Beyond Ethernet: Reusing MACsec for CANsec

Friedrich Wiemer *, Arthur Mutter *, Jonathan Ndop *†,
Julian Göppert †, Axel Sikora † and Thierry Walrant ‡

* *Robert Bosch GmbH, Stuttgart, Germany, firstname.lastname@de.bosch.com*

† *Institute of Reliable Embedded Systems and Communication Electronics (ivESK), Offenburg University of Applied Sciences, Offenburg, Germany, firstname.lastname@hs-offenburg.de*

‡ *NXP Semiconductors NV, Leuven, Belgium, firstname.lastname@nxp.com*

***Abstract*—In the past, Secure OnBoard Communication (SecOC) has been defined to serve as the foundational mechanism for securing in-vehicle networks. For over a decade, it has been used in hundreds of millions of automotive systems. Its application-layer design and AUTOSAR-based specification have enabled broad adoption across diverse platforms. However, this design also introduces challenges: software-centric dependencies complicate full hardware integration and can limit scalability in resource-constrained Electronic Control Units, particularly as bandwidth demands continue to grow.**

To address these constraints, CANsec has been proposed to address the security objectives of CAN XL. As a Layer 2 security protocol, CANsec aims to overcome SecOC's shortcomings and offer modern guarantees comparable to MACsec. However, the CANsec specification remains under active development even after several years of work, leaving a gap between conceptual goals and practical deployment.

This paper proposes a pragmatic and standards-aligned solution: it re-uses existing MACsec specifications and implementations as the security engine for CAN XL. MACsec, standardized for over two decades and widely scrutinized in both academic and industrial contexts, offers robust and well-understood security functions. Our approach introduces a lightweight wrapper that maps CAN XL frames to virtual Ethernet frames, enabling MACsec to provide confidentiality, integrity, authenticity, and freshness. We formally define the wrapping process, frame formats, and protocol data units, while preserving MACsec's security properties, adapting them to the constraints and requirements of automotive networks. This enables a practical and secure path forward for CAN XL deployment, leveraging mature cryptographic algorithms and protocols without compromising performance or assurance.

To support standardization and practical adoption, we have submitted this approach to the CAN in Automation (CiA) CANsec specification task force, CiA's IG 04 SIG 01 TF 03 CAN XL security, contributing to the ongoing effort to define an efficient, standardized and interoperable security solution for CAN XL.

***Index Terms*—Automotive, In-Vehicle-Networks, CAN XL, Cryptographic Protocols, Standards, MACsec, CANsec**

1. Introduction

Modern vehicles are becoming increasingly connected, integrating advanced communication technologies [1] to support features such as autonomous driving [2], infotainment [3], and over-the-air updates [4]. While these advancements enhance functionality and user experience, they also expose vehicles to a growing range of cyberthreats, founded in the seminal work by Miller and Valasek [5]–[7] and since then extended by many other works, e.g., [8], [9] to cite only a few. Cyberattacks on vehicles can result in theft, unauthorized data access, or even manipulation of safety-critical systems, posing significant risks to both passengers and infrastructure.

Traditional in-vehicle networks, such as the Controller Area Network (CAN), were not designed with security in mind and lack built-in mechanisms for authentication, integrity, or confidentiality. As vehicles evolve into complex cyber-physical systems, the need for secure communication within the vehicle becomes paramount.

This need is further emphasized by increasing regulatory requirements aimed at ensuring cybersecurity throughout the vehicle lifecycle. Notable regulations include:

1) **EU Cyber Resilience Act (CRA)** [10]: Establishes cybersecurity obligations for digital products, including connected vehicles, across their entire lifecycle.
2) **UN ECE WP.29 Regulation No. 155** [11]: Requires manufacturers to implement a Cybersecurity Management System (CSMS) and maintain continuous threat monitoring for vehicles.
3) **ISO/SAE 21434** [12]: Provides a CSMS for managing cybersecurity risks in automotive development and operations.

This regulatory environment, combined with real-world risk assessments, leads to the necessity to secure communication at the data link layer. CAN XL, the next-generation CAN protocol, addresses this challenge by specifying CANsec[1] – a Layer 2 security protocol designed to protect CAN XL frames.

1. The latest version of the CANsec draft is accessible within CiA.

CANsec aims to provide authentication, integrity, confidentiality, and freshness for CAN XL communication, drawing inspiration from MACsec, the established security protocol for Ethernet Layer 2. By securing communication at the data link layer, CANsec ensures that even low-level data exchanges within the vehicle are protected against eavesdropping, tampering and unauthorized access. Moreover, reusing MACsec allows to employ very efficient implementations in hardware.

In contrast, a standalone CANsec implementation, as suggested for example by [13], would significantly increase the total cost of ownership, as it would require dedicated development and integration efforts despite the availability of mature hardware support for protocols like MACsec. This is particularly problematic in all those applications, where hardware resources are extremely limited and efficiency is paramount.

The scope of this paper is to propose and discuss an alternative method for reusing existing MACsec implementations and adapting them to the CAN XL environment. This approach leverages established concepts, security analyses, and implementations, as well as the existing ecosystem, including authentication and key agreement solutions.

The remainder of this paper is organized as follows:

- Section 3 provides background information on CAN XL and MACsec.
- Section 4 discusses the methodology for wrapping MACsec functionality within CANsec.
- Section 5 details how multiple traffic priority classes can be effectively handled within the CANsec framework and how we manage to optimize the MACsec bandwidth overhead.
- Section 6 discusses a more straight-forward, but less efficient, alternative mapping.
- Section 7 concludes the paper.

2. Related Works

Since two decades, there have been numerous attempts from industry and academia to secure the CAN bus. This is due to the security constraints brought by real-time communications, as e.g. discussed by Müller et al. [14] in the context of Ethernet, as well as the complexity of bus topologies. Those attempts can be divided into protocol proposals and industry standards.

The Secure Onboard Communication (SecOC) [15] protocol, as a representative of an industry standard specified by AUTOSAR, secures in-vehicle messages at the PDU level by appending a truncated message authentication code (MAC) along with optional truncated freshness value to the original payloay. It provides integrity, and authentication along with replay protection, but no confidentiality. However, due to the incomplete specification in [15], OEM's have extended it with proprietary ways for freshness management, different cipher suites and key management aspects. SecOC is defined on PDU level, thus independent on the transport medium, and hence does not protect all Layer 2 frame fields. This scattering into different flavors prevents a hardware-only implementation and thus always includes software-components, typically as part of an AUTOSAR software stack. Eventually, this results in a poor practical performance, and only a small amount of frames is protected in real-world applications.

