National College of Ireland

Enhancing CAN Bus Security Using Message Authentication

MSc Research Project
MSc Cybersecurity

Corey Gallagher
Student ID: X21179506

School of Computing
National College of Ireland

Supervisor: Raza Ul Mustafa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……..……Corey Gallagher……

Student ID: ……x21179506……..……

Programme: ……MSc Cybersecurity…… **Year:** ……2…….

Module: ……..……

Lecturer: ……Raza Ul Mustafa……..……

Submission Due Date: ……31/01/2024……..……

Project Title: ……Enhancing CAN Bus Security Using Message Authentication

Word Count: ……8133…… **Page Count:** ……16……..……..……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……

Date: ……31/01/2024……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Enhancing CAN Bus Security Using Message Authentication

Corey Gallagher
X21179506

Abstract

The Controller Area Network (CAN) protocol, introduced by Robert BOSCH GmbH in 1983, is a critical component of automotive communication and other domains. This thesis aims to improve the security of the CAN bus by implementing message authentication mechanisms. With modern vehicles containing around 100 Electronic Control Units (ECUs) responsible for critical functions, a reliable communication network is essential. The CAN bus includes advantages such as high immunity to electrical interference and easy wiring, which makes it perfect for the automotive industry.

This research addresses the security challenges within the current CAN bus architecture, with a specific focus on message authentication, and explores how its implementation can improve the overall security. The study includes an extensive literature review of the CAN bus, its security challenges, and existing research on message authentication. Additionally, the methodology section details the setup of the research environment, including the tools and techniques used to investigate the research question.

The findings demonstrate that while implementing Hash-Based Message Authentication (HMAC) can significantly improve the security of the CAN bus communication in vehicles it also has a negative impact on the latency of the network. The study also highlights the need for further research into the security of the CAN bus, especially with the increasing number of ECUs and the growing communication of modern vehicles with back-end servers and other cars.

1. Introduction

In 1983 Robert BOSCH GmbH introduced the Controller Area Network (CAN-BUS) protocol, since then the protocol has been adopted in a wide range of domains such as domestic appliances, medical devices, entertainment, and the area which will be the focus of this thesis automotive communication [1]. The CAN is now one of the main in vehicle communication networks that is responsible for the communication between the ECUs in the vehicle [2]. Modern vehicles contain in the region of around 100 ECUs which are responsible for controlling vehicle functions such as engine control, anti-lock braking systems and airbag deployment along with a wide range of other functions. [2]. It is vital that these systems have a reliable communication network which is why the CAN protocol is used. The CAN bus has a number of well-recognised advantages which make it highly suitable for the automotive industry these include easy wiring, easily repair errors, ability to self-diagnose and high immunity to electrical interference [2].

Despite these advantages of the CAN, the increasing number of ECUs along with the growing communication of modern vehicles to back-end servers and other cars leaves the CAN vulnerable to cyber-attacks. With the priority of the CAN bus being set on ensuring reliable communication and the lack of focus being put on the security of the network it leaves the network vulnerable to cyberattacks [3]. Lack of encryption is also something that is concerning on the CAN bus. The CAN bus is designed to be a broadcast type network which means the nodes (in this case the ECUs) capture the messages relevant to them as they travel through the network. This broadcast data is not encrypted which means there is a vulnerability here that could potentially be exploited by malicious actors [3].

Now that modern vehicles are capable of collecting and transmitting the drivers personal information it could lead to an invasion of privacy and potential loss of sensitive data to cyber criminals [3]

An in depth study on the security analysis of a modern automobile was conducted by Checkoway and Koscher et al. [4] where they investigated a wide arrange of attack surfaces and the challenges faced in vehicles found a number of potential security concerns. One of these concerns were the attackers obtaining access to the vehicle and being able to gain full control over the vehicle's brakes. Not only this the paper discusses the group being able to systematically control a wide variety of components which included, heating, cooling, lights, radio and locks [4]. With these attacks utilising the vulnerabilities found in the CAN bus it proves that the security of the CAN bus which includes message authentication needs to be taken more seriously going forward especially as vehicles are becoming more interconnected.

The research question of this paper is **How can the implementation of message authentication mechanisms enhance the security of the CAN bus communication in vehicles?** To conduct this research effectively we will investigate what problems are found in the current CAN bus architecture with the primary focus on message authentication and how implementing it in the architecture will improve the overall CAN bus security.

The structure of this research paper will be as follows.

- **Literature Review**: Here we will conduct an in-depth literature review focusing on state-of-the-art research that has been conducted in the field. The literature review will start by examining the foundations of the CAN bus, highlighting the security challenges faced and examining some of the other research conducted on message authentication on the CAN bus.
- **Methodology**: This section will detail the set-up of the environment that was used to conduct the research on the message authentication. It will include details of the tools employed to investigate the research question and highlight the main python scripts that were used to carry out the testing of the research. Details of how the testing was conducted along with how the HMAC message authentication works is also included in this section.
- **Results**: Here the findings and results will be interpreted, and meaningful conclusions will be drawn up.
- **Conclusion:** Will cover emphasise the significance of the work carried out the research question will be reviewed and an answer to the question will be provided.

