MixCAN: Mixed and Backward-Compatible Data Authentication Scheme for Controller Area Networks

Teri Lenard, Roland Bolboacă, Béla Genge, and Piroska Haller
University of Medicine, Pharmacy, Science and Technology of Târgu Mureş, Romania
N. Iorga, No. 1, Târgu Mureş, Mureş, Romania, 540088
Email: teri.lenard@umfst.ro, roland.bolboaca@umfst.ro, bela.genge@umfst.ro, piroska.haller@umfst.ro

***Abstract*—The massive proliferation of state of the art interfaces into the automotive sector has triggered a revolution in terms of the technological ecosystem that is found in today's modern car. Accordingly, on the one hand, we find dozens of Electronic Control Units (ECUs) running several hundred MB of code, and more and more sophisticated dashboards with integrated wireless communications. On the other hand, in the same vehicle we find the underlying communication infrastructure, which is struggling to keep up with the pace of these radical changes. This paper presents MixCAN (MIXed data authentication for Control Area Networks), an approach for mixing different message signatures (i.e., authentication tags) in order to reduce the overhead of Controller Area Network (CAN) communications. MixCAN leverages the attributes of Bloom Filters in order to ensure that an ECU can sign messages with different CAN identifiers (i.e., mix different message signatures), and that other ECUs can verify the signature for a subset of monitored CAN identifiers. Extensive experimental results based on Vectors Informatik's CANoe/CANalyzer simulation environment and the data set provided by Hacking and Countermeasure Research Lab (HCRL) confirm the validity and applicability of the developed approach. Subsequent experiments including a test bed consisting of Raspberry Pi 3 Model B+ systems equipped with CAN communication modules demonstrate the practical integration of MixCAN in real automotive systems.**

***Index Terms*—Controller Area Networks, Electronic Control Units, In-Vehicle security, Data authentication.**

I. Introduction

The massive proliferation of state of the art interfaces into the automotive sector has triggered a revolution in terms of the technological ecosystem that is found in today's modern car. Accordingly, on the one hand, we find dozens of Electronic Control Units (ECUs) running several hundred MB of code, and more and more sophisticated dashboards with integrated wireless communications [1]. On the other hand, in the same vehicle we find the underlying communication infrastructure, which is struggling to keep up with the pace of these radical changes.

In today's modern vehicles the "backbone" communication is provided by the Controller Area Network (CAN). Standardised in 2003 [2], it is an International Standardization Organization (ISO) - defined communications bus that describes the rules for exchanging data frames between devices. Given its limitations mainly in terms of bandwidth and payload size, recently, two main improved communication infrastructures have been proposed. The CAN+ protocol was proposed by Ziermann, *et al.* in 2009 [3], and it exploits the time between transmissions to send additional data. More recently, in 2012, Robert Bosch Gmbh developed the CAN with flexible data-rate protocol (CAN-FD) [4], which brings several advantages over CAN and CAN+, amongst which the most significant being higher bandwidth and larger payload.

While this technological advancement has brought upon a wide range of advantages and integrated features, it also exposed modern vehicles to significant threats and a new breed of *cyber-physical* attacks. In this case the attackers (e.g., vehicle owners, malicious actors) exploit software vulnerabilities (or undocumented features) in order to alter the vehicle's behavior, to gain certain advantages, or simply to cause physical damage [5], [6]. Unfortunately, while several notable techniques have emerged in the attempt to address the data authentication requirements in CAN communications [7], [8], [9], as it turns out, the practical integration of such techniques is impaired by the restrictions of the CAN bus. Therefore, most prior works presume the recent technological advancements regarding the CAN protocol (e.g., CAN+, CAN-FD) [10], [11]. However, in order to leverage the features of CAN+ of CAN-FD, expensive hardware update is necessary.

Staring with the work of Radu [8], it was acknowledged that data authentication techniques need to be backward-compatible with the standard CAN protocol. However, while such works are indeed AUTOSAR-compliant [12], they authenticate each frame or group of frames according to each frame's identifier. AUTOSAR is a series of standards that defines the *Secure On-board Communication* module including guidelines for implementing data authentication. The standard recommends using 128-bit keys, 64-bit Message Authentication Codes (MACs), alongside counters or timestamps to enforce freshness. In terms of MAC computations, the standard also recommends that MACs should be computed on the frame identifier, the actual data, and the value of the counter/timestamp. In the case of Radu [8] and other related studies [10], [13], for each distinct frame an additional frame transporting the authentication tag is also issued. However, we observe that such a procedure can significantly affect the available communication bandwidth with impact on mission-critical services.

In order to address the issue of data authentication in CAN networks, this paper presents the MIXed data authentication for Control Area Networks (MixCAN) approach, that mixes different frame signatures (i.e., authentication tags) in order to reduce the overhead of CAN communications. MixCAN leverages the attributes of Bloom Filters in order to ensure that one ECU can sign frames with different CAN identifiers (i.e., mix different frame signatures), and that other ECUs can verify the signature for a subset of monitored CAN identifiers. Given the significant limitations of the CAN protocol in terms of frame size, bandwidth, CAN identifiers, and the fundamental requirement of ensuring backward compatibility, MixCAN encapsulates an application-level data authentication strategy. As a result, MixCAN can be deployed by regular software updates, and can function in a hybrid environment with nodes not supporting MixCAN. Extensive experimental results based on Vector Informatik's CANoe/CANalyzer simulation environment and the data set provided by Hacking and Countermeasure Research Lab (HCRL) confirm the validity and applicability of the developed approach. Subsequent experiments including a test bed consisting of two Raspberry Pi 3 Model B+ systems equipped with CAN communication modules demonstrate the practical integration of MixCAN in real automotive systems.

The remainder of this paper is structured as follows. Related studies are briefly documented in Section II. The developed scheme including a detailed security analysis is presented in Section III. Next, the experimental results are presented in Section IV, and the paper concludes in Section V.

II. Related Work

Several techniques have been developed in order to secure CAN communications. Starting with the work of Herrewege, *et al.* [7], it was shown that, while the CAN bus has significant limitations in terms of payload and bandwidth, data authentication is still achievable via Message Authentication Codes (MAC), and more specifically the HMAC standard, alongside a counter in order to ensure freshness and resistance against replay attacks. The approach entitled *CANAuth* presumes the presence of the CAN+ protocol, which permits out-of-band transmissions [3]. By building on the extensive features provided by CAN+, the authors also developed a key distribution protocol, where each master node (from each group of related nodes), periodically issues a new group key.

