

Article

Addressing Vulnerabilities in CAN-FD: An Exploration and Security Enhancement Approach

Naseeruddin Lodge , Nahush Tambe  and Fareena Saqib *

Department of Electrical and Computer Engineering, University of North Carolina at Charlotte, Charlotte, NC 28223, USA; ntambe@charlotte.edu

* Correspondence: nlodge@charlotte.edu (N.L.); fsaqib@charlotte.edu (F.S.)

Abstract: The rapid advancement of technology, alongside state-of-the-art techniques is at an all-time high. However, this unprecedented growth of technological prowess also brings forth potential threats, as oftentimes the security encompassing these technologies is imperfect. Particularly within the automobile industry, the recent strides in technology have brought about increased complexity. A notable flaw lies in the CAN-FD protocol, which lacks robust security measures, making it vulnerable to data theft, injection, replay, and flood data attacks. With the rising complexity of in-vehicular networks and the widespread adoption of CAN-FD, the imperative to safeguard the protocol has never been more crucial. This paper aims to provide a comprehensive review of the existing in-vehicle communication protocol, CAN-FD. It explores existing security approaches designed to fortify CAN-FD, demonstrating multiple multi-layer solutions that leverage modern techniques including Physical Unclonable Function (PUF), Elliptical Curve Cryptography (ECC), Ethereum Blockchain, and Smart contracts. The paper highlights existing multi-layer security measures that offer minimal overhead, optimal performance, and robust security. Moreover, it identifies areas where these security measures fall short and discusses ongoing research along with suggestions for implementing software and hardware-level modifications. These proposed changes aim to streamline complexity, reduce overhead while ensuring forward compatibility. In essence, the methods outlined in this study are poised to excel in real-world applications, offering robust protection for the evolving landscape of in-vehicular communication systems.



Citation: Lodge, N.; Tambe, N.; Saqib, F. Addressing Vulnerabilities in CAN-FD: An Exploration and Security Enhancement Approach. *IoT* **2024**, *5*, 290–310. <https://doi.org/10.3390/iot5020015>

Academic Editor: Amiya Nayak

Received: 29 February 2024

Revised: 12 May 2024

Accepted: 16 May 2024

Published: 30 May 2024



Copyright: © 2024 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) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: Controller Area Network (CAN); Controller Area Network Flexible Data-Rate (CAN-FD); Hardware Security; Physical Unclonable Functions (PUF); Elliptic Curve Cryptography (ECC); volatile memory (VM); non-volatile memory (NVM)

1. Introduction

In today's modern and technologically advanced era, automobiles have evolved beyond simple electrical components wired together for limited motor control. The digital transformation of vehicular electronics, marked by sophisticated sensors and state-of-the-art control units, has propelled the automotive industry into the realm of PCs, smartphones, and many other Internet of Things (IoT) devices. These advancements have not only significantly enhanced customer interactions and experiences but have also elevated the overall driving experience of vehicles. On-board electronics now encompass a wide array of functionalities like air-fuel mixing, temperature control, thermostats, vents and doors-locking to seat adjustments, radio and in-dash entertainment, windshield wipers, transmission, ignition, engine, steering, brakes, acceleration, airbags, in-built navigation system, cameras, guided braking, parking system and many more. This intricate network of electronic units, along with many other smaller interconnected systems, collectively form what is known as Electronic Control Units (ECUs). An Electronic Control Unit (ECU) is a device that uses embedded systems with a microcontroller to control various vehicular features. ECUs work in coordination with each other, requiring continuous exchange of states and data.

Rather than relying on traditional methods involving dedicated signal wires, which require extensive wiring efforts for information exchange, the automotive industry has transitioned towards using data communication networks. To assist real-time communication among connected ECUs, the CANBus communication network was introduced [1].

