CAN-FD-Sec: Improving Security of CAN-FD Protocol

Chapter · January 2019

DOI: 10.1007/978-3-030-16874-2_6

Megha Agrawal
Indraprastha Institute of Information Technology

Jianying Zhou
SUTD

CAN-FD-Sec: Improving Security of CAN-FD Protocol

Megha Agrawal[1], Tianxiang Huang[2], Jianying Zhou[3], and Donghoon Chang[1]

[1] Indraprastha Institute of Information Technology, Delhi, India
[2] Chongqing University of Posts and Telecommunications, China
[3] Singapore University of Technology and Design, Singapore

Abstract. A modern vehicle consists of more than 70 Electronic Control Unit (ECUs) which are responsible for controlling one or more subsystems in the vehicle. These ECUs are interconnected through a Controller Area Network (CAN) bus, which suffers from some limitations of data payload size, bandwidth, and the security issues. Therefore, to overcome the CAN bus limitations, CAN-FD (CAN with Flexible Data) has been introduced. CAN-FD has advantages over the CAN in terms of data payload size and the bandwidth. Still, security issues have not been considered in the design of CAN-FD. All those attacks that are possible to CAN bus are also applicable on CAN-FD. In 2016, Woo et. al proposed a security architecture for in-vehicle CAN-FD. They used an ISO 26262 standard that defines the safety level to determine the security requirements for each ECU, based on that they provided encryption, authentication, both or no security to each ECU. In this paper, we propose a new security architecture for the communication between ECUs on different channels through gateway ECU (GECU). Our experimental results also demonstrate that using an authenticated encryption scheme has better performance than applying individual primitives for encryption and authentication.

Keywords: Controller Area Network (CAN), CAN-FD (CAN with flexible data rate), Security of In-vehicle Network

1 Introduction

Over the past 30 years, with the emergence of vehicle information and communication technology (ICT), several electronic communication devices have been installed in the vehicles. These electronic communication devices, known as Electronic Control Units (ECU) are responsible for controlling the one or various subsystems of the vehicle including the break, doors, tyre pressure and so on. Nowadays, a typical vehicle consists of more than 70 ECUs [8]. These ECUs are generally on a single chip, using an 8-bit microcontroller with around 100 bytes of RAM, 32 kB of ROM and a few I/O pins to connect to sensors, actuators, and a network interface [16]. Data exchange among these ECUs is facilitated through various communication networks such as Local Interconnect network (LIN), Controller Area Network (CAN), Byteflight [6], and FlexRay [9]. Among all these communication networks, CAN [7] has been standardized for all the communications among various ECUs.

Controller Area Network (CAN) is a serial bus based communication protocol. It was introduced by Robert Bosch GmBH in 1983 and standardized in 1994 under the ISO 11898-1 [1]. All the ECUs in the vehicle are interconnected through CAN bus. There is no security consideration in the design of CAN bus except the standard CRC-15. All the messages between ECUs are transmitted in plaintext without any security feature. An adversary can eavesdrop all communications between ECUs and later can launch a replay attack [12] or he can modify the existing message and inject into the system to alter the usual vehicle behaviour. These kind of attacks can result in some catastrophic consequences. Various attacks on the CAN bus security are shown in [17],[13],[21],[12], [15], [14]. All these attacks become possible because of no implementation of the essential security features: confidentiality, authentication, and integrity. Confidentiality of the data can be achieved using encryption whereas authentication and integrity can be incorporated by using MAC algorithm (Message Authentication Code). One of the other disadvantages of the CAN is the small data packet size. CAN can support data payload of up to 8 bytes only. Due to the low payload size, it is impossible to use MAC algorithm as it adds a tag to the data, which increases the data payload size at least twice.

To match the modern vehicle requirements and to overcome the current limitation of CAN, Bosch developed a new protocol in 2011 known as CAN-FD (CAN with flexible data) [11]. CAN-FD has an almost similar structure as CAN with some additional advantages. Following are the advantages of CAN-FD over CAN [2]:

- CAN-FD supports data payload of up to 64 bytes.
- It can support bandwidth up to 8 Mbps whereas CAN can support up to 1 Mbps only.
- It has lower latency and better real-time performance.
- CAN-FD is compatible with CAN and can support existing software and applications with the minimum changes.

CAN-FD is supposed to replace CAN gradually by 2020 [4]. However CAN-FD also suffers from the same security issues as CAN. Here data is transmitted in the plaintext without any security. Hence all the existing attacks on CAN are also applicable to CAN-FD. In [21], authors have shown a practical wireless attack using a real vehicle in a connected car environment where driver's smartphone is connected to the in-vehicle CAN. This attack assumes that a driver downloads the malicious self-diagnostic application that has been uploaded by an adversary. Once it has been downloaded to the driver's smartphone, the adversary can control the driver's smartphone and can inject malicious CAN data frame that may cause an abnormal behaviour leading an accident or any devastating scenario. The same attack is also applicable on CAN-FD as it does not have any security features as well. Later in [20] , authors proposed a security architecture for CAN-FD to resist against these kind of attacks. In that paper, it considered a characteristics of ISO 26262 Automotive Safety Integrity Level and defined the security requirements for ECUs. More details of their work is explained in the next section. We will demonstrate that the security architecture for CAN-FD proposed in [20] is not practical and has high overhead.

In this paper, we proposed an improvement over the existing security architecture for the CAN-FD bus. The main contributions of this paper are:

- Proposed a group-based approach to the communication among different ECUs. Groups are divided based on the existing channels.
- Modified the existing key management protocol to satisfy the group communication.