By leveraging the breakthrough in the field of sensor networks, namely the Timed Efficient Stream Loss-tolerant Authentication protocol (TESLA) [14], B. Groza, *et al.* [15], developed an approach specially tailored to the CAN protocol. The approach integrated symmetric cryptographic functions (i.e., MAC) for data authentication and for releasing new session keys. Similarly to TESLA, keys are chained and released after the authenticated frames.

Next, we mention the more recent work of Radu and Garcia [8], where the *LeiA* CAN authentication protocol was presented. *LeiA* is designed to be backward compatible with existing vehicle's CAN infrastructure. Similarly to the approach undertaken in the present work, *LeiA* authenticates each CAN frame by adding one additional frame (and new CAN identifier) to each CAN frame. The newly added frame carries the authentication tag and leverages a 64-bit truncated MAC.

By leveraging the advantages of the CAN-FD protocol, Woo, *et al.* [10], developed a secure communication protocol and security architecture for CAN-FD-enabled in-vehicle systems. The approach embraces two distinct layers. A first security layer, where regular ECUs communicate securely by leveraging symmetric cryptographic constructions (e.g., symmetric encryption, and MAC). This is reinforced by a second security layer consisting of an advanced gateway capable of signing and verifying signatures via asymmetric cryptographic computations. In terms of data security, Woo's approach involves data encryption (via symmetric cryptography) and authentication (via MAC computation). However, the additional encryption introduces significant overhead on the CAN, which is mitigated by the assumption of CAN-FD as underlying communication infrastructure.

Wang and Liu [13] acknowledged the importance of securing gateway ECUs (i.e., ECUs enabling external communications), which provide various communication interfaces, including the support for Vehicle to Internet communications. These benefit from significant computational power and communication bandwidth and enable the realisation of new infrastructural paradigms such as the Internet of Vehicles [16]. Wang's approach focuses on the secure distribution of cryptographic keys in and outside the vehicle. In a similar fashion, we find other approaches, where the emphasis is placed on the distribution of cryptographic keys [17]. Unfortunately, the data authentication has received less attention, and, as already mentioned, most recent works [11], [18] presume the availability of the new CAN-FD protocol, which requires expensive hardware changes.

III. Developed Approach

A. Addressed Problems

The data frame authentication envisioned in this work addresses the following two problems, for which solutions are missing in related studies. First, in CAN communication systems, it is customary for one node (e.g., Electronic Control Unit – ECU) to transmit several frames with different CAN identifiers (CIDs). The same node is usually programmed to process and respond to a different set of CIDs. According to existing schemes [8], each frame sent by a node needs to be accompanied by a signature, which means that for each CID an additional CID is used in order to publish the signature as well. A direct consequence of such techniques is that the necessary CIDs are essentially doubled. Furthermore, if individual CAN frames are authenticated, such a procedure can almost double the number of CAN frames, with severe repercussions on the performance of the underlying communication hardware, and ultimately, on the execution of critical functions (e.g., engine control).

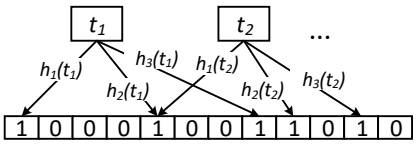

Fig. 1. An example Bloom Filter with $m = 12$, and $l = 3$.

The second, and more significant problem that is addressed by the developed approach is that if one ECU signs all frames (with different CIDs) once, the receiving nodes that would otherwise process only a subset of CIDs, will be forced to process and store all frames issued by the transmitting node in order to verify the signature. Unavoidably, such a procedure would increase not only the impact on the CAN communication infrastructure, but also on the verifier ECUs.

B. Bloom Filters

In order to address the problems mentioned above, the developed solution leverages the Bloom Filter [19]. The Bloom Filter (BF) is a probabilistic data structure that provides a space-efficient solution for representing a set of n items $T = \{t_1, t_2, ..., t_n\}$ by leveraging a set of l independent hash functions $H = \{h_1, h_2, ..., h_l\}$. Each hash function $h \in H$ maps an item $t \in T$ to an integer value in the range of $[0, m)$, where m is the size (in bits) of the Bloom Filter array represented as BF $= \{b_0, b_1, ..., b_{m-1}\}$. Here, b_i denotes a single bit from the BF. Ultimately, each hash function h_l sets one bit in the Bloom Filter array.

In order to insert an item $t \in T$ into the BF, t is hashed with each hash function $h \in H$. The result constitutes a set of values $\{h_1(t), h_2(t), ..., h_l(t)\}$, which set to 1 the corresponding bit in the BF according to:

$$\forall b_i \in \text{BF}, b_i = 1 \text{ iff } i \in \{h_1(t), h_2(t), ..., h_l(t)\}. \quad (1)$$

In order to query if an item t' has been added to a BF, the same procedure is applied as before in order to obtain the set of values $\{h_1(t'), h_2(t'), ..., h_l(t')\}$. If any of the bits at the positions indicated by these values is 0, then certainly we can conclude that $t' \notin$ BF. Otherwise, if all bits are set to 1, then $t' \in$ BF with a high probability level. Note that the Bloom Filter is a probabilistic data structure, where the possibility of collision exists, and the data structure thus exhibits a certain level of false positives. This can be tuned, though, according to the chosen parameters, as shown later in this paper. On the other hand, the Bloom Filter does not exhibit false negatives in the membership verification process. An example BF with $m = 12$ and $l = 3$ is illustrated in Fig. 1.

C. MixCAN: Approach Description

The developed approach named MixCAN (MIXed data authentication for Control Area Networks) builds on the attributes of the BF described above in order to ensure that one node (i.e., ECU) can sign frames with different CIDs (i.e., mix different frame signatures), and that other nodes can verify the signature for a subset of monitored CIDs. The reader should note that within the developed approach, the term "sign" denotes the computation of an authentication tag by leveraging a keyed hash function (i.e., a MAC). For this purpose we presume that a symmetric cryptographic key k has been securely distributed amongst all communicating nodes (ECUs) and it has been stored in a tamper-resistant module (e.g., a Trusted Platform Module). The security protocol for distributing such keys is not within the scope of MixCAN, specific techniques can be found in related studies [11], [17].