The vulnerabilities in the CAN bus system have been exploited by malicious actors, as demonstrated by Hoppe and Dittman in 2007 with their tampering of the electric window lift in a car simulation [2]. Charlie Miller and Chris Valasek further highlighted these vulnerabilities by remotely controlling various systems of a Jeep Cherokee, including the air conditioner and brakes [3]. Subsequent attacks have targeted various electronic systems in CAN-enabled automobiles, revealing vulnerabilities such as Denial of Service (DoS) attacks and packet sniffing [4]. These incidents underscore the importance of robust cybersecurity measures in the automotive industry as vehicles become increasingly reliant on interconnected digital systems.

A comprehensive study on the security evaluation and countermeasures for CAN and CAN-FD protocols is presented in this work. Section 2 provides an extensive literature survey, delving into the functionalities and intricacies of both CAN and CAN-FD protocols. Section 3 elucidates the deficiencies within these protocols, shedding light on their vulnerabilities and susceptibility to various types of attacks. Following this analysis, Section 4 explains mathematical preliminaries essential for understanding the techniques proposed in the work. Section 5 delves into an explanation of state-of-the-art security frameworks designed to mitigate these vulnerabilities, incorporating technologies such as Physical Unclonable Functions (PUFs), blockchain, and advanced cryptographic techniques. Furthermore, hardware-based testing methodologies, including embedded systems and Field-Programmable Gate Arrays (FPGAs) are performed to assess the robustness of these security frameworks. Section 6 presents the results and analysis of the proposed work. Section 7 delves into ongoing research efforts from the author's side and identifies areas warranting additional investigation to further enhance the security of CAN and CAN-FD protocols against emerging threats. Finally, Section 8 concludes the exploration of CAN and CAN-FD underscoring their pivotal roles in modern automotive and industrial applications, offering robust communication protocols for real-time data transmission, enhanced bandwidth capabilities, and resilient networking architectures essential for the advancement of interconnected systems in the digital age.

2. Intra-Vehicular Communication Standard: CAN and CAN-FD

Controller Area Network (CAN) is a communication protocol designed to facilitate seamless connectivity among multiple electronic control units (ECUs) while enabling fast and efficient communication. Introduced by BOSCH in 1986, the CAN protocol emerged as a solution to the escalating complexity of wiring and the demand for fast, reliable communication in the rapidly expanding automobile industry. With the continual evolution of the automobile sector, the demand for a growing number of ECUs and sensors has become more pronounced than ever before. It is precisely to meet this demand that the CAN protocol has become indispensable. Furthermore, it's worth noting that CAN-FD, an improved version of CAN, extends beyond the automotive sector, finding application in various other industries and devices due to its simplicity and reliability.

2.1. Dataframe: CAN vs. CAN-FD

The CAN protocol stands as one of the most extensively employed communication protocols for in-vehicle networks, enabling seamless ECU and sensor communication. The major characteristic that defines the traditional CAN is its maximum bandwidth of up to 1 Mbit/s and a maximum payload of 8 bytes per frame. On the other hand, CAN-FD (CAN with Flexible Data-Rate) comes with a varying bandwidth of 1 to 10 Mbit/s and a maximum payload of 64 bytes per frame. Figure 1 demonstrates the difference between CAN and CAN-FD dataframes [5].

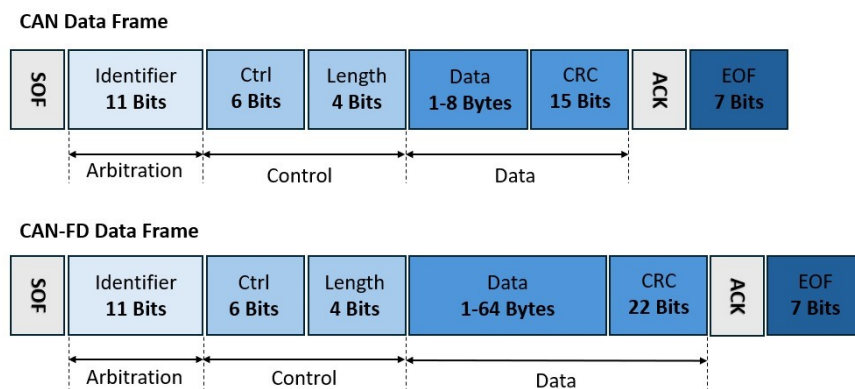


Figure 1. CAN vs. CAN-FD Dataframes [6].

