

Article

Authentication and Key Distribution for SAE J1939 CAN Bus Without Security-Designated ECU

Yufeng Li ^{1,2,3} , Jiajun Xi ^{1,2}, Jun Shen ^{1,2,*} and Jiangtao Li ^{1,2} ¹ School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China² Shanghai Key Laboratory of Intelligent Connected Vehicle Cybersecurity, Shanghai 200444, China³ Purple Mountain Laboratories, Nanjing 211111, China

* Correspondence: j_shen@shu.edu.cn

Abstract

As a higher-layer protocol over a controller area network (CAN) or CAN with a flexible data-rate bus, Society of Automotive Engineers (SAE) J1939 has been widely adopted in commercial vehicles. Although it supports advanced diagnostics, complex data transmission, and network management in harsh environments, SAE J1939 lacks native authentication mechanisms. Consequently, in-vehicle communication remains vulnerable to replay, spoofing, and injection attacks. In practice, deploying a Security-designated Electronic Control Unit (SeCU) is often deemed necessary to provide robust authentication, as generating and distributing session keys is essential. However, this introduces a single point of failure and renders the SeCU a high-value target for attackers. To address these issues, we propose J1939-ADBE, an authentication and key-distribution scheme that operates without a centralized SeCU. The scheme is built on Authenticated Distributed Broadcast Encryption (ADBE), a tightly integrated construction that augments distributed broadcast encryption with publicly verifiable sender authentication in a shared bilinear setting. By leveraging ADBE, we eliminate the requirement for a SeCU while achieving the desired security goals. Using the Tamarin Prover, we formally verify in the Dolev–Yao model that J1939-ADBE satisfies injective agreement, session secrecy, known-key security, and forward secrecy. Furthermore, the broadcast nature of ADBE reduces the communication cost of key distribution from $\mathcal{O}(n)$ to $\mathcal{O}(|\mathcal{G}|)$, where n denotes the number of Electronic Control Units (ECUs) and $|\mathcal{G}|$ denotes the number of ECU logical groups. Experimental results show that our proposal is practical for authentication within SAE J1939 networks.

Keywords: commercial vehicles; SAE J1939; authentication; key management; cybersecurity; network security



Academic Editor: Aryya Gangopadhyay

Received: 9 March 2026

Revised: 1 April 2026

Accepted: 7 April 2026

Published: 15 April 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Intelligent and Connected Vehicles (ICVs) increasingly integrate wireless and remote connectivity, which enhances functionality but weakens the traditional assumption that the In-Vehicle Network (IVN) is isolated and trustworthy. Currently, the IVN is still predominantly built on the Controller Area Network (CAN) bus [1]. In commercial vehicles, Society of Automotive Engineers (SAE) J1939 is widely adopted as a higher-layer protocol over CAN or CAN with a Flexible Data-rate (CAN-FD) bus. Beyond basic real-time messaging, SAE J1939 provides standardized mechanisms for diagnostics, multi-packet data transport, and network management, enabling reliable communication under harsh operating conditions. However, similarly to CAN, SAE J1939 does not provide native cryptographic

authentication, leaving in-vehicle communication exposed if adversaries obtain access to the bus.

With expanded attack surfaces through wireless and remote interfaces, an adversary may eavesdrop on traffic and, further, perform replay or spoofing operations and inject forged messages [2]. Such capabilities undermine both operational correctness and safety-relevant behaviors in commercial vehicles [2]. Practical protection, therefore, typically requires (i) message authentication to ensure origin integrity and (ii) secure session-key establishment and refreshing to support efficient symmetric authentication at scale. In bandwidth- and latency-constrained bus environments, the design of session-key distribution is often a primary determinant of system-wide security and deployability.

Prior studies on in-vehicle security mechanisms can be broadly grouped into three approaches. Physical-property-based approaches [3–5] exploit bus characteristics to derive shared secrets with low overhead, but rely on strong assumptions and may be sensitive to device variability and side channels [6]. Pre-shared-secret schemes [7–11] mainly perform lightweight message or broadcast authentication in a CAN bus through symmetric material and compact authentication checks, yet their security degrades sharply once any long-term secret is compromised. Cryptography-based designs [12–17] provide stronger and more flexible guarantees via standard primitives; in some cases, they additionally incorporate explicit key-establishment support or trusted auxiliary components for authenticated communication [14]. Formal analyses have shown that such designs still require careful protocol engineering under CAN constraints [18]. Among these directions, the latter two are particularly relevant because they capture the trade-off between lightweight broadcast authentication and richer authenticated communication with key-establishment support.

While much of the above work targets the CAN layer, directly authenticating all CAN/CAN-FD frames can impose unnecessary overhead, including for non-critical traffic. In contrast, implementing security at a higher layer allows manufacturers to selectively protect security-sensitive services while preserving existing real-time performance. Moreover, J1939 is an open standard for commercial vehicles, unlike the proprietary protocols in passenger cars, making it a more suitable and general research target. However, dedicated security research on this protocol remains limited [1,19]. As a result, selective protection and practical session-key lifecycle management within SAE J1939 remain insufficiently explored. Among these challenges, key management is particularly critical because practical authentication at scale still depends on efficient session-key generation, distribution, and updating. In current practice, a security-designated ECU (SeCU) is often introduced to centrally hold and distribute current keying material for the network. However, this SeCU-centric architecture introduces two structural risks: (i) it constitutes a single point of failure that can disrupt key updates and thereby degrade secure communications if any hardware or software failure occurs in SeCU; and (ii) SeCU becomes a high-value target whose compromise may expose or manipulate multiple session keys, causing system-wide loss of confidentiality and integrity.

To overcome these limitations, we propose J1939-ADBE, an authentication and key-distribution scheme for SAE J1939 that removes the need for a dedicated runtime SeCU to centrally hold and distribute current keying material. Our approach is built on Authenticated Distributed Broadcast Encryption (ADBE), an authenticated extension of Distributed Broadcast Encryption (DBE), for which we give an efficient construction.

Compared with existing in-vehicle security designs, the novelty of our work lies in three aspects. First, rather than attaching protection uniformly to CAN/CAN-FD traffic or relying on point-to-point rekey exchanges, we achieve authenticated one-to-many rekeying at the SAE J1939 layer, where protection can be selectively applied to security-sensitive services. Second, unlike existing J1939-specific schemes that still depend on a

centralized SeCU for session-key lifecycle management, our design decentralizes runtime key distribution across authorized ECUs and avoids concentrating current keying material at a single in-vehicle node. Third, although DBE offers efficient receiver-selective broadcast delivery, it cannot be directly used for hostile J1939 environments because it lacks publicly verifiable sender authentication for rekeyed messages. ADBE closes this gap by tightly integrating DBE-style receiver selectivity with BLS-based public verifiability in a shared bilinear setting, thereby enabling a single authenticated broadcast to securely rotate keys for a designated receiver set.

The main contributions of this paper are summarized as follows:

- We formalize ADBE and provide an efficient construction that combines receiver-selective encryption with publicly verifiable sender authentication in a shared bilinear setting, while preserving DBE-style confidentiality and correctness.
- Based on ADBE, we design J1939-ADBE for SAE J1939 networks to enable decentralized session-key distribution and rotation without a centralized SeCU, achieving injective agreement, session secrecy, known-key security, and forward secrecy.
- J1939-ADBE reduces the communication cost of key distribution from $\mathcal{O}(n)$ to $\mathcal{O}(|\mathcal{G}|)$, where n denotes the number of ECUs and $|\mathcal{G}|$ denotes the number of ECU logical groups.
- Compared with prior representative SAE J1939 authentication schemes, J1939-ADBE reduces the total message delay of one vehicle-wide key rotation by 41.5–81.6% under typical CAN-FD settings (64-byte payload and 250/500 kbps bus rates).

2. Related Work

2.1. Attacks on In-Vehicle Networks

CAN was designed for reliability and efficiency rather than security, and thus lacks authentication and encryption [1]. Early work demonstrated practical compromises via physical access and in-vehicle interfaces [20], and later work showed that remote attack surfaces can also lead to control of safety-relevant functions [21]. Subsequent studies exploited vulnerabilities in Bluetooth [22,23], Wi-Fi [24,25], cellular networks [26], and radio communications [27,28] to remotely compromise vehicle systems. These findings collectively motivate cryptographic authentication and confidentiality for IVNs [22].

Commercial vehicles using SAE J1939 are similarly exposed. Burakova et al. [2] demonstrated, for the first time, that J1939-based systems are vulnerable to CAN-layer attacks by attackers with physical access to a truck and a school bus. Mukherjee et al. [29] identified three Denial-of-Service (DoS) attacks in J1939, while Murvay et al. [1] further identified five categories of attacks arising from the lack of authentication and encryption in SAE J1939. These works collectively underline the necessity of secure key-exchange and authentication mechanisms for IVNs.

2.2. Defense Mechanisms in the CAN Bus

With an increasing number of attack cases disclosed, researchers have recognized the need to introduce defensive mechanisms into the CAN bus. Existing key-exchange and authentication schemes can be broadly divided into three categories: schemes based on physical properties, pre-shared secrets, and cryptographic primitives.

2.2.1. Schemes Based on Physical Properties

These schemes exploit the electrical or timing properties of a bus to gain access to secrets. Representative examples include pairwise constructions that use CAN arbitration or related wired-AND effects to establish shared material between ECUs [3,4], as well as later group-oriented extensions that build on the same physical behavior for multi-node

key establishment [5]. Because they retrieve secrets from bus interactions rather than from pre-deployed credentials, they can be lightweight and are attractive for constrained ECUs. At the same time, their operation often depends on tight synchronization, stable device characteristics, and additional assumptions about what an adversary can observe on the bus. Subsequent analysis shows that probing and related side channels can weaken these assumptions in practice [6]. Their scalability and deployability in production IVNs are therefore limited.

2.2.2. Schemes Based on Pre-Shared Secrets

These schemes exploit pre-installed long-term symmetric secrets and derive per-session or per-sender authentication keys using hashing or symmetric encryption. Representative designs include LiBrA-CAN, which authenticates CAN broadcasts through key splitting and optional Message Authentication Code (MAC) mixing [7]; CANAuth, which pursues backward-compatible CAN authentication using counters and truncated Hashed Message Authentication Codes (HMACs) derived from pre-shared material [8]; hierarchical derivation to reduce storage [10]; and variants that use multiple MACs or centralized controllers [9,11]. Stabili et al. [30] further analyze the deployment implications of automotive communication-security protocols across the vehicle life cycle. These approaches are efficient, but compromise of a pre-shared secret typically triggers costly re-provisioning and broad loss of security.

2.2.3. Schemes Based on Cryptographic Primitives

These schemes exploit cryptographic primitives such as key exchange [12,13], signatures [12], and MAC-based message authentication [14–17] to establish session keys and protect in-vehicle communication. Compared with schemes based on pre-shared secrets, they target richer authenticated communication. For example, MaCAN authenticates CAN messages with cipher-based MACs while relying on a key server for key management and a time server for freshness [14], while Bruni et al. [18] show that such designs still require careful protocol engineering under CAN constraints. While these approaches substantially strengthen security, they often raise deployment concerns around latency, scalability, and management complexity. Moreover, because many of these protections are attached directly to bus traffic, the overhead is often attached to all messages, including non-critical frames. This motivates a push for comparable protection to be moved to a higher-layer protocol such as J1939, where protection can be applied selectively.

2.3. Defense Mechanisms in SAE J1939

Murvy et al. proposed J1939-ACAN [1], which appends a per-message MAC computed from a session key, timestamp, and counter. A SeCU distributes session keys and provides each ECU with a dedicated synchronization key to maintain consistent timing across the network. Palaniswamy et al. [19] later showed that the identity credentials used during key acquisition are not authenticated, so an attacker can forge credentials, prevent legitimate ECUs from obtaining correct synchronization keys, and trigger DoS via time desynchronization.

To mitigate this issue, Palaniswamy et al. proposed CL-KIMS [19], where ECUs generate key pairs and sign their requests to the SeCU using elliptic curve. However, without a trusted certification mechanism, adversaries can still impersonate ECUs and submit fraudulent requests. In addition, as SeCU responses are not strongly authenticated, adversaries can also impersonate the SeCU and distribute forged session keys, leading to key inconsistency and DoS.

Furthermore, both J1939-ACAN and CL-KIMS rely on a centralized SeCU for session-key lifecycle management. These J1939-specific designs are more protocol-aware

than generic CAN-bus authentication schemes, but they still retain a centralized key-management bottleneck. As a result, their security and availability hinge on the correctness and protection of that single component. A J1939-oriented mechanism that preserves selective protection while avoiding a single point of failure in session-key distribution and update remains to be developed.

2.4. Distributed Broadcast Encryption and Its Limitation

DBE [31] is a group-oriented primitive where a sender encrypts to a chosen recipient set using their public keys and embeds the key material in the ciphertext header. Any authorized receiver recovers the decryption key with its private key, while unauthorized nodes learn nothing about the plaintext. Because the ciphertext targets the entire set, a single broadcast can deliver fresh keys or protected data to many ECUs, reducing communication overhead for vehicle-wide or domain-wide updates. DBE also removes the need for a central key server, avoiding the single point of failure in SeCU-based designs.

A key limitation is that standard DBE lacks publicly verifiable origin authentication: receivers cannot verify whether a ciphertext was produced by an authorized sender. This enables forged broadcasts and ciphertext substitution across groups or update phases, which can cause incorrect key rotation, decryption failures, and DoS.

Short, publicly verifiable signatures such as the Boneh–Lynn–Shacham signature (BLS) [32] are well suited to IVNs due to their compact size and efficient verification. Since both DBE and BLS can be implemented in the same bilinear setting, sender authentication can be added to DBE in a lightweight manner by binding the ciphertext to the sender identity. This motivates the ADBE construction adopted in our broadcast-centric SAE J1939 design.

