS is lowered to decrease the FPR, it would in turn

decrease the true positive rate (TPR), hindering its ability to detect anomalies. Therefore, an IDS is only practical if it can achieve a high TPR while keeping its FPR very close to zero. This constraint limited the practical use of many existing IDS designs, which we will discuss in the next section. The second challenge is that the specifications of ECUs are usually considered as intellectual property and are not shared by automakers. Thus, knowledge about the ECUs could only come through a process called reverse engineering, which needs to be done on each individual car model and is both time-consuming and painstaking. In the next section, a solution that addresses both challenges will be proposed.

C. Cloud-based solutions

To make up for vehicular systems' constraints, cloud-based solutions have been proposed by [13]. By delegating the responsibility to the cloud, which usually has more storage and computation power than a vehicle, more sophisticated detection mechanisms could be employed, such as storing the signatures of all known threats and vehicle malware in a database for comparison. These solutions are more likely to achieve a much lower FPR than typical on-board IDS. However, cloud-based solutions are not perfect either. First, a large rate of real-time data will be transmitted from the vehicle to the cloud whenever the vehicle is on, which brings a huge load to the already resource-limited vehicular systems. Second, the connection between the vehicle and the cloud creates another vulnerability. If this connection is compromised, it would be susceptible to a man-in-the-middle attack, intercepting all communication messages between the vehicle and the cloud while tricking them with forged messages. Third, the latency of a cloud-based solution tend to be longer than an on-board IDS, so it usually takes longer to find

out that the vehicle is being attacked, which could be the difference between life and death. More proper usage of a cloud-based system is to combine it with an on-board IDS, which could scan and validate most of the CAN packets on its own and only send a partial log to the cloud for confirmation when it is not sure about its authenticity. This could greatly reduce the communication load with the cloud-based defense system. The IDS is also responsible for the safety of the communication link. On the other hand, the cloud could provide regular firmware updates to the ECUs and fix any found vulnerabilities, as well as updating the IDS when a new type of threat is found. Therefore, cloud-assisted solutions are only viable when combined with an IDS, and the design of a practical IDS should be prioritized before employing a cloud-based solution.

VI. THEORY

A. IDS Evaluations

In previous works, various approaches were taken to build an IDS. However, each of them has its own flaws, and there has not been a perfect IDS design as of yet. For instance:

- [4] used normal real-world CAN messages combined with manually injected messages to simulate four types of attacks: DOS attack, fuzzy attack, drive gear spoofing attack and RPM gauge spoofing attack. They employed fuzzy logic techniques and obtained a precision of 85%, which is far from 100% and thus unacceptable for practical use.
- [8] constructed a neural network on data collected from a 2012 Sebura Impreza car and evaluated it using normal and suspicious data sequences, but the Receiver Operating Characteristic (ROC) curve shows that the TPR is at most 0.85 when the FPR is 0, which is decent but still has room for improvement.
- [10] used CAN bus data collected from a 2011 Ford Explorer to implement a flow-based anomaly detector that identifies inserted and deleted packets from irregular frequency changes. They also built an OCSVM for comparison and showed results better than expected. However, they only considered timing statistics and ignored all data frames as well as non-periodic packet types, rendering the IDS impractical.
- [16] proposed using recurrent neural networks to detect malicious packets and showed that it performs better than the conventional threshold-based method. However, it may still fail to detect some consecutive malicious packets when the variance of the packet contents is large.

- [18] also built an IDS based on long short-term memory recurrent neural network and obtained satisfactory results on synthesized datasets, but its detectability drops to almost zero when the FPR is kept at zero.

B. Proposed solution

As discussed in the previous section, in order to make an IDS practical, the limitation that the FPR of the IDS has to be zero must be relaxed. To do this, the periodicity and redundancy of the CAN signals could be harnessed. Most ECUs in the CAN bus broadcast their current state variables to the rest of the network with a strict frequency, which gives the system high fault tolerance, noise immunity and resilience to transient attacks. Individual spoofed messages could at best cause a momentary fluctuation in a control signal before it recovers, and are unlikely to cause a catastrophic system crash [7]. Our previous discussion on known attack types also shows that almost all types of attacks would give rise to a significant change in the CAN traffic. The only known exception is diagnosis attack, which is the only type of attack that does not need to send packets at a high rate and vie for the attention of its target recipient ECU. However, since diagnosis packets appear only when the vehicle is under diagnosis, the detection of this type of attack can be trivialized by ignoring all diagnostic packets during the vehicle's normal operation. Given this insight, it is sufficient for an IDS to raise the alarm only when it detects a significant number of suspicious or invalid packets in the CAN bus within a small time interval. If the FPR of one packet is x, then the possibility that n packets are all misclassified will drop to x^n. Therefore, this solution could exponentially reduce the FPR and make the result more reliable.

Most of the previous IDS implementations are signature-based, which means they use signatures of various attack types to train the identifiers. Thus, they are good at identifying signatures of known attack types but are not capable of detecting novel types of attack [22]. This project aims to present an IDS based on OCSVM that does not require any knowledge about the meaning of CAN packets nor assume any attack type, and thus can be easily adapted to any model of car. It is also anomaly-based and is able to detect abnormal behavior caused by both known and novel attacks.

