University of Amsterdam

Project Paper

Cybersecurity in Automotive Networks

February 9, 2020

ing. Arnold Buntsma
arnold.buntsma@os3.nl
UvA ID: 12818674

ing. Sebastian Wilczek
sebastian.wilczek@os3.nl
UvA ID: 12837067

Assessor:
Prof. dr. ir. Cees de Laat
delaat@uva.nl
University of Amsterdam

Supervisor:
ir. Colin Schappin
cschappin@deloitte.nl
Deloitte

Course:
Research Project 1

Abstract

Until recently, vehicles were a stand-alone domain. They were not connected to anything other than themselves. However, new vehicles have a connection to the Internet, often to some service of the manufacturer. This introduces a new attack vector. It has already been proven that this can be used to gain access to the automotive network. This network contains Electronic Control Units (ECUs) that realize almost all functions in a vehicle e.g. engine control, airbags, windows, etc. Most of the network protocols for vehicles have been designed when the vehicles were not connected and therefore do not have any security features implemented. Even recently developed protocols did not take the network security into account. This paper is focused on investigating the measures to secure these protocols with extensions and how they address the CIA triad. We looked at the Controller Area Network (CAN) and FlexRay protocol. There are many CAN extensions proposed in previous literature and most of them behave similarly, by adding authentication to messages. We looked closely at CaCAN and a hashing-based authentication scheme and found one of them to be practically secure. For FlexRay, no extensions were found to improve security. We ported the later extension to FlexRay and deemed it to be practically secure. Even though these extensions improve the security of the network, they might reduce the security of the vehicle because of the added processing time to authenticate messages. Results of our experiments showed an added processing time of 46ms.

Keywords— CAN, FlexRay, Automotive Networks, Electronic Control Units, Car Security

1 Introduction

Automotive vehicles are comprised of multiple Electronic Control Units (ECUs), each controlling a subsystem of the vehicle. These include, but are not limited to, engine controls, brakes, locks, climate control, and multimedia systems [10]. To reduce the number of interconnections required between these ECUs, Bosch developed the Controller Area Network (CAN) protocol, first released in 1986 [5].

CAN allows multiple microcontrollers to communicate on a single bus using the same protocol. CAN has been standardized in ISO 11898 and is used in almost every car on the market, and in other vehicles, machinery and even prosthetics [2]. With the constant evolution of technology, the amount of ECUs in a vehicle has increased and reached a level where management of the network has become complex [22]. Systems like Adaptive Cruise Control (ACC) are part of these developments. ACC requires real-time data from cameras and multiple sensors which demands more bandwidth than CAN offers. Therefore, some automotive companies teamed up to create a new bus that meets those requirements, called FlexRay [19], first introduced in 2006 [26].

A vehicle nowadays consists of multiple function domains [20]. The powertrain and chassis domains are concerned with control (gas, brake, suspension, transmission) of the vehicle. The third function domain is the body domain which entails, among others, the dashboard, wipers, lights, windows and climate control. The telematics, multimedia and Human-Machine Interface (HMI) domains are for human interaction, media and communication like hands-free phones, radio and navigation. The last but not the least important domain is the (active and passive) safety domain. This entails safety systems like ACC, belt pretensioners, airbags and impact sensors. These domains have different requirements and do not use the same bus as other domains. For example: opening and closing a window or changing the ventilation and heating are less important than braking or steering.

Modern vehicles are more likely to be connected to the Internet or services of the manufacturer [27], and some cars have an Internet connection by default [8]. The connectivity of a vehicle increases the possible attack surface [27]. Whereas previously you had to have physical access to a car to get to the connection bus of the network, you can now get to the bus via the Internet [6, 27]. If one has control over the different busses in a vehicle, they might control all the features that are connected to the bus. This could be as invasive as full control over the vehicle with powertrain, steering, braking, multimedia and comfort controls [20].

Car manufacturers offer services that allow the car to be located, turned on or off, and/or control the heating from your phone. Hacking the servers of a manufacturer is another attack vector, and also the most exploited one [27]. If hackers gain access to the servers that realize these functions, they can do the same things to the cars as the original owners and perhaps more. [8] states that fleet attacks can be very devastating. "If only one model year is affected by the hack, we can still expect about 3,000 deaths—about the same as 9/11.".

It is also possible to write a worm that spreads easily through cars on the road. To cite [6]: "Since a vehicle can scan for other vulnerable vehicles and the exploit doesn't require any user interaction, it would be possible to write a worm. This worm would scan for vulnerable vehicles, exploit them with their payload which would scan for other vulnerable vehicles, etc."

This indicates the danger of cyberattacks on vehicles and automotive networks not being secure. We look at the networks, what security features they have by default and what extensions are available to secure the networks. We will advise on the security measures that can be taken. In this research project, we look at the security of the automotive networks themselves. We consider if there are measures taken to protect them against malicious messages and, if not, if there are extensions that do and how those affect the performance of the bus.

This project is done as a research project for the master program Security & Network Engineering of the University of Amsterdam. It was executed under guidance from Deloitte. The subject was also proposed by Deloitte.

