A secure protocol for session keys establishment between ECUs in the CAN bus

Conference Paper · November 2017

DOI: 10.1109/WINCOM.2017.8238149

CITATIONS
22

READS
2,146

4 authors, including:

Younes El Hajjaji El Idrissi
Mohammed V University of Rabat
8 PUBLICATIONS **70** CITATIONS
SEE PROFILE

Noureddine Zahid
Mohammed V University of Rabat
42 PUBLICATIONS **515** CITATIONS
SEE PROFILE

A secure protocol for session keys establishment between ECUs in the CAN bus

Samir Fassak, Younes El Hajjaji El Idrissi, Noureddine Zahid and Mohamed Jedra
Laboratory of Conception and Systems, Faculty of Sciences
Avenue Ibn Batouta, B.P.1014, Rabat, Morocco
{samirfostok,youneselhajjaji,zahid.prof.08,jedra.master.iase}@gmail.com

***Abstract*—This paper presents a secure protocol for authenticating Electronic Control Units (ECUs) in the Controller Area Network (CAN) bus and establishing session keys between them. These keys are used to introduce message source authentication for the CAN network. Our method is based on the elliptic curve cryptography (ECC) which is more suitable for embedded systems where computational power and memory are very reduced. The performance evaluation of the proposed authentication method shows better results in comparison to the existing methods in term of bus load. The security of the proposed mechanism is validated using AVISPA tool which is very efficient for finding potential attacks automatically in cryptographic protocols.**

***Index Terms*—ECU, CAN, AVISPA, ECC, Automotive security, Multicast authentication, Embedded communication networks**

I. Introduction

The connectivity interfaces of modern cars are increasing more and more, making the on-board network, especially the CAN bus, vulnerable to external attacks. In fact, the United States Department of Transportation (USDoT) [1] is working on the implementation of a wireless technology that will enable vehicles to communicate with each other in V2V (Vehicle-to-Vehicle) mode, and also communicate with the elements of the road infrastructure like traffic signals, road signs, etc (V2I : Vehicle-to-Infrastructure). The objective of these technologies is to make the transport system more cooperative and intelligent (C-ITS), which will allow the exchange of safety messages between vehicles. That will certainly contribute in reducing accidents and consequently save lives. In addition, cars are now equipped with several types of Internet connectivity (WiFi, 3G, 4G, etc), and this provides good services to customers and OEMs (Original Equipment Manufacturers) such as upgrading ECUs across the cloud. These connected cars provide a better user experience, but they are based on one common protocol: the Controller Area Network (CAN).

The importance of the CAN bus can be seen in Fig. 1. In the past, manufacturers were connecting ECUs using a point-to-point cabling system. When embedded electronics have been developed in automobiles, the number of ECUs has increased, and as a result the connecting wires have became bulky and costly. For this reason, point-to-point connections have been replaced by a single serial bus linking all stations. This is accomplished by adding specific hardware (CAN Controller) to each ECU, which provides the protocol for transmitting and receiving information via the bus.

978-1-5386-2123-3/17/$31.00 © 2017 IEEE

The CAN bus was developed by the the german manufacturer Robert Bosch GmbH [2] in the 1980s, as a solution for embedded systems in general, and the automotive industry in particular. It is a multi-master serial communication network enabling the exchange of short messages, such as the temperature value, in broadcast mode, with high reliability and a rate that can reach 1 Mbps to a length of 40 meters.

The bus is in the form of a pair of wires (twisted pair for example) connected on resistors of 120 ohm or capacitors of 30 pF resistant to perturbations, as shown in Fig. 2. These two wires are denoted by CAN_L (low) and CAN_H (hight), and the logic states (0 or 1) are coded by the potential difference between the two levels, taking into account the perturbations.