3. Background and Problem Statement

3.1. SAE J1939 Overview

SAE J1939 is a standardized higher-layer protocol family for commercial vehicles; it is defined by SAE within the CAN/CAN-FD bus and is widely used in trucks, construction machinery, and agricultural equipment.

J1939 adopts the 29-bit extended CAN identifier format defined in SAE J1939-21 as shown in Figure 1. Each identifier consists of a 3-bit priority, an 18-bit Parameter Group Number (PGN), and an 8-bit Source Address (SA). The PGN uniquely specifies an application-level parameter group, i.e., a message type with a fixed layout and semantics, and the SA identifies the ECU that transmits the frame. In classical CAN, J1939 data frames are typically carried at 250 kbps or 500 kbps with a payload of up to 8 bytes; on CAN-FD this payload can be extended to 64 bytes. These payload and bit-rate parameters are explicitly taken into account in the performance evaluation of our scheme.

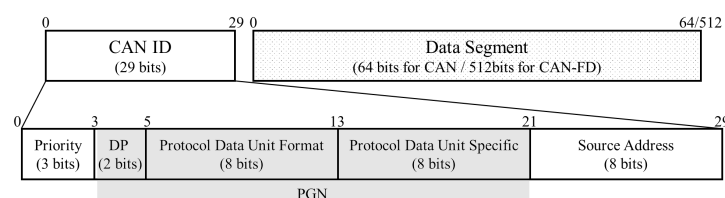


Figure 1. Society of automotive engineers J1939 data message format.

In typical J1939 deployments, ECUs may join and leave the network at runtime (e.g., when connecting or disconnecting trailers), so the protocol explicitly supports ECU addition and removal. When a new ECU is attached, it participates in the address-claim procedure: the ECU announces its NAME and claims an 8-bit SA on the bus, and any conflicts are

resolved according to priority rules. When an ECU leaves, it simply stops using its SA. This dynamic address–claim mechanism allows the set of active ECUs to change over time while preserving a consistent addressing view for higher-layer protocols.

Because individual CAN/CAN-FD frames have limited payload, J1939 defines a Transport Protocol (TP) for messages whose length exceeds a single frame. The TP consists of two sub-protocols: Transport Protocol Connection Management (TP.CM) for connection management in point-to-point transfers and Transport Protocol Broadcast Announce Message (TP.BAM) for broadcast or multicast transfers. In TP.CM, the sender and receiver use Request-To-Send/Clear-To-Send style control messages so that the receiver can regulate the number and rate of incoming segments. In TP.BAM, the sender first broadcasts a short announcement and then transmits all data packets in sequence, so that a single logical message can be sent to all potential receivers using a fixed number of J1939 frames.

In the proposed design, the above J1939 characteristics are explicitly taken into consideration. Group-oriented authentication and key-distribution messages are encoded into J1939 frames such that a single authenticated broadcast can update keys or distribute security-relevant state to all intended receivers.

3.2. Cryptographic Primitives

The scheme construction and security analysis are carried out in a standard bilinear-group setting and rely on conventional notions of confidentiality and unforgeability.

Let G_1 , G_2 , and G_T be cyclic groups of prime order p , generated by $g_1 \in G_1$ and $g_2 \in G_2$. A bilinear pairing is an efficiently computable, non-degenerate map $e : G_1 \times G_2 \rightarrow G_T$ such that $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ holds for all $a, b \in \mathbb{Z}_p$. Bilinear groups of this form serve as the mathematical environment for the ADBE construction.

A digital-signature scheme is existentially unforgeable under chosen-message attacks (EUF-CMA) if no probabilistic polynomial-time (PPT) adversary, even with access to a signing oracle for adaptively chosen messages, can output a valid signature on any new message (i.e., one not previously submitted to the oracle) except with negligible probability. Informally, EUF-CMA prevents an attacker from forging signatures on unseen messages.

A public-key encryption scheme is indistinguishable under chosen-plaintext attacks (IND-CPA) if no PPT adversary can distinguish encryptions of two chosen plaintexts with non-negligible advantage, even given access to an encryption oracle for messages of its choice. Informally, IND-CPA ensures that ciphertexts leak no information about the plaintext beyond its length.

Throughout the remainder of the paper, the pairing groups are assumed to satisfy standard hardness assumptions (of Diffie–Hellman type), and the building blocks used in the ADBE construction are required to achieve EUF-CMA’s unforgeability and IND-CPA’s confidentiality. A detailed proof that the proposed ADBE scheme satisfies these properties under appropriate assumptions is given in Section 4.3.

3.3. Problem Statement

3.3.1. System Model

The in-vehicle architecture of commercial vehicles can be abstracted as shown in Figure 2, and consists of two parts: the tractor and the trailer. The tractor and trailer can be physically connected or detached, and thus the attachment/detachment of a trailer may cause dynamic changes in the set of ECUs. For non-articulated commercial vehicles, the tractor and trailer can be treated as a single inseparable entity.

Within each entity, ECUs are interconnected via a CAN bus. On the tractor side, the gateway ECU is also connected to the T-Box. Thus, the gateway ECU can provide proxy

Internet access for in-vehicle ECUs through the T-Box, enabling low-frequency maintenance and onboarding interactions with the original equipment manufacturer (OEM) backend.

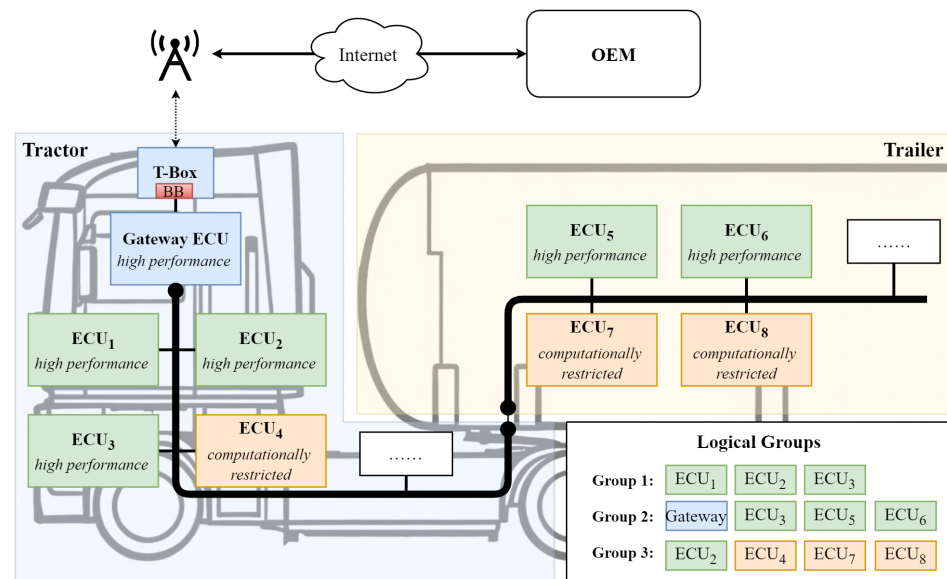


Figure 2. Example in-vehicle architecture of commercial vehicles.

Each component of the model is described as follows:

OEM. The OEM acts as the root provisioning and enrollment authority for the vehicle. During manufacturing, it initializes each vehicle with the required cryptographic materials, including the per-vehicle public parameters and seed, and authorizes the initial slot bindings for onboard ECUs. During ECU Join, it further determines whether a requesting ECU is valid for the target vehicle, releases the per-vehicle public parameters and seed when needed, and decides which new slot should be allocated. The concrete realization of this OEM-assisted provisioning and enrollment procedure is deployment-specific and orthogonal to the ADBE core.

T-Box. The T-Box provides cellular connectivity and serves as the vehicle-side entry point for interactions with the OEM backend. In this scheme, it enables proxy connectivity for in-vehicle ECUs via the gateway ECU using protocols such as SOME/IP without changing the in-vehicle CAN communication model. In addition, the T-Box hosts a bulletin board service (TBOX.BB) used by J1939-ADBE to publish public registration metadata, including ECU identifiers, slot bindings and public-key records. Consistent with the DBE model, this bulletin board is treated as a public append-only registry rather than as a separate cryptographic authority or a SeCU-style key manager. For example, such append-only behavior may be achieved by a monotonic log with sequence numbers and hash chaining, or by any equivalent append-only storage mechanism.

Gateway ECU. The gateway ECU connects and isolates different network domains/subnets and enforces cross-domain forwarding and policies. In prior approaches, the gateway ECU is often treated as a SeCU that generates and distributes keys. In our scheme, it is merely a normal ECU that may join multiple logical groups, but it is not responsible for generating or centrally distributing session keys for all ECUs.

High performance ECU. A high performance ECU has sufficient computational resources to execute the cryptographic operations required by the proposed scheme, and provides protected local storage for long-term secret material and freshness-critical state (e.g., via a hardware security module (HSM) or equivalent cryptographic support). Concretely, such ECUs can participate in authenticated group key distribution, rekeying, and protected-message processing under OEM-defined group policies. In current vehicle ar-

chitectures, this category covers a broad set of ECUs, including gateway ECUs, domain controller ECUs, and a range of powertrain, chassis, body, and trailer-related controllers that provide the required computing and secure-storage support. With the continuous improvement of ECUs' computing capabilities and hardware security support, an increasing fraction of ECUs is expected to fall into this category over time.

Computationally restricted ECU. A computationally restricted ECU does not provide the computational or secure-storage support required by the proposed scheme. Therefore, it cannot participate in authenticated group key distribution and rekeying, and continues to communicate using native J1939. Such ECUs may still exist for cost and compatibility reasons, and the resulting exposure is explicitly acknowledged and bounded by OEM deployment policy.

Remark 1. *The OEM partitions ECUs into logical groups by function/domain according to application requirements during production, and an ECU may belong to multiple groups to support cross-domain coordination. However, a logical group does not always coincide with the set of ECUs that actually participate in communication. In practice, the effective participants may vary because some target ECUs are temporarily offline, trailer ECUs may be attached or detached, or certain ECUs may be excluded after compromise or maintenance events. Therefore, the effective participants in a logical group are determined jointly by the OEM-defined grouping policy and the current operational and trust status of the involved ECUs, and our scheme supports such dynamic participation in group communication. A logical group may contain a mixture of high performance ECUs and computationally restricted ECUs (e.g., Group 3 in Figure 2). To ensure that all currently participating members in such a mixed group can communicate correctly and consistently, the group uniformly uses native J1939 communication when any computationally restricted ECU is present.*

3.3.2. Adversary Model

We assume that an adversary can gain access to the vehicle's J1939 communication channel through a compromised in-vehicle node or an external interface (e.g., a diagnostic interface or infotainment entry point). Consequently, the adversary can eavesdrop on message exchanges among in-vehicle ECUs, and is capable of injecting crafted messages onto the bus, modifying message contents, and replaying previously observed messages. Even without detailed knowledge of the protocol semantics and message formats, the adversary may still attempt to induce high performance ECUs to execute unintended control-related actions via replaying and tampering.

Meanwhile, we assume that the adversary cannot break standard cryptographic primitives within a reasonable time, and cannot obtain long-term secret materials stored in the HSM of high performance ECUs (e.g., private keys, the vehicle seed, and counters). CAN bus DoS and other availability attacks (e.g., bus congestion or physical-layer interference) are out of scope, since such attacks cannot be fundamentally prevented by cryptographic scheme design alone and are typically mitigated via the in-vehicle intrusion detection system. Likewise, invasive physical compromise of high performance ECUs (e.g., hardware dismantling or side-channel extraction of HSM-resident secrets) is not considered in this paper; such threats are better addressed through hardware protections, detection-and-response mechanisms, and operational maintenance procedures.

3.3.3. Design Goals

SAE J1939 networks lack native authentication and are therefore vulnerable to message replay and injection on the broadcast bus. With the continuous improvement of ECU computing capabilities and hardware security support, an increasing fraction of ECUs can deploy stronger cryptographic mechanisms to strengthen in-vehicle communication security.

Existing J1939 security designs typically rely on SeCU to generate and distribute session keys for other ECUs. This architecture concentrates knowledge of all group/session keys at a single node, which violates the principle of least privilege and consequently makes the SeCU a high-value attack target and introduces a single point of failure. Motivated by these limitations, we aim to remove the centralized SeCU dependency for session-key distribution in J1939 networks. Accordingly, we establish the following concrete design goals.

Functional and deployment goals. We aim to deploy a decentralized session key distribution and rekey mechanism on high performance ECUs, so that subsequent in-vehicle messages can be authenticated with session keys and the overall communication security can be strengthened without relying on a centralized SeCU. The design should remain compatible with the existing SAE J1939 protocol stack and current vehicle architectures, i.e., it should not require changing the basic J1939 communication model or message transport, but rather provide cryptographic protection at the application layer in an implementable manner. Moreover, commercial vehicles may involve tractor-trailer attachment and detachment, which introduces dynamic membership changes; therefore, the mechanism should support Join/Leave in maintenance and onboarding scenarios while preserving consistent group communication among authorized ECUs. Finally, since in-vehicle buses carry a large volume of traffic and long-lived keys enlarge the exposure window, the scheme should support automated key updates under configurable policies (e.g., periodic or event-triggered rekeying) with limited operational overhead.