– Provided an experimental analysis by replacing the individual encryption and authentication scheme with a single primitive called authenticated encryption to provide confidentiality and authenticity.

2 Background and Related Work

A typical in-vehicle network consists of several ECUs responsible for controlling various subsystems. Controller Area Network (CAN) uses a serial bus communication to interconnect those ECUs in an in-vehicle network. It is a multicast message-oriented transport protocol which facilitates all in-vehicle data communication where each node (ECU) can act as a transmitter or receiver. An ECU that initiates a message is called a transmitter, which broadcasts the message to all other ECUs on the bus. All receiving ECUs read the message and decide if it is relevant to them. All these ECUs communicate with each other using a fixed length data packet over a CAN bus. It supports data payload of at most 8 bytes and data rate up to 1Mbps. CAN-FD is built upon CAN protocol and retains most of its characteristics. It provides better real-time performance and supports higher bandwidth up to 8Mbps. CAN-FD supports data payload of up to 64 bytes. CAN and CAN-FD data frames are shown in fig. 1. As shown in the figure, some additional bits have been added to CAN-FD in control field.

Communication In an in-vehicle network, ECUs are categorized into several subsystems or channels. All these channels are connected through a gateway ECU which is supposed to be more powerful than the usual ECUs. All ECUs on one channel form the internal subnetwork. The communication between these channels is facilitated through gateway ECU. If an ECU on one channel wants to send messages to another ECU on a different channel, it first sends to GECU. Then GECU broadcasts the messages to another channel.

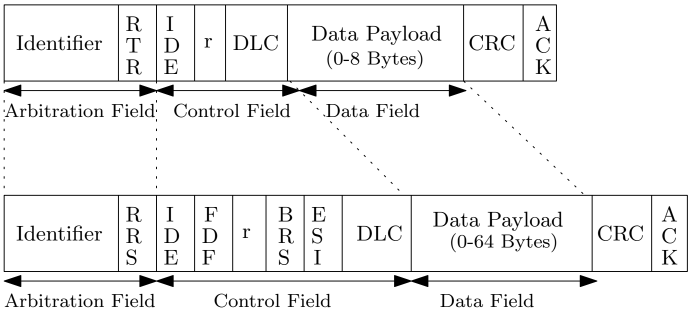


Fig. 1. CAN and CAN-FD Data Frame

Related work The area of securing CAN-based communication has drawn lots of attentions. As a result, various solutions for providing secure communication in CAN networks have been proposed. Vecure [19] and LeiA [18] are two authentication protocols designed for CAN which rely on symmetrically shared keys and MACs for data authentication. LiBrA-CAN [10] provides authentication based on key sharing in groups of nodes. With the introduction of CAN-FD, more efficient transmission of bigger payloads is possible, resulting in development of new security mechanisms. In [20], authors have proposed a security architecture for in-vehicle CAN-FD. The idea is to use an ISO 26262 standard to define security requirements for each ECUs. Based on this they define the four automotive security level (from 0-3) and categorize each ECU under these four levels. The higher the level number, more security it requires. Details are given in table 1. If the ECU having ASL scale 0 wants to broadcasts a message, it just sends it to GECU without any security feature which further broadcast it. Others ECU belonging to ASL scale 1,2 and 3 uses authentication, encryption and both respectively. If these ECUs want to send messages, they encrypt or compute MAC or do both and send it to GECU. GECU then decrypt and verify the MAC. Based on the verification it further encrypts and computes MAC using keys shared between different ECUs and GECU and sends it to respective ECU.

The main issue with this approach is the excess overload on GECU while broadcasting encrypted packet as it needs to encrypt the packet with an individual key for each ECU. For instance, if GECU wants to transmit it to 10 ECU's, then it needs to perform ten encryptions as all the ECU's shared their secret key with GECU only. If the message requires authentication also, then GECU again have to compute individual tags for each ECU as it uses the separate key for encryption and authentication.

ASL	**Security requirement**	**Security provided**
0	No security	Inbuilt CRC
1	Data authentication	Authentication
2	Data confidentiality, Data Authentication	Encryption, Authentication
3	Data confidentiality, Data Authentication, External Access	Encryption, Authentication, Access control

Table 1. Defining Automotive Security Level

3 Security Requirements

In the previous sections, we mentioned the existing vulnerabilities in CAN-FD bus that attacker can exploit to launch an attack to cause damage to the vehicle or the driver. Following are the security requirements that must be followed by CAN-FD bus to work without any vulnerabilities.

- **Confidentiality**: As all the messages over CAN-FD bus are communicated in plaintext, an adversary can eavesdrop the valid communication and analyze the messages to plan an attack. To overcome this, all communication should happen in encrypted form so that only legitimate parties can access the original data. This can be achieved by using an existing block cipher like AES in a valid encryption mode.

- **Authenticity**: Receiving node on a CAN-FD bus identify the data frame based on the sender information. An adversary can eavesdrop the communication and later can replay a data frame by masquerading a valid sender. CAN-FD uses CRC sequence only to check for the error, which fails to detect this kind of attacks. Hence, authentication must be used to verify the identity of the sender and to prevent this kind of attacks. A cryptographic Message Authentication Code(MAC) algorithm can be used to achieve authentication.