C. OCSVM

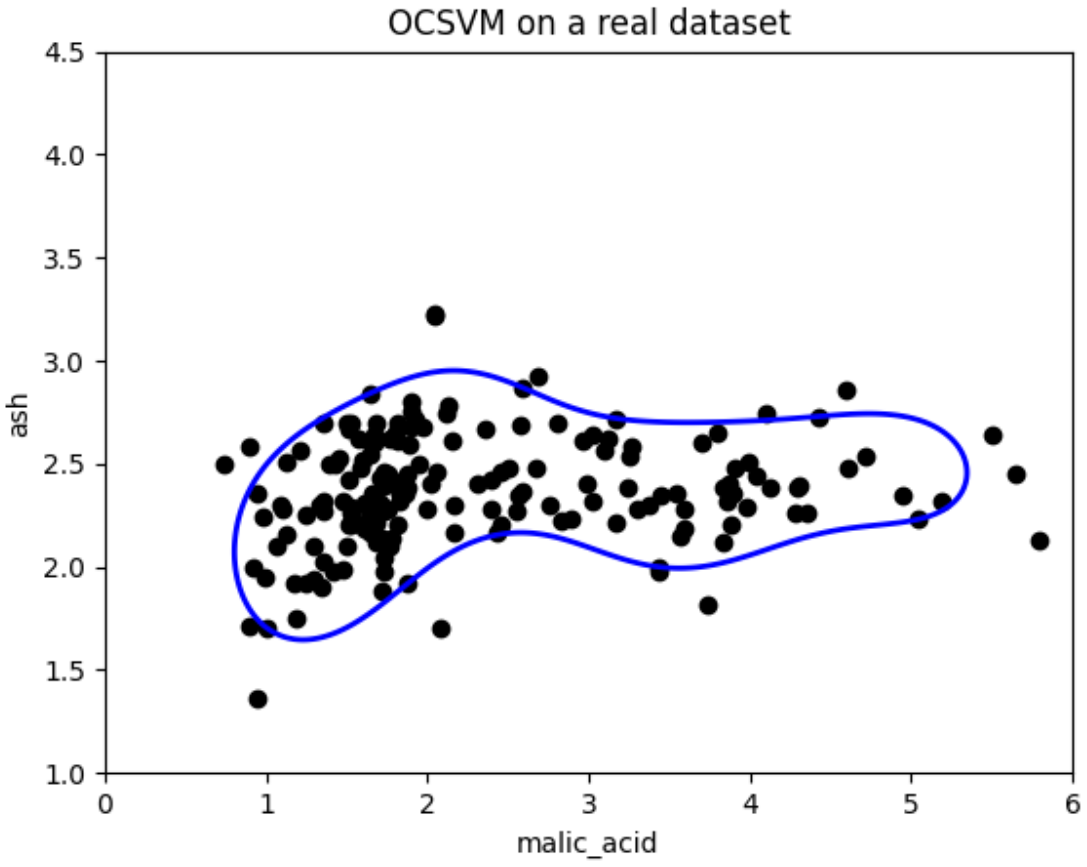

Fig. 4. OCSVM on the wine recognition dataset

For classification problems that involve distinguishing test data amongst different classes based on labeled training data, traditional classification algorithms like Naive Bayes (NB) and Support Vector Machine (SVM) are usually our first choices. However, for problems such as detecting malicious messages in the CAN bus, it is easy to collect normal data, but gathering faulty messages is much harder and more expensive. Even if some data can be collected through simulation or during experiments, there is no way to ensure that the collected data are representative of all known and novel types of attacks. OCSVM is an algorithm that was introduced by Schölkoph et al. to specifically handle this kind of case, where the training data consists mostly or entirely of one of the classes. It will try to build a model that captures the characteristics of the initial observations, and if it encounters a data point that deviates too much from the learned characteristics, then the data point will be labeled as out-of-class. Figure 4 shows an example where OCSVM is applied to the wine recognition dataset and learns a contour of the data distribution [23].

Mathematically, the algorithm first transfers all data points to a hyperplane represented with the equation $\omega^T + b = 0$, where $\omega \in F$ (F is the feature space) and $b \in R$. Then it tries to maximize the distance between the data points and the origin in the feature space. The objective function is

$$\min_{\omega,\xi_i,\rho} \frac{\|\omega\|^2}{2} - \rho + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i$$

subject to:

$$(\omega \cdot \phi(x_i)) \geq \rho - \xi_i \qquad \text{for all i = 1, …, n}$$

$$\xi_i \geq 0 \qquad \text{for all i = 1, …, n}$$

where ξ_i are the slack variables that specify the outliers, x_i are the data points, n is the number of data points, and ν is the parameter that controls both the upper bound of the fraction of outliers in the training data and the lower bound of the number of data points used as Support Vector (the nu parameter in Scikit-learn). Solving this problem using Lagrange multipliers yields the decision function

$$f(x) = sgn(\sum_{i=1}^{n} \alpha_i K(x, x_i) - \rho)$$

where $\alpha_i > 0$ are the Lagrange multipliers that "support" the machine, and $K(x, x_i)$ is the kernel function. Common choices of the kernel function are Radial Basis Function (RBF), linear, poly, and sigmoid. RBF is the most popular choice among them, and this is what we chose for this project. The definition of RBF kernel is

$$K(x, x') = \exp(-\gamma \|x - x'\|^2)$$

where γ is a free parameter (the gamma parameter in Scikit-learn).

VII. EXPERIMENT

A. Dataset

The dataset used in this experiment is downloaded from [24], which includes three sets of CAN bus data collected from an Opel Astra, a Renault Clio and a prototype respectively. The prototype dataset only has about a hundred thousand packets, which is not enough to be used for any purpose. So we only used the first two sets of data, which were captured by connecting a laptop and a CAN-to-USB interface called CANtact to the OBD-II port of the vehicle, followed by driving the car in an urban environment for several minutes. The Opel Astra dataset lasts for 1382.22 seconds and contains 2.7 million packets from 85 unique CAN IDs, while the Renault Clio dataset lasts for 275.09 seconds and contains 0.4 million packets sent by 55 different CAN IDs. The datasets also come with synthesized attack data created by copying and modifying the original data. However, we did not use the attack data and will instead synthesize our own attack data, as this will give us more control over it. The Opel Astra dataset has significantly more packets than the Renault Clio one, so we mainly used this one for training and validation, and used the Renault Clio dataset for the final evaluation only.

B. Data processing

In the raw datasets, every CAN packet has three fields: the timestamp when the message was sent, the sender's CAN ID, and the eight-byte payload of the packet. Each byte of the payload is treated as an individual feature to give them equal significance. Since the timestamp itself does not have any meaning, it was replaced by two new features derived from it: the time passed since the last packet was sent to the CAN bus, and the time passed since the last CAN