1. Literature Review

The Controller Area Network is a hierarchically organised distributed communication system for serial data transfer in real time applications. The CAN protocol is built on the Open System Interconnection model (OSI) of ISO (International Organisation for Standardisation) containing mainly the physical layer and the data link layer of the model [5]. Systems which require real time communication and are operating in harsh environments rely on the CAN system as its main principals are high availability, reliability, and robustness. These factors along with the high quality error detection coupled with the data transfer rate of 1 Mbps for the traditional CAN network make the system comfortable for applications that require real time communication [5]. In 2012 Bosch released the CAN FD (Flexible data-rate) which boosted the transfer speeds of the CAN up to 5 Mbps and is capable of a 64-byte payload compared to the 8 bytes that is the capacity in the traditional CAN. [3] The CAN bus was originally developed for automotive purposes but now has many other applications in areas such as trains, medical equipment, building automation and household appliances. [5].

2.1. Basic Principals of CAN Communication

CAN by design reduces the amount of cabling that is required in the vehicle as it is a single two-wire bus architecture, which is demonstrated in **figure 1**. As the networks architecture is distributed it allows for easy maintenance and reduces the overall system cost. Differential wiring mode is also used in the CAN which are represented in the diagram below by CAN H or CAN L, this reduces the amount of noise or electrical interference that the network may face [3]. Logically the signals on the CAN bus have two states which are controlled by the voltage levels. The dominant logic is '0' and the recessive logic is '1'. This means that as long as the nodes or ECUs in this case release the logic '0' to the bus then the bus signal will always remain '0' which is the dominant logic [3]

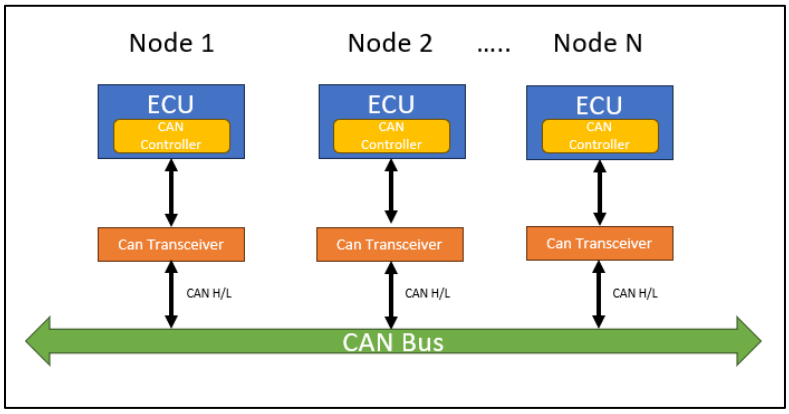

Figure 1. A Two-Wire Controller Area Network (CAN)

The CAN protocol provides its communication with the use of frames. These frames have the same configurations which contains the following Message Identifier Field, Data Field, Cyclic Redundancy Checksum (CRC) and some other control bits. An example of the typical make-up of the traditional CAN frame can be seen in **figure 2.** below. Each node on the network is listening to all frames which are being sent across the network only the frames which are relevant to the specific node will be processed by that node. This decision is made with the use of the Message Identifier Field which is also the bits of the frame which are used for the arbitration [3]

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. Traditional Structure of CAN Frame

For arbitration on the bus the Collision Sense Multiple Access/Collision Detection (CSMA/CD) medium access method is used along with Non-Destructive Arbitration (NDA) [5]. This works in the following way. When **Node 1** is attempting to send a message across the CAN network the first check is done to confirm that the bus is 'idle', this is in relation to the 'Carrier Sense' part of the method. If this check confirms that the bus is 'idle', and no other node wishes to send a message at that time then **Node 1** becomes the master node and transmits its message. All other nodes that are located on the network then switch to receiver mode once the Start of Frame bit (SOF) is sent. [5]. Each node then acknowledges the correct reception of the message the message identifier of the CAN Frame is then checked and if the message is required by the node then it is stored otherwise the message is discarded [5]. 'Multiple Access' however is different as it corresponds to the transmission of a message from two or more Nodes at the same time and this type of collision is avoided by bitwise arbitration. The advanced serial communications protocol carrier sense multiple access/collision detection with non-destructive arbitration is used to handle bus access. Before sending the bits that make up its message identification (MSB), every node checks the bus level. When a node transmits a "recessive" identification bit but receives a "dominant" one in return, it forfeits bus arbitration and enters receive

mode. This scenario arises when a competing node sends a message with a greater priority and its message identifier has a lower binary value, indicating the 'dominant' state or logic 0. [5]

By doing this, the bus node that has the message with the highest priority wins the bus arbitration without wasting time on message repetition. Once the bus returns to the idle state, all other nodes attempt to repeat their transmission intention automatically. Transferring messages with the same identity from separate nodes is not allowed as doing so could cause the bus arbitration to collapse, resulting in errors and collisions. [5]