1) Mixing Time Window: Similarly to prior works, MixCAN decouples the transmitted data from the actual signature. Accordingly, the signature is transmitted at a later time on an aggregated and mixed set of frames according to a time window. Given the criticality of frames (identified according to their priority), the time window needs to be computed according to the frame priority values, and the maximum delay that is tolerated for receiving the authentication tag. The establishment of this time window can be made at design time, since both transmitted and received frames, their periodicity, as well as their CIDs are known apriori. Consequently, MixCAN leverages this information in order to establish, at design time, the value of this time window. More formally, let i denote the i-th frame that is transmitted by a particular ECU, and let δ_i to denote the maximum tolerated delay for receiving the authentication tag for frame i. Then, for a set of transmitted frames I, the time window W is computed as:

$$W = \min_{i \in I} \delta_i. \quad (2)$$

2) Constructing the Secure Bloom Filter: As already mentioned, MixCAN builds on the concept of Bloom Filters, and in particular, on the concept of encrypted Bloom Filters [20]. Accordingly, the family of hash functions responsible for inserting an element into the BF data structure are replaced by encryption algorithms. To this end, in order to insert an item t into the BF, an encryption operation $E_k(t)$ is performed with a cryptographic key k. The result of the encryption is then split into $\lceil log_2(m) \rceil$ parts. Similarly to the approach described in [20], MixCAN leverages the output of a cryptographic function, namely a Message Authentication Code (MAC), as the output of the l hash functions. The MAC possesses the required statistical properties for BF (as also demonstrated in [20]), and it has several advantages in terms of security, namely:

- The MAC benefits from the first preimage resistance, a property that is inherited from hash functions. The first preimage resistance property entails that, given $y = \texttt{MAC}_k(x)$, then it is computationally infeasible to find x if one only has knowledge of y. Here, $\texttt{MAC}_k(x)$ is a MAC function applied on item x with the secret key k.
- The MAC also brings the second preimage resistance property, which entails that given $y = \texttt{MAC}_k(x)$, it is computationally infeasible to find $x' \neq x$ such that $y = \texttt{MAC}_k(x')$. The second preimage resistance property guarantees the collision resistance.
- The practical complexity of the brute force approach for successfully mounting a collision attack is significantly

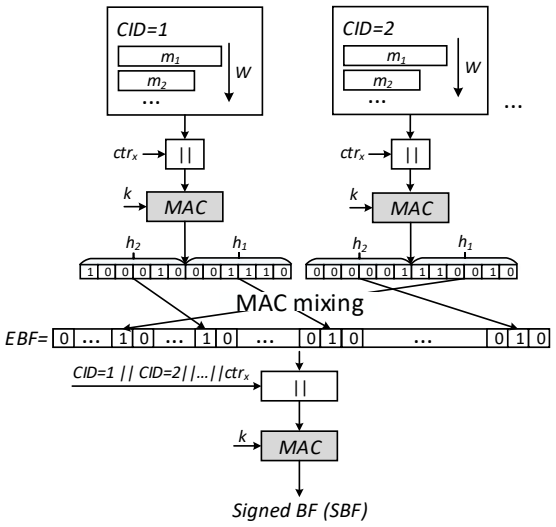

Fig. 2. MixCAN's architecture.

larger in the case of MAC functions (opposed to the case of hash functions), since, for each attempt there is a need to validate the particular "guess". Accordingly, such an attack usually requires the presence of an "oracle", that is, an entity that can be interrogated upon the correctness of a particular guess.

- Since it builds upon cryptographic hash functions, the MAC is computation-efficient and can be implemented with lightweight cryptographic algorithms in resource-constrained scenarios [21].

In a nutshell, MixCAN's construction is visualized in Fig. 2. Let $\mathcal{C}_i$ denote the set of CIDs sent by node i, $x \in \mathcal{C}_i$ to denote CID x from $\mathcal{C}_i$, and F_{ix}^W to denote the sequence of frames (including their CIDs and payload) sent by node i with CID x in time window W. Accordingly, let $\mathcal{C}_i^*$ denote the set of all combinations of subsets of CIDs excluding the empty set ($\emptyset \notin \mathcal{C}_i^*$), and $\mathcal{X} \subseteq \mathcal{C}_i^*$ an element from this set, hereinafter called a *mix set*. Then, given two mix sets $\mathcal{X}_1 \subseteq \mathcal{C}_i^*$ and $\mathcal{X}_2 \subseteq \mathcal{C}_i^*$ from $\mathcal{C}_i^*$, these are considered to be disjoint, namely $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$.

Based on the definitions above, for a specific mix set $\mathcal{X} \subseteq \mathcal{C}_i^*$, and for each $x \in \mathcal{X}$, MixCAN computes:

$$y_x = \texttt{MAC}_k(F_{ix}^W || ctr_x), \tag{3}$$

where $||$ denotes concatenation, and ctr_x is the freshness counter. This first step of MixCAN aggregates a sequence of frames transmitted under a specific CID by applying a MAC function. The use of the MAC function in this particular step is essential, since, in order to mount a brute force attack against the developed scheme, by leveraging this approach, the attacker would also require the presence of an oracle to interrogate the correctness of a particular guess.

The next step is the frame mixing procedure envisioned for MixCAN, which essentially entails the insertion of several aggregated MACs (y_x) into the same encrypted Bloom Filter (EBF). Accordingly, let $\texttt{EBF}^{\mathcal{X}}$ denote the EBF associated to the mix set $\mathcal{X}$. Then, given the size of the EBF m, y_x is split into $\lceil log_2(m) \rceil$ parts. Let V_{y_x} denote the set of values obtained after performing the split operation. According to the fundamental principles of BF, each $v \in V_{y_x}$ indicates the bit position in the BF, which is set to 1. More formally:

$$\texttt{EBF}^{\mathcal{X}}[v] = 1, \forall v \in V_{y_x}. \tag{4}$$

Lastly, MixCAN applies a MAC function on each $\texttt{EBF}^{\mathcal{X}}$ for each $\mathcal{X} \subseteq \mathcal{C}_i^*$. The operation includes the concatenation of the CIDs and of the freshness counter's value. Essentially, the concatenated CIDs provide the unique mix set identifier, while the counter is used to ensure the freshness of the authenticated EBF, which is a fundamental requirement set forth by the AUTOSAR specifications. More formally, this last step computes a MAC signature with the pre-shared cryptographic key k on the encrypted Bloom Filter $\texttt{EBF}^{\mathcal{X}}$ for the mix set $\mathcal{X}$ as:

$$\texttt{SBF}_{ctr_x}^{\mathcal{X}} = \texttt{MAC}_k(\hat{\mathcal{X}} || ctr_x || \texttt{EBF}^{\mathcal{X}}), \tag{5}$$

where $\hat{\mathcal{X}}$ denotes the concatenated CIDs from the mix set $\mathcal{X}$ and ctr_x as the freshness counter.