A cryptographic primitive that achieves Confidentiality and Authenticity simultaneously in a single step is called an Authenticated Encryption(AE). Introduce by Bellare and Rogaway in [5], an AE scheme Π can be defined as set of 3 algorithms $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where $\mathcal{K}$ is a non-empty set of strings, $\mathcal{E}$ is a randomized encryption algorithm and $\mathcal{D}$ represents a deterministic decryption algorithm. Encryption algorithm $\mathcal{E}$ takes a key K, message M and associated data A (optional) as an input and generate a ciphertext tag pair (C, T). Decryption algorithm $\mathcal{D}$ takes a key K, A and (C, T) as an input and returns either M or *Invalid* based on the verification of the tag. There is an advantage of using AE over an individual cryptographic primitive for confidentiality and authenticity. Individual primitive requires a separate key for encryption and authentication and two passes over the message while authenticated encryption uses only one key and require only one pass which results in better performance.

In this paper, we use the AEGIS [23] authenticated encryption from the ongoing CAESAR [3] competition for the implementation and compare its performance against using individual primitives for encryption(AES) and authentication(CCM).

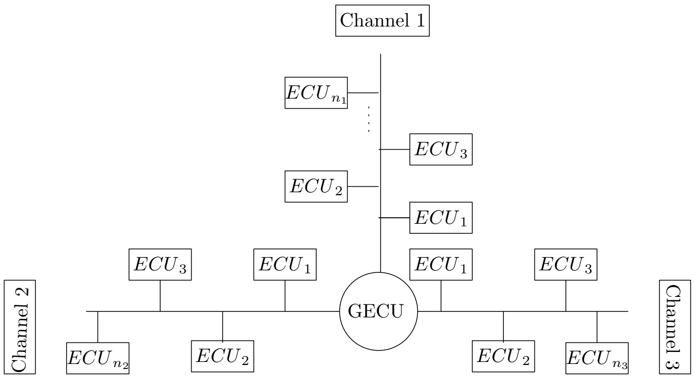


Fig. 2. CAN-FD Architecture

Notations	Meaning
GECU	Gateway ECU
i	i^{th} channel
j	j^{th} ECU
ECU_{ij}	Refers to j^{th} ECU on i^{th} channel
K_{ij}^1, K_{ij}^2	Preshared long term keys between $ECUij$ and GECU
$seed_{ij}^k$	Seed for $k^{thsession}$
sk_{ij}^k	Individual key for ECU_{ij} of k^{th} session
gk_i^k	Group key for i^{th} channel of k^{th} session
AE_K	Authenticated encryption using key K
KDF_K	Key derivation function using key K
M	Plaintext
(C, T)	Ciphertext Tag pair

Table 2. Notations

4 A Secure CAN-FD Protocol

In this work, we present a new security architecture for the CAN-FD. Our proposed solution works on the following assumptions:

- All the communication between different ECUs is done through Gateway ECU (GECU), which is supposed to have more computing power than other ECUs.
- All ECUs are preloaded with the two keys that they share with GECU. Out of which, one is used to authenticate each other and other is used for further session key generation.
- All ECUs have been divided into the fixed no of channels and the communication happens among these channels through gateway ECU (GECU).
  The notations we used in this paper are shown in 2.

4.1 Message Structure

Deployment of the proposed security approach will affect the structure of the message packets. Therefore, the data fields of different types of CAN-FD packets are processed in segments. The following rules need to be followed when adding a new feature to the protocol:

- The impact on non-encrypted data packets is as small as possible. If the protocol structure occupies more effective data segment length, the original communication architecture will be greatly affected, which will make it inconvenient for the

deployment of the program, and will also have an impact on the real-time communication performance.
- Easy maintenance. For the later program improvement, we must reserve a certain amount of space.
- Consistency. Applicable to multiple nodes in the network, both parties can effectively identify the communication.

The details of the message structure formulated in this paper is shown in fig. 3.

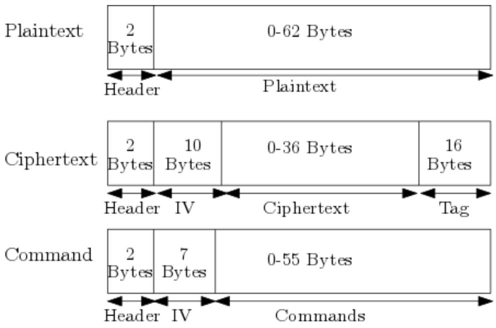


Fig. 3. Modified Data field for CAN-FD Packet

The CAN-FD network message is divided into three categories: plaintext, ciphertext, and command messages. The first two bytes of the data field out of 64 bytes are used as the header segment to distinguish among these 3 type of messages. At present, an only first byte is used where 0x55 represents the plaintext and 0xFF represents the ciphertext, and the second byte is reserved for the future use. For the plaintext message, an effective data payload can be up to 62 bytes. However, we restrict it up to 36 bytes as rest of the bytes we require for the tag and IV usage in the ciphertext. We used an authenticated encryption scheme to generate the ciphertext. Hence its valid data segment can be up to 36 bytes, followed by a 16-byte authentication tag and 10 byte IV(Initialization Vector). Command messages are used for key management. They are divided into address segments (used to indicate the receiving node) and content segments (used to transmit random numbers or authentication codes, etc.).

4.2 Proposed Security Architecture

The overall security procedure is divided into several steps. We will explain these steps in detail below.

Step 1: Initialization During the initialization phase, each ECU is loaded on their respective channel. A typical modern vehicle consists of following channels: powertrain, chassis, body, safety, and Infotainment serving different purposes. Details about the channels and ECUs under these channels are given in table 3.