The rest of the paper is structured as follows: The research questions are defined in section 2. Section 3 gives an overview of related work. In section 4 the methodology and our approach are explained. Section 5 explains the automotive networks covered in this research. Section 6 explains the security features and proposed extensions of those protocols. In section 7 the performed experiments are explained. Section 8 gives the results of the experiments. In section 9 we conclude the research while discussing the research in section 10. And finally, in section 11, suggestions for future work are given.

2 Research Questions

To define the topic to be researched, the following research questions have been defined. These are answered within this research paper.

1. Which automotive communication protocols are used in production, forming the state of practice?
2. What features are built into the protocols utilised in the automotive industry to provide security?
3. What extensions can introduce security to the protocols?
4. How do these extensions compare in terms of security, according to the CIA triad?

3 Related Work

Since the CAN bus was introduced back in 1986, the protocol has matured for a long time. Because of this, the security and performance of the bus have been researched many times, both on its own and compared to alternatives.

CAN itself has been researched to not be secure on its own. According to [7, 10, 14, 22], it is vulnerable since it does not provide authentication or encryption to any sent message, while every connected ECU can receive all messages. The low bandwidth of CAN inhibits well-performing security protocols and methods.

In [21] they researched the security of FlexRay and executed an attack in a simulation environment. This also succeeded and points out flaws in the protocol. However, there is only limited research regarding attacks on the FlexRay protocol. This could be because of its only recent adoption in the automotive industry [12].

Various approaches exist to improve the security of CAN. Some of these include CANAuth [11], CANGuard [16], LeiA [23], SafeCAN [13], an approach by Bosch [17], and some unnamed solutions [3, 25]. The solutions of CaCAN [15] and a hashing-based authentication scheme [9] are considered in-depth in this research.

There has been research on the state of practice of automotive communication protocols in the past but this was done in 2005 and 2006 [20, 22]. We considered if this research still holds or if the state has changed.

4 Approach and Methodology

In this research, we made use of literature and experiments. The literature research was concerned with uncovering what automotive network protocols are in use in the automotive industry and how they are defined or standardised. Specifications and previous research on different protocols and their security extensions were taken into account.

To conduct experiments to research the security of protocols and extensions, we defined attacks that aim to introduce malicious messages onto the bus in question, to trigger undesired behaviour of a target ECU. These experiments were executed on multiple setups, each involving a different protocol or security extension of the network.

The setup itself involved multiple ECUs and one protocol. The experiments themselves were fully simulated using bus simulation software or ran on dedicated hardware. The software used for that purpose is CANoe [28]. In the case of hardware, we used low-cost hardware, namely Arduino microcontrollers, to create ECUs and a network.

The measurements conducted during the experiments involved both the success of the defined attacks and the delays introduced because of the calculations required by the security extensions tested. Other values, such as available bandwidth and payload sizes of sent messages were acquired through literature research.

5 Automotive Networks

In this paper, we focus on the automotive networks that connect and control the most important functions of a vehicle. We limited our research to CAN and FlexRay since they are the most widely used in the industry [20]. This section describes the requirements and constraints of automotive networks and its protocols in more detail.

5.1 Communication requirements

In [22] communication requirements are set to which an automotive network should comply. These have been set up in 2005 but are still relevant. The requirements are:

- **Fault tolerance** Tolerating faults by using for example redundancy, error checking and correction (CRC).
- **Determinism** Determine which message should be used in case of multiple messages and if the message is still 'new' enough to use. In vehicles, it is important to use the latest data possible because the situation can change anytime. Determinism makes sure that the used data is significant.
- **Bandwidth** High bandwidth is needed for new systems like ACC but high bandwidth is costly to implement in automotive networks. Therefore, there is a trade-off for which ECUs need this. The less bandwidth requiring ECUs are connected to a low-bandwidth network to save costs.
- **Flexibility** Networks need to be flexible because they need to handle a variety of messages like event-driven and time-triggered events. The network also needs to be flexible in the load that it can handle because at one moment when a vehicle is idling and standing still there are fewer messages on the bus than when it is driving through a city and moving a lot.
- **Security** The communication on the network should be non-malicious only. Hackers should not be able to alter a message without being noticed. Unauthorized access should not be possible.

5.2 Constraints

[24] summarizes specific hard- and software constraints of automotive networks. Automotive networks differ from other computer networks and have some specific constraints. These are:

- **Hardware limitations** The ECUs in vehicles are embedded systems with limited resources. Often they only have very limited memory and computing power. This makes it hard to implement computationally expensive

operations like cryptography, especially when the cryptographic operations should not have a significant impact on data transmission. The ECUs are also exposed to conditions most other systems are not exposed to, like vibrations, temperature differences, shocks, etc.

- **Timing** Some ECUs perform tasks such as applying the brakes and the Anti-lock Braking System (ABS). These are time-sensitive and must be executed as soon as possible, ideally immediately. Security measures should not add a significant delay to this, otherwise, it could mean the difference between life or death.
- **Autonomy** Everything the vehicle does must be as autonomous as possible to have the driver focusing on their driving.
- **Life-cycle** A car usually has a longer life-cycle than many other consumer electronics. A laptop gets replaced after roughly 3 years, but a vehicle that is 3 years old is still considered very young. So the hard- and software must be durable and possibly (easily) updatable.
- **Supplier Integration** Suppliers of vehicles defend their intellectual property by providing components without releasing the source code. Modifications, to improve security or increase performance, can be difficult to implement. Components originating from different suppliers might have to be able to communicate with each other.

