Performance Analysis of Security Mechanism for Automotive Controller Area Network

Mabrouka Gmiden[1], Mohamed Hedi Gmiden[2], Hafedh Trabelsi[3]
Computer and Embedded System Lab (CES), National Engineers
School of Sfax-Tunisia[1, 2, 3]

***Abstract*—Connectivity of modern cars has led to security issues. A number of contributions have proposed the use of cryptographic algorithms in order to provide automotive Controller Area Network (CAN) security. However, due to CAN protocol characteristics, real time requirements within cryptographic schemes are not guaranteed. In this work, effects of implementing cryptographic approaches have been investigated by proposing a performance analysis methodology of cryptographic algorithm. Until get implanting the proposed method in a real vehicle, a platform based on STMicroelectronics'32F407 (STM32F407) microcontroller board has been deployed to test the proposed methodology. The experiments show that the implementation of a cryptographic algorithm has an impact on clock cycles number and therefore, on real-time performances.**

***Keywords*—Automotive CAN security; cryptographic algorithms; analysis methodology; real-time performances**

I. Introduction

New vehicles are becoming more and more connected machines. In fact, a modern car is able to communicate with the outside via various interfaces like USB, MP3, Bluetooth, etc. Furthermore, CAN protocol is today the most used in automotive networks [1]. However, CAN bus cannot guarantee security because of a lack of authenticity [2] [3]. Therefore, CAN networks are vulnerable to cyber-attacks, which threats in-vehicle subsystems even lives of passengers [4]. Then, security problem is added to the automotive issues [5] [6]. Hence, it is crucial to find solutions that guarantee automotive security.

In order to protect the safety of the system within a modern car, many methods have been developed, such as cryptographic protocols, Intrusion Detection Systems (IDSs), etc. Although, several researches have been oriented towards IDSs, they have been still not 100% robust, and they could not prevent all types of attacks. To exceed limitations of detective measures, many researches aim to adopt cryptographic strategies as they have been improved, in internet networks, their efficiency in thwarting attacks.

Applications in CAN network are characterized, unlike traditional computer systems, by real- time constraints. That is why data encryption or signature mechanisms should not impact real-time performances. In the literature, although the diversity of the proposed solutions, serious performance measures are still limited.

In this paper a tool, that allows the analysis of real-time performances resulting from the implementation of cryptographic algorithms, is designed. The method is based on the analysis of the time intervals of CAN frames.

The main contributions of this work are:

- A general literature review about CAN bus security issues along with proposed solutions for this accomplish.
- A practical methodology to secure CAN bus communication based on analyzing and measuring of real-time performances.
- An efficient experimental platform for analyzing, implementing a securing CAN bus communication and injecting spoofed message.

The rest of the paper is organized as follows. Section 2 provides the necessary background of the security issues and related work. Next, the proposed method is described. Section 4 presents the evaluation result of the proposed method. Finally, Section 5 summarizes this work.

II. Background

A. CAN Bus Security Issues

The objectives provided by a security system are called security services which they are summarized as confidentiality, authenticity, availability, integrity, and non-repudiation. CAN bus cannot guarantee these properties since its characteristics:

- Broadcasted nature: a CAN message sent by a node will be received by all nodes connected to the bus. So, an attacker can connect to the network traffic and read data frame easily. Then, the CAN bus cannot guarantee confidentiality.
- CAN messages have not any authenticator fields. Thus, an attacker connected to the bus could use the identifier (ID) of any node to send a fake message.
- Arbitration scheme: a frame consists, mainly, as Fig. 1 shows, of: the ID, which represents the priority of the message, Data Length Code (DLC), Data, and Cyclic Redundancy Check (CRC). The identifier of the CAN frame is unique. So, the CAN message with the highest priority wins the arbitration and transmits the first. Thus, any node can put the bus in a dominant state and prevent others from sending messages resulting Denial of Service (DoS) attacks.

- CAN protocol uses CRC to verify whether a message has been modified. However, this latter cannot prevent an attacker from modifying a legitimate message. In fact, she could make a correct CRC for a forged message.
- Possibility of repudiation: in CAN protocol, it is impossible for a legitimate ECU to prove that it has sent or received a given message.
- CAN message contain between 1 and 8 bytes. So, the security protocol cannot transmit any extra authenticated data inside the classic data field (Fig. 1).
- In automotive networks, the primary focus is on real-time capabilities, which are needed to respond within a given short time. So, predictability and reliability are the dominating factors.