2.2. CAN Bus Vulnerabilities

The attacks in vehicles are normally grouped into four main categories which are Sensor Initiated, Infotainment Initiated, Telematics Initiated and Direct Interface Initiated. Within these main categories there are two major attack vectors being Wireless Access and Physical Access. Attacks utilise these vectors to get access to the internal network of the vehicle. [6] In a report by Rathore *et al.* six types of attacks on CAN bus systems were highlighted these were Bus-off Attacks, Denial of Service (DoS), masquerading, injection, eavesdropping and replay attacks [6]. Masquerading attacks occur when the attackers gain knowledge of a CAN frame due to the fact that they are not encrypted and normally do not support message authentication and can then gain entry to the network [6]. The broadcasted CAN messages in the vehicle network may be eavesdropped on by attackers and as a result the attackers can infiltrate the CAN network, this type of attack is known as an eavesdropping attack [6]. Injection attacks happen when the attackers can gain access to the OBD-II ports in the vehicle and send false signals on the bus system. By doing this the attackers may be able to establish a connection with then in vehicle network and try to compromise the ECUs [6]. Additionally attackers may try to compromise the vehicle's operation in real time by constantly re-sending legitimate frames across the network, this attack is known as a replay attack [6]. Bus-off attacks occur when the attackers constantly send bits in the identification field and other fields of the CAN frame. Lastly, Denial of Service (DoS) attacks happen when the attackers disrupt the normal processing of the in-vehicle communication by delivering CAN packets with high priority consistently across the network. These high priority packets then effectively block the valid lower priority packets leading to the chance of attackers gaining control of the vehicle. [6]

2.3. Successful CAN Bus Attacks

In a report by Koscher *et al.* [4] in 2010 they detailed their success in being able to carry out numerous attacks on the CAN bus of a vehicle. In this study the participants took two vehicles of the same make and model which were manufactured in 2009 and carried out experiments in different environments. The environments included, on the bench, where the hardware was extracted from the vehicle and tested on in the lab. Stationary, where the car was placed on jack stands and the experiments were carried out. Lastly the experiments were also conducted "On the road" where the vehicle was operated in a controlled environment and the experiments were carried out while the vehicle was moving. In all states which the vehicle was tested in the participants were able to successfully infiltrate the vehicles CA|N network and gain access to the brake control module, engine control module, body control module and the instrument cluster. The attack was made possible by connecting a laptop to the On-Board Diagnostics II (OBD-II) port of the vehicle. The attackers were able to manipulate the fuel levels of the vehicle and the speedometer readings to display falsified data on the instrument cluster. They were also successful in being able to disable the engine along with being able to manipulate the engine parameters like RPM and timing. The research also details an attack on the braking system of the vehicle while it was operational. While the vehicle was travelling in a controlled environment at 40MPH the attackers were able to release the brakes and prevent them from being activated again with the use of a continuous fuzzing method. [4]

It can be argued that the fact that the types of attacks mentioned need actual physical access to the OBD-II port then it makes the risk of them occurring low therefore not much of a concern. However,

research by Woo *et al.* [7] demonstrated how an attack could be carried out remotely using a malicious self-diagnostic smartphone app. This worked by getting the driver/user of the vehicle to download a malicious app to monitor or diagnose the vehicle, this then allows the attackers access to the vehicle without attaching any device physically. The attackers can then implement a long-range attack on the vehicle using the phones internet connectivity. Another remote attack survey was carried out in 12 car brands and 21 commercial cars by Valasek and Miller [8]. In each vehicle they identified the remote attack surfaces and the difficulties that are faced when trying to compromise the car. In their research they detailed that the remote attacks come in three phases, the first stage was compromising the ECU that controls the wireless interface. Stage two was injecting messages to communicate with the safety-critical ECU. The final stage was gaining control of the ECU and forcing it to behave in a malicious manner. The researchers concluded that due to the increasing number of cyber-physical systems in the vehicles the number of vulnerabilities will inevitably increase, however they cannot verify this practically due to the many different applications in the vehicles [8].

Over the Air (OTA) updates are also a new addition to the automotive industry. This allows the vehicle manufacturers to remotely send a patch or an update to the vehicles ECUs. This will now add a new attack surface that potential attackers can exploit in the future. There has been no research papers or evidence of any attack occurring via this vector yet, but it needs to be considered in the future.

In summary he vulnerability of the modern vehicles to cyber-attacks is a huge concern. The work carried out by Koscher *et al.* demonstrated a variety of attacks that could be carried out when they have physical access to the vehicles OBD-II port. This physical access may seem to limit the risk of an attack but the research by Woo *et al.* showed the potential for a remote attack by using a smartphone app for diagnoses however the app was malicious. Valasek and Miller demonstrated the potential for remote attack surfaces to be exploited in various car brands further showing the increase in the challenges faced when safeguarding against cyber threats. This then brings about the work in finding the best potential solutions to the problems being faced.

2.4. Potential Solutions

As the CAN protocol is a broadcast network then this allows any node access to listen to the messages being sent across the network. As CAN systems do not have encryption mechanisms an attacker may easily listen to the CAN traffic and be able to understand the messages being sent. [2]. Some encryption methods have been proposed to provide confidentiality and prevent attacks. A software-based approach to providing encryption exists commercially for example Trillium, who are a small Japanese based start-up company and have claimed that they have created a CAN bus encryption and key management system called SecureCAN. Which can be used for protecting payloads which are less than 8 bytes. Trillium claim to be able to use SecureCAN to encrypt CAN messages in real time with the use of its ultra-light weight block cipher [9]. However, as ECUs have generally low computational power that leads to weak encryption mechanisms this means that software-based encryption methods are not favoured as they are not strong enough. Adding encryption can also introduce latency to the delivery of CAN messages which for some critical components in the vehicle such as the braking system may introduce a delay in the operation which introduces a safety concern. We will examine the delay that can be potentially caused by introducing message authentication in the methodology section of this report.