2.1.1. Frame Structure Overview: CAN

The above figure illustrates the distinction between CAN and CAN-FD protocols. Below, the discussion delves into the frame structure and the functionality of the original CAN [7]:

- **Start of Frame (SOF)**—Initially, the CAN bus remains in an idle state, indicating no ongoing transmissions on the bus. When a node intends to transmit a message, it sends a dominant bit on the bus, thereby altering its state. In this context, a dominant bit represents a logical 0.
- **Identifier/Arbitration Field**—Once the SOF is set, an identifier is generated, reflecting the message's priority level and its transmission or reception status. Frames with lower identifiers denote higher priority. The remote transmission request (RTR) is a 1-bit frame that determining whether the message transmits a data package or solicits data from another node. A dominant bit (logic 0) signifies a data transmission frame, while a recessive bit (logic 1) indicates a message request for data from another device.
- **Control Field**—Within the control field, the 6 bits encompass a 1-bit identifier extension bit (IDE), with the subsequent bit reserved for future CAN standards. Finally, the last four bits are designated for the data length code (DLC). A dominant bit for IDE represents an 11-bit identifier, while a recessive bit represents an extended 29-bit identifier. It's important to note that when using an extended 29-bit identifier, the additional bits extend into the data field. DLC comprises the last four bits that indicate the number of bytes in the message to be transmitted.
- **Cyclic Redundancy Check (CRC)**—This 15-bit frame is employed for error detection purposes. Upon receiving a CRC, if it differs from the calculated CRC, an error is detected. In such scenarios, the message is discarded, and the transmitter endeavors to retransmit it following re-arbitration. Once the transmitter completes the data transmission and redundancy check, it confirms the absence of errors by utilizing the acknowledgment frame (ACK). A dominant ACK frame indicates successful message transmission.
- **End of Frame (EOF)**—EOF signals the conclusion of the frame, returning the bus into an idle state, ready for subsequent transmissions to take place.

2.1.2. Frame Structure Overview: CAN-FD

Below, the specific characteristics of CAN-FD, evolved from the original CAN, are outlined [7]:

- **Remote Request Substitution (RRS)**—In the conventional CAN, the remote transmission request (RTR) bit present in the Identifier/Arbitration Field has been supplanted by RRS in CAN-FD. Unlike RTR, RRS is always dominant. This means that CAN-FD cannot directly solicit data from another device as CAN does; instead, it utilizes an indirect method. This adjustment reduces overhead and complexity while enhancing flexibility and compatibility.

- **Control Field**—The reserved bit present in CAN is replaced with the Flexible Data Rate Format (FDF), which is recessive (logic 1). This serves as a differentiator between classical CAN and CAN-FD frames. Additionally, three new bits are introduced in this field. The RES bit functions similar to the reserved bit in classical CAN, i.e., reserved for future CAN standards. The Bit Rate Switch (BRS) bit is dominant when the data rate in a CAN-FD data frame is transmitted at the arbitration rate (i.e., up to a maximum of 1 Mbit/s), and recessive when the data frame is transmitted at a higher bit rate (up to 5 Mbit/s). The third bit, Error Status Indicator (ESI), defaults to dominant. A dominant ESI indicates ‘error active’, while the transmitter switches to recessive (1) to indicate ‘error passive’ mode when it detects errors in its transmissions or receives error frames from other nodes.
- **Data Frame**—The Data Frame in CAN-FD undergoes a significant enhancement over its classical CAN counterpart, expanding from a maximum payload of 8 bytes to an impressive 64 bytes. This substantial increase not only facilitates much higher data rates but also offers greater flexibility in accommodating diverse payloads. By enabling the transmission of larger volumes of data in a single frame, the upgraded Data Frame minimizes overhead and optimizes bandwidth utilization, resulting in more efficient communication.
- **Cyclic Redundancy Check (CRC)**—The CRC frame is upgraded to be 22 bits in CAN-FD, allocating 17 bits for 16 data bytes and 21 bits for 20–26 data bytes. This enhancement improves upon the original error-checking method, ensuring robust error detection capabilities.

