Improving Timing Behavior on Encrypted CAN Buses

Mingqing Zhang and Alejandro Masrur
Department of Computer Science
TU Chemnitz, Germany

***Abstract*—CAN is probably the most successful bus in the automotive domain, especially, due to its low cost and robustness. However, with increasing connectivity, there is a need to encrypt data to avoid attacks such as Spoofing and Sniffing. This ends up exposing CAN's severe limitations. In particular, each encrypted message requires sending two frames due to its restrictive payload in CAN. Moreover, each frame of an encrypted message undergoes a separate arbitration process which negatively impacts timing and makes it difficult to meet deadlines. In this paper, to work around this problem, we propose a technique that consists in assigning different priorities to encrypted CAN frames so as to compensate for increased delay. The basic idea is that, once the first frame of an encrypted CAN message wins arbitration, its second frame will always win arbitration within a specified scope and can be sent with lesser delay. We have conducted experiments on real hardware and performed extensive simulations indicating that the proposed technique reduces transmission delay to one half or even one third compared with the standard approach allowing us to still meet typical automotive deadlines on an encrypted CAN bus.**

***Index Terms*—CAN, AES, GCM, encryption, authentication, cybersecurity**

I. Introduction

The Controller Area Network (CAN) is a message-based multi-master bus protocol designed for Electrical Control Units (ECUs) in a vehicle to communicate with each other. CAN is widely spread in modern vehicles due to its robustness and low cost. However, with increasing connectivity, CAN shows serious limitations to provide security. Clearly, one can use more sophisticated buses instead like CAN-FD or FlexRay. However, this leads to higher costs, which jeopardizes competitiveness in cost-sensitive domains such as automotive systems.

In particular, since CAN has a multi-master nature, unauthorized nodes could freely send and receive messages. Without needing to take over any legitimate node, generally four types of attacks can be carried out by attackers having physical access to CAN:

• **Denial-of-service (DoS).** CAN uses autonomous bus arbitration based on message priorities. Attackers can flood the bus by continuously sending high-priority messages and thus prevent all other nodes from sending legitimate messages.

• **Replay.** Attackers can simply record and resend a legitimate message an infinite number of times without even being able to read the content of the message.

978-1-7281-3197-9/19/$31.00 ©2019 IEEE

• **Spoofing.** CAN has no message authentication. Any node connected to it can thus send any message with any ID. As a result, unauthorized nodes connected to the bus can masquerade as legitimate nodes and send malicious messages.

• **Sniffing.** Any unauthorized node connected to CAN is able to read and log data traffic on the bus.

So far, although there is no optimal solution, different approaches have been proposed for securing CAN against DoS, replay and spoofing attacks. Some of them, however, are not able to offer enough security [1] and others come at the cost of either significant changes to the CAN protocol [2] [3] — i.e., affecting applicability and inevitably leading to more costs — or non-negligible additional delays [4]. Moreover, not enough attention has been paid to sniffing attacks, which can lead to severe data breaches.

In this paper, our goal is to provide CAN with the highest possible security level — including rejecting sniffing attacks — without making any modification to the protocol itself and, hence, without increasing costs. On the other hand, we are concerned with guaranteeing a good timing behavior and, thereby, being able to continue meeting typical automotive deadlines on encrypted CAN buses.

To this end, we use the 128-bit Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM) mode [5] , which generates a 128-bit authentication tag univocally identifying transmissions on the bus.

However, since CAN's payload is limited to a maximum of 8 bytes, two encrypted data frames need to be transmitted for each message, which leads to increased delay. In particular, each such frame undergoes a separate arbitration process on the CAN bus and may be transmitted way apart from one another, depending on the bus utilization.

To overcome this problem, we propose a technique that allows sending both encrypted frames with acceptable delay by properly selecting message IDs. Further, we extend schedulability analysis to the case of encrypted CAN buses and perform experiments based on real hardware (viz., Arduino + CAN-BUS Shield) and on an OMNeT++ simulation of CAN. Our results indicate that the proposed technique reduces communication delay of encrypted CAN messages to one half or even one third compared to the standard, i.e., state-of-the-art, approach. This allows us to still meet typical automotive deadlines with reduced costs.

