Secure Automotive Ethernet: Implementing and Benchmarking MACsec, IPsec, and TLS

Lukas Eugster*, Friedrich Wiemer†, Mert D. Pesé ‡, Mohammad Hamad*, Sebastian Steinhorst*
*Technical University of Munich, Germany
†Robert Bosch GmbH, Germany
‡Clemson University, USA

***Abstract*—Automotive Ethernet is rapidly replacing legacy in-vehicle buses, offering low-cost gigabit throughput but lacking native security. Standards such as MACsec, IPsec, and TLS 1.3 can address this gap, yet their performance on resource-constrained ECUs remains underexplored, particularly under strict sub-second startup constraints. This paper develops and releases an open-source testbed on an automotive-grade microcontroller running Zephyr RTOS, implementing MACsec with MKA, IPsec with IKEv2, and TLS 1.3. We systematically benchmark handshake latency, throughput, and memory footprint. Results show that MACsec and IPsec achieve handshakes within 110 ms, while TLS 1.3 is about four times slower. Encryption reduces throughput, though low-rate workloads remain supported. Memory fits within a single 128 kB bank, with the network stack and IPsec dominating usage. Overall, the study highlights handshake latency as the main bottleneck and provides a reproducible open-source testbed for systematic evaluation of secure Automotive Ethernet on constrained ECUs, supporting future research on optimization and hardware acceleration.**

***Index Terms*—Automotive Ethernet, Embedded Security, MACsec, IPsec, TLS, ECU Benchmarking, In-Vehicle Networks**

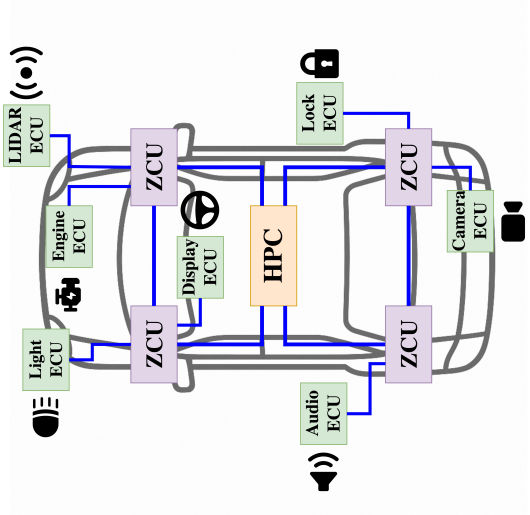

Fig. 1: Zone-based automotive network where ZCUs aggregate ECU traffic to a central HPC over Automotive Ethernet.

I. Introduction

The automotive industry is changing fast. Modern vehicles feature numerous advanced components and sensors, including cameras and LiDAR, which generate large amounts of real-time data. These changes create a strong need for reliable, high-speed, and secure in-vehicle communication. Legacy buses, such as Controller Area Network (CAN) and Local Interconnect Network (LIN), cannot meet these bandwidth demands. As a result, Automotive Ethernet is replacing them and provides gigabit-class throughput at low cost. This shift is further strengthened by the move to zonal architectures. In such designs (see Fig. 1), a Zone Control Unit (ZCU) aggregates data from multiple sensors and forwards it over a high-speed backbone to a central high-performance computer (HPC). Ethernet connects the different ZCUs, which drives the rapid adoption of Automotive Ethernet.

Automotive Ethernet itself does not provide built-in security. Therefore, protection must be added using standards such as Media Access Control Security (MACsec) [1], Internet Protocol Security (IPsec) [2], and Transport Layer Security (TLS) [3]. These solutions must still meet stringent automotive requirements, including high throughput, very low latency, regulatory compliance, and rapid startup times. The behavior of these security protocols on resource-constrained ECUs is not well understood. Prior work often presents isolated case studies or evaluates protocols individually, without consistent on-device measurements or explicit consideration of the key agreement phase. As a result, a clear and comparative view of protocol trade-offs is still missing.