The simplest way to provide security to the CAN bus is to introduce the idea of network segmentation by changing the network topology. This is achieved by separating the Critical and non-critical ECUs. This means that there would be a separate network topology for each grouping of ECUs therefore the messages on the CAN bus would not be an entire broadcast and only ECUs on the critical network will receive and monitor broadcast messages on the critical network. The connection for

communication between the networks is provided by a gateway ECU. This security measure is found on commercial cars already [2]. This method although it provides a better level of security than having no network segmentation it is not a solution that will guarantee full CAN network security as the gateway ECU can potentially be manipulated and the critical network can be infiltrated. This can be done with the use of malicious CAN frames that will contain the ID of a node that belongs to the subnet that is being targeted, as the gateway ECU is programmed to pass the relevant IDs to the subnetwork then the malicious frame with the correct ID will then be passed into the network. [2]. It is good that this type of segregation is being applied to the vehicles that are available commercially today as it is a step in the right direction to providing a better overall security to the CAN network in vehicles. However, network segmentation alone will not be able to stop attacks on the network and as the number of attack surfaces are growing in vehicles as the number of ECUs continue to increase it would be best to use network segmentation as one part of the defence along with another method and not be used in isolation.

Currently in the CAN protocol it is not possible to trace a CAN frame to find its source. There is also currently no authentication on the CAN which means that the CAN messages being sent across the network are not monitored as a result nodes can attach to the network and begin broadcasting messages without any authentication. Nodes with malicious intent can therefore inject CAN frames which will not be checked for authentication and the other nodes on the network will accept and process the message. VeCure was an authentication method which was developed by Wang and Sawhney [10]. The authentication method used in this proposal was based on the use of trust groups, which means that some nodes/ECUs were selected to be in the high-trust category, these high-trust ECUs then share the symmetric secret key. The authentication then works by first sending the data in the message immediately followed by an authentication message. The processing of this delay was only around 50us (microseconds), this is negligible due to the unnoticeable size of the delay however there comes another concern with this authentication method. The concern is that the number of messages that are being sent across the network on the high-trust group are doubled as they are receiving both the data message followed by the authentication message. This can therefore be considered not acceptable due to the limited bandwidth available on the CAN bus [2]. This approach does tackle the authentication of the messages and no doubt does provide a solution that is efficient in the validation of messages with a delay that can be ignored due to its miniature size. The bandwidth issue however may be a concern if the messages could be combined into one then it would be a better solution which would remove the need for the sending of the second authentication method. In this report we will examine this proposal in some detail and analyse the results of the tests based on the latency and the overall security enhancements of the protocol.

The CAN protocol’s vulnerability showcased by its broadcast type network which lacks encryption poses security risks. The encryption solutions mentioned such as SecureCAN face challenges due to the computational restrictions of the ECUs. Network segmentation while being a positive security measure falls short when trying to provide a complete robust security measure. VeCure’s authentication methos used trust groups and symmetric keys this offers a solution but has bandwidth concerns. The approach does validate messages but the doubling of messages on the high-trust groups may strain the bandwidth of the CAN bus. The next section will go through the environment and the potential solution for introducing message authentication to the CAN Network.

2. Methodology

The environment used for the examination of the research question **How can the implementation of message authentication mechanisms enhance the security of the CAN bus communication in vehicles?** Was the following.

After finding that the topic of this study would be to examine the application of message authentication to CAN messages and examine how the security of the message can be improved it was

found that a CAN bus was needed to make this possible. There are two ways to set this kind of environment up, the first option is to acquire hardware that have physical CAN bus connections however these are difficult to acquire as you either need direct access to a vehicle with an OBD-II port or to get the physical components from a vehicle manufacturer. This was not possible to achieve as this would require special permissions and from the vehicle manufacturer which in the timeframe and for this thesis along with the potential costs it was not an option. The second way this experiment could be made possible was with the use of virtual CANs (VCAN) and using phyton scripts to simulate a CAN environment that was possible to create and send CAN messages across. As this could be easily achieved and there are no additional costs or special hardware equipment needed to set this environment up then it was the chosen approach.

Firstly, a virtual machine (VM) was created, and the Ubuntu operating system was installed on it. The reason for using Ubuntu was that it can be used in a controlled safe environment that can be installed on a VM and kept separate from the host system this reduces the possibility of causing harm to your host machine. The Linux operating system which Ubuntu is built upon also has an API called SocketCAN which is a free package that can be installed onto the Linux environment which allows for the creation of virtual CAN sockets to be which can be configured and then used to simulate a realistic CAN network environment. CAN messages then can be configured, controlled, sent, and received without the need for hardware components. Although SocketCAN does allow hardware components to be connected to the PC and monitored and configured through the API we will be only using the virtual CAN environment in this study. Once the environment was decided the virtual machine created and the SocketCAN API installed the interfaces for the virtual CAN needed to be configured.

