CAIBA: Multicast Source Authentication for CAN Through Reactive Bit Flipping

Eric Wagner[*†], Frederik Basels[*], Jan Bauer[*], Till Zimmermann[‡], Klaus Wehrle[†], and Martin Henze[§*]

[*]*Cyber Analysis & Defense*, Fraunhofer FKIE · {firstname.lastname}@fkie.fraunhofer.de

[†]*Communication and Distributed Systems*, RWTH Aachen University · {lastname}@comsys.rwth-aachen.de

[‡]*Distributed Systems Group*, Osnabrück University · zimmermann@uos.de

[§]*Security and Privacy in Industrial Cooperation*, RWTH Aachen University · henze@spice.rwth-aachen.de

Abstract—Controller Area Networks (CANs) are the backbone for reliable intra-vehicular communication. Recent cyberattacks have, however, exposed the weaknesses of CAN, which was designed without any security considerations in the 1980s. Current efforts to retrofit security via intrusion detection or message authentication codes are insufficient to fully secure CAN as they cannot adequately protect against masquerading attacks, where a compromised communication device, a so-called electronic control units, imitates another device. To remedy this situation, multicast source authentication is required to reliably identify the senders of messages. In this paper, we present CAIBA, a novel multicast source authentication scheme specifically designed for communication buses like CAN. CAIBA relies on an authenticator overwriting authentication tags on-the-fly, such that a receiver only reads a valid tag if not only the integrity of a message but also its source can be verified. To integrate CAIBA into CAN, we devise a special message authentication scheme and a reactive bit overwriting mechanism. We achieve interoperability with legacy CAN devices, while protecting receivers implementing the AUTOSAR SecOC standard against masquerading attacks without communication overhead or verification delays.

Index Terms—multicast source authentication, CAN bus, message authentication codes, AUTOSAR SecOC

1. Introduction

Virtually all motor vehicles currently on the roads are equipped with hundreds of small embedded computers, so-called Electronic Control Units (ECUs), that monitor and control vital vehicle functions [34]. To realize overarching functionality, these ECUs require means to communicate reliably with each other in harsh environments. The de-facto standard for such in-vehicle interconnection is the Controller Area Network (CAN) bus. Developed in the 1980s under the assumption of vehicles being physically isolated systems, security was not a design goal of CAN.

However, contrary to this original assumption, modern cars offer more and more (wireless) connectivity and are thus increasingly exposed to cyberthreats with potentially fatal consequences [13]. For example, a remote compromise of the infotainment system of a Tesla Model S enabled attackers to control the car's acceleration, leaving potential passengers at the attacker's mercy [54]. This example is not an isolated incident, but only one of many recent attack demonstrations [2], [13], [17], [20], [50], [54],

[55], [79]. Those attacks exploit that, once an ECU has been compromised, CAN provides no protection against ECU masquerading and arbitrary message spoofing [38]. Concerningly, the dark web offers car theft devices today that exploit the vulnerability of CAN [2]. These devices are actively used by criminals to steal modern cars [2].

To overcome these serious vulnerabilities, different streams of research propose *intrusion detection*, *covert channels*, as well as *cryptographic approaches* [4]. While Intrusion Detection Systems (IDSs) [42] do not interfere with legacy devices, they provide imperfect detection and often only alert after a sequence of manipulations [68]. Moreover, recent research has shown how these IDSs can be evaded by an attacker in practice [8], [65]. Covert channels [24], [49], [57], [68] only provide reduced security and cannot protect against masquerading attacks. Cryptographic approaches [46], also comprising the AUTOSAR specification for Secure Onboard Communication (SecOC) [5] currently in adoption by multiple vendors [34], mostly rely on group-keys to protect against outside attackers. AUTOSAR SecOC reserves a fraction of each CAN payload, *e.g.*, $28\,\mathrm{bit}$, to transmit an authentication tag computed with the help of a secret key shared among all ECUs. These approaches thus do not protect against compromised ECUs (as used during *e.g.*, the Tesla Model S attack [54]) as those can still masquerade as any entities they can receive from.

The main challenge underlying all these approaches is the need for both reliable and immediate protection of multicast communication. Most notably, CAN is a broadcast communication protocol, where messages are often intended for multiple receivers [23]. Typically, to verify the source of a message in a multicast scenario, digital signatures are used as authentication and verification rely on different keys. However, digital signatures are not applicable in CAN due to excessive computational and bandwidth requirements [4]. CAN frames hold at most $8\,\mathrm{bytes}$ of payload and even the up to $64\,\mathrm{bytes}$ of the CAN-FD extension are insufficient to support asymmetric cryptography. An alternative for source authentication in multicast communication relying on more resource-conscious symmetric cryptography is TESLA [60]. TESLA takes advantage of delayed key releases which, however, leads to verification delays and is thus not applicable to the safety-critical operation of CAN [4]. To fully protect CAN against the potentially fatal consequences of a compromised ECU, a multicast source authentication scheme is needed that only occupies a few bytes per message, yet
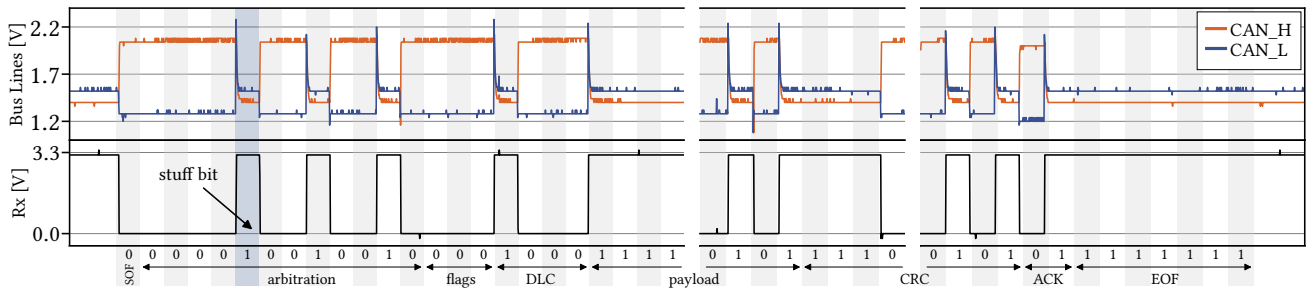
Figure 1: To transmit data, CAN uses differential encoding on two wires, CAN_H and CAN_L. We display a base format CAN frame with 8 bytes of payload as used to control hunreds of millions of cars every day. Note the gap in the payload and CRC field for increased readability.

delivers instantaneous source verification.

In this paper, we first address the general lack of adequate multicast source authentication schemes by proposing Compact And Instantaneous Bus Authentication (CAIBA). To the best of our knowledge, CAIBA is the first multicast source authentication scheme without verification delay while relying on tag sizes as small as traditional Message Authentication Codes (MACs) with the same security strength. CAIBA can cryptographically protect any network against masquerading attacks as long as it is possible to place a node that can listen to and overwrite all communications. In a nutshell, CAIBA introduces a dedicated entity, the so-called *authenticator* to the bus which is tasked with modifying a message's MAC tag at line rate such that receivers can verify the source *and* integrity of a message. Meanwhile, the authenticator cannot generate valid tags itself.

We show how CAIBA can be integrated into CAN while offering full backwards compatibility and thus allowing for incremental deployment and operation alongside legacy CAN. CAIBA could thus be deployed in the same manner as other extensions to CAN, such as CAN-FD or the newly standardized CAN-XL. Most importantly, existing receivers implementing AUTOSAR SecOC [5] can benefit from CAIBA's source authentication without any modifications, as only senders need to support CAIBA.

Contributions. To bring multicast source authentication to CAN, we make the following contributions:

- We propose CAIBA, a multicast source authentication scheme specifically designed for bus communication with short authentication tags and no verification delay.
- We enable the calculation of MAC tags by CAIBA's authenticator at line rate by adapting the novel BP-MAC [75] scheme, whose security is based on AES.
- We design a *reactive* bit flipping mechanism that enables dynamic overwriting of both recessive and dominant CAN bits after preemptively assessing their states.
- We prototypically integrate CAIBA into CAN on a software-defined CAN controller [11] and show backward-compatibility with real ECUs as well as reliability of over $99.99\,\%$ (similar to standard CAN) even for bus lengths of $100\,\mathrm{m}$ and beyond.

2. Background: CAN's Physical Layer

The CAN bus protocol is the de-facto standard in the automotive industry and mandatory in the EU for vehicle diagnostics and nowadays also finds applications in, *e.g.*, industrial automation [41]. To realize CAIBA, we need to dive into some of the less well-known details of CAN's physical layer, which we outline in the following.

Signaling. Physically, the CAN bus is based on two wires, CAN_H (CAN high) and CAN_L (CAN low), to which every ECU is connected [29]. At the physical layer, bits are encoded using a simple non-return-to-zero (NRZ) scheme via different voltage levels specifying the following differential coding [31]:

$$\begin{cases} 0 \text{ (dominant bit)} & \text{if } CAN\_H - CAN\_L \geq 0.9\,V \\ 1 \text{ (recessive bit)} & \text{if } CAN\_H - CAN\_L \leq 0.5\,V \end{cases}$$

We exemplify the resulting differential signaling in Figure 1. The transmitter only actively applies the voltage difference for dominant bits, while the recessive voltage levels resulting in a difference are passively created by resistors within each transceiver. Due to this wired AND gate (bus lines are recessive only when all transmitters transmit a recessive bit), dominant bits (0) always overwrite recessive bits (1) [29]. This property is utilized on the data link layer for arbitration and error handling.

Identifier and Data Frame Format. Two *data frame* formats with different identifier (ID) schemes for addressing are standardizes [29]. The base frame consists of an 11-bit ID with up to $8\,\mathrm{bytes}$ of payload, three control flags (3 bits), the length field (4 bits), the CRC (16 bits), and the acknowledgement (ACK) field (2 bits) as shown in Figure 1. The extended frame format offers a 29-bit ID. Both data frames are enclosed in a start-of-frame (SOF) and an end-of-frame (EOF) delimiter, which are signaled by one dominant and 7 recessive bits, respectively.

Sampling and Synchronization. The fixed duration for which a single bit is present on the wire is called the *nominal bit time* and depends on the used data rate. Within this bit time, each signal is split into four time segments that consist of one or more *time quanta* which are derived from the ECU's clock and usually are configurable by a programmable prescaler [29]. The first segment is exactly one time quanta long and is used to synchronize the different ECUs. The second segment compensates for signal propagation and its length, in combination with the bit rate, are thus the main contributors to determining the maximal bus length. The last two segments are chosen such that the bit sampling, happening between them, is located as close as possible to $75\,\%$ of the nominal bit time [18]. These two segments can also be elongated

or shortened for resynchronization based on continuous monitoring of the edges of the voltage levels.

Bit Stuffing. With NRZ coding, a certain number of consecutive bits of equal value leads to an absence of edges necessary for resynchronization, which is addressed by *bit stuffing*. To prevent that more than five identical bits appear in sequence in a bit stream, an additional bit with the inverse level is inserted (*stuffed*) after five identical bits (cf. Figure 1). Stuff bits are therefore transparently added and removed during transmission by the transmitting and receiving CAN controllers.

3. State-of-the-Art on Securing CAN

Historically, the CAN bus protocol offers no protection against cyberattacks. Recent real-world demonstrations [2], [13], [17], [20], [50], [54], [55], [79] have shown that this lack of security enables the remote compromise of entire vehicles. A famous example takes full control of Tesla Model S by remotely attacking the car's multimedia system and propagating from there through the CAN bus [54]. Consequently, new protection mechanisms are needed to be prepared for the ever-increasing digitalization and interconnection of modern cars.

3.1. Threat Model

We consider a threat model that corresponds to that chosen in the majority of the recent offensive and defensive research on CAN security [6], [8], [13], [20], [46], [50], [54]–[56], [61], [80]: We assume that an attacker has either physical access to the bus to implant a malicious ECU or has compromised an existing ECU, *e.g.*, through Bluetooth or other connectivity. The attacker then intends to inject messages beyond those message types needed for the functionality of the compromised ECU (an implanted malicious ECU is not supposed to send any messages), *i.e.*, masquerading as another ECU, to highjack the car.

3.2. Related Work

Research on protecting CAN against malicious ECUs can be grouped into three categories: *intrusion detection systems (IDSs)*, *covert channels*, and *cryptographic approaches*. These approaches make certain trade-offs to achieve alleged security, which result in certain limitations, vulnerabilities, or overhead. We classify the current state-of-the-art in protecting CAN according to these drawbacks as introduced in the following.

Limitations

⬈ Only point-to-point communication is protected, whereas broadcast communication, *i.e.*, CAN messages meant for multiple receivers, are not protected.

⊹ Only broadcast messages with few (*e.g.*, < 5) receivers are protected, where the level of protection exponentially reduces with the number of receivers.

⟳ Definitively lost CAN frames lead to desynchronizations that cannot be recovered from.

⋰ The protocol deterministically delays message authenticity verification by buffering messages at either the sender or receiver.

Vulnerabilities

⏻ The physical disconnection of a single ECU (*e.g.*, an IDS) covertly disables all protection mechanisms in a way that is unnoticeable by any other ECU.

∞ An attacker that compromises a single CAN ECU can subsequently masquerade as other ECUs, *e.g.*, because all ECUs share a group key. Needing to compromise a dedicated security node (*e.g.*, IDSs) is considered more secure, as those offer no outside connectivity and should be temper-resilient.

