Journal of Information Systems Engineering and Management
2025, 10(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

LiREAP: A Lightweight Robust Encryption and Authentication Protocol for Securing In-Vehicle Communication

Mahmoud A. AttaAlla[1]*, Mohamed T. Ali[2], and Ahmed M. Gawish[3]

[1,2]Arab Open University Arab Open University, Egypt. (e-mail:mahmoud.attalah@aou.edu.eg; S2051810045EG@std.aou.edu.eg)
[3]Arab Open University, HQ, Kuwait. (e-mail: a.gawish@arabou.edu.kw)

ARTICLE INFO

Received: 14 Nov 2024
Revised: 27 Dec 2024
Accepted: 12 Jan 2025

ABSTRACT

Controller Area Network (CAN) is a serial communication protocol for distributed real-time control systems. Its high reliability and low cost enable it to be widely implemented in various domains such as automotive, avionics, and maritime networks. However, security mechanisms were not implemented during the design of the CAN protocol, which resulted in many attacks such as CAN injection. These attacks enable attackers to steal vehicles, change their states, and gain remote access to their control systems. Literature review reports that current CAN's security protocols experienced lack of security robustness or efficiency due to the constrained environment of the CAN. This paper introduces A Lightweight Robust Encryption and Authentication Protocol (LiREAP), that addresses security vulnerabilities in the CAN protocol while preserving its performance. The proposed protocol adopts Ascon, the upcoming NIST standard for lightweight cryptography, to achieve both confidentiality and integrity for CAN data. A hash chain mechanism is implemented for session key generation. In addition, a challenge-response two-factor authentication mechanism (CR2AM) is designed for session key distribution. The proposed protocol is designed to work harmonically with the standard CAN protocol that enables only 8 bytes for the data field. The conducted theoretical analysis of the proposed protocol reports its ability to provide robust security level with minimum overhead that meet the high-speed requirements in-vehicle communication. Compared to the current protocols, the proposed one efficiently fill the security gap of the CAN with very low overhead cost.

Index Terms—Ascon, Lightweight Cryptography, Controller area network (CAN), In-vehicle communication, Automotive Security.

1. INTRODUCTION

CONTROLLER Area Network (CAN) is a widely used protocol for connecting in-vehicle electronics [1]. However, the inherited lack of security mechanisms in
CAN networks has exposed them to various cyber threats, including message interception, manipulation, and unauthorized access.

Research conducted by Koscher et al. [2] illuminated the vulnerabilities within automotive electronic systems, revealing the potential for complete unauthorized control via accessible interfaces. This concern was also showcased, when Miller and Valasek [3] demonstrated a remote control over an unmodified Jeep Cherokee by exploiting a vulnerable ECU that let them to remotely gain access to the CAN bus. In consequence, Chrysler Corporation recalled approximately 1.4 million vehicles.

Furthermore, in 2019, the U.S. Cybersecurity and Infrastructure Security Agency (CISA) issued an alert highlighting the vulnerable implementations of CAN bus networks in aircraft [4]. This alert emphasized the critical consequences of potential attacks, as malicious actors gaining physical access to the CAN bus could manipulate instrument readings and alter the status of an aircraft posing risks such as loss of control. Furthermore, there have been numerous passenger vehicle theft incidents, including instances involving Toyota and Lexus vehicles, perpetrated via CAN injection attacks [5] [6] [7] [8].

In response to these challenges, various efforts have been undertaken to address vulnerabilities in the CAN bus and mitigate these attacks. These initiatives included the implementation of intrusion detection systems [9] [10] [11],
deployment of secure gateways [12] [13] [14], and integration

of cryptography [15] [16] [17] [18]. This work follows the approach of utilizing cryptography to address the security posture of the CAN bus.

This paper proposes a Lightweight Robust Encryption and Authentication Protocol (LiREAP) that aims at securing the CAN data while maintaining the requirements of high-speed communication. Designed with efficiency and robustness in mind, LiREAP integrates state-of-the-art cryptographic techniques and secure key management mechanisms to safeguard CAN networks against various cyber threats, including CAN injection, forgery attacks, and reverse engineering. It consists of three components: The Key Management Component, the Authenticated Encryption Component, and the Counter Management Component. These components work harmonically to ensure confidentiality, integrity, and authenticity for CAN messages. It utilizes Ascon, the upcoming standard for lightweight cryptography by the National Institute of Standards and Technology (NIST) [19]. Ascon's efficiency makes it well-suited for securing constrained environments such as the CAN bus, where reducing computational load is crucial. Moreover, LiREAP incorporates robust mechanisms for a centralized session key management, including a hash chain mechanism for session key generation and a challenge-response two-factor authentication mechanism (CR2AM) for session key distribution.

The rest of the paper is organized as follows: Section 2 addresses the previous work and the related scientific background of the concepts adopted in our proposed solution. Section 3 provides a comprehensive explanation and the technical details of the LiREAP, while Section 4 discusses the results and findings of the conducted security and overhead analysis. Finally, section 5 concludes the paper and highlights the prospected future work.

I. Related work and scientific background

This section reviews the previous works and discusses their findings and drawbacks. It also provides a brief explanation for the scientific concepts related to the proposed protocol.