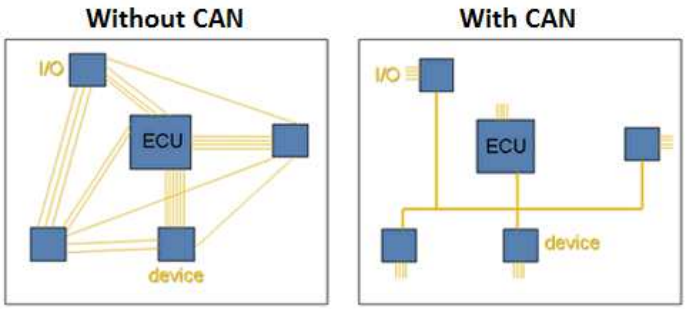

Fig. 1. Role of CAN in reducing cabling

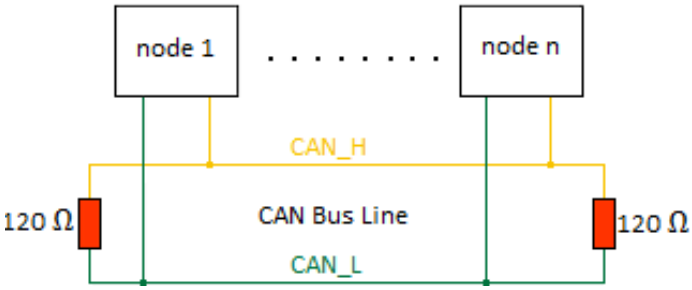

Fig. 2. A generic wiring diagram of a CAN Bus

Messages of CAN (frames) that are exchanged between nodes via the bus are composed of three primary fields (Fig. 3): the frame identifier (ID), the control field or Data Lenght

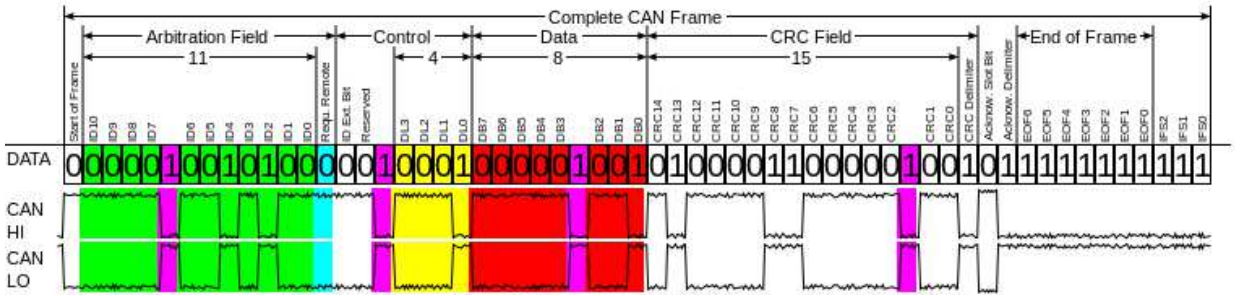

Fig. 3. The fields of a sample CAN message

Code (DLC) and the data to be transmitted. The ID contains typically 11 bits and is used to determine priorities for medium access to prevent collisions between messages on the bus. The lower the value of the frame ID, the higher its priority. The DLC field indicates the number of bytes of the data field of the frame. Its value is generally between 0 and 8. Finally, the data field is the body of the frame, it allows to transmit information of size not exceeding 64 bits (8 bytes).

The CAN bus load τ is defined as the utilization rate of the maximum bandwidth B available on the bus. It depends on the rate B, on the length of the messages exchanged between the nodes and on their frequency exchange. Considering a scenario that assumes that messages of length l bits exchange f times per second on a bus CAN of maximum bit rate B, we can write:

$$\tau = \frac{l \times f}{B} \tag{1}$$

With B expressed in bit/s, and τ generally a percentage (%). The CAN bus load must never exceed 70% in order not to lose the transmitted data.

There is no built-in security in the CAN protocol. It suffer from many vulnerabilities; the first obvious is related to the fact that messages are broadcasted to all nodes. Once an attacker has access to the bus he can read, log and analyse messages exchanged between nodes. The second vulnerability is due to the absence of messages authentication. It is impossible for a node to verify the sender of a received message.Thus, an attacker can send any message he likes. Therefore, the CAN bus presents a significant security risk and the systems built on it are exposed to attacks.