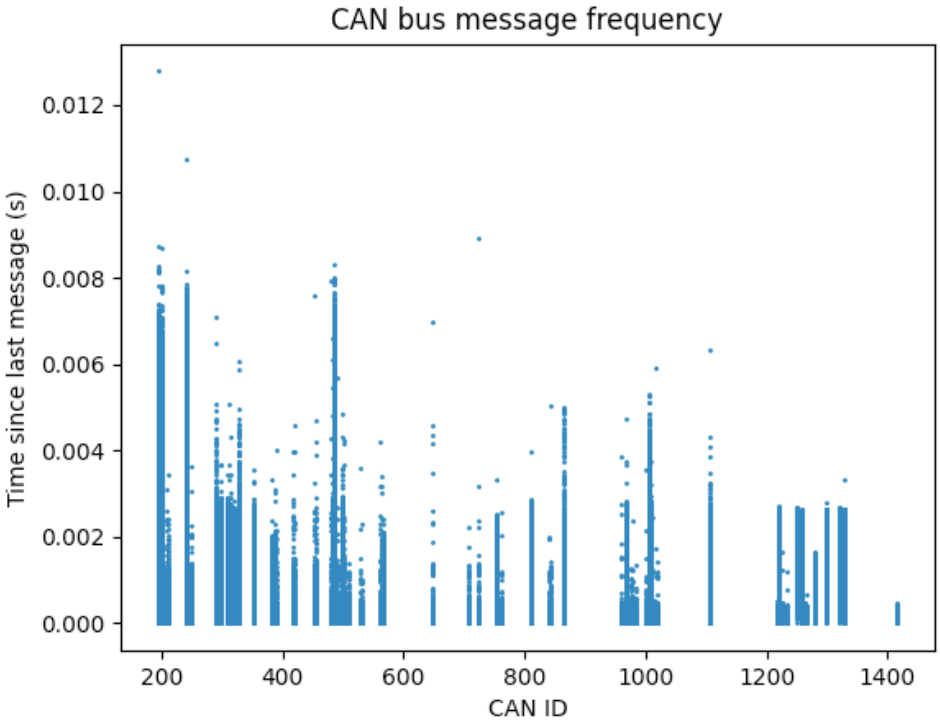

Fig. 5. Message frequency of CAN bus

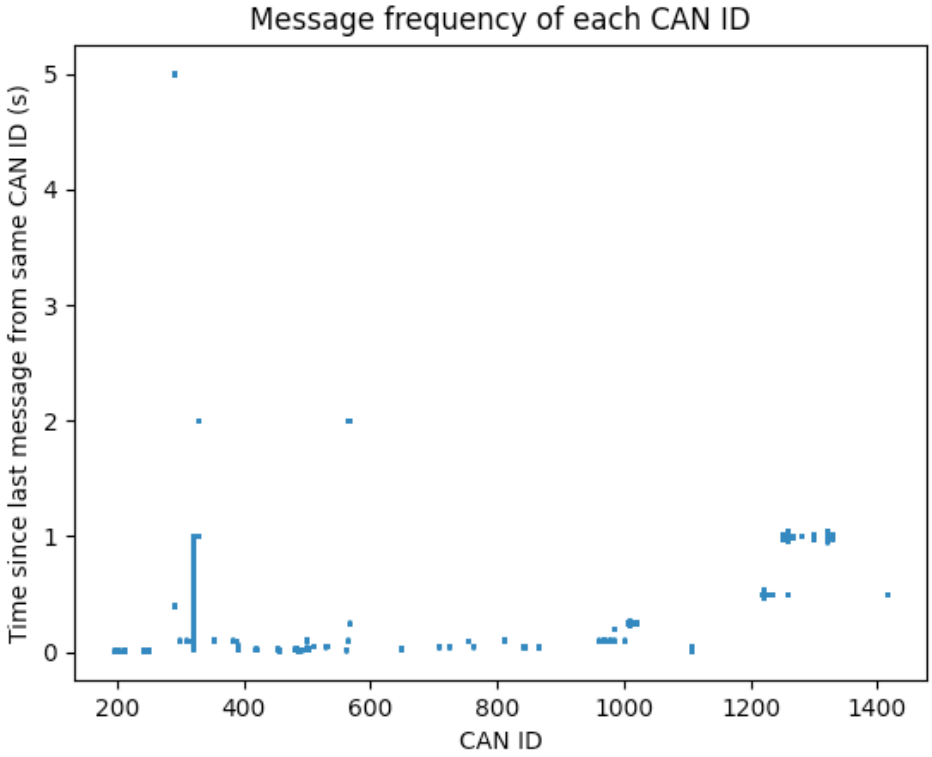

Fig. 6. Message frequency of each CAN ID

message was sent by the same CAN ID. We used both of these two features because both of them have a clear correlation with the CAN ID, as shown in Figures 5 and 6, and most attack types will significantly change either or both of these two features. The first packet sent by each CAN ID does not have these two values, so these packets were discarded. Also in the dataset some CAN packets have their CAN ID greater than 0x700, indicating that they are diagnosis messages. However, as discussed earlier, diagnosis messages are not supposed to appear during a vehicle's normal operation, so we assumed that these packets were produced by the data collecting devices and removed them from the dataset. Since the Opel Astra dataset is sufficiently large, we did not use cross-validation and simply used the first 1.2 million packets for training and the following 0.6 million packets for validation. The last 0.9 million packets were reserved for final testing. After the features were extracted from the raw dataset, we used a standard scaler to center the data points to Gaussian distribution before using them for training and testing.

C. Evaluation criteria

Before training the model, we need to decide on the evaluation criteria. An ideal IDS should keep quiet during the vehicle's normal operation, but when the vehicle is under attack, the IDS should be able to detect and respond to it as soon as possible. We achieve this by periodically counting abnormal CAN packets detected in the CAN bus and compare it to a set threshold. If the number of detected abnormal packets in a time period t exceeds the threshold, then it will be an indication that the vehicle is under attack. The psuedocode of the IDS's behavior is shown as the following, assuming all variables are properly defined and initialized before the loop:

```
while packetBuffer:     # assume new packets are appended to packetBuffer
        packet = packetBuffer.pop(0)
        t = packet[timestamp]
        if t >= nextCheckpoint:
                if count >= threshold: # raise the alarm
                count = 0
                nextCheckpoint += t
        packet.remove(timestamp)
        packet[timeSinceLastPacket] = t - prevTime
        packet[timeSinceLastPacketFromECU] = t - prevTimeFromECU[packet[ECUID]]
        prevTime = t
        prevTimeFromECU[packet[ECUID]] = t
        if clf.predict(packet) == -1: count += 1
```

Because our data points are generated from not only the current packet but also the previous packet in the CAN bus and the previous packet from the same CAN ID, abnormal packets will affect the legitimacy of their neighboring packets as well. Therefore conventional evaluation criteria like the confusion matrix would make little sense in this problem. Instead, we want to maximize the difference in the positive count between normal data and attack data, so that a threshold in between could easily separate the two groups of data. Specifically, we define *maxNormal* as the maximum positive count during any time period *t* in the normal data, and *minAttack* as the minimum positive count during any time period *t* in the attack data. To account for the uncertainty, we are going to use $S = minAttack - maxNormal$ as our scoring function, so a

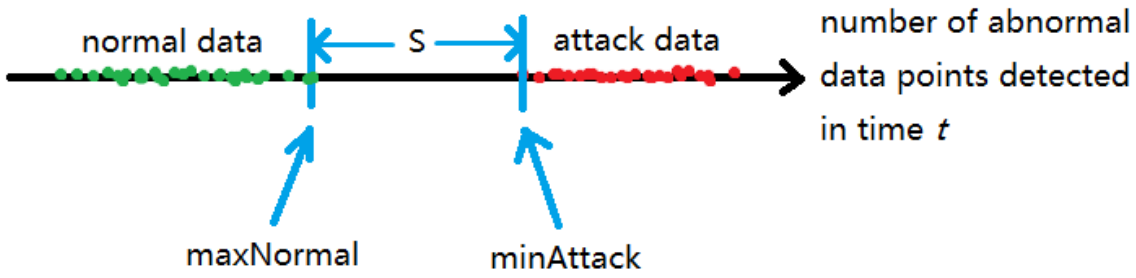

Fig. 7. The scoring function

positive S means that one type of data will never be classified as another. However, generally we want S to be as large as possible, so that we can be more confident in the decision. Figure 7 illustrates how our scoring function is derived.

