

TEE-based On-demand Key Distribution for Hierarchical In-Vehicle Zonal Architecture

Wonseok Song, Sanghoon Jeon, and Jong-Chan Kim

Graduate School of Automobile and Mobility, Kookmin University, Seoul, Korea

{sws9912, sh.jeon, jongchank}@kookmin.ac.kr

Abstract—As vehicles migrate from CAN to Ethernet to support communication-heavy applications, factory-installed static cryptographic keys are exposed to a broader attack surface. Such static keys are no longer secure in the long term throughout a vehicle’s long lifecycle. Thus, we propose an on-demand key distribution system considering the hierarchical zonal E/E architecture. In our scheme, upon receiving a new master key from a server by a central vehicle computer, newly derived keys are distributed to zone controllers, and in turn to low-level ECUs in a hierarchical manner. We specifically focus on isolating each zone by securing the derivation and distribution of sub-master keys between the central computer and zone controllers by leveraging the hardware-level security of trusted execution environment (TEE). We implement a prototype on NVIDIA Jetson platforms, where the overheads of cryptographic operations and end-to-end delays are evaluated. Our experiments show that in a complex zonal architecture, the entire set of keys in a vehicle can be renewed in 480 ms, which is shorter than the engine starting time.

I. INTRODUCTION

Modern vehicles are equipped with hundreds of electronic control units (ECUs) that continuously exchange data over in-vehicle networks (IVNs). To secure these communications, encryption is essential for preventing eavesdropping, tampering, and injection attacks [1], [2]. Due to the resource constraints of ECUs, *symmetric* cryptography is generally preferred over *asymmetric* methods because of its low computational overhead and latency [3]. However, symmetric schemes introduce the fundamental challenge of secure *key management* across distributed nodes.

In current production vehicles, a single pre-shared key is commonly flashed into all the ECUs while manufacturing [4]. However, as the number and diversity of ECUs increase, the “*one-key-for-all*” model is too risky since one compromised ECU can endanger the entire IVN. Besides, the automotive industry is moving from the current flat architecture to the zonal architecture, where ECUs are hierarchically organized as in Fig. 1. As in the figure, a central computer is connected to zone controllers over high-speed Ethernet, and each zone controller is connected with low-level ECUs inside its zone by legacy networks such as CAN. To mitigate the risk of the current static key method targeting the future zonal architecture, we propose (i) hierarchical key management and (ii) trusted execution environment (TEE)-based on-demand key update.

Hierarchical key management. In our proposed system architecture, the central computer works as a *secure gateway* that receives a *master key* from the automaker’s central server. In some cases, the automaker can *push* a new master key to a

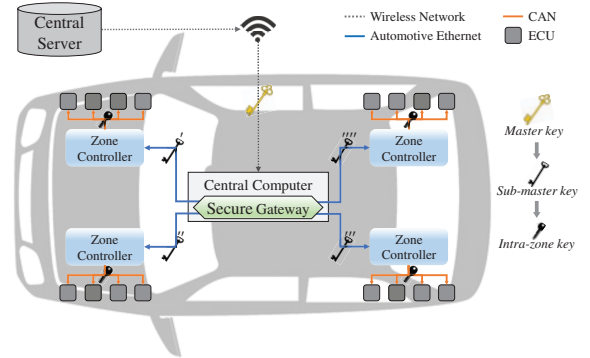


Fig. 1: Hierarchical key distribution with zonal architecture

vehicle. In other cases, the vehicle’s intrusion detection system (IDS) can *pull* a new master key from the server after detecting an attack. Then the secure gateway broadcasts a *key offering* message to zone controllers such that zone controllers can request new *sub-master keys* to the secure gateway. Upon each request, the secure gateway derives a unique sub-master key and sends it back to the requesting zone controller. Then each zone controller generates a unique *intra-zone key* and distributes it through the CAN bus. By this hierarchical key management, all the keys in a vehicle can be updated by renewing only a single master key, while each zone remains cryptographically isolated to prevent the propagation of attacks.

TEE-based on-demand key update. TEEs such as ARM TrustZone and Intel SGX are widely used for isolating security-related operations and data inside a hardware-provided secure enclave. To guarantee the security of our key management scheme, we employ Open Portable Trusted Execution Environment (OP-TEE) [5], which is based on ARM TrustZone. We assume that both the central computer and zone controllers are equipped with TEE. In our scheme, all the cryptographic operations and keys are securely executed and stored within the TEE, achieving hardware-level protection. Regarding low-level ECUs, we do not assume TEEs since many automotive MCUs feature lightweight hardware security modules (HSMs) instead. Also note that TEE comes with non-negligible switching overheads between the *normal world* and the *secure world*.