5.3 Network protocols

Vehicles nowadays use a combination of different network types. The connected ECUs are based on the characteristics of the network. For example, modifying your mirror setting does not need much bandwidth and is a low priority operation. Therefore this is often realized via a CAN bus because it has a lower bandwidth and is less expensive than a FlexRay bus. However, systems like autonomous driving need a high bandwidth, low latency network for fast delivery of unprocessed video and sensor information, therefore needing a FlexRay or Ethernet network [22].

CAN - The (to this day) most implemented protocol is the CAN standard [20]. It resides in almost all vehicles, ranging from cheap to expensive and old to new. The reason for this is that it is a relatively cheap network with moderate performance. There are different standards for CAN used in Europe and the US, however, these all work in a similar way [22] and thus the difference between those will not be considered in this paper. As aforementioned, there are a lot of attacks possible on the CAN bus. This is because when CAN was designed, security was not a requirement. This was not needed because, in the eighties, the only way to get access to the bus was to have physical access. There are two types of CAN busses: CAN Low Speed & CAN High Speed. The low speed has a maximum bandwidth of 125Kb/s, and the high speed has a bandwidth up to 1Mb/s [24]. The low speed is usually used in the body domain and the high speed in the powertrain and chassis domain. The network topology is a bus-type network. This means it is a broadcast-only network and the ECUs 'subscribe' or listen only to messages they are programmed to listen to. This, however, makes the network susceptible to Denial of Service (DoS) attacks by design [3]. Using a prioritization feature based on message IDs does not help this case. CAN messages use a message ID for identification, routing and prioritization [24]. A malicious intended person could abuse this by creating a DoS attack with the ID of 0 and thus being the highest priority on the bus, rendering the CAN bus unusable.

FlexRay - FlexRay is a newer automotive network protocol. This is already adopted by many European car manufacturers, either as a backbone and/or for ECUs that need more bandwidth for advanced functions. FlexRay has a maximum data rate of 10 Mb/s, is deterministic, and fault-tolerant by design [20]. This makes it suitable for real-time use cases, like ACC and autopilot. It is deterministic by using a time-triggered communication model and the fault tolerance is realized by communication channel redundancy [18]. The data rate can be increased to 20 Mb/s by using two channels for simultaneous data transmission, but this comes in at the cost of losing the redundancy [18]. Even though FlexRay is a newer protocol and focused on being the next big protocol, security has been neglected, as the authors of [21] proof in a simulation. Even after those results were published, no additional security has been added to FlexRay. In a later paper [18] DoS attacks have been proven to be possible using the protocol.

In this paper, we will focus on these two network protocols because they are the most used protocols in the global market [20].

6 Security and Extensions

Automotive networks only have features implemented for availability and integrity via redundant channels and CRCs. By default, the CAN and FlexRay protocol do not have any authentication or confidentiality measures in place. Confidentiality is not very important in these networks because if, and only if, authentication and integrity are guaranteed, then one should be able to see the messages and still not be able to (re)create a legitimate message.

To address this incompleteness, extensions for the CAN protocol have been proposed [9, 11, 13, 15]. These extensions use a hash to authenticate messages on the networks. Most of these extensions use similar approaches to secure the network. They use (pre-shared) keys and freshness to make sure that the hashes are not susceptible to replay attacks. They use a cryptographically strong hashing algorithm like SHA256. The freshness will be updated every time the vehicle does an action on an ECU, and the keys can be periodically changed to withstand a brute force attack.

CaCAN [15] uses an 8-bit hash. The hash is calculated by the transmitting ECU and 8 bits of it are transmitted as part of the CAN message payload. Since CAN messages have a maximum size of 64 bits, that leaves 56 bits for the actual payload. The extension furthermore includes a counter which increases with every usage of a specific message, for example, every time the brake is pressed. Both ECUs maintain this counter individually. This counter ensures that the next required hash is always different from a previous one. The paper furthermore proposes the usage of a monitor ECU. Instead of having each ECU validate received hashes themselves, the monitor node shares a key with any ECU that is authenticated to send certain types of messages. If the monitor ECU receives a message with an unfitting hash, it sends out an error frame, causing a collision. This removes the malicious message. Please note that we could not implement such a monitor ECU in the scope of our experiment setup. This part of the proposed extension is therefore discarded, with each receiving ECU validating the received hashes instead.

The extension of [9], however, uses a 24-bit hash. In principle, the extensions are otherwise very similar. Again, a pre-shared key is required, which is, together with a counter, incorporated into a hash that is part of the payload. The increase of the size of the hash results in a maximum message size of 40 bits. Furthermore, a key renewal solution is part of the extension. One specific node may send out a request to renew the keys in all other related ECUs. Please note that this renewal uses the pre-shared key as a seed. On startup of the bus communication, a random value is transmitted by the mentioned node, which can be intercepted. Therefore, if the pre-shared key is ever compromised, an attacker may also renew the key of a compromised ECU. Otherwise, the extension operates by sending out an authenticated message with a hash, with the receiving ECUs calculating the hash, dropping unauthenticated messages.

For this research, we focused on these two extensions for the CAN protocol. We did not find any security extensions proposed for the FlexRay protocol.