Koscher et al. [3] and Checkoway et al. [4] were the first to detect and analyze security flaws in modern vehicles. They have showed that ECUs could be attacked and reprogrammed directly, obtaining pre-programmed security keys from the automotive tuning community. Most of these attacks have been made with direct links to the vehicle, but in [3] attacks via external interfaces such as the integrated telematics unit have also been reported. More recently, C. Miller and C. Valasek [5] published a technical report in which they explain their attacks on the Jeep Cherokee 2014 car. They have used the technique of reverse engineering to connect to the WiFi hotspot of the Jeep and therefore gain access to the CAN bus and perform critical actions such as shutting down the car engine, controlling the brake system, and so on.

II. Related work

The security of vehicle networks is becoming an important issue [6]. Therefore, many solutions has been proposed to solve the security issues in the CAN bus. In this section we present some of these proposals and we discuss their shortcomings.

King et al. [7] have suggested using HMAC algorithm [8] and timestamp to perform authentication of messages on the CAN bus. HMAC is used to authenticate the message and the timestamp allows the system to prevent DoS and replay attacks. The weakness of this solution lies in the fact that it does not support mechanisms to dynamically generate keys used to calculate MACs.

Ueda et al. [9] have proposed centralized authentication system in CAN with improved CAN controller. This solution is based on a monitoring node which authenticates each ECU and verifies the message authentication codes (MACs) assigned to each CAN message sent. Once an unauthorized message is detected the central node destroys it by sending an error frame. However, the protocol requires hardware modification to be done in the central node which leads to extra cost for manufacturers and consumers.

In [10] Farag has presented his CANTrack solution for the CAN bus. CANTrack improve the security of the CAN protocol by using an algorithm that encrypts the payload data using a symmetric key that is being dynamically changed using synchronized key generators in all nodes. The encrypted message is unique at any instance of time which prevent the CAN bus from replay attacks.

Elend et al. [11] have proposed enhancing the CAN bus security without involving cryptography which makes their solution cost-effective. Their proposition is a kind of a distributed intrusion detection system (IDS), working like a firewall that can be implemented in the transceiver.

III. ELLIPTIC CURVE CRYPTOGRAPHY (ECC)

Cryptography with elliptic curves is an approach to public key cryptography (PKC). It was proposed independently by Victor S. Miller [12] and N. Koblitz [13] respectively in 1985 and 1987. It combines a set of methods and techniques that

make data safer by consuming fewer resources. It has recently attracted increasing attention from researchers worldwide, especially those working in the field of embedded systems in which electronic devices have very limited computing power and memory.

A. Definition of the elliptic curve

An elliptic curve is a set of elements, called points, defined on a finite field K, that is to say on a finite set of elements where it is possible to make additions, subtractions, and multiplications. In the domain of cryptography, we use the elliptic curve $E(F_q)$ which is defined on a finite field F_q of order q, with q a prime number very large and $q > 3$. This type of curve is defined by the following equation:

$$y^2 = x^3 + ax + b \pmod{q} \quad (2)$$

Where $a, b \in F_q$, and the discriminant of the curve $\Delta = 4a^3 + 27b^2 \neq 0 \pmod{q}$. This is the curve that will be used in this paper.

B. Computational problems

1) Elliptic Curve Discrete Logarithm Problem (ECDLP): The level of security of the ECC technique depends strongly on the difficulty of solving the discrete logarithm problem on elliptic curves that can be defined as follows: given two points P and Q belonging to $E(F_q)$, such as $Q = aP$ and $a \in F_q^*$. It is computationally difficult to compute a from P and Q.

2) Elliptic Curve Diffie Hellman Problem (ECDHP): This problem is very useful to prevent attackers from inferring a key established by a protocol using ECC. We define it as follows: given aP, bP and P three points belonging to $E(F_q)$, with $a, b \in F_q^*$ integers. It is computationally difficult to deduce the point abP on $E(F_q)$.

IV. The proposed solution for the CAN bus

Our proposal for authentication for the CAN bus is in the form of a hybrid cryptographic system combining two phases. The first phase uses asymmetric cryptography to establish a session key $K_{S_{i,j}}$ between a sending ECU_i and each receiver ECU_j on the CAN bus. The second phase exploits symmetric cryptography. It consists of associating MACs with the messages transmitted by the sender node using the established keys and the HMAC algorithm. The details of the two phases are described in the following paragraphs.