This work addresses these challenges by building an open-source testbed on an automotive-grade microcontroller that runs Zephyr RTOS. On this platform, we implement MACsec with MACsec Key Agreement (MKA), IPsec with Internet Key Exchange version 2 (IKEv2) [4], and TLS 1.3. The setup models a zonal backbone with switch-based forwarding and traffic aggregation. The main contributions of this work are as follows:

- An open-source, automotive-grade testbed for secure Automotive Ethernet on a resource-constrained ECU;
- A consistent on-device implementation of MACsec, IPsec, and TLS 1.3 on the same platform;
- A systematic evaluation of handshake latency, throughput, and memory footprint under identical conditions.

The design is modular, cost-effective, and close to a modern in-vehicle network. We release all implementations, configurations, and measurement tools as open source to support reuse and replication[1].

[1]https://github.com/tum-esi/SecAutoEth

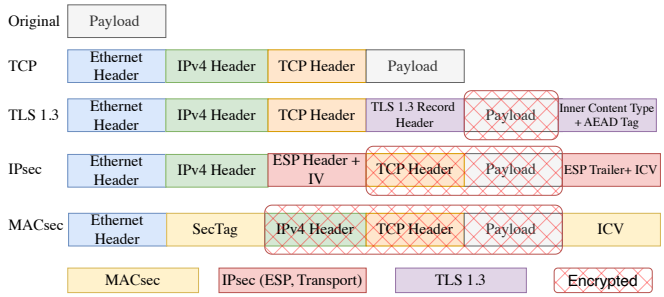

Fig. 2: On-wire layout for one payload over Ethernet. Rows show TCP, TLS 1.3, IPsec ESP transport, and MACsec, with red marks indicating encrypted fields for each protocol.

II. Background and Related Work

A. Automotive Ethernet

Ethernet, standardized as IEEE 802.3, has evolved from its 10 Mbps origins in the 1980s to speeds of 100 Gbps and beyond. For automotive use, a key milestone was IEEE 802.3bw (100BASE-T1), standardized in 2015, which enables 100 Mbps full-duplex communication over a single unshielded twisted pair of copper. This lightweight cabling reduces cost and vehicle weight while tolerating harsh automotive environments. Unlike traditional Ethernet, each link operates in a master–slave configuration to ensure synchronization. Higher-speed variants, such as 1 Gbps (802.3bp) and 10 Gbps (802.3ch), further extend the technology, positioning Automotive Ethernet as a scalable backbone for in-vehicle communication.

B. Security Protocols

As shown in Fig. 2, several security protocols can protect the same packet, and each adds its own headers and trailers. All of them protect the payload. In the following, we provide a brief description of these protocols.

MACsec secures Ethernet frames on the local link and provides confidentiality, integrity, and data-origin authentication on a frame-by-frame basis. It runs between MACsec entities (SecYs) on each port and relies on IEEE 802.1X MKA for key management [5]. A group of authenticated peers forms a Connectivity Association (CA). Each SecY sends data over one transmit Secure Channel (SC) and receives data over peer SCs. Every SC holds up to four Secure Associations (SAs) with individual Secure Association Keys (SAKs), identified by a Secure Channel Identifier (SCI) and a 2-bit Association Number (AN). A protected frame is a MACsec Protocol Data Unit (MPDU) with three parts: a Security TAG (SecTAG), the secured data, and an Integrity Check Value (ICV). The SecTAG carries the MACsec EtherType, control information, the AN, a packet number, and the SCI. The ICV verifies integrity, and the packet number supports anti-replay checks within an SA. MACsec addresses common link-layer threats such as eavesdropping, spoofing, and on-path modification, but it does not protect against brute-force DoS attacks.

MKA provides key management for MACsec on a local Ethernet link, allowing peers to protect frames after link establishment [5]. Peers that share a Connectivity-Association Key (CAK) form a CA and run MKA between their SecY entities. MKA confirms CAK ownership, elects a key server based on priority, and distributes fresh SAKs to all active peers using AES Key Wrap [6]. From the CAK, it derives two keys: the Integrity Check Key (ICK) to sign MKA messages and the Key Encrypting Key (KEK) to wrap SAKs. MKA messages (MKPDUs) carry a random station identifier and a strictly increasing message number. Echo and number checks verify liveness and prevent replay. The key server also announces the cipher suite so each SecY can install matching SAs for its Secure Channel. For automotive use, configuration options are often reduced to speed up startup, and links are typically secured by default.