Security goals. Secure key distribution and rekeying are prerequisites for achieving authenticated communication. If an adversary can manipulate the key-update process, it may drive ECUs into incorrect or stale session states, cause cross-group misuse, and ultimately undermine message authentication, making replay, injection, and man-in-the-middle (MITM) tampering defenses ineffective. Accordingly, under an adversary who can eavesdrop, inject, modify, and replay frames on the J1939 bus, we require the key distribution and rekeying workflow to meet the following goals:

1. **Secure key updates against active attacks.** The fresh keying material used in a key update must remain protected against an attacker who (i) launches attacks by forging fake update information and (ii) can read the key-update information of other groups. Accordingly, forged, tampered, replayed, or cross-group-forwarded update messages must not drive honest ECUs into an incorrect session state, and unauthorized nodes must not recover the fresh keying material of the target group.
2. **Session key secrecy.** Session keys must remain confidential against an attacker who can read all protocol messages and can additionally obtain session keys from previous rounds. In particular, observing all traffic and learning earlier-round keys must not enable the attacker to derive the current session key or any other uncompromised round key.
3. **Forward security.** Leakage of long-term secret material must not affect the security of previously established session keys.
4. **Unforgeability.** As long as sessions keys are not leaked, an attacker who has observed multiple authenticated messages must still be unable to forge a valid signature or MAC for any fresh message.

Efficiency goals. The overall latency of the scheme should be acceptable for practical commercial-vehicle deployments, and key distribution/rekeying should not impose excessive computational delay or disrupt time-critical in-vehicle communication. The communication overhead introduced by key updates should remain bounded and should not create undue bus load on CAN/CAN-FD, while the per-frame overhead should remain compatible with J1939 payload and transport constraints. Finally, computation and com-

munication costs should scale to realistic group sizes for commercial vehicles, keeping the protocol implementable on target ECUs without introducing prohibitive latency on the bus.

4. Building Block: ADBE

ADBE augments DBE with publicly verifiable sender authentication in a shared bilinear setting, where the same algebraic environment supports both DBE-style broadcast encapsulation and BLS-style public verifiability. Concretely, the construction is specified through six algorithms, namely Setup, KeyGen, Encrypt, Sign, Verify, and Decrypt. This formulation preserves a modular presentation of confidentiality and authentication, while allowing the two functionalities to reuse shared public parameters and user key material with only minimal additional public information.

4.1. ADBE Construction

Let λ denote the computational security parameter. Let G_1, G_2, G_T and the bilinear pairing $e : G_1 \times G_2 \rightarrow G_T$ be as in Section 3.2, where the bilinear-group description $\text{bg} = (p, G_1, G_2, G_T, e, g_1, g_2)$ is instantiated with prime order p whose bit-length is determined by λ . Thus, λ fixes the concrete security level of the scheme. Let $H : \{0, 1\}^* \rightarrow G_1$ be a hash-to-curve function modeled as a random oracle. The ADBE scheme consists of the following algorithms.

- **Setup**($1^\lambda, 1^L$): Given the security parameter λ and the maximum number of users L , the authority samples a secret exponent $\alpha \in \mathbb{Z}_p^*$ and publishes a set of precomputed group elements that will be used by both the encryption and decryption procedures: $\text{pp} = (\text{bg}, [\alpha]_1, \dots, [\alpha^L]_1, [\alpha]_2, \dots, [\alpha^L]_2, \dots, [\alpha^{2L}]_2)$, where $[x]_1 := g_1^x$ and $[x]_2 := g_2^x$. These public parameters are shared by all participants and no additional master secret is retained beyond α .
- **KeyGen**(pp, j): For $j \in [L]$, user j (i.e., an ECU during provisioning) samples $t_j \in \mathbb{Z}_p^*$ and derives its long-term secret key and public information: $t_j \leftarrow_{\$} \mathbb{Z}_p^*$, $\text{usk}_j := ([t_j \alpha^{L+1-j}]_2, t_j)$, $\text{upk}_j := ([t_j]_1, \{[t_j \alpha^\ell]_2\}_{\ell \in [L], \ell \neq L+1-j}, [t_j]_2)$. Here, usk_j is used for both DBE-style decryption and BLS-style signing, while upk_j is public and allows other nodes to derive the necessary encapsulation components and to verify signatures.
- **Encrypt**($\text{pp}, \{\text{upk}_j\}_{j \in S}, S, M$): Given a recipient set $S \subseteq [L]$, and a message $M \in G_T$, the encryption algorithm produces a broadcast ciphertext ct using the DBE encapsulation structure. It samples $s \leftarrow_{\$} \mathbb{Z}_p^*$ and computes $\text{ct} = ([s]_1, [s \sum_{j \in S} (t_j + \alpha^j)]_1, [s \alpha^{L+1}]_T \cdot M)$, where $[x]_T := e(g_1^x, g_2)$ for notational convenience. The term $[s \sum_{j \in S} (t_j + \alpha^j)]_1$ can be evaluated using only public information, for example as $[s \sum_{j \in S} (t_j + \alpha^j)]_1 = \prod_{j \in S} ([st_j]_1 \cdot [s \alpha^j]_1)$, and the masking term $[s \alpha^{L+1}]_T$ is obtained as $[s \alpha^{L+1}]_T = e([s \alpha]_1, [\alpha^L]_2)$. The ciphertext ct is then passed to the signing algorithm to attach origin authentication.
- **Sign**($\text{ct}, \text{usk}_s, s$): To authenticate the origin of a ciphertext, the sender s signs the concatenation of the ciphertext components and its own identifier. Let $\text{ct} = (\text{ct}_1, \text{ct}_2, \text{ct}_3)$. Using the secret t_s embedded in usk_s , compute $m = \text{ct}_1 \parallel \text{ct}_2 \parallel \text{ct}_3 \parallel s$, $\sigma = H(m)^{t_s} \in G_1$. The authenticated ciphertext broadcast over the J1939 bus is the tuple (ct, σ, s) .
- **Verify**($(\text{ct}, \sigma, s), \text{upk}_s$): Any receiver first validates the origin of a received tuple (ct, σ, s) . From upk_s it extracts the public verification component $[t_s]_2$ and checks $e(\sigma, g_2) \stackrel{?}{=} e(H(\text{ct}_1 \parallel \text{ct}_2 \parallel \text{ct}_3 \parallel s), [t_s]_2)$. If and only if this equality holds, the ciphertext is accepted as a valid broadcast from sender s and may be forwarded to the DBE decryption routine.
- **Decrypt**($\text{pp}, \{\text{upk}_j\}_{j \in S}, \text{usk}_i, \text{ct}, S, i$): Upon successful verification, any receiver with index $i \in S$ recovers M by applying the DBE decryption structure to the ciphertext

components and the key material associated with the receiver i and the recipient set S . Let $\text{pp}_{2L+1-i} := [-\alpha^{L+1-i}]_2$, $\text{usk}_{i,0} := [t_i \alpha^{L+1-i}]_2$, $\text{upk}_{j,L+1-i} := [t_j \alpha^{L+1-i}]_2$, and $\text{pp}_{2L+1+j-i} := [\alpha^{L+1+j-i}]_2$.

Then the recovered plaintext is given by

$$M' = \text{ct}_3 \cdot e(\text{ct}_2, \text{pp}_{2L+1-i}) \cdot e\left(\text{ct}_1, \text{usk}_{i,0} \cdot \prod_{j \in S, j \neq i} (\text{upk}_{j,L+1-i} \cdot \text{pp}_{2L+1+j-i})\right). \quad (1)$$

By construction of the DBE structure, any legitimate receiver $i \in S$ obtains $M' = M$, whereas nodes without appropriate secret material cannot invert the masking term and thus learn nothing about M .

4.2. Correctness

Correctness follows directly from the DBE decryption equations and the BLS verification equation. First, for an honestly generated signature $\sigma = H(m)^{t_i}$ on $m = \text{ct}_1 \parallel \text{ct}_2 \parallel \text{ct}_3 \parallel i$, and public element $[t_i]_2 = g_2^{t_i}$, bilinearity yields $e(\sigma, g_2) = e(H(m)^{t_i}, g_2) = e(H(m), g_2^{t_i}) = e(H(m), [t_i]_2)$, so the verification equation always holds for ciphertexts produced by an honest sender. The signature is computed solely over the ciphertext components and the sender identifier and does not modify the DBE header or body. Therefore, whenever verification succeeds and $j \in S$, the DBE part of ADBE is identical to the underlying DBE scheme, and applying the decryption Equation (1) recovers M .

4.3. Security Analysis

ADBE combines two security objectives: confidentiality of the broadcast payload and publicly verifiable authentication of the sender. These objectives are proved compositionally, through reductions to the underlying DBE and BLS components, respectively. This compositional proof structure concerns the organization of the security argument rather than the way the scheme is instantiated. In the actual construction, both functionalities are realized in the same bilinear setting and reuse shared public parameters and user key material, with public verification enabled by only minimal additional public information.

We now argue that ADBE provides origin authentication (EUF-CMA) and confidentiality (IND-CPA) under the standard assumptions for BLS and DBE.

Theorem 1 (Origin authentication/EUF-CMA). *If the underlying BLS signature scheme is EUF-CMA secure (e.g., under a suitable bilinear hardness assumption in the random-oracle model [32]), then the ADBE signing mechanism is EUF-CMA secure.*

Proof. By construction, ADBE realizes public origin authentication by evaluating a BLS-style signature within the same bilinear setting used by DBE. Concretely, the authenticated component is a signature on $m = \text{ct}_1 \parallel \text{ct}_2 \parallel \text{ct}_3 \parallel i$, $\sigma = H(m)^{t_i}$, with public key $[t_i]_2$. Any adversary that outputs a new authenticated ciphertext $(\text{ct}^*, \sigma^*, i)$ accepted by the ADBE verifier immediately yields a valid BLS forgery (m^*, σ^*) on a new message m^* . A reduction therefore wraps the ADBE adversary, forwards the BLS public key as $[t_i]_2$, uses the BLS signing oracle to answer all signing queries, and outputs the final pair as a BLS forgery. Hence any non-negligible EUF-CMA advantage against ADBE would contradict the EUF-CMA security of BLS. $\square$

Theorem 2 (IND-CPA confidentiality). *If the underlying DBE encryption scheme is IND-CPA secure, then ADBE is IND-CPA secure.*

Proof. Let $\mathcal{A}$ be an IND-CPA adversary against ADBE. Consider a DBE IND-CPA adversary $\mathcal{B}$ that simulates ADBE for $\mathcal{A}$ as follows. It receives DBE public parameters and forwards them unchanged; it then samples $t_i \leftarrow \mathbb{Z}_p^*$ on its own and publishes $[t_i]_2 = g_2^{t_i}$ as part of the ADBE public key. This extra element is independent of all DBE secret keys and is generated without querying the DBE challenger, so revealing $[t_i]_2$ does not affect the DBE security game.

For any encryption query (S, M) , $\mathcal{B}$ asks its DBE encryption oracle for a ciphertext $ct = (ct_1, ct_2, ct_3)$ and returns (ct, σ, i) , $\sigma = H(ct_1 \parallel ct_2 \parallel ct_3 \parallel i)^{t_i}$, which has exactly the ADBE distribution. For the challenge (S, M_0, M_1) , it forwards the pair to the DBE challenger, obtains a challenge ciphertext ct^* encrypting M_b , computes $\sigma^* = H(ct_1^* \parallel ct_2^* \parallel ct_3^* \parallel i)^{t_i}$, and returns (ct^*, σ^*, i) to $\mathcal{A}$. The view of $\mathcal{A}$ in this simulation is indistinguishable from its view in a real ADBE IND-CPA experiment, since ct comes from the true DBE challenger and σ is a deterministic function of public data and the local secret t_i , independent of the challenge bit. Therefore, $\mathcal{B}$'s advantage against DBE equals (up to a negligible term) $\mathcal{A}$'s advantage against ADBE, and IND-CPA security of DBE implies IND-CPA security of ADBE. $\square$

5. Authentication and Key Management Scheme for J1939 CAN Bus

This section describes the proposed J1939-ADBE scheme for secure group communication in IVNs. The scheme leverages ADBE to distribute group keys efficiently among ECUs, achieving message authentication and confidentiality while maintaining full compatibility with SAE J1939 protocol.

5.1. Scheme Description

Figure 3 illustrates a J1939-based IVN consistent with the architecture and roles defined in Section 3.3.1. ECUs communicate over one or more CAN (or CAN-FD) segments using SAE J1939. The bulletin board service TBOX.BB stores the public parameters pp together with public registration tuples $\langle ID_i, i, upk_i \rangle$, while each ECU $_i$ keeps its long-term secret key usk_i and the vehicle-wide seed $seed$ in HSM-protected storage.

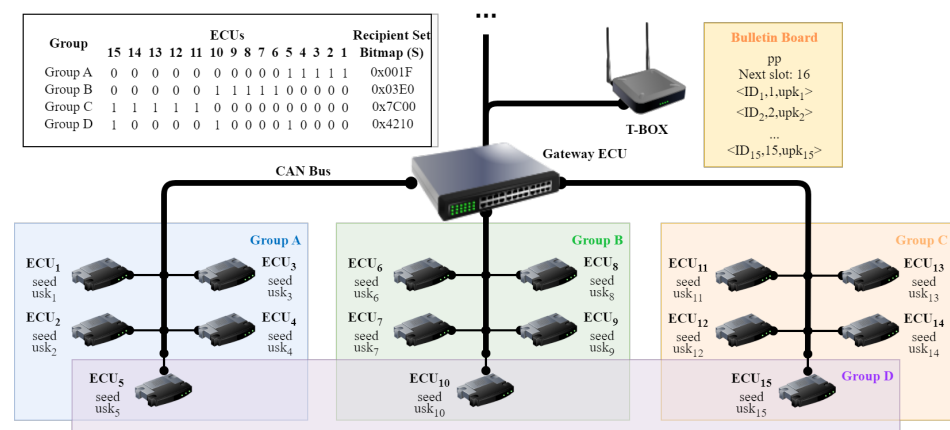


Figure 3. Example J1939-based IVN architecture.