A. Related Work

Previous works proposing cryptographic protocols to secure CAN networks have utilized either single or multiple keys for encryption and authentication. A protocol based on a single key is vulnerable to a single point of failure, meaning the entire protocol is compromised if the secret key is compromised. On the other hand, using multiple keys enhances robustness but could increase the overhead of key management.

The Automotive Open System Architecture (AUTOSAR) released the Secure Onboard Communication (SecOC) Protocol, a framework designed to secure in-vehicle communication, ensuring data integrity and authenticity [15]. The protocol relies on AES-CMAC, adding a Message Authentication Code (MAC) and a freshness value to data transmitted between Electronic Control Units (ECUs). However, researchers were able to break the protocol after successfully extracting the key, demonstrating a vulnerability due to its dependence on a single secret key [20].

Lu et al. [16] proposed a lightweight encryption and authentication protocol (LEAP) to secure CAN while maintaining its high performance. A high-performance ECU, called the Secure ECU, is responsible for the key management. It uses Advanced Encryption Standard (AES) and Secure Hash Algorithm (SHA) in session key generation and distribution, following an Encrypt-then-MAC scheme. The stream cipher RC4 is used for both encrypting and authenticating CAN messages between ECUs. Each ECU possesses a long-term key stored during manufacturing and a symmetric session key assigned to each group of ECUs. Session keys are updated and distributed regularly at vehicle's idle time. A counter is required to synchronize the RC4 key stream of each message. However, the protocol does not mention a mechanism for counter synchronization recovery, which could lead to failure due to network issues or a denial-of-service (DoS) attack. Additionally, the number of keys stored in each ECU is directly related to the number of ECUs in the CAN bus, increasing the key management overhead.

J. Cui et al. [17] proposed a lightweight encryption and authentication protocol for CAN in autonomous vehicles. The protocol uses the Grain-128a variant of Grain cipher for message encryption, keyed-hash message authentication code (HMAC) for message authentication, and Blom scheme for session key generation and distribution. However, the function used for hashing is not specified. A high-performance ECU is set as the key ECU (KECU) and is responsible for session key

Journal of Information Systems Engineering and Management
2025, 10(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/
Research Article

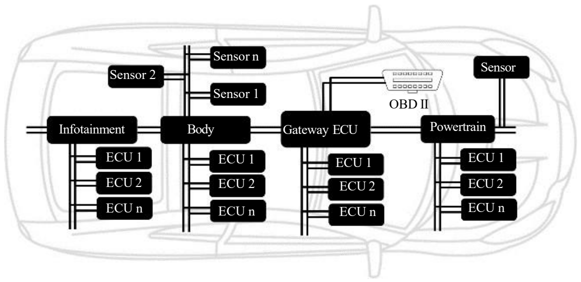

Fig. 1. In-vehicle CAN bus connecting ECUs.

distribution and updating. The protocol assigns pairwise keys for each pair of ECUs in addition to a key for each ECU to securely communicate with the (KECU). Hence, in networks where up to 100 ECUs are connected to the CAN bus, there are up to 100 keys stored in each ECU, which could lead to key management and latency issues. A sender ECU calculates the MAC of the message after encrypting it (Encrypt-then-MAC).
The MAC value is truncated into 4 bytes then divided
into two pairs. The first 2 bytes of the CAN frame are placed in the extended identifier field and the other 2 bytes replace the CRC field. It is worth nothing that according to the CAN specification [1], framing CAN messages is the responsibility of the Transfer layer, which has no freedom for modifications. Also, during the implementation of the protocol proposed by Woo et al. [21], which previously utilized a similar technique in generating secure CAN frames, it was claimed that it is not possible to modify the extended identifier and CRC fields in microcontrollers like DSP-F28335. Consequently, a software- based evaluation was added to the hardware-based evaluation to prepare the implementation experiment. Hence, replacing the CRC field with the authentication bytes is not applicable.
Wiemer and Zeh [22] proposed the CANsec protocol that employed the Ascon cipher to encrypt and authenticate CAN XL messages. Their work investigated the potential of Ascon to have an advantage on AES-GCM in nonce-misuse setting. However, they did not propose a solution to secure Classic CAN frames, which has a highly constrained data field of only 8 bytes [1], unlike CAN XL, which permits data fields up to 2048 bytes [23].

B. Scientific Background

This part addresses two main components in our work: The Controller Area Network and the Ascon Cipher.

Controller Area Network

The Controller Area Network (CAN) protocol was designed by Robert Bosch GmbH and officially released in 1986 [1]. It was developed to meet the requirements of high-speed networks for automotive in-vehicle communications. Due to its high reliability, low cost and simplicity, its applications have extended to avionics [24] and maritime networks [25]. CAN connects engine control unit, infotainment system, anti-lock braking system (ABS), sensors, and other in-vehicle electronics with bitrates up to 1 Mbit/s [1]. Additionally, it is used to communicate onboard diagnostics through an OBD-II port, which offers real-time sensor data and diagnostic trouble codes (DTCs) of various in-vehicle systems. Fig. 1 shows an

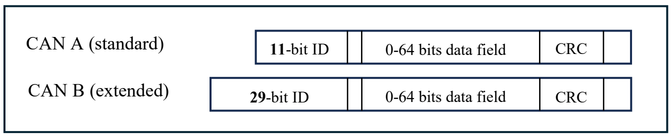

Fig. 2. CAN A and CAN B message formats.