IPsec is a suite that protects IP traffic at the network layer and supports two headers: the Authentication Header (AH) for integrity and data-origin authentication without encryption, and the Encapsulating Security Payload (ESP) for confidentiality, integrity, and anti-replay protection [2]. IPsec operates in two modes: transport mode, which protects the upper-layer payload while leaving the original IP header intact, and tunnel mode, which encapsulates the entire inner IP packet inside a new outer IP header. In this work, we focus on ESP in transport mode. Each protected packet carries an ESP header with a Security Parameters Index (SPI) and a sequence number, a nonce (IV), the encrypted payload, and an ICV. A small trailer adds padding to a 4-byte boundary, the pad length, and the next-header field.

IKEv2 is the control protocol for IPsec [4]. It authenticates peers and establishes Security Associations (SAs) between nodes. An SA records the negotiated protocol (ESP or AH), the mode (transport or tunnel), the algorithms, the keys, the SPI, lifetimes, and other parameters. SAs are stored in the Security Association Database (SAD), and each SA is linked to a Security Policy Database (SPD) entry when traffic requires IPsec protection. IKEv2 runs over UDP and protects its messages after the initial exchange.

TLS 1.3 secures end-to-end sessions above TCP [3]. It provides confidentiality, integrity, and peer authentication. The handshake negotiates algorithms, authenticates peers using certificates or pre-shared keys, and derives session keys. After the handshake, the record layer protects application data using authenticated encryption. TLS is point-to-point and does not support multicast communication. Datagram TLS (DTLS) is the UDP-based variant, but is out of scope in this work.

C. Discussion of Related Work

Analyzing the Automotive Ethernet security protocols, such as MACsec, IPsec, and TLS, has been presented in numerous previous works from various perspectives. Some studies focus on conceptual comparisons and design aspects [7], while others focus on industrial validation [8]. Most of the existing work has focused on individual protocols, such as IPsec [9], [10]. More recent work considers combinations of protocols, such as IPsec and MACsec, but evaluates them sequentially or in isolation rather than in parallel [11], [12]. As summarized in Table I, these studies differ in terms of automotive hardware platform, inclusion of key agreement, and protocol coverage. While several works use real or ECU-like hardware and address individual security mechanisms, none evaluate multiple

TABLE I: Comparison with related work

Work	Real (like-)ECU	Key Agreement	Protocol Set Covered	Simultaneous Protocols Measurement
Lauser et al. [7]	○	○	● (Conceptual)	○
Zelle et al. [13]	●	◑ (TLS handshake)	◑ (TLS only)	○
Hu et al. [9]	●	○ (Symmetric keys)	○	○
Hamad & Prevelakis [10]	◑	○ (Symmetric keys)	◑ (IPsec only)	○
achelos & dSPACE [8]	●	◑ (TLS handshake)	◑ (TLS only)	○
Kristoffersson [11]	●	◑ (IKEv2 & MKA)	◑ (IPsec and MACsec)	○
Puppala [12]	●	○	●	◑
This work	●	●	●	●

●: Yes; ◑: Partially; ○: No

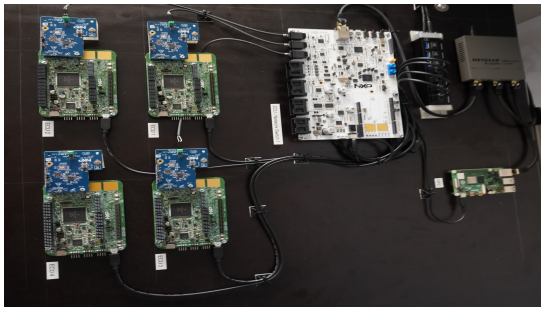

Fig. 3: Zonal-style testbed with S32K148 ECU, S32K344 ZCU, and Raspberry Pi 4 HPC over 100BASE-T1/GbE; used to benchmark MACsec/MKA, IPsec/IKEv2, and TLS 1.3

security protocols running simultaneously on the same ECU with explicit measurement of their key agreement phases. As a result, direct comparison of protocol behavior under concurrent operation and shared resource constraints remains limited. This work complements existing studies by providing a unified, on-device testbed for evaluating MACsec, IPsec, and TLS 1.3 under identical conditions.