↻ An attacker can intercept a frame and use the information gained to impersonate another device, *e.g.*, by replaying the frame at a later time. Some approaches may only enable the injection of malicious frames shortly after the original intercepted frame.

⬃ The detection of malicious messages only happens retroactively after messages have been processed and potentially caused significant harm.

Overhead

⬓ The scheme needs excessive storage overhead, *e.g.*, to store a secret key for each other ECU.

⬚ The scheme requires some additional communication overhead, either by reserving some space in each frame or even transmitting additional CAN frames.

⧗ The transmission of CAN frames is delayed or prioritization is not fully adhered to.

We consolidate our analysis of the state-of-the-art on protecting CAN in Table 1. Our classification is partially based on a recent survey by Lotto *et al.* [46] on weaknesses in CAN authentication protocols. For the five proposals classified as *secure* by Lotto *et al.* and seven proposals not considered in their survey, we present a detailed analysis in Appendix A to substantiate our classification. In the following, we give an overview of the three categories of CAN protection mechanisms.

IDSs add a dedicated device to the network that monitors the physical characteristics and behavior of all ECUs to detect imposters. This monitoring can be based on voltage levels [15], [16], [19], [36], message timings [14], [53], [64], [66], [69], [70], [76], [81], signal characteristics [35], [45], [53], or device behavior [44], [52]. All methods have in common that the IDS device can be disrupted to covertly disable security without this being noticed ⏻. Moreover, IDSs often only detect malicious message flows rather than achieving single-message detection, such that timely reactions, *e.g.*, discarding messages, are hardly possible ⬃ [68]. Finally, IDSs have no perfect detection performance, *i.e.*, malicious messages may not get recognized (false negative) or genuine traffic falsely gets flagged (false positive). Even low false positive rates can have detrimental effects if acted upon, given the amount of genuine CAN messages being constantly sent. On top, recent research shows how sophisticated attacks can also deliberately evade IDSs in CAN [8], [65]. Overall, we conclude that IDS may be useful, but should not be the only line of defense due to their limitations.

A second class of CAN security approaches relies on *covert channels* to achieve message authentication. These covert channels can be created through a high-frequency signal interlaced with regular messages [24],

		Limitation				Vulnerability				Overhead		
Protocol	**Year**	⬈	⸬	⟳	⋰	⏻	∞	↻	✖	▤	⬓	⧖
IDSs [14]–[16], [19], [35], [36], [44], [45], [52], [53], [64], [66], [69], [70], [76], [81]	2016-2023	-	-	-	-	◆	◇	-	◇	-	-	-
Covert Channels												
CANTO [24]	2020	-	-	-	-	-	◆	-	-	-	-	◆
Watermarking [49]	2022	-	-	-	-	-	◆	-	-	-	-	-
ZBCAN [68]	2023	-	-	-	◇	◆	-	◇	-	-	-	◇
CAN-MM [57]	2024	-	-	-	-	-	◆	-	-	-	-	-
Cryptographic Approaches												
AUTOSAR SecOC [5]	2020	-	-	-	-	-	◆	-	-	-	◇	-
CANAuth [74]	2011	-	-	-	-	-	◆	-	-	◆	◇	-
Car2X [67]	2011	-	-	-	◆	-	-	◆	-	◇	◆	-
LiBrA-CAN [23]	2012	-	◆	-	◆	-	◇	-	-	-	◆	-
LinAuth [43]	2012	-	◆	-	-	-	-	-	-	◆	◆	-
LCAP [26]	2012	-	-	-	-	-	-	◆	-	◆	◆	-
CaCAN [39]	2014	-	-	◆	-	◆	◇	-	-	-	◇	-
Woo-Auth [80]	2014	-	-	◆	◇	-	◆	-	-	◇	◇	-
VeCure [77]	2014	-	-	◆	◇	-	◆	-	-	◆	◆	-
LeiA [63]	2016	-	-	-	◆	-	◆	-	-	◆	◆	-
vatiCAN [56]	2016	-	-	◇	◆	-	◆	◇	-	◇	◇	-
VulCAN [73]	2017	-	-	-	◆	-	◆	-	-	◆	◆	-
TOUCAN [6]	2019	-	-	-	-	-	◆	-	-	-	◇	-
LEAP [47]	2019	◆	-	-	-	-	-	-	-	◆	◇	-
CAN-TORO [25]	2020	-	-	-	-	-	◇	-	-	◆	-	-
MAuth-CAN [32]	2020	-	-	◇	◆	◆	◇	-	-	-	◇	-
AuthentiCAN [48]	2020	◆	-	-	-	-	-	-	-	◆	◆	-
S2-CAN [61]	2021	-	-	-	-	-	◆	-	-	-	◇	-
Caiba	2025	-	-	-	-	-	-	-	-	-	◇	-

Limitation: ⬈ no broadcasting support ⸬ limited scalability ⟳ no resynchronization ⋰ delayed verification   ◆ full   ◇ partial
Vulnerability: ⏻ covertly disconnected device ∞ masquerading by other ECU ↻ frame interception ✖ delayed alarms
Overhead: ▤ storage overhead ⬓ communication overhead ⧖ delayed transmission

TABLE 1: Current proposals to protect CAN traffic expose weaknesses that make them either not deployable in cars (*e.g.*, scalability limitations) or offer attack vectors to malicious actors (*e.g.*, no protection against masquerading).

[49], [57], which however requires dedicated transceiver hardware for decoding and still does not provide source authentication ∞. ZBCAN [68], on the other hand, proposes a unique inter-frame spacings for each sender, only known to the sender and a central authority. In case of anomalies, the central authority jams the suspicious frame. However, in ZBCAN, it is neither detectable that the central authority is covertly disconnected ⏻ nor is the protocol secure against bit modification attacks [40] as it only verifies that the sender intended to send at a given time but not the content of the message ↻. Moreover, ZBCAN modifies CAN's prioritization scheme which can lead to a significant delay of up to 25 ms ⋰.

Finally, *cryptographic approaches* rely on the integration of integrity-protecting tags, usually MACs, into data frames. These tags can be transmitted *e.g.*, as a portion of the payload [5], as a substitution for the cyclic redundancy check (CRC) checksum [80], as part of the CAN ID [25], or in an additional frame [56]. A prominent example is the AUTOSAR SecOC standard [5] that uses part of the payload, *e.g.*, 28 bit, for the transmission of integrity protection. Like AUTOSAR SecOC, most of these approaches rely on *group keys* to compute integrity tags. The resulting lack of source authentication means that compromised ECUs are not restricted from simply impersonating other devices ∞. The few exceptions split the limited space for integrity protection among all receivers ⸬ [23], [43], do not support broadcast communication ⬈ [47], [48], rely

on a covertly disconnectable authenticator ⏻ [32], [39], or are vulnerable to frame interceptions ↻ [26], [67].

Concluding, we observe that all current proposal to protect CAN suffer from drawbacks that either make them unsuitable for in-vehicle communication or make them vulnerable to attacks. With Caiba, we thus aim to design a protocol that suffers none of these drawbacks.

4. Source Authentication with Caiba

For traditional point-to-point communication, source authentication and integrity protection is realized by appending an integrity-protecting authentication tag (*i.e.*, a MAC) to each message. This tag is computed with the help of a secret key shared by sender and receiver, and verified by the receiver upon reception of a message. However, in a broadcasting domain like CAN, such authentication tags do not provide source authentication: A receiver cannot differentiate if a message stems from the alleged sender or if it has been spoofed by another member in the group of receivers who has access to the secret key. Hence, a compromised device could masquerade as one or multiple other devices. To protect the CAN bus against masquerading attacks, an effective source authentication protocol for multicast communication is thus required. In the following, we first investigate existing proposals to achieve source authentication and argue why they are
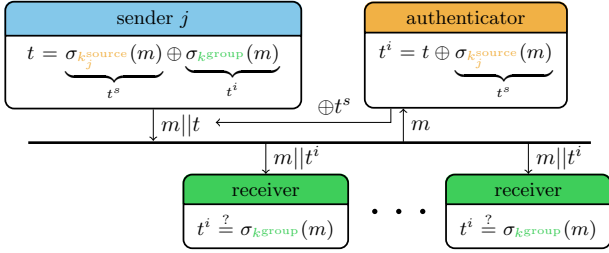
Figure 2: CAIBA employs an integrity-protecting tag t^i and a source-authenticating tag t^s that are computed by the MAC function σ using the source and group keys, k_j^{source} and k^{group}, and are XORed by the sender to protect a message m. During transmission, the authenticator computes t^s and overwrites the message such that only t^i remains. The integrity-protecting tag t^i that is read by the receivers can then be verified with a conventional group key without knowledge of the source key.

not suitable for CAN. Afterward, we propose our novel scheme, CAIBA, that addresses limitations of prior work.

4.1. The Current State-of-the-Art

Source authentication for multicast communication relies on asymmetry between senders and receivers or across receivers [12]. The most widespread form of asymmetry are digital signatures, where receivers only have keys to verify messages, but not authenticate them. The main concern for this type of *cryptographic asymmetry* is the size of signatures (upwards of 32 bytes) that cannot fit into CAN frames. The TESLA protocol [59], [60], on the other hand, relies on *time asymmetry* for multicast source authentication. Here, keys are revealed after they have expired and linked to the sender (*e.g.*, via a hash-chain), such that receivers can verify the authenticity of buffered messages retroactively. Time asymmetry does however introduce significant delays and some additional bandwidth overhead (*e.g.*, for key revelation) which are not acceptable in in-vehicular communication. The final type of multicast source authentication schemes relies on *information asymmetry*, where receivers can only verify parts of the multiplexed source-verifying information [10], [58]. Thus, each receiver has reduced certainty of a message's authenticity, with the benefit that no single receiver can generate an authentication tag that is accepted by all receivers. This, however comes at the cost of longer authentication data (compared to simple MAC schemes), which grows with the number of receivers. To conclude, current multicast source authentication schemes are inapplicable to CAN: they either incur verification delay or require excessive message overhead (in CAN, payload and authentication tag must fit into 8 bytes).

4.2. General Idea of CAIBA

To enable multicast source authentication for CAN, we devise the novel CAIBA scheme. To understand the idea behind CAIBA, consider the following hypothetic scenario. A CAN message is protected by two authentication tags, the *integrity-protecting tag* t^i and the *source-authenticating tag* t^s. These tags are computed with tra-

ditional MAC schemes, where the tag verification corresponds to the re-computation of the tag based on the received data and comparison to the received tag. The integrity-protecting tag t^i is generated using a group key k^{group} and verified by each receiver. Like intrusion detection systems and other proposals to retrofit integrity protection into CAN [23], [39], [68], CAIBA relies on one or multiple dedicated security nodes, the authenticator(s). The source-authenticating tag t^s is only verifiable by this authenticator that shares a symmetric key k_j^{source} with each sender j, but does not know the group key. In this hypothetic scenario, the authenticator also controls a virtual side-channel to securely notify each receiver whether or not the source of a message has been correctly verified.

Obviously, this basic construction occupies additional space (to send two tags instead of one) and time (to wait for the confirmation of the source-authenticating tag t^s verification to receivers). However, in CAIBA, we only transmit a single tag and use an implicit side-channel as shown in Figure 2: The actual transmitted tag t is the aggregation of both tags, *i.e.*, $t = t^i \oplus t^s$.[1] The authenticator no longer verifies t^s but only recomputes t^s based on the received payload and the alleged sender. The authenticator then XORs the tag t with t^s as the bits are transmitted over the bus. When the ordinary receivers in CAN sample the bus, they will then read and verify t^i. If, and only if, neither the data nor the tag has been manipulated beyond the authenticator's actions, the receiver's integrity verification can be successful. In this way, no explicit side-channel communication is needed and only the space of a single authentication tag is occupied in the CAN payload.

4.3. Requirements to Deploy CAIBA

Since CAIBA introduces no verification delay or message overhead compared to an ordinary group key based authentication, our scheme is especially attractive for the CAN bus. However, as other solutions for multicast source authentication, CAIBA can only be applied if certain requirements are fulfilled. Specifically, the deployment scenario must fulfill certain requirements.

No Direct Communication. Entities must not be able to directly communicate with each other without the authenticator also overhearing these transmissions. Otherwise, a malicious group member could only append the integrity-protecting tag t^i and a receiver would have no way of knowing whether the message really stems from the alleged transmitter. Communication buses like CAN fulfill this requirement by design.

Ability to Overwrite Messages. The authenticator must be able to reliably overwrite messages at line rate. We later show how this is possible for CAN (*cf.* Section 5.3). For other communication protocols, it still has to be investigated how to unlock such capabilities.

No Authenticator Collusion. There must be no collusion between the authenticator and any other entity in CAIBA. Otherwise, the entity could transmit a message without a source-authenticating tag and the colluding authenticator would simply ignore it. Practical demonstrations have shown how to compromise individual CAN

---

1. The aggregation with XOR is provably secure and t can be verified by recomputing t^i and t^s individually [7].

ECUs due to external connectivity. However, additionally compromising an authenticator with no external connectivity that can rely on temper-resilient hardware would require a significantly more advanced attack.

4.4. Security Discussion

As we proof in Appendix B, CAIBA provides the same security as a traditional MAC of the given length used to protect an end-to-end connection between sender and receiver. Considering an exemplary $24\,\text{bit}$ tag, an attacker thus has a mere $1/2^{24}$ (~1 in $17\,\text{million}$) chance to guess a valid tag. This security level is achieved because the aggregation of the source-authenticating tag and the integrity-protecting tag achieves the same security level as the individual tags [33]. The difference is that the aggregated tag can only be verified by combining the knowledge to verify both tags individually. As only the sender knows the keys for both tags, a valid tag authenticates the source of a message to each receiver.