overview of how different electronic control units (ECUs) are connected via CAN bus inside an automotive vehicle.
CAN is divided into different layers: object layer, transfer layer, and physical layer. Object and transfer layers perform the functions of the data link layer specified by the ISO/OSI model. The object layer is mainly responsible for message filtering, while the transfer layer is the core of the CAN protocol and is responsible for framing messages, error
detection, and transfer timing. The physical layer is responsible for the actual data transfer in the wires [1].
The standard CAN message format consists of 7 fields: 1-bit start-of-frame field, 11-bit arbitration field, 6-bit control field, 0 to 64-bit data filed, 16-bit cyclic redundancy check (CRC) field, 2-bit acknowledge (ACK) field,

and 7-bit end-of-frame field. An extended CAN message format was introduced by Robert Bosch GmbH in 1991 to provide a larger message address range for more applications by extending the 11-bit identifier of the arbitration field to 29 bits [1]. Fig. 2 shows the main fields in both standard and extended CAN message formats.
CAN provides flexibility in adding new nodes to the bus without requiring any modifications to the software or hardware of the nodes. It also provides flexibility in routing messages, in which any node can decide whether to receive a message or not based on the identifier in the arbitration field. Nodes in the CAN bus can transmit, receive, or monitor the bus by comparing the bit levels detected in the bus with the bit levels of the message to be transmitted. A bitwise arbitration process is used to prevent collisions that might occur when two nodes attempt to send a message to the bus at the same time. During the arbitration process, message identifiers are used to decide which message has the highest priority to be sent to the bus first. CAN also provides measures for the safety of data being transferred including error detection and fault confinement. In addition, receiver nodes can acknowledge a message that is received without errors.

Ascon Cipher

Ascon is a cipher suite that contains lightweight schemes for authenticated encryption with associated data (AEAD) and hashing. Ascon was selected in the final portfolio of the CAESAR competition (2014-2019) as the primary choice for lightweight authenticated encryption [26]. It was also selected as the winner of the NIST lightweight cryptography (LWC) competition (2019-2023) and is considered for standardization [19].
All schemes in Ascon are based on the Ascon permutation. The Ascon permutation p is a substitution permutation network (SPN) consisting of bitwise Boolean functions and rotations. It operates on a 320-bit state, and it consists of three main steps: adding round constant, substitution layer, and linear diffusion layer; denoted as pC, pS, pL respectively. The 320-bit state S is represented with 5 64-bit words: x_0, x_1, x_2, x_3, x_4. The state is divided into the rate r which represents the data block size and the constant c which represents the rest of the state. Therefore, we can represent the state as follows: $S = Sr \; || \; Sc$

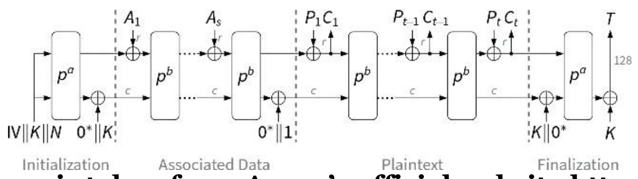

Fig. 3. Ascon AEAD. Figure is taken from Ascon's official website https://ascon.iaik.tugraz.at/

Ascon AEAD is a nonce-based encryption with a mode of operation founded on the duplex construction [27]. Its primary variants, *Ascon-128* and *Ascon-128a*, both operate with a key size of 128 bits [28]. Ascon is inverse-free, meaning it does not require inverse operations for decryption. Both encryption and decryption are evaluated in one direction and have four main phases: Initialization, Processing Associated Data, Processing Plaintext/Ciphertext, and Finalization (Fig. 3). In the Initialization phase, the state is formed by a 64-bit initialization vector (IV), a 128-bit key, and a 128-bit nonce (N). The IV is a constant value that specifies the algorithm as follows:
$IV = k \; || \; r \; || \; a \; || \; b \; || \; 0^{160-k}$, such that
k denotes the key size.
r denotes the rate.
a denotes the number of initial and final permutation rounds.
b denotes the number of intermediate permutation rounds.
0^{160-k} denotes a bitstring of 0s of the size of $160 - k$ bits.
After the state is formed, it is transformed by an initial permutation pa then XORed with the key. Next, the associated data (AD) is divided into blocks, and the last block is padded if its length is less than the required block size. Padding is done by adding 1 and the smallest number of 0s. Then, AD blocks are XORed with the rate r (i.e. absorbed). An intermediate permutation pb is performed after each AD block absorption. After that, the state is XORed with a 1-bit constant for domain separation. Then, the plaintext is also divided into blocks, and the last block is padded if its length is less than the required block size. Then, plaintext blocks are XORed with the rate r (i.e. absorbed). An intermediate permutation pb is performed after each plaintext block absorption, except for the last block. The outputs of the XOR operation are the ciphertext blocks (i.e. squeezed). In the Finalization phase, the state is XORed with the key, then transformed by a final permutation pa. Finally, the state is XORed again with the key to produce a 128-bit tag T, used for message authentication.
The double-keyed initialization and finalization improve Ascon security robustness, which is the ability to reduce damage in case of implementation errors [28]. After the Finalization stage of the decryption process, if

Journal of Information Systems Engineering and Management
2025, 10(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

the calculated tag is not equal to the tag obtained by the sender, decryption will not be successful, and a verification failed error shall be raised.