B. *Requirements of CAN Bus Security Solutions*

Since Electronic Control Units (ECUs) are very limited in computing power and memory space, heavy cryptographic functions are difficult to be performed by these calculators. So, proposed solutions should be as lightweight as possible. Moreover, almost CAN networks applications are hard-real time. Therefore, embedded real time performances should not be impacted by the implementation of security mechanisms.

In addition, the proposed mechanism should provide retro-compatibility, i.e. be compatible with used technologies and interoperability, i.e. external communications should not be prevented by the security system. Furthermore, a CAN data frames are easy to be eavesdropping by an attacker. Thus, a method of encryption should be employed in order to provide confidentiality. On otherwise, a Hash-based Message Authentication Code (HMAC) must be generated and transmitted along with CAN messages to guarantee authentication of transmitted data,

C. *Related Work*

As a countermeasure against various types of vehicle cyber-attacks, there have been two main groups of security solutions: Intrusion Detection Systems and cryptographic mechanisms

1) Intrusion detection: To defend attacks against in-vehicle networks, many solutions based on IDS Systems have been proposed.

Studnia et al. proposed an intrusion detection approach for embedded automotive network [7]. The presented solution based on the definition of a formal language. This proposal is dedicated to generate a set of signature for attacks that aim to detect. In [8], authors presented a novel intrusion detection algorithm which aims to identify malicious CAN messages injected by attackers. By against, an intrusion detection algorithm, which is based on the analysis of time intervals of messages, was proposed [9]. The algorithm did not require any hardware modification, but it could not detect irregular message in coming.

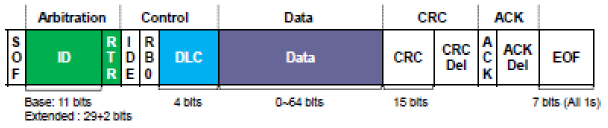

Fig. 1. CAN Format Frame.

2) Message authentication: Although, several researches have been oriented towards IDS system, significant increase of cryptographic schemes have been shown during last years.

Woo et al. in [10] proposed the use of HMAC and Advanced Encryption Standard-128 (AES-128) for encryption. The proposed protocol used 16 bits, in the extended ID field, and the 16-bit CRC field for transmission of 32 bits code. The implementation of the protocol kept the bus load under 50%; hence it provided acceptable overhead. Nurnberger et al. introduced VatiCAN which enabled sender and receiver ECUs to exchange authenticated data using the Keccak algorithm [11]. By contrast to other authentication mechanisms, VatiCAN used individual keys per ECU. So, each calculator should store the key of each ECU exchanges authenticated messages with. In their protocol [12], Bulck et al inspired the idea of VulCAN from the two protocols VatiCAN and Leia. In VulCAN, each authenticated CAN identifier should be associated with a symmetric 128- bit cryptographic key. As in VatiCAN, VulCAN allowed multiple IDs to distribute the same key. While valid ECUs use the key to compute a 64- bit MAC, the value of counter (which prevents re- play attack) increases. Like VatiCAN, authors addressed nonce initialization challenge by the use of short-term session keys and (re-)synchronized counters by a global Nonce Generator (NG).

In [13] a method based on adapting traditional encryption schemes was presented. The proposed system required that the hardware modules installed on each ECU, which made the implementation more difficult. The proposed method differentiated itself from other competitive tools; by not only supporting cryptography mechanism; but also allowing the measure of real-time parameters. Also, in [14], authors used the same platform to implement an IDS. The main contribution in this paper is the design of a platform which allowed the implementation of an IDS. So, the same tool is deployed in this paper to measure real-time performances resulting from the implementation of cryptographic mechanisms. The method is based on the analysis of the time intervals of the CAN message.

III. Evaluation of Cryptographic Algorithms

The system proposed in this paper, aims at analyze the security requirements on CAN bus network after implementing a cryptographic mechanism. This section is devoted to the detailed presentation of the proposed system: subsection A introduces the system model, subsection B gives phases of methodology process and subsection C provides algorithms process.

A. System Model

In this section, the system model, which is adopted for implementing the proposed method, is introduced. As shown in Fig. 2, the system model is composed of 2 CAN nodes connected to a CAN bus to form a network.

Assume that Node 1 sends messages to Node 2 with ID =0x1 every 2ms. Likewise, Node 2 send messages to Node 1 with ID =0x2 every 5ms.