Structure of the paper. The paper is structured as follows: Section II presents some background information on CAN and

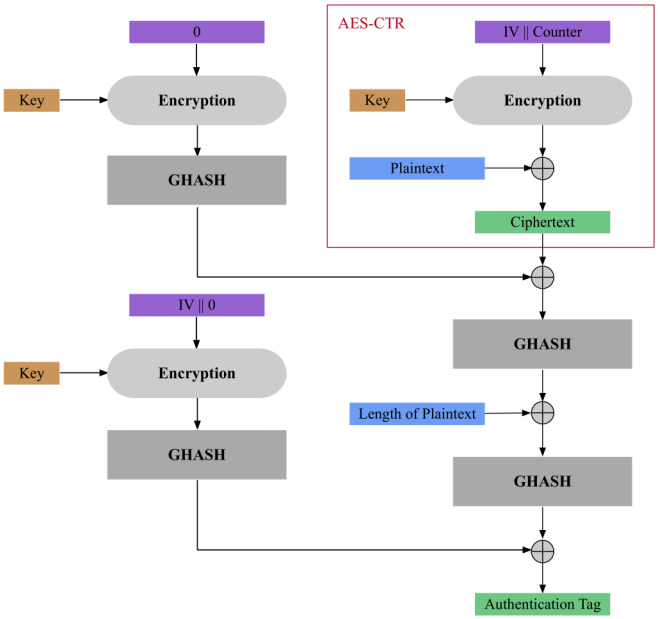

Fig. 1. AES encryption/decryption in GCM mode

AES. In Section III, previous related works are discussed. In Section IV, we explain our proposed approach in detail. Section V deals with schedulability analysis of CAN messages and its extension to the case of encrypted buses. Further, Sections VI and VII present our implementation and simulation results. Finally, Section VIII concludes the paper.

II. Background

A. Controller Area Network (CAN)

A standard CAN data frame has 11-bit ID and maximum 8 bytes of payload. Unique IDs are assigned to each message fixing their priorities. CAN implements a collision avoidance mechanism based on a bitwise arbitration. If the bus is idle and two nodes transmit simultaneously (recall CAN is a multi-master bus), the node sending a dominant bit 0 would win bus arbitration over a recessive bit 1. In other words, the lower the ID, the higher the message's priority will be. Messages losing arbitration will have to compete for the bus again later.

B. Advanced Encryption Standard

The Advanced Encryption Standard or Rijndael is a block cipher encryption standard established by the National Institute of Standards and Technology (NIST) [6].

Since a block cipher such as AES only defines encryption and decryption processes for a single block, different modes of operations need to be used to define how to repeatedly apply the same algorithm on multiple blocks:

• **Counter (CTR) mode.** As shown in Fig. 1, as part of Galois/Counter mode, it uses an Initialization Vector (IV) combined with a counter to generate a counter block, which is then encrypted and combined with the plaintext to produce the final ciphertext. The counter is increased after each encryption.

• **Galois/Counter Mode (GCM).** As shown in Fig. 1, GCM is an Authenticated Encryption (AE) cipher using CTR to generate ciphertext and GHASH function to include the length of plaintext and calculate an authentication tag through an XOR operation with the encrypted and hashed IV. The authentication tag is then transmitted together with the encrypted ciphertext to ensure both data integrity and authenticity.

In the next section, we discuss some previous related work to secure CAN and discuss their advantages and disadvantages.

III. Related Work

There are many ideas and approaches for securing CAN against different kinds of attacks.