One security risk for CAIBA is that an attacker physically disconnects the authenticator from the network. With other security solutions, *e.g.*, IDSs in CAN, such attacks are often not detectable, and the attacker would gain complete control over the bus. In contrast, CAIBA quickly detects that the authenticator is disconnected and can take corrective actions before significant damage can be caused. Ideally, redundant authenticators are available, which can quickly take over upon request by the sender that noticed an inactive authenticator. Alternatively, operators can be notified about the inoperational authenticator and respond similarly as to an alarm by an IDS. Luckily, an authenticator's physical disconnection most likely occurs while the car is stationary, such that a car could be prevented from moving, at least until the driver is notified about the risk. If the authenticator nonetheless disconnects while the car is moving, CAIBA could fall back to insecure CAN, advise the driver to stop the car, and potentially the speed and acceleration of the car could be limited.

Finally, an attacker could compromise the authenticator directly, *e.g.*, through a supply chain attack. However, even a compromised authenticator cannot spoof messages as it cannot compute integrity-protecting tags. To successfully spoof a message, an attacker needs to additionally compromise a genuine receiver within the corresponding multicast group. Thus, CAIBA overall provides strong protection against masquerading attacks and enables the detection of a disconnected authenticator.

5. Integrating CAIBA into the CAN Bus

After presenting the idea of CAIBA in a general and abstract fashion, we now discuss the technical details of integrating CAIBA into CAN. Here, we are faced with two main challenges. First, the authenticator must be able to quickly compute the source-authenticating tag t^s as the first bit of the overwritten tag follows immediately after the last bit of the protected payload. Secondly, the authenticator must be able to dynamically and precisely flip individual bits. In the following, we will successively introduce the design of all entities, *i.e.*, the transmitter, the authenticator, and the receiver. Before, we shortly address

the general challenges of deploying CAN extensions such as CAIBA. Concerning key distribution, we assume that each ECU and the authenticator are initially configured with exactly those keys that they require, *i.e.*, each ECUs shares a unique key with the authenticator, and all ECUs know relevant group keys[2].

5.1. Deployment Considerations

CAN extensions such as CAN-FD for higher bandwidth have in the past been successfully deployed by ensuring interoperability with legacy CAN devices. Thus, while ECUs do not necessarily need to be fully compliant with the existing CAN standard, any changes must by compatible with the existing standard, such that CAN ECUs and CAIBA ECUs can operate on the same network. Only then can an incremental deployment of CAIBA be possible, as the sudden adoption of a new standard by all actors in, *e.g.*, a car manufacturing supply chain, is not realistic. The easiest solution to achieve such interoperability is if all changes are restricted to the inner operation of an individual ECU, while the signals written to the bus are fully compliant to the CAN standard at all times.

Regarding the actual integration and commercialization of Caiba into *e.g.*, cars, ECU and car manufacturers do not have to change much in their operations. Nowadays, CAN controllers are mostly integrated as Semiconductor Intellectual Property (SIP) cores (e.g., [1], [3]) Once an SIP core implements Caiba, ECU manufacturers can integrate them into newly manufactured chips. Then, car manufactures need to add an authenticator to their bus and configure it for CAIBA-supporting ECUs to talk securely if all receivers interpreting a specific frame implement the AUTOSAR SecOC standard.

5.2. Transmitter Design

Transmitters of CAN frames only require minor changes to support CAIBA. These changes, however, require little additional space on the die of the ECU chip.

The main change between a transmitter supporting the AUTOSAR SecOC standard and a transmitter supporting CAIBA is the computation of two authentication tags instead of one over the concatenated CAN ID and payload. For the source-authenticating tag t^s, we rely on BP-MAC, a novel MAC scheme optimized for short messages of only a few bytes [75]. The integrity-protecting tag t^i can be computed by any suitable MAC scheme, which also allows the deployment of CAIBA without any modifications to the receivers, as they only receive and verify this tag. Once both tags are computed, the transmitter simply aggregates them with XOR for the transmitted tag t and integrates it into the payload of the CAN frame. We use $24\,\text{bits}$ reserved in the payload for the tag and keep track of a counter for replay protection by appending its 4 least-significant bits in each frame. The usage of a $24\,\text{bits}$ MAC with the 4 least-significant counter bits corresponds to `SecOC Profile 3 (JASPAR)` [5].

Furthermore, the transmitter must be adapted to support tag overwriting by the authenticator. On the one hand, the CRC checksum and the placement of stuff bits

---

2. There maybe only exists a single group key for the entire bus.

in the final received frame must be ensured. Therefore, the sender computes the checksum and the stuff bits placement based on the expected final frame, after the authenticator's modification, and not based on the transmitted frame. The authenticator knows when to expect stuff bits and skips over them. For the particular CAN controller used for our implementation, this requirement was naturally fulfilled: The controller listens to the bus while transmitting its message to detect higher-priority transmissions and uses this received data for bit stuffing and CRC computation. To the best of our knowledge, this behavior is however not required by the standard.

On the other hand, the transceiver must ignore overwritten signals during the transmission phase of the tag. Otherwise, the transmitted would quickly switch to the bus-off state and virtually disconnect itself from the bus. Therefore, we change bit monitoring to listen to the expected bits after an authenticator overwrote the bits. This change does not lead to security or reliability problems, since every unauthorized modification would result in an invalid tag on the receivers' side. If errors only increase during the tag transmission than an ECU can conclude that the authenticator is faulty or disabled and react accordingly. This reaction could be the fallback to unsecured CAN after alerting the network or the switch to on a backup authenticator.

While CAIBA requires some modifications to CAN transmitters, it is important to note that the communication on the bus still remains CAN compliant and interoperable. Therefore, CAIBA can coexist with legacy CAN transmitters as long as the authenticator knows which communications are CAIBA-protected and which are not.

5.3. Authenticator Design

The authenticator continuously monitors the bus and notices when the transmitter starts writing to the bus. Then, it has to *(i)* identify the transmitter, *(ii)* compute the source-authenticating tag t^s over the CAN ID and the first part of the payload as well as *(iii)* overwrite the second part of the payload based on that computed tag. All of this processing needs to happen concurrently with the ongoing transmission of the CAN frame. In the following, we discuss how these three steps are realized in CAIBA.

5.3.1. Source Key Identification. The authenticator first identifies the alleged source of a frame and whether it supports CAIBA[3]. While CAN uses message identifiers that should be associated with a unique sender, often employed standards ensure an easy coupling without a large lookup table [46]. Thus, the message ID can be used to identify the unique transmitter of a message. It is crucial to securely configure which ECU supports CAIBA during vehicle assembly to thwart downgrade attacks. After the sender has been decoded by the authenticator, the transmitter and thus the relevant key k_j^{source} is available.

5.3.2. Fast In-Line MAC Computation. When a CAIBA transmitter is identified, the authenticator computes the

source-authenticating tag t^s based on the source authentication key k_j^{source}, the protected payload, and a nonce (a $8\,\text{bytes}$ long counter which is synchronized based on the 4 least-significant bits transmitted with each frame). Considering CAN's maximum supported data rate of $1\,\text{Mbit/s}$, the first bit of the tag follows within $1\,\mu\text{s}$ after the last data bit. Actually, the authenticator needs to compute t^i in a fraction of this time, as it needs time to overwrite this first tag bit before it is read by the receiver.

To achieve these speeds while still relying on sound cryptography, CAIBA uses the BP-MAC [75] scheme with its unique performance through bit-wise precomputing tags and enhances it by a custom online computation algorithm. For now, it is only important that the computation of the source-authenticating tag t^s can be achieved with a single XOR operation per read bit, which is fast enough to be performed even in a fraction of the time between two CAN signals. Meanwhile, BP-MAC relies on a Carter-Wegman construction such that it offers the same guarantees as the underlying MAC scheme, *e.g.*, HMAC_SHA256. The authenticator is thus fast enough to know the source-authenticating tag t^s that it must XOR with the transmitted tag t as soon as its first bit is written to the bus. The details of the BP-MAC tag computation in CAIBA are covered in Section 6.

5.3.3. Overwriting CAN Frames. Once the authenticator has read the source-authenticating tag t^s, it must XOR this value with the transmitted tag, *i.e.*, whenever a bit in t^s is set, the signal on the bus must be inverted. For the signal-flipping procedure, the authenticator first needs to receive the transmitted signal. By sampling early in the nominal bit time, the authenticator can read the original signal while still having time to react. This early sampling is similar to the process how CAN-FD achieves higher bitrates than CAN. It is possible because of relatively large tolerances for rise and propagation times in the CAN standard, which are designed for the worst case where the transmitter and receiver are located as far apart as possible. We suggest that the authenticator is placed centrally for bus length close to the maximum supported for a given bitrate, such that its distance from the transmitter is at most half of what is compensated for by CAN (even less if multiple authenticators are used). If the bit on the bus is recessive, the authenticator can simply overwrite it with a dominant bit. However, the opposite situation is not trivial, as dominant bits always overwrite recessive ones. For this purpose, we take advantage of the limited output current of CAN transceivers by simply connecting an additional *inverted* transceiver in our Reactive Bit Flipping (RBF) approach, which is explained in more detail in Section 7. With RBF, the authenticator can ultimately XOR the transmitted tag t with the computed tag t^s, such that receivers sense the integrity-protecting tag t^i when sampling the bus at standard sampling points.

5.4. No Need to Adapt Receivers

The design of CAIBA requires no software or hardware changes at the receiver if the AUTOSAR SecOC standard is already supported since MACs are already implemented. The modifications of CAIBA only overlay the traditionally transmitted integrity-protecting tag t^i with

---

3. We assume that the bus is configured according to AUTOSAR SecOC, *i.e.*, space for authentication tags is reserved and nodes not capable of performing the integrity verification ignore the tag. Alternatively, all ECUs must track for which CAN IDs integrity protection is enabled.

the source-authenticating tag t^s. However, this change is transparent for an ECU which samples the bus as defined by the CANopen standard [18]. Thus, receivers do not need to be changed or replaced to support CAIBA which enables a smooth and iterative deployment of CAIBA.

Once a car manufacturer integrates an authenticator module, all CAN transmitters can decide to employ CAIBA. Each sending ECUs employing CAIBA needs to share a secret key k_j^{source} with the authenticator, which will, in most cases, be statically configured by the manufacturer. Afterward, the authenticator starts overwriting the authentication tag according to CAIBA's design. A receiver can then process CAN frames exactly as before. If a message is tampered with, either through manipulations during transmission or by being sent from an unauthentic source, the verification by the receiver fails and it processes this anomaly accordingly.

5.5. Error Handling and Recovery

Failed CAN transmissions could result from desynchronized nonce counters of the sender, authenticator, and receiver. If this were the case, no future message would be verifiable. However, if CAN transmissions are not received by all ECUs, *e.g.*, due to an error in the CRC checksum at a single ECU, this is announced with a distinct error frame. In that case, the authenticator resets its counter and the sender retransmits the frame with the original MAC. The receiver would not have processed the MAC as the error frame interrupts the transmission, such that no action is required.

Due to the transmission of the four least-significant bits of the nonce in each frame, it is unlikely that a node gets out of sync. To get to such a state, one ECU must miss an error frame or a frame must be discarded due to multiple failed retransmissions, at least 16 times in a row from the same origin. If such an unlikely scenario were to occur, it most likely stems from a malfunctioning ECU or an active denial of service attack, both of which CAIBA cannot protect against. Still, to ensure the best possible resilience, CAIBA implements a distinct recovery mechanism for such cases. If a receiving node fails to verify the tag of five consecutive frames, it requests a counter reset through a specific CAN ID.

Once this request is received by the sending ECU, it updates the two counters for the integrity-protecting and source-authenticating tags. Therefore, the value in the n most-significant bytes of the counter is incremented, where n represents the number of payload bytes in a CAIBA frame. The less-significant bytes are meanwhile reset to zero. The sender then first sends the most significant bytes of the counter for the source-authenticating tags to the authenticator. This CAN frame is protected by a regular MAC (sender and authenticator share a secret key) and the frame must only be authenticated by this one receiver. Then, the updated counter for the integrity-protecting tag is broadcasted by a CAIBA-authenticated CAN frame to all receivers. This procedure guarantees that no nonce is reused (for authentication by the sender), while even for counters that drifted apart significantly, they are reset to the same value.
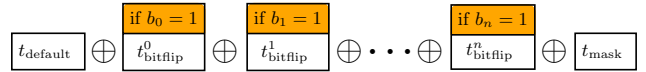

Figure 3: A BP-MAC tag is computed by XORing a default tag $t_{default}$, a masking tag t_{mask}, and a bitflip $t_{bitflip}$ for each bit b_i in a message that is set to 1. All of these individual tags can be computed ahead of time, only the XOR needs to be performed once the message is known.

5.6. Reliability with Multiple Authenticators

A single authenticator is a potential weakness of CAIBA's design. Such a single point of failure can interfere with high-reliability demands of applications relying on CAN. While an inactive or misbehaving authenticator is quickly identified by the sending controllers (*cf.* Section 5.2), downgrading to unprotected communication should always be avoided.