For the validation, we implement a zonal architecture testbed with NVIDIA Jetson platforms and an Ethernet switch, where we implement our on-demand key distribution protocol with OP-TEE. We measure the computation overhead and end-to-end delays with various system configurations. Our testbed shows that we can securely renew all the keys inside a vehicle

with a complex zonal architecture in 480 ms by our hierarchical on-demand key distribution scheme even with the world switching overhead of TEE.

The main contributions of this paper are as follows:

- We design a hierarchical in-vehicle key management system that enables dynamic key renewal and provides fine-grained security isolation between zones.
- We implement the proposed key distribution protocol with SOME/IP and OP-TEE, where we evaluate the end-to-end key renewal delay on NVIDIA Jetson platforms.

II. BACKGROUND AND PROBLEM DESCRIPTION

A. Zonal Architecture

The conventional automotive electrical and electronic (E/E) architecture has long employed the *flat ECU architecture*, where function-specific ECUs (e.g., engine controller and brake controller) are loosely connected via a CAN bus. However, as vehicles integrate more and more ECUs, this flat architecture is facing serious challenges such as wiring complexity, scalability issues, and lack of the centralized control authority. Thus, the industry is moving to the hierarchical *zonal architecture*. In the zonal architecture, ECUs are grouped into zones by physical locations (e.g., front, rear, left, and right zones), and each zone is managed by a *zone controller* that coordinates low-level ECUs within the zone. Meanwhile, the high-performance *central computer* performs cross-zone and computation-heavy tasks such as autonomous driving [6].

B. Scalable service-Oriented Middleware over IP (SOME/IP)

Scalable service-Oriented Middleware over IP (SOME/IP) is an Ethernet-based middleware protocol developed by BMW and standardized by AUTOSAR. SOME/IP provides various communication middleware features such as service discovery, remote procedure call, and event groups over TCP and UDP. Due to its flexibility and scalability, SOME/IP is widely considered the standard communication middleware for the zonal architecture [7]. However, SOME/IP does not provide built-in security features such as authentication, integrity protection, and encryption. Thus, to secure SOME/IP communications, extra cryptographic mechanisms must be integrated [8]. In this regard, this study focuses on the secure key distribution between the central computer and zone controllers assuming SOME/IP. For that, we use the open-source *vsomeip* implementation [9] to evaluate our key distribution scheme.

C. Open Portable Trusted Execution Environment (OP-TEE)

OP-TEE [5] is an open-source TEE implementation based on the ARM TrustZone technology. OP-TEE follows the GlobalPlatform TEE specifications [10] to enable the co-existence of two isolated execution environments: (i) *rich OS (normal world)* and (ii) *secure OS (secure world)*. Rich OS means Linux that is considered untrustable. However, OP-TEE’s secure OS is based on hardware security features, allowing security-guaranteed cryptographic operations and secure storage for sensitive data. All the security-related operations are implemented inside the secure world as *trusted applications* (TAs). Basically, TAs are developed by utilizing OP-TEE’s built-in

TABLE I: Considered threats and defense scenarios

Threat Scenario	Our defense mechanism
Replay attack	Nonce–timestamp validation
Message modification	ECDSA verification
Spoofing (unregistered nodes)	Whitelist check + ECDSA verification
Eavesdropping on reply	AES-GCM encryption
Ciphertext tampering	AES-GCM tag verification
Normal world compromise	TEE-confined operations
Zone controller/ECU compromise	Zone isolation + key rollover

cryptography library functions and secure storage primitives. Only authenticated *client applications* (CAs) in the normal world can invoke TAs to perform security operations inside the secure world.

D. Problem Description

Our problem is to design and implement a secure key distribution framework that supports the hierarchical structure of the SOME/IP-based zonal architecture by exploiting OP-TEE and ARM TrustZone’s OS- and HW-level security guarantees.

III. SECURITY ARCHITECTURE

A. Security Assumptions

Our framework depends on several trust assumptions that must hold to ensure the security objectives. First, the confidentiality and integrity of OP-TEE, including its secure storage, are guaranteed. This ensures that critical keying materials, such as master keys and sub-master keys, remain protected against unauthorized access. Second, the cryptographic algorithms used in our system (e.g., digital signature and symmetric-key algorithm) are securely implemented and correctly operated. Third, the initial whitelist registration is assumed to be performed by the automaker in a trusted environment free from tampering, ensuring that only legitimate nodes (i.e., central computer, zone controllers, and ECUs) are admitted into the system. The update or revocation of the whitelist is assumed to be achievable through over-the-air (OTA) updates or equivalent mechanisms, while the detailed process is beyond the scope of this study. Finally, we assume sufficient time synchronization across nodes to enable reliable validation of nonce–timestamp pairs, which is essential for defending against replay attacks.

