5-17-2017

Adding Security to Control Area Network of Vehicles by Using SHA-3

Aidin Ameri
aa9683@rit.edu

ADDING SECURITY TO CONTROL AREA NETWORK OF VEHICLES BY USING SHA-3

By:

Aidin Ameri

Committee Members:

Professor Sumita Mishra

Hrishikesh Acharya

In partial fulfillment of the requirements for the degree of

Master of Science in Computing Security

Rochester Institute of Technology

B. Thomas Golisano College of Computing & Information Sciences

Department of Computing Security

May 17, 2017

Acknowledgment

I would like to express my immense appreciation to Professor Sumita Mishra, my research chair, for the valuable and constructive feedback and advice throughout the planning and development of the research process. I would also like to express my deep gratitude to Hrishikesh Acharya, my advisor who encouraged and challenged me during this project. Without their help and support, it would be almost impossible to work on this research which I believe, if it been used in vehicles can save lives.Finally, I wish to thank my wife, Touba Siavashpour who sacrificed many hours to support me to work on this research, my sister, Aida who encouraged me to pursue my academic interest, my father Amirteymour who taught me to be always curious, and my mother Azarmidokht who always believes in me.

Contents

8

List of Figures

Abstract

The lack of security in the vehicles on the road is real and should be taken seriously. Since the lifespan of vehicles has average of eleven years, this means if we start to implement new changes to vehicles today, it would takes eleven years to make sure most vehicles on the road support our implementation. This is important as the number of lines of codes in vehicles are ever increasing and becoming more autonomous with the ability for vehicles to drive themselves. But there is no security implemented in their low level systems such as the Control Area Network which is being used to transfer real time critical information and commands such as engine speed and the brake control. This project attempts to solve the lack of security by using SHA3 hashing algorithm based on the Keccak algorithm. The reason Keccak was chosen to be the SHA3 algorithm because it is hardware friendly and fast. Vehicle's manufacturers do not share information about the electrical parts used in the vehicles and their specification, this project presents the lowest hardware specification required to use SHA3 on the Control Area Network which is a process with a clock frequency of approximately 400 *MHz*. It is important to have a real-time communication network for the Control Area Network which also known as CAN. SHA3 is used to create a hash of the CAN message along with a node specific key and an IV to provide authentication and semantic security respectively; The digest will be transferred along with the message. To use the mechanism proposed in this project, all CAN nodes communicate with each other required to support CAN FD and also have the minimum hardware specification.

Keywords–Vehicles, Computing Security, Control Area Network, CAN FD, SHA3, Keccak.

1 Introduction

It is obvious that the number of physical systems around us that are getting connected to the Internet. As we see more devices internet-assessable, the number of potential vulnerabilities also grows exponentially. Many systems, such as vehicles, when designed were not intended to be connected to other systems, but now the first priority is to connect those devices to Internet to make them accessible. Vehicles are one of many systems that started as strictly mechanical, but have transformed into what are considered today as Cyber-physical devices.[1] In any vehicle, there are many connected small electronic nodes that are able to perform computation and the information from these nodes should be transferred to other parts of the vehicle so that they could work. The Control Area Network (CAN) communication protocol is one of the five protocols used in the on-board diagnostics version two (OBD-II) standard, which is mandatory in the United states for all cars and trucks since 1996[7]. The CAN protocol was designed and proposed by Bosch in 1983. The second generation of the protocol, CAN 2.0 was released in 1991. The current version was developed in 2012 and is called CAN Flexible Data-Rate of for short CAN FD. The CAN 2.0 is the most common used in vehicles around the globe. This protocol does not have any security controls to limit the access of a malicious actor to the internal car systems. There has been a lot of research and projects conducted that the vulnerabilities of CAN and also many effort to secure this system, but none of them so far used the newly developed hashing algorithm Keccak sponge function which has recently been standardized by NIST[8]. The goal of this project is to analyze SHA3 and create a report on the overhead of using this hash algorithm which is used for authentication and integrity in the CAN. Also, the minimum hardware requirement for the nodes in a vehicle to be able to use this approach so that the communication be real-time is presented. This is important because the nodes in any vehicle have limited resources such as computational power and memory. The reason for selecting SHA-3 is that this hashing algorithm can computes different length message digests and it has less requirements then any another hashing algorithm in order to add integrity. Up to the advent of the CPS, the average end user did not understand security implications and viewed it as too inconvenient to deploy secure systems. However, now because people's life can be more in danger because they increasing the attack surface by using more automated and connected devices, they should start to take security seriously. This project is important because securing vehicles would save lives. Not only the people inside the vehicle, but also those around it may depend on the security of it. No vehicle is secure unless the communication between critical nodes is secure. By the end of this project, the minimum requirements to secure CAN via SHA-3 will be analyzed. This project is outlined in the following order: Section II is the literature review, followed by Section III, The Securing mechanism, Section IV testing and experiments, Section V data analysis, and Section VI the conclusion of this project.

[1] NIST defines it as: Cyber-Physical Systems (CPS) are smart systems that include engineered interacting networks of physical and computational components.

2 Literature Review

2.1 Introduction

In the digital age, no system is not guaranteed to be secure. Vehicles started as a purely mechanical device, but through the years they have changed. The vehicle manufacturers try to make their product safer every day, however, the number of lines of code has also significantly increased leading to more potential vulnerabilities. Software made cars more efficient, but unintentionally they also introduced vulnerabilities in this complex system. During this literature review first the definition and basic concerns around computer security are reviewed. Next, the raised security vulnerabilities in vehicles are investigated. Most of these security concerns are around the vehicle's Control Area Network, which are going to be investigated in depth in this review. Then, the previous solutions along with their pros and cons are investigated. Finally, the proposed solutions and its advantages and disadvantages will be discussed.

2.2 Computing Security

Today we face many issues related to the Cyber security. This phenomena is due to the fact that during the design phase, the goal was to make sure the system works and that was satisfying to everyone from the designer and developer up to the customer. The advent of the Internet has impacted nearly all citizens and influenced how most industries conduct business; These device have operating systems and are networkable, even traditional household appliances such as refrigerator and ovens; These devices are called Internet of Things (IoT). Based on an article by Gartner number of IoT devices will reach 21 Billion by 2020[9]. On one side, these systems ,especially the legacy ones, are not designed with security in mind while the attack surface is increasing due to more devices being interconnected with the Internet. The very first vehicles started a purely mechanical machines but now they are complex systems. In 1911, the first electronic starter was patented and installed on an Arnold[10] also, many electrical devices were added to the vehicles to increase fuel efficiency and provide safety to passengers; a few of the features include Electronic Fuel Injection, airbags, and diagnostics. In 2010, vehicles become Cyber-physical systems which means there are systems that are developed and depend on computational algorithms and physical components. Today, the software used in a vehicle can cost up to 50 % of the total cost of the whole vehicle. Based on Koscher and et al. article[11] most vehicles today contains over 100 MB of binary code which is used in 50 to 70 different independent micro-chips (nodes) overseeing everything from the critical function of vehicle to non-critical functions. These nodes and systems have their own specific names such as Electronic Control Unit (ECU), Controller Area Network (CAN), Radio Data System (RDS), Traffic Message Channel (TMC) just to name a few. Some of these new systems are also hybrid which means they are consist of more than of type of nodes. The primary focus of this project is the security of the CAN communication protocol, which is used to transfer some of the most critical

information and commands in vehicles such as engine temperature and brake command. There are also emerging communications systems such as Vehicle-to-Vehicle (V2V) and Vehicle-to-infrastructure (V2X) that enables the intra-vehicle communication which are not in the scope of this project.