Therefore, we propose to operate CAIBA with multiple authenticators. However, if many authenticators overwrite the same bit simultaneously, the physical signal is soon disturbed and reliable sensing is unlikely, especially due to different propagation delays. Instead, authenticators should be distributed along the bus and only the closest authenticator takes care of overwriting a signal. Which authenticator is responsible for which ECU can be preconfigured, as ECUs are usually stationary. Such a multi-authenticator deployment has the advantage that the sender is always close to its dedicated authenticator, such that propagation delays are reduced. If an authenticator is malfunctioning, it or a noticing ECU, can inform the next authenticator in line to take over. Thus, multiple authenticators increase reliability by (1) reducing the physical distance between senders and authenticators, and (2) offering fallback authenticators in case of malfunctions.

6. Fast MAC Scheme

For CAIBA, we adapt the BP-MAC [75] scheme to allow the online computation of the final tag that can keep up with the speed requirements of the authenticator. We first briefly recapitulate BP-MAC in Section 6.1 (the original paper [75] provides more details), then show how we achieve online tag computability in Section 6.2 before discussing its integration into CAIBA in Section 6.3.

6.1. A Primer on BP-MAC

BP-MAC [75] is based on the Carter-Wegman MAC construction and optimized for short messages that are only a few bytes long. An authentication tag is thus composed of a digest computed by a universal hash function over the message generated with an AES key k_1 and a masking tag in the form of a pseudo-random number generated with a distinct AES key k_2. The masking tag is responsible for hiding the digest, as the MAC is only secure as long as no attacker learns any of these digests. For BP-MAC's universal hash function, each bit is processed individually which allows the preprocessing of these results with linear space overhead. Concretely, the tag computation in BP-MAC works as shown in
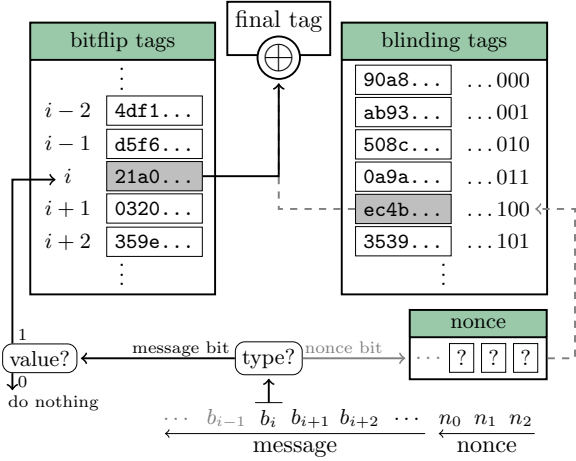
Figure 4: BP-MAC tags can be computed incrementally with each received bit, such that a single XOR operation suffices to compute the final tag once the final bit is read.

Figure 3. In advance, a default tag t_{default} is computed by XORing the bit tags of a message only composed of zeros. Here, bit tags are the AES-encrypted bit index and value, *i.e.*, the bit tag t_i of the i-th bit b_i is computed as $\text{AES}_k(i, b_i)$. Then, depending on where bits in the actual authenticated message are one, the t_{default} is XORed with bitflip tags, *i.e.*, $t_{\text{bitflip}}^i = \text{AES}_{k_1}(i, 0) \oplus \text{AES}_{k_1}(i, 1)$. Finally, the resulting tag is masked with a masking tag that is computed by AES-encrypting a counter that is incremented with each message, *i.e.*, $\text{AES}_{k_2}(counter)$. The counter for the masking tags is known in advance and can thus be precomputed.

6.2. Online BP-MAC Computations

We specifically choose the BP-MAC scheme for the source-authenticating tag in CAIBA since we can modify it for online tag computation. Only with this online tag computation, the authenticator can operate fast enough to be ready to overwrite the next bit. This adapted process to compute BP-MAC tags is illustrated in Figure 4.

The final tag variable is initialized with the default tag for a message composed only of zeros. As we observe in the timeline at the bottom, data is then received bit by bit, and currently the bit b_i of the message is processed. Its value is checked and if the bit is set, the corresponding bitflip tag is XORed with the final tag. These bitflip tags are precomputed and stored alongside the key for the corresponding transmission source. Nonce bits are either explicitly transmitted or implicitly tracked by the sender and receiver. If transmitted, the (least-significant) nonce bits can be received and processed before or after the message bits. In Figure 4, we show an example where the last three bits n_0, n_1, and n_2 are updated with the transmission, while the rest of the expected nonce is tracked implicitly. This update process allows self-synchronization in case of a short burst of failed transmissions. Once the complete nonce is known, the corresponding masking tag is selected and XORed with the final tag. If the message and nonce bits are processed, the order of which actually does not matter, the final tag is known. In either order, the process after the last bit is received is composed

of a simple conditional check and at most one XOR operation. Thus, the authenticator can compute source-authenticating tags over the payload at line rate and be ready to potentially flip the first bit of the transmitted tag.

6.3. Integration into the CAIBA Authenticator

In CAIBA, we use 24 bit tags and 4 bit for nonce synchronization, which corresponds to `SecOC Profile 3` (JASPAR) in the AUTOSAR standard [5]. The blinding tag in BP-MAC is computed by AES encrypting a counter. In CAIBA, we take advantage of the fact that only each fifth blinding tag requires an invocation of the AES algorithm. A 16 byte encrypted AES block is divided into five blinding tags of 3 bytes (1 byte is discarded). Thus, the counter is only encrypted if it is dividable by five, otherwise, the already computed blinding tags are used in order. This encryption of an AES block happens whenever the message using the last blinding tag is transmitted. If the message frequency is too high for the authenticator's processor, the AES algorithms can also be computed iteratively between every single message.

7. CAIBA's Overwriting Mechanism

Altering CAN messages during their transmission is one key function of the authenticator in CAIBA. Technically, it requires a modification of the voltage levels on the bus lines and has to be timed precisely to not disturb ongoing transmissions and to ensure the desired message is received by other participants. While the authenticator itself does not verify the transmitted tag t ($t = t^i \oplus t^s$), it calculates the source-authenticating tag t^s based on the payload, the nonce, and the known key k_j^{source}. By then applying the XOR operation on t and t^s (recomputed based on the received data) again within the transmission, the resulting tag t^i is read by the other devices of the bus. Recently, the feasibility of real-time perfect bit modification attacks, where the original bus state is overwritten with a static value, has been demonstrated [40]. However, neither t nor t^i is known to the authenticator in advance, such that the authenticator has to carry out these bit modification operation *reactively* and in a bitwise manner. Specifically, the authenticator must write the inverse bit signal of what was originally written to the bus whenever a bit in t^i is set. In the following, we will describe the process of overwriting the authentication tag of a message in Section 7.1. With RBF, we then introduce a novel technique to reactively change the physical CAN signals on-the-fly in Section 7.2. Even though we use this technique exclusively for the CAIBA authenticator, it is worth to emphasize that RBF can be used more generally, for both defensive and offensive purposes.

7.1. Physical Signal Modification

Flipping a single bit signal requires modifying voltage levels on the bus lines. Overwriting a recessive bit to a dominant one is a core functionality of CAN to realize arbitration. Hence, the authenticator can use the normal process to write a dominant bit and thus overwrite the recessive bit of another ECU. However, if the transmitter
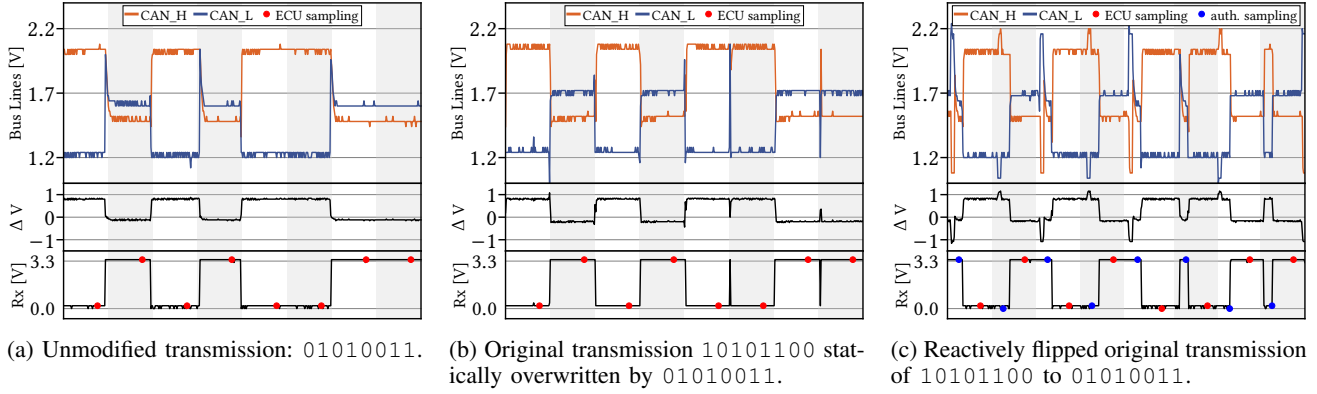
(a) Unmodified transmission: `01010011`.

(b) Original transmission `10101100` statically overwritten by `01010011`.

(c) Reactively flipped original transmission of `10101100` to `01010011`.

Figure 5: Recording of the bus lines and transceiver's RX output during transmissions including sampling points.
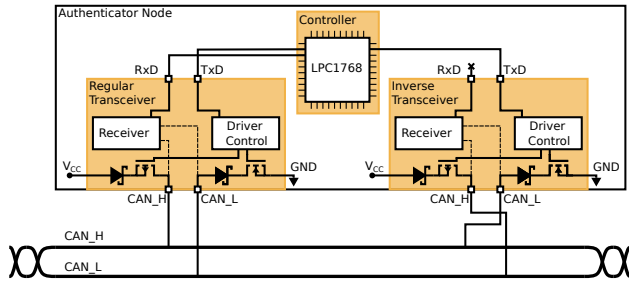

Figure 6: The authenticator node controls an additional CAN transceiver, that is inversely connected to the bus lines to *erase* voltage differences between the bus lines.

sends a dominant bit, the authenticator must actively drive the voltages on the bus lines to the recessive state to overwrite with a recessive bit. To achieve this overwriting, the applied voltages of the transmitter have to be reverted by sourcing CAN_L while sinking CAN_H to ground. This inverse connection, in relation to the transmitting transceiver, enables the current flow from the transceiver through the terminating resistors to be diverted off the bus, which reduces the measurable voltage drop of dominant signals. However, as CAN always operates based on the differential voltage, it is enough to keep the voltage drop below $0.5\,\text{V}$ to make sure it is read as a recessive state [31]. Thus, additional components, which effectively erase the voltage difference would typically need to have an internal impedance that is significantly lower than the effective impedance of both terminating resistors.

We could build such a device from discrete components. However, ordinary CAN transceivers often already offer the inverted functionality. A common internal structure of the transceiver contains two transistors to pull CAN_H to the supply voltage and CAN_L to ground [22]. We can thus connect an additional CAN transceiver with inverted bus lines to the authenticator, as shown in Figure 6. Then, we use the transistors to reduce the voltage difference when transmitting an apparent zero bit at this transceiver. As this would typically result in a dominant transmission, it enables a current flow through the internal transistors, which sinks the current from CAN_H to ground and sources CAN_L from the supply voltage. This changes the bus voltage to the original, recessive state as can be seen in Figure 5a and b.

7.2. Reactive Bit Flipping

The process to flip a bit depends on the current bit of the source-authenticating tag t^s and the transmitted authentication tag t. While t^s is known, the authenticator, since not in possession of k^{group}, cannot compute a single bit of t^i before the according bit of t has been transmitted by the original transmitter. However, calculating this bit after sampling the bus would result in a failing verification, since the receiver would already expect the authenticated corresponding bit of t^i. Hence, signal modifications, as described in the previous Section 7.1, requires a priori knowledge about the current bit signal.

Our proposed RBF technique makes the overwriting mechanism of the authenticator *reactive*, *i.e.*, the authenticator simultaneously reads and reacts to the current state of the bus within the transmission of the same bit. To be able to read the original signal, we modify the transmission only between the third and the last time quantum of each bit signal. The first two time quanta are used to let the bus enter the state of the originally transmitted bit. At the end of the second time quantum, the authenticator samples the bus and reads the latest transmitted bit of t (*cf.* Figure 5c). We can expect the bus to be collision free at this time, because it was reserved to the sender during the arbitration phase of the frame. Furthermore, we can assume that the signal has already propagated via the bus due to an approximately constant propagation delay and shift of the bit timings between ECUs. The sampled signal is then used for calculating the corresponding bit of t^i and to select the suitable CAN transceiver (*i.e.*, regular or inverse (*cf.* Figure 6) for a possible bit alternation.

In the remaining time quanta, the bus is forced into the desired state by the authenticator. After $75\,\%$ of the nominal bit time has passed, ECUs sample the bus [18] and will read the authenticated bit of t^i.

7.2.1. Bit Synchronization Conflict. Changing the voltage levels on the bus within a bit time can conflict with CAN's edge-oriented synchronization. Due to RBF, bits are overwritten in the third time quantum which introduces additional edges. Concretely, a CAN controller synchronizes based on the edge when changing from a recessive state to a dominant state by prolonging the expected duration of the bit by the preconfigured Synchronization Jump Width (SJW) [29]. The SJW is 1 to 4

time quanta long and defines the maximum time by which a controller extends/shortens a bit, with a larger number generally chosen to improve robustness. However, altering the transmitted recessive bit to dominant will cause an edge in the third time quantum. Exactly if the j-th bit is read recessive ($t_j^i = 1$) and is followed by a dominant bit that is flipped ($t_{j+1} = 1$ and $t_{j+1}^i = 0$), the first detected edge occurs when the authenticator changes a signal from recessive to dominant, and all nodes resynchronize their bit time by this delayed edge. We compensate for the time shift by increasing the current bit time at the authenticator by two time quanta. The increased time the authenticator overwrites the transmission ensures that the bit value remains constant until all nodes have sampled the bus. Furthermore, we avoid additional disturbances to the bus, caused by multiple signal changes in a short period. Thus, CAIBA's authenticator compensates for the synchronization procedure embedded into all CAN ECUs.