B. Threat Model

Table I summarizes the considered threat scenarios. The attacker is assumed to have access to the in-vehicle Ethernet segment [11], [12], enabling them to perform eavesdropping [13], replay [14], modification [15], or spoofing [16] of key distribution messages. The attacker may compromise the normal world to generate and inject crafted malicious messages. Additionally, the attacker may compromise a specific zone controller or an ECU to extract the keys of the compromised zone. However, according to the security assumptions in Section III-A, physical compromise of TEE or direct extraction of keys from its secure storage is considered not possible. Availability-related threats such as denial-of-service (DoS) attacks to the secure gateway are also beyond the scope of this work and are assumed to be mitigated through complementary operational measures.

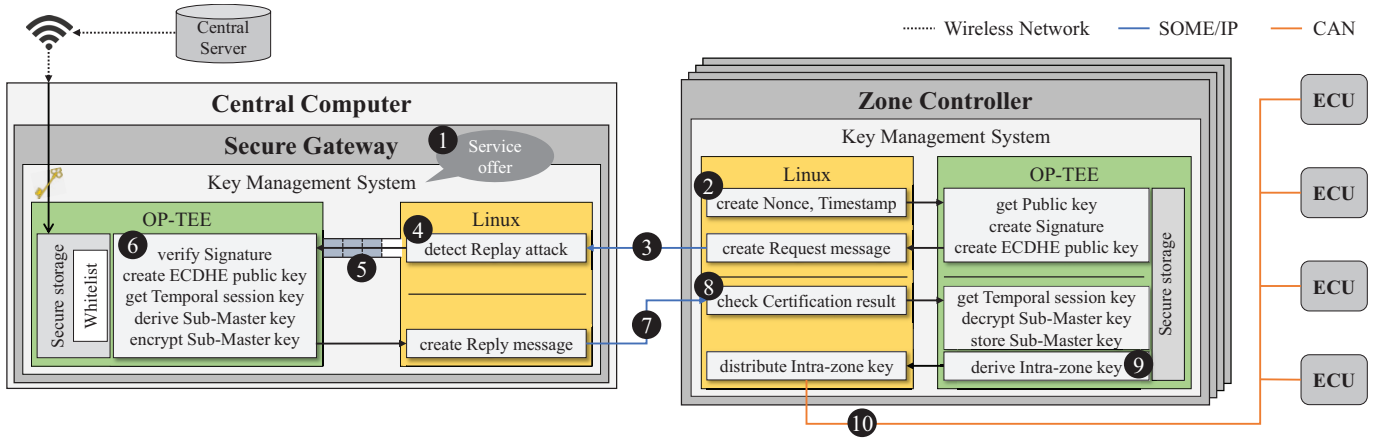


Fig. 2: Proposed system architecture and hierarchical key distribution protocol details

C. Defense Strategy Against Threats

Table I also summarizes our defense mechanisms for each threat, which are based on well-established cryptographic techniques such as Elliptic Curve Diffie–Hellman Ephemeral (ECDHE), Elliptic Curve Digital Signature Algorithm (ECDSA), and Advanced Encryption Standard-Galois/Counter Mode (AES-GCM). ECDHE is a lightweight handshake algorithm based on elliptic curves that first derives a shared secret, from which HMAC-based key derivation function (HKDF-SHA256) generates a *temporal session key* for encrypting the sub-master key itself to be derived with low computational overhead. However, since ECDHE alone does not provide a means to authenticate the peer’s identity, it is combined with ECDSA, which is a lightweight signature algorithm, for the entity authentication. Once the temporal session key is established, it is used by the AES-GCM algorithm to encrypt the sub-master key (derived from the master key). This approach leverages AES-GCM to ensure confidentiality through encryption and integrity through the generation of an authentication tag. In this way, both confidentiality and integrity are achieved in a single operation with less latency than deploying separate mechanisms.

As in Table I, *replay attacks* are mitigated by validating the nonce–timestamp pair at the secure gateway, ensuring that any attempt to resend old messages is immediately rejected; *message modifications* are prevented through ECDSA verification, which guarantees both message authenticity and integrity; *spoofing* attempts from unregistered nodes are blocked by the whitelist check and the ECDSA verification; for the *eavesdropping* of the reply message, sub-master keys are encrypted with AES-GCM. Any *ciphertext tampering* is instantly detected through AES-GCM’s tag verification. All the cryptographic operations and data including the derived keys in TEE are isolated from the normal world. Thus, even in the case of *normal world compromise*, attackers cannot access the keys. Also, by isolating each zone with separated intra-zone keys, the *compromise of a zone controller or an ECU* cannot endanger the security of other zones. The impact can be further minimized by our on-demand key rollover across all the nodes in the system.