A. Key establishment phase

The manufacturer of the CAN system must install the parameters required for the proper operation of the protocol. For this purpose, it selects an elliptic curve $E(F_q)$ defined by 2 previously seen, on a finite field F_q of order q, generated by a base point Q. The manufacturer must then install for each ECU_i a pair of public/private keys (U_i, d_i), where U_i=$d_i \times Q$ and "$\times$" denotes the scalar multiplication operation on the elliptic curve $E(F_q)$. The public key U_i is stored in all the other ECUs of the CAN bus. An example of installing the keys in the ECU_1 is shown in Table. I. The manufacturer must also choose a secure hashing function H to be used by the ECUs during the protocol.

TABLE I. TABLE OF KEYS INSTALLED IN ECU_1

IDs	Keys
ID_1	(U_1, D_1)
ID_2	U_2
ID_3	U_3
ID_4	U_4
ID_5	U_5

After having installed all the required parameters including the elliptic curve, the keys (U_i, d_i) and the hash function H. An ECU that wants to communicate via an authenticated channel with the rest of the ECUs must establish a session key $K_{S_{i,j}}$ with each ECU_j. Then we propose the exchange of messages shown in Fig. 4 to generate this key. Our method is carried out by following steps:

STEP 1: ECU_i performs the following calculations:
(a) Choose $r_i \in F_q^*$, calculate $R_i = r_i \times U_i$ and $AUTH_i = H(ID_i||R_i||d_i \times U_j)$.
(b) Send $(ID_i||R_i||AUTH_i)$ to ECU_j.

STEP 2: Upon receiving the message $(ID_i||R_i||AUTH_i)$, ECU_j executes the following:
(a) Calculate $AUTH_i^{'} = H(ID_i||R_i||d_j \times U_i)$.
(b) ECU_j compares $AUTH_i^{'}$ with $AUTH_i$, if they match then the authentication of ECU_i by ECU_j has succeeded. Then ECU_j chooses $r_j \in F_q^*$ and computes:

$$\begin{aligned} K_j &= r_j \times d_j \times R_i \\ &= r_j \times d_j \times r_i \times U_i \\ &= r_i \times r_j \times d_i \times d_j \times Q \end{aligned} \quad (3)$$

STEP 3: After having authenticated ECU_i, ECU_j performs:
(a) Calculate $R_j = r_j \times U_j$ and $AUTH_j = H(ID_j||R_j||d_j \times U_i)$ and generate the session key:

$$K_{S_{i,j}} = H(ID_i||ID_j||R_i||R_j||AUTH_i||AUTH_j||K_j) \quad (4)$$

(b) Send $(ID_j||R_j||AUTH_j)$ to ECU_i.

STEP 4: On receiving the message $(ID_j||R_j||AUTH_j)$, ECU_i executes the following:
(a) Calculate $AUTH_j^{'} = H(ID_j||R_j||d_i \times U_j)$.
(b) ECU_i compares $AUTH_j^{'}$ with $AUTH_j$, if they match then the authentication of ECU_j by ECU_i has succeeded. Then ECU_i calculates:

$$\begin{aligned} K_i &= r_i \times d_i \times R_j \\ &= r_i \times d_i \times r_j \times U_j \\ &= r_i \times r_j \times d_i \times d_j \times Q \end{aligned} \quad (5)$$

and it generates the same session key:

$$K_{S_{i,j}} = H(ID_i||ID_j||R_i||R_j||AUTH_i||AUTH_j||K_i) \quad (6)$$

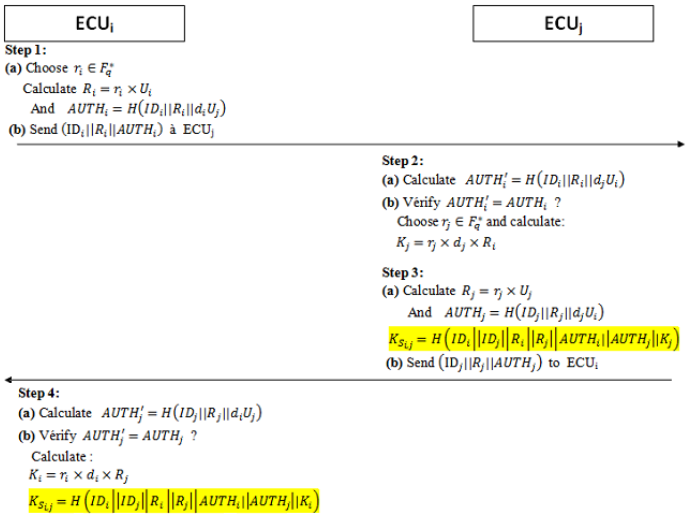

Fig. 4. The proposed key establishment protocol

B. Messages authentication phase

A sender ECU_i calculates the MACs only for the nodes that will accept the data after CAN filtering, i.e it does not need to compute a MAC for certain nodes and increase the bandwidth, whereas these nodes will reject the message because their CAN filters do not allow to accept the message. This objective is achieved by maintaining for each node in its flash memory a table TAB_ID which associates with each ID of a message: the ID of the sending node, the number of receiving nodes and the list of IDs of its receivers. The structure of this table is illustrated in Table. II.

TABLE II. EXAMPLE OF THE STRUCTURE OF TAB_{ID}

IDs	sender's ID	Number of receivers	List of IDs of receivers
k	e_i	n_k	$r_{k,1}, r_{k,2}, ..., r_{k,n_k}$
l	e_j	n_l	$r_{l,1}, r_{l,2}, ..., r_{l,n_l}$
m	e_u	n_m	$r_{m,1}, r_{m,2}, ..., r_{m,n_m}$

When a sending node ECU_i wants to transmit on the CAN bus a message M_k having an ID equal to k, it will proceed as follows:
1) Access the TAB_{ID} table to retrieve n_k and the list of IDs of receivers.
2) Read the time of the internal clock T_i.
3) For each receiver $r_{k,j}$ compute a MAC: $A_{k,j} = MAC_{K_{S_{i,r_{k,j}}}}(M_k||T_i)$. With $K_{S_{i,r_{k,j}}}$ is the established key between ECU_i and $ECU_{r_{k,j}}$.
4) Send $M_k||T_i||[A_{k,1}]_b||[A_{k,2}]_b||...||[A_{k,n_k}]_b$. With $[A]_b$ denotes the truncation of the MAC to a b bits. In our case $b = 8$.

A receiver node ECU_j of the message M_k sent by ECU_i will follow these steps in order to validate the received data:
1) Receive $M_k||T_i||[A_{k,1}]_b||[A_{k,2}]_b||...||[A_{k,n_k}]_b$.
2) Obtain the transmission time T_i and the local time T_j.
3) Obtain the identity i of the sender node of M_k.
4) Compute $A = MAC_{K_{S_{i,r_{k,j}}}}(M_k||T_i)$.
5) Accept M_k if and only if $[A]_b = [A_{k,j}]_b$ and $|T_i - T_j| \leq \Delta t$, where Δt is the time interval tolerated by the protocol.

C. Performance evaluation of the authentication method

Our authentication mechanism associates a MAC value with the message sent for each receiver node. We aim to study the impact of this authentication data on the CAN bus load. For this, we set some parameters. The nodes of the CAN bus are supposed sending messages with a frequency of 10 messages per second. Messages are supposed with data length equal to $l = 32\ bits$. The CAN is supposed a standard version (2.0A) (ID=11 bits). The maximum bandwidth available on the CAN bus is $B = 1Mbps$.

We consider the 3 following scenarios:

- Scenario 1: The nodes exchange messages without any authentication data.
- Scenario 2: The nodes exchange messages by associating a MAC value for each node connected to the CAN bus.
- Scenario 3: The nodes exchange messages between them using our authentication method based on CAN filtering.

In order to calculate the load of the CAN bus we used the tool "CAN bus load calculator" (Fig. 5) developed by OptimumG. OptimumG [14] is a company that offers solutions for vehicle dynamics, data acquisition and solutions for the international racing car industry. The tool is widely used when installing applications that implement the CAN bus to determine the maximum bandwidth consumption.

In order to study the impact of the 3 scenarios mentioned above on the CAN bus bandwidth. We have varied the total number of nodes n in the interval [0,70] and for each value of n we have calculated the values of the bus loads τ_{SC1}, τ_{SC2} and τ_{SC3} corresponding to scenarios 1, 2 and 3 respectively. Using the same "CAN Bus Load Calculator" calculation tool, the results are represented in Fig. 6.