Subsystems/Channels	**Powertrain**	**Chassis**	**Body**	**Telematic**	**safety**
Functions	Engine control, automatic transmission, hybrid control	Steering, brake, suspension	Instrumental panel, door, key, window	audio, navigation, traffic information	Pre crash safety
No of ECUs	3-6	6-10	14-30	4-12	11-12
Safety	High	High	Low	Low	Very High
ECUs example	ECM TCM	BCM SUM	DDM PDM	TBOX	ABS

Table 3. Characteristics of Various Channels on an in-vehicle network

Step 2: Key Loading As mentioned above each ECU in the vehicle is preloaded with the two keys K_{ij} and LK_{ij} which are shared with GECU. So, if there are n ECUs in the vehicle, GECU is loaded with total 2n keys (n no. of $K_{ij}'s$ and n $LK_{ij}'s$). These loading of the keys are done during the manufacturing of a vehicle or when some ECU need to be changed.

Step 3: Session-key Generation After a vehicle starts, each ECU performs a session key generation with GECU in a fixed order. This process is divided into 2 steps. In the first step, every ECU on different channels perform an individual session key generation with GECU. During the second step, when every ECU on the different channel have their session key, GECU generate a group key for each channel by hashing the individual session key for each ECU on that channel and then distribute this group key on that channel by encrypting it with the ECU's session key. Algorithms for individual session key generation and group key generation are shown in Algorithm 1 and 2 respectively.

Step 4: Authenticated Encryption Once the individual session key and the group key has generated, confidentiality and authenticity is provided to the data frames using Authenticated Encryption (AE) scheme. For example, if ECU_1 on channel 1 wants to send a message to the ECU's on channel 2, then ECU_1 will use the AE using their session key and send it to GECU. GECU will then verify the message, if it gets verified, then GECU will encrypt the message with channel 2 group key and forward it on that channel otherwise it will discard the message. Details are given in Algorithm 3.

Step 5: Key Update Individual and the group session key between ECUs and GECU are updated in the following scenarios :

- After a fixed predefined period say T.
- If any ECU leaves or joins the network.
- if any external device tries to connect.

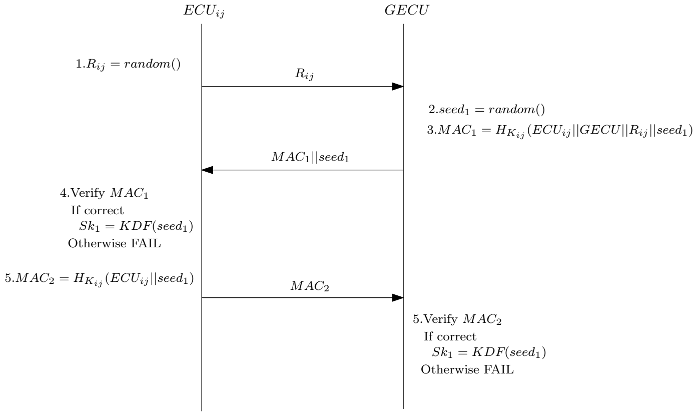


Fig. 4. Data Flow Diagram Showing Session Key Generation

ALGORITHM 1: Individual session key generation

```
for $1 \rightarrow i$ do
    for $1 \rightarrow j$ do
        $ECU_{ij}$ generates random no $R_{ij}$ and send it to GECU.
        GECU generates random $seed_1$ and compute
          $MAC_1 = H_{K_{ij}}(ECU_{ij}||GECU||R_{ij}||seed_1)$
        GECU sends $seed_1||MAC_1$ to $ECU_{ij}$.
        $ECU_{ij}$ verifies $MAC_1$,
        if correct then
            $sk_{ij} = KDF_{LK_{ij}}(seed_1)$
        else
            FAIL
        end
        $ECU_{ij}$ generates $MAC_2 = H_{K_{ij}(ECU_{ij}||seed_1)}$ and send it to GECU.
        GECU verifies $MAC_2$,
        if correct then
            $sk_{ij} = KDF_{LK_{ij}}(seed_1)$
        else
            FAIL
        end
    end
end
```

ALGORITHM 2: Group key generation

```
for $1 \rightarrow i$ do
    $gk_i = H(sk_{i1}||sk_{i2} \ldots ||sk_{ij})$
end
```

ALGORITHM 3: Authenticated Encryption

```
1. Sender $ECU_{ij}$ applies a AE scheme on message $M$ using his own session key and
   compute $(C, T) = AE_{sk_{ij}}(M)$ and send it to GECU.
2. GECU receives $(C, T)$ pair and verifies it.
3. if correct then
    GECU further applies AE algorithm using group key $gk_i$ of the receiving
      channel and compute $(C', T')$ and forward it on that channel.
else
    Discard the message.
end
4. All ECUs on the receiving channel will receive $(C, T)$ and do the verification.
5. if correct then
    Keep the message
else
    Discard
end
```

5 Discussion on Security

Security of the proposed protocol depends on the underlying key management and authenticated encryption scheme.

5.1 Key Management

We assumed that each ECU is preloaded with two long-term symmetric keys during the manufacturing phase. Each ECU share these keys with GECU only. Further, these keys are used to generate individual session keys using existing AKEP2 protocol. It is a three pass protocol and provides perfect forward secrecy which means compromise of long-term keys does not compromise the past session keys. A protocol is considered to be secure if compromise of these keys doesn't have any adverse effect like

- It should not subvert subsequent authentication.
- it should not reveal any information about other session keys.

Group session key is generated by taking a hash of all individual keys on the same channel whose security depends on the hash function used in the computation.

5.2 Authenticated Encryption

The proposed protocol is considered to be secure if the underlying AE scheme is secure. In this work, we used AEGIS, an existing authenticated encryption scheme from CAESAR competition. Hence, we can directly adapt their security analysis from [22].