IV. PROPOSED KEY MANAGEMENT SYSTEM

A. Our System Architecture

Fig. 2 illustrates our system architecture with a secure gateway (i.e., *key provider*) and multiple zone controllers (i.e., *key requesters*). The secure gateway receives and manages the *master key*, stores credentials, and maintains a whitelist of each zone controller’s node ID and public key. Upon receiving a key distribution request from a zone controller, the secure gateway derives a zone-specific *sub-master key* from the master key and securely delivers it through the key distribution protocol. Each zone controller then uses the received sub-master key to derive and distribute the intra-zone encryption keys (i.e., *intra-zone keys*) to the ECUs within its zone. All the cryptographic operations are executed entirely by OP-TEE of both the secure gateway and the zone controllers. With this hierarchical key distribution design, multiple zones can renew their keys in parallel, which significantly enhances the scalability of key distributions across the entire in-vehicle network.

B. Key Distribution Protocol

This section provides a detailed description of the key distribution protocol in Fig. 2, which governs the interaction between the secure gateway and one or more zone controllers.

1 Key distribution service offer. Upon obtaining the master key, the secure gateway broadcasts an offer of the key distribution service to all the zone controllers on the same Ethernet network. This offer is delivered using the service discovery mechanism of the SOME/IP protocol, which enables zone controllers to detect and subscribe to the key distribution service.

2 Key request preparation. A zone controller begins by generating a fresh nonce and timestamp to prevent replay attacks. It then invokes OP-TEE to generate a digital signature, using its ECDSA key pair (private and public keys) stored in secure storage, over the data fields consisting of the node ID, nonce, timestamp, and ECDSA public key. It also generates an ECDHE public key, which will be used to derive a temporal session key in the secure gateway.

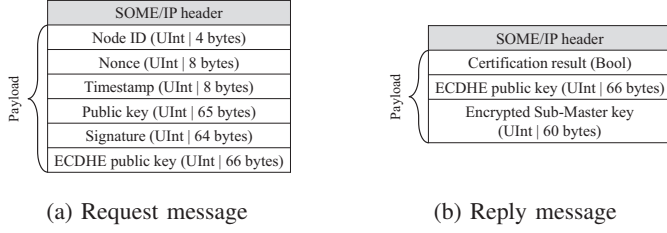


Fig. 3: Our SOME/IP message formats

3 Key request to secure gateway. The zone controller constructs a request message as in Fig. 3a and sends it to the secure gateway. The payload fields are:

- Node ID: a unique identifier of the zone controller;
- Nonce & timestamp: fresh values to prevent replay attacks;
- ECDSA public key: a public key for signature verification;
- Signature: a digital signature using ECDSA computed over (node ID || nonce || timestamp || public key);
- ECDHE public key: the zone controller’s ECDHE public key for the temporal session key derivation.

4 Verify the freshness of the message. The secure gateway first verifies the freshness of the message using the nonce and timestamp to detect replay attacks.

5 Wait for TEE’s availability. The CA in our secure gateway tries to invoke the TAs inside TEE for executing security operations. However, due to the limited number of available threads (e.g., 2 by default for OP-TEE) in TEE, there can be a *queueing delay* for the TEE’s availability if all the TEE threads are servicing other key requests.

6 Request verification and key generation. After the queueing delay, the secure gateway checks the node ID against the whitelist maintained inside TEE and validates the signature to confirm the requester’s integrity. Once validated, the gateway computes a temporal session key using the zone controller’s ECDHE public key and its own ECDHE private key. In parallel, it applies the HKDF-SHA256 to derive a dedicated sub-master key from the master key. Finally, the derived sub-master key is encrypted with AES-GCM under the temporal session key.

7 Key reply to zone controller. The secure gateway constructs a reply message in the format shown in Fig. 3b and sends it back to the zone controller. The payload fields are:

- Certification result: status code indicating authentication/signature verification success;
- ECDHE public key: the secure gateway’s ECDHE public key for temporal session key derivation;
- Encrypted sub-master key: sub-master key encrypted with AES-GCM using the temporal session key.

If the authentication fails, the certification result contains a failure code, and no key material (e.g., temporal session key or sub-master key) is neither generated nor transmitted.