The key distinction between CAN and CAN-FD lies in the data frame sizes, with CAN-FD offering nearly eight times larger data frames. This substantial increase enhances throughput, enabling faster packet transmission. The expanded CRC data frame in CAN-FD significantly improves the error-checking mechanism compared to the original CAN protocol. A quantitative comparison was performed between CAN and CAN-FD, analyzing message size in bits, overhead in percentage, and worst-case transmission time (WCTT) across various payloads. The experimentation incorporated payloads ranging from 1 to 64 bytes. For CAN modules, payloads greater than 8 bytes were segmented into multiple messages. The results were tabulated and graphed for visual representation, with Figures 2 and 3 illustrating the findings.

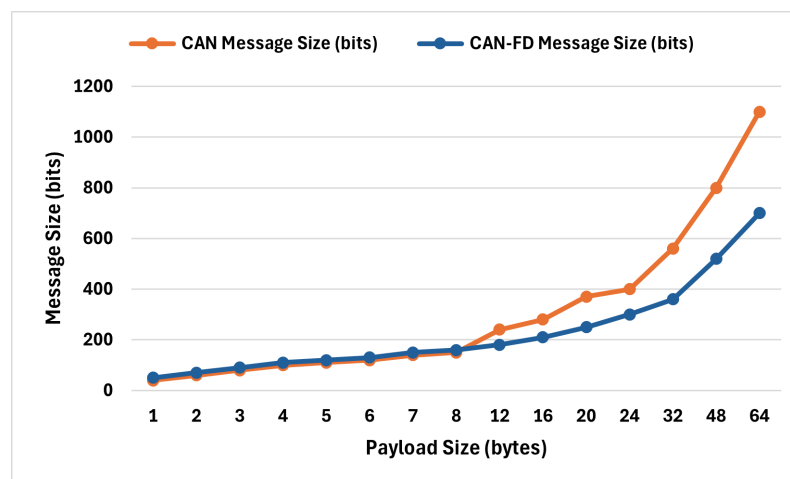


Figure 2. Comparison of Message Size with Varying Payload for CAN vs. CAN-FD [8].

CAN and CAN-FD exhibit similar performance for payload sizes below 8 bytes. As the payload size increases, CAN-FD outperforms CAN substantially in terms of message size and overhead. Moreover, while the message size of CAN increased significantly for payloads over 12 bytes, the message size of CAN-FD shows a steady increase and, on average, remains lower for the same payloads. The second experiment focused on

bandwidth utilization, using a real message set provided by a Japanese OEM. In this test, CAN-FD showed a superior bandwidth performance compared to CAN [8].

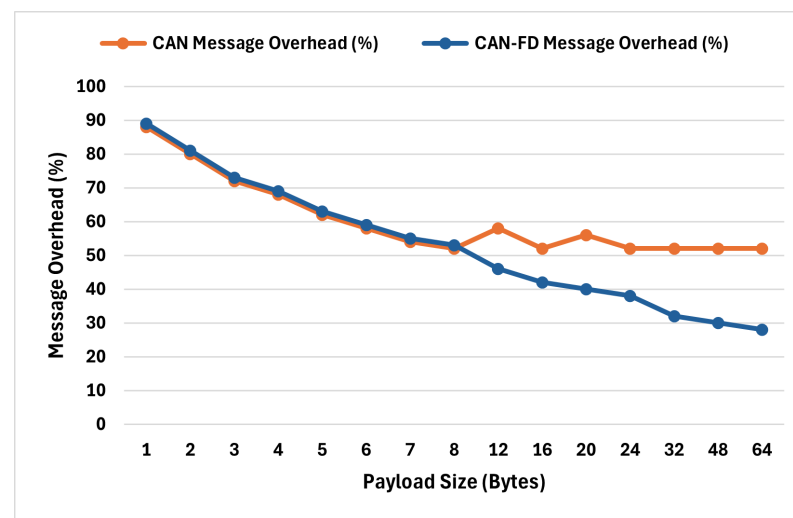


Figure 3. Comparison of Message Overhead with Varying Payload for CAN vs. CAN-FD [8].