B. Fundamental Idea

The Main problem, on the communication side, is the overhead caused by the additional data in combination with possible additional latency. Both are especially challenging when dealing with short signals requiring real time operation and low latencies. The goal of this work is to develop a system which can be deployed for implanting a cryptographic mechanism along with analyzing real-time performances and injecting spoofed message. Therefore, a platform based on STM32F4 board, is deployed. The proposed method enables to determine effects of implementing security mechanism on CAN bus performances.

The fundamental idea is to apply a cryptographic mechanism on a given message in Node 1 and send it to Node 2. After the transmission of a message frame, performances of the related algorithm is measured according to a method will be detailed later.

C. Phases of Methodology Process

Since the proposed approach is designed to be implemented in the standard version of CAN protocol, the transmission process of CAN messages will be different from the classic one. The whole transmission process is summarized in Fig. 2.

When the sender node receives a request from the receiver, it encrypts data; divides it into segments. Then, it sends segments via CAN bus. To guarantee confidentiality and integrity of automotive data network, the encryption of messages is required. The CAN message encryption phase is insured by encryption mechanisms and MAC methods.

1) Fragmentation technique: As the maximum payload length of CAN data field is only 8 bytes, the available space, for appending a Message Authentication Code (MAC), is very limited. Rather than appending a MAC in one CAN frame's data field, dividing data into segments is suggested; and, then, each segment is transmitted.

2) CAN message transmission phase: The transmission of CAN frame is carried out from the sender to the receiver according to CAN protocol and via CAN bus.

3) CAN message reconstitution phase: Arrival messages need to be reconstituted for obtaining the complete message.

4) CAN message decryption phase: The resulted message is decrypted to obtain the original message.

5) Calculating clock cycle: The last step of the methodology is to calculate the clock cycle needed to perform a CAN data transmission.

IV. Test and Evaluation

Testing the proposed method on a real car, with the same requirements and conditions, is very difficult. For that, the authors tried to establish a platform which can resemble same conditions of automotive network. Subsection A details different components of the system and results of experimental tests are assessed in subsection B.

A. System Implementation

1) Experimental setup: In order to design the model system, several solutions are possible (PIC microcontroller, ARDUINO board, Raspberry PI board...). In this paper, STM32F407 microcontroller board, with a 32 bit ARM Cortex-M4 core clocked at 16 MHz and an adaptive real-time accelerator is used[1]. The high-speed CAN transceiver MCP 2551 is used as a transceiver. In this work, the two nodes connected with an oscillator clock of 16MHz. Node 1 ,which is connected to the PC, is dedicated to send messages to Node 2 across the CAN bus. The setup is shown in Fig. 3.

2) Implementation: For the implementation of algorithms, Keil Microcontroller Development Kit (MDK) 5 is adopted as an integrated development environment (IDE) to program STM32 in C language.

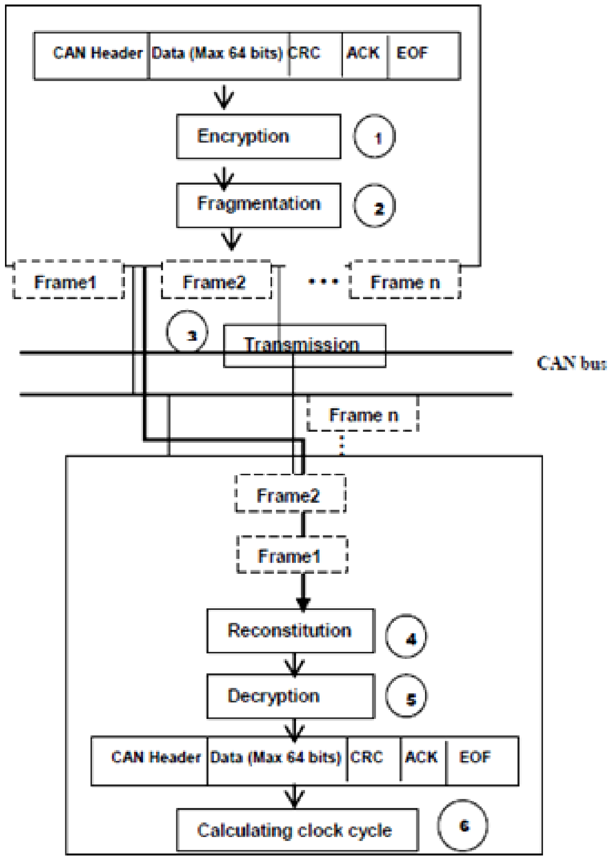

Fig. 2. Overall Process of Analyzing Methodology for a Secure CAN Bus Communication.

[1]STM32F407VG, http://www.st.com/en/microcontrollers/stm32f407vg.html, 2017.