6 Implementation Results

In this section, we will discuss the implementation results of our proposed security architecture. To make the implementation scenario more realistic, we design a small-size CAN-FD network and schedule communication of each ECU. Fig. 5 shows the HIL (Hardware-In-the-Loop) network topology.

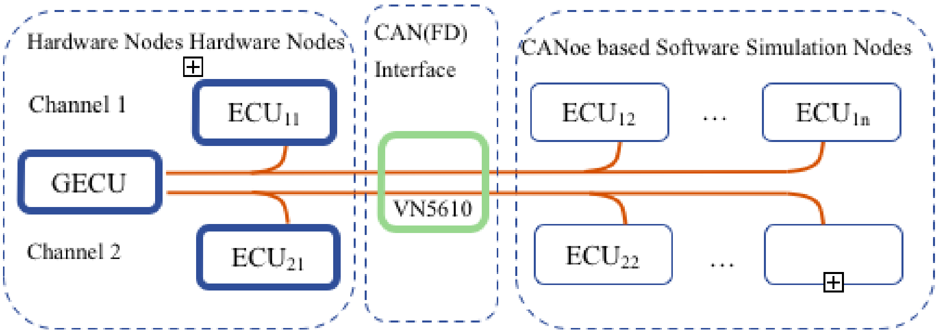


Fig. 5. CAN-FD Network Topology

To simulate nodes on the hardware, we use LPC54618 microcontroller which supports two CAN-FD channels and baud-rate up to 180Mhz. On the software, we use CANoe testing tool to build virtual ECUs which are programmed in CAPL (Communication Access Programming Language). VN5610 is a CAN(FD) interface connecting those two parts. Fig. 6 shows our setup for the hardware and software environments. We divide real-time performance test into 2 phases:

1. First phase known as initialization includes session key generation and distribution, group key generation and distribution.
2. Second phase includes authenticated encryption and decryption for a one message forwarding event by the ECU.

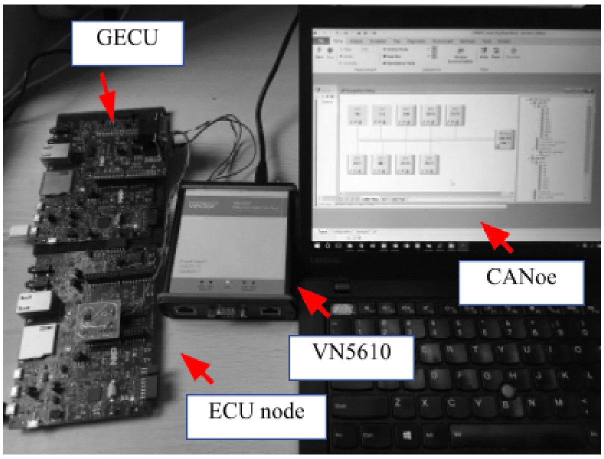
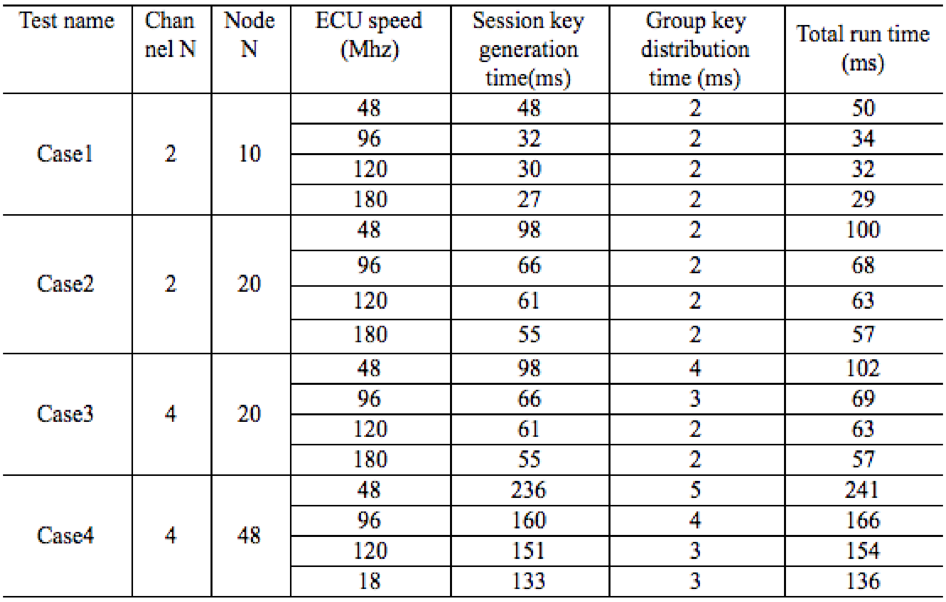


Fig. 6. Evaluation Environment

Initialization Phase During this phase initial key distribution time is recorded. The usual startup time of vehicle is around 200ms after ignition turned on. All ECUs need to execute key distribution during these 200ms only. Our mechanism, to some extent, is limited by the hardware performance of ECU nodes. We compute the initialization time by considering key distribution between GECU and n no. of ECU's which includes individual and group key distribution. To compute the timing results more accurately, we use the Trace function of CANoe to capture the timestamp of first and last frame of this program. Results are shown in fig. 7 considering different number of channels and ECU's.
We can see in the fig. 8, with the increase in the number of ECU's over a channel, the delay also increase. But most of the points are within allowable limits.