2.3 Control Area Network and related security concerns

2.3.1 Control Area Network

Control area network is a protocol developed by BOSCH in 1983[12]. This protocol traditionally uses an 8 byte payload as shown in the figure 1 and is used in various industrial automation systems.

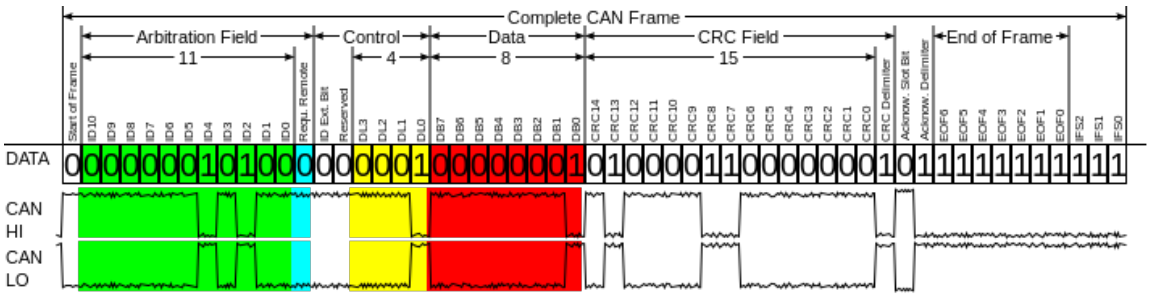


Figure 1: CAN 2.0 Frame[1]

This protocol is a broadcast based communication system, which connects the embedded electronic control units (ECUs) in the vehicles to different nodes. Each node is responsible to generate information or execute a set of controls based on the signal detected by its sensor or based on the information received from other nodes[13]. Bousch is still working on the CAN protocol to make it more suitable since they released CAN FD 1.0 back in 2012. This protocol has a flexible data-rate with the maximum payload size of 64 bytes. The CAN FD frame is shown in figure 2. This information and commands include but not limited to, engine speed, temperature of engine, inside and outside temperature, brake pedal, actuator position, tire pressure, and airbag status are all transferred using CAN. There are many decisions made based on the information gathered by the sensors and transferred by CAN, for example, car engine will change its idle speed based on the current engine temperature.If in an accident the airbag system engages, all of the doors automatically would be unlocked to minimize the possible danger to the passengers. Back in mid 1990s car manufacturers started to use more advanced nodes; UNIX-like operating systems used peripherals such as Global Positioning Systems. Using all these different communication systems described placed in any vehicle increases the number of vulnerabilities.
Each network in a vehicle is categorized based on their functionality and their speed by the Society of Automobile Engineers (SAE)[14].

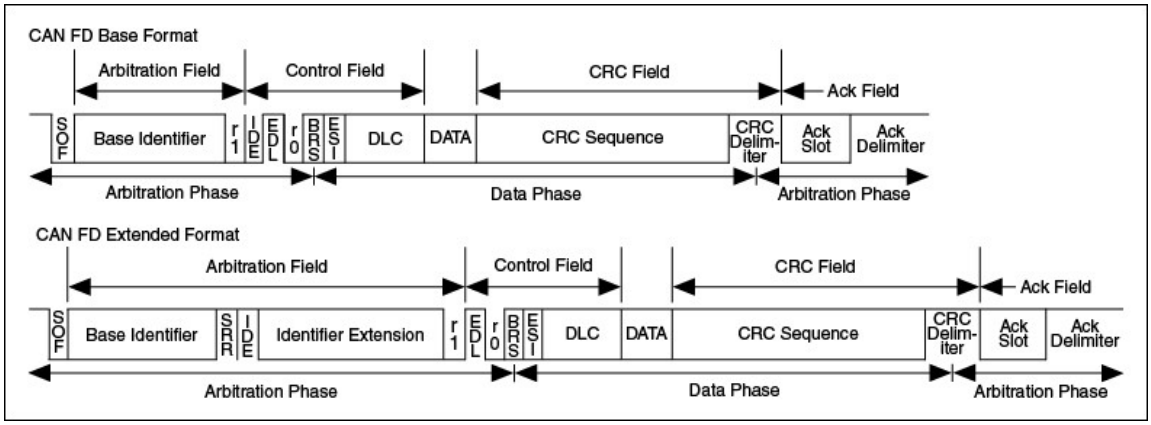


Figure 2: CAN FD Frame[2]

- **Class A** - Low speed networks, Speed less than 10 *Kbps* which are for non-critical body control functions.
- **Class B** - Medium speed networks, Speed between 10 *Kbps* and 125 *Kbps*, e.g. low-speed CAN.
- **Class C** - High speed networks, Speed between 125 *Kbps* and 1 *Mbps*, which is used for critical real time communication, e.g. high-speed CAN.
- **Class D** - Ultra high speed networks, speed above 1 *Mbps* which is used for multimedia systems.

2.3.2 Security Concerns

CAN security was purely based on the security of data transfer that is done by using error detection and self-monitoring built in the protocol[15] which means the only security objective was availability of information without confidentiality nor integrity. Some manufactures started to use AES-128 encryption on the CAN bus, but research by Hamilton Et al.[16] shows that this type of security is not immune to the use of Correlation Power Analysis (CPA) which analyses the correlation between power consumption and the Hamming weight of the data to find the plaintext message without knowing the key to decrypt the message. In 2015, Andy Greenberg a journalist from the Wired magazine willingly sat inside a Jeep Grand Cherokee that had been accessed remotely by two cyber security researchers, Charlie Miller and Chris Valasek. He describes the incident as follows, "Though I hadn't touched the dashboard, the vents in the Jeep Cherokee started blasting cold air at the maximum setting, chilling the sweat on my back through the in-seat climate control system. Next to the radio switched to the local hip hop station and began blaring Skee-lo at full volume. I spun the control knob left and hit the power button, to no avail. Then the windshield wipers turned on, and wiper fluid blurred the glass. As I tried to cope with all this, a