7.2.2. Bit Stuffing Conflict. Most parts of a CAN frame are affected by bit stuffing, including the authentication tag t within the payload. Changing single bits in t can require additional stuff bits or make existing ones obsolete. Both cases could lead to an incorrect authentication tag received by other nodes or an incorrect length of the data frame and would interrupt the transmission. As stated in Section 5.2, we expect the sender to place stuff bits according to the modified bit stream that is received. To prevent overwriting stuff bits in t, the authenticator pauses the bit modification for one bit time whenever it expects a stuff bit from the sender. The position of a stuff bit is determined based on the last five regular sampling points. Although bit modification is paused, the authenticator samples the bus during the expected stuff bit to detect stuff errors or error frames.

8. Evaluation

We presented CAIBA as an innovative source authentication scheme to protect CAN without requiring significant changes to existing systems. In the following, we present a proof-of-concept implementation and show the general applicability of CAIBA. We start with the introduction of our evaluation setup and limitations in Section 8.1. In Section 8.2, we compare the reliability of our scheme regarding CAIBA-protected traffic. CAIBA's compatibility with legacy CAN devices is demonstrated in Section 8.3, its processing overhead investigated in Section 8.4, and potential bus length restrictions discussed in Section 8.5. Finally, we take a look at potential adverse long-term effects on the employed hardware in Section 8.6.

8.1. Evaluation Setup and Limitations

For our proof-of-concept implementation, we set up a testbed consisting of different ECUs as shown in Figure 7. To rapidly prototype the necessary modifications within the CAN controller, we used the software-defined CAN controller (SDCC) [11] with the recommended NXP LPC1768 microcontroller platform and TI SN65HVD230 CAN transceivers. Additionally, three unmodified off-the-shelf CAN controllers were used to
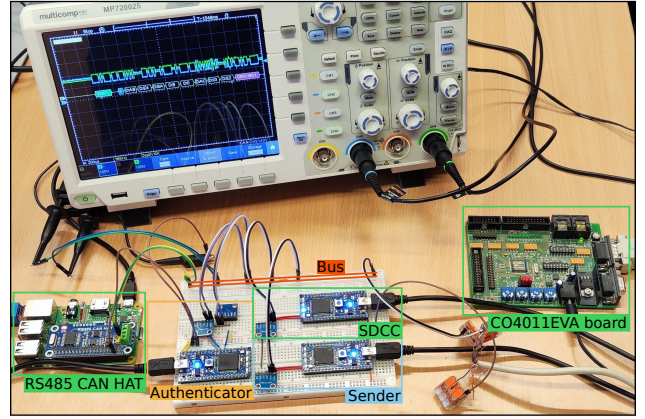

Figure 7: Evaluation setup of CAIBA consisting of a CAIBA-capable transmitter •, three unmodified receivers •, and a CAIBA authenticator •.

evaluate the backward compatibility of our solution. As the SDCC is a pure software implementation on a regular microcontroller without any hardware acceleration, the maximum bitrate is limited to $40\,\mathrm{kbit/s}$ [11]. To show the general feasibility of CAIBA, this is sufficient.

However, to evaluate CAIBA with higher bitrates, a hardware- or FPGA-based implementation would be necessary. This is an effort we consider disproportionate to demonstrate CAIBA's general feasibility. Instead, our evaluation focuses on the practical feasibility at low bitrates, while we theoretically analyze the effects of higher bitrates *e.g.*, w.r.t. to the bus length. However, as commercial adaptions of CAIBA are expected to be implemented in hardware, *e.g.*, as SIP cores (*cf.* Section 5.1), we do not anticipate any processing limitations even at higher speeds. For example, the most time-critical aspect is the final computation of t^s by the authenticator after the last payload bit has been received, which requires a single XOR operation of three bytes and can be computed within a single clock cycle by a hardware-based authenticator.

With a faster hardware prototype, we could validate that CAIBA and legacy CAN ECUs can coexist on the same bus in an actual car. However, even then, we could not easily investigate if receiving ECUs implementing the AUTOSAR SecOC standard could be retrofitted with CAIBA's source authentication because we have no access to the secret group key used by the car's ECUs. Consequently, we have to fall back to a physical testbed for our proof of concept evaluation of CAIBA.

8.2. Reliability

Ideally, CAIBA can enable source authentication to CAN without impacting its reliability. Therefore, we compare the reliability of CAIBA to a regular CAN deployment. For data rates varying between 10 and $40\,\mathrm{kbit/s}$, we sent 100,000 frames with 4 to 8 bytes of payload (incl. authentication tag) and analyze the ratio of correctly received frames at the receiver, including a valid integrity-protecting tag. To have an equal number of ECUs connected to the bus, and thus have comparable desynchronization potential, we connect an additional listening ECU for the CAN measurements that does not need the authen-
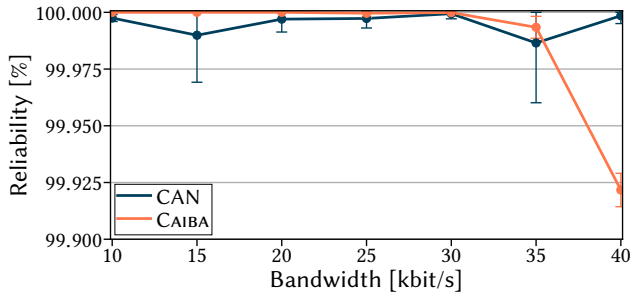
Figure 8: CAIBA achieves similar reliability to CAN for low data rates but it reaches the timing limitations of the SDCC slightly earlier when increasing the rate.

Figure 9: Processing overhead of CAIBA's tag computation on a representative ARM Cortex M3 chip is significantly lower than a single frame transmission ($128\,\mu s$) even if CAN is operated at $1\,\mathrm{Mbit/s}$.

ticator. We repeat each measurement ten times and show our results (incl. $95\,\%$ confidence intervals) in Figure 8.

For low data rates, we surprisingly observe that CAIBA achieves slightly higher reliability than CAN with no frame loss observed for data rates below $35\,\mathrm{kbit/s}$. The most likely explanation for this effect is that the authenticator ECU, which acts as an additional CAN receiver for these measurements, leads to rare misinterpretations of the ACK bit. As we reach the maximum data rates supported reliably by the SDCC (*i.e.*, $40\,\mathrm{kbit/s}$), CAIBA's reliability starts to decrease. Here, we reach the limitations of the timing accuracy achievable in a pure software-defined controller slightly earlier than for CAN due to the delicate bit synchronization necessary during RBF. A similar drop in reliability can also be observed in CAN when increasing the data rates further.

Overall, CAIBA can operate with similar reliability as CAN. This high reliability can however only be achieved with an authenticator providing precise timing relative to the bus speed. For higher data, this is only achievable through hardware-implementations of CAIBA controllers.

8.3. Compatibility with Legacy CAN Devices

We cannot expect that each device on a bus is aware of CAIBA. Otherwise, car manufacturers would have to convince each ECU supplier to adopt CAIBA, before the first car can employ multicast source authentication. Thus, CAIBA is specifically designed to not interfere with legacy communication. Also, only transmitters have to be altered to support CAIBA, while AUTOSAR SecOC-implementing receivers can still benefit from the provided security without any modifications (*cf.* Section 5).

To validate the compatibility with legacy devices, we tested CAIBA in combination with three additional off-the-shelf receivers. In particular, we used CAIBA in combination with an MCP2515 CAN controller connected to a Raspberry Pi 3 trough SPI in the form of the *Waveshare RS485 CAN HAT* [78], the integrated CAN interpreter of a *MULTICOMP PRO MP720025 EU-UK* [51] oscilloscope, and a CANOpen Evaluation Board using the *Frenzel+Berg CO4011A Controller* [21]. All three devices served as regular receivers for authenticated CAN messages that were modified by the CAIBA authenticator. As a result, no anomalies have been observed when reading CAN messages with any device. Thus, we conclude that CAIBA is indeed compatible with unmodified receivers and legacy CAN communication.
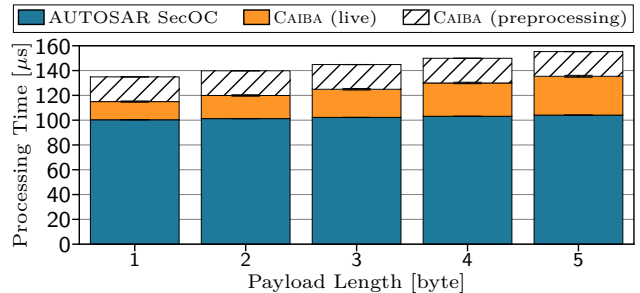
8.4. Upper Bound on Processing Overhead

In the literature, cryptographic approaches to protect the CAN bus are often criticized for excessive processing overhead. Indeed, the computation of a single HMAC-SHA256 tag on an ARM Cortex M3 ($32\,\mathrm{MHz}$) takes $1641.4 \pm 0.1\mu s$. The ARM Cortex M3 is a typical prototyping processor with similar processing power to commercial ECUs, which are however built to operate reliably even in harsh environments. The computation of one tag thus takes over 10 times longer than the transmission of a single CAN frame at $1\,\mathrm{Mbit/s}$ ($128\,\mu s$). In other words, a corresponding ECU could only verify or authenticate one tenth of all CAN frames transmitted on a fully utilized $1\,\mathrm{Mbit/s}$ bus. To assess the overhead of CAIBA on the ECUs, we measured the duration to compute a tag (consisting of t^s and t^i) with a length of $24\,\mathrm{bit}$ for payload lengths varying between 1 and 5 bytes.

For the integrity-protecting tag t^i, we compute a CMAC tag as recommended in AUTOSAR SecOC. On the other hand, the source-authenticating tag t^s is computed based on BP-MAC for fast online computation by the authenticator. As our results in Figure 9 demonstrate, the overhead of the additional tag computation is at most $51\,\%$ in total. Here, a fixed overhead of $19.9\,\mu s$ is caused by the preprocessed computation of blinding tags. The actual additional delay during tag computation only amounts to between $14.6\,\mu s$ and $31.3\,\mu s$, depending on the payload length. Moreover, we observe that if AUTOSAR SecOC were to adopt BP-MAC for its tag computation, it could reduce its processing time by $69.0\,\mu s$, *i.e.*, CAIBA would then be faster than AUTOSAR SecOC with its recommended MAC scheme of today.

As we expect that commercially deployed CAIBA controllers are implemented in hardware, the overhead of BP-MAC is suspected to further reduce significantly [75]. Overall, we found that cryptographic processing is not a significant drawback for ECUs, even for cryptographic processing in software, due to the selection of a fast, yet secure, MAC scheme.

8.5. No Limitations to the Bus Length

As the authenticator's overwritten signal must be reliably sampled by the receiving ECUs, we now look at potential bus length restrictions. In the worst case, the originally transmitted signal and the overwritten signal

are delayed by twice the distance between the sender and the authenticator. This maximal offset occurs if the receiver is placed near the sender on one extremity of the bus with a maximum distance to the centrally placed authenticator. Hence, one might expect a reduction in the maximum CAN cable length by the employment of CAIBA. However, adding up the delays for signal propagation, overwriting, and synchronization even for the maximal bus lengths, CAIBA's delays are still tolerable according to the acceptable intervals for all bit rates of CAN (assuming a single centrally placed authenticator).

Looking at the example for $1 \, \mathrm{Mbit/s}$, CAN supports a maximum cable length of $25 \, \mathrm{m}$ and a receiver sampling point $750 \, \mathrm{ns}$ after the start of a bit. Adding the propagation delay of traveling halfway and back ($125 \, \mathrm{ns}$), three quanta idle time for synchronization and back-propagation ($375 \, \mathrm{ns}$), and the authenticator's transceiver delay ($210 \, \mathrm{ns}$) results in a worst-case delay of $690 \, \mathrm{ns}$[4]. Hence, even with a maximum CAN cable length and worst-case sender and receiver placement, the overwritten signal is still stable at the receiver before it samples the bus $750 \, \mathrm{ns}$ after the start of the signal. Using CAIBA thus does not restrict maximum cable length, as other aspects of CAN are more restrictive, *e.g.*, arbitration and ACK bits that must function over the entire length of the bus. Meanwhile, the centrally placed authenticator is closer to the sender and thus operates with lower propagation delay.

To practically validate that CAIBA can operate on longer buses, we placed a $50 \, \mathrm{m}$ twisted pair cable between the authenticator on the one end of the cable, and the sending as well as the receiving ECU on the other end of the cable. This corresponds to the worst-case scenario for a $100 \, \mathrm{m}$ long bus. We transmit 10,000 CAN frames at $40 \, \mathrm{kbit/s}$ and repeat this measurement ten times. Note that at these speeds CAN could operate on a bus 10 times as long, but only reliably with the use of optocouplers [18]. We achieve a reliability of $99.95 \pm 0.03 \, \%$. While this reliability is even slightly higher than for the short bus ($99.92 \pm 0.01 \, \%$), potentially due to better stabilizing of the signal during propagation, they lie within the margin of error of each other. Overall, we can thus conclude that CAIBA can operate reliably even on longer buses and does, in theory, not restrict the maximum bus length at all.