Once the signature on the encrypted Bloom Filter is computed, the transmitter outputs $\texttt{EBF}^{\mathcal{X}}$ together with $\texttt{SBF}_{ctr_x}^{\mathcal{X}}$, as the tuple:

$$< \texttt{EBF}^{\mathcal{X}}, \texttt{SBF}_{ctr_x}^{\mathcal{X}} > . \tag{6}$$

3) Counter Synchronization: It is essential that nodes have the opportunity to synchronize on the value of the counter. Since MixCAN does not send the value of the counter ctr_x with each authenticated frame, its value needs to be periodically transmitted. For this purpose, a MAC-based construction is used, and the following tuple is periodically sent on the CAN bus:

$$< ctr_x, \texttt{MAC}_k(CID, ctr_x) > . \tag{7}$$

In Eq. (7) CID denotes the CAN identifier of the frame that is used to send the value of ctr_x. This construction, once again, corresponds to the requirements set forth by the AUTOSAR standard, which requires that signed (i.e., authenticated) frames encompass both freshness and authentication.

4) Signature Verification: After having received the sequence of frames F_{ix}^W from node i, and the signed encrypted Bloom Filter $< \texttt{EBF}^{\mathcal{X}}, \texttt{SBF}_{ctr_x}^{\mathcal{X}} >$, the recipient first verifies the validity of the MAC of $\texttt{EBF}^{\mathcal{X}}$ by applying the same pre-shared cryptographic key k and the incremented value of the counter ctr_x. If successful, then it proceeds with the computation of the MAC for F_{ix}^W according to Eq. (3), obtaining thus the value y_x. By following similar steps as in the case of the insertion procedure, the verifier then splits y_x into $\lceil log_2(m) \rceil$ parts, thus obtaining the set of values V_{y_x}. As a last step, the verifier checks if all bits identified by each $v \in V_{y_x}$ are in $\texttt{EBF}^{\mathcal{X}}$. If so, then the verifier concludes that the signature is valid.

Considering that the BF is a probabilistic data structure, intuitively, an attacker may possess realistic chances of injecting additional data frames into a encrypted Bloom Filter. However,

we observe that in the case of in-vehicle systems communications patterns are static, and we rarely find new nodes (e.g., replacement of an ECU, the addition of new sensors/ECUs). As a result, two subsequent failed signature verifications should raise serious alarms. Therefore, we observe that while the BF is probabilistic, as discussed later, the attacker's success can be significantly limited by the implementation of thresholds, alarms, and mitigation strategies.

D. Security Analysis

Given that the BF is a probabilistic data structure, the rate of false positives, that is, the probability for an interrogation to return true for an item that is not present in the BF, needs to be analyzed. To this end, we find two main computations for the false positive rate. The first computation, as described in [22] (denoted by P^1), and the more recent one presented in [23] and revised in [24] (denoted by P^2).

In the former case, the false positive rate is computed as:

$$P^1 = \left(1 - \left(1 - \frac{1}{m}\right)^{ln}\right) \approx \left(1 - e^{-ln/m}\right)^l, \quad (8)$$

where m denotes the size of the BF, n is the number of inserted items, and l is the number of hash functions. By leveraging (8), the optimal configuration for l, as a parameter of n and m can be computed as:

$$l = \frac{m}{n}\ln(2). \quad (9)$$

More recently [23], [24] it was observed that the probability value P^1 is rather imprecise. This is owed to the fact that in [22] it was assumed that the items that are added to the BF are independent from each other. Accordingly, the more recent approximation of the false positive rate (as given in [24]) is:

$$P^2 = \frac{m!}{m^{l(n+1)}} \sum_{i=1}^{m} \sum_{j=1}^{i} (-1)^{i-j} \frac{j^{ln} i^l}{(m-i)!j!(i-j)!}. \quad (10)$$

Considering that the BF is protected by a MAC with a size of at least 64 bits, which, according to the National Institute of Standards and Technology (NIST) [25] provides a sufficient level of security, the security level of MixCAN is then equal to the security of the BF. More specifically, MixCAN's security and, in particular, its susceptibility to collision attacks (i.e., false positives in the BF domain), equals the probability of false positives in BF. According to Eq. (10), the false positive rate is given by the computation of P^2.

Numerically, the probability of false positives is illustrated in Table I. Here, we have computed the probability of false positives (for both P^1 and P^2), in the case of two hash functions used to compute the MAC: MD5 and SHA-1. According to the computations, for a BF size of $m = 64$ bits, with n between 1 and 5 elements, there is a significant difference between the values computed for P^1 and P^2. For example, in the case of 3 elements in the BF with MD5 as hash function, the value of P^1 is $0.6 \cdot 10^{-4}$, while that of P^2 is of $0.148 \cdot 10^{-3}$. Similar differences are visible for the other cases of n. The value of the false positives increases with the size of the hash value (i.e., by using SHA-1). This is owed to the increase of l (the number of hash functions in the BF), since each function alters one bit in the BF.

TABLE I
PROBABILITY OF FALSE POSITIVES COMPUTED FOR $m = 64$, $l = 22$ FOR MD5 AND $k = 28$ FOR SHA-1.