As discussed earlier, the three major categories of attacks are insertion, modification and deletion of the original CAN messages, so these were the ways we generated our attack data. The modification and deletion cases has not been considered by many IDS evaluations due to their relatively rare occurrences, but we included them as extra challenges. As for the rate of change in the dataset, works like [4], [10] and [18] synthesized their attack data by injecting 0.5 to 10 times of the number of packets in the normal traffic. Hence we decided to create our attack data by injecting, modifying or deleting twenty percent of the number of packets in our original data, since a higher rate of change would be easier to detect, and a lower rate of change would not be likely to cause significant damage. For insertion, we created various types of new CAN packets and inserted them into the original data in a frequency of four hundred packets per second, or one packet every 2.5 milliseconds, which is about twenty percent of the frequency of the normal CAN traffic. For simplicity, we will use “random ID” to represent a randomly generated CAN ID between 0 and 0x700, and use “valid ID” to represent a valid CAN ID that is randomly chosen

from the pool of CAN IDs appeared in the original data. Likewise, we will use "random payload" to represent a randomly generated 8-byte payload, and use "valid payload" to represent a valid payload that is randomly chosen from the pool of payloads that appeared in the original data. Then the five types of CAN packets inserted are "random ID, random payload", "random ID, valid payload", "valid ID, random payload", "valid ID, valid payload", and "replay" which simulates the replay attack. For modification, we will randomly modify twenty percent of the original data points with "random ID", "valid ID", "random payload", and "valid payload". And for deletion, we will simulate two scenarios: randomly deleting twenty percent of the original data, and randomly choosing one CAN ID from the pool of valid CAN IDs and deleting all packets sent from it. To prove our model's ability to detect novel types of attacks, we performed our validations on the "replay" and "random ID" scenarios only and used the outcome to find the best gamma and nu values. The other scenarios were reserved for our final evaluations on the final model.

D. Model training

Once the evaluation criteria are decided, building the model is quite straightforward with Scikit-learn's built-in OneClassSVM class. We used default values for most of the hyper-parameters except for gamma and nu, for which we used grid search to find the best values. Gamma could be any positive number, while nu should be very close to zero in this problem as we only use normal data to train the classifier. The optimal values we decided on were gamma = 0.4 and nu = 0.0025. The data in figures 8 to 10 were captured with gamma = 0.4, nu = 0.0025, training size = 1.2m and t = 1s except for the x-variables in the graphs. Figures 8 and 9 show the

