




# Foundational security from Ethernet to CAN

## MACsec, CANsec & the FD Adaptation Layer

Friedrich Wiemer and Arthur Mutter
Robert Bosch GmbH, September 16th, 2026

Public C-SC0

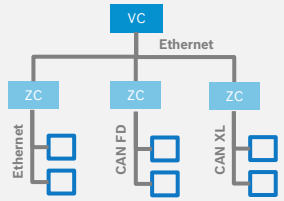
BOSCH

# Motivation
## E/E Architecture Trend

Future

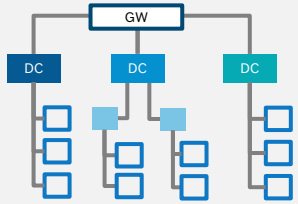
Past

### Vehicle-centralized



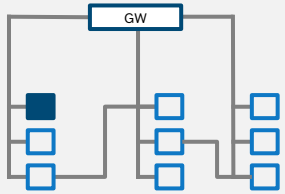

- **Elements**: Vehicle Computers (VC), Zone Contr. (ZC), Edge Nodes
- **Orientation**: Geographical (no domain-orientation)
- **Functions**: strongly centralized
- **Typical L1/2:** Ethernet, CAN

### Domain-centralized




- **Elements**: Gateway (GW), Domain Controller (DC), Single ECUs
- **Orientation**: Function Domain
- **Functions**: partly centralized (in DC)
- **Typical L1/2:** Ethernet, CAN, Flexray, LIN, ...

### Distributed




- **Elements**: Gateway (GW), Single ECUs
- **Orientation**: Function Domain
- **Functions**: not centralized, “One function = One Box”
- **Typical L1/2:** mostly CAN, LIN

**Centralization**

**Data travels longer distance**

**New/More Requirements for Layer 2 Communication**

- Transport of any HLP
- Time Synchronization
- Quality of Service
- Safety (Integrity & Availability)
- **Security (for Ethernet & CAN)**
- Scalability & Cost efficiency



Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

HLP: Higher Layer Protocol
SDV: Software Defined Vehicle

BOSCH

# Motivation

## How to protect communication **in Ethernet**

**Available protocols for secure in-vehicle comm.**

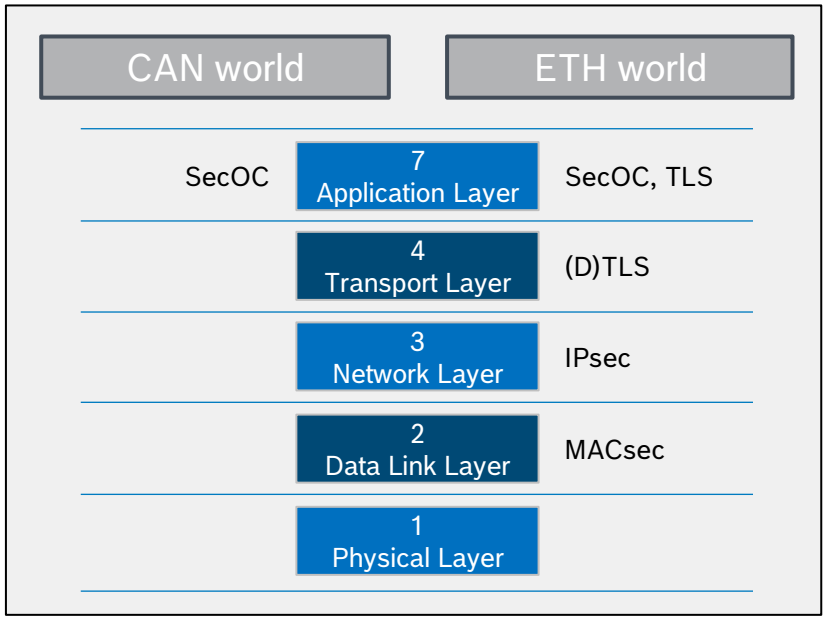
- **SecOC** (CAN, FR, ETH),
- **(D)TLS** (ETH) for TCP (UDP)
- **IPsec** (ETH) for IP,
- **MACsec** (ETH) for MAC / LLC
- “ASAsec” (ASA ML)
- MIPI CSE (MIPI)

**Legend**

Confidentiality / Authenticity

Standardized / Custom Implementations

Communication and security protocols




**Layer 2 security can protect all traffic** within a network.
**MACsec** is the favorite choice.



Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

BOSCH

# What is MACsec

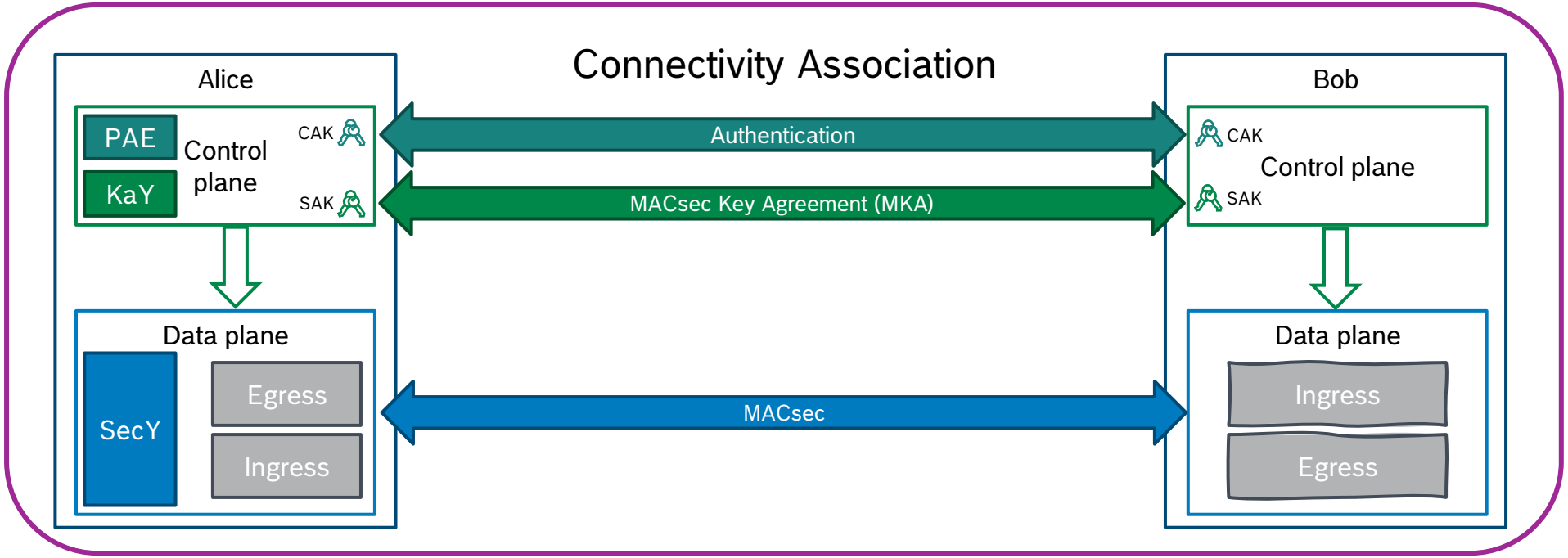


Public C-SC0

BOSCH

# What is MACsec
## MACsec Communication Architecture

CAK Secure Connectivity Association Key
SAK Secure Association Key
(both 128- or 256-bit AES keys)

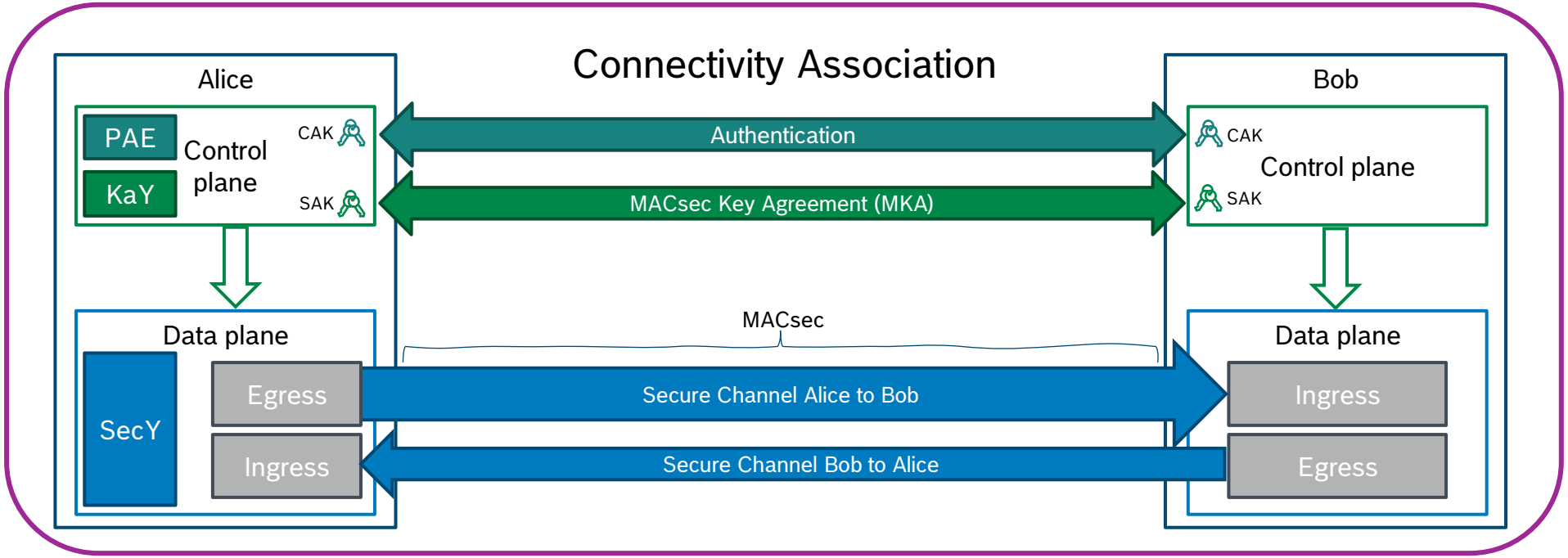



- Communication separated in three steps, first two (Authentication & MKA) covers the control plane
- MACsec, the actual secure communication protocol, represents the data plane


Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

# What is MACsec
## MACsec Communication Architecture

CAK Secure Connectivity Association Key
SAK Secure Association Key
(both 128- or 256-bit AES keys)




- MACsec splits into unidirectional secure channels (multicast)
- Per connectivity association (with n nodes): at least
  - one transmitting secure channel per node and (n-1) receiving secure channel per node

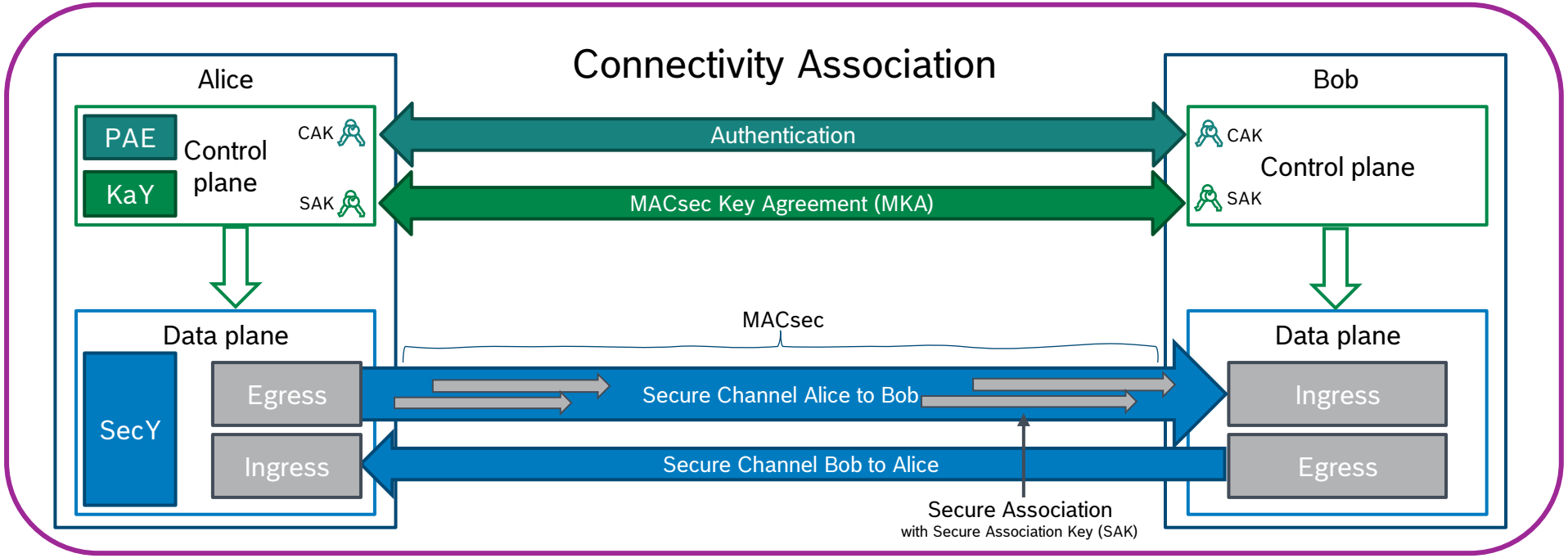
Required to prevent collisions in freshness value


Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

# What is MACsec

## MACsec Communication Architecture

CAK Secure Connectivity Association Key
SAK Secure Association Key
(both 128- or 256-bit AES keys)




- Each secure channel contains up to four secure associations
- Each secure association knows one Secure Association Key (SAK) and further details

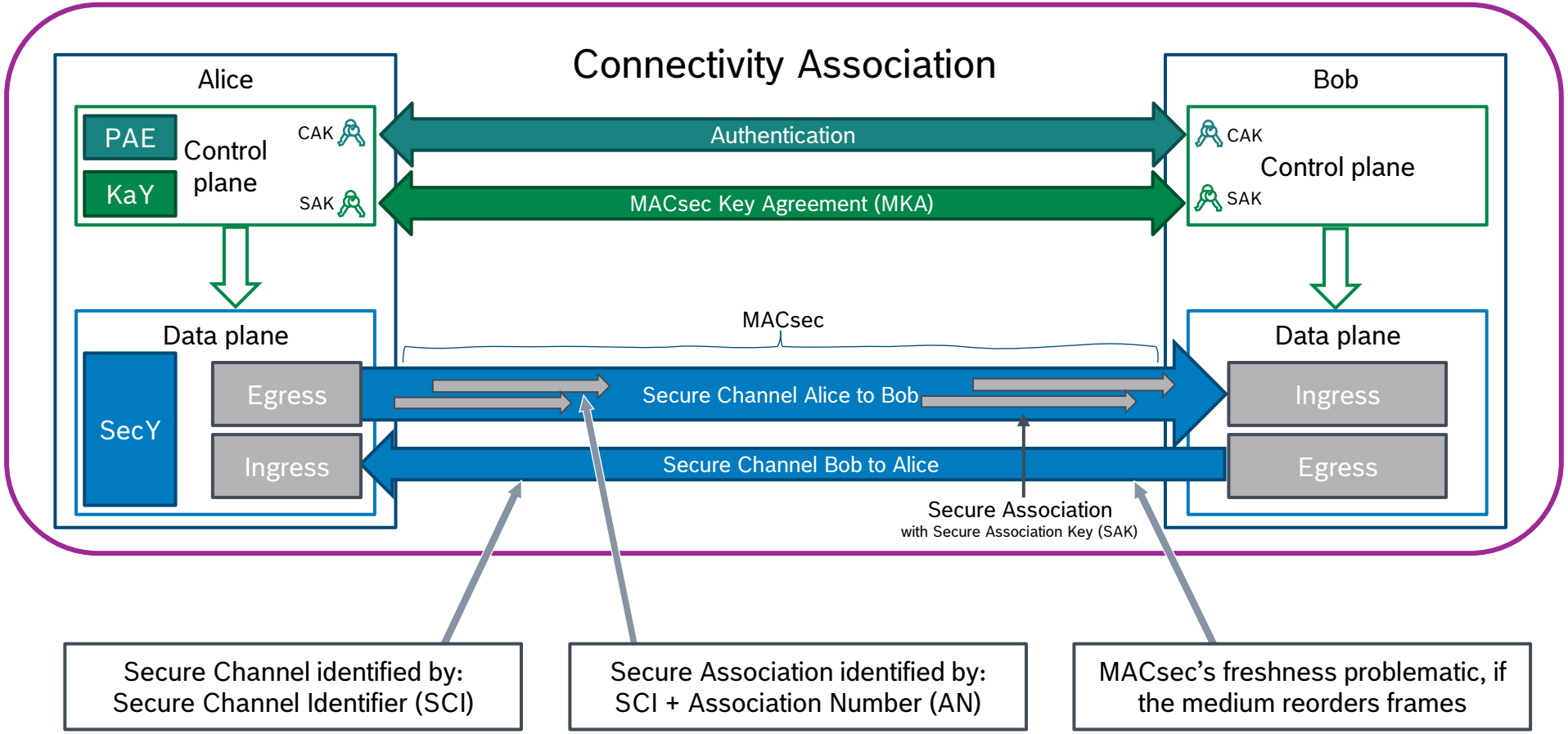
Required to hide key agreement time


Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

# What is MACsec

## MACsec Communication Architecture

CAK Secure Connectivity Association Key
SAK Secure Association Key
(both 128- or 256-bit AES keys)