8.6. Long-Term Impact of Overwriting Bits

Finally, we study potential long-term adverse effects of CAIBA on the hardware. When the authenticator overwrites a dominant bit, the inversely connected transceiver sinks the current from the bus line, resulting in an actively driven recessive state. While this can be seen as a short circuit on the bus (from the view of the original transceiver) with a dampened current, all transceivers must be short-circuit proof for fault tolerance [27].

Additionally, we measure the current that flows through the affected transceivers during the overwriting of dominant bits. We observe a maximum current of $28 \, \mathrm{mA}$, which is within specifications for common CAN transceivers such as the Philips TCA1050 ($100 \, \mathrm{mA}$) [62] or the TI SN65HVD25x ($160 \, \mathrm{mA}$) [71]. Most importantly, these currents only affect CAIBA transceivers. Legacy

4. Baseline numbers stem from the CANopen standard [18].

ECUs connected to the same bus are not affected as they only listen when dominant bits are overwritten. Consequently, there exists no risk for long-term effects of operating CAIBA in CAN.

9. Limitations and Future Challenges

We present CAIBA as a novel solution to the multicast source authentication problem and thus protect CAN. In this context, we provide a proof-of-concept implementation and show its compatibility and reliance in a range of evaluations. In the following, we identify current limitations and potential future improvements for CAIBA.

Broader Applicability of CAIBA. CAIBA promises multicast source authentication without verification delay or excessive bandwidth requirements. Indeed, we demonstrate CAIBA's seamless integration into the CAN bus protocol. However, it remains to be investigated how widely applicable CAIBA is beyond CAN. Other automotive buses, such as LIN [30] and FlexRay [28], show great potential but also pose some unique challenges. The potential of CAIBA in these and other networks, such as star topologies with the central switch taking on the role of the authenticator, is yet unclear.

Hardware-based CAIBA ECUs. We prototypically show the applicability of CAIBA on a software-defined CAN controller and its interoperability with off-the-shelf CAN controllers. However, inherent performance limitations and timing inaccuracies of software-defined controllers limit the bus speeds. A hardware-based CAIBA controller is thus needed for the deployment of CAIBA with typical bus speeds in cars.

CAIBA alongside Intrusion Detection. At the beginning of this paper, we argued that we should not rely solely on IDS because of their imperfect detection, false alarms, and potential evasions. Nonetheless, IDSs do not become superfluous because of CAIBA. In contrast, the operation of CAIBA produces characteristic behavior that can even facilitate monitoring by an adapted intrusion detection mechanism to detect any anomalous behavior early. Such potential symbiosis should be further investigated.

Hardening the Authenticator Module. Throughout this paper, we assume a trusted authenticator that is hard to compromise even with physical access to the CAN bus. This assumption is in line with other proposals, *e.g.*, IDSs for in-vehicular communication. Still, the concrete steps to maximize temper-resilience beyond offering no external connectivity must be worked out.

10. Ethics Considerations

By proposing a preventive measure to secure communication on the CAN bus, our research might not raise obvious major ethical concerns. Still, during the conceptualization and execution of our research we carefully followed established best-practices and guidelines to identify and address any existing ethics-related concerns [37], [72].

Our survey and especially Appendix A summarize the weaknesses of existing CAN security mechanisms proposed in research. The information summarized there could potentially be exploited by malicious actors. However, our analysis is based on information that is mostly

already publicly available and, with the exception of AUTOSAR SecOC [5], we are not aware that any of these proposed schemes are used in commercial products. For the AUTOSAR SecOC specification, the vulnerability to masquerading attacks is a known and accepted risk [5]. Thus, we carefully ensured to not give malicious actors any advantage to compromise any CAN-controlled system through the information we provide in this paper. In contrast, we raise awareness to weaknesses in previously proposed schemes, thus potentially preventing the deployment of vulnerable security schemes which could offer a false sense of security.

Concerning publishing the idea of CAIBA, we may give malicious actors early access to understand the mechanism behind a security scheme that may be later deployed in cars or military vehicles. Thus, malicious actors have more time to identify weaknesses and prepare for practically exploiting them. However, exposing security solutions to public scrutiny is important, such that the research community can jointly identify weaknesses before they can be exploited to derive overall more secure solutions.

11. Conclusion

The increasing connectivity of modern vehicles, coupled with the vulnerability of the CAN bus, has raised significant concerns regarding the safety and security of automotive systems. Compromised ECUs of, *e.g.*, infotainment systems, can easily masquerade as other entities in the network and take over critical control of a vehicle. Current security mechanism mostly rely on intrusion detection or group key-based authentication tags, with both approaches not protecting sufficiently against masquerading attacks. In contrast, we propose a novel multicast source authentication scheme (CAIBA) for buses that, unlike prior approaches, does not require longer tags or delayed verification. CAIBA relies on an authenticator that reactively overwrites bits of the authentication tag included in each message. The keys to generate the original tags are only known by the genuine source, while all receivers can verify the altered tags. We show the applicability of CAIBA in CAN, providing source authentication for ECUs implementing the AUTOSAR SecOC specification. CAIBA is incrementally deployable and interoperable with legacy CAN traffic, while achieving high reliability with minimal processing overhead.

References

[1] "AMD LogiCORE™ CAN IP core ," https://www.xilinx.com/products/intellectual-property/do-di-can.html, last accessed: 26.6.2024.

[2] "CAN Injection: Keyless Car Theft," https://kentindell.github.io/2023/04/03/can-injection/, last accessed: 29.8.2024.

[3] "NXP FlexCAN Controller," https://www.nxp.com/products/nxp-product-information/ip-block-licensing/flexcan-controller:FLEXCAN-CONTROLLER, last accessed: 26.6.2024.

[4] E. Aliwa, O. Rana, C. Perera, and P. Burnap, "Cyberattacks and Countermeasures for In-Vehicle Networks," *ACM Computing Surveys (CSUR)*, vol. 54, no. 1, pp. 1–37, 2021, DOI: 10.1145/3431233.

[5] Autosar, "Specification of Secure Onboard Communication Protocol, AUTOSAR FO R20-11," Standard, 2020, https://www.autosar.org/fileadmin/standards/R20-11/FO/AUTOSAR_PRS_SecOcProtocol.pdf.

[6] G. Bella, P. Biondi, G. Costantino, and I. Matteucci, "TOUCAN: A proTocol tO secUre Controller Area Network," in *Proceedings of the ACM Workshop on Automotive Cybersecurity (AutoSec)*, 2019, DOI: 10.1145/3309171.3309175.

[7] M. Bellare, R. Guérin, and P. Rogaway, "XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions," in *Proceedings of the Annual International Cryptology Conference (CRYPTO)*, 1995, DOI: 10.1007/3-540-44750-4_2.

[8] R. Bhatia, V. Kumar, K. Serag, Z. B. Celik, M. Payer, and D. Xu, "Evading Voltage-Based Intrusion Detection on Automotive CAN," in *Proceedings of the Network and Distributed Systems Security Symposium (NDSS)*, 2021, DOI: 10.14722/ndss.2021.23013.

[9] D. Boneh and V. Shoup, *A Graduate Course in Applied Cryptography*, 2023.

[10] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and some efficient constructions," in *Proceedings of the 18th Annual Joint Conference on Computer Communications (INFOCOM)*, vol. 2, 1999, DOI: 10.1109/INFCOM.1999.751457.

[11] G. Cena, I. C. Bertolotti, T. Hu, and A. Valenzano, "On a software-defined CAN controller for embedded systems," *Computer Standards & Interfaces*, vol. 63, pp. 43–51, 2019, DOI: 10.1016/j.csi.2018.11.007.

[12] Y. Challal, H. Bettahar, and A. Bouabdallah, "A taxonomy of multicast data origin authentication: Issues and solutions," *IEEE Communications Surveys & Tutorials*, vol. 6, no. 3, pp. 34–57, 2004, DOI: 10.1109/COMST.2004.5342292.

[13] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive Experimental Analyses of Automotive Attack Surfaces," in *Proceedings of the 20th USENIX Security Symposium (USENIX Sec'11)*, 2011, DOI: 10.5555/2028067.2028073.

[14] K.-T. Cho and K. G. Shin, "Fingerprinting Electronic Control Units for Vehicle Intrusion Detection," in *Proceedings of the 25th USENIX Security Symposium (USENIX Sec'16')*, 2016, DOI: 10.5555/3241094.3241165.

[15] ——, "Viden: Attacker Identification on In-Vehicle Networks," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017, DOI: 10.1145/3133956.3134001.

[16] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "VoltageIDS: Low-Level Communication Characteristics for Automotive Intrusion Detection System," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, pp. 2114–2129, 2018, DOI: 10.1109/TIFS.2018.2812149.

[17] A. de Faveri Tron, S. Longari, M. Carminati, M. Polino, and S. Zanero, "CANflict: Exploiting Peripheral Conflicts for Data-Link Layer Attacks on Automotive Networks," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2022, DOI: 10.1145/3548606.3560618.

[18] E. C. for Electrotechnical Standardization, "Industrial communications subsystem based on ISO 11898(CAN) for controller-device interfaces – Part 4: CANopen," Cenelec, Standard, 2002.

[19] M. Foruhandeh, Y. Man, R. Gerdes, M. Li, and T. Chantem, "SIMPLE: single-frame based physical layer identification for intrusion detection and prevention on in-vehicle networks," in *Proceedings of the 35th Annual Computer Security Applications Conference (ACSAC)*, 2019, DOI: 10.1145/3359789.3359834.

[20] I. Foster, A. Prudhomme, K. Koscher, and S. Savage, "Fast and Vulnerable: A Story of Telematic Failures," in *Proceedings of the 9th USENIX Workshop on Offensive Technologies (WOOT)*, 2015, DOI: 10.5555/2831211.2831226.

[21] Frenzel+Berg, "CANopen Chip CO4011," https://www.frenzel-berg.de/fileadmin/FrenzelBerg/Datenblaetter/CANopen_Chip/ds_co4011b_en.pdf.

[22] J. Griffith, "Learn the inner workings of a CAN bus driver and how to debug your system," Texas Instruments, Technical Article, 2016, https://www.ti.com/lit/ta/ssztbo8/ssztbo8.pdf.

[23] B. Groza, S. Murvay, A. Van Herrewege, and I. Verbauwhede, "LiBrA-CAN: A Lightweight Broadcast Authentication Protocol for Controller Area Networks," in *Proceedings of the 11th International Conference on Cryptology and Network Security (CANS)*, 2012, DOI: 10.1007/978-3-642-35404-5_15.

[24] B. Groza, L. Popa, and P.-S. Murvay, "Canto-covert authentication with timing channels over optimized traffic flows for can," *IEEE Transactions on Information Forensics and Security*, vol. 16, 2020, DOI: 10.1109/TIFS.2020.3017892.

[25] ——, "Highly Efficient Authentication for CAN by Identifier Reallocation With Ordered CMACs," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, 2020, DOI: 10.1109/TVT.2020.2990954.

[26] A. Hazem and H. Fahmy, "LCAP-A Lightweight CAN Authentication Protocol for Securing In-Vehicle Networks," in *10th Embedded Security in Cars Conference (ESCAR '12)*, vol. 6, 2012.

[27] International Organization for Standardization, "Road vehicles - Controller area network (CAN) – Part 3: Low-speed, fault-tolerant, medium-dependent interface," Iso 11898-3:2006, 2006.

[28] ——, "Road vehicles - FlexRay communications system – Part 1: General information and use case definition)," Iso 17458-{1}:2013, 2013.

[29] ——, "Road vehicles - Controller area network (CAN) – Part 1: Data link layer and physical signalling," Iso 11898-1:2015, 2015.

[30] ——, "Road vehicles - Local Internconnect Network (LIN) – Part 1: General information and use case definition)," Iso 17987-1:2016, 2016.

[31] ——, "Road vehicles - Controller area network (CAN) – Part 2: High-speed medium access unit," Iso 11898-2:2022, 2022.

[32] H. J. Jo, J. H. Kim, H.-Y. Choi, W. Choi, D. H. Lee, and I. Lee, "MAuth-CAN: Masquerade-Attack-Proof Authentication for In-Vehicle Networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, 2020, DOI: 10.1109/TVT.2019.2961765.

[33] J. Katz and A. Y. Lindell, "Aggregate Message Authentication Codes," in *Cryptographers' Track at the RSA Conference (CT-RSA)*, 2008, DOI: 10.1007/978-3-540-79263-5_10.

[34] S. Kim and R. Shrestha, "AUTOSAR Embedded Security in Vehicles," in *Automotive Cyber Security: Introduction, Challenges, and Standardization*. Springer, 2020, DOI: 10.1007/978-981-15-8053-6_5.

[35] M. Kneib and C. Huth, "Scission: Signal Characteristic-Based Sender Identification and Intrusion Detection in Automotive Networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2018, DOI: 10.1145/3243734.3243751.

[36] M. Kneib, O. Schell, and C. Huth, "EASI: Edge-Based Sender Identification on Resource-Constrained Platforms for Automotive Networks," in *Proceedings of the Network and Distributed Systems Security Symposium (NDSS)*, 2020, DOI: 10.14722/ndss.2020.24025.

[37] T. Kohno, Y. Acar, and W. Loh, "Ethical Frameworks and Computer Security Trolley Problems: Foundations for Conversations," in *32nd USENIX Security Symposium (USENIX Sec'23)*, 2023.