3.1. Configuring Environment

Each time the Ubuntu VM is launched then by default there is no VCAN interface, this interface is needed for the CAN environment and the CAN messages cannot be sent unless a VCAN interface is present. The first step was to create a script that can be ran through the terminal of the Ubuntu VM, this script can be run while having root privileges to launch the VCAN interface.

The shell script first gets super user access before then checking if a virtual CAN interface is already up and running. If this interface is not running yet, then the shell script will bring the vcan0 interface online. Once it is online a simple test can then be done to ensure that the interface is running and working as expected. The command ***“cangen vcan0”*** can be used to send randomly generated CAN messages across the network. By opening a second terminal and running the command ***“candump vcan0”*** the messages being sent and received will be displayed. If this second terminal displays a growing list of randomly generated CAN messages, then the virtual can interface has been set up and configured correctly.

3.2. Necessary CAN Message Python Scripts

To carry out this experiment it was decided that there would be four main scripts written, these were the following:

1. **attack_script.py:** This script was written to contain a basic CAN script with no authentication or encryption included. The script continuously sends a CAN message with a “malicious” payload on the virtual CAN interface. The script will continue to send messages every second until the user interrupts the script via a keyboard entry.

2. **no_security_receiver.py:** This script was created to be a CAN message receiver on the virtual CAN interface. The script has no security or authentication so it will log all kinds of CAN messages received be it malicious or not with a timestamp and store them in a file in the ***“can_logs”*** folder.

3. **can_sender_with_auth.py:** This script will continuously send authenticated CAN messages across the virtual CAN interface while using HMAC for message authentication. The HMAC value is calculated and appended to the original data. The messages are sent every one second until the user interrupts the script.

4. **can_receiver_with_auth.py:** This script receives the authenticated CAN messages on the VCAN0 interface. The incoming messages are read, and they are verified using the HMAC (Hash-based Message Authentication Code) with SHA-256 and then the messages are logged along with the authentication result.

These four scripts will be the four main scripts that will be used during the testing of the message authentication that was applied to the CAN messages. The scripts that do not have any authentication added to them will be used as the baseline for the experiment where the latency and the security of the scripts sending the CAN messages will be measured. These values will then be compared with the scripts that send and receive the messages that have the message authentication applied. From this we will then have a good idea of how the scripts compare against each other and then we can draw our conclusions from these results. The message authentication that was used in these scripts was HMAC with the hash function used in the scripts being SHA-256 (Secure Hash Algorithm 256-bit).

3.3 HMAC (Hash Based Message Authentication Code)

HMAC is a specific type of message authentication code that utilises a cryptographic hash function in combination with a secret key. The purpose of the HMAC is to provide data integrity and authentication. The main properties of HMAC are:

1. **Hash Function:** HMAC uses a cryptographic hash function which is denoted as H. The choice of the hash function is essential for the security of the HMAC.
2. **Secret Key:** The secret key (K) is required, and it is shared between the communicating parties. Keeping this secret key confidential is also key to the security.
3. **Message Padding:** The message is padded to match the block size of the underlying hash function. This padding ensures that the input size aligns with the hash functions requirements.
4. **Double Hashing:** The HMAC applies two hash operations to the data which it is encrypting. The secret key is used in both hash operations.

The general hash function algorithm is as follows:

$$\mathbf{HMAC\ (K, M) = H\ (\ (K \oplus opad\ )\ ||\ H\ (\ (K \oplus ipad\ )\ ||\ M)\ )}$$

- **K** is the secret key – It is established between the sender and the receiver.
- **M** is the message – This will contain the CAN data that will be sent on the network.
- $\oplus$ denotes the XOR operation.
- **ipad** is the inner padding – The padded message is XORed with a specific inner padding value (denoted as ipad). The result is then hashed using the SHA-256 function.
- **opad** is the outer padding – The original key is XORed with a specific outer padding value (denoted ad opad). The result is concatenated with the hash output from the inner hashing. The concatenated result is hashed using the SHA-256 function.
- **||** denotes concatenation. [11]

SHA-256 is a cryptographic hash function that belongs to the SHA-2 family of hash functions. The SHA-256 can take an input message and then produce a hash-value of a fixed size, in this case 256 bits. The main properties of SHA-256 are:

1. **Fixed output size:** The output size of the function will always be 256 bits.
2. **Collison resistance:** Computationally it is not possible to find two different inputs that will produce the same output. Therefore, avoiding any potential collisions.
3. **Deterministic:** The same input to the algorithm will always produce the same output.
4. **One-way Function:** It is computationally not possible to reverse the process and obtain the input to the algorithm from the output. [12]