According to Fig. 6 we can see that our method allows to have a load value in the recommended margins up to 70 nodes. While the scenario 2 exceeds the limit load value recommended (70 %) at 60 nodes and reachs a critical bus load (98.2 %) at 70 nodes. So it is clear that our authentication technique reduces considerably the load of the CAN bus.

D. Security validation of the proposed protocol using AVISPA

We have defined our protocol in an initiator model (ecu_i) and responder (ecu_j) and the model is expressed in HLPSL [15] under AVISPA software [16]. Fig. 7 and Fig. 8 illustrate the two roles of the protocol ecu_i and ecu_j. We used the instructions "Request" and "witness" to test the validity of the mutual authentication property between the two agents. The line request(ECUi, ECUj, ni, AUTHi) means that the entity ECUi is authenticated by the entity ECUj based on the value AUTHi, and ni is used to refer this property when specifying security goals in the goal section. The declaration secret(Ks, sec_k1, ECUi, ECUj) indicates that the established session key Ks must be kept secret between ECUi and ECUj.

To ensure that our HLPSL specification is correct, we used the SPAN tool to generate the Message Sequence Chart (MSC)

CAN Bus Load Calculator

www.optimumg.com

Inputs		
Data rate	1000	kBAUD (1 kBAUD = 1 kbit/s)
Protocol	CAN 2.0A	2.0A = 11 bit ID, 2.0B = 29 bit ID
Frequency	600	Hz

Description	Data Length	Total Message Length
Message 1	32	89
Message 2	0	0
Message 3	0	0
Message 4	0	0
Message 5	0	0
Message 6	0	0
Message 7	0	0
Message 8	0	0
Message 9	0	0
Message 10	0	0
Message 11	0	0
Message 12	0	0
Total data/cycle		89

Outputs		
Baud	1000	kBAUD
Address Bits	12	12 or 32
Start/Stop/Etc Bits	32	
Frequency	600	Hz
Other Message Bits		
Start of Frame		1
Arbitration Field		12
Control Field		6
CRC Field		16
Acknowledge Field		2
End of Frame		7
Bits before stuff bit		5
%Max BUS load		5,3%

Fig. 5. Screenshot of the tool "CAN Bus Load Calculator"

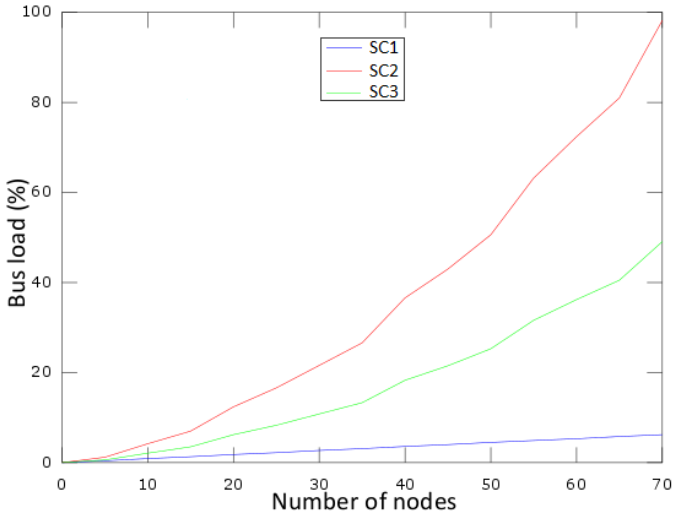

Fig. 6. The variation of the bus load τ with n

corresponding to the HLPSL modeling. As we can see in Fig. 9 the agents ecu_i and ecu_j exchange a total of two messages between them, which confirms that the HLPSL specification is not erroneous.