7 Experiments

To answer our research questions, we set up two experiment environments. The first environment consisted of a vehicle network simulated in software. The software used for this purpose was CANoe [28]. It enabled us to create ECUs, and the ability to program them. These ECUs were connected to a virtual bus, without the need for any hardware. The busses simulated were both of a CAN and a FlexRay variant, depending on the experiment. An example of the simulation interface can be seen in Appendix A, Figure 3.

The programming of the ECUs was done in CAPL, a dedicated programming language for CANoe simulation environments. We defined the various messages that can be transmitted on the busses in a database connected to the simulations. These messages contain information regarding the various functionalities of any ECU and additional definitions required for the different authentication extensions. The configuration created for these experiments can be found on GitHub [29].

We created another setup using CAN hardware. In this setup, we used two Arduino microcontrollers to act as ECUs, connected through a CAN shield, creating a CAN bus with two ECUs. This setup can be seen in Figure 1. A FlexRay setup was not created due to time and budget constraints. The scripts developed for this setup can be found on GitHub [4].

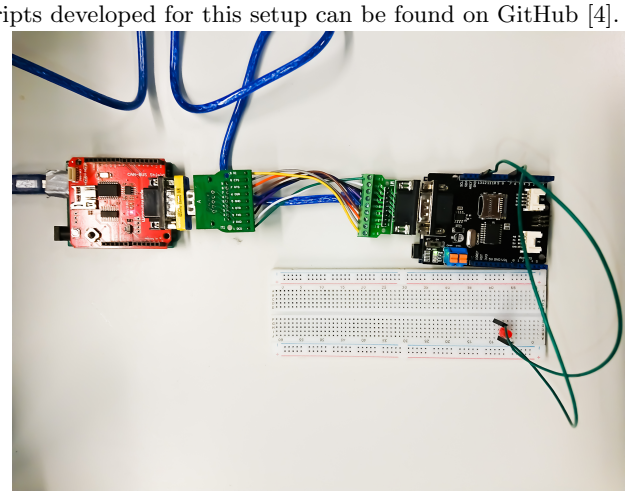

Figure 1: CAN Setup with Arduino Microcontrollers

In both cases, the goal of the experiment was to introduce malicious behaviour using the CAN or FlexRay bus. In our setup, this meant turning on the brake lights without pressing the brakes. Please note that the lights represent functionality that may be exploited or activated by an attacker. We assumed that if we could turn on a light, we could also perform other malicious actions on other ECUs, as long as they perform in a similar pattern and are connected to the same bus. In our setups, we programmed some ECUs to perform as expected, as in, for example, a brake ECU to send a brake signal, while others we were free to program as we wish, as long as these ECUs remain unaware of the source code of the normal ones. These ECUs represent any component that may be compromised by an attacker. Depending on the experiment, the ECUs were programmed to make use of the aforementioned extensions.

The experiments measured whether the proposed authentication scheme or the busses by themselves can perform tasks as expected, if they are susceptible to replay or exhaustive search attacks and how the performance and bandwidth of the transmission are affected. Due to the software simulation nature of CANoe, performance evaluations are only performed on the hardware setup.

8 Results

During the experiments, we observed that ECU that are connected to either CAN and FlexRay are susceptible to replay attacks. A compromised ECU could read messages sent to the bus since these messages are receivable by any connected ECU. Retransmitting the same message resulted in the receiving ECU to repeat the same behaviour, in our case turning on a brake light. This happened whether or not the brake ECU had any brake pressure applied.

Making use of CaCAN prevented the use of replay attacks. The hash calculated was transmitted with every message, and since there is freshness in the form of a counter, a replayed message resulted in the CaCAN-aware ECUs to recognize the replayed message as invalid. However, the transmitted hash consists of only 8 bits. The malicious ECU could transmit random hashes, eventually turning on the brake lights, after at most 256 attempts.

The hashing-based authentication scheme also prevented replay attacks successfully. We also could not break the authentication within any reasonable time frame. We propose that a reasonable time is the time between brake state changes, as in how often brake pressure may be applied in an actual car. We implemented a version which could trigger a key renewal once a dedicated ECU observed an unauthenticated message. This renewal further prevented exhaustive search attacks. The authentication took up 24 bits of the message, leaving only 40 bits for the rest of the payload. This did not impact the functionality of our test setup since the only other information sent is a single bit indicating the to-be-state of the brake light. However, other implementations would need to limit themselves to a message size of at most 40 bits.

An attacker would have to crack 24 bits in the average period of about 60 seconds, which is not feasible: if we assume that an attacker uses the full capacity of the network to brute force the hash, that would be 500 Kb/s for CAN. It will take at most $2^{24}/(500000/64) = 2147.48$ seconds or 35.79 minutes.