8 Key reception and storage. After receiving the reply message, the zone controller first checks the certification result to ensure that the response is legitimate. It then extracts the secure gateway’s ECDHE public key from the payload and uses its own ECDHE private key to compute the temporal session key. With this temporal session key, the zone controller decrypts

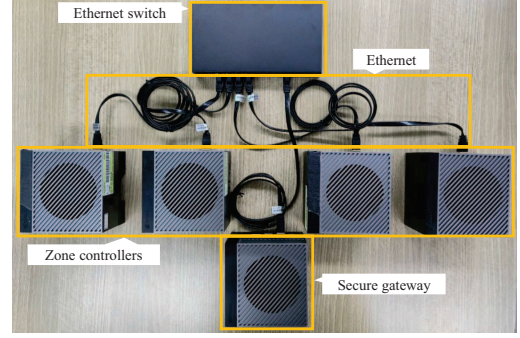


Fig. 4: Prototype implementation

the encrypted sub-master key using AES-GCM and stores it securely in OP-TEE secure storage.

9, 10 Intra-zone key derivation and distribution. Finally, the stored sub-master key is used to derive intra-zone encryption keys that protect communications among the low-level ECUs within the same zone, which is distributed through a CAN bus. For the CAN-based key exchanges, we can use already established methods (e.g. [17]).

V. EXPERIMENTS

A. Hardware and Software Setup

To validate the feasibility and effectiveness of the proposed key distribution architecture, we implement a prototype system using NVIDIA Jetson AGX Orin platforms with OP-TEE, as shown in Fig. 4. The system consists of one secure gateway and multiple (up to 8) zone controllers, which are connected through a 1-Gigabit Ethernet switch for high-speed communications.

Each Jetson AGX Orin features 12 ARMv8-A cores with ARM TrustZone support that enables secure executions through OP-TEE. Each node is configured with the following software specifications:

- OS: Ubuntu 20.04 LTS with Jetson Linux 35.4.1;
- TEE: OP-TEE v4.2.0;
- Cryptographic library: TEE Internal Core API / mbedTLS;
- Middleware: vsomeip 3.3.0 + CommonAPI 3.2.0.

B. Runtime Overhead of Security Primitives

To quantify the runtime overhead of security operations during the key distribution process, we measure each cryptographic operation’s computation time on both Linux (normal world) and OP-TEE (secure world). Fig. 5 compares the five operations required in the secure gateway and the other five operations required in the zone controller across Linux and OP-TEE. The results are the averages of 100 repeated experiments. As shown in the result, in some operations, OP-TEE’s overheads are significant. The secure gateway’s signature verification requires 23.21 ms while it only takes 10.38 ms in Linux. ECDHE public key creation takes 13.72 ms, which is increased from 1.03 ms. Deriving the temporal session key also takes longer (13.71 ms) than its counterpart (1.02 ms). In the zone controller, the overhead for creating the signature is the most significant (from 1.01 ms to 34.98 ms). One interesting observation is

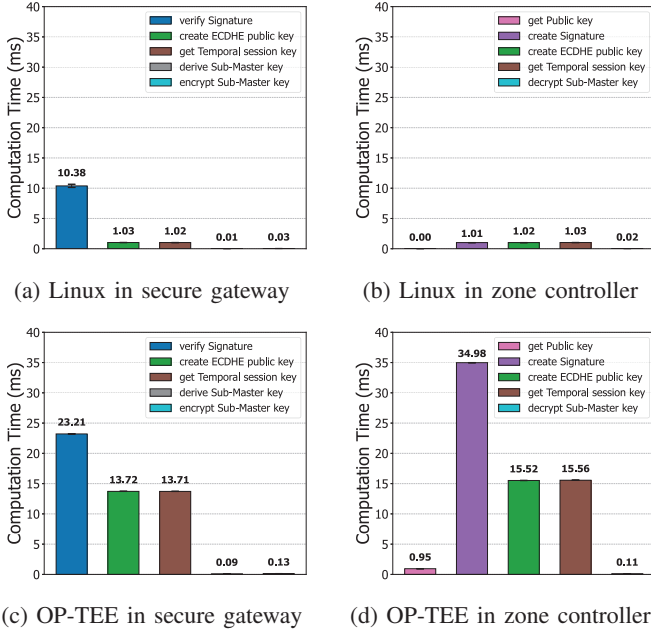


Fig. 5: Comparison of computing times for cryptographic operations in Linux (normal world) and OP-TEE (secure world)

that the signature generation (34.98 ms) in OP-TEE exhibits even higher latency than the verification (23.21 ms), which contrasts with the typical expectation in ECDSA where verification is usually more expensive. This result stems from the characteristics of the OP-TEE cryptographic library, which enforces secure random nonce generation and strict key isolation during signing, as mandated by the TEE Internal Core API policies. As a result, every ECDSA signing operation requires a hardware-backed random nonce generated through `TEE_GenerateRandom()`, ensuring stronger security at the cost of higher latency.