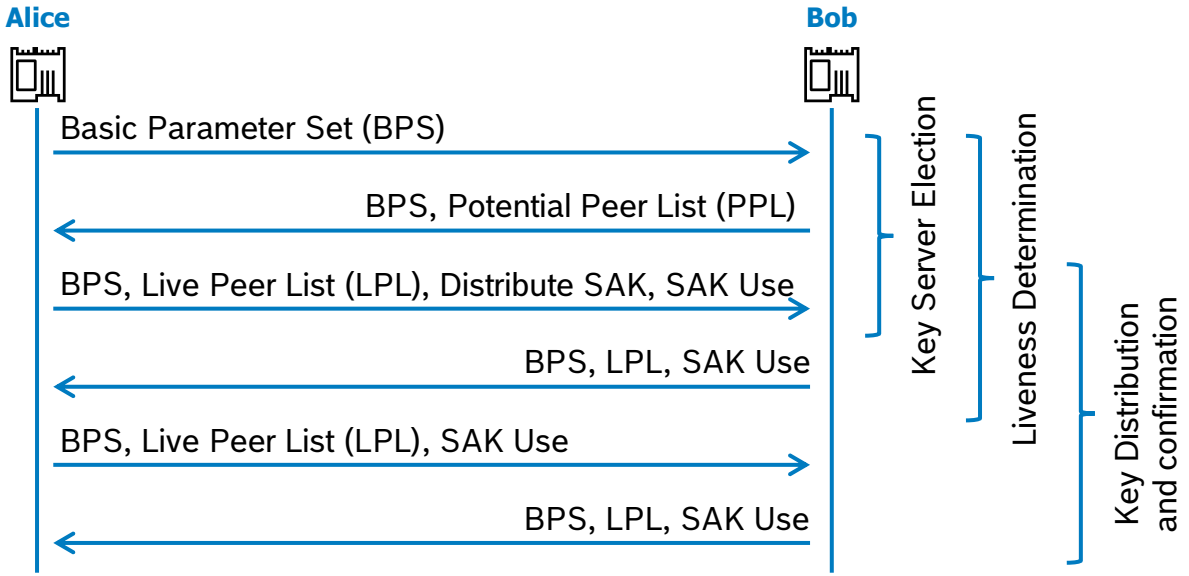



Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

BOSCH

# What is MACsec

## MACsec Key Agreement and the Automotive Profile







OA TC17 Goal: Reduce Time-To-Key-Agreement (TTKA)

- Speed up group formation
- Reduce bandwidth needs

Simulations and first plugfest show very promising results. We will have a first draft ready for IEEE Tech Day Mid-Oct.



Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

BOSCH

# CAN security



Public C-SC0

BOSCH

# Motivation

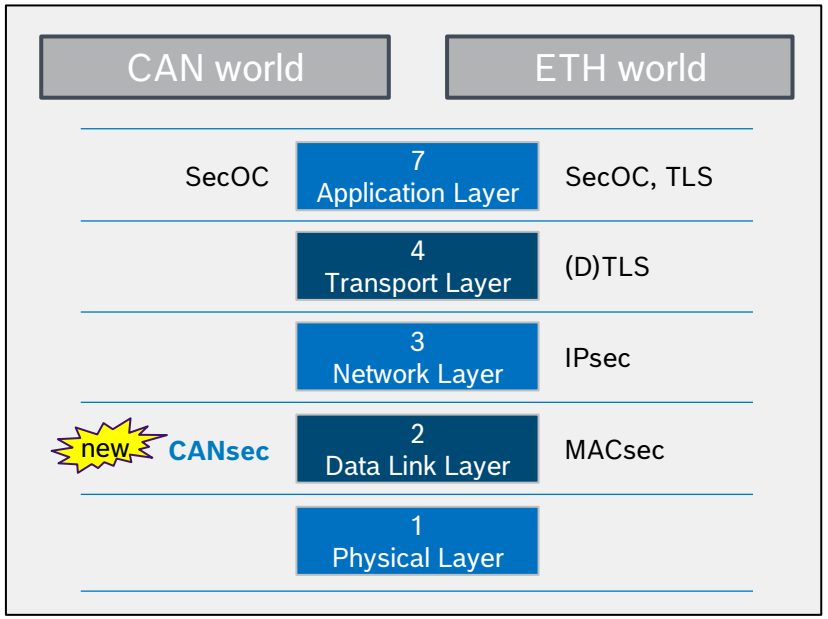
## How to protect communication **in CAN**

**Why not go with SecOC?**

1. No complete specification
2. No standardized key agreement
3. No confidentiality supported with SecOC
4. No pure HW implementation of SecOC – never!

➔ SecOC cannot protect all traffic!

Communication and security protocols




**CAN we develop a new protocol?**

- This is extremely expensive!
- Ballpark estimate: **>25Mio$** for TLS1.3 or MACsec

**Decision**: Specify **CANsec** – and try to reuse as much as possible from MACsec


Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

# CANsec for CAN XL
## Task Force Security @ CiA

### Original plan @ CiA SIG CAN XL

Target: Mimic MACsec, optimize for CAN




### New Target: Reuse MACsec as is!

Supporters of approach:

VECTOR, RENESAS, NXP, ST life.augmented, CARIAD

2020 | 2021 | 2022 | 2023 | 2024 | 2025 | 2026

release

### Where are we?

- CANsec is specified in CiA 613-2, released 05.2026
- AUTOSAR concept technically stable [link1, link2]


Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.



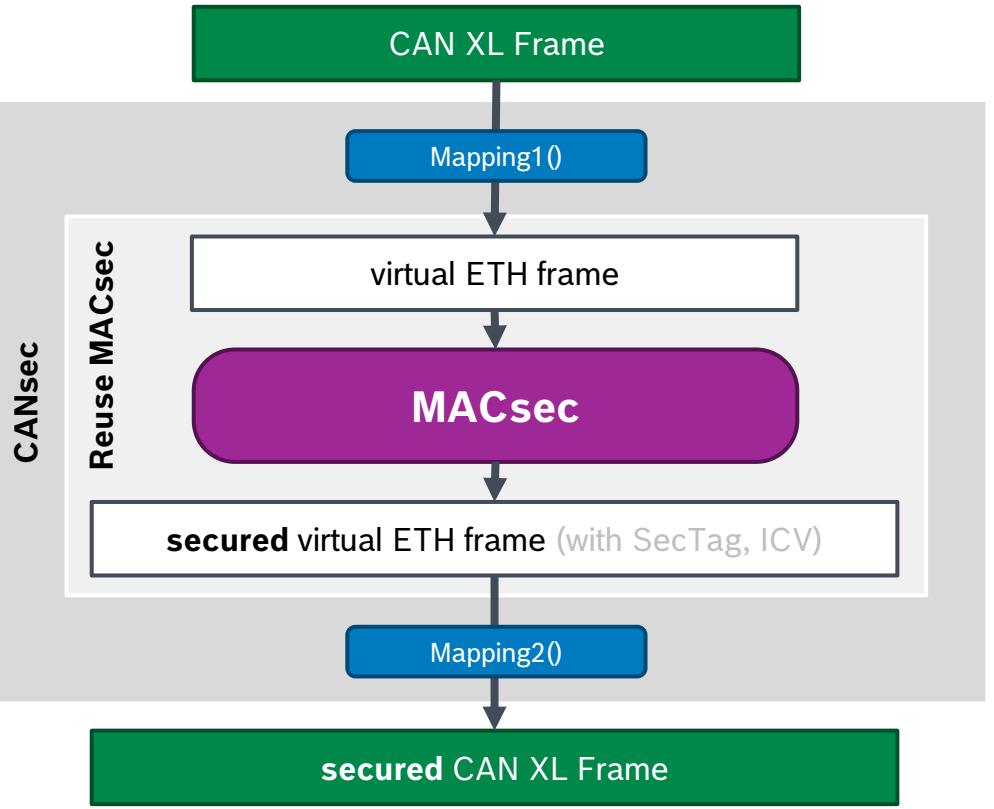

AI generated image

# CANsec for CAN XL
## Overview

### Strategy

**Reuse MACsec**

**Reuse MKA**




MKA: MACsec Key Agreement


Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

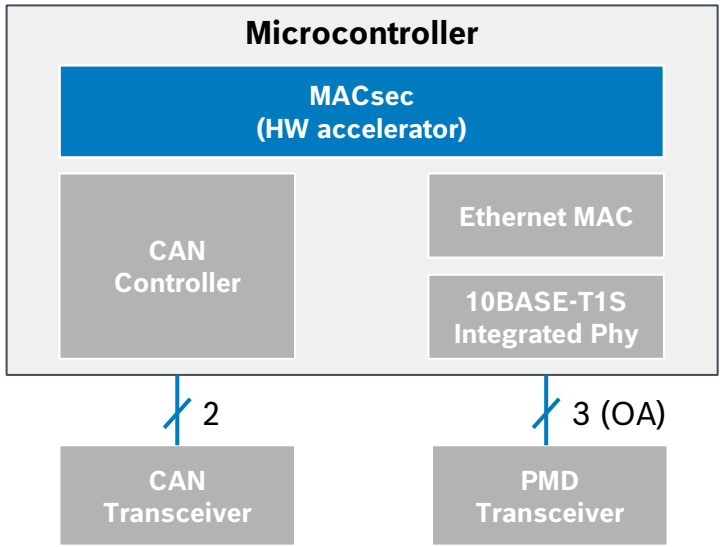
# Motivation
## Reuse MACsec – Advantages

- **No discussion needed**
  whether ETH or CAN security is better ➔ it is the same!