A. Against DoS, Spoofing and Replay

An approach proposed in [7] calculates an 8-bit message authentication code, which is then transmitted together with the original data on CAN. In this case, two data frames need to be transmitted for each message with an original payload larger than 7 bytes. A monitoring node is also used for detecting unauthorized messages and destroying the same by sending error frames. [8] proposed a similar approach using asymmetric encryption to calculate message authentication codes. Another similar idea against DoS and spoofing attacks is proposed by Matsumoto et al. [9]: Each node monitors whether there are other nodes sending messages with IDs also used by itself and sends error frames on detection to override unauthorized messages. All the proposed ideas against DoS attacks are only partial solutions and not able to provide any defense against sniffing attacks as mentioned in [5].

The idea of sending data using message authentication codes is also used in [1] and [4]. An "out-of-band" channel is used in [1] to send a 112-bit hashed message authentication code. The problem is that CAN needs to be modified and it still does not provide a sufficient level of security. An even longer hashed message authentication code is used in [4]. For each standard message, three additional authentication messages are required to be transmitted. Although this approach provides enough security, sending four frames for each message can lead to significant delay. Some secure protocols for CAN such as MaCAN and LiBrA-CAN are proposed in [2] and [3] respectively. These protocols require modifying the CAN protocol in a significant manner and, as a result, are not easy to implement or can lead to a non-negligible increase in costs.

B. Against Sniffing

All approaches mentioned above are ineffective against sniffing attacks. Without encrypting the payload, valuable information can still be acquired by unauthorized nodes connected to CAN. A concept for sending encrypted data is proposed in [10]. In this approach, the keys need to be dynamically generated and synchronized across all nodes. Further implementation details, however, are not mentioned in the paper and can be problematic due to the need of

global synchronization. In the next section, we introduce our approach with the aim of overcoming these issues.

IV. The Proposed Approach

A. Encryption

We use AES in GCM mode as introduced above to encrypt and authenticate data sent over a CAN bus and, thereby, protecting it against replay, spoofing and sniffing attacks, since attackers can neither send nor receive legitimate messages on the bus (without the key and previous ciphertext). On the other hand, the proposed approach is not effective against DoS, particularly, if attackers gain physical access to the bus. For automotive systems, however, CAN can only be accessed via a gateway. Therefore, it is possible to adopt further security measures such as firewalls to reject DoS attacks. This latter is out of scope in this paper though. Since data is encrypted at each individual node before being sent, neither the CAN controllers nor the CAN protocol itself need to be modified.

Note that, an 8-byte authentication tag need to be attached to each encrypted piece of data, as a result, since CAN provides a limited payload of 8 bytes per frame, two data frames need to be transmitted for each authenticated and encrypted message. Once a node receives both these frames, it checks the authentication tag and restores/decrypts the original data.

Clearly, for implementing AES in GCM mode, not only the key, but also IVs and current counter values have to be stored on the local persistent memory of each node. Note that different symmetric keys, IVs and, as a result, different counters, are used for different CAN messages. A counter is incremented at the sender and at every receiver node each time the corresponding message is sent/received. This implies that nodes can only *understand* the messages they are supposed to send/receive, increasing robustness in the unlikely case that one of the nodes might get compromised.

B. ID Assignment

As already mentioned, two frames need to be sent for each authenticated and encrypted message. This leads to considerable additional delay, in particular, since individual frames undergo different arbitration processes on the CAN bus and may potentially be transmitted with a large separation between them.

To overcome this problem, we divide CAN's ID space into disjoint clusters as depicted in Fig. 2. Within each such cluster, we propose sending the first encrypted frame with a *mirrored* ID of the original (unencrypted) message, whereas the second encrypted frame is sent with the original message ID. To this end, we divide standard CAN's ID space into *clusters* as shown in Fig. 2 and assume that only the first half of the IDs in one cluster is used for the original messages.[1] IDs of original messages are mirrored to the second half of the ID space in the cluster and assigned to the first encrypted data frames. As shown in Fig. 2 for Cluster 1, 0x000, 0x001, 0x002 and 0x003 are IDs of original messages and second encrypted data frames. The corresponding mirrored IDs: 0x004, 0x005, 0x006 and 0x007 are assigned to the first encrypted data frames.