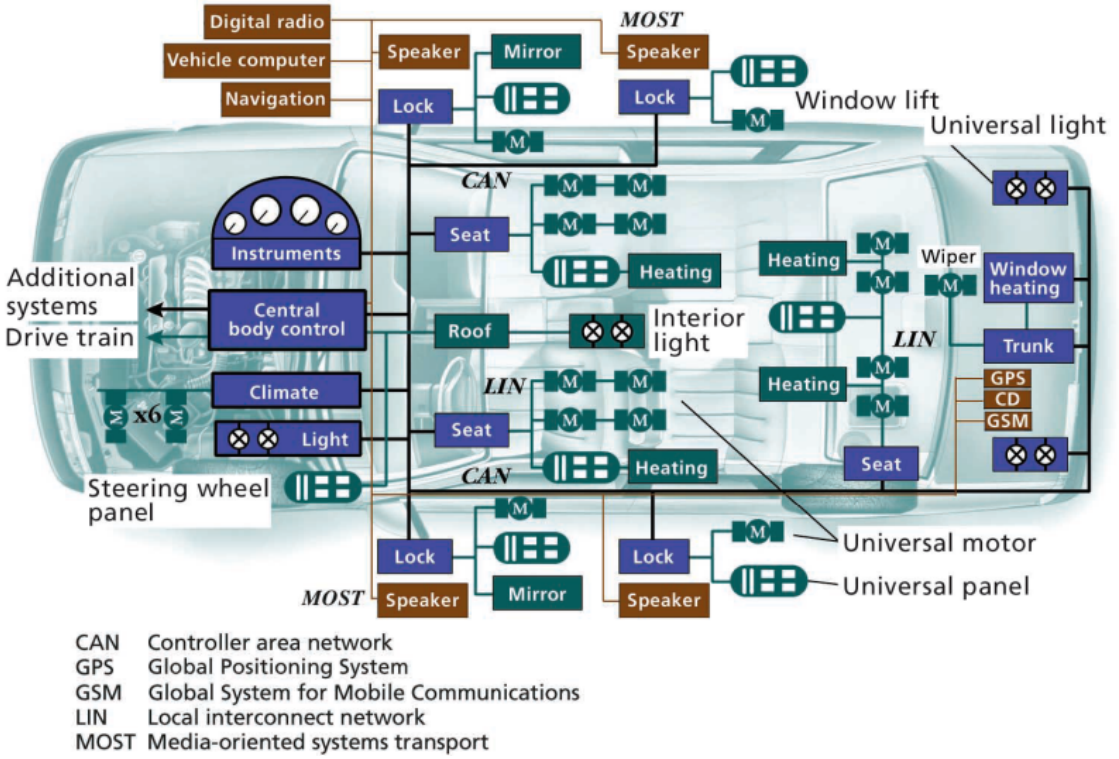


Figure 3: In Vehicle Communication[3]

picture of the two hackers performing these stunts appeared on the car's digital display: Charlie Miller and Chris Valasek, wearing their trademark track suits. A nice touch, I thought."[17] As you can see the security of vehicles has direct connection to the life of people in or around the vehicle. Also, it is important to mention that Jeep Grand Cherokee is not the only vehicle that have been compromised, many other vehicles such as the Ford Escape and Toyota Prius also have been compromised. In addition to these, a series of compromises has been done by Hoppe, Klitz, and Dittman[18]. The electric window lift and warning light compromised by performing Denial of Service (DoS) attack and sending false data about air bag module presented to the other nodes. Additionally, the messages passing through the gateway ECU (those nodes that connect different communication protocols in the vehicle) were captured.

2.4 Challenges and Goal

Many features that made CAN protocol, successful at the beginning also made securing it challenging[19][20][21]:

- Broadcast communication

All messages in the CAN are broadcast to all other nodes that are connected to the CAN. All nodes can read these messages because they are unencrypted; they are only checked against any error such as bit stuffing, redundancy check and packet format, etc. The nodes after reading the message decides if they should process the message further or drop it.

- Source address and Authentication

  There is no source address in the messages being sent, so the receiver cannot identify the sender. The only content that can be used to identify the sender is the type of the message. By design, the important part of the system was the information in the message, not the source or destination address. However, the challenge and security risk arises when a message is being sent by a malicious node (which can be an easy made microcontrollers with a can transceiver which cost less than $50). The message range from turning the lights on or off which can be disastrous if the vehicle has a high speed in the middle of a dark highway. Alternatively, an attacker could change the engine temperature, or sudden unwanted brake or can endanger the passenger by providing incorrect information to other system sensors.

- Bus-off mode

  CAN protocol has a fault isolation protocol, called buss-off mode which can turn off any nodes selectively if that node generates error messages. This action can be done by sending fake error messages by the malicious node into the CAN.

- Gateways and Super-gateways

  There are several control area networks and also other networks such as Local Interconnect Network(LIN)[22] LIN and FlexRay[23] running in vehicles. These networks are connected through what is known as gateways and super-gateways. The former is when connected networks use the same protocol and the latter is when different protocols are used to connect to each other. To help avoid a problem there is a filtering mechanism in each gateway which uses a routing table which has a list of known routes for different type of messages. This routing table can be used against the system by a malicious node by leveraging this table to craft a message to be routed from one network to another. The design of gateways have also made a remote attack possible since in the modern vehicles there are networks that are connected from one side to the outside networks such as Internet and from other side they are connected to the CAN! There are products in the market that claim they make any vehicle smart

and secure, one of them is Hum by Verizon[2] which connects directly to the vehicle's ODBII connection which is directly connected to the CAN. An adversary can bypass the routing table and from anywhere in the world get access to the vehicle's control area networks.

- Real-Time communication

  It is essential that the information and commands reach to the critical node in time because of the nature and criticality of the communication. For example, the communication between the brake pedal and the brake discs should be seamless. Encryption in general is not a good approach for any real time communication because it will add latency due to the fact that it needs more computation, both for encrypting and decrypting the data.

- Non-Repudiation

  There is no mechanism in the CAN bus to identify the sender and receiver so there are no proof of the origin. One such example is an adversary can easily capture any commands such as an acceleration command and use it when the vehicle is located near a cliff.

There are some general security goals that should be achieved in order to have a vehicle more immune to cyber attacks such as integrity, availability, authentications and freshness identified by other researchers like Wolf[24] and D. Nilsson and Larson [25] which most of them need to redesign the whole protocol. So the goal of this project is providing Integrity, authentication and data freshness to the CAN.

2.4.1 Non-Cryptographic methods to secure CAN

There have been research about securing the CAN in the vehicles by using non-cryptographic methods such as Intrusion Detection System (IDS), firewall, which are discussed in depth in the next section.

- Firewall

  Back in 2010, Arilou Technologies[26] proposed a standalone hardware that functioned as a firewall between nodes and the CAN. The problem with this approach is the new hardware will add more concerns such as more energy usage, there are more system to update and maintain and the fact that what happens if this node compromise. If the system is backward compatible a malicious attacker can send a spoofed error

[2] www.hum.com

message, so that the node disables the firewall. On the other hand, if this firewall system is not backward compatible, a malicious actor can cause danger by performing a denial of service attack on the firewalls behind critical nodes which may delays the transmission of critical messages; One example is that a driver hit the brake pedal to avoid accident but that command may reach to the brake pads late which causes the car to crash. In recent years, there have been solutions that claim they will secure any vehicle, VisualThreat is one of many[27] that offer a framework which includes a firewall, a general policy for the whole vehicle, updating the system over the air, and artificial intelligence of the vehicle. The problem with this type of solution is that they are provided by third parties and may not fully supported by the vehicle manufacturers. The vehicle manufacturers and these companies do not communicate well and there is no documentation for vehicles. This may even endanger the life of passengers as these third parties may not know the vehicle well.

- Intrusion Detection System

  Another non-cryptographic method to secure the CAN is to have an intrusion detection system which monitors the events and normal behavior of the network and detect any abnormality or possible malicious activity in the system[28]. An IDS needs to scan all messages sent on the CAN and at best, the IDS can detects malicious activity, there are many false positives with any intrusion detection system. An IDS is not a good choice of securing mechanism because vehicle needs real time communication, but IDS adds undesired and unacceptable latency. Also, IDS alone is not effective, we need to have another system to use the data analyzed by the IDS to protect the system. It is life threatening if an IDS identifies the legit brake signal as malicious activity and so that the vehicle does not stop where it should. So, we need a system to be near perfect, no false positive is accepted in cyber-physical devices that can endanger the life of humans and animals.