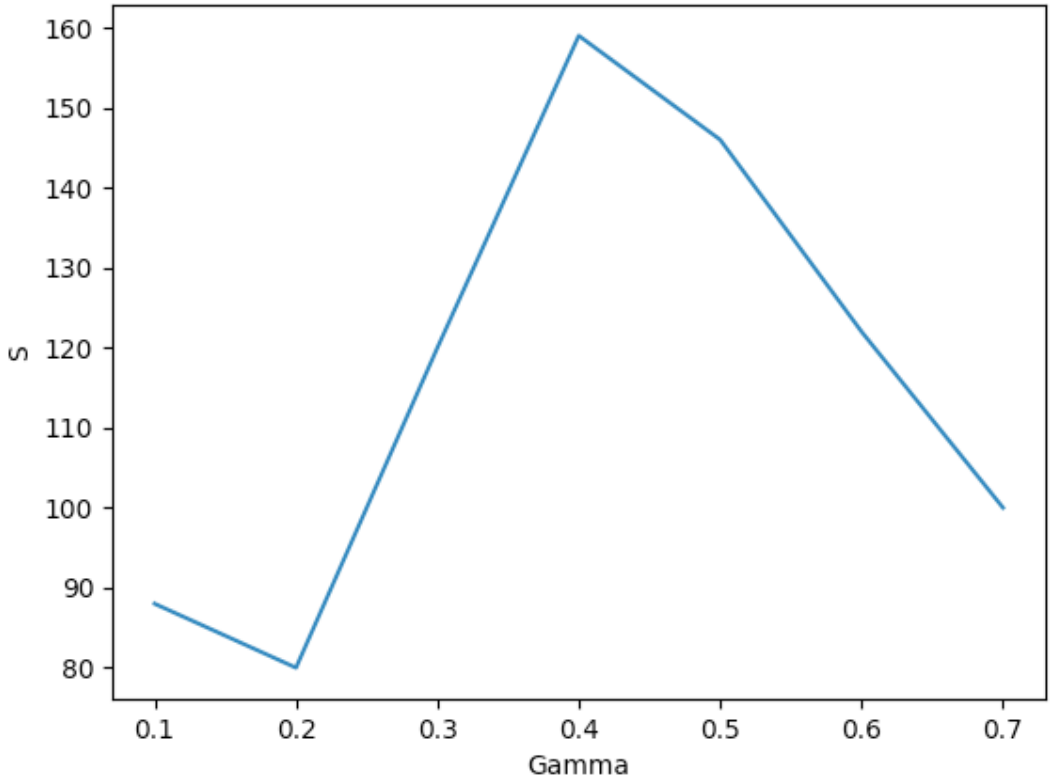

Fig. 8. Relationship between gamma and *S*

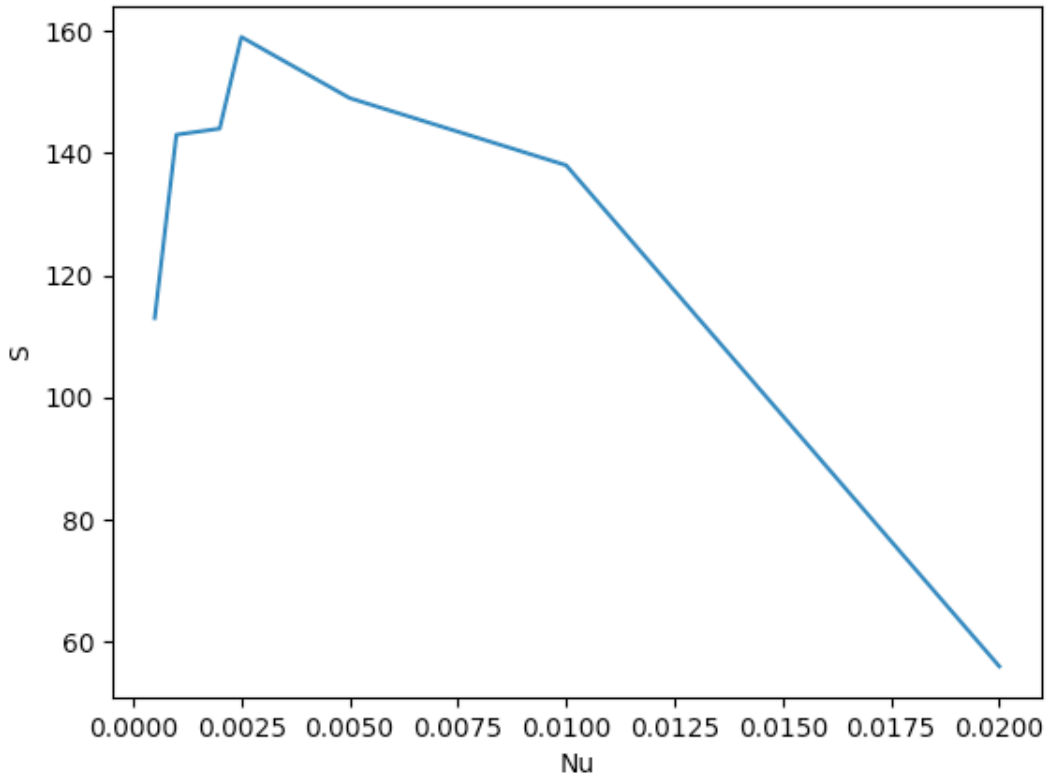

Fig. 9. Relationship between nu and *S*

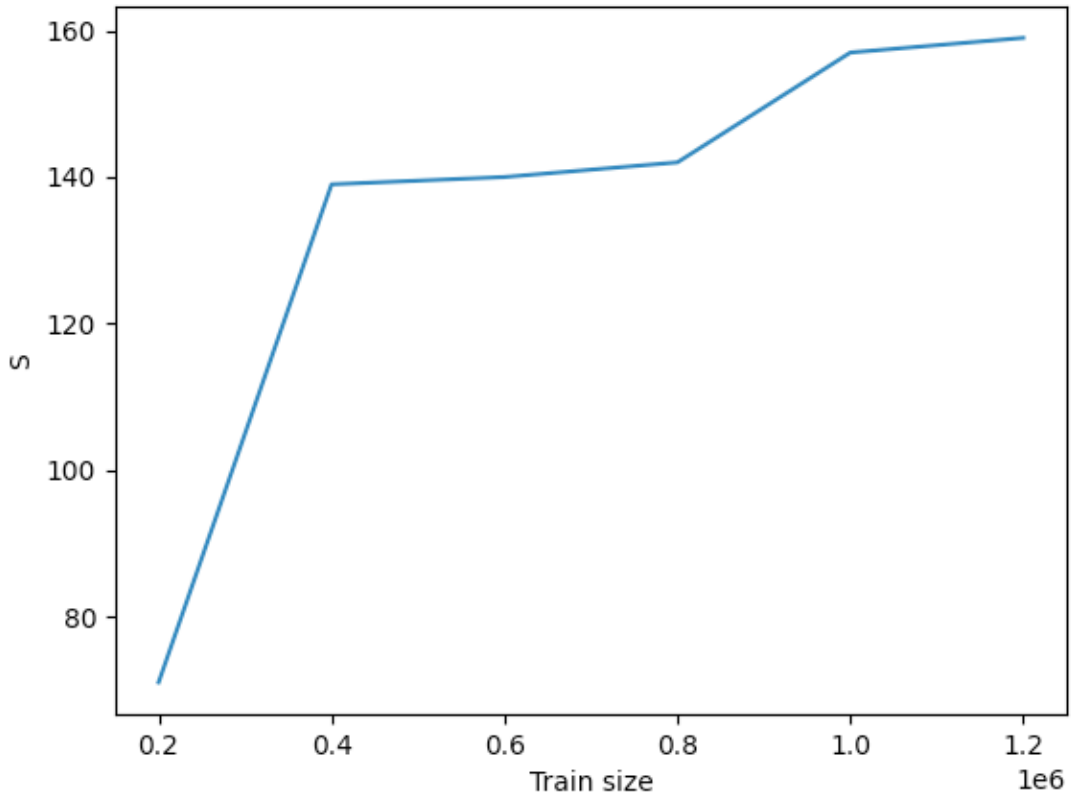

Fig. 10. Relationship between training size and *S*

process of finding the optimal values for gamma and nu respectively, while figure 10 shows the relationship between the training size and *S*, which proves the dataset size is quite sufficient for our purpose.

E. Final evaluations

The final evaluations on the Opel Astra dataset and the Renault Clio dataset are shown in Table V and VI respectively. As a reminder, the min, max and average columns mean the minimum, maximum and average number of positive data points detected in time *t* respectively, and our scoring function is $S = minAttack - maxNormal$. To reduce the influence of all the randomness on the results, all values were sampled five times and then averaged. For the Opel

Astra dataset, the first 1.8 million packets were used for training and the rest 0.9 million packets were used for testing. The result scores for insertion and modification are all positive and sufficiently large, even though we only used the replay and random ID scenarios for training. This proves OCSVM's ability to detect novel attack types. Note that even though more positive data points on average were detected with random ID and/or payload than with valid ones, the minimum number of positive data points detected was lower due to its higher randomness. On the other hand, S is only -17 for random deletion and -56 for deletion from one CAN ID. This implies that deletions of CAN packets are harder for our IDS to detect. For the Renault Clio dataset, the first half was used for training and the second half was used for testing. Other parameters remained the same, i.e. gamma = 0.4, nu = 0.0025 and t = 1 second. The results were very similar to the Opel Astra dataset, which proves that our solution works for different models of cars.

TABLE V
EVALUATION ON THE OPEL ASTRA DATASET

Type	Min	Max	Average	S
Original	1	58	17	-
Insertion - random ID, random payload	175	447	411	117
Insertion - random ID, valid payload	168	447	408	110
Insertion - valid ID, random payload	314	416	360	256
Insertion - valid ID, valid payload	274	366	316	216
Insertion - replay	192	300	233	134
Modification - random ID	213	492	427	155
Modification - random payload	166	243	201	108
Modification - valid ID	224	323	275	166

Modification - valid payload	114	194	147	56
Deletion - randomly	41	122	69	-17
Deletion - from one ID	2	65	21	-56

TABLE VI
EVALUATION ON THE RENAULT CLIO DATASET

Type	Min	Max	Average	S
Original	4	79	28	-
Insertion - random ID, random payload	247	454	406	168
Insertion - random ID, valid payload	241	445	400	162
Insertion - valid ID, random payload	469	533	491	390
Insertion - valid ID, valid payload	356	425	382	277
Insertion - replay	428	553	489	349
Modification - random ID	148	434	348	69
Modification - random payload	246	342	287	167
Modification - valid ID	247	352	294	168
Modification - valid payload	110	195	145	31
Deletion - randomly	78	144	104	-1
Deletion - from one ID	4	79	28	-75