- **Mature**
  because MACsec is mature

- **Short time to market**
  MACsec implementations need to be only slightly extended

- **Share implementation between CAN and ETH,**
  CAN & ETH can use the same hardware accelerator on MCU





Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

OA: Open Alliance

BOSCH

# Reuse MACsec for CANsec

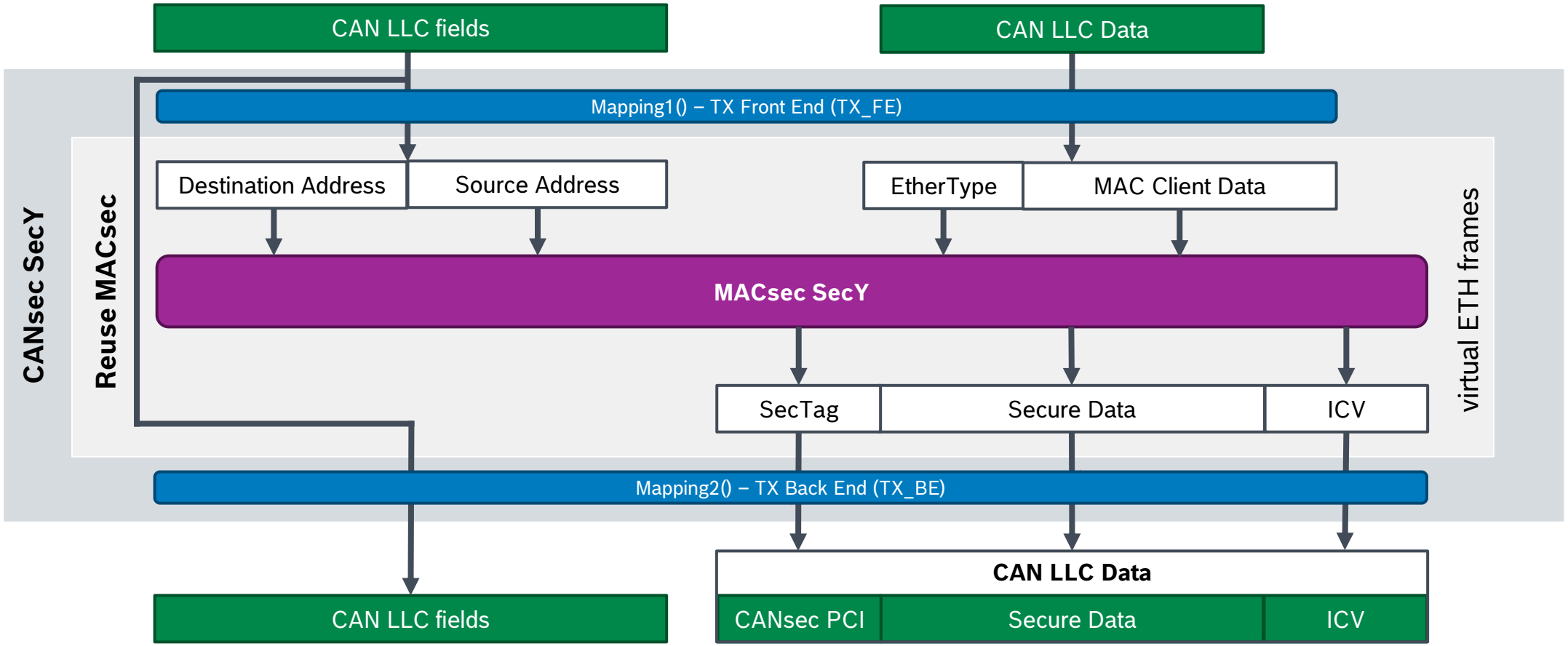
## Deep Dive



Public C-SC0

BOSCH

# Wrapping MACsec for CANsec
## High-Level View





Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

LLC: Logical Link Control, L2 sublayer

BOSCH

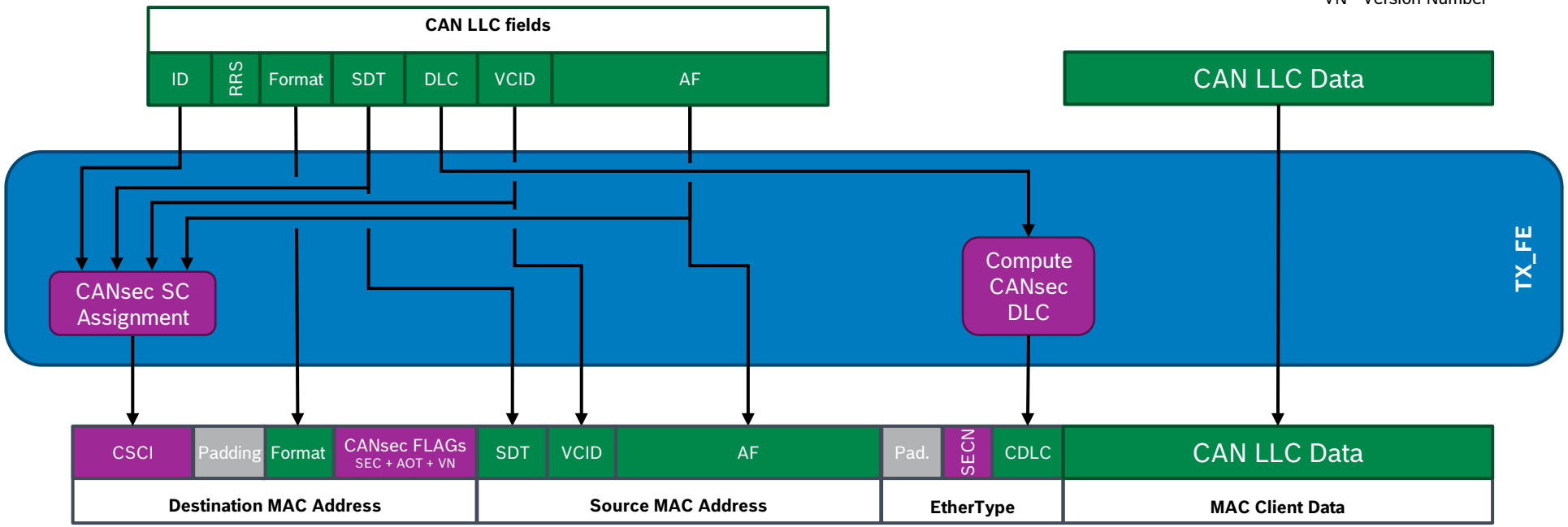
# CANsec
## Mappings

EASY!?


Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

# Wrapping MACsec for CANsec
## Mapping1: TX Frontend (TX_FE)

| | |
|---|---|
| ID | Priority Identifier |
| RRS | Remote Request Substitution |
| SDT | Service Data Type (--> EtherType) |
| DLC | Data Length Code |
| VCID | Virtual CAN Channel Identifier (--> VLAN ID) |
| AF | Acceptance Field |
| CSCI | CANsec Secure Channel Identifier |
| AOT | Add-On Type |
| SEC | Simple Extended Content |
| VN | Version Number |




**Mapping Strategy**

- **ETH.DA:** constant - MACsec SC selection bases on ETH.DA
- **CSCI:** CANsec SCI; look-up CSCI value and map it to ETH.DA
- **ETH.SA:** non-static


Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

# CANsec & MACsec
## Comparison

| | | CANsec (CAN XL) | MACsec (10BASE-T1S) |
|---|---|---|---|
| identical | **Security properties** | Authenticity, Integrity, Confidentiality, Freshness | |
| | **Cipher Suites** | AES-GCM (in the future: ASCON) | |
| | **Key lengths** | 128 bit<br>256 bit | |
| | **Freshness value** | 32 bit<br>64 bit | |
| optimized | **Overhead** | 26 Byte | 24 Byte (End Station)<br>32 Byte (e.g. Zone Controller) |
| | **Transmit SCs (per node)** | Many 🙃<br>up to $2^{16}$ SCs | IEEE: Up to 8, based on TCT<br>OA AutoMACsec: 1 |
| to be adapted | **Key management** | Plan: Adjusted AutoMKA | IEEE: MKA<br>OA: AutoMKA under develop. |

TCT Traffic Class Table
SC Secure Channel


Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

# CAN FD

## Can it use CANsec too?



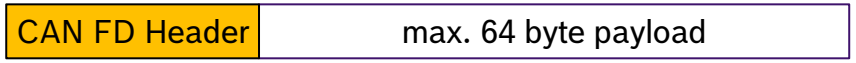
Public C-SC0

BOSCH

# CAN FD
## Overview

**CAN FD Protocol Limitations Today**

- Payload length max. 64 byte




- Indication, if CANsec was applied to the frame (yes/no) (= no EtherType)