At the application layer, the OEM specifies logical groups $\mathcal{G} = \{G_1, G_2, \dots, G_{|\mathcal{G}|}\}$ over authorized ECU identities and assigns each group an identifier. Because the actual vehicle composition may change over time, communication within a logical group is realized at runtime through a receiver set $S_\ell \subseteq [L]$, obtained by resolving the currently valid ECU identities of G_ℓ to their valid slot identifiers through TBOX.BB. This receiver set is re-evaluated whenever vehicle membership changes, and the corresponding rekey is then carried out for the updated S_ℓ . For each runtime receiver set S_ℓ , member ECUs derive

a session key $K_{\text{ses}}^{(\ell)}$ and a synchronization key $K_{\text{sync}}^{(\ell)}$ from $(\text{seed}, \delta_\ell)$; cross-group isolation holds because only ECUs in S_ℓ can decrypt the corresponding update value δ_ℓ .

To abstract this architecture, we adopt the following notation. Let the vehicle network expose a preconfigured slot space of size L , indexed by $[L] = \{1, 2, \dots, L\}$. These slot identifiers serve as the DBE indices for ECU registration, whereas the number of ECUs actually present in a vehicle at a given time may be much smaller than L . Given a subset $S_\ell \subseteq [L]$ of legitimate receivers, DBE enables one-to-many distribution of $M \in \mathbb{G}_T$ through a single broadcast.

The J1939-ADBE scheme consists of four phases: Setup, Session Key Distribution, Authenticated Message Transmission, and Join/Leave. We use ADBE as a building block that aims to provide the following guarantees:

- Confidentiality against unauthorized ECUs (DBE property);
- Publicly verifiable authentication of the sender identity (BLS property);
- Replay protection via a monotone group-level rekey counter $\text{ctr}_{\text{sig}}^{(\ell)}$ and per sender message counters $\text{ctr}_{\text{msg},i}^{(\ell)}$.

5.1.1. Setup Phase

The setup procedure, summarized in Algorithm 1, is executed once per vehicle during manufacturing by the OEM. The OEM first runs the ADBE Setup algorithm to obtain the global public parameters pp and generates the vehicle-wide seed seed . It then publishes pp as public system metadata on TBOX.BB (steps 1–2), assigns each initial high-performance ECU a unique slot identifier i , and provisions $(\text{pp}, \text{seed}, i)$ to ECU_i (step 4). Each ECU_i locally runs $\text{KeyGen}(\text{pp}, i)$ to derive its long-term key pair $(\text{usk}_i, \text{upk}_i)$ (step 5), while the OEM publishes the registration tuple $(\text{ID}_i, i, \text{upk}_i)$ to TBOX.BB (step 6).

Finally, the OEM fixes the logical groups $\mathcal{G}$ as part of the vehicle configuration, defining membership for each secure communication domain. After this phase, the ECU slot allocation, logical group configuration, and bulletin-board contents match the architecture in Figure 3 (step 8).

Algorithm 1 Setup phase

```

OEM:
1:  $\text{pp} \leftarrow \text{Setup}(1^\lambda, 1^L)$ 
2:  $\text{seed} \xleftarrow{\$} \{0, 1\}^k$ 
3:  $\text{TBOX.BB} \leftarrow \{\text{pp}\}$ 
4: for each high-performance  $\text{ECU}_i$  do
5:   assign slot identifier  $i$  and provision  $(\text{pp}, \text{seed}, i)$  to  $\text{ECU}_i$ 
 $\text{ECU}_i$ :
6:    $(\text{usk}_i, \text{upk}_i) \leftarrow \text{KeyGen}(\text{pp}, i)$ 
OEM:
7:    $\text{TBOX.BB} \leftarrow \text{TBOX.BB} \cup \{(\text{ID}_i, i, \text{upk}_i)\}$ 
8: end for
9: define logical groups  $\mathcal{G}$  according to OEM policy
10: for each logical group  $G_\ell \in \mathcal{G}$  do
11:   for each currently provisioned  $\text{ECU}_i \in G_\ell$  do
12:      $\widehat{\text{ctr}}_{\text{sig}}^{(\ell)} \leftarrow 0$ 
13:   end for
14: end for

```

In addition, for each logical group G_ℓ , all currently provisioned member ECUs initialize the latest accepted rekey counter $\widehat{\text{ctr}}_{\text{sig}}^{(\ell)} \leftarrow 0$ (step 12). This counter is used to ensure the freshness of key-update messages for logical group G_ℓ , and is therefore maintained at the

logical-group level rather than at the runtime receiver-set level. Since the actual receiver set S_ℓ may vary with the current vehicle composition, some ECUs that logically belong to G_ℓ may be temporarily absent from the current S_ℓ . Such ECUs do not participate in the initial runtime communication and synchronize $\widehat{\text{ctr}}_{\text{sig}}^{(\ell)}$ when they later join the vehicle and accept the first valid rekey message for that logical group.

5.1.2. Session Key Distribution Phase

At power-on, or whenever a fresh communication session is required, each logical group G_ℓ executes a session key distribution procedure, as summarized in Algorithm 2. For the current vehicle composition, G_ℓ is first resolved to a runtime receiver set S_ℓ through the slot bindings published on TBOX.BB. The actual key-update message is then generated and disseminated with respect to this current receiver set S_ℓ . Among the ECUs in S_ℓ , the one with the smallest valid slot acts as the leader and is solely responsible for initiating the key update for logical group G_ℓ . Since the receiver set is explicitly encoded in the bitmap carried by the update request, every receiver can verify whether the sender is indeed the designated leader of the current S_ℓ . Freshness of key-update requests is tracked by a monotone counter $\text{ctr}_{\text{sig}}^{(\ell)}$ associated with logical group G_ℓ , while each member stores the latest accepted value $\widehat{\text{ctr}}_{\text{sig}}^{(\ell)}$ and accepts a new key-update request only if it carries a strictly larger counter.

To initiate an update, the leader increments its signing counter $\text{ctr}_{\text{sig}}^{(\ell)}$ and samples a fresh $\delta_\ell \in \mathbb{Z}_p^*$ (steps 1–2). It then derives the session keys $(K_{\text{ses}}^{(\ell)}, K_{\text{sync}}^{(\ell)})$ from $(\text{seed}, \delta_\ell)$ via HMAC-based Key Derivation Function (HKDF) (step 3). Next, the leader encrypts δ_ℓ for the receiver set S_ℓ using ADBE, signs the resulting ciphertext together with the receiver-set bitmap and counter, and broadcasts the tuple to all members (steps 4–6).

Algorithm 2 Session key distribution phase in receiver set S_ℓ

```

ECUleader:
1:  $\text{ctr}_{\text{sig}}^{(\ell)} \leftarrow \widehat{\text{ctr}}_{\text{sig}}^{(\ell)} + 1$ 
2:  $\delta_\ell \xleftarrow{\$} \mathbb{Z}_p^*$ 
3:  $(K_{\text{ses}}^{(\ell)}, K_{\text{sync}}^{(\ell)}) \leftarrow \text{HKDF}(\text{seed}, \delta_\ell)$ 
4:  $\text{ct}_\ell \leftarrow \text{Encrypt}(\text{pp}, \{\text{upk}_j\}_{j \in S_\ell}, S_\ell, \delta_\ell)$ 
5:  $\sigma_\ell \leftarrow \text{Sign}(\text{pp}, \text{usk}_{\text{leader}}, \text{ct}_\ell \parallel \text{ctr}_{\text{sig}}^{(\ell)} \parallel S_\ell, \text{leader})$ 
6: Send  $(\text{ct}_\ell, \sigma_\ell, \text{leader}, S_\ell, \text{ctr}_{\text{sig}}^{(\ell)})$ 
ECUmember:
7: if  $\text{leader} = \min(S_\ell)$  and  $\text{Verify}(\text{pp}, \text{upk}_{\text{leader}}, \text{ct}_\ell \parallel \text{ctr}_{\text{sig}}^{(\ell)} \parallel S_\ell, \text{leader}, \sigma_\ell)$  and  $\text{ctr}_{\text{sig}}^{(\ell)} > \widehat{\text{ctr}}_{\text{sig}}^{(\ell)}$  then
8:    $\delta_\ell \leftarrow \text{Decrypt}(\text{pp}, \{\text{upk}_j\}_{j \in S_\ell}, \text{usk}_{\text{member}}, \text{ct}_\ell, S_\ell, \text{member})$ 
9:    $(K_{\text{ses}}^{(\ell)}, K_{\text{sync}}^{(\ell)}) \leftarrow \text{HKDF}(\text{seed}, \delta_\ell)$ 
10:   $\widehat{\text{ctr}}_{\text{sig}}^{(\ell)} \leftarrow \text{ctr}_{\text{sig}}^{(\ell)}$ 
11:  for  $i \in S_\ell$  do
12:     $\text{ctr}_{\text{msg}, i}^{(\ell)} \leftarrow 0, \widehat{\text{ctr}}_{\text{msg}, j \in S_\ell \setminus \{i\}}^{(i, \ell)} \leftarrow 0$ 
13:  end for
14: else
15:  return  $\perp$ 
16: end if

```

Upon reception, each member first verifies the leader's signature and checks both sender legitimacy and counter freshness before accepting the update (step 7). If all checks

hold, it decrypts ct_ℓ to recover δ_ℓ and re-derives $(K_{ses}^{(\ell)}, K_{sync}^{(\ell)})$ via the same HKDF procedure (steps 8–9). At this point, the member also synchronizes its local signing counter state (step 10) and reinitializes the ordinary message-counter state associated with S_ℓ (step 12). Otherwise, it rejects the message and returns $\perp$.

5.1.3. Authenticated Message Transmission Phase

After session keys are established, ECUs in each receiver set S_ℓ authenticate application data using the symmetric key $K_{ses}^{(\ell)}$, as summarized in Algorithm 3. For each receiver set S_ℓ , every ECU maintains one local message counter for its own outgoing traffic, denoted by $ctr_{msg, sender}^{(\ell)}$, and one locally stored latest accepted counter value for each other ECU in S_ℓ , denoted by $\widehat{ctr}_{msg, sender}^{(receiver, \ell)}$. These message-counter states are reinitialized whenever a fresh session key is established for S_ℓ .

For each outgoing frame from ECU_{sender} in receiver set S_ℓ , the sender first increments $ctr_{msg, sender}^{(\ell)}$ (step 1), computes an authenticator $\tau = \text{HMAC}_{K_{ses}^{(\ell)}}(\text{data} \parallel ctr_{msg, sender}^{(\ell)})$, and transmits $(ctr_{msg, sender}^{(\ell)}, \text{data}, \tau)$ (steps 2–3).

Upon reception, $ECU_{receiver}$ verifies the HMAC and checks the freshness of the received counter against $\widehat{ctr}_{msg, sender}^{(receiver, \ell)}$ (step 4). If both checks hold, it accepts the message and updates the stored counter value accordingly (step 5); otherwise, it rejects the message. The key $K_{ses}^{(\ell)}$ protects session data messages, whereas $K_{sync}^{(\ell)}$ is used for time-synchronization messages following the same HMAC-and-counter format and is omitted for brevity.

Algorithm 3 Authenticated message transmission in receiver set S_ℓ

```

 $ECU_{sender}$ :
1:  $ctr_{msg, sender}^{(\ell)} \leftarrow ctr_{msg, sender}^{(\ell)} + 1$ 
2:  $\tau \leftarrow \text{HMAC}_{K_{ses}^{(\ell)}}(\text{data} \parallel ctr_{msg, sender}^{(\ell)})$ 
3: Send  $(ctr_{msg, sender}^{(\ell)}, \text{data}, \tau)$ 
 $ECU_{receiver}$ :
4: if  $\text{Verify}_{\text{HMAC}_{K_{ses}^{(\ell)}}}(\tau, \text{data} \parallel ctr_{msg, sender}^{(\ell)})$  and  $ctr_{msg, sender}^{(\ell)} > \widehat{ctr}_{msg, sender}^{(receiver, \ell)}$  then
5:    $\widehat{ctr}_{msg, sender}^{(receiver, \ell)} \leftarrow ctr_{msg, sender}^{(\ell)}$ 
6:   Accept data
7: else
8:   return  $\perp$ 
9: end if

```

5.1.4. Join and Leave Phase

Commercial vehicles may experience dynamic ECU membership, especially when detachable subsystems such as trailers are involved. In our setting, Join/Leave is treated as a low-frequency maintenance event that affects subsequent authenticated in-vehicle communication. More specifically, the Join/Leave mechanism determines whether an ECU is admitted to subsequent session-key updates of the vehicle. Since vehicle-specific public parameters pp , seed $seed$, and the initial slot allocation are established by the OEM during manufacturing, the OEM already maintains the vehicle-level registration information required for ECU admission. Accordingly, the admission of a newly joining ECU requires one-time OEM authentication and registration for the target vehicle, whereas the subsequent key-update process is carried out within the vehicle by the local protocol. Meanwhile, each ECU generates and retains its own long-term secret key locally.

Join. The join procedure is summarized in Algorithm 4. When a new ECU intends to participate in protected in-vehicle communication, it first sends its activation information to

the OEM backend through the T-Box relay (Step 1). The OEM verifies the received activation information and, if the request is accepted, returns a newly assigned slot i together with the vehicle-specific seed (Step 2). After receiving these materials, ECU_{new} obtains the public parameters pp from TBOX.BB, runs $\text{KeyGen}(\text{pp}, i)$ to generate its long-term key pair $(\text{usk}_i, \text{upk}_i)$, and submits the registration tuple $\langle \text{ID}_{\text{new}}, i, \text{upk}_i \rangle$ to the OEM backend through the T-Box relay (Step 3). After validating this tuple, the OEM instructs the T-Box to append the corresponding registration record to TBOX.BB (Step 4).