III. Proposed Testbed

The proposed testbed, shown in Fig. 3, mirrors a zonal automotive layout while simplifying the backbone; we describe its hardware and software implementation below.

A. Hardware Implementation

Electronic Control Unit (ECU) We approximate a production ECU using NXP's S32K148-Q176 board, built around a 32-bit Arm Cortex-M4F (80 MHz, 2 MB Flash, 256 kB RAM). On-chip peripherals include CAN/LIN, UART/SPI/I^2C, I^2S, ADC/DAC, and a 10/100 Mb/s Ethernet MAC. The Ethernet interface is provided via the SABRE connector, where an ADTJA1101-RMII adapter hosts the TJA1101A/B 100BASE-T1 PHY. The adapter connects through RMII, supports master/slave selection by jumper, and exposes a screw terminal for single-pair Automotive Ethernet cabling.

HPC Modern zonal designs aggregate traffic and security services on a central HPC. While an NXP GoldBox 3-class device offers integrated 100BASE-T1 PHYs and ample resources, cost and availability motivated the use of a developer-friendly substitute. We use a Raspberry Pi 4 with on-board GbE and USB 2.0/3.0, which serves as a practical proxy without requiring hardware modification.

ZCU To bridge the Raspberry Pi's 1000BASE-T to 100BASE-T1 for ECUs, we employ NXP's S32K344-WB board as a ZCU. It features a dual lock-step Cortex-M7 MCU, RF and analog I/O, an audio codec, CAN/LIN, and an on-board SJA1105 Ethernet switch with five 10/100/1000 Mb/s ports. One port connects to the MCU. The switch integrates three 100BASE-T1 ports via TJA1101/TJA1102 PHYs and one 100BASE-TX port via a DP83848C PHY with RJ-45, enabling straightforward backbone-to-edge bridging.

B. Software Implementation

HPC The HPC runs Raspberry Pi OS and uses standard Linux tools. MACsec was enabled by recompiling the Linux kernel and configuring it with the Technica Engineering MKA daemon [14]. IPsec/IKEv2 was configured using StrongSwan [15], and TLS 1.3 used OpenSSL with self-signed certificates. Only minor adaptations were required, such as changing the default EAPoL group address to bypass switch filtering.

ZCU The ZCU operates as a transparent Ethernet switch based on the NXP SJA1105. Configuration is compiled into firmware and flashed via J-Link. Because the default configuration filtered EAPoL frames, the MKA destination address was modified to allow successful handshakes.

ECU The ECU software is the core contribution of this work. Unlike the HPC and ZCU, which rely on existing Linux tools and firmware, the ECU required a new, lightweight implementation of MACsec/MKA, IPsec/IKEv2, and TLS 1.3 on a resource-constrained MCU. All components were written in C and are released as open source. The software stack is based on Zephyr RTOS [16] and Cyclone Crypto/IPsec/TLS. A custom MACsec shim was implemented inside the NXP Ethernet driver to transparently encrypt and decrypt frames. The Technica Engineering MKA daemon was ported to Zephyr with adaptations for raw socket handling, timing, and logging. Cyclone IPsec was integrated through shim layers that hook into Zephyr's networking stack and translate packets between Zephyr and Cyclone data structures, enabling encryption, authentication, and secure reinjection. IKEv2 runs in a dedicated Zephyr thread and handles key exchange and security association setup. Cyclone SSL was integrated at the socket layer using lightweight wrappers around Zephyr's BSD sockets. Configuration was reduced to PSK mode with `AES-256-GCM`, `X25519`, and `ECDSA` to minimize memory usage while retaining interoperability with standard OpenSSL endpoints.

Together, these components form a modular and reproducible platform for benchmarking secure in-vehicle communication on constrained ECUs. The open-source release serves as a practical reference for future research and education in Automotive Ethernet security.

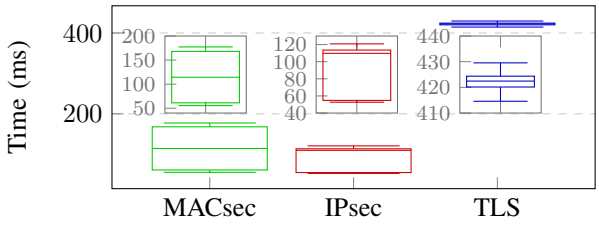

Fig. 4: Handshake times for the security protocols.