2.4.2 Cryptographic methods to secure CAN

There have been many approaches that use cryptographic methods to add confidentiality, integrity, authentication, and semantic security to the CAN communication protocol. These cryptographic methods are including but not limited to, usage of public and private keys in the ECU[29] which provides confidentiality and also non-repudiation but it is expensive in terms of computation power and latency. Another research project introduces using authentication for external nodes outside of the gateway and the internal nodes[30]. But if an adversary identifies itself as an internal node and try to communicate with other nodes, in that case the purposed authentication method is ineffective. As it is explained in a research by D. K. Nilsson et al.[31] it is crucial for the CAN network to have authentication, integrity and also semantic security more than confidentiality in any vehicular system.

There have been approaches to add authentication and semantic security to the CAN by using message authentication codes (MACs) [32] which also uses AES to encrypt the session key but again, using AES is vulnerable to the hamming weight and hamming distance attacks which are attacks that can be used against any cipher text encrypted with AES algorithm. The result of these attacks are finding the plain text of encrypted message with analyzing the consumption power of nodes that encrypt or decrypt the message. Also, there other cryptographic approaches to secure the CAN[33]. On the other hand, there are no research on using SHA3 in the CAN.

2.5 Literature review conclusion

There have been research involving how to secure the embedded systems, but most of them did not consider the limitations posed by these embedded systems, for example, these systems should consume low energy; their computation power is limited and they have limited cache, memory, and storage size. Especially many cryptographic solutions need more cache and RAM to store the key so that the node can encrypt and decrypt the message in shortest time. In the real world, the non-cryptographic methods could be more dangerous to the driver and people outside of the vehicle than having no security in the CAN at all.

3 Proposed Approach

3.1 Why SHA3

There are two important security related characteristics that are not in the Control Area Network of vehicles which can be used against it as we discussed in the previous sections. These are lack of semantic security, and lack of source identifier. Lack of semantic security enables an adversary to capture and replay any packets and get the same effect just like a legit sender send the captured packet. In addition, lack of source enables an adversary to use any nodes on the network or add its own node and use it to send any desired message to the network. The proposed solution in this section tries to address these problems by using SHA3.
SHA3 provide authentication and semantic security if it is used along with unique keys and counter for each node. Authentication is when we can identify the sender, so if each node has a unique key by computing the digest of message + key, the nodes with the key can create compute and recompute the same digest. Because we assume that every time that the vehicle starts, the ECU will generate new sets of key, it is almost computationally impossible that an adversary can test all the possible digest messages in time. In the section "Security strength of SHA-3" there are more information regarding the security bits of this hashing algorithm. Same outcome is possible when a unique counter is used by each node. The new counter will change the digest as even changing one bit of the message will change all of the digest computed from that message. Thus, if adversary capture a packet it can not be used again because the receiver will compute different digest.

3.2 Keccak Algorithm - SHA3

Keccak (pronounced like "ketchak") algorithm is a family of sponge functions as shown in the Figure 4 that has been standardized in the form of SHA3-224 to SHA3-512 hash functions in FIPS 202[3]. A hash function is a one-way function that has an input called message of variable length sizes and an output called a digest or hash value which has a fixed length based on the hash algorithm.

[3]http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf

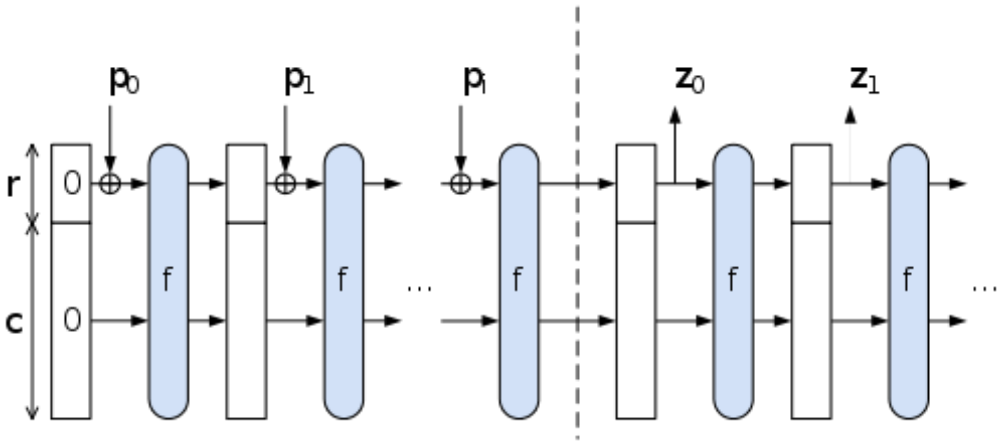


Figure 4: Sponge function[4]

The four different SHA-3 hash functions are named SHA3-224, SHA3-256, SHA3-384, and SHA3-512; in each case the suffix after the dash indicates the fixed length of the digest. This algorithm can have seven permutations named KECCAK-f [b], with b=25, 50, 100, 200, 400, 800 or 1600. For the SHA3 the b=1600 is purposed but the smaller the permutation, can be used in lightweight and constrained environments. The Keccak-1600 function has 24 rounds of operation as shown in the Figure 5, where each round has five sequential steps. These 1600 bits are organized in a 3-D array, as shown in the Figure 6. Each bit is addressed with three coordinates, written as $S(x, y, z), x, y \in \{0, 1, 2, 3, 4\}, z \in \{0, 1, 2, 3, ..., 63\}$..2-D entities, plane, sheet and slice, and 1-D entities, lane, column and row, are also defined in Keccak.

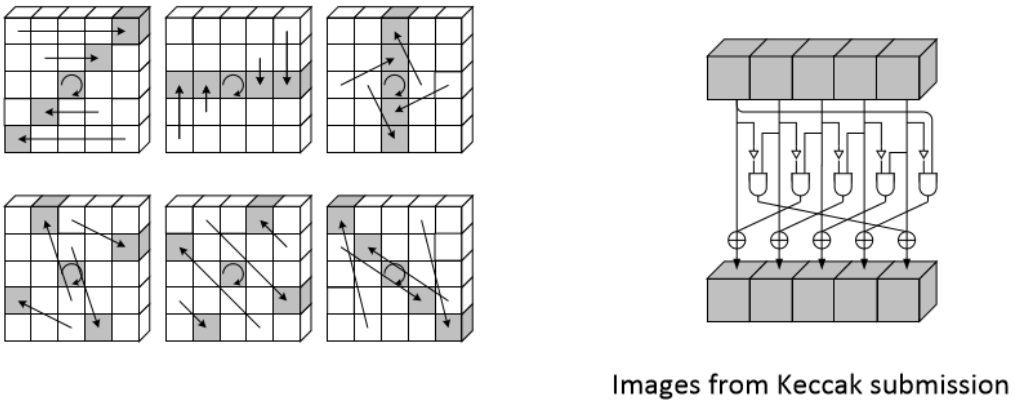


Figure 5: Rounds[5]

For solution in this paper MAC-Keccak has been purposed which is recommended by the Keccak designers[34] [35]:

$$MAC(Message, Key) = Hash(Key||Message).$$

As you can see from the MAC function, Keccak does not require the regular key padding that is required in other HMAC[4] functions. Using padding makes HMAC immune against length extension attack.

$$HMAC(Message, Key) = Hash((Key' \oplus opad||Hash(Key' \oplus ipad)||Message)).$$