Since we could not find any security extensions for FlexRay, we instead attempted to port the aforementioned hashing-based authentication scheme to FlexRay. We left the ba-

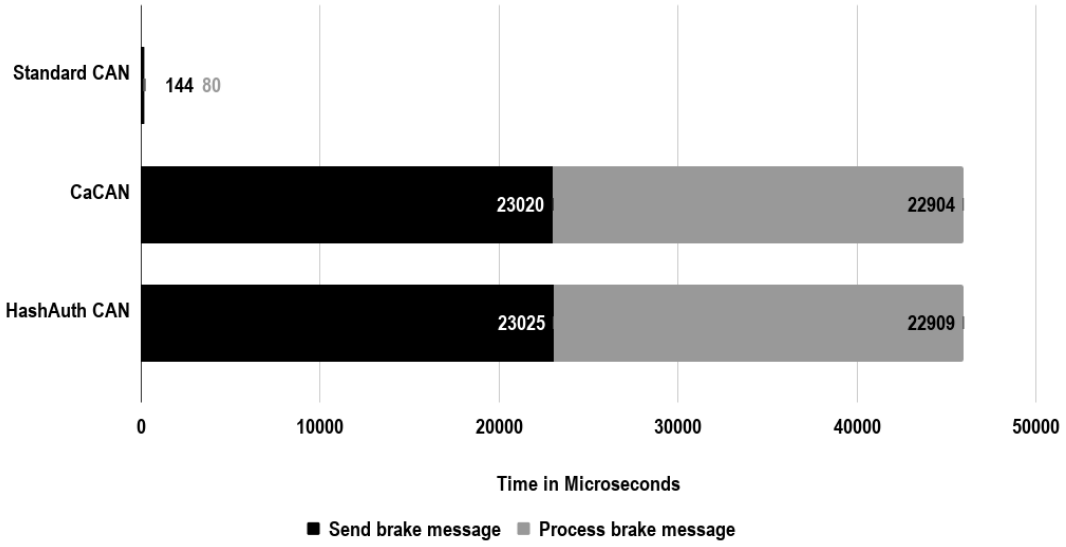

Figure 2: CAN Hash Calculation Time per extension, rounded to microseconds

sic functionality unchanged. The extension calculates a hash based on the information being sent and a key that is generated from pre-installed information. The receiving ECU then calculates the same hash based on the message it receives. In case of success, a counter is changed for freshness. The message is only processed if the hash matches at the receiving ECU. When considering that FlexRay has a capacity of 10 Mb/s it will take a most $2^{24}/(10000000/256) = 429.50$ seconds or 7.16 minutes. To increase that time, we modified the hash size to 28 bits, increasing the required time to a maximum of 6871.95 seconds or 114.53 minutes.

Given that FlexRay has a maximum message payload size of 256 bits, the size of the hash sent is less of a problem compared to CAN. The remaining bits still offer enough space for larger messages. However, CaCAN, the hashing-based authentication scheme and various other extensions that we briefly considered for evaluation [3, 11, 13, 16, 23, 25] come back to the same issue. They either are not secure enough because only a few bits are allocated for authenticity or they compromise the bandwidth of the bus.

Aside from the reduced capabilities of the bus itself, we also observed an introduced delay due to the hash calculations required for the extensions. A delay comparison can be seen in Figure 2.

CAN messages take 224 microseconds on average to be both sent by one ECU and to be processed by another, as measured on the Arduinos in the hardware experiment setup. The transmission time is not taken into account here since it remains unchanged between different extensions and only depends on the physical relay speed of the bus. However, if the ECUs have to calculate a hash whenever they send or receive a message, the processing time increases drastically. In both the case of CaCAN and the hashing-based authentication proposal, the time to complete computation increases to an average of 45,924 and 45,934 microseconds, respectively, with a standard deviation of fewer than 9 microseconds over a set of 1000 measurements.

Given that a car might drive at 100 kilometres per hour, 224 microseconds amount to 6.2 millimetres of distance travelled before the brakes engage. With almost 46 milliseconds calculating hashes, that distance increases to 1.28 metres.

The receiving ECU has to calculate a hash every time it receives a message that is supposedly authenticated. Even if a compromised ECU sends random bit strings, the receiver is forced to perform cost-intensive calculations to compare the two values. We could prove that these messages, when sent in quick succession, can easily drown out authenticated messages, causing unpredictable behaviour on the receiving ECU. Since we only had one receiving and one sending ECU in our hardware setup, we could not execute such a DoS attack on CAN hardware. With only one Arduino sending messages and one receiving, one single Arduino would have to both send legitimate messages as well as perform the DoS attack. The scheduling and the performance of the sending Arduino would have massively influenced our results.

9 Conclusion

This research has shown that the most commonly used standard for automotive networks used in the industry is CAN, with FlexRay gaining ground. A CAN bus is used by almost every current vehicle manufacturer and in other machinery.

Our experiments have shown that CAN and FlexRay are not secure in terms of confidentiality, integrity and availability. Confidentiality is a negligible issue since all messages have an observable effect on receiving ECUs and do not share secret information. Integrity, however, especially authenticity, is a major issue in automotive networks. Using a compromised ECU, we could perform replay attacks, changing the state of a simulated vehicle in an undesired way. It is also possible to perform DoS attacks by inserting colliding messages.

The CaCAN security extension showed a similar integrity issue. Since it operates by sending an 8-bit hash, it is very susceptible to an exhaustive search attack. The 8 bits reserved for an authentication code are not enough to provide authenticity in an automotive network. CaCAN is therefore easily breakable and should not be used in practice.