	P^1		P^2	
n	MD5	SHA-1	MD5	SHA-1
1	$0.159 \cdot 10^{-11}$	$0.242 \cdot 10^{-12}$	$0.551 \cdot 10^{-11}$	$0.158 \cdot 10^{-11}$
2	$0.21 \cdot 10^{-6}$	$0.276 \cdot 10^{-11}$	$0.637 \cdot 10^{-11}$	$0.129 \cdot 10^{-11}$
3	$0.6 \cdot 10^{-4}$	$0.153 \cdot 10^{-3}$	$0.148 \cdot 10^{-3}$	$0.475 \cdot 10^{-3}$
4	0.0016	0.0048	0.0033	0.0106
5	0.0129	0.0357	0.0218	0.0617

While the BF suffers from an increased level of false positives, we make the following observations. First, implementations should limit the value of n to at most 3. Otherwise, the value of m should be increased. Second, false positives entail that, from the attacker's point of view 1 in approximately 1000 attempts to inject a data frame will succeed. However, the CAN is not a randomly accessed environment, and ECU to ECU communications need to be 100% successful. In other words, failing to verify an authentication tag should have significant repercussions, at least from the accountability perspective. As such, failed authentications should be logged by ECUs and should be disclosed to the user or to relevant authorities in order to signal the presence of a malicious actor. Lastly, since the developed approach leverages a MAC function to insert an element into the BF, the attacker needs to use an oracle (e.g., a verifier ECU) in the attempt to mount a successful frame injection attack. This is a significant aspect that will force the attacker to interact with valid ECUs. Consequently, any failed authentication will disclose the presence of malicious devices, thus automatically triggering alarms and defence mechanisms [26]. A more detailed analysis of this aspect is also provided later in the experimental results section.

IV. EXPERIMENTAL RESULTS

The experimental assessment focuses on several aspects, as discussed earlier in the paper, namely: backwards compatibility with the classic CAN protocol; minimized impact upon integration into a real environment; resistance to attack vectors frequently seen in automotive networks (e.g., man-in-the-middle, replay, impersonation); and assessment of MixCAN's computation time.

A. Experimentation Environments and Implementation Details

The experiments were conducted in two distinct environments: Vector Informatik's CANoe/CANalyzer [27] software to simulate a close-to-reality CAN bus environment (denoted as *Scenario A*); and a physical test bed assembled with Raspberry Pi 3 Model B+ boards, each connected to a MCP2515 CAN controller and TJA1050 CAN transceiver, as shown in Fig. 5 (denoted as *Scenario B*).

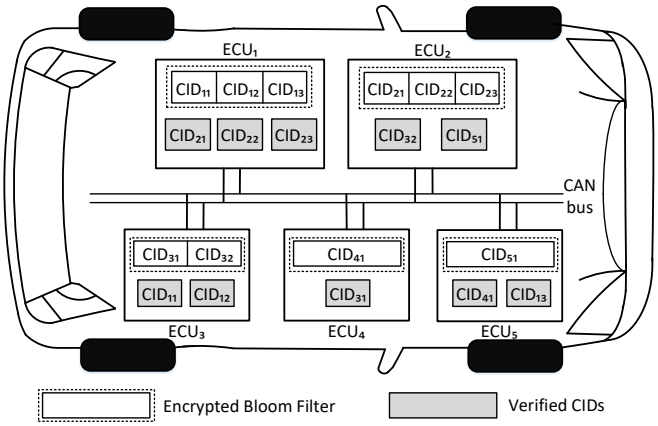

Fig. 3. Experimental setting (*Scenario A*) including ECUs, transmitted encrypted BF, and verified CIDs.

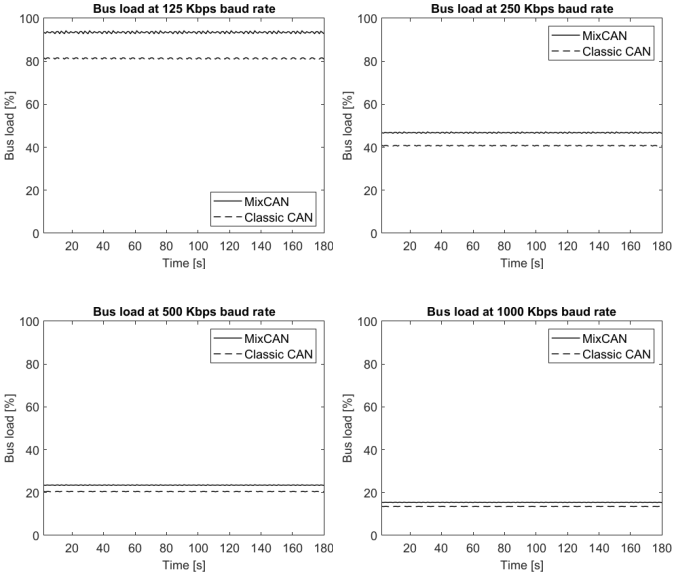

Fig. 4. MixCAN's impact on the bus load in relation to the configured baud rate.

1) Scenario A: MixCAN was integrated into CANoe as a Dynamic Link Library(DLL). As data, the set provided by Hacking and Countermeasure Research Lab (HCRL) [28], [29] was integrated into the CANoe simulation environment. The HCRL data set comprises 2 million classic CAN frames distributed across 45 CIDs. The data set originates from a KIA SOUL vehicle, and it is utilized in the *Information Security R&D Data Challenge* organized by HCRL and the Korea University from Seoul, South Korea.

Fig. 3 represents the architecture of the experiments conducted with the CANoe simulator. In the designed experiments each ECU is responsible for aggregating and constructing an EBF for a collection of CIDs (denoted by white squares), and for the verification of a set of CIDs (denoted by gray squares). The experiment was designed to emphasize the core concept behind MixCAN, that is, a given ECU is not required

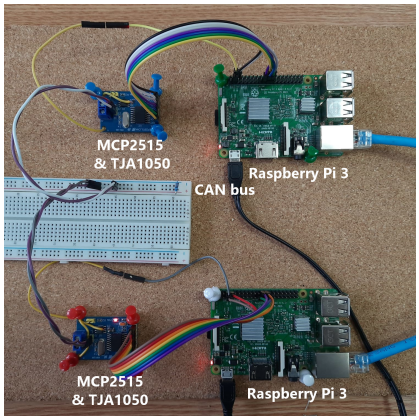

Fig. 5. Experimental test bed (*Scenario B*), including two Raspberry Pi 3 Model B+ boards as ECUs, two CAN controllers and a CAN bus (breadboard).

to authenticate every item in a EBF, but only a sub-collection of relevant items. In terms of implementation details, the size of the BF was set to 64 bits in order to ensure that the BF can be transmitted in one CAN frame. The size of the freshness counter (ctr_x) was also set to 64 bits, which ensures that the counter will practically never need to be reset and that its value fits into one CAN data frame.