The proposed ID mirroring guarantees that the second frame of an encrypted message has always higher priority than the first frame of any other message within the same cluster. As a result, the two frames of any encrypted message are transmitted contiguously without interruption, which significantly improves the transmission delay for low-priority messages. On the other hand, the same technique leads to increased transmission delay for high-priority messages. However, this is limited to the scope of a cluster and will not affect higher-priority messages in other clusters.

Finally, note that the number of clusters is a parameter that can be adjusted according to requirements. The more clusters we select, the lesser the impact of ID mirroring on higher-priority messages. However, the benefit for lower-priority messages also diminishes. In an extreme case, the number of clusters is bounded by the number of messages in the system.

In the next section, we analyze schedulability of CAN messages with and without encryption.

V. Schedulability Analysis

A. Without encryption

Let us denote by M the set of messages m_i sent over CAN with $1 \leq i \leq n$ and n being the total number of messages in M. Further, let us assume without loss of generality that all m_i are sorted in the order of decreasing priority, i.e., m_1 has higher priority than m_2 and m_2 has higher priority than m_3 and so on.

As discussed above, if an m_i is not encrypted, it can be sent within one frame. We denote its transmission or communication time by c_i, which depends on CAN's bandwidth and the number of overhead, data and stuff bits sent.[2] Note that we consider the Intermission Field, i.e., the minimum separation between two consecutive frames on the bus, to be part of the overhead bits. In addition, we model m_i's period of repetition by p_i and its deadline by d_i with $d_i \leq p_i$.

From [11] we know that M's schedulability is guaranteed, if the following holds for all i and k:

$$r_{i,k} \leq d_i + (k-1) \cdot p_i, \tag{1}$$

with $1 \leq k \leq 1 + \left\lfloor \frac{t_{busy} - d_i}{p_i} \right\rfloor$ and t_{busy} is the longest possible busy interval on CAN, i.e., the longest time interval without idling, given by the fixed point of:

$$t_{busy} = \sum_{i=1}^{n} \left\lceil \frac{t_{busy}}{p_i} \right\rceil \cdot c_i. \tag{2}$$

The variable $r_{i,k}$ in (1) represents the worst-case response time (WCRT), i.e., the maximum possible delay, by the k-th

[1] Note that this is rarely a restriction, since CAN allows for 2^{11} possible IDs and, hence, 2^{10} IDs remain possible after mirroring. This is sufficient for most applications. If not, extended CAN can be used instead with up to 2^{29} possible IDs and 2^{28} after mirroring. Apart from having a larger ID space, the proposed approach remains valid for extended CAN without any modification.

[2] Note that CAN uses bit stuffing to embed the clock signal into data, in the end, increasing the transmission/communication time.

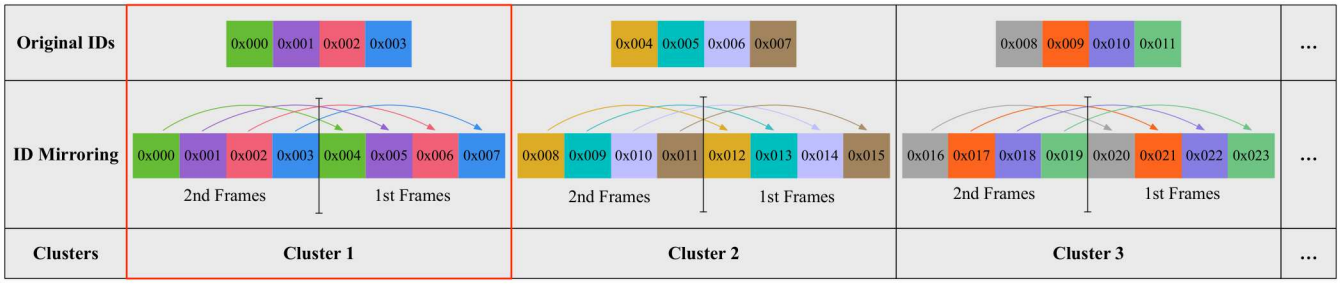

Fig. 2. Clustered Message ID Mirroring for encrypted CAN buses

transmission of m_i in t_{busy}. In other words, m_i is schedulable on CAN, if it can meet its deadline each time it is sent within t_{busy}. Now, to compute $r_{i,k}$ for a given i and k, we proceed as follows:

$$r_{i,k} = b_i + k \cdot c_i + \sum_{j=1}^{i-1} \left\lceil \frac{r_{i,k}}{p_j} \right\rceil \cdot c_j, \tag{3}$$

which is again a fixed point computation. In (3), $b_i = \max_{i+1 \leq j \leq n}(c_j)$ is m_i's blocking time, i.e., the maximum delay that m_i may incur due to lower-priority messages.

In (3), we assume that signals' propagation delays have been properly compensated at transceivers. In other words, if two or more nodes start sending simultaneously, the node with highest priority wins arbitration independent of its position/distance to other nodes on the bus.

B. With encryption

Under encryption, each message requires sending two frames (instead of only one), each of which contains 8 data bytes. Assuming that a maximum number of stuff bits are sent, we have that the transmission/communication delay of any encrypted data frame is equal to c_{max}, i.e., the maximum possible, and the whole message requires $2c_{max}$. The longest possible busy interval in this case is given by:

$$\hat{t}_{busy} = \sum_{i=1}^{n} \left\lceil \frac{\hat{t}_{busy}}{p_i} \right\rceil \cdot 2c_{max}. \tag{4}$$

In other words, since CAN's utilization increases under encryption, the busy interval also increases from t_{busy} to $\hat{t}_{busy}$. Now, if we follow a standard approach, i.e., where the original message ID is used for both encrypted frames, there are no significant changes in schedulability analysis as per (1), apart from the fact that each of the two frames of an encrypted m_i will have to be checked for $1 \leq i \leq n$. This time, clearly, $\hat{t}_{busy}$ has to be considered instead.

On the other hand, however, schedulability analysis needs to be extended to the proposed approach. Let us first consider the case of **only one cluster**. Here, once the first frame of an encrypted message wins arbitration, its second frame cannot be preempted due to the proposed ID assignment. That is, an encrypted message under the proposed approach can be modeled as a single frame with a transmission/communication delay of $2c_{max}$. This leads to a WCRT as follows:

$$\hat{r}_{i,k} = \hat{b}_i + k \cdot 2c_{max} + \sum_{j=1}^{i-1} \left\lceil \frac{\hat{r}_{i,k}}{p_j} \right\rceil \cdot 2c_{max},$$

and, since $\hat{b}_i = 2c_{max}$ is the blocking time by lower-priority messages, this can be reshaped to:

$$\hat{r}_{i,k} = 2c_{max} \cdot \left(1 + k + \sum_{j=1}^{i-1} \left\lceil \frac{\hat{r}_{i,k}}{p_j} \right\rceil \right). \tag{5}$$

Finally, schedulability of encrypted messages can be guaranteed, if the following holds for $1 \leq i \leq n$ and $1 \leq k \leq 1 + \left\lfloor \frac{\hat{t}_{busy} - d_i}{p_i} \right\rfloor$ with $\hat{t}_{busy}$ given as per (4):

$$\hat{r}_{i,k} \leq d_i + (k-1) \cdot p_i. \tag{6}$$

Now, in the case of **multiple clusters**, messages of higher-priority clusters might preempt the first as well as the second frame of a message in a lower-priority cluster. This is because the proposed ID assignment is restricted to the messages within one cluster. However, each message in a higher-priority cluster, can only preempt a message in a lower-priority cluster (either its first or second frame) by a total amount of time equal to $2c_{max}$ every time it is released. In addition, the blocking time $\hat{b}_i$ remains equal to $2c_{max}$ due to the lower-priority messages within the same cluster. The only exception is the lowest-priority message in each cluster. This will have a blocking time equal to c_{max} instead, since this can preempt the second frames of every message in its lower-priority clusters. On the other hand, we can safely ignore this situation assuming that $\hat{b}_i$ is always equal to $2c_{max}$ and, hence, (5) remains valid for the case of multiple clusters too.