In the above experiment, Linux and OP-TEE use the same algorithms (ECDSA, ECDHE, and AES-GCM), but with different implementation libraries since the availability of cryptographic libraries are different for each OS. In Linux, we use the Crypto++ library, which is optimized for speed and flexibility in user space. In contrast, OP-TEE uses its Internal Core API together with the mbedTLS library, which emphasizes secure key handling, strong randomness, and isolation inside TEE. Moreover, every cryptographic call in OP-TEE requires additional context switching between the normal world and the secure world, further contributing to the higher latency. Nevertheless, the OP-TEE's runtime overheads still remain well within the acceptable latency budget (all under 40 ms). Also, since these operations occur only during the key establishment, not for usual data communications, the additional cost is negligible in practice. We claim that the overhead is not so costly considering its security assurance by the OP-TEE's OS- and hardware-level guarantees.

C. End-to-End Key Distribution Delay

Fig. 6 shows the end-to-end delays by our scheme including the ten delay components in Fig. 2 as increasing the number

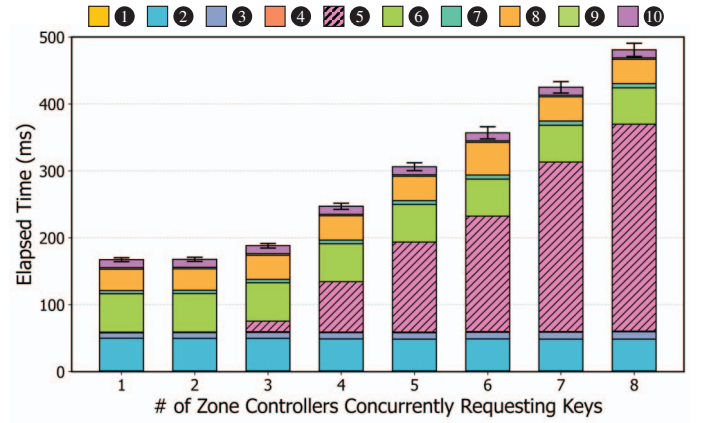


Fig. 6: End-to-end key distribution delays

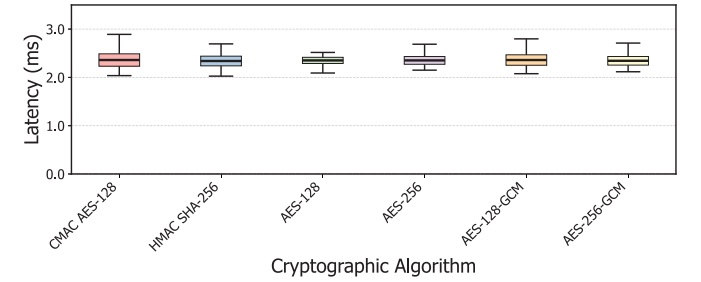


Fig. 7: Intra-zone key derivation latencies by zone controllers

of zone controllers. Each zone is assumed to have 20 low-level ECUs. For the measurement, zone controllers continuously request keys to the secure gateway to evaluate the worst-case scenario and the scalability of our key distribution protocol. Each experiment is executed 100 times, and the reported values are the averages. Key distribution service offer delay (1) is constant (< 1 ms) since it is a broadcast message. Key request preparation delay (2) is also constant inside each zone controller. Key request delay (3) slightly increases due to network contentions as more zone controllers are involved. The delay for verifying message freshness to detect replay attacks (4) also remains constant (< 1 ms). Queueing delay (5) represents the waiting time for TEE's availability. As in the figure, queueing delay significantly increases as the number of zone controllers increases. It is because all the requests are serialized by the limited number of TEE threads, which turns out to be the major bottleneck. After the queueing delay, request verification and key generation delay (6) is also constant in the secure gateway. Key reply delay (7) also slightly increases with the number of zone controllers. Key reception and storage delay (8) is also constant. Intra-zone key derivation delay (9) and per-zone key distribution delay (10) are also considered, which will be further explained in Section V-D. As a result, the worst-case delay is just 480 ms for 8 zones and 160 low-level ECUs, which is far shorter than the engine starting time.