[38] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental Security Analysis of a Modern Automobile," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2010, DOI: 10.1109/SP.2010.34.

[39] R. Kurachi, Y. Matsubara, H. Takada, N. Adachi, Y. Miyashita, and S. Horihata, "CaCAN-Centralized Authentication System in CAN (Controller Area Network)," in *14th Embedded Security in Cars (ESCAR '14)*, 2014.

[40] Y. Lee, Y.-E. Kim, J.-G. Chung, and S. Woo, "Real Time Perfect Bit Modification Attack on In-Vehicle CAN," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 12, pp. 15154–15171, 2023, DOI: 10.1109/TVT.2023.3295695.

[41] J.-Q. Li, F. R. Yu, G. Deng, C. Luo, Z. Ming, and Q. Yan, "Industrial internet: A survey on the enabling technologies, applications, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, 2017, DOI: 10.1109/COMST.2017.2691349.

[42] T. Limbasiya, K. Z. Teng, S. Chattopadhyay, and J. Zhou, "A systematic survey of attack detection and prevention in Connected and Autonomous Vehicles," *Vehicular Communications*, vol. 37, 2022, DOI: 10.1016/j.vehcom.2022.100515.

[43] C.-W. Lin and A. Sangiovanni-Vincentelli, "Cyber-Security for the Controller Area Network (CAN) Communication Protocol," in *Proceedings of the International Conference on Cyber Security (CyberSecurity)*, 2012, DOI: 10.1109/CyberSecurity.2012.7.

[44] S. Longari, M. Penco, M. Carminati, and S. Zanero, "CopyCAN: An Error-Handling Protocol based Intrusion Detection System for Controller Area Network," in *Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy*, 2019, DOI: 10.1145/3338499.3357362.

[45] S. Longari, C. A. Pozzoli, A. Nichelini, M. Carminati, and S. Zanero, "CANdito: Improving Payload-Based Detection of Attacks on Controller Area Networks," in *International Symposium on Cyber Security, Cryptology, and Machine Learning*, 2023, DOI: 10.1007/978-3-031-34671-2_10.

[46] A. Lotto, F. Marchiori, A. Brighente, and M. Conti, "A Survey and Comparative Analysis of Security Properties of CAN Authentication Protocols," 2024, arXiv:2401.10736, DOI: 10.48550/arXiv.2401.10736.

[47] Z. Lu, Q. Wang, X. Chen, G. Qu, Y. Lyu, and Z. Liu, "LEAP: A Lightweight Encryption and Authentication Protocol for In-Vehicle Communications," in *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, DOI: 10.1109/ITSC.2019.8917500.

[48] E. O. Marasco and F. Quaglia, "AuthentiCAN: a Protocol for Improved Security over CAN," in *Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, 2020, DOI: 10.1109/WorldS450073.2020.9210290.

[49] A. J. Michaels, V. S. S. Palukuru, M. J. Fletcher, C. Henshaw, S. Williams, T. Krauss, J. Lawlis, and J. J. Moore, "CAN Bus Message Authentication via Co-Channel RF Watermark," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 3670–3686, 2022, DOI: 10.1109/TVT.2022.3143708.

[50] C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," *Black Hat USA*, 2015, https://illmatics.com/Remote%20Car%20Hacking.pdf.

[51] Multicomp Pro, "4 Channel Digital Storage Oscilloscope," https://www.farnell.com/datasheets/3155232.pdf.

[52] M. Müter and N. Asaj, "Entropy-Based Anomaly Detection for In-Vehicle Networks," in *IEEE Intelligent Vehicles Symposium (IV'11)*, 2011, DOI: 10.1109/IVS.2011.5940552.

[53] A. Nichelini, C. A. Pozzoli, S. Longari, M. Carminati, and S. Zanero, "CANova: A hybrid intrusion detection framework based on automatic signal classification for CAN," *Computers & Security*, vol. 128, 2023, DOI: 10.1016/j.cose.2023.103166.

[54] S. Nie, L. Liu, and Y. Du, "Free-Fall: Hacking Tesla from Wireless to CAN Bus," *Black Hat USA*, vol. 25, no. 1, p. 16, 2017, https://www.blackhat.com/docs/us-17/thursday/us-17-Nie-Free-Fall-Hacking-Tesla-From-Wireless-To-CAN-Bus-wp.pdf.

[55] S. Nie, L. Liu, Y. Du, and W. Zhang, "Over-the-Air: How we Remotely Compromised the Gateway, BCM, and Autopilot ECUs of Tesla Cars," *Black Hat USA*, vol. 91, 2018, https://i.blackhat.com/us-18/Thu-August-9/us-18-Liu-Over-The-Air-How-We-Remotely-Compromised-The-Gateway-Bcm-And-Autopilot-Ecus-Of-Tesla-Cars-wp.pdf.

[56] S. Nürnberger and C. Rossow, "–vatiCAN–Vetted, Authenticated CAN Bus," in *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, 2016, DOI: 10.1007/978-3-662-53140-2_6.

[57] F. Oberti, A. Savino, E. Sanchez, P. Casasso, F. Parisi, and S. D. Carlo, "CAN-MM: Multiplexed Message Authentication Code for Controller Area Network Message Authentication in Road Vehicles," *IEEE Transactions on Vehicular Technology*, 2024, DOI: 10.1109/TVT.2024.3402986.

[58] A. Perrig, "The BiBa one-time signature and broadcast authentication protocol," in *Proceedings of the 8th ACM Conference on Computer and Communications Security (CSS)*, 2001, DOI: 10.1145/501983.501988.

[59] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient and Secure Source Authentication for Multicast," in *Proceedings of the Network and Distributed Systems Security Symposium (NDSS)*, 2001, https://www.ndss-symposium.org/wp-content/uploads/2017/09/Efficient-and-Secure-Source-Authentication-for-Multicast-Adrian-Perrig.pdf.

[60] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "Efficient authentication and signing of multicast streams over lossy channels," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2000, DOI: 10.1109/SECPRI.2000.848446.

[61] M. D. Pesé, J. W. Schauer, J. Li, and K. G. Shin, "S2-CAN: Sufficiently Secure Controller Area Network," in *Proceedings of the 37th Annual Computer Security Applications Conference (ACSAC)*, 2021, DOI: 10.1145/3485832.3485883.

[62] Philips Semiconductors, "TJA1050 - High speed CAN transceiver," Tech. Rep., 2002.

[63] A.-I. Radu and F. D. Garcia, "LeiA: A Lightweight Authentication Protocol for CAN," in *Proceedings of the 21st European Symposium on Research in Computer Security (ESORICS)*, 2016, DOI: 10.1007/978-3-319-45741-3_15.

[64] M. Rogers, P. Weigand, J. Happa, and K. Rasmussen, "Detecting CAN Attacks on J1939 and NMEA 2000 Networks," *IEEE Transactions on Dependable and Secure Computing*, 2022, DOI: 10.1109/TDSC.2022.3182481.

[65] S. U. Sagong, X. Ying, A. Clark, L. Bushnell, and R. Poovendran, "Cloaking the Clock: Emulating Clock Skew in Controller Area Networks," in *Proceedings of the 9th International Conference on Cyber-Physical Systems (ICCPS)*, 2018, DOI: 10.1109/IC-CPS.2018.00012.

[66] O. Schell and M. Kneib, "SPARTA: Signal Propagation-based Attack Recognition and Threat Avoidance for Automotive Networks," in *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIA CCS)*, 2023, DOI: 10.1145/3579856.3595788.

[67] H. Schweppe, Y. Roudier, B. Weyl, L. Apvrille, and D. Scheuermann, "Car2X Communication: Securing the Last Meter - A Cost-Effective Approach for Ensuring Trust in Car2X Applications Using In-Vehicle Symmetric Cryptography," in *IEEE Vehicular Technology Conference (VTC Fall'11)*, 2011, DOI: 10.1109/VETECF.2011.6093081.

[68] K. Serag, R. Bhatia, A. Faqih, M. O. Ozmen, V. Kumar, Z. B. Celik, and D. Xu, "ZBCAN: A Zero-Byte CAN Defense System," in *Proceedings of the 32nd USENIX Security Symposium (USENIX Sec'23)*, 2023, DOI: 10.5555/3620237.3620623.

[69] J. Shin, H. Kim, S. Lee, W. Choi, D. H. Lee, and H. J. Jo, "RIDAS: Real-time identification of attack sources on controller area networks," in *Proceedings of the 32nd USENIX Security Symposium (USENIX Sec'23)*, 2023, DOI: 10.5555/3620237.3620624.

[70] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion Detection System Based on the Analysis of Time Intervals of CAN Messages for In-Vehicle Network," in *Proceedings of the International Conference on Information Networking (ICOIN'16)*, 2016, DOI: 10.1109/ICOIN.2016.7427089.

[71] Texas Instruments, "SN65HVD25x Turbo CAN Transceivers for Higher Data Rates and Large Networks Including Features for Functional Safety," Tech. Rep., 2015.

[72] U.S. Department of Homeland Security, "The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research," 2012, https://www.dhs.gov/sites/default/files/publications/CSD-MenloPrinciplesCORE-20120803_1.pdf.

[73] J. Van Bulck, J. T. Mühlberg, and F. Piessens, "VulCAN: Efficient Component Authentication and Software Isolation for Automotive Control Networks," in *Proceedings of the 33rd Annual Computer Security Applications Conference (ACSAC)*, 2017, DOI: 10.1145/3134600.3134623.

[74] 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.

[75] E. Wagner, M. Serror, K. Wehrle, and M. Henze, "BP-MAC: Fast Authentication for Short Messages," in *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2022, DOI: 10.1145/3507657.3528554.

[76] Q. Wang, Y. Qian, Z. Lu, Y. Shoukry, and G. Qu, "A Delay based Plug-in-Monitor for Intrusion Detection in Controller Area Network," in *Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, 2018, DOI: 10.1109/AsianHOST.2018.8607178.

[77] Q. Wang and S. Sawhney, "VeCure: A Practical Security Framework to Protect the CAN bus of Vehicles," in *International Conference on the Internet of Things (IOT)*, 2014, DOI: 10.1109/IOT.2014.7030108.

[78] Waveshare, "Rs485 Can Hat," https://www.waveshare.com/wiki/RS485_CAN_HAT.

[79] H. Wen, Q. A. Chen, and Z. Lin, "Plug-N-Pwned: Comprehensive Vulnerability Analysis of OBD-II Dongles as A New Over-the-Air Attack Surface in Automotive IoT," in *Proceedings of the 29th USENIX Security Symposium (USENIX Sec'20)*, 2020, DOI: 10.5555/3489212.3489266.

[80] 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*, vol. 16, no. 2, pp. 993–1006, 2015, DOI: 10.1109/TITS.2014.2351612.

[81] C. Young, H. Olufowobi, G. Bloom, and J. Zambreno, "Automotive Intrusion Detection Based on Constant CAN Message Frequencies Across Vehicle Driving Modes," in *Proceedings of the ACM Workshop on Automotive Cybersecurity (AutoSec)*, 2019, DOI: 10.1145/3309171.3309179.

Data Availability

Our modifications to the SDCC [11] to support CAIBA are available at https://github.com/fkie-cad/caiba.

Appendix

A. Detailed Analysis of Selected Schemes

A recent survey on CAN authentication schemes [46] revealed weaknesses with many of the proposed CAN authentication schemes but classified others as *secure protocols* (LiBrA-CAN [23], LinAuth [43], LCAP [26], CaCAN [39], and AuthentiCAN [48]). With regard to the eleven drawbacks defined in Section 3.2, we took a closer look at the latter schemes (*cf.* Section A.1 – A.6), as well as some additional authentication schemes not considered by the authors (*i.e.*, Watermarking [49], CANTO [24], ZBCAN [68], CAN-MM [57], LEAP [47], and CAN-TORO [25] in Section A.8 – A.12), and re-assess their security in this section. Overall, we find that none of the current schemes offer full protection (as shown in Table 1), hence justifying the existence of CAIBA.

A.1. LiBrA-CAN.

Description. LiBrA-CAN [23] uses a Mixed MAC approach for integrity protection, *i.e.*, keys are shared among a subset of nodes and each node can only verify a fraction of each authentication tag. These keys are initially distributed by computationally superior master node. To authenticate a frame sent to the bus, the sending ECU first authenticates it to the master node or an helper node. This additional node knows additional keys to compute additional authentication data such that the remaining ECUs can eventually (more than one helper node may be required) verify the integrity of the original message.

Analysis. LiBrA-CAN introduces verification delays and communication overhead, similarly to TESLA-based protocols [59], [60] (*cf.* Section 4.1). This overhead grows linearly with the number of nodes and, for a certain number of nodes, the use of digital signatures may even become beneficial over using LiBrA-CAN [23]. Additionally, a master node with knowledge of all secret keys exposes a single point of failure, as an attacker that compromises this node gains full control over the network.

Takeaway. LiBrA-CAN's scalability is limited ⊷, the authentication requires the sending of additional frames 🖧 that lead to a verification delay ⋰, and the master node offers an attractive target to attackers as it is in possession of all keys ∞.

A.2. LinAuth.

Description. In LinAuth [43], each ECU pair shares a secret key. For each CAN ID an ECU transmits or receives, the ECU additionally needs to know the group of other ECUs that are also interested messages with this ID. A sender of a message computes an authentication tag for each interested receiving ECU. The available space is then evenly split among each receiver and fills with the truncated authentication tags as well as the least-significant bits of each ECU-specific counter. Each receiving ECU knows where its authentication data starts and ends within a frame and can thus verify the truncated authentication tag.