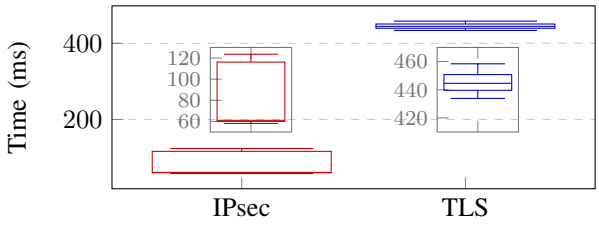

Fig. 5: Handshake times for IPsec and TLS over MAcsec.

IV. Evaluation

To evaluate the implementation, three key measurements were performed: handshake time, data throughput, and memory overhead.

A. Handshake Times

All handshake times were measured on the ECU. For each protocol, we ran 100 trials. Each trial was triggered by a GPIO toggle. We corrected for the toggle delay, which is a few microseconds and therefore negligible. We executed MKA as specified in the standard. We used IKEv2 with ECDH and a pre-shared key. We used TLS 1.3 with ECDH and a server certificate. As shown in Fig. 4, across 100 trials, MACsec completes in the low hundreds of milliseconds (median ≈120 ms, IQR ≈90–160 ms). IPsec shows similar but slightly tighter results (median ≈110 ms, IQR ≈80–120 ms). TLS 1.3 is significantly slower (median ≈425 ms, IQR ≈415–440 ms). All three protocols complete well under one second on the ECU, but TLS 1.3 is about $4\times$ slower than MACsec or IPsec. As a result, MACsec and IPsec better fit tight startup budgets. To simulate scenarios where IKEv2 or TLS runs after MKA, we repeated the handshake tests with MACsec active on the link. IKEv2 and TLS use the same message flow, but each packet now passes through the MACsec processing path. This adds per-frame overhead and reduces the effective MTU. Handshake times increase slightly, as shown in Fig. 5. All runs still complete well under one second.

B. Throughput Measurements

We evaluate data rate and throughput under different security configurations. We measure the maximum achievable throughput for each setup on the constrained ECU. We use ESP for all IPsec tests because it provides confidentiality and integrity and represents a stricter case than AH. All experiments use AES-256-GCM; for TLS 1.3 this corresponds to `TLS_AES_256_GCM_SHA384`. To contextualize the measured throughput values, realistic in-vehicle communication requirements were taken from automotive literature and industrial partners [7], [9], [13]. These requirements are summarized in Table II and serve as benchmarks for evaluating protocol suitability. For MACsec and IPsec, the ECU runs `zPerf` (iPerf-compatible) as a UDP client, and the HPC acts as the server. Each test runs for 10 s, is repeated 10 times, and the average is reported. For TLS 1.3, the HPC runs an OpenSSL server, and the received data is piped to `pipeviewer (pv)` to measure total bytes over time (10×10 s, averaged). Fig. 6 shows a monotonic decrease in throughput as additional security layers are applied. The throughput of MACsec and IPsec is similar. TLS achieves slightly lower throughput, and stacked configurations show the lowest values. Table III compares the measured throughput against the application requirements. Type 1 requirements are met in all scenarios, including the fully stacked case. Type 2 requirements are met in all cases except when all three protocols are stacked. Types 3 and 4 meet their requirements only when no encryption is used.

TABLE II: Representative data traffic in Automotive Ethernet networks

Data Type	Packet Size	Interval	Data Rate
Type 1: Audio / Misc.	128 bytes	10 ms	102 Kbps
Type 2: Audio / Misc. 2	256 bytes	10 ms	205 Kbps
Type 3: Various data	100 bytes	1 ms	≈ 1 Mbps
Type 4: ADAS object data	1400 bytes	< 1 ms	≈ 5 Mbps

TABLE III: Comparison of measured throughput against the requirements of the 4 types of traffic.