VI. Implementation

We implemented the proposed approach on Arduino UNO Boards equipped with CAN-BUS Shields. A sender node applies AES-128 in GCM mode to each original 8-byte CAN message. The result is a 16-byte encrypted and authenticated message, which the node then divides into two 8-byte CAN frames that it sends over the bus. A receiver node receives the two CAN frames and combines them back into a 16-byte encrypted message, which it then decrypts and checks for authenticity using the same with AES in GCM mode.

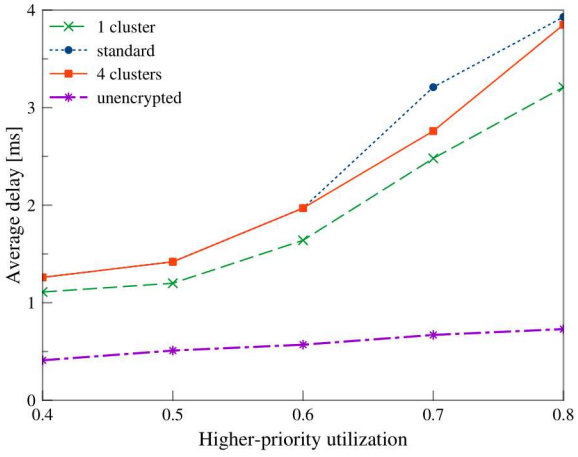

Fig. 3. Average transmission delay of the lowest-priority message with RM priorities and increasing higher-priority utilization

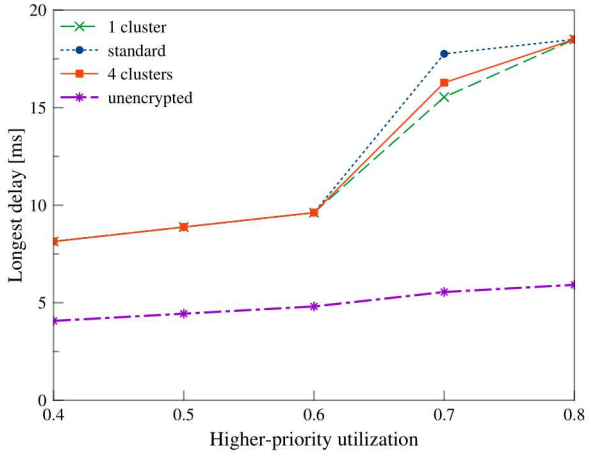

Fig. 4. Longest transmission delay of the lowest-priority message with RM priorities and increasing higher-priority utilization

We used the aforementioned setup to transmit around 10,000 CAN messages and collect relevant timing information. The overall average delay S of such an encrypted CAN message is around $2.5ms$ and is given by:

$$S = E + 2 \cdot (T + O) + D, \tag{7}$$

where E denotes the $878us$ GCM encryption and authentication time at the sender, T stands for the $250us$ average transmission delay of a single frame, O is $125us$ processing overhead at the CAN transceiver, and D is $874us$ GCM decryption and authentication time at the receiver.

Note that O, D and E are constant times that depend on used platform (in our case Arduino UNO Board/CAN-BUS Shield), whereas T depends on CAN's bandwidth and on how much contention occurs on the bus. In the next section, we evaluate the proposed approach in terms of improving timing under different levels of contention.

VII. Evaluation

For the sake of evaluating the proposed approach under different conditions, we make use of an OMNeT++ simulation of CAN [12] [13], which we configured using the parameter O, E, and D reflecting the used platform from previous section.

In particular, we simulated a number of nodes, each of which sends a message with a unique ID periodically. All periods in our simulations are generated from $10ms$ to $50ms$ — being typical values from automotive systems. We measured the transmission delay of a CAN message from the time its first frame is ready to be sent until the time its last frame finishes transmitting. Approximately 10,000 messages were transmitted in each simulation comparing the following four approaches:

• **Unencrypted** approach, which requires sending one unencrypted data frame per message.