**Strategy**

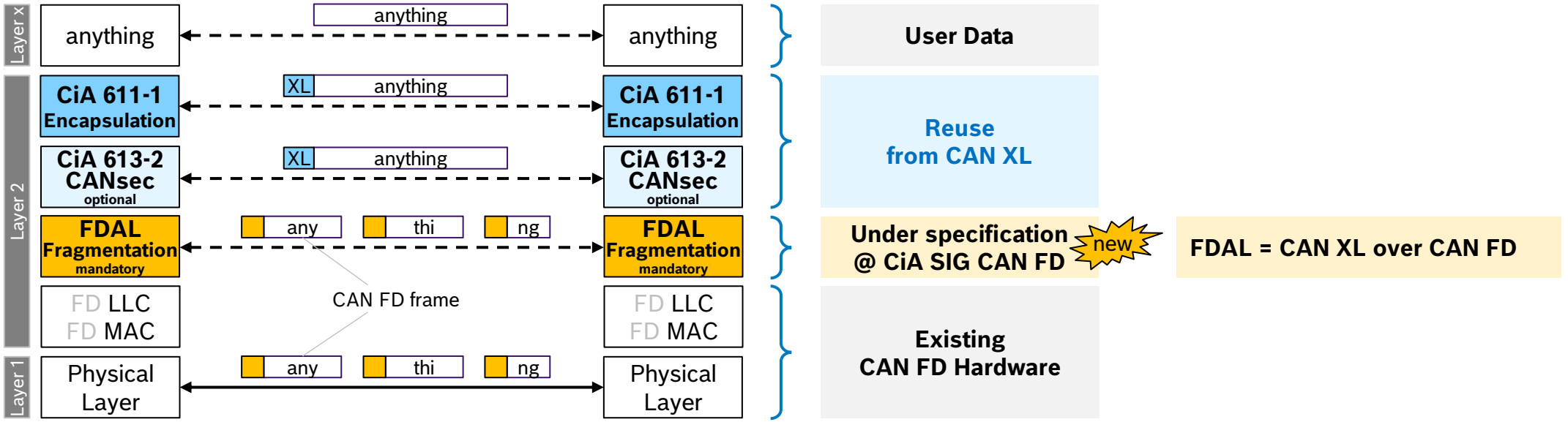
**A) Overcome payload limitations with FDAL**

**B) Reuse CAN XL features (e.g. CANsec)**


Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

FDAL: CAN FD Adaptation Layer

# FDAL – FD Adaptation Layer
## Protocol Stack

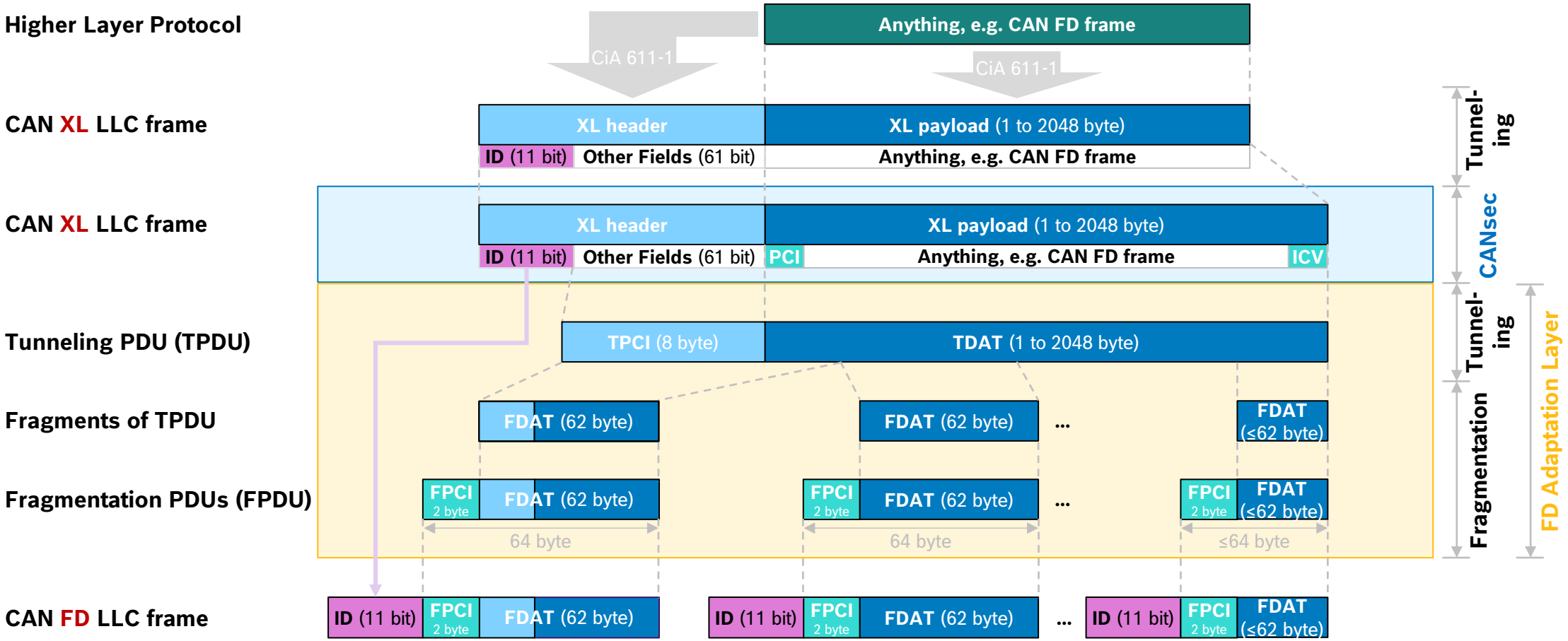



**Basic Idea: Tunnel CAN XL LLC frames via CAN FD LLC frames**


Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

# FDAL – CAN FD Adaptation Layer

## How it works with CANsec




TPDU: Tunneling PDU
TPCI: Tunneling PCI (= Header)
TDAT: Tunneling Data

FPDU: Fragmentation PDU
FPCI: Fragmentation PCI (= Header)
FDAT: Fragmentation Data
FPAD: Fragmentation Padding


Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applica

# Summary

## CANsec = MACsec + Wrapper

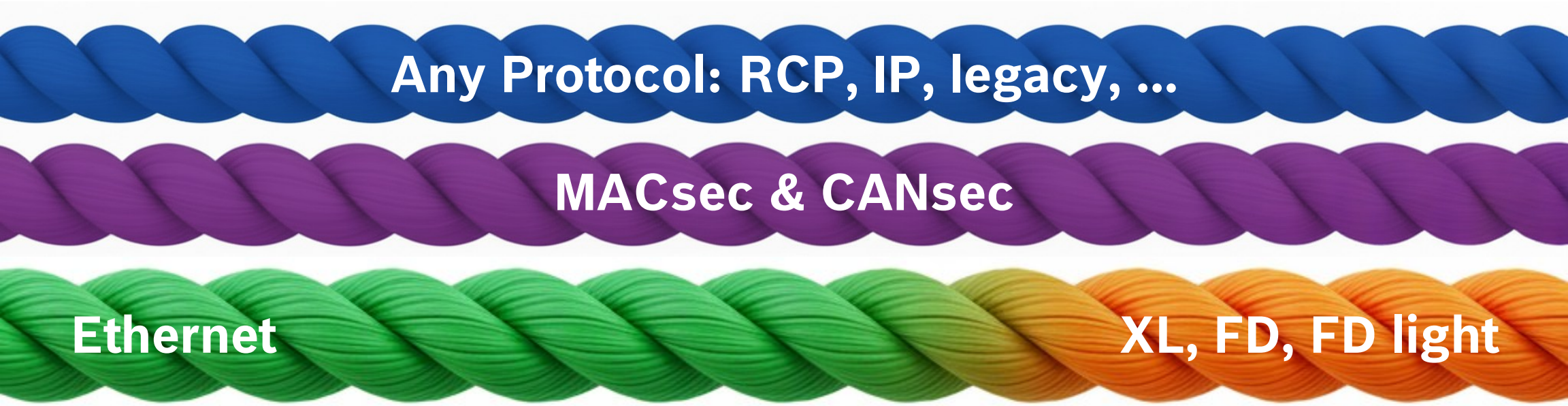


Public C-SC0

BOSCH

# CANsec
## Summary

**Ethernet & CAN master the needs of E/E Architectures**




**Gbit/s** **Mbit/s**


Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

# Wrap up
## Questions?



All this was enabled by unifying the Security Layer.


Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.