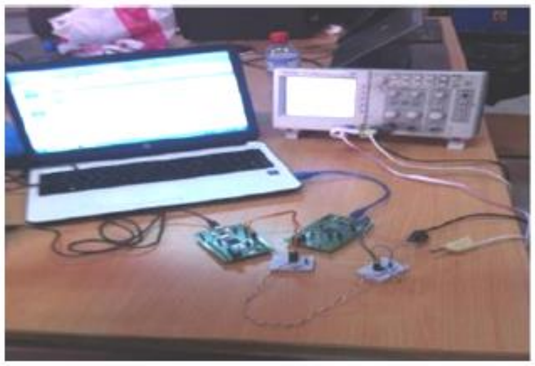

Fig. 3. Test Environment.

On otherwise, STM32CUBEMX is used to configure the STM32 board.

To integrate security into the CAN bus network, authors included the STM32 cryptographic library package (X- CUBE CRYPTOLIB) in particular AES in CMAC mode (AES_CMAC) and HMAC_SHA (Hash-based Message Authentication Code (HMAC) à l' aide de SHA).

3) Algorithms: In order to design a secure CAN network, an encryption algorithm block or a hash function is added to CAN network. This algorithm enables to encrypt data sent by the sender and to decrypt data received by the receiver.

The following section focuses on AES_128_CMAC, AES _256_CMAC, HMAC_SHA196 and HMAC_SHA256 algorithms. Due to the complexity in automotive architecture, the implementation of these algorithms, as codes which can be implemented in CAN nodes, is a challenge. For the implementation of this approach, the same platform as well as the same configuration steps as those indicated in [14].

a) AES_128_CMAC: AES_128_CMAC has a key with 128 bits and gives a message with 128 bits in output. Since the CAN data message can contain 108 bits in totally, authors chose to append MAC in the data field and truncate it to two segments.

Assuming that node 1 require sending data D0, of 64 bits, to node 2 via a secure CAN bus. In first example, AES_128_CMAC algorithm is applied. The transmission process of the encrypted message is summarized in Fig. 4.

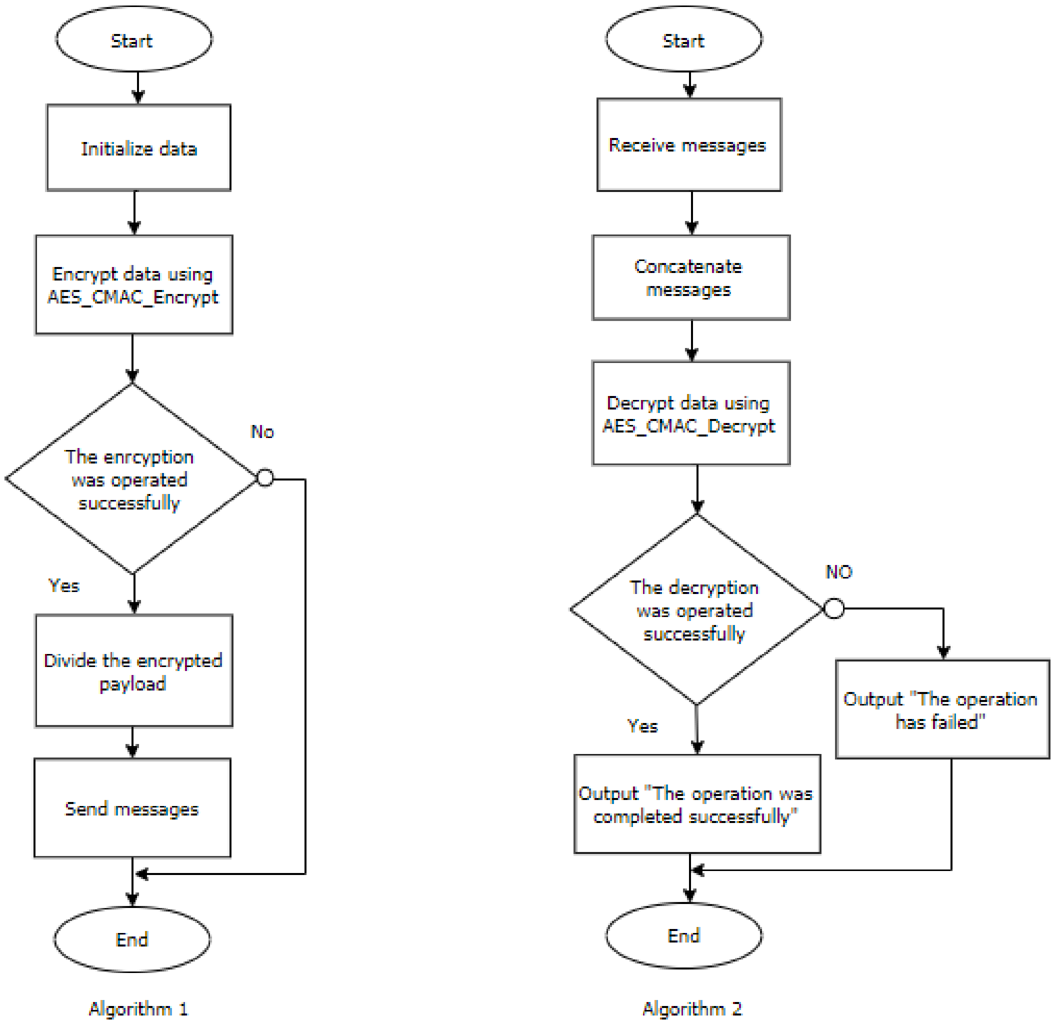
Start
Initialize data
Encrypt data using AES_CMAC_Encrypt
The enrcyption was operated successfully
No
Yes
Divide the encrypted payload
Send messages
End
Algorithm 1
Start
Receive messages
Concatenate messages
Decrypt data using AES_CMAC_Decrypt
The decryption was operated successfully
NO
Yes
Output "The operation has failed"
Output "The operation was completed successfully"
End
Algorithm 2