The tested hashing-based authentication scheme did not have an issue with integrity and provided sufficient security in

that regard. However, the number of bits reserved for authentication reduced the maximum payload size drastically. If this is sufficient must be decided by manufacturers, possibly on a case-by-case base. We found other proposals that theoretically provide better integrity of messages but similarly impact the bandwidth. The version of the authentication scheme ported to FlexRay showed similar integrity protection, with bandwidth being less of an issue because of the larger maximum size of FlexRay messages. We could not find any publications regarding proposed security extensions for FlexRay.

We observed that the hash calculations used by the security extensions introduce a drastic delay to the operation of sending and processing CAN messages. Making use of the proposed security scheme improves the integrity of the messages sent on the network, but makes the usage of the network less performant. It further compromises the availability of ECUs since they are now susceptible to DoS attacks, whereas before only the messages themselves could be drowned out. Whether this is usable in a real vehicle depends on the processing power of the individual ECUs.

Since it is up to the manufacturers to implement security in their vehicles, they would have to consider where and how it should be implemented. Different components, specifically different domains in a car might have different requirements regarding security. If integrity is more important than availability, we propose to them to implement such a hashing-based authentication scheme. If the performance of an ECU is more important, not making use of such authentication is perhaps a valid alternative.

10 Discussion

The found results of the experiments may not be translatable to a real-world environment because we did not use a real ECU, but simulated those with either an Arduino or a software simulation. Therefore, the results of implementing the extensions in a vehicle could differ from what was found in this research. Relating to this is also the time it takes to create the (brake) message and the processing. This was done using Arduino microcontrollers and not with actual ECU hardware. This could affect the time it takes to create and process these messages leading to a different added brake distance. However, if similar behaviour is observed using real ECUs, it would be possible to extrapolate our results to real vehicles.

We did not have access to FlexRay hardware to set up a hardware experiment. Therefore, it was not possible to perform FlexRay experiments in a more realistic environment.

Only two extensions to CAN have been reviewed in this project. There are more extensions available which perform similarly in slightly different ways. They may perform better or worse than CaCAN and the hashing-based solution.

The performed experiments were executed in a controlled environment with self-programmed ECUs. In a real-world environment, the IDs of CAN messages have to be found first to perform a replay attack on the CAN bus. These are known for several vehicles, but a lot of manufacturers keep this information proprietary.

We have used SHA256 as a hashing algorithm in the experiments. We calculated the full hash of the data and then concatenated the last bits (depending on how many were required) and used that as the hash of the message. This is not as secure as using the full hash or using other hashing algorithms that result in a hash length of the required bits.

11 Future Work

This research has shown that the different automotive networks tested are not secure on their own. The extensions tested either provide too little security or compromise the bandwidth of the bus in question. To further specify, similar experiments should be conducted on other networks and extensions. Especially automotive Ethernet, which is emerging in the industry, should be considered for a security evaluation. Given that CAN is still the most widespread standard, more extensions for it should be tested in the same way.

The ported hashing-based authentication scheme for FlexRay proposed in this paper should be improved upon. More security proposals for FlexRay are required to keep the protocol secure and future-proof. Once these proposals are defined, a security evaluation should be conducted on them, to consider whether they provide sufficient security and if they still leave automotive vehicles to be performant enough for public operation.

The delay measurements conducted as part of this research should be repeated using actual ECUs that are being used in the industry, within current vehicle models. Only such measurements can give a definitive answer whether the introduced delay is to be considered an issue in securing vehicle networks. These experiments should also consider dedicated components built-in for security, such as controllers calculating hashes only. Should these experiments prove that ECUs also show performance problems, it should be researched how optimization can be done to the security extensions to make the issue less drastic.

Given that the security of vehicle networks might impact the performance, the security of the car also becomes an ethical question. If, for instance, the integrity is improved, the brakes of the car might engage a few milliseconds later. The Massachusetts Institute of Technology proposed the question of autonomous cars deciding about life and death [1], and securing a vehicle network presents a similar issue: Is it more ethical to secure the driver by securing the car or is it better to engage the brakes quicker, potentially saving pedestrians or even the driver in a different situation? However, as for this research, answering or discussing this question was out of scope.

Since the decision between securing a vehicle and making critical functions well performant can be considered of ethical nature, we also propose for this issue to be discussed publicly. If a consensus is reached on a case-by-case base, that consensus should be evaluated for regulations to be adhered to by automotive manufacturers. Perhaps this issue should even take the form of government regulations, of which many already exist. Currently, security of the automotive networks is not obligatory by law. However, the potential impact on the general public, if a car model is hacked, is huge. This paper does not discuss the impact of requiring security in automotive networks in terms of laws and regulations.

References

[1] Awad, E. et al. *The Moral Machine experiment*. Cambridge, Massachusetts, United States of America: Springer Nature Limited., 2018. DOI: 10.1038/s41586-018-0637-6.

[2] Bochem, A. et al. *FPGA Design for Monitoring CANbus Traffic in a Prosthetic Limb Sensor Network*. Fredericton, Canada: University of New Brunswsick, 2011. DOI: 10.1109/RSP.2011.5929972.

[3] Bruton, J. A. *Securing CAN Bus Communication: An Analysis of Cryptographic Approaches*. Galway, Ireland: National University of Ireland, 2014. DOI: 10.13140/RG.2.2.22154.93127.