Once TBOX.BB is updated, the vehicle performs a global rekeying operation across all logical groups. Existing ECUs then execute the Session Key Distribution Phase for the current runtime receiver set S_ℓ of each group G_ℓ . As a result, the new ECU can derive fresh session keys for those logical groups that include it.

Here, ECU_{new} denotes an ECU that does not yet possess a valid slot binding and public-key registration for the current vehicle. An ECU that is only temporarily unavailable and later recovers is not treated as a new ECU, since it already possesses legitimate long-term key material and vehicle-specific parameters for that vehicle. By contrast, an ECU moved from a different vehicle (e.g., a trailer ECU transferred from another tractor) is treated as a new ECU for the current vehicle, because the corresponding pp and seed are vehicle-specific and, therefore, inconsistent with those of the previous vehicle.

Algorithm 4 New ECU join in J1939-ADBE

```

 $\text{ECU}_{\text{new}}$ :
1: send activation information to the OEM backend via the T-Box relay
OEM backend:
2: verify the activation information and return  $(i, \text{seed})$ 
 $\text{ECU}_{\text{new}}$ :
3: obtain  $\text{pp}$  from TBOX.BB
4:  $(\text{usk}_i, \text{upk}_i) \leftarrow \text{KeyGen}(\text{pp}, i)$ 
5: send  $\langle \text{ID}_{\text{new}}, i, \text{upk}_i \rangle$  to the OEM backend via the T-Box relay
OEM backend:
6: validate the registration tuple and instruct the T-Box to update TBOX.BB
T-Box:
7:  $\text{TBOX.BB} \leftarrow \text{TBOX.BB} \cup \{ \langle \text{ID}_{\text{new}}, i, \text{upk}_i \rangle \}$ 
Existing ECUs:
8: for each logical group  $G_\ell$ , execute the Session Key Distribution Phase
  
```

Leave. When an ECU leaves the vehicle, either because it is physically detached or because it is logically excluded after compromise or maintenance events, it is removed from subsequent protected group communication by excluding it from future receiver sets so that it can no longer derive future session keys or generate valid authentication tags for protected communication. Since the bulletin board is append-only and serves only as a public registration mechanism for ECU admission, Leave does not require deleting or overwriting prior records on TBOX.BB.

6. Security Analysis of the Scheme

This section provides an integrated security analysis of J1939-ADBE under a standard Dolev–Yao adversary that controls the J1939 bus. We first give an informal, threat-oriented argument showing how the concrete mechanisms of the scheme mitigate various attack scenarios. We then model J1939-ADBE in Tamarin to verify injective agreement, session secrecy, known-key security, and forward secrecy, which together provide formal evidence for the security goals stated in Section 3.3.3.

6.1. Informal Analysis

We now discuss, at an informal level, how the J1939-ADBE workflow mitigates replay, forgery, MITM manipulation, cross-group substitution, desynchronization, and key-compromise scenarios.

We present the main security claims as propositions in the following, together with brief proof sketches that explain the core intuition. The full formal arguments for all propositions are deferred to Appendix A.

Adversarial model and cryptographic assumptions. We follow the adversary model in Section 3.3.2. The attacker has Dolev–Yao control over the J1939 bus, while HSM-protected long-term secrets, the per-vehicle seed, and freshness-critical local counter state are assumed not to be extractable unless an explicit compromise is stated in the corresponding proposition. We additionally assume that the append-only bulletin board TBOX.BB provides authentic public registration metadata, so that honest ECUs resolve each OEM-defined logical group G_ℓ to the current runtime receiver set S_ℓ using the published slot/public-key bindings. We assume that ADBE provides confidentiality and correct binding to the intended receiver set S_ℓ for the group update value δ_ℓ , signatures are EUF-CMA secure, HMAC is unforgeable as a MAC, and HKDF behaves as a pseudorandom function in its inputs.

Proposition 1. *Given the J1939-ADBE scheme, any attempt at a replay attack on either rekey or data frames will succeed with negligible probability.*

Replay resistance follows from the joint use of the monotone group-level rekey counter $\text{ctr}_{\text{sig}}^{(\ell)}$ and the per sender message counters $\text{ctr}_{\text{msg},i}^{(\ell)}$ together with signature and HMAC verification under the currently installed session keys. Honest ECUs reject any stale rekey whose counter is not strictly larger than the locally stored $\widehat{\text{ctr}}_{\text{sig}}^{(\ell)}$ and reject any stale data frame whose sender counter is not strictly larger than $\widehat{\text{ctr}}_{\text{msg},\text{sender}}^{(\text{receiver},\ell)}$. Any stale message carrying an old counter value is rejected by honest ECUs.

The full proof is given in Appendix A.1. This proposition supports Security Goal (1) in Section 3.3.3 by preventing acceptance of stale rekey states and rollback via replay.

Proposition 2. *Given the J1939-ADBE scheme, any attempt of message forgery or ECU impersonation without possession of valid private keys will succeed with negligible probability.*

An accepted rekey must verify under the public key published on TBOX.BB for the claimed sender slot, and the sender must match the designated leader $\min(S_\ell)$ of the current receiver set. The signature binds the ciphertext, rekey counter, and receiver-set bitmap, so neither a bus adversary nor a non-leader ECU can create a fresh accepted rekey without the relevant private key. Moreover, a rogue ECU that has not completed OEM-backed admission cannot cause honest nodes to treat it as a legitimate participant because it will not appear as a valid slot/public-key binding on TBOX.BB and, therefore, will not be resolved into any authorized runtime receiver set. Application data frames are authenticated using HMAC under a current session key derived from the per-vehicle seed and the fresh group update value δ_ℓ . An adversary lacking the required session key cannot compute a valid fresh tag. The detailed argument appears in Appendix A.2. This proposition supports Security Goal (1) by ensuring that only legitimate ECUs can produce acceptable rekey updates, and supports Security Goal (4) by preventing unauthorized generation of valid MACs for authenticated data frames.

Proposition 3. *Given the J1939-ADBE scheme, any MITM attack attempting to alter, reorder, or downgrade rekey frames will succeed with negligible probability.*

Bit-level modification of rekey frames is detected by signature verification, while replay or downgrade attempts using older, validly signed rekeys are rejected due to the strict monotonicity requirement on the group-level rekey counter. Consequently, a network adversary cannot induce ECUs to accept an outdated or adversarially modified session state. The complete proof is provided in Appendix A.3.

This proposition supports Security Goal (1) by ruling out adversarial manipulation that would drive ECUs into incorrect or stale session states.

Proposition 4. *Given the J1939-ADBE scheme, cross-group substitution and key leakage across groups will succeed with negligible probability.*

Each ADBE ciphertext is cryptographically bound to a specific receiver set S_ℓ and is accepted only if accompanied by a valid signature explicitly covering S_ℓ . ECUs outside the intended group cannot decrypt the ciphertext to recover δ_ℓ . As a result, ciphertext replay or forwarding across groups does not lead to key reuse or leakage. A complete proof is given in Appendix A.4. This proposition supports Security Goal (1) by preventing cross-group misuse/substitution and by preserving the confidentiality of group update values from unauthorized nodes outside the target group.

Proposition 5. *Given the J1939-ADBE scheme, the compromise of one session key does not reveal any other past or future session keys.*

Session keys are independently derived in each rekey round from the per-vehicle seed and a freshly sampled update value δ_ℓ . Under the pseudorandomness of HKDF, knowledge of one derived key does not enable computation of keys associated with different update values. The complete proof is given in Appendix A.5. This proposition supports Security Goal (2) by establishing cryptographic separation across rekey rounds, even when previous-round session keys are exposed.

Proposition 6. *Given the J1939-ADBE scheme, past session keys remain confidential under subsequent compromise of a single key component (either an ECU's long-term private key or the per-vehicle seed). Recovering past session keys from recorded transcripts requires the joint compromise of both components.*

Reconstruction of a past session key requires knowledge of both the per-vehicle seed and the corresponding group update value δ_ℓ . Compromise of only one component is insufficient to recompute historical keys from recorded transcripts. This establishes forward secrecy under single-component compromise. The stronger setting in which an adversary can extract *both* key components from the same ECU (contrary to Section 3.3.2) is not covered by this guarantee. The full argument appears in Appendix A.6. This proposition supports Security Goal (3) under the compromise model stated in Section 3.3.2.

Proposition 7. *Given the J1939-ADBE scheme, the system maintains robustness against desynchronization and secure join/leave operations.*

Because session keys are not chained across rekey rounds, an ECU can resynchronize upon receiving any fresh rekey message without reconstructing intermediate states. A joining ECU becomes effective only after OEM-backed registration is appended to TBOX.BB and the vehicle executes the subsequent global rekey across all logical groups so that the

joining ECU can derive fresh keys only for those future runtime receiver sets that include it. A leaving ECU is excluded by omitting it from the updated runtime receiver sets used in the next rekey. The associated rekey-counter synchronization and message-counter reinitialization ensure that honest ECUs converge to the latest accepted epoch without accepting stale traffic from a pre-change state. The same reasoning also applies if an ECU reboots with outdated or temporarily inconsistent local counter state: since both the rekey counter and the message counters are monotone, the ECU can safely resynchronize by accepting the next valid rekey carrying a fresher counter, while stale traffic is rejected by the corresponding freshness checks. A complete proof is given in Appendix A.7. This proposition supports Security Goal (1) by ensuring that, after Join/Leave and state changes, ECUs can converge to the latest valid session state without accepting stale updates.

Propositions 1 to 7 establish that J1939-ADBE has the ability to resist typical adversarial behaviors in vehicular networks, and collectively cover Security Goals (1)–(4) in Section 3.3.3.

6.2. Formal Security Verification

We investigate J1939-ADBE in a symbolic model using the Tamarin Prover. We follow the adversary model in Section 3.3.2; in Tamarin, this is instantiated as a standard Dolev–Yao attacker controlling the J1939 bus, with idealized cryptographic primitives and non-extractable HSM-protected secrets unless explicitly revealed by the model.

Our model abstracts three entities (ECU1, ECU2, ECU3) and two communication groups (Group1, Group2), and it captures two key-distribution procedures: the initial distribution (`init`) and a generic rollover (`next`). Any subsequent rollover is isomorphic to `next`; together, these two procedures cover the full lifecycle of the rekeying logic from deployment to an unbounded number of updates. For tool alignment, we introduce lightweight equations for encryption/decryption correctness and instrument the scheme with `Running/Commit` events that record the relevant parameters of each rekey execution. These modeling choices affect only the representation, not the semantics, and follow the scheme description in Section 5.

To provide formal evidence for the updated Security Goals (1)–(4) in Section 3.3.3, we verify the following standard security properties in Tamarin, using the property definitions and specification methodology in the Tamarin manual [33] and the authentication hierarchy of Lowe [34]:

- **Injective agreement** captures strong authentication agreement for rekeying runs: each successful acceptance of a rekey is linked to a unique peer execution with matching session parameters (e.g., sender identity, counter, and receiver-set context). This supports Security Goal (1) by ruling out forged/tampered rekey acceptance, rollback, and cross-group confusion when the modeled events bind the receiver-set context.
- **Session secrecy** states that the attacker cannot derive the established session keys (or declared secret session material) from the protocol execution trace. This directly supports the confidentiality part of Security Goal (2).
- **Known-key security** states that compromise of one session key does not enable derivation of other session keys from different rekey rounds/sessions, capturing key separation across updates. This directly supports the previous-round key exposure part of Security Goal (2).
- **Forward secrecy** states that, under the compromise events modeled in this paper, the attacker still cannot recover session keys used before the compromise from recorded traces. This directly supports Security Goal (3).

In addition, we check a freshness property for the signing counter: each accepted rekey must carry a counter strictly larger than the receiver’s local value, which is consistent

with Security Goal (1) and prevents acceptance of stale session states. The above standard properties therefore provide formal evidence for Security Goals (1)–(3); Security Goal (4) is addressed by the informal unforgeability argument in Proposition 2.

We encode J1939-ADBE in Tamarin and verify it against the above security goals under unbounded concurrency. To put these guarantees into context, we compare them with the security properties reported in the literature for J1939-ACAN [1] and CL-KIMS [19]. Table 1 summarizes this comparison: all three schemes provide injective agreement, session secrecy, and known-key security, while forward secrecy is not achieved by J1939-ACAN but is satisfied by CL-KIMS and J1939-ADBE under the respective key-compromise models.

Table 1. Security property comparison of J1939 authentication schemes.

Property	J1939-ACAN [1]	J1939-CL-KIMS [19]	J1939-ADBE
Injective agreement	✓	✓	✓
Session secrecy	✓	✓	✓
Known-key security	✓	✓	✓
Forward secrecy	×	✓	✓

✓: the property is achieved; ×: the property is not supported.

Under unbounded concurrency and the stated assumptions, Tamarin proves all targeted properties for the modeled subset of J1939-ADBE. The complete J1939-ADBE Tamarin script is available at <https://github.com/Kachun-QLJG/J1939-ADBE-tamarin-script> (accessed on 6 April 2026).

7. Performance Evaluation

This section compares the proposed ADBE-based broadcast scheme (J1939-ADBE) with two representative point-to-point designs (J1939-CL-KIMS and J1939-ACAN) along four dimensions: complexity, message cost, computational cost, and total message delay (TMD) on a CAN-FD bus. Let L denote the size of the preconfigured slot space exposed by the vehicle network and n denote the number of ECUs. Let $\mathcal{G} = \{G_1, \dots, G_{|\mathcal{G}|}\}$ denote the set of OEM-defined logical groups over authorized ECU identities. For a given vehicle composition, each logical group G_ℓ is resolved through TBOX.BB to a runtime receiver set $S_\ell \subseteq [L]$. For each scheme, we first derive closed-form expressions for payload size and frame count under 64-byte CAN-FD payloads, and then combine these with measured computation times to obtain reproducible side-by-side comparisons. The results show that J1939-ADBE completes distribution with a single broadcast whose frame count is nearly constant for realistic slot-space size L (only a small linear term from the receiver set bitmap); once TMD is defined as the sum of cryptographic computation time and on-bus propagation time, this broadcast structure yields a significantly lower overall delay.