The HMAC and SHA-256 were applied to the scripts **can_receiver_with_auth.py** and **can_sender_with_auth.py.** This added the message authentication to the scripts, and it was carried out in the following manner. Firstly, if we look at the sender script, the first step was to define the secret key, in this case for this example a basic secret key of **"MyKey"** was used and it was defined in the type bytes. The next step was the HMAC calculation which was handled by the line of code "***hmac_value = hmac.new(secret_key, data_to_send, hashlib.sha256).digest()[:4]".*** This line of code in the script uses the function ***"hmac.new"*** along with the SHA-256 hash function and the secret key. The output of this is then truncated to the first 4 bytes of the message. There is then a check on the message length of the CAN message. There is a comparison done that checks the total message length which is the original data + the HMAC which was calculated. This check is done as the maximum length of message that be sent on the CAN bus is 8 bytes. Before then sending the CAN message the authenticated CAN message needs to be created this is done by concatenating the data to send with the calculated HMAC value. The CAN message is then sent using a specific arbitration ID in this case ***'0x123'***

The receiver script named **can_receiver_with_auth.py,** handles the authentication of the received CAN messages. It verifies the HMAC of each message using the SHA-256 hash function and the shared secret key. After receiving the authenticated CAN message the first step is to calculate the received message and the HMAC. The received message is read and the HMAC value is then extracted from the last 4 bytes. Then a new HMAC value is calculated for the remaining data which is the received message minus the HMAC which was calculated in the sender script. This handling in the script is carried out in the code on the following lines:
received_hmac_value = can_msg.data[-4:]
calculated_hmac_value = hmac.new(secret_key, can_msg.data[:-4], hashlib.sha256).digest()
The last step of the CAN receiver script is to carry out the HMAC verification. The script will compare the received HMAC from the sender script with the newly calculated HMAC on the receiver script. If the message is authentic then these values will match if they do not match the message is considered not authentic and the script will log the event.

In summary, the sender script generates a HMAC value for each message being sent using SHA-256 and a shared secret key. The receiver script then verifies the authenticity of the received messages by comparing the received HMAC with the calculated HMAC. The use of HMAC and SHA-256 ensures the integrity and the authenticity of CAN messages in the communication.

3.4. Testing of Authenticated Scripts.

To get a baseline for comparing CAN messages including authentication we first needed to see the results of latency and security with CAN messages that did not have any authentication techniques implemented. Latency is the delay, or the time passed between the sending of the CAN message and the reception or the processing of message by its intended recipient. Latency in the CAN communication is a critical factor as it influences the responsiveness of the overall system since CAN messages are sent in real time. Therefore, along with the handling of the unauthenticated messages there will be the two metrics that will be used to judge the overall effectiveness of this message authentication method that used in this report.

The baseline test was conducted using the scripts which did not contain any authentication measures these scripts being **attack_script.py** and **no_security_receiver.py**. These scripts are configured to

both send and receive any type of CAN message without any kind of security built in. By running this baseline test we can then measure the latency of the messages as they are being sent across the network. As the messages are being sent across the same VCAN network then we can easily agree that no factors such as the environment or hardware difference can affect the latency of the messages.

To test the enhanced security of the messages we will use the sender script **attack_script.py** and the receiver script **can_receiver_with_auth.py.** The sender script will constantly send the receiver script *'malicious'* messages every second until the user stops the script. The receiver script should be able to handle these messages accordingly and not allow them to communicate due to the missing authentication on the sender script.

To then test the latency of the authenticated CAN messages we will use the scripts **can_receiver_with_auth.py** and **can_sender_with_auth.py** the baseline results that were collected from the previous scripts will be compared with the results from running the latency test on the scripts that contained the authentication handling of the messages.

4. Results

The first test was to conduct how effective the authentication management was of the sending of the CAN message without authentication to the CAN receiver without authentication. By running the attack script and sending it across the CAN network to the CAN receiver with no authentication we could see clearly that each message was being logged. Which demonstrates that without any authentication measures put in place and due to the broadcast nature of the CAN protocol the CAN messages without authentication will always be logged and read by the can receiver without authentication. This is a security concern as it allows for the potential of attack scripts that do not contain any authentication and may include malicious content to be passed along the CAN network and be received by a node, therefore allowing for the potential to cause harm to the normal functionality of the vehicle.

The second test on the authentication was the used the same attack sender script on the CAN receiver with authentication. By sending the attack script to the receiver node that has the authentication measures added then the receiver script should not accept these messages due to the lack of the HMAC and SHA-256 on the sender script. The result of this was that the receiver script still receives the message however after the comparison of the expected HMAC values on the data being sent the CAN message is flagged as not authentic. The CAN receiver script (Node) will then discard this message and no further action will be taken until a CAN message with the correct and expected HMAC value is being received. By running this test and seeing the results we could clearly see and demonstrate how by adding authentication measures to the node then it improves the security as only authenticated messages will be read and logged. This therefore increases the overall security of the CAN network and reduces the risk of falling victim to an attack.

The final test on the authentication handling of the scripts was the using the CAN message with authentication script to send data to the CAN receiver with authentication. Here both scripts have the necessary HMAC and Sha-256 included. Therefore, the HMAC comparison can be carried out on the sent and received CAN messages and once the same HMAC value was found on both ends then the message was labelled as successful. The CAN message can then be delivered to the receiver node and the necessary actions be carried out. The result of this test shows that only when the expected HMAC values with the correct authentication is received then the message will be successful and logged. By adding this authentication to the CAN messages, it protects against scripts like the attack script earlier from sending malicious data to a vulnerable node. By stopping these messages when they are first received it reduces the attack surface and the potential of disrupting the normal operation of the CAN network.