Table 1. SUMMARY OF RELATED WORK

Proposal	CIA	Freshness	CAN XL	HW-impl
SecOC [15]	○●●	◐	◐	○
Yushev et al. [16]	●●●	●	◐	○
CANauth [19]	○●●	●	○	◐
LiBrA-CAN [20]	○●●	◐	◐	◐
LCAP [21]	●●●	●	◐	○
CaCAN [22]	○◐◐	◐	●	●
This work	●●●	●	●	●

CIA: Confidentiality Integrity Authentication
● full consideration ◐ partial consideration ○ no consideration

Other approaches, such as Yushev et al. [16], leverages TLS 1.2 [17], to provide end-to-end security over CAN. Similar to SecOC, and due to its complexity, TLS is never fully implemented in hardware and thus will not be able to protect all frames on an embedded fieldbus system.

The survey by Lotto et al. [18] provides a comprehensive analysis of the 15 most discussed proposals from academia. Of these, we present the most relevant ones for our work in the following, Table 1 summarizes them.

CANauth [19] protects classical CAN using the CAN+ protocol [23] as an out-of-band channel to transfer an 80-bit hash-based MAC (HMAC) tag, including counters for freshness. Relying on CAN+ prevents application to CAN XL and thus requires adoption, if it should be taken into account for CANsec. Specific cryptographic algorithms for the HMAC are not specified but are expected to be implementable in HW somewhat efficiently.

LiBrA-CAN [20] is designed to offer protection for CAN FD based on Mixed MACs (M-MACs), which uses an array of keys to compute a MAC and is verifiable by any of those keys. Freshness is ensured by message counters, but synchronisation is not covered. Making it usable for CANsec requires adoption to CAN XL. Further, applying M-MACs for the security checksums relies on non-standard cryptographic constructions, which are not available in today's hardware – and thus requires developing new microcontrollers with new cryptographic hardware accelerators that are not reusable for other tasks.

LCAP [21] is developed for classical CAN and uses hash chains. The protocol proposes a 5-step encrypted handshake between each pair of sending and receiving nodes, where the established session keys are used to encrypt the communication and the hash-chain ensures freshness. However, in addition to standard CAN IDs, LCAP requires additional identifiers, unique for each handshake step and for each sending and receiving ECU. This is a scalability issue and makes it less suitable as a basis for a CANsec consideration. Maintaining the hash-chain requires handling a state of size linear in the number of messages each node processes, which induces a big overhead in implementations.

CaCAN [22] follows a similar approach with a central authentication point, the "monitor node" (MN), that validates messages in real-time and cancels unauthenticated frames by injecting error frames. This approach does not provide confidentiality and relies on the central MN – if the MN is separated from the network, security breaks down. However, from implementation efficiency, this is realizable on CAN XL as well and requires no changes in the actual sending and receiving nodes.

To conclude, previous security approaches on CAN prove feasibility. However, due to the flexibility of automotive architectures and requirements of different manufacturers, security on CAN remains fragmented across formats, key management, and deployment models. Therefore, manufacturers typically test and implement proprietary or disclosed solutions. The shift to CAN XL, mixed with CAN/CAN-FD coexistence, will magnify these inconsistencies and further emphasizes the need for a standardized security layer for the CAN bus. Our CANsec approach aims to define this baseline by finalizing the conceptual core of this layer, borrowed from Ethernet and enabling interoperable, real-time security for CAN XL at scale.

3. Background

CANsec operates at the Data Link Layer (DLL) of the CAN protocol stack, specifically as the highest sublayer of the DLL. Its implementation is designed to process CAN XL Logical Link Control (LLC) frames, securing data exchange against eavesdropping, tampering and unauthorized authentication.

We will explain the existing CAN XL and MACsec specifications in Section 3.1 and Section 3.2 respectively, before providing more details of our CANsec proposal in Section 4.

3.1. CAN XL

An unprotected CAN XL LLC frame, as shown in Fig. 1, see also [24], makes use of the following fields:

- **Priority ID (11-bit)**: Determines bus access during arbitration.
- **RRS / FTYPE (1-bit)**: Frame type indicator.
- **Format (3-bit)**: Fixed to 011_b, includes IDE, FDF, and XLF fields.
- **Service Data Type (SDT, 8-bit)**: Comparable to Ethernet's EtherType, identifies the payload type.
- **Simple Extended Content (SEC, 1-bit)**: Indicates use of CAN XL extensions (e.g., CANsec or fragmentation).
- **Data Length Code (DLC, 11-bit)**: Encodes payload length in bytes, where $\mathrm{DLC} = \mathrm{length}(\mathrm{payload}) - 1$.
- **Virtual CAN ID (VCID, 8-bit)**: Comparable to Ethernet's VLAN ID in Q-tags [25].
- **Acceptance Field (AF, 32-bit)**: Identifies the frame's content.
- **Data field** (up to 2048-byte): Contains the payload.

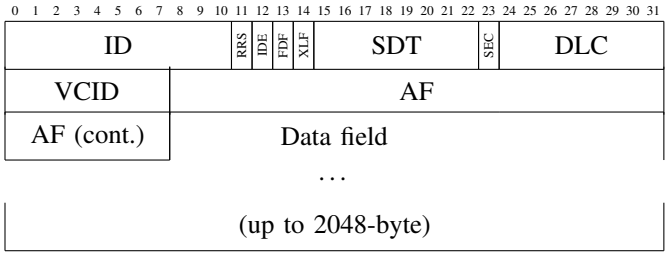

Figure 1. Frame format of a CAN XL LLC frame.

In case the SEC bit is set, the data field is required to start with a 3-bit Add-On Type (AOT) (specifying the actual extension used) and a 1-bit SECN field (indicating if another extension follows), see [26].

3.2. MACsec and Ethernet

MACsec [27] operates at Layer 2 of the Ethernet protocol stack, securing IEEE 802.3 Ethernet frames [28]. While MACsec in principal can be applied hop-by-hop (i.e. protect each link segment individually) or end-to-end (from the sending end station to the receiving end station with transparent switches inbetween), the automotive industry plans only hop-by-hop applications. In contrast to end-to-end applications, hop-by-hop usage drastically reduces the keys to be handled, which eventually results in cheaper hardware implementations.

An unprotected Ethernet frame includes:

- **Destination Address (DA, 6 bytes)**: Identifies the receiving node.
- **Source Address (SA, 6 bytes)**: Identifies the sending node.
- **MAC Service Data Unit (MSDU, up to 1502 bytes)**:
  - **EtherType (2 bytes)**: Indicates the payload type.
  - **MAC Client Data (up to 1500 bytes)**: Contains the actual payload.