• **Standard** approach, which requires sending two encrypted data frames per message, both with same ID as the original (unencrypted) message.

• Proposed approach with **1 cluster**, which sends (encrypted) two encrypted data frames per message with mirrored IDs.

• Proposed approach with **4 clusters**, which sends two encrypted data frames per message with clustered and mirrored IDs as discussed in Section IV-B, Fig. 2.

We performed evaluations comparing all four approaches (proposed clusters approach with 1 cluster and 4 clusters respectively) with respect to their average transmission delays in a CAN network up to 16 nodes — i.e., a typical automotive setting — under varying higher-priority utilization. We used two methods for assigning message priorities based on periods of messages: Rate Monotonic (RM), i.e., the shorter the period, the higher the message priority and Reversed Rate Monotonic (RRM). Whereas RM does not require much justification, RRM reflects the fact in an extreme manner that, in practice, priorities are often assigned to messages without taking their periods into account. This results in high-priority messages with long and low-priority messages with short periods.

Note that we configured the simulation to send original messages with a maximum payload of 8 bytes with worst-case stuff bits. As a result, these have a fixed transmission time on the bus and, hence, a fixed utilization for a specified period. We change the higher-priority utilization from 40% to 80% by adding more higher-priority messages in our simulated CAN.

In the RM case, as shown in Fig. 3, our proposed approach with 1 cluster and 4 clusters shows improvements with respect to average transmission delay as higher-priority utilization increases. As shown in Fig. 4, with respect to the longest delay, the proposed approach with 1 cluster and 4 clusters shows an improvement in the range of 60% to 80% utilization. This

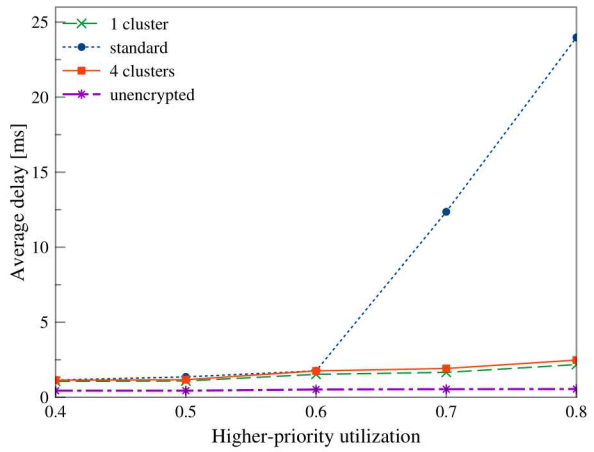

Fig. 5. Average transmission delay of the lowest-priority message with RRM priorities and increasing higher-priority utilization

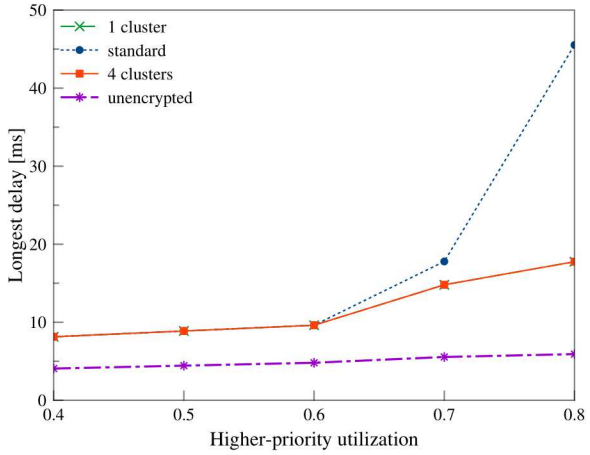

Fig. 6. Longest transmission delay of the lowest-priority message with RRM priorities and increasing higher-priority utilization

improvement reaches a peak of around $5ms$ at 70% utilization.