To get a feeling of how various parameters affect the result, more experiments were performed where one parameter varies and the other ones remains fixed. First, we measured the relationship between t and S in the replay attack, as shown in figure 11. We can see that if we only want a positive S, then t could be set to as low as 0.2 seconds. However, since S increases significantly as t increases, for the best performance t should be set to the maximum time the vehicle could spend on determining if it is under attack before taking countermeasures, which is something the vehicle designers need to measure. Next, we plotted the relationship between the

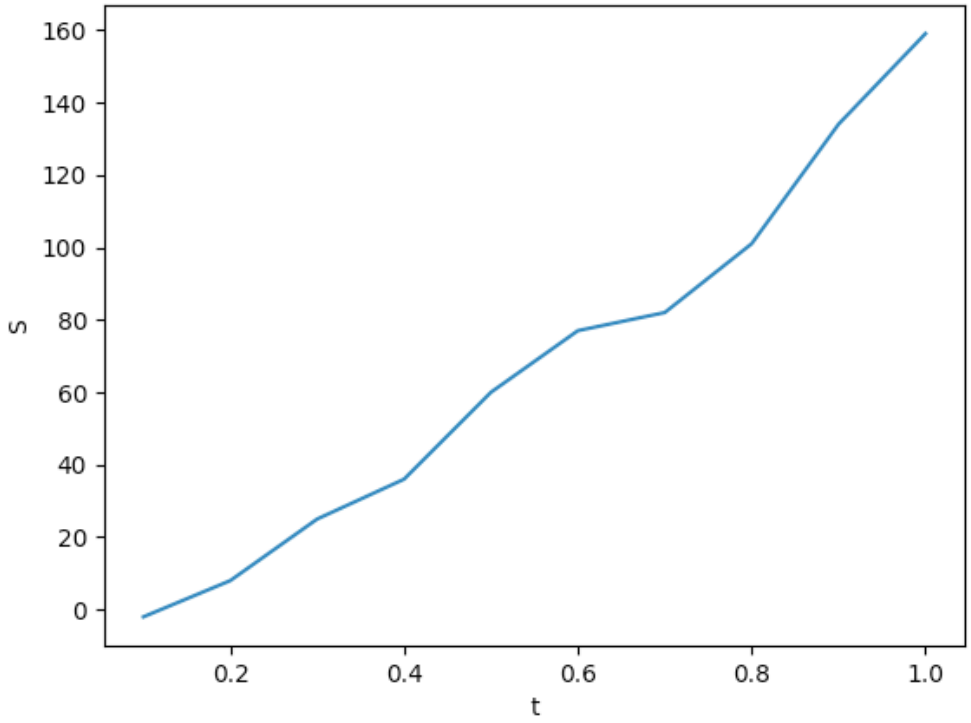

Fig. 11. Relationship between *t* and *S*

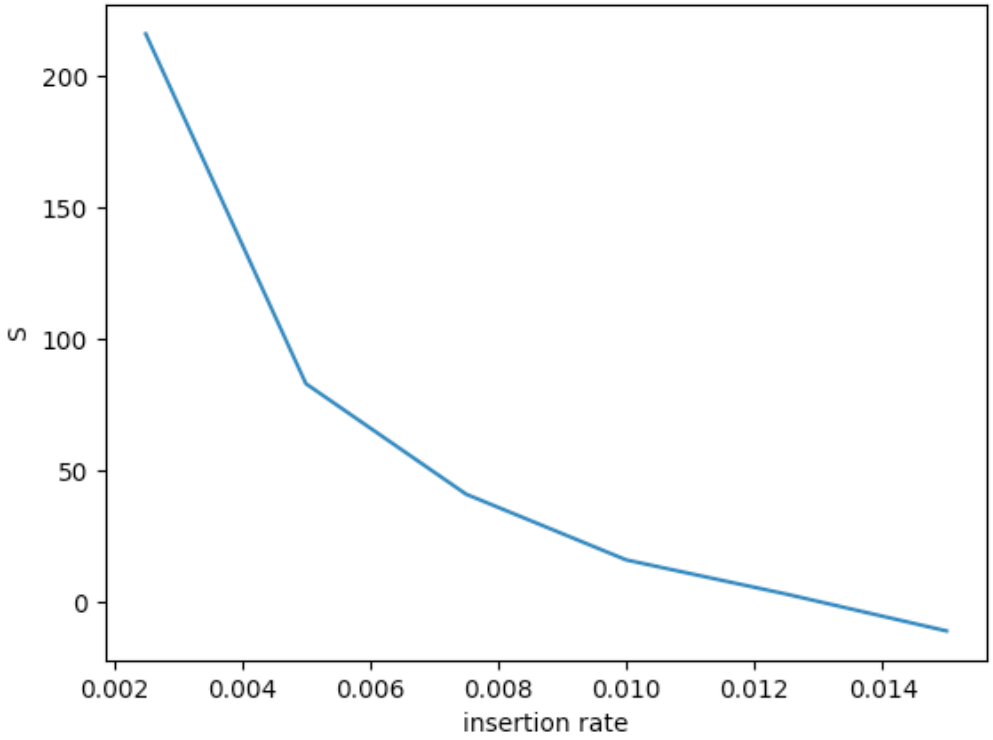

Fig. 12. Relationship between insertion frequency and *S*

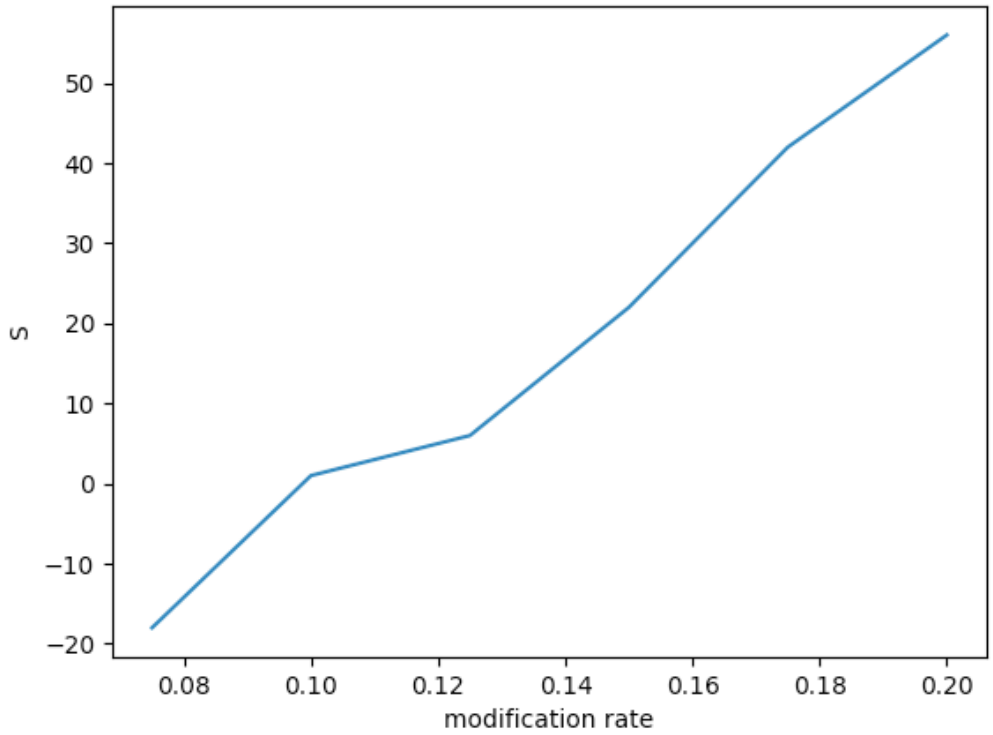

Fig. 13. Relationship between modification rate and *S*

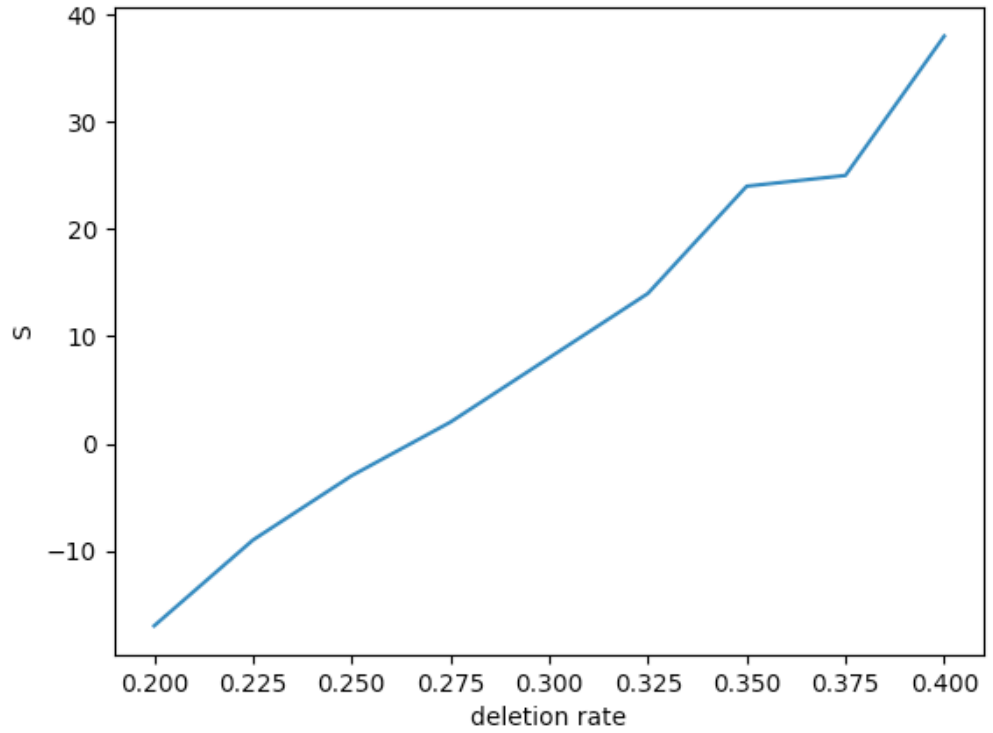

Fig. 14. Relationship between deletion rate and *S*

rate of change in the data and *S*, as shown in figures 12 to 14. We can see from these figures that our IDS is capable of perfectly detecting an insertion frequency of one packet every 12.5 milliseconds or less, a modification rate of at least 10%, or a deletion rate of above 27.5%. This confirmed that our solution has satisfactory detection ability on injection-based or modification-based attacks, but its ability to detect deletion-based attacks still needs improvement. One idea is to combine the IDS with a monitoring device which periodically checks if all the CAN IDs are still alive. We also looked into the scenario of deletion from one CAN ID, and found that our IDS cannot effectively detect the silence of any CAN ID, probably because the silence of one ECU does not cause a significant rate of change in the whole CAN traffic. Finally, we tested the case where the packets are injected sporadically rather than periodically, and the result showed that it does not really affect the result as long as the number of packets injected in time *t* stays the same.

TABLE VII
COMPARISON TO OTHER IDS IMPLEMENTATIONS

	Our solution	[4]	[10]
Insertion frequency	> 12.5 ms	0.5 ms	< 0.5 ms
Insertion accuracy	100%	> 82.3%	> 90.7%
Modification rate	> 10%	Not tested	Not tested
Modification accuracy	100%	Not tested	Not tested