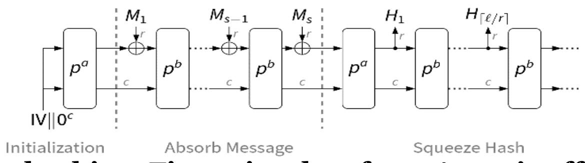

Fig. 4. Ascon hashing. Figure is taken from Ascon's official website https://ascon.iaik.tugraz.at/

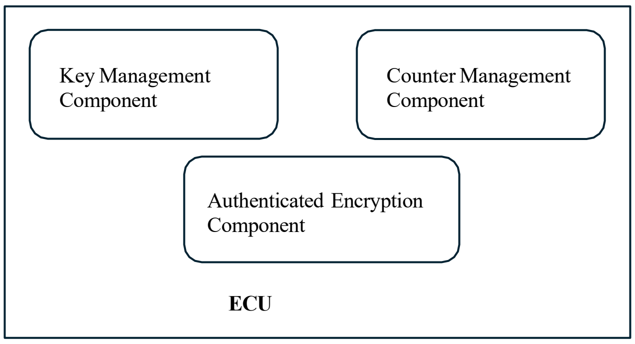
Key Management Component
Counter Management Component
Authenticated Encryption Component
ECU

Fig. 5. LiREAP components present in each secured ECU.

Ascon cipher also includes a sponge-based hashing algorithm for both fixed output and extendible output functions (XOF) [28]. As illustrated in Fig. 4, Ascon hashing includes three phases: Initialization, Message Absorption, and Hash Squeezing. In the initialization phase, the 320-bit state is formed by a 64-bit IV and a 256-bit of 0s. The IV is a constant value that specifies the algorithm as follows:
$IV = 0^8 \,||\, r \,||\, a \,||\, a - b \,||\, h$, such that
0^8 denotes a bitstring of 0s of the size 8.
r denotes the rate.
a denotes the number of initial and final permutation rounds.
b denotes the number of intermediate permutation rounds.
h denotes the hash output length.
After the state is formed, it is transformed by an initial permutation pa. Next, the message is divided into blocks, and the last block is padded if its length is less than the required block size. Then message blocks are XORed with the rate r (absorbed) and an intermediate permutation pb is performed after each message block absorption except for the last block. Then, an intermediate permutation pb is performed in the beginning of the Hash Squeezing phase and after each hash block squeeze until the required hash size is obtained.

II. PROPOSED PROTOCOL

This paper proposes A Lightweight Robust Encryption and Authentication Protocol (LiREAP) that aims at ensuring confidentiality, integrity, and authenticity for CAN data while meeting the high-speed and constrained environment requirements of the standard CAN protocol. Each Electronic Control Unit (ECU) has three components: Key Management Component, Authenticated Encryption Component, and Counter Management Component. These components, as shown in Fig. 5, are working harmonically to achieve the security objectives of the protocol as outlined in Table I.
Among the ECUs, one is designated by the manufacturer as the

TABLE I REQUIREMENTS FULFILLMENT SPECIFICATION.

Requirement	**Protocol Component**	**Mechanism / Primitive**
Confidentiality and Integrity	Authenticated Encryption Component	Ascon-128
Key Generation	Key Management Component	Hash Chain Mechanism, Ascon-XOF
Key Distribution	Key Management Component	Challenge-Response Two-factor Authentication Mechanism (CR2AM)
Counter Management	Counter Management Component	Monotonic Counter

Journal of Information Systems Engineering and Management
2025, 10(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

MASTER_KEY = 16_BYTE_RANDOM_NUMBER

SUBKEY = XOF_16 (MASTER_KEY)

FIRST_SESSION_KEY = XOF_16 (MASTER_KEY ⊕ SUBKEY)

NEXT_SESSION_KEY = XOF_16 (FIRST_SESSION_KEY ⊕ SUBKEY)

Fig. 6. Hash chain mechanism.

Master ECU (MECU), tasked with session key management and counter re-synchronization. It is advisable for the MECU to be situated in a physically secure location within the vehicle, inaccessible to unauthorized individuals.

A. Key Management Component

Key Management Component is responsible for key storage in all ECUs, and for session key generation and distribution in the MECU. Each ECU, including the MECU, possesses a secret key known as ECU_KEY, securely stored during manufacturing. This key is uniform across all ECUs and is only used for the session key distribution process. Therefore, each ECU maintains two secret keys: ECU_KEY and SESSION_KEY, regardless of the number of ECUs in the CAN bus.

The MECU employs a hash chain mechanism, illustrated in Fig. 6, to generate session keys. Initially, a secret key called MASTER_KEY, securely stored in the MECU during manufacturing, is used to generate a sub-key called SUBKEY. The SUBKEY is generated by calculating the 16-byte output of Ascon-XOF of the MASTER_KEY. The first session key is derived by calculating the 16-byte output of Ascon-XOF of the MASTER_KEY XORed with the SUBKEY. The first session key is also stored in all ECUs during manufacturing. Subsequent session keys are generated by calculating the 16- byte output of Ascon-XOF of the SUBKEY XORed with the previous session key. Session keys are regularly updated and distributed by the MECU at predefined intervals. One viable approach is to update the session keys upon each vehicle ignition.