In the RRM case, as shown in Fig. 5, between 60% to 80% higher-priority utilization, our proposed approach with 1 and 4 clusters allows for much shorter average delays than the standard approach. At 80% higher-priority utilization, the average delay of the proposed approach with 1 cluster and 4 clusters is less than one-tenth of the standard approach.

As shown in Fig. 6, the proposed approach with 1 cluster and 4 clusters allows improving longest delays between 60% and 80% higher-priority utilization. At 80% higher-priority utilization, the longest delay of the proposed cluster approache in either configuration is around one third of the standard approach. This is due to the fact that higher-priority messages start being released multiple times at that utilization level leading to a considerably longer interruption/preemption time for the lowest-priority message compared to our proposed cluster approaches.

VIII. Conclusion

In this paper, we proposed an approach to secure CAN with acceptable overhead and without modifying the CAN protocol or CAN controller. By using AES in GCM mode for both encryption and authentication, CAN is secured against replay, spoofing and sniffing attacks.[3] However, two encrypted data frames need to be transmitted for each message leading to increased delay, in particular, since frames may be sent apart. To overcome this problem, we proposed assigning different IDs to the two encrypted data frames so as to guarantee that they are sent without interruption by other messages in the same cluster. Further, we extended the known schedulability analysis to consider encryption and illustrated the advantages of the proposed approach based on an implementation on real hardware and on an OMNeT++ simulation. We were able to show that our proposed approach reduces the transmission delay to one half or even one third of the standard approach, allowing us to use encryption on CAN and still meet typical automotive deadlines in the order of 10 to 50 milliseconds.

[3]Note that there is no satisfactory solution for DoS attacks due to CAN's multi-master nature.

References

[1] D. S. A. Van and I. Verbauwhede, "CanAuth - A Simple Backward Compatible Broadcast Authentication Protocol for CAN Bus," in *9th Embedded Security in Cars Conference*, 2011.

[2] C. R. O. Hartkopp and R. Schilling, "MaCAN - message authenticated CAN," in *10th Escar Conference on Embedded Security in Cars*, 2012.

[3] A. V. H. B. Groza, S. Murvay and I. Verbauwhede, "LiBrA-CAN: a Lightweight Broadcast Authentication protocol for Controller Area Networks," in *11th International Conference on Cryptology and Network Security*, 2012.

[4] Z. King and S. Yu, "Investigating and securing communications in the Controller Area Network (CAN)," in *2017 International Conference on Computing, Networking and Communications*, 2017.

[5] D. A. McGrew and J. Viega, "The Galois/Counter Mode of Operation (GCM)," *NIST Comput. Security Div., Comput. SecurityResour. Center, Gaithersburg, MD, Tech. Rep.*, 2005.

[6] J. Daemen and V. Rijmen, *The Design of Rijndael*. 2002.

[7] H. U. et al., "Security authentication system for in-vehicle network," *SEI Technical Review*, 2015.

[8] S. Fassak, Y. E. H. E. Idrissi, N. Zahid, and M. Jedra, "A secure protocol for session keys establishment between ecus in the can bus," in *2017 International Conference on Wireless Networks and Mobile Communications*, 2017.

[9] T. Matsumoto, M. Hata, M. Tanabe, K. Yoshioka, and K. Oishi, "A Method of Preventing Unauthorized Data Transmission in Controller Area Network," in *2012 IEEE 75th Vehicular Technology Conference*, 2012.

[10] W. A. Farag, "CANTrack: Enhancing automotive CAN bus security using intuitive encryption algorithms," in *2017 7th International Conference on Modeling, Simulation, and Applied Optimization*, 2017.

[11] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, 2007.

[12] J. M. et al., "A Simulation Environment based on OMNeT++ for Automotive CAN–Ethernet Networks," in *4th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*, 2013.

[13] K. Kawahara, Y. Matsubara, and H. Takada, "A Simulation Environment and preliminary evaluation for Automotive CAN-Ethernet AVB Networks," *CoRR*, 2014.