The table below details the results of the above tests. Each script was tested against each other and the results of whether the message was successful was recorded. From the results table the only way to protect fully against the sending or receiving of a malicious CAN message is to have message authentication on the CAN at both ends.

Sender Script	Receiver Script	Sender Authentication	Receiver Authentication	Message Accepted	Protected From Attacks
can_sender_with_auth.py	can_receiver_with_auth.py	✓	✓	✓	✓
attack_script.py	no_security_receiver.py	X	X	✓	X
attack_script.py	can_receiver_with_auth.py	X	✓	X	X

The next step in the results was to test the latency of the messages being sent and compare the latency between the CAN messages that had authentication included and the CAN messages that did not include any authentication measures. To conduct this test, we ran the scripts concurrently for one minute and logged the results of the latency on both types of CAN messages. This test was conducted 10 times, and the average latency was then calculated compiling the results from all 10 tests. The latency results between the authenticated and the non-authenticated messages were then compared. To record these results, we needed to create two separate scripts one to measure the metrics with authentication called **"metrics_with_auth.py"** and one the measure the metrics without authentication called **"metrics_no_auth.py".** These scripts would constantly measure the CAN messages being received on the CAN receiver either with authentication or without authentication and then the average latency was then calculated on each run of the test. These results were recorded 10 times, and the average of the 10 tests latency was calculated. The results from these tests are shown in the metrics table below.

		Test 1	Test 2	Test 3	Test 4	Test 5
Sender Script	Receiver Script	Latency (seconds)	Latency (seconds)	Latency (seconds)	Latency (seconds)	Latency (seconds)
can_sender_with_auth.py	can_receiver_with_auth.py	0.000510915	0.000377569	0.000416466	0.000377225	0.000332774
attack_script.py	no_security_receiver.py	0.000272133	0.000427481	0.000361685	0.00025535	0.000257285
attack_script.py	can_receiver_with_auth.py	0.000521207	0.000438295	0.000363906	0.000315858	0.000311527

Sender Script	Receiver Script	Test 6	Test 7	Test 8	Test 9	Test 10
can_sender_with_auth.py	can_receiver_with_auth.py	Latency (seconds)	Latency (seconds)	Latency (seconds)	Latency (seconds)	Latency (seconds)
		0.000494566	0.000295741	0.000427985	0.000543471	0.000602202
attack_script.py	no_security_receiver.py					
		0.000272004	0.000274904	0.000218528	0.000205461	0.000272235
attack_script.py	can_receiver_with_auth.py					
		0.000309901	0.000328545	0.00028133	0.000353332	0.000446812

Results

Sender Script	Receiver Script	Average Latency (Seconds)
can_sender_with_auth.py	can_receiver_with_auth.py	0.000437891
attack_script.py	no_security_receiver.py	0.000281707
attack_script.py	can_receiver_with_auth.py	0.000367071

The table above shows the results of the latency on each test along with the scripts that were used to get the latency results. The smaller table that is shown below is the results table. This takes the results that were received over the 10 tests and then calculates the average latency on the messages.

These results show that the average latency when there was no authentication in the script was **'0.000281707 seconds'** this average latency however then grew to an average latency of **'0.000437891 seconds'** once the authentication was applied to the message. This represents an average increase of '**0.000156185 seconds',** or in other words it represents approximately a **35.67%** increase in the latency between the CAN messages without authentication to the CAN messages with authentication. Although it can be argued that the increase in latency is so small that the difference

could be ignored the percentage increase gives a better understanding of how much of an increase this is. It also shows how quickly these messages are being sent across the CAN network, the latency is so small that to the human eye there seems to be no delay at all when in reality the difference could be seen once the metrics scripts were ran. As CAN messages are supposed to be in real time and the CAN messages are responsible for sending critical messages such as brake pedal controls, fuel levels, warning lights, etc. then this increase could potentially lead to driver safety concerns if a vital CAN message is sent.

5. Conclusion

The research paper took an in depth look in enhancing the security of the Controller Area Network bus communication in vehicles with the implementation of message authentication on the CAN messages. The research aimed to address the vulnerabilities and the security challenges that are faced by thew CAN bus protocol as now modern vehicles rely on electronic control unites (ECUs) for many different functionalities. The literature review provided a comprehensive overview of the CAN protocol and explained in detail the areas which the protocol was implemented in. The CAN protocol continues to evolve with the introduction of CAN FD (Flexible Data-Rate) which shows the increasing demand for the higher transfer speeds and payload capacities as the application of the CAN protocol spreads beyond the automotive industry. The literature review also highlighted the main reasons why CAN protocol is an ideal fit for the automotive industry this being the robustness, reliability, and suitability for real-time communication in harsh environments.

The methodology section detailed the setup of the experimental environment which explains the use of the virtual can s (VCANs) and the python scripts which were created to simulate a working CAN environment. Using the Ubuntu operating system along with the SocektCAN API it allowed for the creation of a controlled safe environment for conducting the research. This approach also eliminated the need for physical CAN bus connections and allowed for the configuration and transmission of CAN messages without the need for additional hardware components.

The focal point of the study was to investigate the message-based authentication on the CAN bus Hasb Based Message Authentication (HMAC). The use of baseline tests and vulnerability analysis which were conducted using python scripts showed the usefulness and demonstrated the security concerns of the CAN bus architecture.