The security properties specified in our protocol, namely mutual authentication and the secret of the session key Ks, have been tested using the OFMC and CL-AtSe tools. All tests have been passed successfully and no attacks or vulnerabilities have been found, which confirms the security of the proposed key exchange protocol. Fig. 10 shows the messages returned by the OFMC and CL-AtSe verification tools.

```
role ecu_i(
          ECUi, ECUj              : agent,
          SND, RCV                : channel(dy)
          H                       : hash_func,
          Di, Q                   : symmetric_key,
          IDi, IDj                : text
         )

played_by ECUi
def=
   local
      State                      : nat,
      Ui, Uj                     : public_key,
      Ks                         : symmetric_key,
      Ki, Kj                     : symmetric_key,
      Ri, Rj                     : text,
      Ri_2, Rj_2, AUTHi, AUTHj   : message

   const
         sec_k1                  : protocol_id

   init State  := 0
   transition
    1.State   = 0 /\RCV(start)=|>
      State' := 2 /\Ri'     := new()
                  /\Ri_2'   := H(Ri, H(Di, Q))
                  /\AUTHi' := H(IDi, H(Ri_2, H(Di, Uj)))
                  /\SND(IDi.Ri_2'.AUTHi')
                  /\witness(ECUi, ECUj, nj, AUTHj)

   2.State =2/\RCV(IDj.Rj_2'.AUTHj')/\AUTHj'=H(IDj,H(Rj_2',H(Di, Uj)))=|>
      State' :=4 /\Ki':=H(H(Ri, Di), Rj_2')
                 /\Ks':=H(IDi,H(IDj,H(Ri_2,H(Rj_2',H(AUTHi,H(AUTHj',Ki'))))))
                 /\secret(Ks', sec_k1, {ECUi, ECUj})
                 /\request(ECUi, ECUj, ni, AUTHi)
end role
```

Fig. 7. Role ecu_i in HLPSL

```
role ecu_j(
          ECUj, ECUi              : agent,
          SND, RCV                : channel(dy)
          H                       : hash_func,
          Dj, Q                   : symmetric_key,
          IDi, IDj                : text
         )

played_by ECUj
def=

  local
    State                   : nat,
    Ui, Uj                  : public_key,
    Ks                      : symmetric_key,
    Ki, Kj                  : symmetric_key,
    Ri, Rj                  : text,
    Rj_2,Ri_2, AUTHj, AUTHi : message

    const
       sec_k2               : protocol_id

    init State:= 1

    transition
     1.State = 1 /\ RCV(IDi.Ri_2'.AUTHi')/\AUTHi'=H(IDi,H(Ri_2, H(Dj, Ui)))=|>
       State' := 3  /\witness(ECUj, ECUi, ni, AUTHi')
                    /\Rj'   :=new()
                    /\Rj_2' :=H(Rj', H(Dj, Q))
                    /\Kj'   :=H(H(Rj', Dj), Ri_2')
                    /\AUTHj':=H(IDj, H(Rj_2', H(Dj, Ui)))
                    /\Ks'   :=H(IDi,H(IDj,H(Ri_2',H(Rj_2',H(AUTHi',H(AUTHj',Kj'))))))
                    /\SND(IDj.Rj_2'.AUTHj')
                    /\secret(Ks', sec_k2, {ECUj, ECUi})
                    /\request(ECUj, ECUi, nj, AUTHj')
end role
```

Fig. 8. Role ecu_j in HLPSL

V. CONCLUSION

This paper aims at presenting a protocol for authenticating ECUs and establishing session keys between them on the CAN bus. These keys are then used to authenticate messages exchanged between the nodes. The proposed key distribution method satisfies the expected security properties that are the secret of the session key and the mutual authentication between the ECUs. We have evaluated the performance of our authentication method by calculating the load introduced