Benchmark platform. All cryptographic timing results in this section were obtained on a Raspberry Pi running Linux 6.6.51+rpt-rpi-2712 (aarch64, Debian build dated 8 October 2024), equipped with a quad-core ARM Cortex-A76 CPU (1.5–2.4 GHz) and 8 GiB RAM. This aarch64/ARM Cortex-A76 platform is used to approximate high performance ECUs that are built on ARM-based application processors and are capable of running a full OS/runtime stack, which is a common design choice for higher-end in-vehicle computing nodes such as in-vehicle infotainment, T-Box, and gateway devices. These measurements therefore reflect an upper-end ECU class where ADBE is computationally feasible and secure storage/HSM support can be provisioned to protect long-term secrets, and provide a practical basis for evaluating the feasibility and relative overhead of the proposed design on such platforms.

7.1. Complexity Analysis

Per-group rekey (one logical group). In J1939-ADBE, refreshing the group keys for a target logical group G_ℓ over its current runtime receiver set S_ℓ requires producing a single ADBE ciphertext (together with a publicly verifiable BLS signature) and broadcasting it once on the bus. Upon receiving this broadcast, every ECU in S_ℓ can derive the new group keys locally; no per-member point-to-point sessions are needed. Therefore, the round complexity of one group rekey is $\mathcal{O}(1)$.

Vehicle-wide rekey across all logical groups. During a maintenance window, multiple logical groups may be refreshed. For each logical group G_ℓ , its current runtime receiver set S_ℓ is first determined from TBOX.BB, and the corresponding rekey broadcast is then issued for that receiver set. Rekeys for different groups are independent and can be triggered in parallel by their respective initiators. On a single CAN-FD segment, broadcasts from different groups still contend for bus access and are serialized by arbitration; thus the completion time of a full rekey epoch is dominated by the number of group broadcasts that must be delivered. Since each group needs exactly one broadcast, the bus-level traffic complexity of refreshing all groups once scales as $\mathcal{O}(|\mathcal{G}|)$.

In contrast, in J1939-CL-KIMS and J1939-ACAN, each ECU establishes an individual point-to-point session with a central SeCU to obtain (or refresh) the session secret. Hence, refreshing keys of all ECUs once requires $\mathcal{O}(n)$ exchanges. The overall asymptotic behavior of J1939-ADBE and the two baseline schemes is summarized in Table 2.

Table 2. Asymptotic cost of one vehicle-wide session-key distribution.

Scheme	Communication Cost	SeCU Required
J1939-ACAN [1]	$\mathcal{O}(n)$	Yes
J1939-CL-KIMS [19]	$\mathcal{O}(n)$	Yes
J1939-ADBE	$\mathcal{O}(\mathcal{G})$	No

n : the number of ECUs; $|\mathcal{G}|$: the number of ECU logical groups.

7.2. Message Count and Payload Size (CAN-FD, 64-Byte Payload)

A single J1939-ADBE broadcast $(\text{ct}_\ell, \sigma_\ell, \text{leader}, S_\ell, \text{ctr}_{\text{sig}}^{(\ell)})$ carries the following fields: a ciphertext ct_ℓ (composed of 3 group elements, 90 B each), a BLS signature σ_ℓ (90 B), a leader identifier leader (2 B), and a runtime receiver-set bitmap S_ℓ of length $\lceil L/8 \rceil$ bytes, together with a group-level rekey counter $\text{ctr}_{\text{sig}}^{(\ell)}$ (8 B). The total payload length is therefore $|P_{\text{ADBE}}(n)| = 370 + \lceil L/8 \rceil$ bytes. Under a 64-byte payload per CAN-FD frame, the resulting frame count is $N_{\text{ADBE}}(n) = \left\lceil \frac{370 + \lceil L/8 \rceil}{64} \right\rceil$. For $L \in (0, 112]$ and $L \in [113, 624]$, this evaluates to six and seven frames, so one J1939-ADBE broadcast suffices to complete distribution to the current receiver set and the dependence on L is only through the small bitmap term.

For J1939-CL-KIMS, the ECU $\rightarrow$ SeCU request carries the elliptic-curve public key point and its elliptic curve digital signature, while the SeCU $\rightarrow$ ECU response carries the SeCU's elliptic-curve public key point, a symmetric ciphertext of the session secrets (under the elliptic-curve Diffie–Hellman derived key), and a signature. When mapped to 64-byte CAN-FD payloads, this two-step exchange requires approximately four frames per ECU (about six frames per ECU if a time-synchronization exchange is also included), yielding $N_{\text{CL-KIMS}}(n) = 4n$ (or $6n$).

For J1939-ACAN, each ECU initialization and SeCU response include Diffie–Hellman or elliptic-curve cryptography public-key material, nonces, and signatures/verification data. Under the same 64-byte payload assumption this corresponds to approximately seven frames per ECU: $N_{\text{ACAN}}(n) = 7n$.

In summary, J1939-ADBE achieves a frame count that is essentially independent of n , whereas both point-to-point designs grow strictly linearly with the number of ECUs.

7.3. Computational Cost

Since different OEMs may partition ECUs into logical groups of different sizes, the practical cost of J1939-ADBE depends on both the preconfigured slot-space size L and the runtime receiver-set size $|S_\ell|$. Therefore, this experiment is designed to verify that, under different but realistic $(L, |S_\ell|)$ configurations, the computation overhead of ADBE remains acceptable, so that J1939-ADBE can adapt to different OEM grouping strategies in practice. For this purpose, we benchmark one complete ADBE run over a grid of $(L, |S|)$ combinations, including encryption, signing, verification, and decryption. For each setting, the system parameters and user keys are generated accordingly, a receiver set S of size $|S|$ is randomly selected from the slot space $[L]$, and the sender and receiver are chosen within S . Each test is repeated 10 times, and the average total time is summarized as a heatmap in Figure 4.

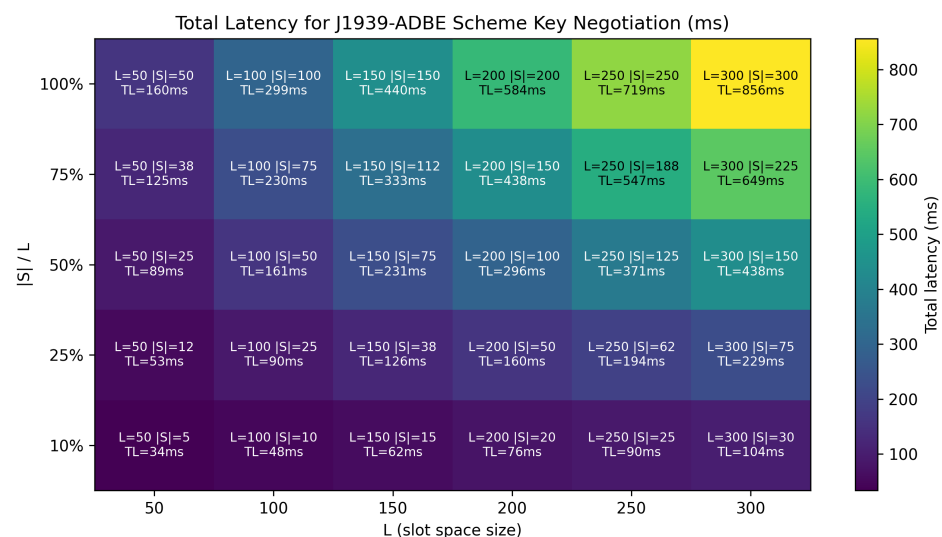


Figure 4. Computation-time heatmap of J1939-ADBE over $(L, |S|)$.

The measurements show that the total cost is determined mainly by the runtime receiver-set size $|S|$, whereas the influence of the slot-space size L is marginal. For example, when $|S| = 50$, the computational cost changes only from about 89 ms at $L = 50$ to about 90 ms at $L = 250$; similarly, when $|S| = 75$, it changes from about 230 ms at $L = 150$ to about 229 ms at $L = 300$. In contrast, increasing $|S|$ leads to a clear rise in total cost, and this growth mainly comes from the encryption phase, while signing, verification, and decryption remain nearly constant or increase only mildly. This result is practically important because it means that an OEM can pre-allocate a relatively large slot space L to reserve room for future ECU joins and leaves without introducing a significant increase in the computation time of a single J1939-ADBE broadcast.

Additionally, even in the worst evaluated case for a single-group rekey, namely $L = 300$ and $|S_\ell| = 300$, one complete ADBE run still finishes in about 856 ms on the target platform. Since this setting corresponds to an extreme deployment in which all 300 occupied slots are simultaneously grouped into one logical set, the result indicates that the intrinsic computation overhead of J1939-ADBE remains practically acceptable for rekey operations performed at vehicle startup or after security-relevant membership changes, and that the scheme remains deployable across different OEM grouping strategies.

7.4. Total Message Delay on a CAN-FD Bus

To compare the end-to-end completion time on the bus, we define the total message delay (TMD) of one vehicle-wide session-key distribution as the sum of cryptographic computation time and on-bus propagation time. Since J1939-CL-KIMS and J1939-ACAN do not introduce logical grouping, a fair whole-vehicle comparison is conducted by considering a single group that covers all ECUs, i.e., $|S_\ell| = n$. Under this setting, J1939-ADBE completes one distribution by a single broadcast to the full receiver set, whereas J1939-CL-KIMS and J1939-ACAN must establish n point-to-point sessions to reach all ECUs. If multiple logical groups are present in a real deployment, all schemes would need to repeat the distribution procedure for each group; therefore, the $|S_\ell| = n$ case serves as a common baseline for comparing their per-distribution efficiency.

Let T_{comp} denote the overall computation time to complete a full distribution under this baseline (one broadcast for J1939-ADBE; n sessions for J1939-CL-KIMS and J1939-ACAN), and let N_{frames} be the number of CAN-FD frames required. On a single bus with bitrate R and 64-byte payload per frame, the propagation time is modeled as $T_{\text{comm}} = \frac{N_{\text{frames}} \times 512}{R}$, where 512 is the number of payload bits per frame. The total delay is then $T_{\text{total}} = T_{\text{comp}} + T_{\text{comm}}$.

For J1939-ADBE, we substitute $N_{\text{frames}} = N_{\text{ADBE}}(n)$ from the payload analysis above, and use the measured computation time under $|S| = L$. For J1939-CL-KIMS and J1939-ACAN, we use $N_{\text{CL-KIMS}}(n)$ and $N_{\text{ACAN}}(n)$, respectively, and accumulate their per session computation and communication costs over all n point-to-point sessions. Combining these frame-count models with the measured computation times yields the TMD curves at $R \in \{250, 500\}$ kbps for $|S_\ell| = n$, as depicted in Figure 5a,b.

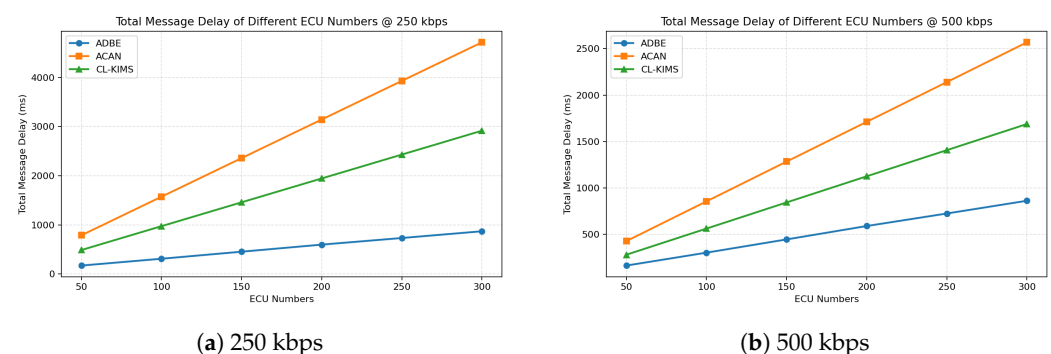


Figure 5. TMD comparison for $|S_\ell| = n$: J1939-ADBE vs. J1939-CL-KIMS and J1939-ACAN.

As n increases, the frame-induced propagation term grows linearly for the point-to-point designs and gradually dominates T_{total} . This is because both J1939-CL-KIMS and J1939-ACAN require an independent session for each ECU, so both the frame count and the accumulated computation cost scale with n . In contrast, under the single-group vehicle-wide baseline used for comparison, J1939-ADBE requires only one broadcast covering the full ECU set in the vehicle-wide case and thus keeps N_{frames} nearly constant, so its TMD grows much more slowly. Across the evaluated ranges of n and R , J1939-ADBE consistently achieves the lowest total delay for vehicle-wide session-key distribution.

All experiments in Section 7 evaluate one complete rekey operation for a single logical group. This setting is adopted to enable a fair comparison with existing schemes at the same distribution granularity, while avoiding additional variables introduced by different logical grouping strategies; at the same time, it is also consistent with the design of J1939-ADBE, which performs key distribution and update on a per-group basis. Furthermore, when rekeying across multiple logical groups in the vehicle is considered, rekey operations in the proposed scheme are initiated by the corresponding group leaders rather

than being uniformly scheduled by a single SeCU. Therefore, in multi-group scenarios, the rekeying process does not need to be centrally serialized by one node, but instead admits independent and potentially parallel progression across different groups. In addition, according to the foregoing payload analysis, one complete rekey operation for a single logical group requires only about 6–7 CAN-FD frames. Hence, even if multiple logical groups trigger rekeying within a similar time window, the resulting increase in bus communication overhead remains limited. By contrast, if existing SeCU-based schemes are extended to support logical-group-granular rekeying, the SeCU must still handle the distribution process for each group, making it more likely to incur additional processing and communication overhead in multi-group scenarios. Taken together, the above results indicate that J1939-ADBE achieves good efficiency and practical feasibility for logical-group-oriented key distribution.