The findings of the tests which were conducted were analysed and two main points were deducted. The first being that by adding the HMAC and SHA-256 message authentication technique to the CAN messages then it did increase the overall security. By adding this, only messages that contained the correct and matching HMAC values were recorded by the nodes and all other messages were rejected. The results showed that only by adding the authentication to both the sender and the receiver scripts then the CAN network will be safe from potential attacks. In conclusion for this the CAN protocol undoubtedly does become more secure after adding message authentication. However, the latency was also measured before and after adding the message authentication. As the CAN network is supposed to operate in a “Real-Time” environment the latency in the messages needs to be at an absolute minimum. The results of latency were recorded on the network both before and after adding the message authentication. The results showed that there was an increase of **0.000156185 seconds** on average once the message authentication was added, which represents an increase of around **35.67%.** Although the value in seconds does not represent a large delay in seconds evaluating the increase as a percentage gives a bit more insight into how big of an increase this is. As the CAN bus is built on the premise of providing real-time high reliability communication between ECUs minimal latency is crucial to its operation. As a result, increasing the latency even ever so slightly is detrimental to the expected performance of the network. Many applications in the vehicle require real-time responsiveness, for example the communication between ECUs for functions like engine control, braking, and transmission and having delays in the transmission of these messages can lead to reduced

system responsiveness and therefore affect the safety and performance of the vehicle overall. The performance of the system also depends on the efficient communication between ECUs, so the latency needs to be kept to the bare minimum. Any increase in latency can increase bottlenecks and reduce the overall system performance which will affect the user experience and may also cause disruption to the normal operation of the system. Therefore, minimal latency is crucial in CAN bus architecture and to ensure real-time communication, responsiveness, safety, and overall system performance. Any increase in latency can have detrimental effects on functionality and reliability of the system.

To answer the research question **How can the implementation of message authentication mechanisms enhance the security of the CAN bus communication in vehicles?** We can say the following. Adding message authentication measures to the CAN bus protocol with the use of HMAC and SHA-256 the security of the CAN bus communication was improved. It protected against non-authenticated messages and reduced the likeliness of falling victim to an attack. However, the introduction of the message authentication also increased the latency which is not a desirable metric to increase in the operation of the CAN bus. This increase may diminish the attractiveness of the implementation of this solution since the CAN bus us supposed to be a real time network.

Future work may include introducing a hybrid type solution using both network segmentation and message authentication. The vehicle network can be split into two areas time-critical and non-time critical. The message authentication can be applied to all non-time critical CAN messages and the time critical messages can exist inside their own segmented network. This segmentation and application of the message authentication can then lead to a more robust and overall, more effective solution to Enhancing CAN Bus Security Using Message Authentication.

Bibliography

[1] O. Avatefipour and H. Malik, 'State-of-the-Art Survey on In-Vehicle Network Communication "CAN-Bus" Security and Vulnerabilities'.

[2] M. Bozdal, M. Samie, and I. Jennions, 'A Survey on CAN Bus Protocol: Attacks, Challenges, and Potential Solutions', in *2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, 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', *Sensors*, vol. 20, no. 8, Art. no. 8, Jan. 2020, doi: 10.3390/s20082364.

[4] K. Koscher *et al.*, 'Experimental Security Analysis of a Modern Automobile', in *2010 IEEE Symposium on Security and Privacy*, Oakland, CA, USA: IEEE, 2010, pp. 447–462. doi: 10.1109/SP.2010.34.

[5] N. Liang and D. Popovic, '2 - The CAN bus', in *Intelligent Vehicle Technologies*, L. Vlacic, M. Parent, and F. Harashima, Eds., in Automotive Engineering Series. , Oxford: Butterworth-Heinemann, 2001, pp. 21–64. doi: 10.1016/B978-075065093-9/50004-9.

[6] R. S. Rathore, C. Hewage, O. Kaiwartya, and J. Lloret, 'In-Vehicle Communication Cyber Security: Challenges and Solutions', *Sensors*, vol. 22, no. 17, Art. no. 17, Jan. 2022, doi: 10.3390/s22176679.

[7] 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 Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 993–1006, Apr. 2015, doi: 10.1109/TITS.2014.2351612.

[8] C. Miller and C. Valasek, 'A Survey of Remote Automotive Attack Surfaces'.

[9] 'CAN Bus Can Be Encrypted, Says Trillium', EE Times. Accessed: Nov. 28, 2023. [Online]. Available: https://www.eetimes.com/can-bus-can-be-encrypted-says-trillium/

[10] 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)*, Oct. 2014, pp. 13–18. doi: 10.1109/IOT.2014.7030108.

[11] 'What is Hash-based Message Authentication Code (HMAC)? – TechTarget Definition', Security. Accessed: Nov. 14, 2023. [Online]. Available: https://www.techtarget.com/searchsecurity/definition/Hash-based-Message-Authentication-Code-HMAC

[12] 'What is SHA 256 Algorithm? Functions & Applications', upGrad blog. Accessed: Nov. 14, 2023. [Online]. Available: https://www.upgrad.com/blog/sha-256-algorithm/