Analysis. LinAuth requires significant configuration and storage to track which node is interest in which CAN IDs. Moreover, LinAuth does not scale to realistic network sizes. Reserving $24\,\text{bit}$ for authentication data and transmitted $4\,\text{bit}$ for each counter, as proposed by the `SecOC Profile 3 (JASPAR)` of AUTOSAR [5], already leaves no space for authentication data with 6 receiving ECU.

Takeaway. LinAuth requires significant upfront configuration and additional storage 🗄 and does not scale to realistic network sizes regarding the number of ECUs that are interested in a given CAN ID ⊷ as space in each frame must be reserved for each receiver 🖧.

A.3. LCAP.

Description. In LCAP [26], CAN frames are authenticated by including a 2-byte "magic number" in the payload. This magic number is an element of a hash chain only known to the sending ECUs of a given communication group, while the receiving ECUs know the previous element of this chain. Thus, upon reception of a frame, receivers can verify that the magic number indeed stems from the alleged sender.

Analysis. In LCAP, all communication groups must be known in advance and the sender in each such group must precompute and store a hash chain of a significant length, amounting to several kB of data per group. If all elements of a chain are consumed, a new chain must be computed and handshake messages must be exchanged on the bus, leading to protocol and communication overhead. Most importantly, LCAP is vulnerable to the hijacking of magic numbers. By decoding the magic number of a payload while immediately jamming the bus afterwards, an attacker could ensuring the frame is rejected by other ECUs. Before the original frame is retransmitted by the sender, the attacker can now exploit this (still valid) magic number to inject allegedly legitimate messages.

Takeaway. LCAP produces storage 🗄 and comunication 🖧 overhead and is vulnerable to frame interceptions ↻.

A.4. CaCAN.

Description. Similarly to CAIBA, CaCAN [39] relies on an authenticator node to assist in authenticating CAN frames. This authenticator shares a secret key with each ECUs and has the capability of overwrite in-transmission frames with an error frame, ensuring that the frame is not received by other ECUs. A sending ECU computes a MAC for the message including a counter and transmits the message, the least-significant bits of the counter, and the first byte of the MAC in a payload of a CAN frame. The authenticator verifies this MAC and disrupts the message if the verification fails.

Analysis. In CaCAN no mechanism for resynchronization for the counter after lost frames is devised. Concerning vulnerabilities, CaCAN's authenticator has access to all keys, an attack does not need to compromise it to circumvent CaCAN's protection. It is sufficient to disconnect the authenticator from bus, to disable all authenticity verification without anyone noticing. Afterwards, the attacker can masquerade as any ECU through an attached or compromised ECU as with legacy CAN networks.

Takeaway. Disconnecting CaCAN's authenticator disable all security measures ⏻. Additionally, the authenticator may be compromised to gain access to all key and masquerade as other devices ∞. Finally, frame loss leads to desynchronizations of counters that cannot be recovered from ↺.

A.5. MAuth-CAN.

Description. MAuth-CAN [32] also relies on an authenticator node to authenticate CAN frames. For each CAN ID, the sending ECU and authenticator establish a session key. With this key, a 32-bit authentication tag is generated and included in each CAN frame, split into the extended identifier and payload fields. The authenticator verifies this tag and broadcasts a report only if the verification fails. Receiving ECUs thus wait for a predefined time after frame detection for this report and only process the message when no report got received. If a report is received, its authenticity if verified by the ECU and the frame it reports on is discarded.

Analysis. The MAuth-CAN authenticator knows all keys and if it is disconnected, no frame verification takes place and no reports are sent. The receiving ECUs can, however, not differentiate if reports are not sent because the frame is authentic or because the authenticator is not operational. Thus, after disconnecting the authenticator, the attack can masquerade as any ECU through an attached or compromised ECU. Even without attacks, MAuth-CAN introduces a delay for all frames as ECUs wait for a potential report, and resynchronization requires to perform new session key exchange.

Takeaway. Disconnecting the authenticator disable all security measures ⏻. Then, or if the authenticator is

compromised to gain access to all key and masquerade as other devices ∞. Moreover, all frames are buffered by the receiver ∴ and resynchronizations is expensive ↺.

A.6. AuthentiCAN.

Description. In AuthentiCAN [48], a nonce list is exchanged between each ECU pair, secured by asymmetric encryption based on a broadcasted public key. The payload of each frame then consists of the encrypted concatenation of a message and the first yet unused nonce from the list. The receiver of a message can decrypt a message and verify that the transmitted nonce matches the expected nonce, which allegedly verifies the authenticity of a message.

Analysis. AuthentiCAN introduces significant overhead w.r.t. to memory and bandwidth consumption to exchange and keep track of the nonce lists and consequently primarily addresses CAN-FD with its higher data rate. Secondly, AuthentiCAN only protects the communication between two ECUs and does not support broadcast communication, *i.e.*, frames intended for multiple receivers.

Takeaway. AuthentiCAN does not support broadcast communication ➤ and causes significant memory ☰ and communication ⇄ overhead.

A.7. Watermarking.

Description. The idea of watermarking [49] is to overlay a high-frequency signal over ordinary CAN transmissions. These overlayed signals transmit a time-varying watermark, generated by a random number generator that is seeded with a key known to all legitimate devices connected to the bus.

Takeaway. A compromised ECU can transmit messages with legitimate watermarks, enabling the attack to masquerade as any other device ∞.

A.8. CANTO.

Description. CANTO [24] provides frame authentication through covert timing channels. It assumes that all CAN IDs are transmitted in a regular pattern that is known in advance by all receivers, which is assisted by an optimal a priori frame scheduling to maximize inter-frame spacing and avoid collisions. At the scheduled time, a sending ECU computes a MAC of the frame with a shared group key and uses it to derives an additional delay for the frame. The receiving ECUs then verify that this delay from the expected arrival time matches the expectation for the transmitted frame.

Analysis. Because of its assumption that traffic is sent in repeating patterns, the real-world applicability of CANTO to protect in-vehicle communication may be limited due to spontaneous actions, *e.g.*, by humans in the loop. Moreover, even if CAN traffic exhibits the required regularity, the deployment of CANTO requires the modification of each ECU to support the modified scheduling. Finally, CANTO relies on a shared group key, thus an attack that compromises one ECU has access to this key and can masquerade as any other ECU.

Takeaway. CANTO does not protect against a single compromised ECU ∞ and reschedules the transmission of frames ⧗, ultimately limiting deployability in vehicles as all ECUs must support CANTO.

A.9. ZBCAN.

Description. In ZBCAN [68], each ECU shares a pairwise secret with an authenticator node. Transmissions do not occur instantaneously, but instead are executed at specific time slots. Therefore, the time after the last transmission is split into discrete time spans, each consisting of a predefined number of time slots. The sending ECU then computes its time slot based on the CAN ID, the secret shared with the authenticator, and an implicit message counter. For a given CAN ID, only a subset of all time slots are available to divide them into priority classes. The authenticator monitors the network and verifies the time slots of each message based on its ID, and overwrite mis-timed messages with an error frame, such that only verified messages are received by the other ECUs.

Analysis. While ZBCAN does not consume any additional bandwidth, it is also susceptible to several weaknesses. First, the priority of IDs can be inverted: Consider a high priority message that is generated just after its time slot has passed. Then, a lower priority message would take precedence within this time span before the high priority message has another chance to be transmitted, delaying potentially critical messages. Secondly, with the proposed time span length of 64 nominal bit times for each priority class ZBCAN only achieves the equivalent security of a 6-bit authentication tag. Thirdly, the authenticator can be simply disconnected from the bus as all messages (spoofed or not) are accepted by the network without an authenticator. Finally, ZBCAN only authenticates the intention of the sender to transmit, but not the content of the frame. Hence, an attacker may wait for a transmission to overwrite its content (as shown in Section 7) and the resulting error frame to compromise the channel.

Takeaway. ZBCAN causes some transmission delays ⧗, only protects the sender intention to send but not the content of this transmission with relatively low security levels ↻, and does not protect against disconnecting the authenticator from the bus ⏻.

A.10. CAN-MM.

Description. With CAN-MM [57], ECUs can be incrementally retrofitted with transmitter and receiver modules. These modules compute a MAC based on a frame's content and a group key and then multiplex the transmission of the CAN frame and the MAC with On-Off keying. A receiving ECU equipped with the receiver module can demulitplex the transmission and verify the MAC, while all other ECUs decode ordinary CAN frames.

Takeaway. CAN-MM does not protect against masquerading attacks by compromised ECU because of its reliance on group keys ∞.

A.11. LEAP.

Description. In LEAP [47], each pair of ECUs share a secret key which is used to compute a keystream by encrypting the CAN ID of a message. The first eleven bits of this keystream are embedded into the payload at a location determined by next bits of the keystream such that an eavesdropper does not know which payload bits consist of the authentication tag. Finally, the remaining

bits of the keystream are used to encrypt the payload. The intended receiver of a message can perform these steps in reverse order to decrypt the payload and verify the correct embedding.

Takeaway. LEAP does not support broadcast communication ✈, and requires relatively high amounts of storage for key material 🖥 to compute authentication tags embedded into the payload 🖧.

A.12. CAN-TORO.

Description. CAN-TORO [25] proposes to encrypt CAN IDs with order-preserving encryption to hide the sender from eavesdroppers and to authenticate them without interfering with message prioritization. Each legitimate ECU keeps track of a mapping of IDs to encrypted IDs which is derived from a group key and updated regularly, *e.g.*, once a second. Received CAN frames are discarded if they contain an invalid ID, where the valid IDs change constantly.

Analysis. For CAN-TORO, all ECUs software needs to be modified, as individual ECUs not supporting the protocol interferes with prioritization and may lead to the discarding of valid CAN frames. Moreover, an attack has a short period of time to replay a valid ID before the mapping changes or to outright overwrite the payload of a frame. Finally, by compromising a single ECU, an attacker gains access to the group key and can then masquerade as any other ECU.

Takeaway. CAN-TORO is susceptible to ID reuse by frame interception ↻ and does not protect against masquerading attacks by compromised ECUs ∞. Moreover, tracking the mapping of encrypted IDs cost valuable storage 🖥.

A.13. Other Approaches.

All approaches not further analyzed here (CANAuth [74], Car2X [67], Woo-Auth [80], VeCure [77], LeiA [63], vatiCAN [56], VulCAN [73], TOUCAN [6], and S2-CAN [61]) have their weaknesses discussed in detail by Lotto *et al.* [46].

B. Security Proof

In this section, we want to formalize the security of CAIBA. CAIBA's security relies on two key assumptions:

- An *adversary* $\mathcal{A}$ cannot simultaneously compromise an ECU and the authenticator.

- $\mathcal{A}$ cannot undetectably overwrite bits, unless they compromise the authenticator.

From these two assumptions, we can proof that CAIBA achieves the same security levels as an AUTOSAR SecOC instance with the same tag length. Here, we assume that the underlying MAC schemes are deterministic (m has exactly one valid t) and ideal (an adversary best strategy is to guess a valid tag with a success rate of $1/2^{|t|}$, where $|t|$ is the bit-length of t). Thus, *e.g.*, 3-byte long tags lead to a 1 in 2^{24} chance of a tag being misclassified as valid, *i.e.*, a security level of 24 bit, for CAIBA and AUTOSAR SecOC alike.

We proof the security of CAIBA with a game as typically done for MAC schemes [9]: $\mathcal{A}$ may query an oracle with messages $m_i \in \mathcal{M}$ for t_i and eventually outputs a candidate forgery $(m', t'), m' \notin \mathcal{M}$, where $\mathcal{M}$ is the message space. $\mathcal{A}$ wins this game if the tag t' is valid for the message m'. The security of the scheme is then expressed as $P[\mathcal{A}$ wins$]$, i.e., the probability that $\mathcal{A}$ wins this game.

We have to consider two cases. First, an adversary may have compromised an ECU. In this case, $\mathcal{A}$ can generate t^i but not t^s (unless the compromised ECU is authorized to send CAN ID). $\mathcal{A}$ may query the oracle for t_i^s or t_i for $m_i \in \mathcal{M}$. However, to interfere $t^{s\prime}$, $\mathcal{A}$ has no better strategy than guessing as the underlying MAC scheme is considered secure. Meanwhile, there exists no better strategy than guessing t' either, as otherwise the MAC scheme to compute t^s were not ideal. As the receiver expects to receive $t^{i\prime}$, *i.e.*, t must be $t^i \oplus t^s$ before modification, $\mathcal{A}$'s best strategy is to randomly guess a tag, i.e., $P[\mathcal{A}$ wins$] = 1/2^{|t|}$.

Secondly, an adversary may have compromised the authenticator. In this case, $\mathcal{A}$ knows the keys to compute all source-authenticating tags t^s, but cannot compute integrate-protecting tags t^i. $\mathcal{A}$ could inject a frame without overwriting it. However, therefore they would need to guess a valid t^i, which only succeeds with a probability of $1/2^{|t^i|}$. Alternatively, $\mathcal{A}$ could modify a transmitted message to modify its origin (CAN ID) or content. However, if any of these two fields are modified, t^i is no longer valid and $\mathcal{A}$ would need to guess a new valid tag.

In both cases, $P[\mathcal{A}$ wins$] = 1/2^{|t|}$. Thus, the security of CAIBA depends on $|t|$. In practice, MAC schemes are not ideal, so the MAC scheme chosen in a concrete deployment will cause marginally lower security.