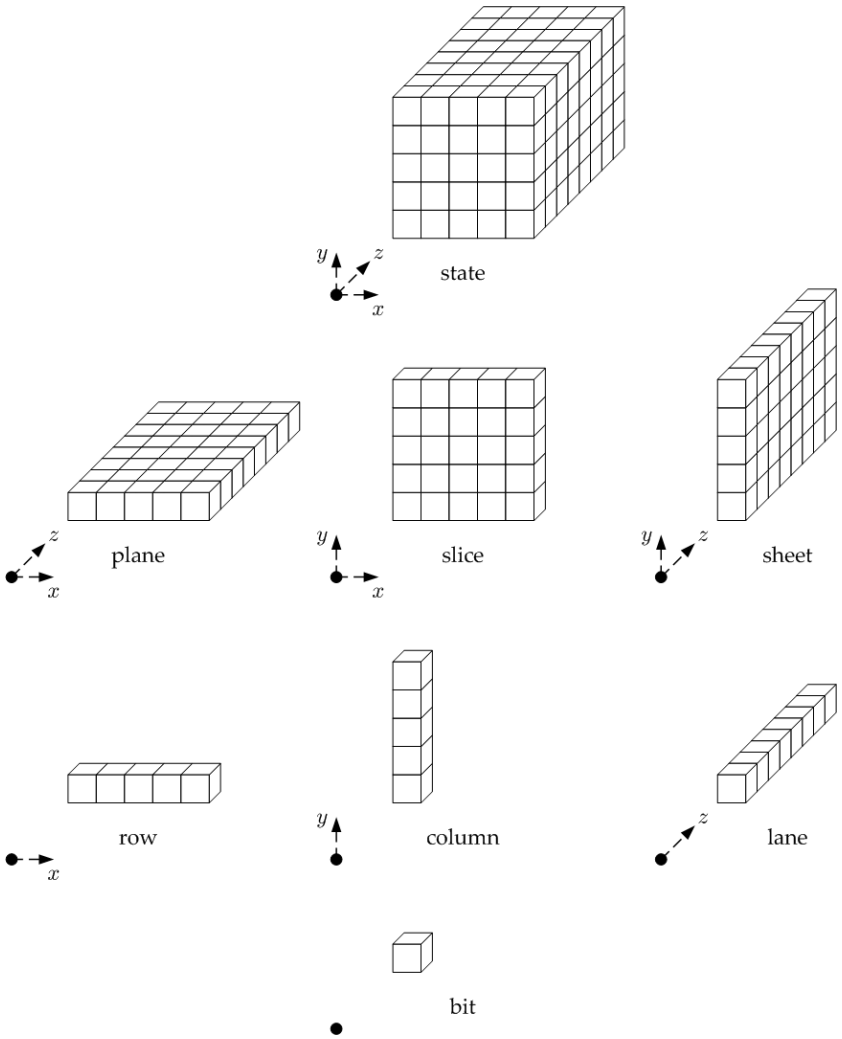


Figure 6: Entities defined in Keccak[5]

[4]https://tools.ietf.org/html/rfc2104

3.3 Assumptions

There are two main prerequisites in our proposed solution that are not in the scope of this paper. To provide the source authenticity to the nodes, each node should be identified with a unique key, 112 bits long, which can be computed by the vehicle's main ECU and distributed as a two-dimensional array to all nodes when a vehicle starts. All nodes should receive this table which also should be encrypted before transferring to the node. The key to encrypt this table can be written inside each node by the manufacturer. Only the certified dealer can update the list inside the main ECU in the case that any nodes needed to be replaced. The other requirement is a counter that can be provided by the vehicle's main ECU along with the key tables; Both tables also can be distributed using alternative channels. There are other approaches such as implicit counter synchronization or timestamp IV, which can be used. There should be a mechanisms to change each IV after using it once and all nodes should know the current value.

3.4 Security strength of SHA-3

There are several factors that directly connected to the security strength of SHA-3 such as collision, Pre-Image, security levels and capacity.

- Collision:
  Find X, Y so that

  $$hash(X) == hash(Y)$$

  Which is directly related to the capacity of sponge function no the output of the SHA-3. SHA-3 with 224 bit output has a capacity of 256.

- Pre-Image: Given Y, find X so that

  $$hash(X) == Y$$

  This property is important to the CAN because if an adversary captures the CAN traffic, he has access to both message and its digest. If the hash function lacks pre-Image resistance and the adversary can derive the key.
- Security Level Security level in cryptography is the measure of the strength of a cryptographic primitive which in our case is a hash function.
  Mathematically the collision resistance and pre-Image resistance and security level are equal to

  $$2^{c/2} \; where \; c = 256$$

  which means an adversary needs to perform 2^{128} operations which are more than

  $$3.4 \times 10^{38} operations.$$

3.5 Using SHA-3

To add source authenticity and semantic security, each node will use the 112 bit key and the IV(counter) received from the ECU along with the CAN message to generate a hash by using SHA3 algorithm. The result is a 224 bit message digest. This digest will be appended to the message itself and will be broadcasted to the network. One advantage of appending the hash to the CAN message is that the original CAN message has not been changed, so there is no need to change any mechanism already in place to prioritize messages. The system with little modification can be backward compatible, but in compatibility mode, there is no security in place. The reason this project did not propose a backward compatible solution is that the vehicle is almost a close system so the possibility of a manufacturer mixes both CAN FD nodes with CAN 2.0 nodes are very slim. From the type of message the receiver can guess the sender node, for example, if the data is the pressure of the front passenger side tire, the receiver logically knows that the pressure sensor node on the right passenger tire is the sender, so it will use the received key related to that node along with the counter, to generate the digest of the message. If the message is from the right node with correct counter both sender and receiver should produce the same message digest otherwise the node should generate an error message and broadcast that message to other nodes. Also, if an adversary got access to the right key and counter, they can use this information to craft a message and send that message, the legit node should broadcast a message to notify all nodes that there is a rouge node in the system.

4 Implementation

This section will present the implementation of the proposed solution to add security using SHA3-224 to the Control Area Network. This section is divided into 4 different subsections: 1- Implementing SHA3, 2- Sending Procedure, 3- Receiving Procedure, and 4- Underlying Infrastructure.

4.1 Implementing SHA3

So far there is no standard implementation of SHA3 in OpenSSL or any other solution on Linux. Thus, there were two options, first to write the code from scratch, which is normally a good way for any cryptographic solution, or use the available codes. The second approach was used for this project. There are more than 100 solutions in 10 different languages available on the GitHub. Most of the solutions were written in C language because of the importance of the performance, which is also a key factor in our implementation. For this project, a source code from Mattias Andrée is selected and modified to fit the purpose of this project and to measure the performance of hashing operations. You can find the information and a link to the code in the 8.1 This program is used standalone to check the performance of SHA3-224 with the infrastructure at hand.

4.2 CAN Communication

To send and receive CAN messages in this project, SocketCAN Functionality of Linux kernel is being used. SocketCAN is a set of drivers and networking stack, which is contributed by Volkswagen Research to the Linux Kernel. SocketCAN supports both CAN 2.0 and also the new version CAN-FD[36]. The main functionality of SocketCan is as follows[37]:

- PF_CAN - a protocol layer that provides a kernel for CAN frame sending and receiving.
- CAN_RAW - a protocol to copy CAN frame into the kernel space
- socket - creates a socket
- close - eliminate the created socket
- listen - accept incoming connection and receive the communicated data.