Scenario/Type	Type 1	Type 2	Type 3	Type 4
No Enc + SW Checksum	✓	✓	✓	✓
MACsec	✓	✓	✗	✗
IPsec	✓	✓	✗	✗
MACsec+IPsec	✓	✓	✗	✗
TLS	✓	✓	✗	✗
MACsec+TLS	✓	✓	✗	✗
IPsec+TLS	✓	✓	✗	✗
MACsec+IPsec+TLS	✓	✗	✗	✗

C. Memory Overhead

RAM usage was measured with all three protocols enabled, including their handshakes, using Zephyr's RAM Report tool. Only one of the two 128 kB RAM banks on the S32K148 was safe to use because DMA cannot cross the bank boundary. Even with this limitation, the full system fits into a single RAM bank. The results are shown in Fig. 7. The largest memory consumer is the modified Zephyr network stack (23.33%). IPsec with IKEv2 is the second largest consumer (18.72%), mainly due to IKE security association state. MKA accounts for 9.06%. The system heap and the Ethernet driver each consume about 13% (13.32% and 13.34%). Free memory accounts for 8.52%. Within the *Misc.* category (13.71%), TLS contributes 0.69% and MACsec 0.02%. The remaining memory is used by the main application, Zephyr kernel components, and the NXP HAL.

D. Discussion

The evaluation highlights that MACsec, IPsec, and TLS 1.3 can be deployed on resource-constrained ECUs, but at a clear cost in terms of latency and throughput. Handshake latency and software-based cryptographic processing are the

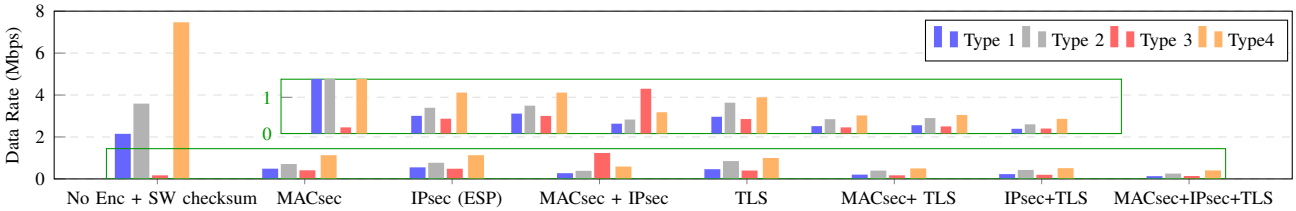

Fig. 6: Throughput for different traffic types under different measurement scenarios.

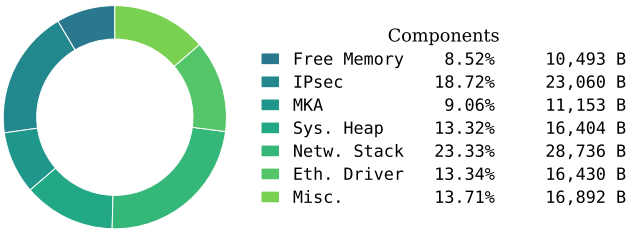

Fig. 7: Memory consumers of the implementation

primary performance bottlenecks, particularly for high-data-rate workloads and stacked security configurations. These results show that hardware-accelerated cryptography, either within the MCU or offloaded to the Ethernet PHY, is a promising direction to improve performance while preserving protocol functionality. At the same time, hardware offloading reduces flexibility, whereas software-based implementations remain more adaptable to changing requirements. It should be noted that the MACsec results presented in this work reflect a software-based implementation and do not capture the performance characteristics of MACsec when implemented in hardware, such as in Ethernet PHYs. In addition, none of the evaluated protocols provides inherent protection against denial-of-service attacks, which remains an important consideration for in-vehicle networks.

V. Conclusion

This work demonstrates that MACsec, IPsec, and TLS 1.3 can be deployed on a resource-constrained ECU using the Zephyr RTOS. We present an open-source automotive testbed and a systematic on-device evaluation of handshake latency, throughput, and memory usage. The results demonstrate that secure communication is feasible under automotive constraints; however, software-based cryptography imposes clear performance limitations, particularly for higher data rates and stacked security configurations. Overall, the study highlights the trade-offs between security, performance, and flexibility in Automotive Ethernet deployments. The released testbed enables reproducible evaluation and supports future work on hardware acceleration and alternative security designs. This work represents a proof-of-concept focused on latency and throughput characterization; security hardening and performance optimization were left for future work.