Additional functionalities are provided via extensions such as the 4-byte Q-tag, which encodes frame priority and VLAN IDs for logical segmentation of a physical LAN.

MACsec introduces another logical segmentation mechanism called *Connectivity Associations* (CAs), grouping nodes that can securely communicate. Communication of nodes within a CA are protected against unauthorized access from outside the CA.

MACsec provides the following security guarantees:

- Confidentiality of the MSDU,
- Integrity of the frame,
- Authenticity of the data origin,
- Freshness of the communication.

The MACsec security framework within a CA includes:

- **Authentication**: May involve pre-shared keys or protocols like EAP-TLS [29].

- **Key Agreement**: [29] specifies the MACsec Key Agreement (MKA).
- **Data Plane**: Focus of this work.

The control plane, consisting of the authentication and key agreement parts, standardized in [29], are out of scope for this work.

In an actual system, the MACsec data plane is implemented by the *MACsec Security Entity* (SecY), which represents the node's participation in one CA. Within the data plane, nodes establish unidirectional (multicast) *Secure Channels* (SCs). Each node has at least one transmit SC, and MACsec supports up to eight transmit SCs per node. Multiple SCs are useful for handling different traffic priority classes and ensuring in-order frame reception, especially when the network may reorder frames, e.g. based on the frame's priority.

SCs are unidirectional because their state includes the last used packet number, which serves as a freshness value and nonce in the underlying cipher suite. Thus, ensuring uniqueness and preventing reuse of packet numbers is paramount for MACsec's security.

Each Secure Channel (SC) is uniquely identified by its *Secure Channel Identifier* (SCI), an 8-byte value. The SCI consists of two components: a *system identifier*, typically the nodes MAC address, and a 2-byte *port identifier*. It is important to note that the system identifier is not necessarily equal to the frame's source address.[2]

Each SC contains one or more *Secure Associations* (SAs), with up to four theoretically possible. The SA is identified by the 2-bit *Association Number* (AN). In practice, two parallel SAs are used to mask key establishment delays during MKA operation.

The MACsec Protocol Data Unit (MPDU) structure is shown in Fig. 2. The Tag Control Information (TCI) includes the following bitfields:

- **Version (V)**
- **End Station (ES)**
- **Single Copy Broadcast (SCB)**
- **Secure Channel (SC)**
- **Encryption (E)**
- **Changed Text (C)**

For this work, the SCB functionality is not relevant, as it is used for a specific optimization in passive optical networks, and we assume the SCB bit is not set. The following aspects of the TCI are relevant:

- The **SC bit** is set if the Secure Channel Identifier (SCI) is encoded in the SecTAG.

2. Consider a hop-to-hop application of MACsec involving end station A, a switch, and end station B. A and the switch form one Connectivity Association (CA), while the switch and B form another. When A sends a MACsec-protected frame to B, the switch verifies and strips the MACsec protection upon reception. When forwarding the frame to B, the switch re-applies MACsec protection. However, the frame's source address remains unchanged, as it is simply forwarded within the LAN. Therefore, the switch's MACsec SecY must use a system identifier that differs from the frame's source address.

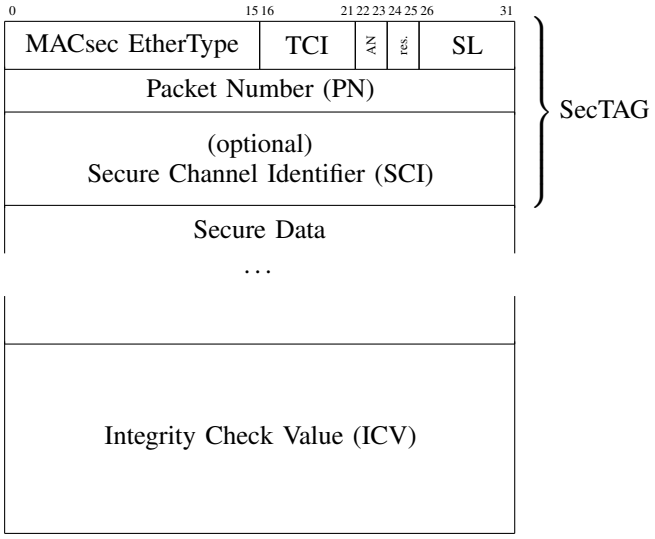

Figure 2. Frame format of the MACsec Protocol Data Unit (MPDU).

- The **ES bit** is set if the frame's Source Address determines the SCI. In this case, the SC bit is not set, and the SCI is not encoded in the SecTAG.
- If the ES bit is set, the SCI's port identifier is set to `0x00 01`.
- If the **E** and **C bits** are set, the full frame is authenticated and the MSDU is encrypted.
- If the E and C bits are not set, the full frame is authenticated but the MSDU is transmitted in plaintext.

The *Integrity Check Value* (ICV) contains the authentication tag, and is in practice 16-byte long.

4. Reusing MACsec for CANsec

MACsec has been available for two decades as a mature and widely adopted Layer 2 security protocol for Ethernet. Its robustness and standardization make it a compelling candidate for securing other link-layer technologies. In parallel, CANsec has been developed to provide similar Layer 2 security guarantees for CAN XL, aiming to replicate MACsec's functionality in the context of automotive networks.

The CANsec specification has been under active development since 2020, with the goal of mimicking MACsec's architecture and security properties. However, a finalized version of CANsec is still pending. This delay motivates our proposal: rather than designing a new protocol from scratch that imitates MACsec, we suggest directly reusing MACsec itself and adapting it for use with CAN XL.

Our approach centers around leveraging an existing MACsec implementation, which represents one (or more) SecYs. Our proposal acts as a lightweight wrapper around this implementation, defined by two mapping functions, f and g:

- f: Maps a CAN XL LLC frame to a virtual Ethernet frame.

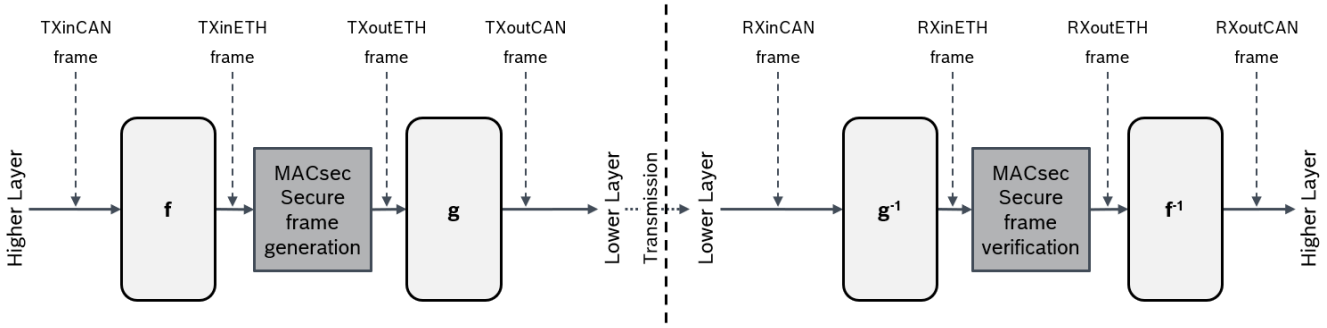

Figure 3. CANsec flow and wrapping of MACsec.