Message Forwarding Phase This test is about the real-time performance of secure data transmission. We consider a test scenario where GECU facilitates a transfer of message between two channels, fig. 9 shows a data flow diagram.



Test name	Chan nel N	Node N	ECU speed (Mhz)	Session key generation time(ms)	Group key distribution time (ms)	Total run time (ms)
Case1	2	10	48	48	2	50
			96	32	2	34
			120	30	2	32
			180	27	2	29
Case2	2	20	48	98	2	100
			96	66	2	68
			120	61	2	63
			180	55	2	57
Case3	4	20	48	98	4	102
			96	66	3	69
			120	61	2	63
			180	55	2	57
Case4	4	48	48	236	5	241
			96	160	4	166
			120	151	3	154
			18	133	3	136

Fig. 7. Key Distribution Results

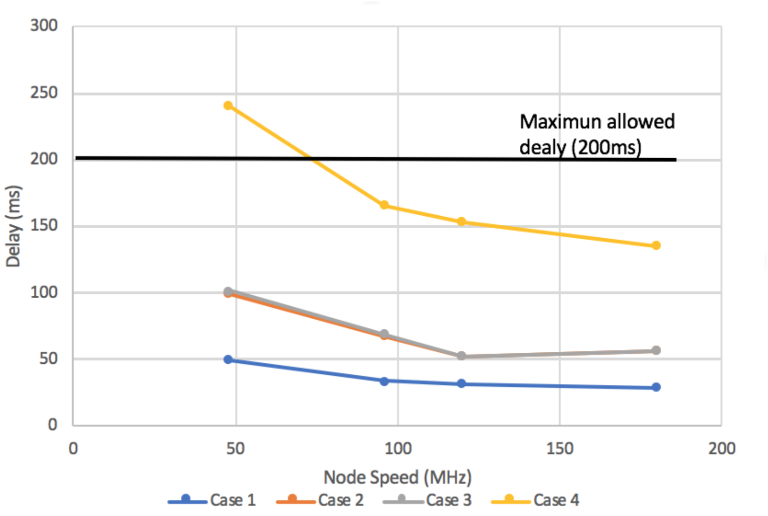


Fig. 8. Graph Representation of Key Distribution Results

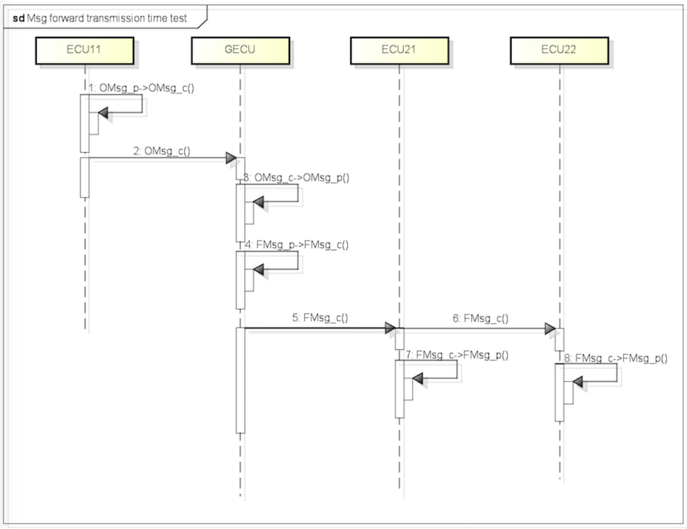
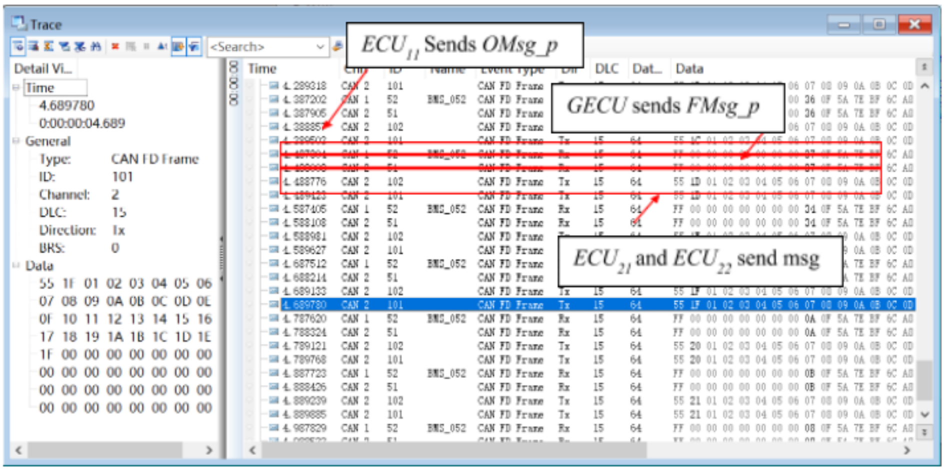


Fig. 9. Data Flow Diagram showing communication between ECU's over different channels through GECU.

As shown in fig. 9 ECU_{11} encrypt the original message $OMsg_p$ with the key GK_1 to get the ciphertext $OMsg_c$ and then sends it to GECU. On receiving $OMsg_c$, GECU decrypts and verifies it. If the verification fails, GECU discards the message otherwise it encrypts the decrypted message $OMsg_p$ with key GK_2(group key for channel 2) to get ciphertext $FMsg_c$ and sends it over channel 2. All EC U's on channel 2 receive $FMsg_c$, and they decrypt and verify it with GK_2 to get the original message $OMsg_p$. By capturing the start time of ECU_{11} to generate the $OMsg_p$ and finish time of ECU_{21} and ECU_{22} to decrypt and verify $FMsg_c$, we can get the total time of computation. Fig. 10 shows the Trace interface of CANoe as we can see to get the start time, we add an initial message. After ECU_{21} and ECU_{22} finishes decryption and verification, they send an end message. Therefore, by noticing these two-time stamps, we can get the accurate time.