4.3 Sender

In this section the implementation of Sender is described. In order to prepare the packet, the sender needs to receive the message that is required to broadcast to the all CAN. The sender will append the key and the IV/Counter, which in this program is called the Pre-Digest message in this message and calculate the SHA3-224 digest of it which results in a 224 bit value. The best practice for vehicles is to implement the SHA3-224 directly into the hardware which makes it faster and more reliable[6]. But for this project we used a modified version of a program by Mattias Andrée. The sender will call the SHA3 program with a Pre-Digest message as an argument. The result of this execution is a 224-bit digest

which will be appended to the original message to be sent. To send the messages, a program called "cansend" from the CAN-Utils package[5] which is written in the language of C is being used. More information can be found in appendices 8.2 and 8.3.

4.4 Receiver

The receiving part of the message is more complicated than the sender's because the receiver uses more operations to make sure the packet is arriving from the legitimate source. After receiving the CAN message, the receiver will capture the last 224-bits of the message and store it into a variable named ReceivedHash and the rest of the message is stored into the variable named message. Then, the receiver will find the key by using a table that maps common messages to a key that should be used by legitimate node. Also, it will retrieve the counter that the node should use by using another mapping table that has the IV(counter) from previously communicated messages. The receiver appends both the key and the counter to the message and call the SHA3-224 hash function. The receiver will examine the calculated hash with the value in the RecievedHash variable, if they match that means the legitimate node has been sent an uncorrupted message. If the hash values were not equal, the receive should send an error message to notify the sender and other nodes. To receive the messages, a program called "candump" under the CAN-Utils package is being used. More information can be found at Appendix 8.2.

4.5 Underlying Infrastructure

There are three different infrastructures have been used for this project, The first one is Canberry Dual v2.1 which is an extension board for Raspberry Pi. This board is based on MCP2515 SPI controllers and ISO1050 transceivers. This board is being used to test the logic of the algorithm purposed in this project. The only reason Canberry was only used to only test the logic of the mechanism is that only CAN 2.0 are supported by MCP2515 controller 8. CAN interfaces on the Linux platform are like other networking interfaces such as Ethernet as you can see the both physical CAN interfaces and the virtual CAN on the figure 7.

4.5.1 Enabling CAN interfaces

The following commands are to enable CAN interfaces on a Raspberry Pi:

[5] https://packages.debian.org/source/sid/can-utils

```
root@raspberrypi:~# ifconfig can0
can0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          UP RUNNING NOARP  MTU:16  Metric:1
          RX packets:407 errors:0 dropped:0 overruns:0 frame:0
          TX packets:414 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
          RX bytes:2453 (2.3 KiB)  TX bytes:2378 (2.3 KiB)

root@raspberrypi:~# ifconfig can1
can1      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          UP RUNNING NOARP  MTU:16  Metric:1
          RX packets:414 errors:0 dropped:0 overruns:0 frame:0
          TX packets:407 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
          RX bytes:2378 (2.3 KiB)  TX bytes:2453 (2.3 KiB)

root@raspberrypi:~# ifconfig vcan0
vcan0     Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          UP RUNNING NOARP  MTU:72  Metric:1
          RX packets:261545 errors:0 dropped:0 overruns:0 frame:0
          TX packets:261545 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:16729941 (15.9 MiB)  TX bytes:16729941 (15.9 MiB)
```

Figure 7: CAN interfaces

- Enable the first CAN interface on a Canberry dual-channel:
  $ ip link set can0 up type can bitrate 1000000

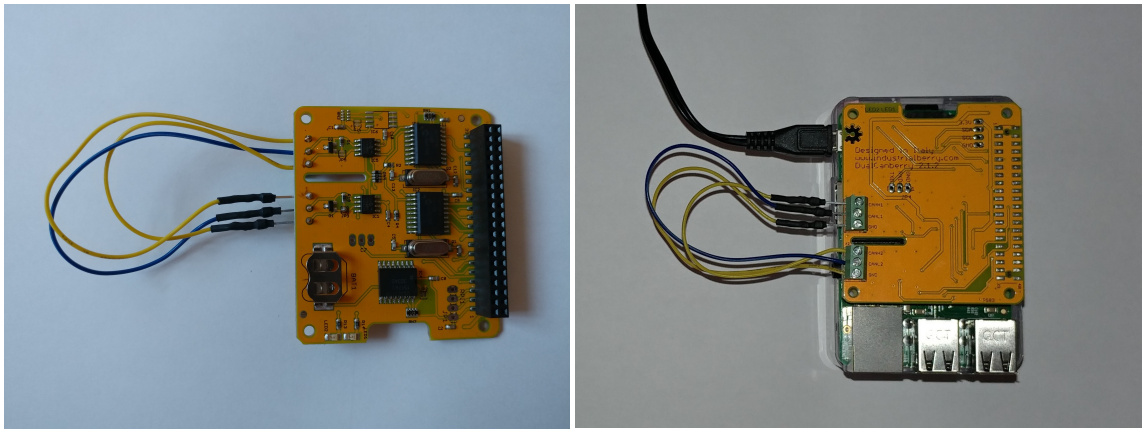
- Enable the second CAN interface on a Canberry dual-channel:
  $ ip link set can1 up type can bitrate 1000000

- Add virtual CAN kernel module:
  $ modprobe vcan

- Add Virtual CAN, set the payload size and turn the interface up:
  $ ip link add dev vcan0 type vcan
  $ ip link set vcan0 mtu 72
  $ ip link set vcan0 up

The second infrastructure being used is a Raspberry pi 3 with the following specifications:

- SOC: Broadcom BCM2837
- CPU: 4x ARM Cortex-A53, 1200 MHz (Our implementation only used one core)
- RAM: 1GB LPDDR2(900MHz)
- OS: Linux raspberrypi 4.4.50-v7+ armv7l GNU/Linux



(a) Dual-channel Canberry (b) Canberry on a Raspberry Pi

Figure 8: Canberry Kit

The third infrastructure is SAKURA-X which is an FPGA board used to measure the performance of SHA3-224 by using a Keccak high speed core, single cycle implementation with a 12MHz CPU clock. This infrastructure is being used to illustrate the power of SHA3 on the hardware.

5 Results

In this section the raw result is presented for each infrastructure, the first one is from the FPGA board Sakura-X to show the power of SHA3 base on a hardware design, the second one is the result from testing the SHA3-224 performance and the last one is the result of the performance of sending and receiving CAN FD messages via purposed security mechanism. Also, at the end of this section a performance result of an Keccak implementation of RFID devices is presented to further support the idea of this project.

5.1 SHA3-224 hardware implementation results

The result by running SHA3-224 on 10,000 random input message with a size of 1024-bits is 2 minutes and 34 seconds.

$$Numberofoperations\ per\ Second = \frac{10,000\ total\ messages}{154\ seconds} \approx 64\ Msg/Sec$$

The bottle neck is the file I/O. The operation itself only requires 47 clock cycles for single 1024-bits message.

5.2 SHA3-224 performance results

The result in this section is based on running the SHA3-224 program by using a binary data with a size of 256-bits as an input. As you can see the result in the figure 9, SHA3-224sum program operated 100000 times which took approximately 27.79 seconds.

$$Numberofoperations\ per\ Second = \frac{100,000\ total\ messages}{27.79\ seconds} \approx 3598.132\ Msg/Sec$$

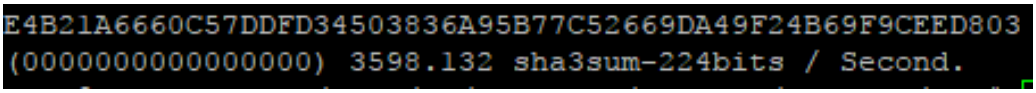


Figure 9: Result of SHA3-224

5.3 Send and Receive

The result in this section is based on the result of two programs, the sender and the receive. The former does work independently sue to the fact that CAN FD is broadcast based so there is no need for the sender to wait for any acknowledgement message but to measure the performance of the receiver there should be CAN FD messages present on the network so the receiver can parse and calculate the hash to check the authenticity and integrity of the message.