# … and Scapy

## Implementations & Demo



Public C-SC0

BOSCH

# CAN XL in Scapy
## What's available for Linux?



- Support in Linux since 6.1 by Oliver Hartkopp
- CAN XL support added to SocketCAN
- Since 6.9 tested with real Hardware, but CAN XL config still done at compile-time
- Since 6.19 device configuration available

- CAN XL support in SocketCAN builds on top of CAN FD structures
- This differs slightly from the "representation" of CAN XL "LLC frames" in ISO 11898-1.*

**Table 4 — LLC frame**

| Field | Size | Description |
|---|---|---|
| ID | (11 +18) bit | Priority identifier: ID(28) to ID(18) are used for all frames with 11-bit identifier format (CBFF, FBFF, XLFF) while ID(28) to ID(0) are used for all frames with 29-bit identifier format (CEFF, FEFF). |
| Format | 3 bit | The frame format is coded by the three bits IDE, FDF, and XLF:<br>— if $000_b$ the MAC frame format is CBFF;<br>— if $100_b$ the MAC frame format is CEFF;<br>— if $010_b$ the MAC frame format is FBFF;<br>— if $110_b$ the MAC frame format is FEFF;<br>— if $011_b$ the MAC frame format is XLFF.<br>Other bit values do not code a valid frame format. |
| FTYP | 1 bit | In CBFF and CEFF, the FTYP bit corresponds to the MAC frame's RTR bit, this means that FTYP distinguishes between the frame types RF (FTYP = $1_b$) and DF (FTYP = $0_b$).<br>In FBFF and FEFF, the FTYP bit is ignored for transmission but corresponds to the MAC frame's RRS bit for reception.<br>In XLFF, the FTYP bit corresponds to the MAC frame's RRS bit. |
| BRS | 1 bit | Bit rate switch: only used in FBFF and FEFF frames. |
| ESI | 1 bit | Error state indicator: only used in FBFF and FEFF frames. |
| SDT | 1 byte | SDU type: describes the structure of the frame's data field content, only used in XLFF frames. |
| SEC | 1 bit | Simple/extended content: flags if the data field contains simple or extended content. This is only used in XLFF frames. |
| DLC | (7+4) bit | Data length code: 7 MSB only used in XLFF frames. For XLFF frames the numbers 0 to 2047 code 1 to 2048 data bytes. For DFs of all other formats the 4 LSB encode the number or data bytes as specified in Table 5. |
| VCID | 1 byte | Virtual CAN channel ID: only used in XLFF frames. |
| AF | 4 byte | Acceptance field: only used in XLFF frames. |
| LLC data | 0 to 2048 byte | Payload of the LLC frame |

* The problem: there is no "wire"-representation of an CAN XL LLC frame standardized

Two constraints for Scapy impl.: built on top of used SocketCAN interface & provide "LLC-style" interface for CANsec.


© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

BOSCH

# CAN XL in Scapy

```python
class CANXL(CAN):
    """CAN XL frame - wire-format compatible with Linux struct canxl_frame.

    Uses the Linux kernel data representation (``struct canxl_frame``) for
    field names and layout.  ISO 11898-1:2024 field accessors are available
    via ``@property`` methods (``dlc``, ``xlf``, ``sec``, ``ftype``,
    ``frame_format``), and ``show(style="11898-1")`` renders using ISO
    terminology.

    Example::

        >>> from scapy.layers.can import CANXL
        >>> pkt = CANXL(priority=0x42, vcid=0x10, sdt=3, af=0xDEAD) / b'\\x01\\x02
        >>> pkt.show()
        >>> pkt.show(style="11898-1")
    """
    name = "CAN XL"
    fields_desc = [
        # prio word (4 bytes, LE on socket, swapped to BE by pre_dissect)
        BitField('reserved2', 0, 8),        # bits 31-24
        XBitField('vcid', 0, 8),            # bits 23-16
        BitField('reserved1', 0, 5),        # bits 15-11
        XBitField('priority', 0, 11),       # bits 10-0
        # flags byte (1 byte, no swap needed)
        # ISO 11898-1:2024: CAN XL requires XLF=1, FDF=1, IDE=0
        FlagsField('flags', CANXL_XLF | CANXL_FDF, 8,
                   ['sec', 'rrs', 'res_f2', 'res_f3',
                    'res_f4', 'ide', 'fdf', 'xlf']),
        # sdt (1 byte, no swap needed)
        XByteField('sdt', 0),
        # length (2 bytes, LE on socket, swapped to BE by pre_dissect)
        # Auto-computed from payload in post_build.
        # ISO 11898-1:2024 defines this as an 11-bit field (range 1-2048),
        # but Linux struct canxl_frame uses a full 16-bit field.
        # For kernel compatibility we use ShortField; post_build warns
        # if the computed length falls outside the valid range.
        ShortField('length', 0),
```

```python
    def _show_iso(self, dump=False, indent=3, lvl="", label_lvl=""):
        # type: (bool, int, str, str) -> Optional[str]
        """Render using ISO 11898-1:2024 field names."""
        if dump:
            from scapy.themes import ColorTheme, AnsiColorTheme
            ct = AnsiColorTheme()
        else:
            ct = conf.color_theme

        fmt_val = self.frame_format
        fmt_names = []
        if fmt_val & 0x04:
            fmt_names.append("XLF")
        if fmt_val & 0x02:
            fmt_names.append("FDF")
        if fmt_val & 0x01:
            fmt_names.append("IDE")
        fmt_str = "+".join(fmt_names) if fmt_names else "0"

        s = "%s%s %s %s\n" % (
            label_lvl,
            ct.punct("###["),
            ct.layer_name("CAN XL (ISO 11898-1)"),
            ct.punct("]###"))

        # Field order follows ISO 11898-1:2024 Table 4
        fields = [
            ("Priority", "0x%x" % self.priority),
            ("Format", "%s (0x%x)" % (fmt_str, fmt_val)),
            ("FTYPE", "%d" % int(self.ftype)),
            ("SDT", "0x%x" % self.sdt),
```

```python
    # -- ISO 11898-1:2024 property accessors ---------------------------------

    @property
    def dlc(self):
        # type: () -> int
        """ISO 11898-1 Data Length Code (length - 1, range 0..2047)."""
        return max(0, self.length - 1) if self.length else 0
```



Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

# CAN XL in Scapy

Finally…

## [Draft] Feature: CAN XL Support #5152

Open RB-FriedrichWiemer wants to merge 6 commits into secdev:master from RB-FriedrichWiemer:feature/canxl-support

https://github.com/secdev/scapy/pull/5152

Conversation 0 · Commits 6 · Checks 0 · Files changed 6

**RB-FriedrichWiemer** commented 2 minutes ago

### Description

This is my first pull request to scapy.
It adds CAN XL support to the existing CAN layer, making use of the CAN XL implementa

**RB-FriedrichWiemer** added 6 commits 12 minutes ago

- Add CAN XL frame support
- Add CAN XL unit tests
- Add CAN XL support to NativeCANSocket
- test: add CAN XL TestSocket regression tests
- doc: add CAN XL layer documentation
- Add copyright notes

```
>>> from scapy.layers.can import CANXL
>>> frame = CANXL(priority=0x7ff, sdt=0xCA, vcid=0xFE, af=0xDEADBEEF) / raw(2)
>>> frame.show()
###[ CAN XL ]###
  reserved2 = 0
  vcid      = 0xfe
  reserved1 = 0
  priority  = 0x7ff
  flags     = fdf+xlf
  sdt       = 0xca
  length    = 0
  af        = 0xdeadbeef
###[ Raw ]###
     load      = b'\x00\x00'

>>> frame.show2()
###[ CAN XL ]###
  reserved2 = 0
  vcid      = 0xfe
  reserved1 = 0
  priority  = 0x7ff
  flags     = fdf+xlf
  sdt       = 0xca
  length    = 2
  af        = 0xdeadbeef
###[ Raw ]###
     load      = b'\x00\x00'

>>> frame._show_iso()
###[ CAN XL (ISO 11898-1) ]###
  Priority  = 0x7ff
  Format    = XLF+FDF (0x6)
  FTYPE     = 0
  SDT       = 0xca
  SEC       = 0
  DLC       = 0
  VCID      = 0xfe
  AF        = 0xdeadbeef
  Data      = b'\x00\x00'
```



Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

BOSCH

# CANsec in Scapy
## … in 3 slides

```python
class CANsec(Packet):
    name = "CANsec"
    fields_desc = [
        XBitField("aot", 0b001, 3),
        BitField("secn", 0, 1),
        BitField("vn", 0, 1),
        BitField("CE", 0, 1),
        BitField("AN", 0, 2),
        BitField("reserved", 0, 8),
        XIntField("PN", 0),
        XBitField("csci", 0, 16),
        StrFixedLenField("icv", None, length=16),
    ]
```