Trace
<Search>
ECU_{11} Sends $OMsg\_p$
GECU sends $FMsg\_p$
ECU_{21} and ECU_{22} send msg

```
Detail Vi...
Time
    4.689780
    0:00:00:04.689
General
    Type:       CAN FD Frame
    ID:         101
    Channel:    2
    DLC:        15
    Direction:  Tx
    BRS:        0
Data
    55 1F 01 02 03 04 05 06
    07 08 09 0A 0B 0C 0D 0E
    0F 10 11 12 13 14 15 16
    17 18 19 1A 1B 1C 1D 1E
    1F 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00
```

```
Time      Chn   ID   Name     Event Type    Dir  DLC  Dat..  Data
4.289318  CAN 2 101           CAN FD Frame                   06 07 08 09 0A 0B 0C 0D
4.387202  CAN 1 52   BMS_052  CAN FD Frame                   00 36 0F 5A 7E BF 6C A0
4.387905  CAN 2 51            CAN FD Frame                   00 36 0F 5A 7E BF 6C A0
4.388857  CAN 2 102           CAN FD Frame                   06 07 08 09 0A 0B 0C 0D
4.389502  CAN 2 101           CAN FD Frame  Tx   15   64     55 1C 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D
4.487904  CAN 1 52   BMS_052  CAN FD Frame  Rx   15   64     FF 00 00 00 00 00 00 00 37 0F 5A 7E BF 6C A0
4.488608  CAN 2 51            CAN FD Frame  Rx   15   64     FF 00 00 00 00 00 00 00 37 0F 5A 7E BF 6C A0
4.488776  CAN 2 102           CAN FD Frame  Tx   15   64     55 1D 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D
4.489423  CAN 2 101           CAN FD Frame  Tx   15   64     55 1D 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D
4.587405  CAN 1 52   BMS_052  CAN FD Frame  Rx   15   64     FF 00 00 00 00 00 00 00 34 0F 5A 7E BF 6C A0
4.588108  CAN 2 51            CAN FD Frame  Rx   15   64     FF 00 00 00 00 00 00 00 34 0F 5A 7E BF 6C A0
4.588981  CAN 2 102           CAN FD Frame                   0A 0B 0C 0D
4.589627  CAN 2 101           CAN FD Frame                   0A 0B 0C 0D
4.687512  CAN 1 52   BMS_052  CAN FD Frame                   7E BF 6C A0
4.688214  CAN 2 51            CAN FD Frame                   7E BF 6C A0
4.689133  CAN 2 102           CAN FD Frame  Tx   15   64     55 1F 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D
4.689780  CAN 2 101           CAN FD Frame  Tx   15   64     55 1F 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D
4.787620  CAN 1 52   BMS_052  CAN FD Frame  Rx   15   64     FF 00 00 00 00 00 00 00 0A 0F 5A 7E BF 6C A0
4.788324  CAN 2 51            CAN FD Frame  Rx   15   64     FF 00 00 00 00 00 00 00 0A 0F 5A 7E BF 6C A0
4.789121  CAN 2 102           CAN FD Frame  Tx   15   64     55 20 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D
4.789768  CAN 2 101           CAN FD Frame  Tx   15   64     55 20 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D
4.887723  CAN 1 52   BMS_052  CAN FD Frame  Rx   15   64     FF 00 00 00 00 00 00 00 0B 0F 5A 7E BF 6C A0
4.888426  CAN 2 51            CAN FD Frame  Rx   15   64     FF 00 00 00 00 00 00 00 0B 0F 5A 7E BF 6C A0
4.889239  CAN 2 102           CAN FD Frame  Tx   15   64     55 21 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D
4.889885  CAN 2 101           CAN FD Frame  Tx   15   64     55 21 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D
4.987829  CAN 1 52   BMS_052  CAN FD Frame  Rx   15   64     FF 00 00 00 00 00 00 00 08 0F 5A 7E BF 6C A0
```

Fig. 10. Trace interface of CANoe

As the GECU is always fixed, we set it's clock speed at 180Mhz and varies the clock speed for rest of the ECU's. Fig. 11 shows the time on the same microcontroller with different clock speed and different baud rate for CAN-FD. Here we use AEGIS [23] authenticated encryption scheme for the secure communication and compare its results against AES with CCM mode for authentication. In the graph shown in fig. 11, solid lines shows the secure communication with AEGIS and the dotted line denotes no security at all. Obviously, in this result, the time depends on the microcontroller performance, and the time is controlled below 2ms, which is allowed in-vehicle communication. The graph in fig. 12 shows the results using AES (no hardware acceleration) with CCM. As we can see the time taken by AES-CCM is more than AEGIS, especially at lower frequency. Hence, we can conclude that use of fast authenticated encryption scheme can optimize the secure communication and meet the real-time requirements.