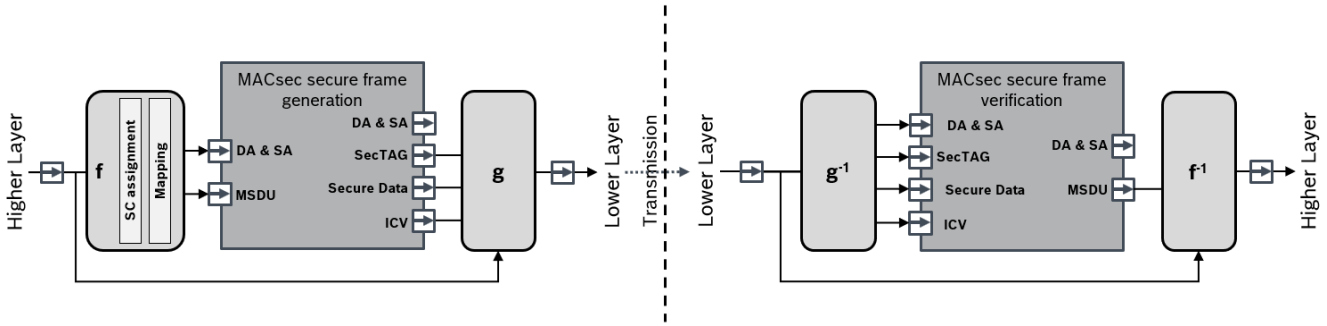

Figure 4. CANsec wrapping details.

- g: Maps the protected virtual Ethernet frame back to a protected CAN XL LLC frame.

Fig. 3 illustrates this process and gives names for the in- and output frames of f and g, as well as their inverses ("TXinCAN", etc).

To fully specify CANsec, it is necessary to define the functions f, g, their inverses f^{-1} and g^{-1}, and the CANsec frame format. Details of the inputs and outputs of f, g, and their inverses are shown in Fig. 4.

Function f takes the TXinCAN frame and produces a virtual Ethernet frame "TXinETH", which includes an Ethernet destination and source address, as well as an MSDU. The MACsec SecY then processes this frame, producing a SecTAG, secure data, and an Integrity Check Value (ICV) – collectively forming the MSDU of the "TXoutETH" secured virtual Ethernet frame. Function g maps this to the "TXoutCAN" CAN XL LLC frame.

On the reception side, g^{-1} maps the received CAN XL frame (RXinCAN) to the secured virtual Ethernet frame (RXinETH), consisting of destination and source address, SecTAG, secure data, and ICV. The MACsec verification outputs the original MSDU, which is then mapped by f^{-1} to the "RXoutCAN" CAN XL LLC frame.

To better describe the components involved in these mappings, we introduce the CANsec Protocol Data Unit (CPDU) structure, shown in Fig. 5. This figure, based on [27, Figure 8.1], highlights the CAN XL LLC frame fields that are passed through the CANsec protection: format bits (IDE, FDF, XLF), SDT, DLC, VCID, and AF. Among these, only the DLC field is adjusted to account for CANsec-induced overhead.

The original data field, the CANsec SDU (CSDU), is protected by CANsec and transformed into the CPDU. The CPDU consists of:

- **CANsec Protocol Control Information (CPCI)**: Header fields introduced by CANsec.
- **CANsec Encryption Padding (CEP)**: Padding required due to mapping details, explained in Section 6.
- **CANsec Secure Data (CSD)**: The protected (and possibly encrypted) CSDU.
- **Integrity Check Value (ICV)**: Ensures frame integrity and authenticity.

The proposed format for the CPDU is shown in Fig. 6 CAN XL requires add-on functionalities such as CANsec to identify their usage by setting the SEC bit to 1. Further, the data field then needs to start with an AOT and SECN field, indicating the type of add-on used and potentially further add-ons, [26]. The CPCI further contains:

- VN, 1 bit, encoded version number of CANsec (for now VN=0, indicating CANsec Version 1),
- CE, 1 bit, *CANsec Encrypt*, indicating whether authentication-only (CE=0) or authenticated-encryption (CE=1) is applied,
- AN, 2 bit, taken over from MACsec,
- PN, 32 bit, taken over from MACsec, and