Fig. 4. Transmission Process of a Secure Data.

The process of decrypt frames is summarized in Algorithm 1.

Step 1: Start.
Step 2: Initialization.
Step 3: Encrypt data using AES_CMAC_Encrypt.
Step 4: Check encryption operation by comparing the encrypted payload by the expected text.
Step 5: Encryption is failed? Pass to Step7.
Step 6: Divide the encrypted payload in two segments.
Step 7: Send the two messages to node 2.
Step 8: End process.

When node 2 receives messages, it decrypted them as shown in Algorithm 2.

Step 1: Start
Step 2: Receive messages.
Step 3: Concatenate the two messages.
Step 4: Decrypt data using AES_CMAC_Decrypt
Step 5: Check decryption operation by comparing the obtained text by the expected text.
Step 6: Decryption is failed? Pass to Step 7 then Step 9. Else pass to Step 8.
Step 7: Output "The operation was completed successfully".
Step 8: Output "The operation has failed".
Step 9: End process.

b) Algorithm of HMAC_SHA256: As a second example, we applied to the D0 a HMAC SHA256 block. HMAC_SHA256 gives a message with 256 bits in output. In this case, MAC was appended in the data field then truncated to four segments.

The transmission of a secure message using HMAC_SHA256 is summarized in Fig. 5.

Algorithm 3 shows the authentication of the data.

Step 1: Start.
Step 2: Initialization.
Step 3: Hashing data using HMAC_SHA256.
Step 4: Hashing is failed? Pass to Step7.
Step 5: Divide the hashed payload in four segments.
Step 6: Send the four messages to node 2.
Step 7: End process.