For the MAC computation over a specific mix set (i.e., to obtain y_x according to Eq. (3)) the MD5 hash function was considered suitable based on the results obtained in Table I. For the MAC computation over the EBF structure and over the freshness counter the SHA-1 hash function was used, which, in accordance to the AUTOSAR specifications, was truncated to 64 bits in order to fit into a single CAN data frame. For both MAC computations, a pre-shared cryptographic key of 256-bits was used throughout the experiments. The EBF parameters were determined using equation (9), resulting in maximum 3 inserted items and 22 hash outputs per item. Note, however, that MixCAN is not restricted to this configuration, and based on Eq. (10) and (9) the EBF parameters can be tuned and set to the designer's criteria.

Each EBF aggregated maximum three CIDs, following the results from Table I, while the size of the time window W was computed for each CID such that the maximum tolerated delay for each CID (i.e., δ_i) equals two transmission time periods. In terms of transmitting the EBF, two additional CIDs have been added for each ECU: one for transmitting the EBF data structure (of 64 bits), and one for transmitting the MAC. Two additional CIDs per ECU have been added for counter synchronization.

By leveraging the HCRL data set, and the associated EBF, the transmission period for each CID was determined, and the data set (alongside the secure BF) were replayed on the CAN bus by leveraging the architecture depicted in Fig. 3.

2) Scenario B: Two Raspberry Pi 3 model B+ (RPi) boards were programmed to emulate two ECUs participating in the MixCAN protocol. Each RPi was equipped with a Quad Core 1.2GHz Broadcom BCM2837 64bit CPU with 1GB RAM. As for the underlying operating system, the Raspbian Buster Lite OS was installed.

MixCAN's main operations (construction and verification of the EBF, signature of the EBF) were implemented in the C++ language. The CAN communication was implemented with the help of the *SocketCAN* Linux package [30]. In terms of hardware, to create a CAN network each RPi board was connected to a MCP2515 CAN controller and a TJA1050 CAN transceiver. In terms of MixCAN's implementation details, the same configuration and the same hash functions were chosen for the EBF as in the case of *Scenario A*. The first ECU was programmed to continuously send CAN messages distributed across three CIDs, and to construct the EBF and SBF after five aggregated messages (from each CID). The second ECU was programmed to verify the signature of one of the received CIDs.

The main objective of this experiment was to evaluate the computational time of MixCAN's steps in a close-to-reality environment. Namely, the EBF construction time, and the item verification time have been evaluated.

B. Scenario A: Bus Measurements

The integration of cryptographic algorithms within a real-time system brings an additional computational overhead in term of storage, memory consumption, utilization of processor time and network bandwidth, which, ultimately, affects the system's baseline behavior. Given the limited resources that ECUs possess and the critical aspects in terms of communication for a automotive network, the additional impact brought by MixCAN when integrated in such a system requires a thorough analysis. Accordingly, we analyzed MixCAN's impact in terms of bus load in the context of the simulated CAN bus network, as described above.

Given that the CAN protocol's communication bandwidth is affected by the underlying hardware's baud rate, a series of experiments were conducted for baud rates of 125 Kbps, 250 Kbps, 500 Kbps, and 1000 Kbps. The measured results are summarized in Fig. 4. It can be observed that the differences between the traditional CAN measurements, compared to MixCAN, are directly proportional. The additional frames brought by MixCAN do not affect the network communications, and a significant impact is only visible in the case of lower baud rates (e.g., 125 Kbps). Nevertheless, as also summarized in Table II, the impact on the bus load does not exceed 12.61% for 125 Kbps baud rate, and 1.99% for a 1000 Kbps baud rate. Furthermore, the average increase in bus load for a typical baud rate of 500 Kbps is of 1.92%, which is negligible.

TABLE II
MIXCAN'S IMPACT ON CAN'S BUS LOAD WITH DIFFERENT BAUD RATES.

Baud Rate	Avg increase	Min increase	Max increase
125 Kbps	12.08%	11.84%	12.61%
250 Kbps	6.04%	5.92%	6.31%
500 Kbps	3.02%	2.96%	3.16%
1000 Kbps	1.92%	1.89%	1.99%

C. Scenario A: Experimental Security Assessment

The signature aggregation constitutes the main advantage that the Bloom Filter brings into MixCAN. However, due to their probabilistic nature, Bloom Filters manifest a false positive probability upon item query. From a security protocol point of view, the false positive probability can be interpreted as a vector of attack, and more precisely, as a vulnerability that a man-in-the-middle (MITM) or replay attacks can exploit. In order to assess the practical exploitation of this attack vector, an experiment was designed assuming the following: a malicious entity has direct access to the CAN bus network on which MixCAN runs; the attacker doesn't know the cryptographic key shared by the ECUs; he/she is able to sniff CAN frames (owed to CAN's broadcast communication pattern); and he/she has unlimited time to execute the attack.

Taking into account those mentioned above, an experiment using CANoe was developed consisting of two ECUs: ECU_1 and ECU_2 participating in the MixCAN protocol, and a malicious ECU, ECU_m, which tries to deceive the authentication protocol. ECU_1 periodically transmits a CAN frame every 10 ms and buffers the frames for the latter computation of the EBF. Let's denote the frames sent by ECU_1 as F_{ECU_1}. ECU_2 receives the F_{ECU_1} frames and buffers them for later authentication. After having transmitted five frames, ECU_1 computes the EBF and its MAC signature and sends $< \texttt{EBF}^{\mathcal{X}}, \texttt{SBF}^{\mathcal{X}}_{ctr_x} >$. After having received the secure Bloom Filter, ECU_2 proceeds with the authentication of the received frames.

Conducting a MITM or replay attack over a CAN bus is trivial, since one is only required to know the structure of the CAN data frame, and the frame's cycle time. Accordingly, ECU_m periodically replays (at 50 ms) a valid CAN frame (from the HCRL data set) with a CID sent by ECU_1 in the past, in order to attempt to inject a new frame into the EBF by exploiting the probabilistic nature of this data structure. Throughout our experiments, from a total of 500.000 CAN frames that have been injected by ECU_m, 70 attempts were

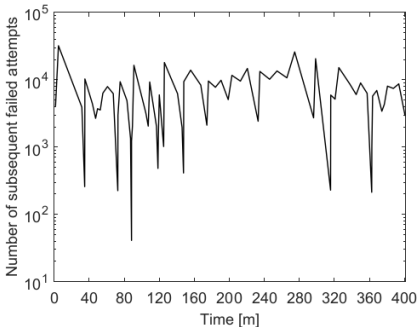

Fig. 6. Number of subsequent failed attempts to attack MixCAN before one successful frame injection.

successful, leading to 70 false data authentication. This yields a false positive rate of 0.00014, which confirms the theoretical probabilistic computations presented in Table I.