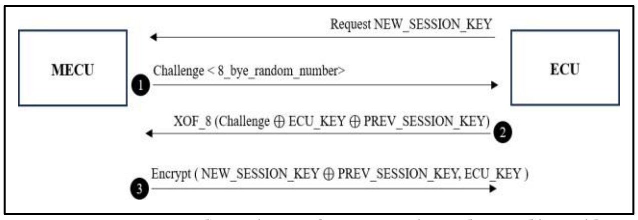
MECU
ECU
Request NEW_SESSION_KEY
1 Challenge < 8_bye_random_number>
XOF_8 (Challenge ⊕ ECU_KEY ⊕ PREV_SESSION_KEY) 2
3 Encrypt (NEW_SESSION_KEY ⊕ PREV_SESSION_KEY, ECU_KEY)

Fig. 7. CR2AM mechanism for session key distribution.

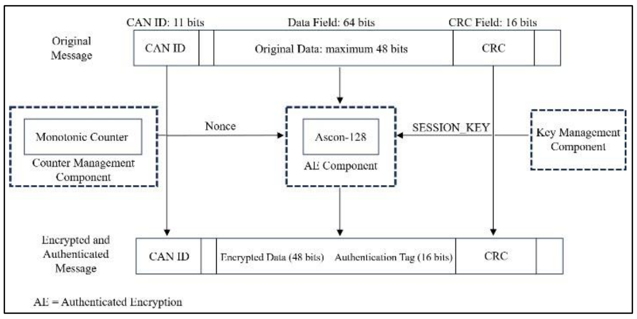

Fig. 8. Counter recovery mechanism.

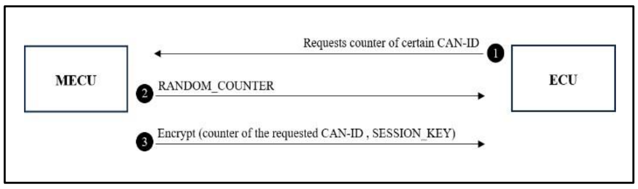
MECU
ECU
Requests counter of certain CAN-ID 1
2 RANDOM_COUNTER
3 Encrypt (counter of the requested CAN-ID , SESSION_KEY)

Fig. 9. Authenticated Encryption Component operating with other components of LiREAP.

To securely distribute the SESSION_KEY to other ECUs, the MECU employs a challenge-response two-factor authentication mechanism (CR2AM), as illustrated in Fig. 7. During CR2AM, an ECU requests the new session

key from the MECU, which responds with a random 8-byte challenge. The ECU computes the 8-byte output of Ascon-XOF for the challenge XORed with the ECU_KEY and the SESSION_KEY of the previous session, sending the response back to the MECU. Upon receiving the response, the MECU performs the same computation to verify the ECU response. Upon successful verification, the MECU sends the result of XORing the new SESSION_KEY with the previous SESSION_KEY encrypted with the ECU_KEY. The ECU decrypts the message using the ECU_KEY, subsequently recovering the new SESSION_KEY by XORing the decrypted message with the previous SESSION_KEY.
CR2AM requires ECUs to retain the SESSION_KEY of the previous session, as it is used to receive the SESSION_KEY of the subsequent session. This enhances the robustness of the protocol by mandating the presence of two keys instead of one, which are the previous SESSION_KEY and the ECU_KEY.

B. Counter Management Component
The Counter Management Component is responsible for generating and storing nonces used for Ascon encryption, which is done by the Authenticated Encryption Component. Upon initialization, it assigns a random 16-byte number to each CAN-ID, ensuring that each identifier has its own counter. The Counter Management Component acts as a monotonic counter, incrementing the counter by one after a successful transmission or reception of a CAN message.
Counters are reset with each SESSION_KEY update. Following the CR2AM process, the MECU sends newly generated counters encrypted with the updated SESSION_KEY to the recipient ECU.

C. Counter Recovery Mechanism
To ensure synchronization of counters across all ECUs for all CAN-IDs, a counter is incremented only upon successful transmission or reception of a message. Successful reception implies successful decryption of the message. However, if a recipient ECU attempts to decrypt an injected message that doesn’t contain the correct tag, it will discard the message, leaving the counter unchanged.
Issues such as network disruptions, ECU malfunctions, DoS, or CAN injection attacks may cause messages to be dropped, leading to a loss of synchronization between sender and recipient ECUs. This is because the sender ECU increments the counter after sending the message, while the recipient ECU does not increase the counter if the message is not received. To address this issue, a counter recovery mechanism is developed (illustrated in Fig. 8). Initially, an ECU sends a request for a specific counter value to the MECU. The MECU responds with a random counter value in plaintext, designated solely as a nonce for the next encrypted message. The subsequent message contains the required counter value encrypted with the SESSION_KEY. The recipient ECU can then decrypt the message using the SESSION_KEY and the nonce received from the previous message, thereby recovering the necessary counter value.