8. Discussion

8.1. Deployment-Side Threats and Security Boundary

The main objective of J1939-ADBE is to remove the need for a dedicated in-vehicle SeCU that centrally holds and distributes current keying material during runtime. To achieve this goal, the scheme relies on several engineering assumptions that are compatible with current commercial-vehicle deployments. In particular, the OEM performs manufacturing-time provisioning and one-time admission control for newly joining ECUs; the T-Box hosts an append-only bulletin board and provides authenticated, integrity-protected connectivity to the OEM backend for ECU admission and other maintenance operations; and ECUs locally protect their long-term secret keys, the vehicle seed, and the freshness-critical state via an HSM or equivalent secure hardware support. Under these assumptions, the maintenance plane supports provisioning and membership changes, whereas subsequent session-key updates and protected communication are carried out locally within the vehicle.

T-Box failure or bulletin-board unavailability. If the T-Box fails or the bulletin board becomes temporarily unavailable, already admitted ECUs can continue operating with a locally cached copy of the bulletin board. Since the bulletin board stores only public registration data and is append-only, previously established receiver-set resolution remains usable for existing ECUs. Therefore, the immediate impact is mainly on future maintenance operations, such as admission of new ECUs or registration changes, rather than on the continuity of already established protected communication.

Bulletin-board pollution. If unauthorized entries are appended to the bulletin board, they do not modify prior valid registrations and do not directly expose existing key material. The main consequence is in future admission consistency and configuration management. For example, invalid appended entries may consume available slot identifiers and prevent new ECUs from joining until OEM-authorized maintenance reconstructs the registration state. In this case, the deployed vehicle network can remain operational, although future membership changes may be blocked.

OEM disconnection or Join-path relay attack. If connectivity to the OEM backend is lost, or if the relay path used during Join is attacked, the primary impact is on admission of future ECUs rather than on the operation of the existing in-vehicle network. Under the assumption that the admission channel is authenticated and integrity protected end-to-end between the requesting ECU and the OEM backend, an adversary on the relay path may delay or block admission, but should not be able to alter the assigned slot, the returned seed, or the resulting registration outcome without detection. Consequently, compromise of the T-Box or its relay functionality mainly leads to denial of service for newly joining

ECUs, while established protected communication among existing ECUs continues to rely on locally held secrets and subsequent in-vehicle rekeying.

8.2. Robustness Against Compromised or Malicious ECUs

The detection of a compromised or malicious ECU is outside the scope of J1939-ADBE itself and should be provided by complementary mechanisms such as intrusion detection, attestation, or maintenance diagnostics. Once such an ECU is identified, the OEM or vehicle maintenance workflow can remove it from the subsequent runtime receiver set S_ℓ of the affected logical group and then trigger a fresh rekey. Because the new session key is derived from a fresh update value and distributed only to the updated receiver set, the excluded ECU can no longer generate valid fresh authenticated traffic for that group, and any subsequent injection or replay attempt under the old session state will be rejected. In this sense, J1939-ADBE provides an efficient post-detection isolation mechanism for compromised ECUs.

At the same time, the protection offered here should be understood as damage containment rather than standalone compromise detection. Before a compromised ECU is detected and excluded, an already authorized ECU that still belongs to the current receiver set may continue to produce authenticated traffic within the groups for which it legitimately holds the current session key; similarly, if it currently acts as the designated leader of a receiver set, it may still issue syntactically valid rekeys for that set until the compromise is discovered. Therefore, the robustness of J1939-ADBE against malicious ECUs comes from limiting the security impact to the compromised ECU's currently authorized groups and enabling fast exclusion through receiver-set updates and rekeying, rather than from independently identifying malicious behavior.

8.3. Applicability to Passenger Vehicles

We focus on SAE J1939 in this work because it is an open and standardized higher-layer vehicular protocol, which makes its message organization and supporting mechanisms explicit enough for rigorous system modeling, protocol design, and performance evaluation in a public study. Moreover, SAE J1939 targets commercial vehicles in a relatively general manner rather than a single OEM platform. By contrast, application-layer protocols in passenger vehicles are often OEM-specific. On the one hand, the publicly available background knowledge is usually limited; on the other hand, even a concrete design for one platform may only apply to a particular OEM. Therefore, this subsection does not attempt a one-to-one instantiation for passenger vehicles, but instead discusses whether the underlying design idea of our scheme can be transferred to such environments.

The core idea of the proposed scheme is not tied to SAE J1939 itself. Rather, it is to exploit the broadcast nature of the CAN bus and use an authenticated broadcast-encryption mechanism to disseminate session-key update material. This design has several appealing properties. First, it does not rely on a centralized in-vehicle SeCU that continuously holds all current ECU session keys, and is therefore more consistent with the principle of minimizing key exposure. Second, each rekey operation requires only one authenticated broadcast to deliver update material to the authorized receiver set, which significantly reduces the point-to-point communication overhead of key updates. Third, the protection is applied to higher-layer security-sensitive services, instead of uniformly processing all underlying CAN/CAN-FD frames. In this sense, the design idea is conceptually transferable to passenger-vehicle environments.

At the operational level, the main entities involved in our scheme are also not unique to SAE J1939-based commercial vehicles. The roles played by ECUs, an admission-assisting T-Box relay, an OEM-backed registration service, and a bulletin board for key-binding

information can all be realized within passenger-vehicle platforms. Likewise, the basic workflow of authenticated admission, receiver-set formation, broadcast rekeying, and local session-key derivation does not inherently depend on J1939-specific semantics. What changes in passenger vehicles is mainly how these roles and messages are mapped to OEM-specific application-layer formats and deployment architectures.

Based on the above considerations, we believe that the proposed approach can also serve as a useful design pattern for application-layer protocols in passenger vehicles. A concrete migration, however, would still require adaptation to the corresponding OEM-specific protocol formats, system architectures, and deployment constraints. We hope that this work can provide a useful reference point for such extensions and contribute to future instantiation and validation in passenger-vehicle settings.

9. Conclusions

This work addresses scalable key management for SAE J1939 in commercial vehicles by introducing the ADBE as an authenticated DBE construction and instantiating it as J1939-ADBE, a decentralized authentication and group key-distribution scheme. ADBE combines DBE-style broadcast encryption with BLS-based public verifiability, enabling any authorized ECU to broadcast fresh group updates without relying on a centralized key server. J1939-ADBE derives per-group session and synchronization keys from a vehicle-wide seed and fresh update values, enforces freshness via counters, and authenticates protected data frames with HMAC. We prove that ADBE preserves DBE confidentiality while achieving EUF-CMA authentication, and Tamarin verification shows that J1939-ADBE satisfies injective agreement, session secrecy, known-key security, and forward secrecy under our threat model. Finally, our evaluation indicates that using a single broadcast per rekey yields a frame count largely independent of ECU scale, reducing total latency and improving scalability compared with SeCU-based baselines.

Author Contributions: Conceptualization, Y.L.; methodology, J.X. and Y.L.; software, J.X.; validation, J.X.; formal analysis, J.X. and J.L.; investigation, J.S.; resources, Y.L.; writing—original draft preparation, J.X.; writing—review and editing, J.X., J.L. and J.S.; visualization, J.X.; supervision, J.S.; project administration, J.L.; funding acquisition, Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Major Science and Technology Special Projects in Henan Province (No. 241100240300), the Natural Science Foundation of Jiangsu Province (No. BK20230134), and the National Key Research and Development Program of China (No. 2023YFB2504800).

Data Availability Statement: The Tamarin Prover model and proof script for J1939-ADBE is publicly available at <https://github.com/Kachun-QLJG/J1939-ADBE-tamarin-script> (accessed on 6 April 2026). Other supporting scripts used in this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Proof of Propositions in Section 6

Appendix A.1. Proof of Proposition 1

The following is the proof of proposition 1.

Proof. For a rekey of logical group G_ℓ , each honest receiver accepts the broadcast only if the signature verifies and $\text{ctr}_{\text{sig}}^{(\ell)} > \widehat{\text{ctr}}_{\text{sig}}^{(\ell)}$. Hence, a replayed rekey frame is rejected after the receiver has already accepted a fresher epoch for that logical group. This remains true even when the runtime receiver set S_ℓ changes after a Join/Leave event, because the

fresh post-change rekey is tied to the updated receiver set and supersedes all previously accepted ones.

For authenticated data transmission in receiver set S_ℓ , $\text{ECU}_{\text{receiver}}$ stores $\widehat{\text{ctr}}_{\text{msg, sender}}^{(\text{receiver}, \ell)}$ for each sender. A replayed data frame may still carry a syntactically valid tag with respect to an old session state, but it is rejected whenever $\text{ctr}_{\text{msg, sender}}^{(\ell)} \leq \widehat{\text{ctr}}_{\text{msg, sender}}^{(\text{receiver}, \ell)}$. Moreover, when a fresh rekey is accepted, the message-counter state associated with S_ℓ is reinitialized, so stale data frames from the previous epoch also fail under the newly installed session key. Therefore, both the rekey channel and the authenticated data channel are robust against replay. $\square$

Appendix A.2. Proof of Proposition 2

The following is the proof of proposition 2.

Proof. A rekey message for runtime receiver set S_ℓ is accepted only if two conditions hold simultaneously. First, the signature verifies under the public key published on TBOX.BB for the claimed sender slot. Second, the claimed sender must equal the designated leader $\text{min}(S_\ell)$ of the current receiver set. Because the signed payload explicitly binds $(\text{ct}_\ell \parallel \text{ctr}_{\text{sig}}^{(\ell)} \parallel S_\ell)$, an adversary who does not hold the relevant private key cannot create a fresh accepted rekey, nor can it modify an existing one without invalidating the signature. Likewise, a non-leader ECU with its own valid private key still cannot impersonate the designated leader, because honest receivers recompute the valid leader from the signed receiver-set bitmap and reject any sender that is not $\text{min}(S_\ell)$.

A newly introduced ECU must also complete OEM-backed admission before it can affect protected communication. Until its slot/public-key registration tuple has been appended to TBOX.BB, honest ECUs will not resolve it into any authorized runtime receiver set, so it cannot become an accepted sender or receiver in the rekey workflow.

Authenticated data frames are protected by HMAC under the current session key $K_{\text{ses}}^{(\ell)}$, where $(K_{\text{ses}}^{(\ell)}, K_{\text{sync}}^{(\ell)}) \leftarrow \text{HKDF}(\text{seed}, \delta_\ell)$. Without knowledge of the current session key, the adversary cannot compute a valid fresh MAC for a protected message. Since seed is HSM-protected and δ_ℓ is recoverable only from the ADBE ciphertext by an ECU in the intended receiver set, an external attacker or unauthorized ECU cannot forge a fresh authenticated data frame. $\square$

Appendix A.3. Proof of Proposition 3

The following is the proof of proposition 3.

Proof. The adversary may arbitrarily intercept, drop, or reorder rekey frames on the bus, but cannot forge signatures. Each accepted rekey is signed over $(\text{ct}_\ell \parallel \text{ctr}_{\text{sig}}^{(\ell)} \parallel S_\ell)$, so any bit-level modification of the ciphertext, any relabeling of the receiver-set bitmap, or any alteration of the sender identity that is inconsistent with the published public-key binding causes verification to fail.

A downgrade or reordering attack relies on injecting an older rekey that still carries a valid signature. However, every receiver enforces the strict monotonicity condition $\text{ctr}_{\text{sig}}^{(\ell)} > \widehat{\text{ctr}}_{\text{sig}}^{(\ell)}$. Once a newer rekey has been accepted, any older one is rejected as stale and cannot revert the session key to an earlier epoch. Furthermore, because the signed message fixes the current receiver set S_ℓ , an attacker cannot move a valid ciphertext into another receiver-set context and have it interpreted as a valid rekey there. Therefore, MITM attempts to alter, reorder, or downgrade rekey frames cannot lead to an accepted but compromised state. $\square$

Appendix A.4. Proof of Proposition 4

The following is the proof of proposition 4.

Proof. For each logical group G_ℓ , honest ECUs first resolve the current runtime receiver set S_ℓ from TBOX.BB and then invoke ADBE encryption as $ct_\ell \leftarrow \text{Encrypt}(pp, \{upk_j\}_{j \in S_\ell}, S_\ell, \delta_\ell)$, while a member ECU decrypts using $\delta_\ell \leftarrow \text{Decrypt}(pp, \{upk_j\}_{j \in S_\ell}, usk_{\text{member}}, ct_\ell, S_\ell, \text{member})$. Both algorithms are parameterized by the intended receiver set S_ℓ , and only ECUs that are actually included in that set can recover δ_ℓ . Consequently, an ECU outside S_ℓ cannot derive the corresponding $(K_{\text{ses}}^{(\ell)}, K_{\text{sync}}^{(\ell)})$.

This also rules out unauthorized receiver-set inclusion through stale registration information. Although TBOX.BB is append-only, a leaving ECU that is no longer part of the current runtime receiver set is simply omitted from the signed bitmap S_ℓ used in the next rekey. Historical registration tuples, therefore, do not grant decryption capability for future epochs.

Moreover, rekey acceptance requires a valid signature on $(ct_\ell \parallel \text{ctr}_{\text{sig}}^{(\ell)} \parallel S_\ell)$. Even if a ciphertext is replayed or forwarded across groups, the verification step enforces agreement on the specific receiver set encoded in the signed message, so cross-group substitution does not result in a valid rekey for an unintended group. Consequently, the confidentiality of each receiver set's fresh update value is preserved, and key material does not leak across groups via ciphertext mixing. $\square$

Appendix A.5. Proof of Proposition 5

The following is the proof of proposition 5.