Acknowledgment

This work is supported by the European Union-funded project CyberSecDome (Agreement No.: 101120779). An AI tool (ChatGPT) was used to improve language clarity and readability.

References

[1] "Ieee standard for local and metropolitan area networks—media access control (mac) security," IEEE, 2018, revision of IEEE Std 802.1AE-2006; incorporates 802.1AEbn-2011, 802.1AEbw-2013, 802.1AEcg-2017. [Online]. Available: https://1.ieee802.org/security/802-1ae-2018/

[2] K. Seo and S. Kent, "Security Architecture for the Internet Protocol," RFC 4301, Dec. 2005. [Online]. Available: https://www.rfc-editor.org/info/rfc4301

[3] E. Rescorla, "The transport layer security (tls) protocol version 1.3," RFC 8446, Aug. 2018, online: https://www.rfc-editor.org/rfc/rfc8446.

[4] C. Kaufman, P. E. Hoffman, Y. Nir, P. Eronen, and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)," RFC 7296, Oct. 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7296

[5] IEEE, "Ieee standard for local and metropolitan area networks–port-based network access control," IEEE Std 802.1X-2020, Feb. 2020, defines the MACsec Key Agreement (MKA) protocol.

[6] R. Housley and J. Schaad, "Advanced Encryption Standard (AES) Key Wrap Algorithm," RFC 3394, Oct. 2002. [Online]. Available: https://www.rfc-editor.org/info/rfc3394

[7] T. Lauser, D. Zelle, D. Kern, C. Krauß, and L. Völker, "Security protocols for ethernet-based in-vehicle communication," in *2024 IEEE Vehicular Networking Conference (VNC)*, 2024, pp. 148–155.

[8] achelos GmbH and dSPACE GmbH, "Validating tls-protected ethernet communication," achelos GmbH and dSPACE GmbH, Whitepaper, 2022. [Online]. Available: https://www.achelos.de/fileadmin/user_upload/achelos/Downloads/Whitepaper/EN/WhitePaper_-_TLS-Security-Testing_achelos_dSpace_en.pdf

[9] S. Hu, Q. Zhang, A. Weimerskirch, and Z. M. Mao, "Gatekeeper: A gateway-based broadcast authentication protocol for the in-vehicle ethernet," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 494–507. [Online]. Available: https://doi.org/10.1145/3488932.3517396

[10] M. Hamad and V. Prevelakis, "Implementation and performance evaluation of embedded ipsec in microkernel os," in *2015 World Symposium on Computer Networks and Information Security (WSCNIS)*, 2015, pp. 1–7.

[11] J. B. Kristoffersson, "Zero trust in autonomous vehicle networks utilizing automotive ethernet," Master's thesis, Chalmers University of Technology and University of Gothenburg, Gothenburg, Sweden, 2022. [Online]. Available: https://odr.chalmers.se/items/f88ee313-cea2-4814-9294-162e634f9dab

[12] S. S. Puppala, "On secure data-in-transit in modern and future in-vehicular network systems: Featuring automotive ethernet in telematics gateway," Master's thesis, Blekinge Institute of Technology, Karlskrona, Sweden, Sep. 2024, mSc in Telecommunication Systems, Faculty of Computing, Blekinge Institute of Technology. [Online]. Available: http://www.diva-portal.org/smash/get/diva2:1916895/FULLTEXT01.pdf

[13] D. Zelle, C. Krauß, H. Strauß, and K. Schmidt, "On using tls to secure in-vehicle networks," ser. ARES '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3098954.3105824

[14] Technica Engineering GmbH, "Mkadaemon: Macsec key agreement daemon," https://github.com/Technica-Engineering/MKAdaemon, 2025, gitHub repository, commit 23c987f, accessed 2025-09-12.

[15] The strongSwan Project, "strongswan," https://strongswan.org/, [Online; accessed 14-Dec-2025].

[16] The Zephyr Project, "Zephyr project," https://www.zephyrproject.org/, 2025, [Online; accessed 14-Dec-2025].