D. Per-Zone Key Distribution Delay

After receiving a sub-master key from the secure gateway, each zone controller derives an intra-zone key for the message

TABLE II: Comparison of in-vehicle key management approaches

Approach	Communication protocol	In-vehicle architecture	Real-time guarantee	Hierarchical structure	Secure storage	Trusted Execution
LASAN [3]	CAN	Flat	Yes	No	No	No
ECQV-GDH [18]	CAN	Flat	Yes	No	No	No
Semi-centralized IBE/IBBE [17]	CAN	Domain	Partial	No	No	No
Fast Group Key Exchange [19]	CAN	Domain	Yes	No	No	No
Key Management for Ethernet [20]	Ethernet	Domain	Yes	No	No	No
Securing SOME/IP [8]	SOME/IP (Ethernet)	Domain	Yes	No	No	No
Ours	SOME/IP (Ethernet)	Zonal	Yes	Yes	Yes	Yes

encryption inside its zone. Fig. 7 shows the latency for deriving intra-zone keys using the HKDF-SHA256. The evaluation considers six algorithms representing different security levels, including Message Authentication Code (MAC) (AES-based and SHA-based), AES, and AES-GCM. The results indicate that key derivation can be completed within 2.3 ms regardless of the security level of the cryptographic algorithm, demonstrating extremely low latency.

Once derived, this intra-zone key can be rapidly distributed to low-level ECUs using legacy in-vehicle networks such as CAN, as done in many existing works [3], [17]–[19]. For example, [17] measures the CAN bus key distribution delay while varying the number of ECU groups and the number of ECUs. According to this study, assuming a zonal architecture with 20 low-level ECUs in each zone, the per-zone key distribution delay is 12 ms. Since the key distribution in each zone can be executed in parallel due to our hierarchical key distribution architecture, the delay does not need to be accumulated across zones. As a result, the worst-case end-to-end key distribution delay from the secure gateway to the low-level ECUs is about 480 ms including the intra-zone key derivation delay (2.3 ms) and the per-zone key distribution delay (12 ms).

This end-to-end latency is well within the practical real-time constraint. Typical engine-start sequences allow up to 1–2 seconds before critical subsystems must be operational. Furthermore, to minimize service disruption, key updates triggered by detected threats (e.g., replay or spoofing attacks) are generally expected to complete within a few hundred milliseconds.

VI. RELATED WORK

Hierarchical key management. Hierarchical key management has been widely explored in wireless sensor networks and fog computing to improve scalability, reduce storage overhead, and enhance resilience [21], [22]. Since such techniques remain largely unexplored in the automotive domain, we adopt the hierarchical key management scheme in accordance with the zonal architecture’s hierarchical structure targeting future E/E architectures.

CAN. Many previous studies focus on key distribution protocols for CAN buses. For example, LASAN [3] employs a centralized handshake to distribute group keys but lacks support for in-service re-keying. IBE/IBBE frameworks [17] employ pairing-based broadcasts for key distribution but partially incur high computational overhead and have only been evaluated within the context of CAN. Likewise, ECQV-GDH [18] and four-party ECDH methods [19] provide fast group key agreement but are not scalable to zonal architectures and operate in the non-secure domain, thus exposing long-term secrets. CAN-focused designs are based on flat or domain topologies and

do not incorporate the central gateway–zone controller–ECU hierarchy. When applied to zonal backbones, such approaches cause key distribution to either cross administrative boundaries or require per-ECU updates at the gateway, thereby degrading parallelism and increasing the key renewal latency.

Ethernet. [20] proposes a key management approach that improves the efficiency of key generation and distribution. However, its design shows limited suitability for service-oriented architectures, as it does not explicitly address integration with middleware such as SOME/IP. [8] introduces security extensions for SOME/IP with formal verification and practical evaluation. While effective for service-level protection, it remains unsuitable for hierarchical zonal architectures. Moreover, both works are purely software-based and do not leverage hardware-backed security mechanisms such as TEE, leaving cryptographic keys vulnerable to memory disclosure attacks.

Table II compares our work with other key management schemes in the automotive domain. As summarized in the table, our unique contribution lies in providing a TEE-based hierarchical key distribution scheme for the Ethernet-based zonal architecture with real-time requirements.

VII. CONCLUSION

This study proposes a TEE-based in-vehicle key distribution framework targeting the hierarchical zonal architecture. For that, we adopt the hierarchical key management scheme and tailor it for the hierarchical automotive E/E architecture. All the cryptographic operations are confined inside TEE, and all the sensitive data remain inside TEE to assure the hardware- and OS-level security. We implement a zonal architecture tested with multiple NVIDIA Jetson platforms equipped with OP-TEE. By rigorous measurements about the end-to-end key distribution delay, we demonstrate that it takes just 480 ms to renew all the keys inside a complex zonal architecture.

In our future work, we plan to integrate intrusion detection systems along with our on-demand key renewal framework for more practical security enhancement for future connected vehicles.