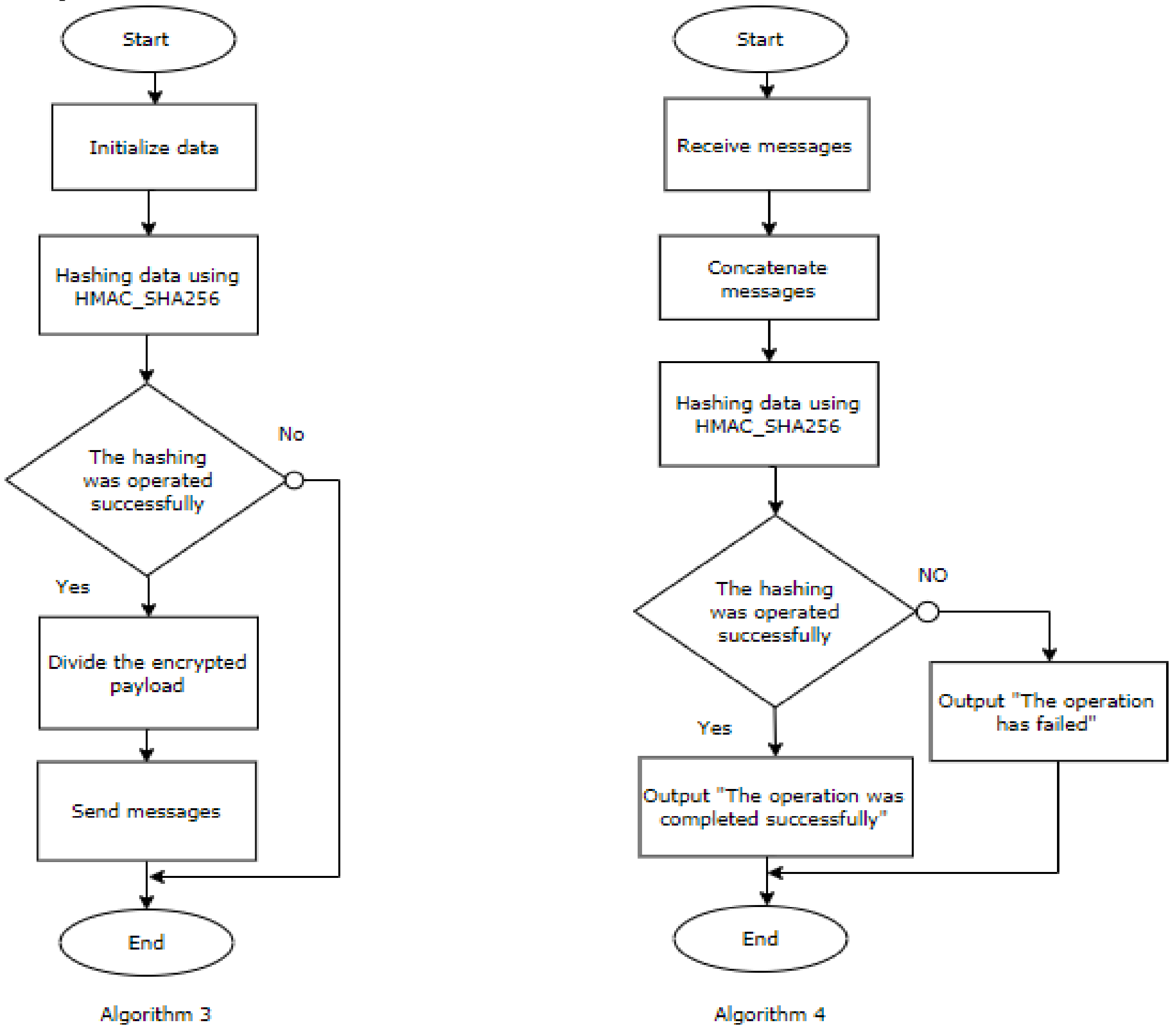
Start
Initialize data
Hashing data using HMAC_SHA256
The hashing was operated successfully
No
Yes
Divide the encrypted payload
Send messages
End
Algorithm 3
Start
Receive messages
Concatenate messages
Hashing data using HMAC_SHA256
The hashing was operated successfully
NO
Yes
Output "The operation has failed"
Output "The operation was completed successfully"
End
Algorithm 4

Fig. 5. Transmission Process of an Authantecated Data.

Algorithm 4 shows the verification of authentication using HMAC_SHA256.

```
Step 1:  Start
Step 2:  Receive messages.
Step 3:  Concatenate the four messages.
Step 4:  Operate HMAC_SHA256
Step 5:  Check hashing function by comparing the obtained
    text by the expected text.
Step 6:  Hashing is failed? Pass to Step7 then Step 9. Else pass
    to Step 8.
Step 7:  Output "Operation success".
Step 8:  Output "Operation fails".
    End process.
```

4) Clock cycle calculation: The tests were executed on STMF4 which their CPU is running at 168MHz.

To determine the impact of using a cryptographic algorithm in CAN bus communication, the number of clock cycles should be calculated. At first, the number of cycles needed to perform each process is defined as follows:

$$Cycles_{CAN} = Init\ key\ cycle + Init\ message\ cycle + \text{Process block of data cycle} * \text{number of blocks} \quad (1)$$

So, the number of cycles required to transmit a CAN message encrypted by AES_128_ CMAC is calculated as follows:

$$\text{Cycles}_{\text{CAN}}\text{= Init key cycle + Init message cycle} + \text{Process block of data cycle} * \text{number of blocks} \quad (2) + 2*(\text{min of CAN data transmission cycle})$$

The number of cycles required to transmit a CAN message encrypted by HMAC_SHA256 is calculated as follows:

$$\text{Cycles}_{\text{CAN}} = \text{Cycle de Init\_key + Cycle de Init\_message} + \text{cycle de block de donnée} *(\text{nombres de block}) \quad (3) + 4*(\text{min de cycle de transmission de donnée CAN})$$

V. Evaluation

The size of the code requested by each proposed algorithm is presented in Table 1.