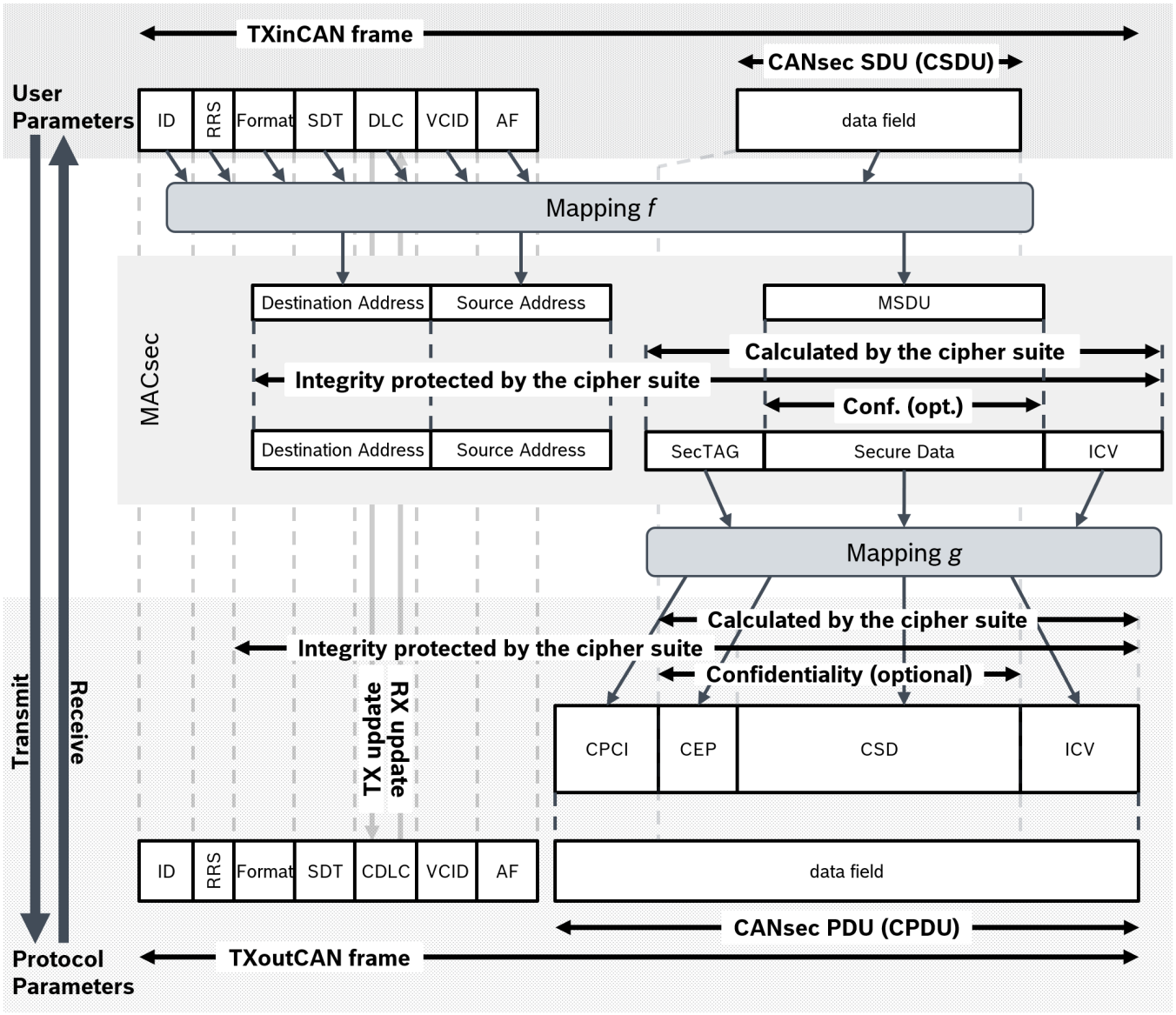

Figure 5. CANsec frame parts.

- CSCI, 16 bit, *CANsec SCI*, described in Section 5.

The transmit input mapping f, see Fig. 7, outputs the TXinETH frame, consisting of the destination / source address, EtherType and MAC client data. Further, f identifies the CSCI by a user-configured SC lookup function:

$$\mathbb{F}_2^{11} \times \mathbb{F}_2^8 \times \mathbb{F}_2^8 \times \mathbb{F}_2^{32} \to \mathbb{F}_2^{11}$$
$$\mathrm{ID}, \mathrm{SDT}, \mathrm{VCID}, \mathrm{AF} \mapsto \mathrm{CSCI}$$

The resulting destination address (DA) contains the identified CSCI, the format fields (IDE, FDF, and XLF) from TXinCAN, as well as the SEC bit that will be send in TXoutCAN (and thus is fixed to 1). The destination address also contains the AOT and VN fields. The AOT field is set to the value 001_b, identifying the CANsec extension, the VN field is set to the value 0_b, identifying CANsec Version 1. Remaining bits are zero-padded.

The source address contains the SDT, VCID fields, and the AF from TXinCAN.

EtherTypes that are smaller than 1500 are treated as length field, not as type field [28]. To avoid any dependencies and special handling in real MACsec implementations, f outputs a padded EtherType, that always starts with 1000_b, rendering it greater than 1500. The remaining bits of the EtherType are set to the SECN bit (= `TXinCAN.SEC`) followed by the CANsec DLC (CDLC), where $\mathrm{CDLC} = \mathrm{DLC}{+}8{+}2{+}16$. This adds the CANsec overhead to the payload length encoding of CAN XL. The overhead results from 8 bytes for the CPCI, 2 bytes for the CEP and 16 bytes for the ICV. While the MACsec implementation may encode an SCI in

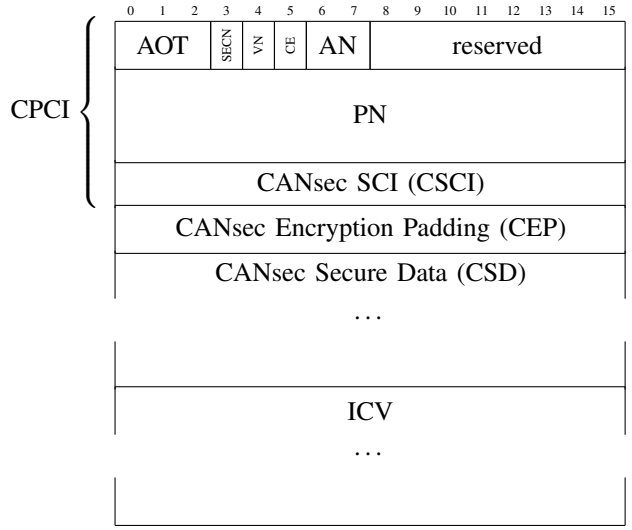

Figure 6. Frame format for CANsec PDUs (CPDUs)

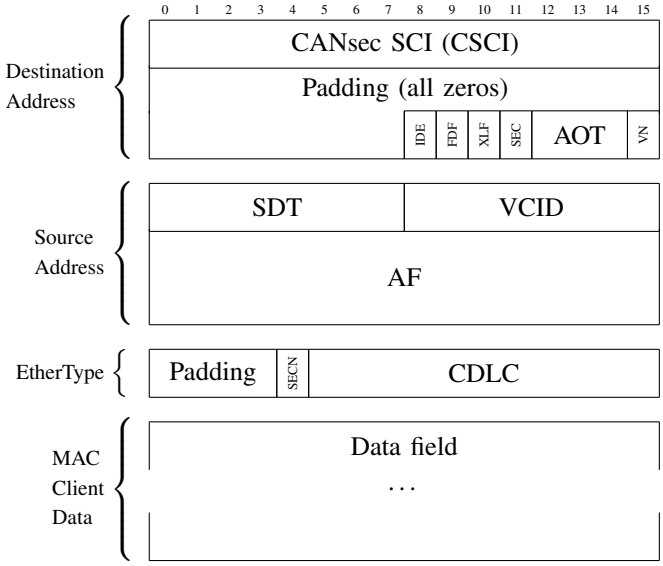

Figure 7. The mapping f, outputting TXinETH, consisting of the fields destination/source address, EtherType and MAC client data.

the SecTAG, we later discuss how this can be optimized with the help of the CSCI, see Section 5.

Finally, the payload from TXinCAN's data field is copied to the MAC client data.

The resulting TXinETH is then fed into the MACsec secure frame generation implementation, resulting in the MACsec protected TXoutETH, consisting of destination/-source address, SecTAG, secure data and ICV.