Deletion rate	> 27.5%	Not tested	50%
Deletion accuracy	100%	Not tested	> 73.1%
Delay	1 second	Instantly	0.2 - 0.5 seconds

Lastly, we compared our results to the performances of [4] and [10] representing other IDS implementations, as shown in Table VII. As we can see, they mainly focused on the

insertions case, and in all the tested cases they had a worse detection accuracy on a more heavily-modified attack dataset than our solution. In conclusion, our solution was superior in both the sensitivity and the accuracy. The only downside is that our solution requires a one-second delay in detection, but it could be adjusted flexibly at the cost of the sensitivity, as shown in figure 10. Furthermore, our solution is the only one that can perfectly identify normal data and attack data, which is an important prerequisite for the IDS to be potentially viable.

VIII. CONCLUSION

This project has first provided the background about vehicle cybersecurity, then proposed a detection-based solution to defend vehicles against cyberthreats. It has introduced concepts about vehicle subnetworks necessary to understand the topic of vehicle cybersecurity, including ECU, CAN bus and OBD. It has also discussed previous car hacking experiments, known attack types and proposed defense methods. Finally, a new design of IDS based on OCSVM has been implemented and evaluated. The main contribution of this project is proving that most sophisticated cyberattacks against CAN bus would result in a high frequency of packet injection, modification or deletion, thus eliminating the need to detect every single abnormal packet, which makes the design of a practical IDS possible. As future work, more experiments on the insertion and modification cases could be run to get a better sense of the solution's reliability, and a solution to improve its ability to detect deletion-based attacks needs to be found. Moreover, the proposed method ultimately needs to be tested on a real vehicle and have its feasibility confirmed. This will ultimately help vehicle manufacturers design an effective and efficient security solution that can prevent vehicles from most cyberattacks at a reasonable cost.

REFERENCES

[1] H. Kong, T. Kim and M. Hong, "A Security Risk Assessment Framework for Smart Car," *2016 10th Int. Conf. Innov. Mob. Intern. Serv. Ub. Comput. (IMIS)*, Fukuoka, 2016, pp. 102-108.