A. Comparison between AES-CMAC

Table 2 shows the number of clock cycles requested by AES_CMAC.

Referring to Fig. 6, the performances of AES _CMACs don't much differ when the key size differs. It is because of the change of cycle numbers taken by each algorithm.

TABLE I. Code Size of Each Algorithm to Process a Block of Data

Algorithm Mode	Code Size (Byte)	Constant Data Size (Byte)
AES (128, 192, 256) CMAC	5 796	6 040
HMAC_SHA256 ,128	3485	6040

TABLE II. Performance of AES-CMAC Algorithms

Algorithm Mode	Operation	Init key	Init message	Data block processing
AES_128_CMAC	Decryption	636	639	1628
	Encryption	618	525	1628
AES_192_CMAC	Decryption	632	719	1859
	Encryption	616	608	1859
AES_256_CMAC	Decryption	840	758	2141
	Encryption	816	649	2141

The difference between the numbers of clock cycles, used by the three algorithms, is relatively little. However, AES_256_CMAC requires more clock cycles than AES_128_CMAC and AES_192_CMAC; hence it is the slowest one. On the other hand, AES_128_CMAC is the least secure since it has the shortest key. However, this latter is considered faster than the others. It takes the least number of clock cycles; hence it has better performance than AES_192_CMAC and AES_256_CMAC.

Therefore, the variation of key size affects the number of clock cycles and subsequently the performances of algorithm. On the one hand, the key of AES_256_CMAC is longer than the key of AES_128_CMAC. Thus, AES_256 is more secure. But the AES_ 256 is slower than AES_128.

B. Comparison between HMAC_SHA

Table 3 shows the number of clock cycles requested by HMAC_SHA.

Referring to Fig. 7, the performances of HMAC_SHA don't much differ when the key size differs. It is because of the change of cycle numbers taken by each algorithm. The difference between the numbers of clock cycles, used by the three algorithms, is relatively little. However, HMAC_SHA256 requires more clock cycles than HMAC_SHA224; hence it is the slowest one. On the other hand, HMAC_SHA224 is the least secure since it has the shortest key. However, this latter is considered faster than HMAC_SHA256; hence it has better performances. Therefore, the variation of key size affects the number of clock cycles and subsequently the performances of algorithm.

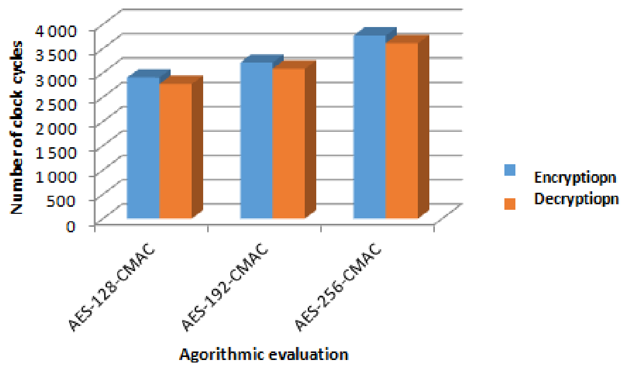

Fig. 6. Comparison between AES_CMAC.

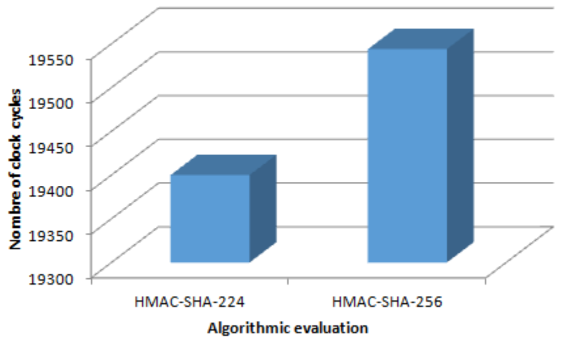

Fig. 7. Comparison between HMAC_SHA.

TABLE III. PERFORMANCE OF HMAC_SHA ALGORITHMS

Algorithm Mod	**Init key**	**Init message**	**Finalization**
HMAC_SHA224	4 708	3 352	11 340
HMAC_SHA256	4 789	3 352	11 403

C. Comparison between HMAC_SHA256 and AES_256_CMAC

The comparison between number of clock cycles of HMAC_SHA256 and AES_256_CMAC is shown in Fig. 8, since they have the same key size. The difference between the numbers of clock cycles, taken by the two algorithms, is not large. However, the HMAC_SHA requires more of number of clock cycles than the AES_CMAC; hence this latter is faster. Therefore, AES_CMAC algorithm requires has better performances than the HMAC_SHA.