The transmit output mapping g converts this to TXoutCAN, by setting the CAN XL LLC header fields ID, RRS, IDE, FDF, XLF, VCID and AF to the values of the corresponding fields from TXinCAN. The TXoutCAN.SEC field is fixed to 1, indicating the use of CANsec. The TXoutCAN.DLC field is set to CDLC. The data field contains the CPDU and is build from the SecTAG, secure data and

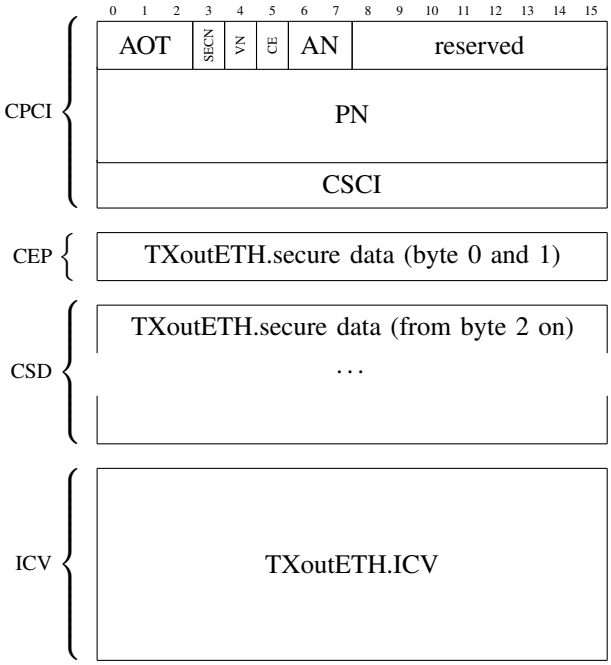

Figure 8. CANsec PCDU (CPDU) as created by the mapping g.

ICV, see Fig. 8.

On receiving side, the original MACsec SecTAG can be reconstructed, as the MACsec EtherType (`0x88 E5`) is fixed and known, and the other SecTAG fields not transported by the CPCI can be reconstructed as:

- V = 0,
- ES = 0,
- SC = 1,
- SCB = 0,
- E = CE,
- C = CE,
- SL from the encoded payload length in DLC,
- SCI from the CSCI, as described below.

Fig. 9 shows g^{-1}, mapping the received, protected RXinCAN frame to the virtual Ethernet frame RXinETH.

RXinETH can now be given to MACsec's secure frame verification function, that checks the cryptographic checksum and (in case of encrypted secure data) decrypts the payload. The output of MACsec, RXoutETH, is then mapped back again to a CAN XL LLC frame by f^{-1}. The receive output mapping f^{-1} sets the RXoutCAN ID, RRS, format, SDT, VCID and AF fields to the corresponding values from RXinCAN. RXoutCAN.SEC is set to the SECN bit, received in RXinCAN.CPCI.SECN. RXoutCAN.DLC is computed from the CDLC by subtracting the CANsec overhead: $\mathrm{DLC} = \mathrm{CDLC} - 8 - 2 - 16$. Eventually, the RXoutCAN.data field is set to the payload of the virtual Ethernet frame, namely the MAC client data in RXoutETH.MSDU.

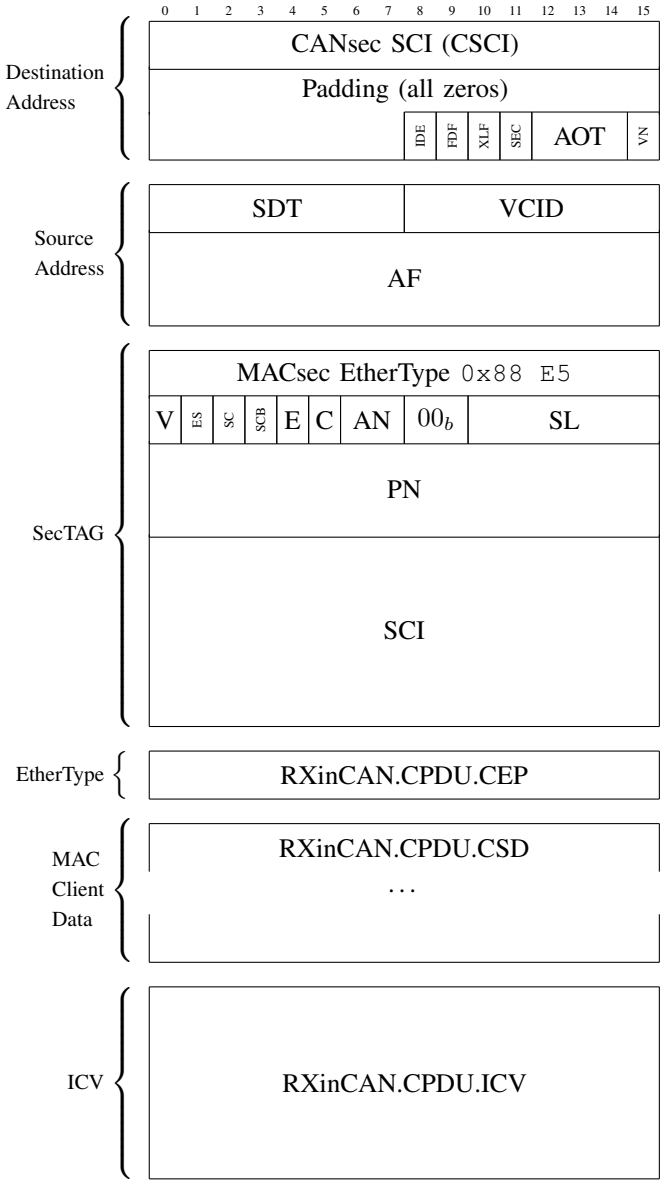

Figure 9. The mapping g^{-1}, outputting RXinETH, consisting of the fields destination/source address, SecTAG, EtherType, MAC client data and ICV.

5. Multiple transmit SCs, CANsec SCI and optimizing the SCI field

As described in Section 3, MACsec can handle multiple transmit SCs to separate traffic based on priority classes. In [27] this is specified via a so called *traffic class table* (TCT), which – on look up of the frames priority (encoded in the QTag.PCP field) – sets the four most significant bits (MSBs) of the SC's port identifier. However, using the traffic class table has some disadvantages:

1) we are not aware of any automotive MACsec implementation that supports TCTs,
2) Open Alliance currently discusses to overwrite the MSBs of the port identifier for a shared-medium-optimized MKA variant [30], rendering any approach relying on the TCT incompatible, and
3) encoding the QTag increases the bandwidth overhead, see also Section 6.