CAN-FD offers significant upgrades over CAN that are in high demand within the current automobile industry. It provides a smooth integration with existing systems, as the current CAN bus can easily support CAN-FD nodes on their systems with minimal effort through network configuration modifications and ECU software upgrades. The expanded payload capacity allows for the transmission of more data in a single frame, while CAN-FD's flexible data rate capability enables faster communication speeds, thereby enhancing reliability. These enhancements make position CAN-FD as a much superior protocol compared to its older counterpart [9].

2.2. Challenges of CAN/CAN-FD

One challenge lies in forward compatibility, especially in scenarios where hardware remains outdated. Original CAN controllers lack forward compatibility with CAN-FD, necessitating a complete system update when transitioning from Classical CAN to CAN-FD. This transition can potentially introduce compatibility issues and incur additional costs.

Moreover, the enhanced payload capacity and data rates supported by CAN-FD contribute to protocol complexity. Managing higher data rates and larger payloads demands sophisticated hardware and software solutions, which can pose challenges during implementation, debugging, and ongoing maintenance processes.

Adjusting multiple data rates in CAN-FD also presents challenges, especially in ensuring proper timing and synchronization between nodes operating at different speeds. Real-time systems, in particular, require precise coordination to prevent data corruption or loss.

Another significant concern revolves around security. The enhanced capabilities of CAN-FD, including increased data rates and larger payload sizes, expose the protocol to potential security vulnerabilities. Addressing these concerns is paramount to ensuring secure communication and safeguarding against unauthorized access and malicious attacks.

3. Security Vulnerabilities in CAN-FD Systems

While CAN-FD has significantly elevated in-vehicular communication in terms of ease and efficiency, a prominent drawback is that it lacks security protocols. The vulnerabilities present in CAN-FD can be divided into three aspects: Weak access control, lack of encryption, and lack of message authentication.

Weak access control refers to the process of compromising the CAN-FD protocol through physical connections. This includes illicit activities like information theft via the external OBD interface (onboard diagnosis) and the transmission of unauthorized messages

to change and manipulate ECUs in vehicles. Such vulnerabilities pose a serious threat to the safety of both, drivers and passengers.

The lack of encryption in the CAN-FD protocol exposes all broadcasted data frames to potential eavesdropping by attackers. This vulnerability allows malicious adversaries to seize and analyze the information, providing them with the means to decipher the data framework. Such illicit access can lead to ECU control manipulation, posing a significant security risk to the overall operation of the in-vehicular system.

Another area lacking in the current CAN-FD protocol is authentication. On the receiver side, control data frame identification is dependent solely on the sender ID within the data frame. A malicious actor can easily resend a control data frame disguised as a legitimate sender, leaving room for possible security threats. The state-of-the-art CAN-FD implementation provides only one CRC code for transmission error detection [10]. Resolving these authentication vulnerabilities is crucial to strengthening the security of the CAN-FD protocol.

3.1. Exploring Vulnerabilities

The notable vulnerability in both CAN and CAN-FD protocols lies in their absence of encryption, leaving transmitted messages susceptible to interception and manipulation by attackers. The lack of effective encryption compromises the integrity, confidentiality, and availability of data within CAN-FD [11].