[4] Buntsma, A. and Wilczek, S. *ArduinoCAN: C scripts for Arduino to run CAN simulations*.
Available at: https://github.com/sebastianwilczek/CANoe-Configurations [Accessed 28 Jan. 2020]. 2020.

[5] CAN in Automation. *CAN in Automation (CiA)*.
Available at: https://www.can-cia.org/can-knowledge/can/can-history/ [Accessed 07 Jan. 2020]. 2020.

[6] Charlie Miller, C. V. *Remote Exploitation of an Unaltered Passenger Vehicle*.
Available at: http://illmatics.com/Remote%20Car%20Hacking.pdf [Accessed 12 Jan. 2020]. 2015.

[7] Checkoway, S. et al. "Comprehensive Experimental Analyses of Automotive Attack Surfaces". In: *2010 IEEE Symposium on Security and Privacy*.
Available at: http://www.autosec.org/pubs/cars-oakland2010.pdf [Accessed 16 Jan. 2020]. 2011.

[8] Consumer Watchdog. *Why Connected Cars Can Be Killing Machines and How to Turn Them Off*.
Available at: https://www.consumerwatchdog.org/sites/default/files/2019-07/KILL%20SWITCH%20%207-29-19.pdf [Accessed 15 Jan. 2020]. 2019.

[9] Cros, O. and Chênevert, G. "Hashing-based authentication for CAN bus and application to Denial-of-Service protection". In: (2019).
Available at: https://www.researchgate.net/publication/336749708_Hashing-based_authentication_for_CAN_bus_and_application_to_Denial-of-Service_protection [Accessed 30 Jan. 2020].

[10] Hartzell, S. and Stubel, C. *Automobile CAN Bus Network Security and Vulnerabilities*.
Available at: https://canvas.uw.edu/files/47669787/download [Accessed 30 Jan. 2020]. Seattle, Washington, United States of America: University of Washington, 2017.

[11] Herrewege, A. V., Singelee, D., and Verbauwhede, I. *CANAuth - A Simple, Backward Compatible Broadcast Authentication Protocol for CAN bus*.
Available at: https://pdfs.semanticscholar.org/007e/e2559d4a2a8c661f4f5182899f03736682a7.pdf [Accessed 07 Jan. 2020]. 2011.

[12] *Road vehicles — FlexRay communications system — Part 1: General information and use case definition*. Standard. International Organization for Standardization, 2013.

[13] Karamba Security. *SafeCAN®: A New Model for In-Vehicle Network Security*.
Available at: https://karambasecurity.com/products/safecan [Accessed 07 Jan. 2020]. 2019.

[14] Koscher, K. et al. "Experimental Security Analysis of a Modern Automobile". In: *2010 IEEE Symposium on Security and Privacy*. 2010, pp. 447–462. DOI: 10.1109/SP.2010.34.

[15] Kurachi, R. et al. *CaCAN - Centralized Authentication System in CAN (Controller Area Network)*.
Available at: https://www.researchgate.net/publication/320083914_CaCAN_-_Centralized_Authentication_System_in_CAN [Accessed 30 Jan. 2020]. Nagoya: Nagoya University, 2014.

[16] Mercury Systems. *CANGuard^{TM} - Defending automotive vehicle data networks against malicious vehicle intrusion*.
Available at: https://www.mrcy.com/resourcehub/document/canguard-defending-automotive-vehicle-data-networks-against-malicious-vehicle-intrusion [Accessed 07 Jan. 2020]. 2017.

[17] Mueller, A. and Lothspeich, T. *Plug-and-Secure Communication for CAN*.
Available at: https://pdfs.semanticscholar.org/2bb1/bc642612048465a42b1c3286451a3dfd5014.pdf [Accessed 07 Jan. 2020]. Gerlingen, Germany, 2015.

[18] Murvay, P.-S. and Groza, B. "Practical Security Exploits of the FlexRay In-Vehicle Communication Protocol". In: *Risks and Security of Internet and Systems*. Ed. by Zemmari, A. et al. Cham, Germany: Springer International Publishing, 2019, pp. 172–187. ISBN: 978-3-030-12143-3.

[19] National Instruments. *FlexRay Automotive Communication Bus Overview.*
Available at: https://www.ni.com/nl-nl/innovations/white-papers/06/flexray-automotive-communication-bus-overview.html [Accessed 30 Jan. 2020]. Austin, Texas, United States of America, 2019.

[20] Navet, N. et al. "Trends in Automotive Communication Systems". eng. In: *Proceedings of the IEEE* 93.6 (2006), pp. 1204, 1223. ISSN: 0018-9219.

[21] Nilsson, D. K. et al. "A First Simulation of Attacks in the Automotive Network Communications Protocol FlexRay". In: *Proceedings of the International Workshop on Computational Intelligence in Security for Information Systems CISIS'08.* Ed. by Corchado, E. et al. Berlin, Germany: Springer Berlin Heidelberg, 2009, pp. 84–91. ISBN: 978-3-540-88181-0.

[22] Nolte, T., Hansson, H., and Bello, L. L. *Automotive Communications - Past, Current and Future.* Västerås, Sweden: Mälardalen University, 2005. DOI: 10.1109/ETFA.2005.1612631.