D. Authenticated Encryption Component
The Authenticated Encryption Component serves as the central component of LiREAP, responsible for encrypting and authenticating CAN messages using the Ascon cipher. Prior to transmitting a CAN message onto the bus, an ECU forwards it to the Authenticated Encryption Component. Utilizing the SESSION_KEY as the encryption key and the counter generated by the Counter Management Component as the nonce, the Authenticated Encryption Component encrypts the CAN messages. Due to the CAN standard's limitation, which restricts the maximum length of the data field to 8 bytes, the Authenticated Encryption Component utilizes the first 6 bytes of the data field for the encrypted message and the remaining 2 bytes for the truncated authentication tag. Consequently, the maximum length of a CAN message transmitted using LiREAP is 6 bytes. Messages exceeding this length are divided and sent across multiple CAN messages.
The sender ECU encrypts a CAN message and embeds the ciphertext along with the first two bytes of the authentication tag into the data field of a CAN frame. Upon receiving the frame, the recipient ECU attempts to decrypt the message. If the authentication of the tag fails, indicating tampering or unauthorized access, the recipient ECU discards the packet. Fig.
9 illustrates the interoperability of LiREAP components in generating secured CAN messages.

III. SECURITY ANALYSIS
In a CAN network secured by LiREAP, an attacker is unable to decrypt, forge, inject, or reverse engineer CAN messages. Additionally, compromising one secret key does not imply compromising the whole protocol. Moreover, periodically updating session keys limits the compromise time of a key. Although LiREAP cannot prevent DoS attacks, it can recover from it using the Counter Recovery Mechanism mentioned in the previous section.

A. Decryption and Chosen Plaintext Attacks

LiREAP ensures confidentiality and integrity for CAN data by employing Ascon cipher, implemented within the Authenticated Encryption Component. All schemes in Ascon, including AEAD and hashing, offer 128-bit security if the implementation requirements are met [28]. To fulfill these requirements, counters used as nonces for Ascon encryption are monotonically increased by the Counter Management Component after each successful message transmission or reception. This practice ensures that a nonce is never repeated for more than one message under the same key, thereby avoiding the nonce-misuse scenario that is vulnerable to chosen-plaintext attacks (CPA) [29][30].

B. Forgery Attacks
In a forgery attack, an attacker tries to craft a message with a valid authentication tag without possessing the encryption key. However, if the SESSION_KEY, serving as the encryption key, is securely stored in a hardware security module (HSM), such attacks become obsolete.

Forgery attacks can occur even without knowledge of the encryption key, where an attacker aims to find a collision, causing two distinct plaintext messages to yield the same authentication tag. It is noteworthy that reducing the authentication tag to 8 bytes can maintain acceptable security of Ascon [31]. However, LiREAP, constrained by the CAN protocol's environment, utilizes only 2 of the 16 bytes of the authentication tag, potentially increasing the likelihood of a successful forgery attack. Despite this consideration, in a LiREAP environment, finding a collision in the tag proves exceedingly challenging due to the non-public nature of counters. An attacker must guess the current counter value on the bus to initiate a collision search. Since counters are 8 bytes in length, there are 2^{64} possibilities for a counter value. Consequently, by concealing the counter value from attackers and unauthorized ECUs, the probability of successfully discovering a collision in the tag is significantly reduced.

C. CAN Injection Attacks
CAN injection attack is a kind of replay attack, in which an attacker intercepts encrypted CAN messages from the bus, stores them, and attempts to resend them later. This type of attack is mitigated by using counters that are updated for each message.Hence, even if two plaintext messages are identical, their corresponding ciphertext will be different because each message encryption utilizes a different nonce. Therefore, if an attacker stores an encrypted message and attempts to send it later, its authentication tag will not be correct. This is because the receiving ECU will use a new counter to decrypt the message and calculate its tag. As the tags will not match, the ECU will drop the packet.

D. Reverse Engineering
Reverse engineering CAN messages involves identifying the function of each message, which can be exploited by attackers to manipulate vehicle systems, such as unlocking doors or disabling engine immobilizers. LiREAP mitigates this risk by encrypting each message using a unique counter, ensuring that the same message produces different ciphertexts in separate encryptions. As a result, intercepting CAN messages does not provide attackers with actionable information, effectively preventing reverse engineering and unauthorized manipulation of vehicle systems.

E. Robustness
Robustness in LiREAP is ensured by the requirement for critical operations to involve multiple secret components instead of relying on one. For example, in CR2AM, an ECU must demonstrate possession of both the ECU_KEY and the previous SESSION_KEY to obtain the new SESSION_KEY from the MECU. This reduces the risk of protocol failure if one component is compromised.

The MASTER_KEY and SUBKEY, stored within the MECU in a hard-to-reach location inside the vehicle, are considered highly secure and difficult to compromise. However, the robustness of LiREAP extends to scenarios involving compromised secret keys stored in an ECU, including the SESSION_KEY and the ECU_KEY. The following scenarios emphasizes the protocol robustness:

1) If the ECU_KEY is compromised, an attacker cannot send authenticated messages or authenticate to the MECU for a new SESSION_KEY without compromising the previous SESSION_KEY.
2) If the SESSION_KEY is compromised, an attacker cannot send authenticated messages or authenticate to the MECU without the ECU_KEY. Also, producing the next SESSION_KEY requires knowledge of the SUBKEY.

To further mitigate the risk of compromising both keys simultaneously, it is advisable to store the SESSION_KEY and ECU_KEY in separate locations within the ECU with distinct sets of protections.