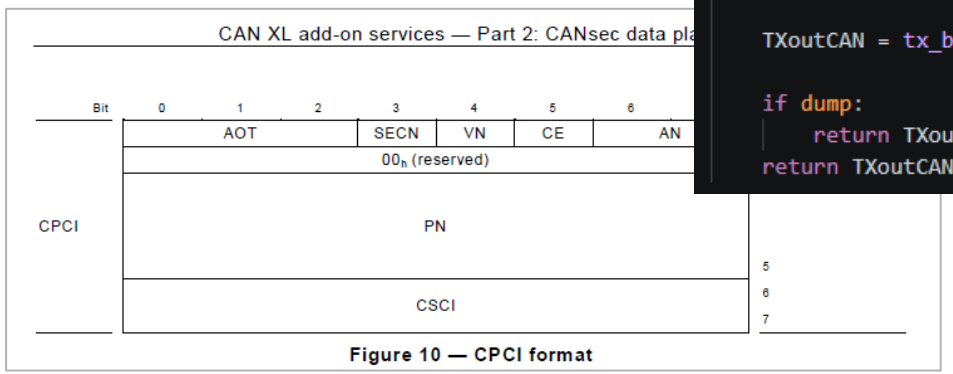


Figure 10 — CPCI format

```python
def cansec_protect(TXinCAN, csciMapping, an, pn, key, encrypt, dump=False):
    if not isinstance(TXinCAN, CANXL):
        raise Scapy_Exception("input frame must be a CAN XL packet")

    csci = mapToSCI(TXinCAN, csciMapping)

    TXinETH = tx_fe(TXinCAN, csci)

    sa = MACsecSA(
        sci=csci2sci(csci),
        an=an, pn=pn, key=key,
        icvlen=16, encrypt=encrypt, send_sci=1)
    TXoutETH = sa.encap(TXinETH)
    TXoutETH = sa.encrypt(TXoutETH)

    TXoutCAN = tx_be(TXoutETH, TXinCAN, csci)

    if dump:
        return TXoutCAN, TXinETH, TXoutETH
    return TXoutCAN
```

```python
def cansec_verify(RXinCAN, key):
    if not isinstance(RXinCAN, CANXL):
        raise Scapy_Exception("input frame must be a CAN XL packet")
    if not RXinCAN.haslayer(CANsec):
        raise Scapy_Exception("no CANsec layer found")

    RXinETH = rx_fe(RXinCAN)

    sa = MACsecSA(
        sci=reconstructSCI(RXinCAN, prefix=0),
        an=RXinCAN[CANsec].AN,
        pn=RXinCAN[CANsec].PN,
        key=key,
        icvlen=16,
        encrypt=RXinCAN[CANsec].CE,
        send_sci=1)
    # sa.decrypt raises InvalidTag if the ICV verification fails;
    # we let that propagate to the caller unchanged.
    RXoutETH = sa.decap(sa.decrypt(RXinETH))

    RXoutCAN = rx_be(RXoutETH, RXinCAN)
    return RXoutCAN
```


Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

# CANsec in Scapy
## TX Front End

```python
def tx_fe(TXinCAN, csci):
    # type: (CANXL, int) -> Ether
    """TX front-end: map plain CAN XL frame to virtual Ethernet frame.

    dst = CSCI | IDE | FDF | XLF | SEC | <padding> | AOT | VN
    src = SDT | VCID | AF
    type = 0x8000 | SECN(1bit) | (DLC + 26)
    data = plaintext payload
    """
    if not isinstance(TXinCAN, CANXL):
        raise Scapy_Exception("input frame must be a CANXL packet")

    plain_data = bytes(TXinCAN.payload)

    if TXinCAN.sec:
        raise Scapy_Exception("sec bit already set on input frame")

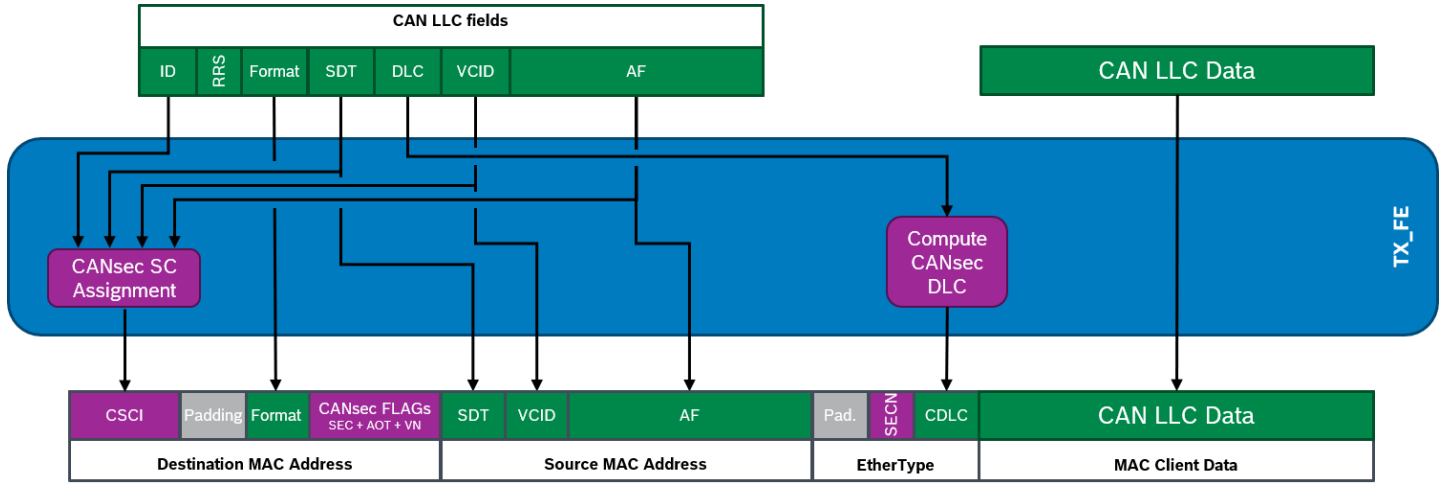
    dlc = len(plain_data) - 1
    if dlc > (2048 - 26):
        raise Scapy_Exception(
            "payload too long, no room for adding CANsec overhead")

    # Build temp frame with SEC set for ethDst mapping
    sec_flags = int(TXinCAN.flags) | CANXL_SEC
    CPCI = CANsec(aot=0b001, secn=int(TXinCAN.sec), csci=csci, vn=0)
    TXinCANsec = CANXL(priority=TXinCAN.priority, flags=sec_flags,
                       sdt=TXinCAN.sdt, vcid=TXinCAN.vcid,
                       af=TXinCAN.af) / CPCI / Raw(plain_data)

    dst = ethDstFromCanXlFrame(TXinCANsec, csci)
    dst = dst.to_bytes(6, "big")
    src = ethSrcFromCanXlFrame(TXinCANsec)

    secn = int(TXinCAN.sec)
    type_val = 0x8000 | (secn << 8) | (dlc + 24 + 2)

    return Ether(src=src, dst=dst, type=type_val) / Raw(plain_data)
```




```python
def ethDstFromCanXlFrame(canxl_frame, csci=None):
    # type: (CANXL, Optional[int]) -> int
    """Map CAN XL frame fields to a 48-bit Ethernet destination address.

    DA = CSCI(16) | padding(24) | IDE | FDF | XLF | SEC | AOT(3) | VN
    """
    cansec_flags = 0b0010  # default: AOT=001, VN=0
    if canxl_frame.haslayer(CANsec):
        cansec_flags = (canxl_frame[CANsec].aot << 1) | canxl_frame[CANsec].vn
        csci = canxl_frame[CANsec].csci

    if csci is None:
        raise Scapy_Exception("no CSCI given")

    # Bit layout (MSB first, 48 bits total):
    #   [47:32] CSCI   [31:8] padding   [7] IDE   [6
    #   [4] SEC   [3:1] AOT   [0] VN
    dst = (csci << 32) | \
        (int(canxl_frame.ide) << 7) | \
        (int(canxl_frame.fdf) << 6) | \
        (int(canxl_frame.xlf) << 5) | \
        (int(canxl_frame.sec) << 4) | \
        cansec_flags
    return dst
```

```python
def ethSrcFromCanXlFrame(canxl_frame):
    # type: (CANXL,) -> bytes
    """Map CAN XL frame fields to a 6-byte Ethernet s

    SA = SDT(1) || VCID(1) || AF(4)
    """
    src = canxl_frame.sdt.to_bytes(1, "big")
    src += canxl_frame.vcid.to_bytes(1, "big")
    src += canxl_frame.af.to_bytes(4, "big")
    return src
```


Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, repro

# CANsec in Scapy
## The other mappings

```python
def tx_be(TXoutETH, TXinCAN, csci):
    # type: (Ether, CANXL, int) -> CANXL
    """TX back-end: map MACsec Ethernet output back to CANsec CAN XL frame."""
    if not TXoutETH.haslayer(MACsec):
        raise Scapy_Exception("no MACsec layer found")

    m = TXoutETH[MACsec]
    raw_pay = bytes(m.payload)
    secure_data = raw_pay[:-16]
    icv = raw_pay[-16:]

    CPDU = CANsec(aot=0b001, secn=int(TXinCAN.sec), csci=csci, vn=0,
                  icv=icv, CE=m.E, AN=m.AN, PN=m.PN)

    sec_flags = int(TXinCAN.flags) | CANXL_SEC
    TXoutCAN = CANXL(priority=TXinCAN.priority, flags=sec_flags,
                     sdt=TXinCAN.sdt, vcid=TXinCAN.vcid,
                     af=TXinCAN.af)

    return TXoutCAN / CPDU / Raw(secure_data)
```

```python
def rx_fe(RXinCAN):
    # type: (CANXL,) -> Ether
    """RX front-end: map CANsec CAN XL frame to virtual MACsec Ethernet."""
    if not isinstance(RXinCAN, CANXL) or not RXinCAN.haslayer(CANsec):
        raise Scapy_Exception("no CANXL or CANsec layer found")

    dst = ethDstFromCanXlFrame(RXinCAN, csci=RXinCAN[CANsec].csci)
    dst = dst.to_bytes(6, "big")
    src = ethSrcFromCanXlFrame(RXinCAN)

    pay = bytes(RXinCAN[CANsec].payload)
    type_val = int.from_bytes(pay[:2], "big")

    # MACsec SL = len(secure_data) which is len(pay)
    # (Derived from: total = CPCI(8) + len(pay) + ICV(16), SL = total - 24)
    # This works on both dissected and freshly-constructed packets.
    sl = len(pay)

    macsec_pci = MACsec(
        Ver=0, ES=0, SC=1, SCB=0,
        E=RXinCAN[CANsec].CE, C=RXinCAN[CANsec].CE,
        AN=RXinCAN[CANsec].AN,
        SL=sl if sl < 46 else 0,
        PN=RXinCAN[CANsec].PN,
        SCI=reconstructSCI(RXinCAN, prefix=0),
        type=type_val)

    return Ether(src=src, dst=dst, type=0x88e5) / \
        macsec_pci / Raw(pay[2:] + RXinCAN[CANsec].icv)
```

```python
def rx_be(RXoutETH, RXinCAN):
    base_flags = (int(RXinCAN.flags) & ~CANXL_SEC) | \
        (int(RXinCAN[CANsec].secn) * CANXL_SEC)

    RXoutCAN = CANXL(
        priority=RXinCAN.priority,
        flags=base_flags,
        sdt=RXinCAN.sdt,
        vcid=RXinCAN.vcid,
        af=RXinCAN.af)

    return RXoutCAN / Raw(bytes(RXoutETH.payload))
```


Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

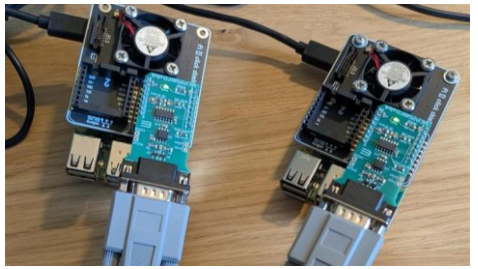
# Putting everything together

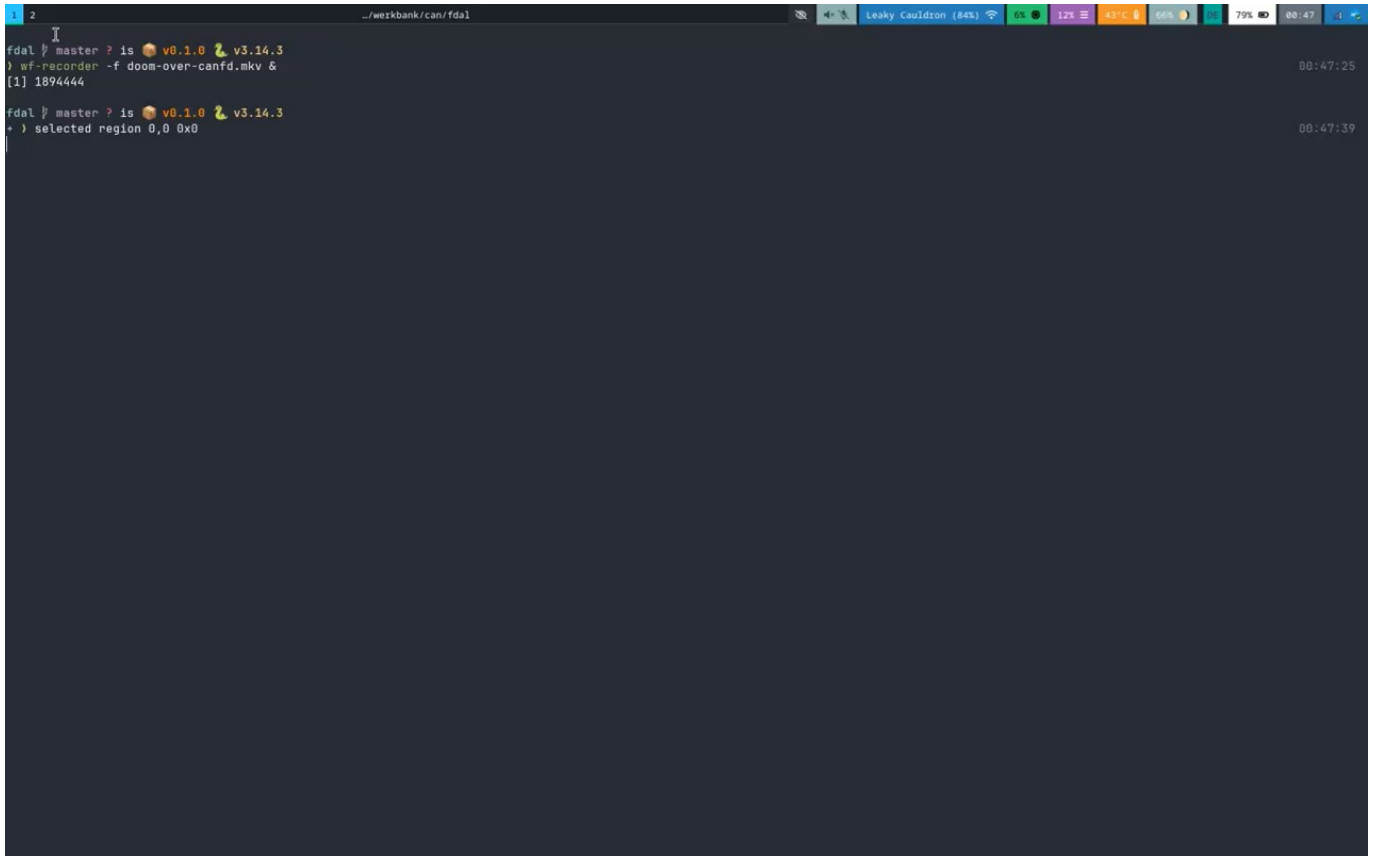


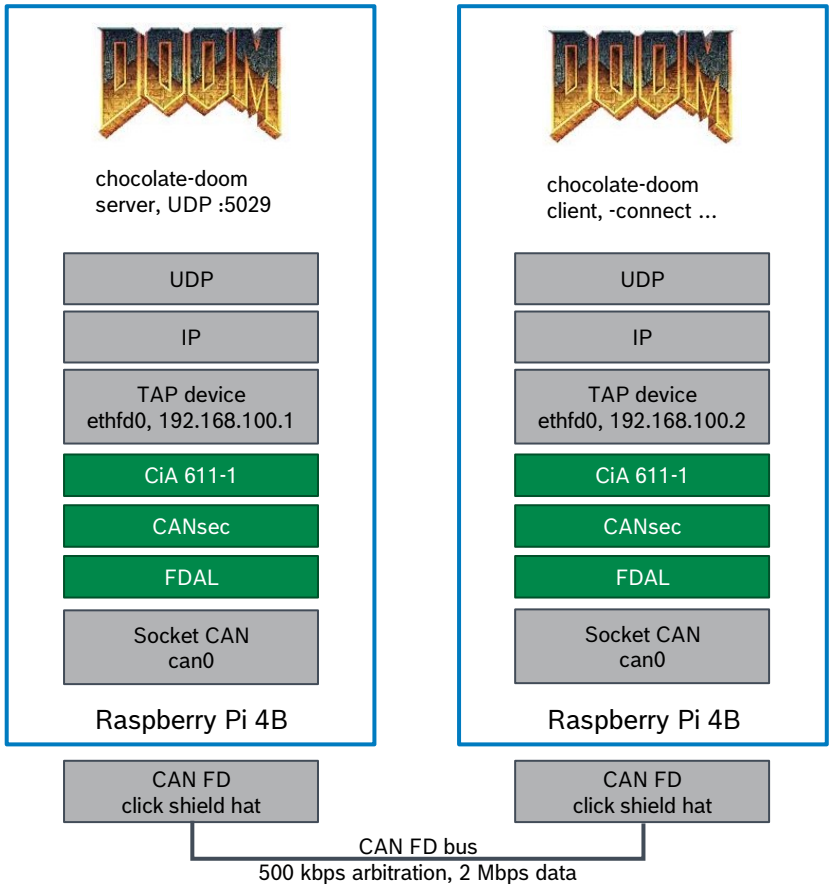
Public C-SC0

BOSCH

# One Layer, Every Wire
## ... but does it run Doom?







Public C-SC0 | Robert Bosch GmbH | 2026-09-17
© Robert Bosch GmbH 2026. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.