7 Conclusion

In this paper, we provided an improved security architecture for the CAN-FD network. We proposed a new group-based approach for the secure communication between ECUs connected on the CAN-FD network. In addition, we also used the authenticated encryption scheme instead of applying individual primitives for encryption

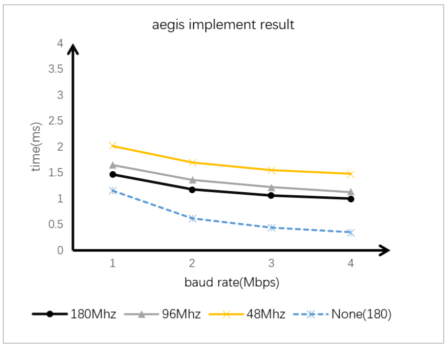


Fig. 11. AEGIS Implementation Results

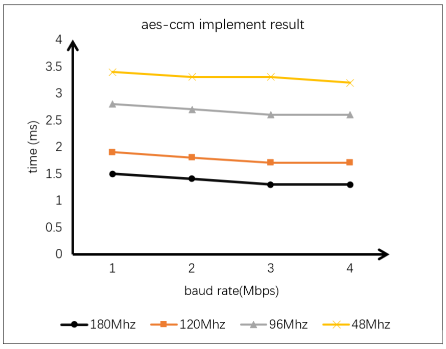


Fig. 12. AES-CCM Implementation Results

and authentication. We tested our results using AEGIS which is a fast authenticated encryption scheme satisfying real-time requirements. We compared the AEGIS implementation results against individual primitives (AES for encryption and CCM for authentication), and found that AEGIS performs better than AES-CCM. Hence, we can conclude authenticated encryption is a good choice for providing secure communication over CAN-FD network.

Acknowledgement This work was supported by SUTD start-up research grant SRG-ISTD-2017-124. The first author's work was done during her internship in SUTD.

References

1. Can standardization. `http://elearning.vector.com/index.php?&wbt_ls_seite_id=489557&root=378422&seite=vl_can_introduction_en`.
2. Comparing can fd with classical can. `https://www.kvaser.com/wp-content/uploads/2016/10/comparing-can-fd-with-classical-can.pdf`.
3. Caesar: Competition for authenticated encryption: Security, applicability, and robustness, 2014. `http://competitions.cr.yp.to/caesar.html`.
4. Can 2020: The future of can technology. `https://www.can-cia.org/news/cia-in-action/view/can-2020-the-future-of-can-technology/2016/3/21/`, 2016.
5. Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *J. Cryptol.*, 21(4):469–491, September 2008.
6. Josef Berwanger, Martin Peller, and Robert Griessbach. Byteflight - a new protocol for safety critical applications, 2000.
7. Can specification, 1991. http://esd.cs.ucr.edu/webres/can20.pdf.
8. Robert N. Charette. This Car Runs on Code. *IEEE Spectrum*, February 2009.
9. Next generation car network- flexray, 2006. http://www.fujitsu.com/downloads/CN/fmc/lsi/FlexRay-EN.pdf.
10. Bogdan Groza, Stefan Murvay, Anthony van Herrewege, and Ingrid Verbauwhede. Libra-can: A lightweight broadcast authentication protocol for controller area networks. In Josef Pieprzyk, Ahmad-Reza Sadeghi, and Mark Manulis, editors, *Cryptology and Network Security*, pages 185–200, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
11. Florian Hartwich and Robert Bosch Gmbh. icc 2012 can in automation can with flexible data-rate, 2012.

12. T. Hoppe and J. Dittman. Sniffing/replay attacks on can buses: A simulated attack on the electric window lift classified using an adapted cert taxonomy. In *Proceedings of the 2nd Workshop on Embedded Systems Security(WESS)*, 2007.
13. Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. Security threats to automotive can networks – practical examples and selected short-term countermeasures. In Michael D. Harrison and Mark-Alexander Sujan, editors, *Computer Safety, Reliability, and Security*, pages 235–248, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
14. Tianxiang Huang, Jianying Zhou, and Andrei Bytes. ATG: An Attack Traffic Generation Tool for Security Testing of In-vehicle CAN Bus. In *ARES*, 2018.
15. Tianxiang Huang, Jianying Zhou, Yi Wang, and Anyu Cheng. On the security of in-vehicle hybrid network: Status and challenges. In Joseph K. Liu and Pierangela Samarati, editors, *Information Security Practice and Experience*, pages 621–637, Cham, 2017. Springer International Publishing.
16. H. Kopetz. Automotive electronics: Present state and future prospects. In *Proceedings of the Twenty-Fifth International Conference on Fault-tolerant Computing*, FTCS'95, pages 66–75, Washington, DC, USA, 1995. IEEE Computer Society.
17. Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. Experimental security analysis of a modern automobile. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 447–462, Washington, DC, USA, 2010. IEEE Computer Society.
18. Andreea-Ina Radu and Flavio D. Garcia. Leia: A lightweight authentication protocol for can. In *ESORICS*, 2016.
19. Q. Wang and S. Sawhney. Vecure: A practical security framework to protect the can bus of vehicles. In *2014 International Conference on the Internet of Things (IOT)*, pages 13–18, Oct 2014.
20. S. Woo, H. J. Jo, I. S. Kim, and D. H. Lee. A practical security architecture for in-vehicle can-fd. *IEEE Transactions on Intelligent Transportation Systems*, 17(8):2248–2261, Aug 2016.
21. S. Woo, H. J. Jo, and D. H. Lee. A practical wireless attack on the connected car and security protocol for in-vehicle can. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):993–1006, April 2015.
22. Hongjun Wu and Bart Preneel. AEGIS: A fast authenticated encryption algorithm. In *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, pages 185–201, 2013.
23. Hongjun Wu and Bart Preneel. AEGIS: A Fast Authenticated Encryption Algorithm (v1). `http://competitions.cr.yp.to/round1/aegisv1.pdf`, 2015.