As can be seen, the effectiveness of the algorithm for providing security depends on the key length and the protocol mode. These parameters can have an impact on the speed execution of algorithm. In addition, the security mechanism has an impact on system bus load and message latencies.

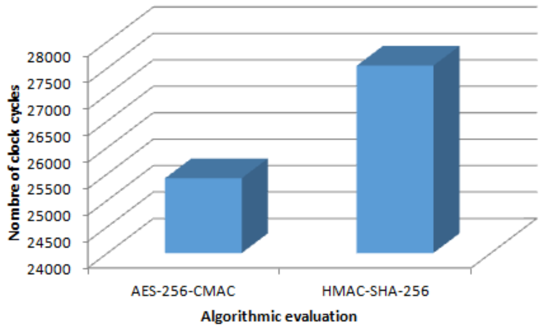

Fig. 8. Comparison between HMAC_SHA_256 and AES_256_CMAC.

VI. CONCLUSIONS

To defend against vehicle attacks, many approaches have been proposed in the literature. However, the greater part did not address real-time requirements in CAN bus communication. In this paper, effects of implementing cryptographic approaches have been investigated by proposing a performance analysis methodology of cryptographic algorithm. In this paper, after describing the fundamental idea, a description of the proposed system was given. The advantage of the presented method is that addresses safety and security of CAN bus by calculating the performances of cryptographic algorithm.

In this manuscript, it has been proved that the CAN network has limitations and it will not be able to meet requirements. Car manufacturers and researchers are invited to turn their attention to the recent protocols such as: Ethernet and VANET. In fact, despite being in progress, they will be very efficient in the future. Future work can include these new protocols to design a secure automotive network in the presence of recent communication interfaces.

REFERENCES

[1] R.B.GMBH, Bosch Automotive Electrics and Automotive Electronics, 5 ed. Bosch Professional Automotive Information. Springer Vieweg, 2014.

[2] R. Currie, “Hacking the CAN Bus: Basic Manipulation of a Modern Automobile Through CAN Bus Reverse Engineering”, The SANS Institute, 2017.

[3] Introduction to the Controller Area Network (CAN), TEXAS INTRUMENTS, Application Report, SLOA101B–August 2002–Revised May 2016, 2016.

[4] K. Koscher, A. Czeskis, et al., Experimental Security Analysis of a Modern Automobile, IEEE Symposium on Security and Privacy, 2010.

[5] S. Checkoway, D. McCoy, et al., Comprehensive experimental analyses of automotive attack surfaces, Proc.20th USENIX Security, San Francisco, CA, 2011.

[6] C. Miller and C. Valasek, Adventures in automotive networks and control units. Last Accessed from http://illmatics. com/-car_ hacking. pdf on, 2013. (Cite en pages 37, 38, 76 et 116).

[7] I. Studnia, E. Alata, et al., A language-based intrusion detection approach for automotive embedded networks, The 21st IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2015), Nov 2014, Zhangjiajie, China.

[8] M. J. Kang and J. W. Kang, “A novel intrusion detection method using deep neural network for in-vehicle network security,” in 2016 IEEE 83rd Vehicular Technology Conference (VTC Spring), May 2016, pp.

[9] Song, H. M., Kim, H. R., & Kim, H. K. (2016). Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. 2016 International Conference on Information Networking (ICOIN). doi:10.1109/icoin.2016.7427089.

[10] S. Woo, H. J. Jo, and D. H. Lee, “A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN,” "In IEEE Transactions On Intelligent Transportation Systems", 2014.

[11] S. N¨urnberger and C. Rossow, “– vatiCAN – vetted, authenticated CAN bus,” in Cryptographic Hardware and Embedded Systems: 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016.

[12] J.V. Bulck, VulCAN: Efficient Component Authentication and Software Isolation for Automotive Control Networks, ACSAC’17, December 2017, San Juan, Puerto Rico, USA.

[13] P. Mundhenk, A. Paverd, et al., Security in Automotive Networks: Lightweight Authentication and Authorization, ACM Trans. Design Automation of Electronic Systems, vol. 22, no. 2, 2017.

[14] M. Gmiden, M.H. Gmiden,H. Trabelsi, An intrusion detection method for securing in-vehicle CAN bus, Conference: Conference: 2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), IEEE, 2016.