Send

Sending 10000 CAN FD messages along with the added security took approximately 13.24 seconds, figure 10.

$$Number\ of\ Message\ broad{-}casted\ per\ Second = \frac{10,000\ total\ messages}{13.24\ seconds} \approx 755.21\ Msg/Sec$$

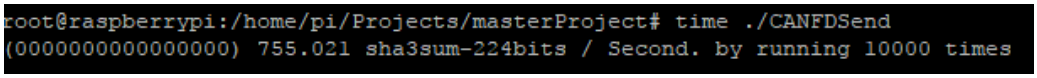


Figure 10: Number of messages broadcasted per second

Receive

Receiving 10000 CAN FD messages along with added security took approximately 8.01 seconds, figure 11.

$$Number\ of\ messages\ received\ per\ Second = \frac{10,000\ total\ messages}{8.01\ seconds} \approx 1248.43\ Msg/Sec$$

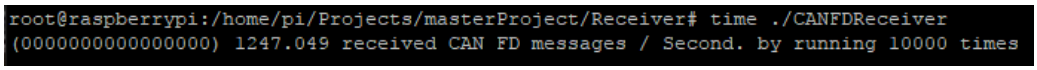


Figure 11: Number of message received per second

5.4 Keccak on RFID device

The result presented on the figure 12 is from the research by Elif Bilge Kavun and Tolga Yalcin[6] which they implement Keccak on a lightweight devices such as radio frequency identification cards. Based on their research we see that the parallel Keccak-f[1600] has the throughput speed of 4533 Kbit/second at the CPU clock frequency of only 100 KHz.

	Hash output size	Data path size	Input data size	Cycles per block	T/put at 100 KHz (Kbps)	Area (KGE)	Efficiency (bps/GE)	Power cons. (µW/MHz)
Parallel *Keccak-f*[1600]	256	64	1088	24	4533	47.63	95.40	315.1
Parallel *Keccak-f*[400]	128	16	144	20	720	10.56	68.18	78.1
Parallel *Keccak-f*[200]	64	8	72	18	400	4.9	81.63	27.6
Serial *Keccak-f*[1600]	**256**	**64**	**1088**	**1200**	**90.66**	**20.79**	**4.36**	**44.9**
Serial *Keccak-f*[800] (estimate)	**128**	**32**	**544**	**1100**	**49.45**	**13.00**	**3.80**	**28.2**
Serial *Keccak-f*[400]	**128**	**16**	**144**	**1000**	**14.4**	**5.09**	**2.83**	**11.5**
Serial *Keccak-f*[200]	**64**	**8**	**72**	**900**	**8**	**2.52**	**3.17**	**5.6**

Figure 12: Performance of Parallel vs Serial Keccak[6]

6 Discussion

In this section, the information about the baseline specification necessary to satisfy the Class C specification of CAN network is provided. Class C is for critical real-time communication and the speed should be between 125 *Kbps* and 1 *Mbps*[14]. Class C has a minimum speed of 125 *Kbps* which means:

$$Number\ of\ bits\ per\ second = 125\ Kbps \times 1024 = 128,000\ bit/s$$

In order to satisfy the minimum speed, the sender should transmit 128,000 bits every second which is equal to 250 CAN FD messages per second.

$$Number\ of\ CAN\ FD\ message\ per\ second = \frac{128,000\ bit/s}{512\ bits} = 250\ CAN\ FD\ messages$$

With the current hardware specification, our implementation satisfies the minimum requirement for the sender.

$$Secure\ CAN\ FD\ sender\ Throughput = \frac{755.21\ MSG/s \times\ 512 bits MSG}{1024} \approx 377.605\ Kbps$$

The results are satisfactory, but this is based on the hardware used for this project. The sender used one core of the CPU with the clock frequency of 1200 *MHz*. If we simplify the relationship of the clock frequency to the performance of computing and transferring CAN FD to be a linear relationship, the lowest CPU clock frequency to satisfy the Class C is approximately 400 *MHz*.

$$Min.\ required\ CPU\ clock\ frequency = \frac{125 Kbps}{377.605\ Kbps} \times 1200\ MHz \approx 400 MHz$$

The receiver had the same environment as the sender and like the sender satisfied the requirement to be in Class C.

$$Secure\ CAN\ FD\ receiver\ Throughput = \frac{1248.43\ MSG/s \times\ 512 bits MSG}{1024} \approx 624.215\ Kbps$$

$$Min.\ required\ CPU\ clock\ frequency = \frac{125 Kbps}{624.215\ Kbps} \times 1200\ MHz \approx 240 MHz$$

on the other hand, the result from FPGA shows that we need to have a CPU with the clock frequency of 23.4 *MHz* to compute SHA3-224 with the throughput of 125 *Kbps*.

$$Min.\ required\ CPU\ clock\ frequency\ for\ FPGA = \frac{125\ Kbps}{64\ iteration/s} \times 12\ MHz \approx 23.43 MHz$$

This value far less than the amount of CPU required to have the same SHA3-224 throughput from the software implementation which is approximately 167 *MHz*.

$$\frac{(3598.132\ \textit{SHA3-224}\ \textit{MSG/s} \times 256\ \textit{bit/MSG})}{1024} \approx 900\ \textit{Kbps}$$

$$\frac{125\ \textit{Kbps}}{900\ \textit{Kbps}} \times 1200\ \textit{MHz} \approx 167\ \textit{MHz}$$

Still the best way to use a CAN node and implement the mechanism because the relationship between CPU clock frequency and SHA3 computation is complex and also there are other factors such as the speed of memory affecting the result.

The results are all provided in the table below:

Function	Message/Second	Throughput	Note
SHA3-224	3598.132	787.0 Kbps	Tested speed
CAN FD Sender	755.21	377.6 Kbps	Security enabled CAN FD
CAN FD Receiver	1248.43	624.2 Kbps	Security enabled CAN FD
CAN 2.0 Standard	N/A	1024 Kbps	The currently used CAN standard
CAN FD Standard	N/A	3.7 Mbps	Max. achieved speed in the lab environment[38]

7 Conclusion

In this project, a new mechanism is introduced to use SHA3 hashing algorithm to add authentication and integrity to the Control Area Network of vehicles. There is no transparency in the Vehicle industry related to the hardware specification they use, nor the software implemented in vehicles, this project utilized the virtual CAN and Canberry kit which uses Raspberry Pi. The simplified mechanism worked on Canberry which only supports CAN 2.0. The reason CAN 2.0 is used is to test the logic of the proposed mechanism since the CAN FD trancievers are not available for public. The results on the Virtual CAN is promising, but the problem is the hardware used in this project is like a supercomputer compared to the micro controller implemented in the actual vehicles. Thus, the minimum hardware specification required to use the method in this project is presented. More work needs to be done so the future researche can be in the following areas, but not limited to:

- **Hardware Implementation**: As it was presented in the FPGA and RFID implementation, SHA3 is far faster; Thus, there are room to implement and test it on a physical CAN node capable of processing CAN FD messages.