We thus propose a different approach to handle multiple transmit SCs, that is less restrictive and allows further optimizations.

For this, note that typical MACsec implementations can support multiple transmit SCs, where these transmit SCs do not correspond to the same MACsec SecY (and are therefore not differentiated by the TCT), but instead represent distinct SecYs. The MACsec implementation thus needs to assign an incoming frame, based on some criteria, to the correct transmit SC. This is as well discussed in the Open Alliance TC17 [31]: the current draft requires implementations to evaluate at least the destination address and VLAN ID of an transmitting frame for SC assignment.

We propose the following, two-step, SC handling:

1) **CANsec SC lookup**: as described in the previous section, look up the CSCI based on ID, SDT, VCID and AF fields. Then encode the identified CSCI in the destination address of TXinETH.
2) **MACsec SC assignment**: assign the transmit SC based on the destination address, i.e. $\mathrm{DA} \mapsto \mathrm{SCI}$.

This SC handling needs to be initially configured.[3] Thus, the CANsec configuration interface needs to handle the corresponding correct configuration of its internal SC lookup functionality, as well as MACsecs's SC assignment.

To lift a further optimization potential here, we propose to use SCIs of a particular form that depends only on the CSCI. One example is, to fix a prefix (can for example be done by the system architect) and to then construct the SCI as:

$$\mathrm{CSCI} \mapsto \mathrm{SCI} = \mathrm{prefix} \mid\mid \mathrm{CSCI} \mid\mid \texttt{0x00 01}\ , \qquad (1)$$

where `0x00 01` is the default port identifier for a SecY with a single transmit SC only.

Given this dependency between the CSCI and the SCI, we only need to transmit either the CSCI or the SCI, as both can be derived from the other value.

We encode the CSCI in the CPCI, see Fig. 6, and leave out the SCI. The transmit output mapping g thus creates the CPCI header, without including the SCI from the MACsec output (TXoutETH) and hence saves 8 Byte compared to MACsec's SecTAG. To ensure that MACsec also always encodes the SCI in the SecTAG, the CANsec configuration interface configures the MACsec implementation to `alwaysIncludeSCI = True` [27, Fig. 10-5].

On reception side, g^{-1} recreates the SCI, based on the transmitted CSCI and Eq. (1), and encodes it in the SecTAG of RXinETH.

Besides the mentioned savings in the bandwidth, this utilization of the MACsec implementation also allows CANsec to offer more than eight transmit SCs per node.

3. Note that today's MACsec implementations already need to offer configuration interfaces for this, because SC assignment happens in any typical automotive MACsec implementation.

6. Alternative mapping

The CAN XL LLC header fields are in total 72-bit long (see Fig. 1), while the Ethernet header fields (destination/-source address and EtherType) require $48 + 48 + 16 = 112$ bits. If additionally an Ethernet QTag is encoded, additionally 3 bit for the PCP and 12 bit for the VLAN ID are available. This leaves a lot of possibilities for mapping CAN XL LLC header fields to Ethernet header fields.

One obvious alternative mapping to our proposal is, to "map fields to their equivalents", i.e., fields that serve a similar purpose in CAN XL and Ethernet are mapped onto each other:

- SDT $\mapsto$ EtherType
- VCID $\mapsto$ QTag.VID
- ID $\mapsto$ QTag.PCP
- Data field $\mapsto$ MAC client data

Further, AF serves the purpose of addressing the target application, so mapping it to destination address seams meaningful. As mentioned, EtherTypes smaller 1500 are interpreted as length field, thus mapping the DLC to the EtherType would as well fits this strategy – however, due to mapping the SDT to EtherType, only 1 Byte is left free in the EtherType, which is not enough to include the full DLC. The DLC thus needs to be mapped to either the destination or source address. This leaves us with the five bits RRS, IDE, FDF, XLF and SEC, which do not have a corresponding field in Ethernet frames. However, any field still has enough free bits to take these five constant bits, so they can be distributed in any way.

The described mapping has one final problem to be fixed: the QTag.PCP field has only 3 bits, while the ID is 11-bit long. One possibility to fix this, is to partition the IDs in two parts. An often used scheme is a 3-bit long part identifying the actual priority class and an 8-bit long part identifying the sender. Using this, we can map the 3-bit priority class part of the ID to the QTag.PCP, while the 8-bit long sender part naturally would go to the source address.

Using this alternative mapping would strictly follow Ethernet and MACsec concepts (encoding priorities in the PCP field, using TCTs for multiple transmit SCs), but comes at the cost of increased bandwidth overhead.

In order to compare the induced overhead, we look at the initial payload length (length of the TXinCAN.data field or TXinETH.MAC client data) as well as the output payload length (length of the TXoutETH.MAC client data or TXoutCAN.data field) for our main proposal as well as for the alternative mapping. The mapped EtherType is – from the perspective of MACsec – part of the payload. As MACsec can optionally encrypt the payload, the whole payload needs to be transmitted on the wire. To highlight this, we term this part of the secured data the CANsec Encryption Padding (CEP). For our main proposal, the CEP is 2-byte long, including the EtherType only. For the alternative mapping, the CEP additionally includes the encoded QTag, adding another 4 Byte to the CEP length.

Table 2. COMPARING OVERHEADS FOR MAIN AND ALTERNATIVE MAPPING. COLUMNS INDICATE LENGTH OF PAYLOAD FIELD (DATA FIELD FOR CAN XL LLC FRAMES, MAC CLIENT DATA FIELD FOR ETHERNET FRAMES) IN BYTES, WHERE INPUT PAYLOAD IS N BYTE LONG.

	TXinETH	TXoutETH	TXoutCAN
Main	N+2	N+2 +32	N+2 +24
Alternative	N+2+4	N+2+4+32	N+2+4+32

The SecTAG (including the SCI) and ICV add further 32 Byte to the output. In our main proposal, we can save the 8 Byte long SCI, as described above, reducing the overall overhead to 24 Byte. Table 2 summarizes these results.

Overall, we end up with 26 Bytes of overhead for our main proposal versus 38 Bytes of overhead for the alternative, i.e., the alternative mapping increases bandwidth overhead by 46%.

Another downside of the alternative mapping is, that it allows to use only eight transmit SCs, while our main proposal supports as many as the MACsec implementation offers, up to a theoretical bound of 2^{11} minus the number of receive SCs.

7. Conclusion

Our work demonstrates a practical Layer 2 security design for CAN by reusing the proven MACsec engine along with functions that map CAN frames into Ethernet frames. The resulting CANsec protocol keeps the same MACsec security properties while trimming per-frame overhead from 32 Bytes to 26 Bytes (a 20% reduction in bandwidth overhead). An alternative construction, allowing a more logical and straightforward mapping was also discussed. However, it results in 12 Bytes additional overhead and a tight limitation on the number of transmit SCs.