The outcome of these experiments has also been depicted in Fig. 6. While it is clear that an attacker may eventually succeed to inject a frame into MixCAN, the results summarized in this figure also exhibit a different view of such an attack. Namely, that a successful MITM attack is usually preceded by a large number of failed attempts, which result in failure to verify the EBF. According to our experiments, the number of failed attempts ranged from 41 to 32.065, which essentially means that, before managing to successfully inject one single CAN frame, an attacker needs to disturb the normal functioning of the system, thus triggering alarms. This is a significant building block in the construction of MixCAN, which relies upon the deterministic nature of CAN communications and secure frame verification, which should not fail under normal operating circumstances. However, if an attacker attempts to inject frames into a system running MixCAN, before it succeeds, the attacker will undoubtedly need to disclose his/her presence. Subsequently, a successfully detected attack attempt can trigger defence mechanisms including the reconfiguration of ECU control logic or disabling attack packets [26].

From a practical (i.e., real-world) perspective, the attack conducted in this experiment is frequently seen in the automotive sector, in terms of car tuning, or tampering attempts, that tend to alter the data sent by certain ECUs and sensors in a way that still keeps the system's core components running with false, unauthentic data. Consequently, this leads to a world-wide behavior where automotive systems are compromised almost as soon as they leave the manufacturing process [31]. For example, heavy load transport automotive vehicles are frequently tampered to report false data regarding gas consumption, thus drastically raising the amount of NOx emissions in the public environment [32]. To this end, the implementation of MixCAN can help prevent such fraudulent attempts, while ensuring a traceable (i.e., audit-enabled logging) approach that is backward compatible with existing hardware.

D. Scenario B: MixCAN Computation Assessment

Based on the experimental environment shown in Fig. 5, the execution time of the following operations was measured:

- EBF construction time: the time for constructing the Encrypted Bloom Filter, which includes: the computation of the MAC for each item inserted into the EBF; the splitting into $\lceil log_2(m) \rceil$ parts; the insertion of the hash results into the EBF; and the computation of the EBF's signature via the SHA-1 function.
- EBF verification time: the time for verifying the signature of the SBF, and for verifying the presence of an item upon receiving the tuple of messages $< \texttt{EBF}^{\mathcal{X}}, \texttt{SBF}^{\mathcal{X}}_{ctr_x} >$, for a given CID.

As shown in Fig. 7, the computation time in both cases does not exceed one millisecond. More specifically, the average time for constructing the EBF, including the computation of the SBF, is of 0.63 ms, while in the case of the verification the average computation time does not exceed 0.35 ms. The results have also been summarized in Table III. These demonstrate, once again, MixCAN's lightweight construction, and its feasible integration into today's automotive systems.

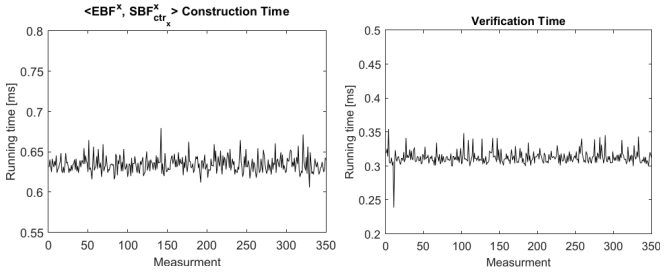

Fig. 7. MixCAN's computation time in a real test bed comprising two Raspberry Pi 3 Model B+ boards equipped with CAN communication interfaces.

TABLE III
MIXCAN'S COMPUTATION TIME.

Operation	Avg time	Min time	Max time
Construction	0.63 ms	0.60 ms	0.67 ms
Verification	0.31 ms	0.23 ms	0.35 ms

V. CONCLUSIONS

We developed a novel data authentication strategy named MixCAN for Controller Area Networks (CAN) that exhibits two major features, namely: (i) a procedure for aggregating different CAN identifiers in order to reduce the communication overhead; and (ii) an approach for mixing different data frame signatures in order to reduce the computational impact on both the signer and the verifier nodes (i.e., Electronic Control Units). In order to achieve its objectives, MixCAN leverages the features of Bloom Filters, and symmetric key cryptography (i.e., Message Authentication Codes). As a result, MixCAN exhibits a lightweight construction, and is backward compatible with classical CAN communications. While an attacker may attempt to exploit the probabilistic nature of Bloom Filters, as demonstrated by the experimental result, one successful data frame injection is usually preceded by a large number of failed attempts. Each such failed attempt can

subsequently disclose the presence of the attacker and lead to the automated triggering of defence mechanisms. Lastly, we mention that the experimental results also demonstrated the feasible application of the developed approach in a close-to-reality simulation environment comprising CAN frames from a KIA SOUL vehicle, as well as in a real test bed consisting of Raspberry Pi systems equipped with CAN communication modules. In terms of future improvements, we envision enriching MixCAN with a lightweight key distribution protocol. Furthermore, we consider that significant research needs to be done to develop new defence mechanisms that would be activated once an attack attempt is discovered. Such techniques should have a limited impact on the normal functioning of in-vehicle communication systems and on the ECUs in order to ensure that malicious actors do not exploit these defense mechanisms.

ACKNOWLEDGEMENT

This work was funded by the European Union's Horizon 2020 Research and Innovation Programme through DIAS project (*https://dias-project.com/*) under Grant Agreement No. 814951. This document reflects only the author's view and that the Agency is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] R. Coppola and M. Morisio, "Connected Car: Technologies, Issues, Future Trends," *ACM Comput. Surv.*, vol. 49, no. 3, pp. 46:1–46:36, Oct. 2016. [Online]. Available: http://doi.acm.org/10.1145/2971482

[2] ISO, "ISO 11898-1:2003 - Road vehicles - Controller area network (CAN) - Part 1: Data link layer and physical signalling," *International Organization for Standardization*, 2003.

[3] T. Ziermann, S. Wildermann, and J. Teich, "CAN+: A new backward-compatible Controller Area Network (CAN) protocol with up to 16 higher data rates." in *2009 Design, Automation Test in Europe Conference Exhibition*, April 2009, pp. 1088–1093.

[4] Robert Bosch Gmbh, "Can with flexible data-rate," *Vector CANtech, Inc., MI, USA, Specification Version 1.0*, 2012.