Proof. Each session-key pair for logical group G_ℓ and rekey epoch t is derived as $(K_{\text{ses}}^{(\ell,t)}, K_{\text{sync}}^{(\ell,t)}) \leftarrow \text{HKDF}(\text{seed}, \delta_{\ell,t})$, where $\delta_{\ell,t}$ is freshly sampled from $\mathbb{Z}_p^*$ for that epoch. Under the assumption that HKDF behaves as a secure pseudorandom function, the outputs corresponding to distinct inputs $(\text{seed}, \delta_{\ell,t})$ are computationally independent. Thus, even if an adversary learns $(K_{\text{ses}}^{(\ell,t)}, K_{\text{sync}}^{(\ell,t)})$ for one specific epoch, this does not help it compute the keys for any other epoch $t' \neq t$ of the same logical group.

The same argument applies across different receiver sets, because distinct rekeys use independently sampled update values and, therefore, induce independent HKDF inputs even when they share the same vehicle-wide seed. Hence, the compromise of one session key does not reveal other past or future session keys. $\square$

Appendix A.6. Proof of Proposition 6

The following is the proof of proposition 6.

Proof. Each session-key pair $(K_{\text{ses}}^{(\ell,t)}, K_{\text{sync}}^{(\ell,t)})$ is derived as $(K_{\text{ses}}^{(\ell,t)}, K_{\text{sync}}^{(\ell,t)}) \leftarrow \text{HKDF}(\text{seed}, \delta_{\ell,t})$, where seed is the per-vehicle secret stored inside HSM-protected memory, and $\delta_{\ell,t}$ is the fresh update value distributed by ADBE to the intended receiver set in epoch t . Suppose that the adversary compromises an ECU's long-term private key usk_i after passively recording all bus traffic, while the per-vehicle seed remains protected. With usk_i , the adversary may decrypt past ADBE ciphertexts for receiver sets that include ECU_{*i*} and thereby recover the corresponding $\delta_{\ell,t}$ values, but it still cannot evaluate $\text{HKDF}(\text{seed}, \delta_{\ell,t})$ without knowing seed. By the pseudorandomness of HKDF, the past session keys remain hidden.

Conversely, suppose the adversary somehow compromises the per-vehicle seed seed but does not possess a valid long-term private key for a member of the target receiver set. Then, by the security of ADBE encryption, it cannot recover the associated $\delta_{\ell,t}$ values from recorded ciphertexts and, therefore, cannot reconstruct $(K_{\text{ses}}^{(\ell,t)}, K_{\text{sync}}^{(\ell,t)})$.

Hence, under the forward-secrecy model where at most one of the two key components (the per-vehicle seed or an ECU long-term private key) is compromised after the fact, past session keys remain confidential. Only a joint compromise of both components, which corresponds to a full break of the protected storage assumptions for a group member, would allow reconstruction of historical keys from transcripts. $\square$

Appendix A.7. Proof of Proposition 7

The following is the proof of proposition 7.

Proof. The key schedule does not chain round-to-round in a way that requires knowledge of intermediate keys: each $(K_{\text{ses}}^{(\ell,t)}, K_{\text{sync}}^{(\ell,t)})$ depends only on $(\text{seed}, \delta_{\ell,t})$ for that epoch. An ECU that misses one or more rekeys, therefore, remains desynchronized only temporarily and can resynchronize as soon as it receives any later rekey that passes signature verification and satisfies $\text{ctr}_{\text{sig}}^{(\ell)} > \widehat{\text{ctr}}_{\text{sig}}^{(\ell)}$. By decrypting ct_{ℓ} to obtain $\delta_{\ell,t}$ and applying HKDF with its locally stored seed, the ECU derives the current session keys without reconstructing intermediate ones.

For Join, a newly introduced ECU first completes OEM-backed admission via the T-Box relay, receives the vehicle-specific slot assignment and seed, locally generates its own long-term key pair from the public parameter pp , and has its registration tuple appended to TBOX.BB . It does not obtain any protected communication ability merely from this registration step. Rather, it becomes an effective member only when the vehicle executes the subsequent global rekey across all logical groups. At that time, the new ECU accepts the first valid rekey only for those authorized runtime receiver sets that now include it and derives only the fresh current session keys for those sets; it does not learn any earlier $\delta_{\ell,t'}$ values or the corresponding past session keys. Its local rekey-counter state is synchronized on acceptance of that first valid rekey, and its ordinary message-counter state is initialized for the new epoch.

For Leave, no deletion from the append-only bulletin board is required. The leaving ECU is simply omitted from future runtime receiver sets, and the next rekey for each affected logical group is generated with a signed bitmap that excludes it. Since the leaving ECU cannot decrypt the fresh $\delta_{\ell,t}$ for those future rekeys, it cannot derive future session keys or produce fresh valid authentication tags for protected traffic. Because honest ECUs also advance to the new rekey-counter state, stale pre-change traffic is rejected after the membership change. Therefore, the scheme remains robust against desynchronization while supporting secure Join/Leave through rekey-based admission and exclusion. $\square$

References

1. Murvay, P.S.; Groza, B. Security shortcomings and countermeasures for the SAE J1939 commercial vehicle bus protocol. *IEEE Trans. Veh. Technol.* **2018**, *67*, 4325–4339. [\[CrossRef\]](#)
2. Burakova, Y.; Hass, B.; Millar, L.; Weimerskirch, A. Truck hacking: An experimental analysis of the SAE j1939 standard. In Proceedings of the 10th USENIX Workshop on Offensive Technologies (WOOT 16), Austin, TX, USA, 8–9 August 2016.
3. Mueller, A.; Lothspeich, T. Plug-and-secure communication for CAN. *CAN Newsl.* **2015**, *4*, 10–14.
4. Jain, S.; Guajardo, J. Physical layer group key agreement for automotive controller area networks. In *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 85–105.
5. Groza, B.; Popa, L.; Murvay, P.S. Tricks—Time triggered covert key sharing for controller area networks. *IEEE Access* **2019**, *7*, 104294–104307. [\[CrossRef\]](#)
6. Jain, S.; Wang, Q.; Arafat, M.T.; Guajardo, J. Probing attacks on physical layer key agreement for automotive controller area networks. In *Proceedings of the 2018 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*; IEEE: Hong Kong, China, 2018; pp. 7–12.
7. Groza, B.; Murvay, S.; Van Herrewege, A.; Verbauwhede, I. LiBrA-CAN: A lightweight broadcast authentication protocol for controller area networks. In *Proceedings of the Cryptology and Network Security: 11th International Conference, CANS 2012, Darmstadt, Germany, 12–14 December 2012*; Proceedings 11; Springer: Berlin/Heidelberg, Germany, 2012; pp. 185–200.

8. Van Herrewege, A.; Singelee, D.; Verbauwhede, I. CANAuth—a simple, backward compatible broadcast authentication protocol for CAN bus. In *Proceedings of the ECRYPT Workshop on Lightweight Cryptography*; ECRYPT: Louvain-la-Neuve, Belgium, 2011; Volume 2011, p. 20.
9. Szilagyi, C.; Koopman, P. Low cost multicast authentication via validity voting in time-triggered embedded control networks. In *Proceedings of the 5th Workshop on Embedded Systems Security*; ACM: New York, NY, USA, 2010; pp. 1–10.
10. Youn, T.Y.; Lee, Y.; Woo, S. Practical sender authentication scheme for in-vehicle CAN with efficient key management. *IEEE Access* **2020**, *8*, 86836–86849. [[CrossRef](#)]
11. Yu, D.; Hsu, R.H.; Lee, J.; Lee, S. EC-SVC: Secure CAN bus in-vehicle communications with fine-grained access control based on edge computing. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 1388–1403. [[CrossRef](#)]
12. Groza, B.; Murvay, P.S. Identity-based key exchange on in-vehicle networks: CAN-FD & FlexRay. *Sensors* **2019**, *19*, 4919.
13. Sun, H.; Luo, W.; Weng, J.; Liu, Z.; Li, M. ECQV-GDH-Based Group Key Exchange Protocol for CAN Bus. *IEEE Trans. Veh. Technol.* **2023**, *72*, 12857–12872. [[CrossRef](#)]
14. Hartkopp, O.; Reuber, C.; Schilling, R. MaCAN—message authenticated CAN. In *Proceedings of the 10th International Conference on Embedded Security in Cars (ESCAR 2012)*, Berlin, Germany, 28–29 November 2012.
15. Nürnberger, S.; Rossow, C. –vatican–vetted, authenticated can bus. In *Proceedings of the Cryptographic Hardware and Embedded Systems—CHES 2016: 18th International Conference, Santa Barbara, CA, USA, 17–19 August 2016*; Proceedings 18; Springer: Berlin/Heidelberg, Germany, 2016; pp. 106–124.
16. Pfeiffer, O. *Implementing Scalable Can Security with Cancrypt*; Embedded Systems Academy: San Jose, CA, USA, 2017.
17. Cui, J.; Chen, Y.; Zhong, H.; He, D.; Wei, L.; Bolodurina, I.; Liu, L. Lightweight encryption and authentication for controller area network of autonomous vehicles. *IEEE Trans. Veh. Technol.* **2023**, *72*, 14756–14770. [[CrossRef](#)]
18. Bruni, A.; Sojka, M.; Nielson, F.; Riis Nielson, H. Formal security analysis of the MaCAN protocol. In *Proceedings of the Integrated Formal Methods: 11th International Conference, IFM 2014, Bertinoro, Italy, 9–11 September 2014*; Proceedings 11; Springer: Cham, Switzerland, 2014; pp. 241–255.
19. Palaniswamy, B.; Ansari, K.; Reddy, A.G.; Das, A.K.; Shetty, S. Robust certificateless authentication protocol for the SAE J1939 commercial vehicles bus. *IEEE Trans. Veh. Technol.* **2022**, *72*, 4493–4509. [[CrossRef](#)]
20. Koscher, K.; Czeskis, A.; Roesner, F.; Patel, S.; Kohno, T.; Checkoway, S.; McCoy, D.; Kantor, B.; Anderson, D.; Shacham, H.; et al. Experimental security analysis of a modern automobile. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*; IEEE: Oakland, CA, USA, 2010; pp. 447–462.
21. Checkoway, S.; McCoy, D.; Kantor, B.; Anderson, D.; Shacham, H.; Savage, S.; Koscher, K.; Czeskis, A.; Roesner, F.; Kohno, T. Comprehensive experimental analyses of automotive attack surfaces. In *Proceedings of the 20th USENIX Security Symposium (USENIX Security 11)*, San Francisco, CA, USA, 10–12 August 2011.
22. Woo, S.; Jo, H.J.; Lee, D.H. A practical wireless attack on the connected car and security protocol for in-vehicle CAN. *IEEE Trans. Intell. Transp. Syst.* **2014**, *16*, 993–1006. [[CrossRef](#)]
23. Luo, Q.; Cao, Y.; Liu, J.; Benslimane, A. Localization and navigation in autonomous driving: Threats and countermeasures. *IEEE Wirel. Commun.* **2019**, *26*, 38–45. [[CrossRef](#)]
24. Nie, S.; Liu, L.; Du, Y. Free-fall: Hacking tesla from wireless to can bus. *Brief. Black Hat USA* **2017**, *25*, 16.
25. Nie, S.; Liu, L.; Du, Y.; Zhang, W. Over-the-air: How we remotely compromised the gateway, BCM, and autopilot ECUs of Tesla cars. *Brief. Black Hat USA* **2018**, *91*, 1–19.
26. Miller, C.; Valasek, C. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA* **2015**, *2015*, 1–91.
27. Benadjila, R.; Renard, M.; Lopes-Esteves, J.; Kismi, C. One car, two frames: Attacks on hitag-2 remote keyless entry systems revisited. In *Proceedings of the 11th USENIX Workshop on Offensive Technologies (WOOT 17)*, Vancouver, BC, Canada, 14–15 August 2017.
28. Garcia, F.D.; Oswald, D.; Kasper, T.; Pavlidès, P. Lock it and still lose it—On the (In)Security of automotive remote keyless entry systems. In *Proceedings of the 25th USENIX Security Symposium (USENIX Security 16)*, Austin, TX, USA, 10–12 August 2016.
29. Mukherjee, S.; Shirazi, H.; Ray, I.; Daily, J.; Gamble, R. Practical DoS attacks on embedded networks in commercial vehicles. In *Proceedings of the Information Systems Security: 12th International Conference, ICISS 2016, Jaipur, India, 16–20 December 2016*; Proceedings 12; Springer: Cham, Switzerland, 2016; pp. 23–42.
30. Stabili, D.; Ferretti, L.; Marchetti, M. Analyses of secure automotive communication protocols and their impact on vehicles life-cycle. In *Proceedings of the 2018 IEEE International Conference on Smart Computing (SMARTCOMP)*; IEEE: Taormina, Italy, 2018; pp. 452–457.
31. Kolonelos, D.; Malavolta, G.; Wee, H. Distributed broadcast encryption from bilinear groups. In *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*; Springer: Singapore, 2023; pp. 407–441.
32. Boneh, D.; Lynn, B.; Shacham, H. Short signatures from the Weil pairing. In *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 514–532.

33. The Tamarin Team. Tamarin Prover Manual: Property Specification. Available online: https://tamarin-prover.com/manual/master/book/007_property-specification.html (accessed on 5 March 2026).
34. Lowe, G. A hierarchy of authentication specifications. In *Proceedings of the Proceedings 10th Computer Security Foundations Workshop*; IEEE: Rockport, MA, USA, 1997; pp. 31–43.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.