F. Overhead Analysis
LiREAP is designed with efficiency in mind, particularly regarding the limited resources typically available in

CAN environments. It aims to minimize overhead while still achieving the desired security objectives. One key aspect of LiREAP is its use of Ascon, which is anticipated to become the NIST standard for lightweight cryptography. Ascon's benchmarks demonstrate its efficiency, particularly for short messages [26], making it well-suited for constrained

TABLE 2 MESSAGES OVERHEAD FOR THIS WORK AND PREVIOUS WORKS

	Number of required CAN frames		
Original message size (bytes)	**Without cryptography**	**[18] [32] [33] [34]**	**LiREAP**
1 to 6	1	2	1
7 to 8	1	2	2
9 to 12	2	4	2
13 to 16	2	4	3
17 to 18	3	6	3
19 to 24	3	6	4

environments like CAN networks. Also, as Ascon is an authenticated encryption cipher, it achieves both confidentiality and integrity using the same cryptographic primitive. Moreover, both Ascon's AEAD and hashing functions are based on the same permutation, which reduces the code size and memory footprint required for implementation. Furthermore, LiREAP uses session keys and a centralized MECU for key management, thereby reducing overhead associated with key management.
By embedding the truncated authentication tag with the ciphertext within the same CAN frame, LiREAP enables small messages that do not exceed 6 bytes to be sent in a single secured CAN message. Table II demonstrates that LiREAP requires fewer messages compared to previous works that requires sending a separate message to authenticate the original CAN message, which increases the busload [18] [32] [33] [34].

IV. Conclusion

The famous CAN protocol was designed to achieve high reliability and speed for in-vehicle communications. However, its implementation suffers from severe security risks that results in various attacks along the last decay. This paper proposed a Lightweight Encryption and Authentication Protocol (LiREAP) to mitigate these risks. By integrating the state-of-the-art lightweight cipher Ascon with proposed key management mechanisms, the proposed protocol offers robust security with very low overhead. The LiREAP succeeded to keep the high speed and reliability of the CAN protocol along with ensuring confidentiality, integrity, and authenticity of the CAN data. Future work includes implementing LiREAP in other in-vehicle communication standards, and optimizing it for FPGA platforms to achieve highest possible speed.

References

[1] CAN Specification, Robert BOSCH GmbH, Version 2.0, 1991.
[2] K. Koscher et al., "Experimental security analysis of modern automobile", Security and Privacy (SP) 2010 IEEE Symposium on Oakland CA USA, pp. 447-462, May 2010.
[3] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle", Black Hat USA, vol. 2015, pp. 91, 2015.
[4] Rapid7, CAN Bus Network Implementation in Avionics, 2019, [online] Available: https://www.cisa.gov/news-events/ics-alerts/ics-alert-19-211- 01.
[5] Toyota UK, Toyota GB statement on vehicle theft, 2024, [online] Available: https://mag.toyota.co.uk/toyota-gb-statement-on-vehicle-theft.
[6] K. Tindell, CAN Injection: keyless car theft, 2023, [online] Available: https://kentindell.github.io/2023/04/03/can-injection.
[7] Z. Palmer, Thieves are now stealing cars via a headlight 'CAN injection', 2023, [online] Available: https://www.autoblog.com/2023/04/18/vehicle- headlight-can-bus-injection-theft-method-update.
[8] E. Kovacs, Thieves Use CAN Injection Hack to Steal Cars, 2023, [online] Available: https://www.securityweek.com/thieves-use-can-injection-hack- to-steal-cars.
[9] S. Lee, W. Choi, H. J. Jo and D. H. Lee, "ErrIDS: An enhanced cumulative timing error-based automotive intrusion detection system", IEEE Trans. Intell. Transp. Syst., vol. 24, no. 11, pp. 12406-12421, 2023.
[10] Y. Zhao, Y. Xun and J. Liu, "ClockIDS: A real-time vehicle intrusion detection system based on clock skew", IEEE Internet Things J., vol. 9, no. 17, pp. 15593-15606, 2022.
[11] J. Zhou, G. Xie, S. Yu and R. Li, "Clock-based sender identification and attack detection for automotive CAN network", IEEE Access, vol. 9, pp. 2665-2679, 2021.

[12] J. Park, D. Kim and I, Suh, “Design and Implementation of Security Function According to Routing Method in Automotive Gateway”, in International Journal of Automotive Technology, vol. 22, pp. 19–25, 2021.

[13] Luo, F. and Hou, S., "Security Mechanisms Design of Automotive Gateway Firewall," SAE Technical Paper 2019-01-0481, 2019.

[14] S. Seifert and R. Obermaisser, "Secure automotive gateway-secure communication for future cars", in 2014 12th IEEE International Con- ference on Industrial Informatics (lNDIN), IEEE, pp. 213-220, 2014.

[15] AUTOSAR. Specification of Secure Onboard Communication AUTOSAR CP R19–11, 2019.

[16] Z. Lu, Q. Wang, X. Chen, G. Qu, Y. Lyu and Z. Liu, "LEAP: A Lightweight Encryption and Authentication Protocol for In-Vehicle Communications," 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, pp. 1158-1164, 2019.

[17] J. Cui et al., "Lightweight Encryption and Authentication for Controller Area Network of Autonomous Vehicles," in IEEE Transactions on Vehicular Technology, vol. 72, no. 11, pp. 14756-14770, 2023.

[18] H. Oguma et al., “Message authentication method in communication system and communication system”, European Patent 2775660B1, Jun. 2016.

[19] National Institute of Standards and Technology, Lightweight Cryptography (LWC) Standardization project, 2019, [online] Available: https://csrc.nist.gov/projects/lightweight-cryptography.