[23] Radu, A.-I. and Garcia, F. D. *LeiA: A Lightweight Authentication Protocol for CAN.*
Available at: https://www.cs.bham.ac.uk/~garciaf/publications/leia.pdf [Accessed 30 Jan. 2020]. Birmingham, United Kingdom: University of Birmingham, 2016.

[24] Scalas, M. and Giacinto, G. "Automotive Cybersecurity: Foundations for Next-Generation Vehicles". In: *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS).* 2019, pp. 1–6. DOI: 10.1109/ICTCS.2019.8923077.

[25] Siddiqui, A. S., Plusquellic, Y. G. J., and Saqib, F. *Secure Communication over CANBus.* Charlotte, United States of America: University of North Carolina, 2017. DOI: 10.1109/MWSCAS.2017.8053160.

[26] Strobel, O. *Communication in Transportation Systems.* Esslingen, Germany: Esslingen University of Applied Sciences, 2013. ISBN: 9781466629776.

[27] Upstream Security Ltd. *Global Automotive Cybersecurity Report 2020.*
Available at: https://www.upstream.auto/upstream-security-global-automotive-cybersecurity-report-2020/ [Accessed 15 Jan. 2020]. 2020.

[28] Vector Informatik GmbH. *CANoe — ECU & Network Testing on Highest Level — Vector.*
Available at: https://www.vector.com/int/en/products/products-a-z/software/canoe/ [Accessed 08 Jan. 2020]. 2020.

[29] Wilczek, S. *CANoe-Configurations.*
Available at: https://github.com/sebastianwilczek/CANoe-Configurations [Accessed 28 Jan. 2020]. 2020.

Appendices

A CANoe Bus Simulation

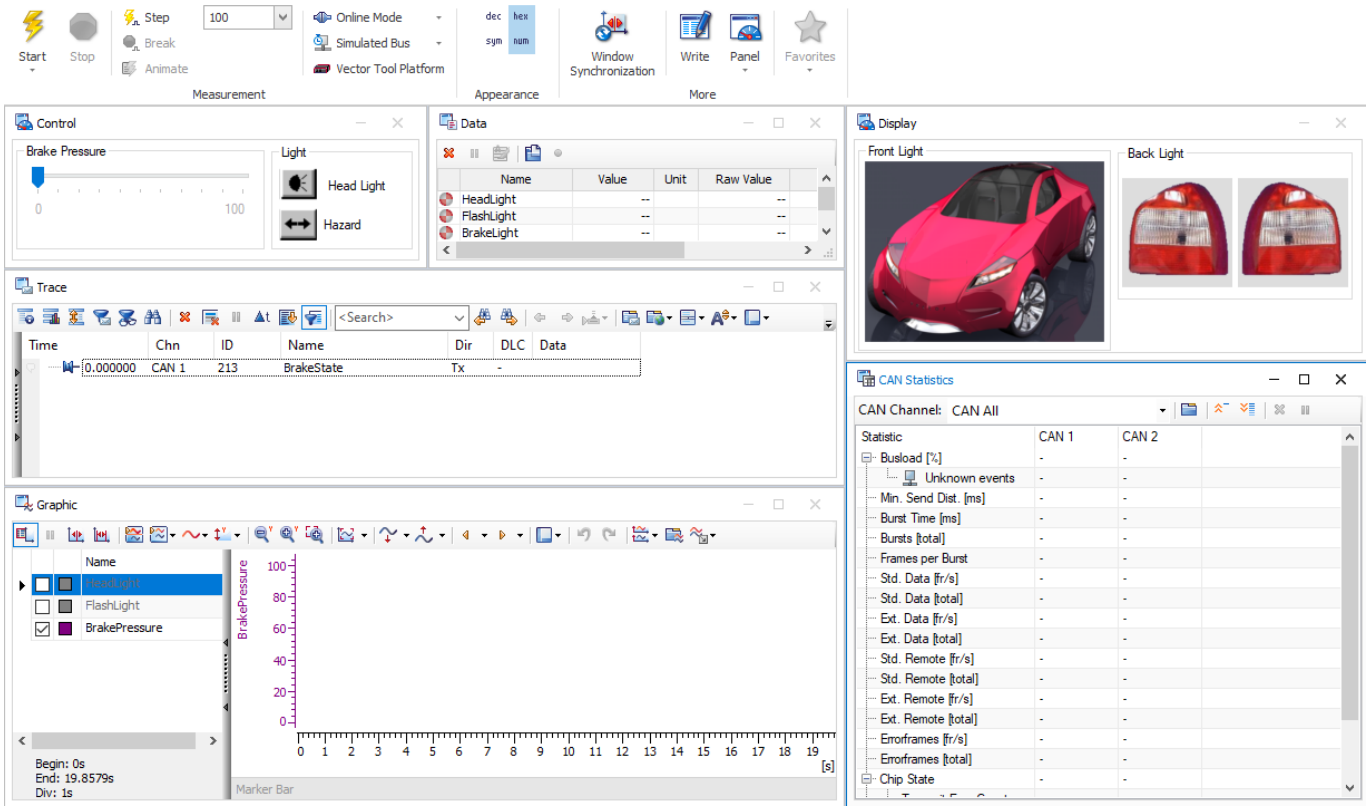

Figure 3: CANoe Bus Simulation Screenshot