[2] C. Miller, "Lessons learned from hacking a car," in *IEEE Design & Test*, vol. 36, no. 6, pp. 7-9, Dec. 2019.

[3] Y. Zhang, B. Ge, X. Li, B. Shi and B. Li, "Controlling a Car Through OBD Injection," *2016 IEEE 3rd Int. Conf. Cyber Secur. Clo. Comput. (CSCloud)*, Beijing, 2016, pp. 26-29.

[4] F. Martinelli, F. Mercaldo, V. Nardone and A. Santone. “Car hacking identification through fuzzy logic algorithms,” *2017 IEEE Int. Conf. Fuz. Syst. (FUZZ-IEEE)*, Naples, 2017, pp. 1-7

[5] S. Jafarnejad, L. Codeca, W. Bronzi, R. Frank and T. Engel, "A Car Hacking Experiment: When Connectivity Meets Vulnerability," *2015 IEEE Globecom Works. (GC Wkshps)*, San Diego, CA, 2015, pp. 1-6.

[6] W. Yan, "A two-year survey on security challenges in automotive threat landscape," *2015 Int. Conf. Conn. Veh. Expo (ICCVE)*, Shenzhen, 2015, pp. 185-189.

[7] P. Koopman and C. Szilagyi, “Integrity in embedded control networks,”, in *IEEE Secur. & Priv.,* vol. 11, no. 3, pp. 61-63, May-June 2013.

[8] S. Boumiza and R. Braham, "An Anomaly Detector for CAN Bus Networks in Autonomous Cars based on Neural Networks," *2019 Int. Conf. on Wireless and Mobile*

Comput., Netw. and Commun. (WiMob), Barcelona, Spain, 2019, pp. 1-6, doi: 10.1109/WiMOB.2019.8923315.

[9] A. Kurbanov, S. Grebennikov, S. Gafurov and A. Klimchik, "Vulnerabilities in the vehicle's electronic network equipped with ADAS system," *2019 3rd School on Dynamics of Complex Networks and their Application in Intellectual Robotics (DCNAIR)*, Innopolis, Russia, 2019, pp. 100-102, doi: 10.1109/DCNAIR.2019.8875529.

[10] A. Taylor, N. Japkowicz and S. Leblanc, "Frequency-based anomaly detection for the automotive CAN bus," *2015 World Congr. on Ind. Control Syst. Security (WCICSS)*, London, 2015, pp. 45-49, doi: 10.1109/WCICSS.2015.7420322.

[11] B. Palaniswamy, S. Camtepe, E. Foo and J. Pieprzyk, "An Efficient Authentication Scheme for Intra-Vehicular Controller Area Network," in *IEEE Trans. on Inf. Forensics and Security*, vol. 15, pp. 3107-3122, 2020, doi: 10.1109/TIFS.2020.2983285.

[12] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaaniche and Y. Laarouchi, "Survey on security threats and protection mechanisms in embedded automotive networks," *2013 43rd Annu. IEEE/IFIP Conf. on Dependable Syst. and Netw. Workshop (DSN-W)*, Budapest, 2013, pp. 1-12, doi: 10.1109/DSNW.2013.6615528.

[13] T. Zhang, H. Antunes and S. Aggarwal, "Defending Connected Vehicles Against Malware: Challenges and a Solution Framework," in *IEEE Internet of Things J.,* vol. 1, no. 1, pp. 10-21, Feb. 2014, doi: 10.1109/JIOT.2014.2302386.

[14] A. Lopez, A. V. Malawade, M. A. Al Faruque, S. Boddupalli and S. Ray, "Security of Emergent Automotive Systems: A Tutorial Introduction and Perspectives on Practice," in

IEEE Des. & Test, vol. 36, no. 6, pp. 10-38, Dec. 2019, doi: 10.1109/MDAT.2019.2944086.

[15] J. Khan, "Vehicle network security testing," *2017 Third Int. Conf. on Sens., Signal Process. and Security (ICSSS)*, Chennai, 2017, pp. 119-123, doi: 10.1109/SSPS.2017.8071577.

[16] H. Suda, M. Natsui and T. Hanyu, "Systematic Intrusion Detection Technique for an In-vehicle Network Based on Time-Series Feature Extraction," *2018 IEEE 48th Int. Symp. on Multiple-Valued Log. (ISMVL)*, Linz, 2018, pp. 56-61, doi: 10.1109/ISMVL.2018.00018.

[17] R. Buttigieg, M. Farrugia and C. Meli, "Security issues in controller area networks in automobiles," *2017 18th Int. Conf. on Sci. and Techn. of Autom. Control and Comput. Eng. (STA)*, Monastir, 2017, pp. 93-98, doi: 10.1109/STA.2017.8314877.

[18] A. Taylor, S. Leblanc and N. Japkowicz, "Anomaly Detection in Automobile Control Network Data with Long Short-Term Memory Networks," *2016 IEEE Int. Conf. on Data Sci. and Adv. Analytics (DSAA)*, Montreal, QC, 2016, pp. 130-139, doi: 10.1109/DSAA.2016.20.

[19] M. Hashem Eiza and Q. Ni, "Driving with Sharks: Rethinking Connected Vehicles with Vehicle Cybersecurity," in *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 45-51, June 2017, doi: 10.1109/MVT.2017.2669348.

[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," in *IEEE Trans. on Intell. Transp. Syst.*, vol. 16, no. 2, pp. 993-1006, April 2015, doi: 10.1109/TITS.2014.2351612.

[21] G. Burzio, G. F. Cordella, M. Colajanni, M. Marchetti and D. Stabili, "Cybersecurity of Connected Autonomous Vehicles : A ranking based approach," *2018 Int. Conf. of Elect. and Electron. Technol. for Automot.*, Milan, 2018, pp. 1-6, doi: 10.23919/EETA.2018.8493180.

[22] K. M. Ali Alheeti and K. McDonald-Maier, "An intelligent security system for autonomous cars based on infrared sensors," *2017 23rd Int. Conf. on Automat. and Comput. (ICAC)*, Huddersfield, 2017, pp. 1-5, doi: 10.23919/IConAC.2017.8082084.

[23] Scikit-learn developers, “Outlier detection on a real dataset,” 2007 - 2020, https://scikit-learn.org/stable/auto_examples/applications/plot_outlier_detection_wine.html#sphx-glr-auto-examples-applications-plot-outlier-detection-wine-py.

[24] [Dataset] Dupont, Guillaume; Lekidis, Alexios; den Hartog, J. (Jerry); Etalle, S. (Sandro) (2019): Automotive Controller Area Network (CAN) Bus Intrusion Dataset v2. 4TU.ResearchData. Dataset. https://doi.org/10.4121/uuid:b74b4928-c377-4585-9432-2004dfa20a5d.