[20] W. Melching, "Extracting Secure Onboard Communication (SecOC) keys from a 2021 Toyota RAV4 Prime," icanhack.nl, Mar. 2, 2024. [online]. Available: https://icanhack.nl/blog/secoc-key-extraction/.

[21] S. Woo, H. J. Jo and D. H. Lee, "A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN", in IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 2, pp. 993-1006, 2015.

[22] F. Wiemer and A. Zeh, “Enabling Secure Communication for Automotive Endpoint-ECUs through Lightweight-Cryptography”, In Proceedings of the 7th ACM Computer Science in Cars Symposium (CSCS '23), Association for Computing Machinery, New York, NY, USA, Article 9,
pp. 1-10, Dec. 2023.

[23] F. Hartwich, Introducing CAN XL into CAN networks, 2020, [online] Available: https://www.can-cia.org/fileadmin/resources/documents/proceedings/2020_hartwich.pdf.

[24] J Klüser, CAN-based Protocols in Avionics, 2012, [online] Available: https://cdn.vector.com/cms/content/know-how/_application- notes/canopen/AN-ION-1-0104_CAN-based_protocols_in_Avionics.pdf.

[25] G.C. Kessler, “The can bus in the maritime environment–technical overview and cybersecurity vulnerabilities”, TransNav: International Journal on Marine Navigation and Safety of Sea Transportation, vol. 15, no. 3, 2021.

[26] The CAESAR Committee. Caesar: competition for authenticated encryption: security, applicability, and robustness, 2014.

[27] G. Bertoni, J. Daemen, M. Peeters and G. V. Assche, "Duplexing the sponge: Single-pass authenticated encryption and other applications", Proc. Sel. Areas Cryptograph. (SAC), pp. 320-337, 2011.

[28] Dobraunig, M. Eichlseder, F. Mendel and M. Schläffer, Ascon Submission to the NIST Lightweight Cryptography Standardization Process, 2021, [online] Available: https://csrc.nist.gov/CSRC/media/Projects/lightweight- cryptography/documents/finalist-round/updated-spec-doc/ascon-spec- final.pdf.

[29] Serge Vaudenay and Damian Vizár. Can Caesar Beat Galois?: Robustness of CAESAR Candidates Against Nonce Reusing and High Data Complexity Attacks. In Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings, pp. 476–494, 2018.

[30] M. T. Ali, "Generic CPA Decryption Attack on Ascon-128 in Nonce- Misuse Setting by Exploiting XOR Patterns," 14th International Conference on Electrical Engineering (ICEENG), Cairo, Egypt, pp. 172- 174, 2024.

[31] B. Chakraborty, C. Dhar and M. Nandi, “Exact Security Analysis of ASCON”, In: Guo, J., Steinfeld, R. (eds) Advances in Cryptology – ASIACRYPT 2023. ASIACRYPT 2023. Lecture Notes in Computer Science, vol 14440. Springer, Singapore, 2023.

[32] B. Palaniswamy, S. Camtepe, E. Foo, and J. Pieprzyk, “An efficient authentication scheme for intra-vehicular controller area network,” IEEE Transactions on Information Forensics and Security, vol. 15, pp. 3107– 3122, 2020.

[33] P.-S. Murvay and B. Groza, “Security shortcomings and countermeasures for the SAE J1939 commercial vehicle bus protocol,” IEEE Trans. Veh. Technol., vol. 67, no. 5, pp. 4325–4339, May 2018.

[34] A. Radu, F.D. Garcia, “LeiA: A Lightweight Authentication Protocol for CAN”. In I. Askoxylakis, S. Ioannidis, S. Katsikas, C. Meadows (Eds.), Computer Security – ESORICS 2016, ESORICS 2016. Lecture Notes in Computer Science, vol 9879, pp. 283-300 Springer, 2016.

	Dr. Mahmoud A. AttaAlla is a computer science professional with extensive education and research experience. He received his B.Sc., M.Sc., and Ph.D degrees in Computer Science from Mansoura University, Egypt in 2004, 2009, and 2017 respectively. Currently, he is with the Faculty of Computer Studies, AOU in Egypt, where he also serves as the program coordinator for the M.Sc. in Cybersecurity and Forensics. His primary research interests lie in the areas of network security, wireless networks, and routing. Dr. Attalah has published many papers in international conferences and journals, in the field of network and cyber security**.**
	Mohamed T. Ali (student member, IEEE) is graduated from faculty of Computer Studies at Arab Open University, Egypt. He served as a summer intern at Nokia, Egypt in 2023. He developed "playascon," an online tool for the Ascon cipher. Additionally, he attained first place at the International Competition of Military Technical College (ICMTC) for Cyber Security in July 2024 and second place in July 2023. He instructed Cryptography with Python course at AOU-Training and Community Services Center (TCSC) During 2023-2024. His current area of focus is lightweight
	Dr. Ahmed M. Gawish, received his B.Sc. and M.Sc. degree in Computer Science in 1997 and 2002, respectively; both from Ain Shams University, Cairo, Egypt. He got his Ph.D. degree from Tohoku University, Japan in 2009. He is currently with the Faculty of Computer Studies, Deanship-HQ, Arab Open University, Kuwait. His major research interest is in Cloud Computing, Security, modelling and simulation. He published over 50 referred papers in international conferences, journals, transactions, and book chapters. He is a member of IEEE and ACM.