- **Improving the mechanism**: There is always room to make the mechanism perform better. One way is to write the code in binary to make it faster or to use better libraries. The other approach to improve the mechanism is to eliminate all the disc I/O which was the bottleneck in the proof of concept code for this research as on the real CAN node there is no disc I/O.

- **Testing the security after implementation**: It is valuable to also test the security of implementing the method. There are two approaches, one is to test the strength of SHA3 and the other is to test the way SHA3 is being used, because many times the algorithm itself is almost impossible to break but the way it is implemented can be exploited due to the poor implementation.

References

[1] N. Instrument. (2016) Can fd frames. [Online]. Available: https://http://zone.ni.com/reference/en-XX/help/372841L-01/nixnet/canfdframes/

[2] Endres. (2014) Can fd frames. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/5/5e/CAN-Bus_in_base_format_without_stuffbits.svg

[3] G. Leen. (2002) One subset of a modern vehicle's network architecture. [Online]. Available: http://ieeexplore.ieee.org.ezproxy.rit.edu/mediastore/IEEE/content/media/2/21069/976923/976923fig1hires.gif

[4] Armbrust. (2011) Sponge construction. [Online]. Available: https://commons.wikimedia.org/wiki/File:SpongeConstruction.svg

[5] G. Bertoni. (2008) Sponge construction. [Online]. Available: http://keccak.noekeon.org/

[6] E. B. Kavun and T. Yalcin, "A lightweight implementation of keccak hash function for radio-frequency identification applications," in *International Workshop on Radio Frequency Identification: Security and Privacy Issues*. Springer, 2010, pp. 258–269.

[7] T. Schutte, "Phil 308 research paper privacy implications of vehicular edr and gps tracking devices."

[8] "Sha-3standardization," http://csrc.nist.gov/groups/ST/hash/sha-3/sha-3_standardization.html, 2013.

[9] N. Eddy, "Gartner: 21 billion iot devices to invade by 2020." [Online]. Available: http://www.informationweek.com/mobile/mobile-devices/gartner-21-billion-iot-devices-to-invade-by-2020/d/d-id/1323081

[10] G. Nicholas, *Cars 1886-1930*. Beekman House, 1985.

[11] K. Koscher, A. Czeskis, F. Roesner, S. Patel, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "2010 ieee symposium on security and privacy experimental security analysis of a modern automobile." [Online]. Available: http://www.autosec.org/pubs/cars-oakland2010.pdf

[12] R. B. GmbH, "History of can technology." [Online]. Available: https://www.can-cia.org/can-knowledge/can/can-history/

[13] R. S. Marques, N. Navet, and F. Simonot-Lion, "Configuration of in-vehicle embedded systems under real-time constraints," in *2005 IEEE Conference on Emerging Technologies and Factory Automation*, vol. 1, Sept 2005, pp. 8 pp.–414.

[14] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaâniche, and Y. Laarouchi, "Survey on security threats and protection mechanisms in embedded automotive networks," in *Dependable Systems and Networks Workshop (DSN-W), 2013 43rd Annual IEEE/IFIP Conference on*. IEEE, 2013, pp. 1–12.

[15] C. Leon, M. Leon, R. Hernandez, C. Hernandez, H. Rodriguez, and F. Rodriguez, "Achieving confidentiality security service for can," in *Proceedings of the 15th International Conference on Electronics, Communications and Computers*, ser. CONIELECOMP '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 166–170. [Online]. Available: http://dx.doi.org/10.1109/CONIEL.2005.13

[16] M. D. Hamilton, M. Tunstall, E. M. Popovici, and W. P. Marnane, "Side channel analysis of an automotive microprocessor," in *IET Irish Signals and Systems Conference (ISSC 2008)*, June 2008, pp. 4–9.

[17] A. Greenberg, "Hackers remotely kill a jeep on the highway-with me in it." [Online]. Available: https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway

[18] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive {CAN} networks, practical examples and selected short-term countermeasures," *Reliability Engineering and System Safety*, vol. 96, no. 1, pp. 11 – 25, 2011, special Issue on Safecomp 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0951832010001602

[19] A. Groll and C. Ruland, "Secure and authentic communication on existing in-vehicle networks," in *2009 IEEE Intelligent Vehicles Symposium*, June 2009, pp. 1093–1097.

[20] A. Van Herrewege, D. Singelee, and I. Verbauwhede, "Canauth-a simple, backward compatible broadcast authentication protocol for can bus," in *ECRYPT Workshop on Lightweight Cryptography*, vol. 2011, 2011.

[21] P. Kleberger, T. Olovsson, and E. Jonsson, "Security aspects of the in-vehicle network in the connected car," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 528–533.

[22] M. Meier, "Local interconnect network," 2007.

[23] R. Makowitz and C. Temple, "Flexray-a communication network for automotive control systems," in *2006 IEEE International Workshop on Factory Communication Systems*, 2006, pp. 207–212.

[24] M. Wolf, "Security analysis and characteristical constraints in the automotive domain," in *Security Engineering for Vehicular IT Systems*. Springer, 2009, pp. 91–103.

[25] D. K. Nilsson and U. Larson, "A defense-in-depth approach to securing the wireless vehicle infrastructure." *JNW*, vol. 4, no. 7, pp. 552–564, 2009.

[26] "Feasible car cyber defense by arilou technologies," in *embedded security in cars (escar) conference. (2010)*, 2010.

[27] "3+1 cyber security protection framework - fuse," https://www.visualthreat.com/UIproducts.action, 2016.

[28] C. Miller and C. Valasek, "Adventures in automotive networks and control units," *DEF CON*, vol. 21, pp. 260–264, 2013.

[29] M. Wolf, A. Weimerskirch, and C. Paar, "Security in automotive bus systems," in *Workshop on Embedded Security in Cars*, 2004.

[30] K. Han, S. D. Potluri, and K. G. Shin, "On authentication in a connected vehicle: Secure integration of mobile devices with vehicular networks," in *Cyber-Physical Systems (ICCPS), 2013 ACM/IEEE International Conference on.* IEEE, 2013, pp. 160–169.

[31] D. K. Nilsson, U. E. Larson, and E. Jonsson, "Efficient in-vehicle delayed data authentication based on compound message authentication codes," in *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th.* IEEE, 2008, pp. 1–5.

[32] O. Hartkopp and R. M. SCHILLING, "Message authenticated can," in *Escar Conference, Berlin, Germany*, 2012.

[33] J. A. Bruton, "Securing can bus communication: An analysis of cryptographic approaches," 2014.

[34] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Keccak sponge function family main document," *Submission to NIST (Round 2)*, vol. 3, p. 30, 2009.

[35] ——, "The keccak sha-3 submission," *Submission to NIST (Round 3)*, vol. 6, no. 7, p. 16, 2011.

[36] O. Hartkopp and A. Volkswagen, "The can networking subsystem of the linux kernel," *Proceedings of the 13th iCC*, 2012.

[37] M. Kleine-Budde, "Socketcan–the official can api of the linux kernel," in *Proceedings of the 13th International CAN Conference (iCC 2012), Hambach Castle, Germany CiA*, 2012, pp. 05–17.

[38] "Can and can-fd a brief tutorial," http://computer-solutions.co.uk/info/Embedded_tutorials/can_tutorial.htm, 2017.

8
Appendix A: Source Code Information

8.1 SHA3-224

8.2 CAN-Utils

8.3 SHA3-224 on CAN-FD Source Code

The source code of the programs written for this project can be accessed by using the following link: *https://github.com/aidinski/SHA3-224-on-CAN-FD*