VIII. ACKNOWLEDGMENT

This work was partially supported by the BK21 Four Program (2120240815267) of the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Korea. Also this work was partially supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) RS-2024-00342879. J.-C. Kim is the corresponding author.

REFERENCES

- [1] R. S. Rathore, C. Hewage, O. Kaiwartya, and J. Lloret, "In-vehicle communication cyber security: Challenges and solutions," *Sensors*, vol. 22, no. 17, p. 6679, 2022.
- [2] A. Chattopadhyay, K.-Y. Lam, and Y. Tavva, "Autonomous vehicle: Security by design," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 11, pp. 7015–7029, 2020.
- [3] P. Mundhenk, A. Paverd, A. Mrowca, S. Steinhorst, M. Lukasiewicz, S. A. Fahmy, and S. Chakraborty, "Security in automotive networks: Lightweight authentication and authorization," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 22, no. 2, pp. 1–27, 2017.
- [4] T. Sakon and Y. Nakamoto, "Poster: Simple key management scheme for in-vehicle system," in *Proc. VNC*, 2016.
- [5] "OP-TEE: Open portable trusted execution environment." [Online]. Available: <https://optee.readthedocs.io/en/latest/general/about.html>. Accessed: Sep. 9, 2025.
- [6] D. Kenjić and M. Antić, "Connectivity challenges in automotive solutions," *IEEE Consum. Electron. Mag.*, vol. 12, no. 5, pp. 53–59, 2022.
- [7] G. Tortorelli, *A Service-Oriented Design Framework for Automotive Architectures*. PhD thesis, Politecnico di Torino, 2025.
- [8] M. Iorio, M. Reineri, F. Risso, R. Sisto, and F. Valenza, "Securing some/ip for in-vehicle service protection," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13450–13466, 2020.
- [9] "vsomeip: Covesa's open-source implementation of the some/ip protocol." [Online]. Available: <https://github.com/COVESA/vsomeip>. Accessed: Sep. 9, 2025.
- [10] "Globalplatform api." [Online]. Available: https://optee.readthedocs.io/en/latest/architecture/globalplatform_api.html. Accessed: Sep. 9, 2025.
- [11] S. Jadhav and D. Kshirsagar, "A survey on security in automotive networks," in *Proc. ICCUBEA*, 2018.
- [12] Y. Wu, L. Xiong, C. Wang, and Y. Xu, "Efficient secure mechanisms for in-vehicle ethernet in autonomous vehicles," *Electronics*, vol. 13, no. 18, p. 3625, 2024.
- [13] B. Shang, L. Liu, J. Ma, and P. Fan, "Unmanned aerial vehicle meets vehicle-to-everything in secure communications," *IEEE Commun. Mag.*, vol. 57, no. 10, pp. 98–103, 2019.
- [14] Y. Mo and B. Sinopoli, "Secure control against replay attacks," in *Proc. Allerton*, 2009.
- [15] V. L. Thing and J. Wu, "Autonomous vehicle security: A taxonomy of attacks and defences," in *Proc. iThings*, 2016.
- [16] J. Grover, V. Laxmi, and M. S. Gaur, "Attack models and infrastructure supported detection mechanisms for position forging attacks in vehicular ad hoc networks," *CSI Trans. ICT*, vol. 1, no. 3, pp. 261–279, 2013.
- [17] I. E. Carvajal-Roca, J. Wang, J. Du, and S. Wei, "A semi-centralized dynamic key management framework for in-vehicle networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 10864–10879, 2021.
- [18] H. Sun, W. Luo, J. Weng, Z. Liu, and M. Li, "Ecqv-gdh-based group key exchange protocol for can bus," *IEEE Trans. Veh. Technol.*, vol. 72, no. 10, pp. 12857–12872, 2023.
- [19] A. Musuroi, B. Groza, L. Popa, and P. Murvay, "Fast and efficient group key exchange in controller area networks (can)," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9385–9399, 2021.
- [20] J. Chen, Q. Zuo, W. Jin, Y. Wu, Y. Xu, and Y. Xu, "Study of network security based on key management system for in-vehicle ethernet," *Electronics*, vol. 13, no. 13, p. 2524, 2024.
- [21] H. Barati *et al.*, "A hierarchical key management method for wireless sensor networks," *Microprocess. Microsyst.*, vol. 90, p. 104489, 2022.
- [22] A. Albakri, M. Maddumala, and L. Harn, "Hierarchical polynomial-based key management scheme in fog computing," in *Proc. TrustCom and BigDataSE*, pp. 1593–1597, IEEE, 2018.