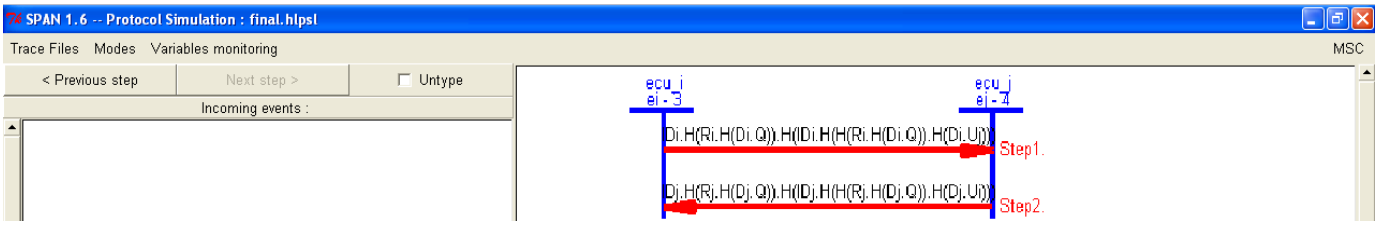

Fig. 9. Animation of the proposed protocol with SPAN

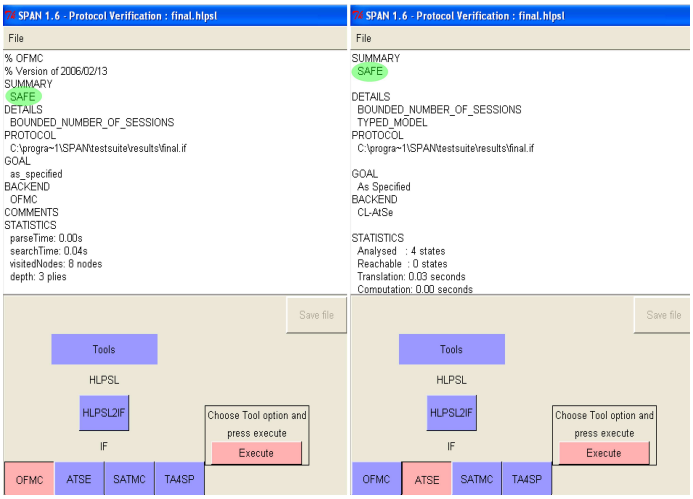

Fig. 10. Messages returned by OFMC and CL-ATse tools

by the proposed protocol. The load caused by the proposed technique is considerably reduced in comparison with that introduced by the existing authentication methods which associates MACs for all the nodes of the bus. In order to validate the security of the key distribution protocol, we have performed simulations with the AVISPA software. The results of the tests with the various AVISPA tools indicate that the protocol satisfies the desired security properties.

As a perspective for this work, we are working on other methods for authentication of messages on the CAN bus, as well as encryption to protect CAN systems from attacks.

REFERENCES

[1] "United States Department of Transportation." [Online]. Available: http://www.transportation.gov/
[2] "Robert Bosch GmbH." [Online]. Available: http://www.bosch.com/
[3] K. Koscher *et al.*, "Experimental security analysis of a modern automobile," in *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 2010, pp. 447–462.
[4] S. Checkoway *et al.*, "Comprehensive experimental analyses of automotive attack surfaces." in *USENIX Security Symposium*. San Francisco, 2011.
[5] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, 2015.
[6] P. Mundhenk *et al.*, "Security in automotive networks: lightweight authentication and authorization," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 22, no. 2, p. 25, 2017.
[7] Z. King and S. Yu, "Investigating and securing communications in the controller area network (can)," in *Computing, Networking and Communications (ICNC), 2017 International Conference on*. IEEE, 2017, pp. 814–818.
[8] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in *Crypto*, vol. 96. Springer, 1996, pp. 1–15.
[9] H. Ueda *et al.*, "Security authentication system for in-vehicle network," *SEI Technical Review*, no. 81, 2015.
[10] W. A. Farag, "Cantrack: Enhancing automotive can bus security using intuitive encryption algorithms," in *Modeling, Simulation, and Applied Optimization (ICMSAO), 2017 7th International Conference on*. IEEE, 2017, pp. 1–5.
[11] B. Elend and T. Adamson, "Cyber security enhancing CAN transceivers." [Online]. Available: https://www.can-cia.de/fileadmin/resources/documents/proceedings/2017_elend.pdf
[12] V. S. Miller, "Use of elliptic curves in cryptography," in *Conference on the Theory and Application of Cryptographic Techniques*. Springer, 1985, pp. 417–426.
[13] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.
[14] "OptimumG - Vehicle Dynamics Solutions." [Online]. Available: http://www.optimumg.com/
[15] D. von Oheimb, "The high-level protocol specification language hlpsl developed in the eu project avispa," in *Proceedings of APPSEM 2005 workshop*, 2005, pp. 1–17.
[16] [Online]. Available: http://www.avispa-project.org/