[5] C. Urquhart, X. Bellekens, C. Tachtatzis, R. Atkinson, H. Hindy, and A. Seeam, "Cyber-Security Internals of a Skoda Octavia vRS: A Hands on Approach," *IEEE Access*, vol. 7, pp. 146 057–146 069, 2019.

[6] Y. Takefuji, "Connected Vehicle Security Vulnerabilities [Commentary]," *IEEE Technology and Society Magazine*, vol. 37, no. 1, pp. 15–18, March 2018.

[7] A. Van Herrewege, D. Singelee, and I. Verbauwhede, "CANAuth - A Simple, Backward Compatible Broadcast Authentication Protocol for CAN bus," in *ECRYPT Workshop on Lightweight Cryptography 2011*, ser. ECRYPT '11, 2011, pp. 1–7.

[8] A.-I. Radu and F. D. Garcia, "LeiA: A Lightweight Authentication Protocol for CAN," in *Computer Security – ESORICS 2016*, I. Askoxylakis, S. Ioannidis, S. Katsikas, and C. Meadows, Eds. Cham: Springer International Publishing, 2016, pp. 283–300.

[9] B. Groza and P. Murvay, "Security Solutions for the Controller Area Network: Bringing Authentication to In-Vehicle Networks," *IEEE Vehicular Technology Magazine*, vol. 13, no. 1, pp. 40–47, March 2018.

[10] 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*, vol. 17, no. 8, pp. 2248–2261, Aug 2016.

[11] B. Groza and P.-S. Murvay, "Identity-Based Key Exchange on In-Vehicle Networks: CAN-FD & FlexRay," *Sensors*, vol. 19, no. 22, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/22/4919

[12] AUTOSAR, "Specification of Secure Onboard Communication AUTOSAR CP Release 4.3.1," *AUTOSAR*, 2017.

[13] L. Wang and X. Liu, "NOTSA: Novel OBU With Three-Level Security Architecture for Internet of Vehicles," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3548–3558, Oct 2018.

[14] A. Perrig, R. Canetti, J. D. Tygar, and Dawn Song, "Efficient authentication and signing of multicast streams over lossy channels," in *Proceeding 2000 IEEE Symposium on Security and Privacy. S P 2000*, May 2000, pp. 56–73.

[15] B. Groza and S. Murvay, "Efficient Protocols for Secure Broadcast in Controller Area Networks," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2034–2042, Nov 2013.

[16] J. Contreras-Castillo, S. Zeadally, and J. A. Guerrero-Ibañez, "Internet of Vehicles: Architecture, Protocols, and Security," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3701–3709, Oct 2018.

[17] K. Kang, Y. Baek, S. Lee, and S. H. Son, "An Attack-Resilient Source Authentication Protocol in Controller Area Network," in *2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, May 2017, pp. 109–118.

[18] R. Hussain, F. Hussain, and S. Zeadally, "Integration of VANET and 5G Security: A review of design and implementation issues," *Future Generation Computer Systems*, vol. 101, pp. 843 – 864, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X19306909

[19] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970. [Online]. Available: http://doi.acm.org/10.1145/362686.362692

[20] S. M. Bellovin and W. R. Cheswick, "Privacy-Enhanced Searches Using Encrypted Bloom Filters," 2004, under submission smb@research.att.com 12449 received 1 Feb 2004. [Online]. Available: http://eprint.iacr.org/2004/022

[21] A. Duka, B. Genge, and P. Haller, "Enabling authenticated data exchanges in industrial control systems," in *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, March 2018, pp. 1–5.

[22] M. Mitzenmacher, "Compressed Bloom filters," *IEEE/ACM Transactions on Networking*, vol. 10, no. 5, pp. 604–612, Oct 2002.

[23] P. Bose, H. Guo, E. Kranakis, A. Maheshwari, P. Morin, J. Morrison, M. Smid, and Y. Tang, "On the false-positive rate of Bloom filters," *Information Processing Letters*, vol. 108, no. 4, pp. 210 – 213, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020019008001579

[24] K. Christensen, A. Roginsky, and M. Jimeno, "A new analysis of the false positive rate of a Bloom filter," *Information Processing Letters*, vol. 110, no. 21, pp. 944 – 949, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020019010002425

[25] Q. H. Dang, "Recommendation for Applications Using Approved Hash Algorithms, Special Publication (NIST SP) - 800-107 Rev 1," 2012. [Online]. Available: https://www.nist.gov/publications/recommendation-applications-using-approved-hash-algorithms

[26] H. Kwon, S. Lee, J. Choi, and B. Chung, "Mitigation mechanism against in-vehicle network intrusion by reconfiguring ECU and disabling attack packet," in *2018 International Conference on Information Technology (InCIT)*, Oct 2018, pp. 1–5.

[27] Vector Informatik, "CANoe v 12.0: Testing ECUs and Networks with CANoe," 2020, last access: January 6th, 2020. [Online]. Available: https://www.vector.com/int/en/products/products-a-z/software/canoe/

[28] H. Lee, S. H. Jeong, and H. K. Kim, "Otids: A novel intrusion detection system for in-vehicle network by using remote frame," in *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, vol. 00, Aug 2017, pp. 57–5709. [Online]. Available: doi.ieeecomputersociety.org/10.1109/PST.2017.00017

[29] S. H. J. Hyunsung Lee and H. K. Kim, "CAN Dataset for intrusion detection (OTIDS)," 2018, last access: January 6th, 2020. [Online]. Available: http://ocslab.hksecurity.net/Dataset/CAN-intrusion-dataset

[30] O. Hartkopp, et al., "SocketCAN: Controller Area Network Protocol Family," 2020, last access: January 6th, 2020. [Online]. Available: https://github.com/linux-can/can-utils

[31] U. EPA, "DOJ. EPA Announce One-Billion-Dollar Settlement with Diesel Engine Industry for Clean Air Violations. News Release, 22 October 1998," 1998, last access: January 6th, 2020. [Online]. Available: https://archive.epa.gov/epapages/newsroom_archive/ newsreleases/93e9e651adeed6b7852566a60069ad2e.html

[32] D. Pöhler, T. Adler, C. Krufczik, M. Horbanski, J. Lampel, and U. Platt, "Real Driving NOx Emissions of European Trucks and Detection of Manipulated Emission Systems," in *EGU General Assembly Conference Abstracts*, ser. EGU General Assembly Conference Abstracts, Apr 2017, p. 13991.