The approach favors the familiarity with the well-established MACsec protocol and opens clear migration paths from MACsec. What remains is the control-plane: in future work the adaption of the MACsec Key Agreement (MKA) to work for CANsec needs to be scrutinized. This has to include mappings of MKPDUs to CAN XL frames as well as how CANsec secure channel handling, including the CSCI, can be supported by an adapted MKA.

We will further work on full implementations and validations of the proposed solution, transferring this approach to legacy CAN FD systems, as well as continuing driving the CiA specification.

References

[1] V. Antinyan, "Revealing the complexity of automotive software," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 1525–1528.

[2] H. Bardt, "Autonomous driving—a challenge for the automotive industry," *Intereconomics*, vol. 52, no. 3, pp. 171–177, 2017.

[3] A. K. Mandal, A. Cortesi, P. Ferrara, F. Panarotto, and F. Spoto, "Vulnerability analysis of android auto infotainment apps," in *Proceedings of the 15th ACM International Conference on Computing Frontiers*, ser. CF '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 183–190. [Online]. Available: https://doi.org/10.1145/3203217.3203278

[4] M. Shavit, A. Gryc, and R. Miucic, "Firmware update over the air (fota) for automotive industry," SAE Technical Paper, Tech. Rep., 2007.

[5] C. Miller and C. Valasek, "Adventures in automotive networks and control units," *Def Con*, vol. 21, no. 260-264, pp. 15–31, 2013.

[6] ——, "A survey of remote automotive attack surfaces," *Black Hat USA*, vol. 2014, no. 2014, p. 94, 2014.

[7] ——, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, no. 2015, pp. 1–91, 2015.

[8] F. Sommer, J. Dürrwang, and R. Kriesten, "Survey and classification of automotive security attacks," *Information*, vol. 10, no. 4, 2019. [Online]. Available: https://www.mdpi.com/2078-2489/10/4/148

[9] M. Rumez, D. Grimm, R. Kriesten, and E. Sax, "An overview of automotive service-oriented architectures and implications for security countermeasures," *IEEE Access*, vol. 8, pp. 221 852–221 870, 2020.

[10] Council of European Union, "Regulation (EU) 2024/2847 of the European Parliament and of the Council of 23 October 2024 on horizontal cybersecurity requirements for products with digital elements and amending Regulations (EU) No 168/2013 and (EU) 2019/1020 and Directive (EU) 2020/1828 (Cyber Resilience Act) (Text with EEA relevance)," 2024, http://data.europa.eu/eli/reg/2024/2847/oj.

[11] "UN Regulation No. 155 – Uniform provisions concerning the approval of vehicles with regards to cybersecurity and cybersecurity management system [2021/387]," pp. 30–59, Mar 2021.

[12] "Road vehicles — Cybersecurity engineering," International Organization for Standardization, Geneva, CH, Standard, Aug. 2021.

[13] F. Wiemer and A. Zeh, "Enabling secure communication for automotive endpoint-ecus through lightweight-cryptography," in *Proceedings of the 7th ACM Computer Science in Cars Symposium*, ser. CSCS '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3631204.3631861

[14] T. Müller, A. Walz, M. Kiefer, H. Dermot Doran, and A. Sikora, "Challenges and prospects of communication security in real-time ethernet automation systems," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2018, pp. 1–9.

[15] "Specification of Secure Onboard Communication Protocol," AUTOSAR, Standard, Nov. 2020.

[16] A. Yushev, M. Barghash, M. P. Nguyen, A. Walz, and A. Sikora, "Tls-over-can: An experimental study of internet-grade end-to-end communication security for can networks," *IFAC-PapersOnLine*, vol. 51, no. 6, pp. 96–101, 2018, 15th IFAC Conference on Programmable Devices and Embedded Systems PDeS 2018.

[17] E. Rescorla and T. Dierks, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, Aug. 2008. [Online]. Available: https://www.rfc-editor.org/info/rfc5246

[18] A. Lotto, F. Marchiori, A. Brighente, and M. Conti, "A Survey and Comparative Analysis of Security Properties of CAN Authentication Protocols," *IEEE Communications Surveys & Tutorials*, vol. 27, no. 4, pp. 2470–2504, 2025.

[19] A. V. Herrewege, D. Singelee, and I. Verbauwhede, "CANauth - a simple, backward compatible broadcast authentication protocol for can bus," *ECRYPT Workshop on Lightweight Cryptography*, p. 20, 2011.

[20] B. Groza, S. Murvay, A. Van Herrewege, and I. Verbauwhede, "LiBrA-CAN: Lightweight Broadcast Authentication for Controller Area Networks," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 3, 2017.

[21] A. Hazem and H. Fahmy, "LCAP - a lightweight can authentication protocol for securing in-vehicle networks," *10th escar Embedded Security in Cars Conference*, vol. 6, p. 172, 2012.

[22] R. Kurachi, Y. Matsubara, H. Takada, N. Adachi, Y. Miyashita, and S. Horihata, "CaCAN - Centralized Authentication System in CAN (Controller Area Network)," *14th Int. Conf. on Embedded Security in Cars (ESCAR 2014)*, p. 9, 2014.

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

[24] "Road vehicles - Controller area network (CAN) - Part 1: Data link layer and physical coding sublayer," International Organization for Standardization, Geneva, CH, Standard, May 2024.

[25] "IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks," *IEEE Std 802.1Q-2014 (Revision of IEEE Std 802.1Q-2011)*, pp. 1–1832, 2014.

[26] "CAN XL add-on services – Part 1: Simple/extended content (SEC) indication," CAN in Automation, Draft, 2025.

[27] "IEEE Standard for Local and metropolitan area networks-Media Access Control (MAC) Security," *IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006)*, pp. 1–239, 2018.

[28] "IEEE Standard for Ethernet," *IEEE Std 802.3-2022 (Revision of IEEE Std 802.3-2018)*, pp. 1–7025, 2022.

[29] "IEEE Standard for Local and Metropolitan Area Networks–Port-Based Network Access Control," *IEEE Std 802.1X-2020 (Revision of IEEE Std 802.1X-2010 Incorporating IEEE Std 802.1Xbx-2014 and IEEE Std 802.1Xck-2018)*, pp. 1–289, 2020.

[30] OPEN Alliance, "OPEN Alliance internal discussion on candidates for 10BASE-T1S optimized MKA variants," 2025.

[31] ——, "Automotive 10BASE-T1S MACsec specification v0.4 draft," OPEN Alliance internal draft document, Sep. 2025.