- **Integrity**—Integrity safeguards the accuracy and completeness of transmitted data, protecting it from unauthorized modifications. Although CAN employs techniques like cyclic redundancy checks (CRC) and error confinement mechanisms (ECM) to ensure data accuracy, these methods fall short in providing sufficient integrity against malicious alterations. Research has revealed the potential for attackers to manipulate vehicle modules through the On-Board Diagnostics II (OBD-II) port, jeopardizing critical systems such as brakes and engine control [12].
- **Confidentiality**—Confidentiality shields data from unauthorized access and exploitation. However, the CAN protocol lacks built-in measures for confidentiality, such as password authentication and encryption. Although some contemporary vehicle manufacturers have implemented cryptographic methods, these solutions have limitations and may not offer comprehensive protection. For instance, even the encrypted ECU in the Nissan GT-R was susceptible to hacking [13].
- **Availability**—Availability guarantees uninterrupted access to a system by authorized users, yet the arbitration rule within the CAN protocol undermines this aspect. When a high-priority node continues to transmit data for an extended period, lower-priority nodes may be denied access to the bus, potentially leading to Denial of Service (DoS) attacks [11].

Exploring distinct attack models is imperative for a comprehensive understanding of the potential avenues through which malicious actors can gain unauthorized access to data on the CAN-FD bus.

Data Theft—The susceptibility of CAN-FD to data theft emerges from its reliance on broadcast communication within the data link layer, aggravated by the absence of authentication protocols. This vulnerability exposes the potential for unauthorized manipulation, allowing attackers to acquire sensitive data, including but not limited to, vehicle driving track data, user identity details, and vehicle mileage numbers.

Malicious Data Injection—In light of the absence of a data authentication mechanism within the CAN-FD bus, an attacker can analyze stolen information to gain administrator permissions for the vehicle terminal. Subsequently, using compromised or unauthorized nodes, the attacker can transmit deceptive messages across the bus, inducing confusion. This manipulation empowers the attacker to generate and inject diagnostic messages, prompting the Electronic Control Unit (ECU) to execute arbitrary actions. This extends control over various ECU components in the vehicle, enabling manipulation of its behavior.

Denial-of-Service (DoS) Attack—This type of attack involves sending messages at an elevated rate which disrupts the normal transmission of frames and impedes the reception or processing of information by legitimate users. The severity of flooding attacks is contingent upon the intervals between packet transmissions. In scenarios where the packet transmission frequency surpasses the normal rate, nodes with lower priorities are forced to wait until the bus is idle before transmitting packets, resulting in a breakdown of bus transmission. Conversely, when the transmission frequency is low, the receiving node may execute operations based on the malicious messages, thus granting the attacker complete control over the vehicle's operational status. An experimental assault on a 2012 Alfa Romeo Giulietta illustrates the vulnerability of the Controller Area Network (CAN) protocol to selective denial-of-service (DoS) attacks without any alterations [11,14].

Data Replay—In this form of attack, an attacker intercepts and reads messages transmitted on the bus, subsequently replaying them to overwrite legitimate data. This manipulation can be used to control parts of vehicles, such as lights and doors. If the attacker impersonates a genuine node on the bus, they can send packets with high-priority IDs that are not expected in the current operational state. By acquiring priority to send messages, this deceptive activity results in a significant waste of communication resources [15].

Remote Access Attacks—Remote access attacks have emerged as a major concern with the proliferation growth of wireless technologies. ECUs now integrate various wireless protocols such as bluetooth and radio. Reverse engineering efforts have been pursued to compromise the FM channel, Bluetooth, and cellular network of vehicles [13]. Additionally, wireless attacks have exploited malicious apps to monitor vehicle and execute attacks from a considerable distances [16].

3.2. Security Countermeasures and Evaluation

A lightweight MAC-based solution termed Chaskey, proposed by Nicky Mouha addresses security concerns in CAN-FD. Chaskey is specifically designed for 32-bit low-power microcontrollers with speed limitations, offering robust 128-bit security for critical embedded applications. The algorithm utilizes an Addition-Rotation-XOR design along with a permutation π at its core. The message is divided into blocks of 128 bits, and a 128-bit key is employed. The key is transformed into two subkeys, and the XOR of two 128-bit streams feeds into the permutation. This streamlined approach ensures efficient computation and robust security for critical embedded applications. However, Chaskey suffers from the vulnerability of data replay, where attackers intercept and retransmit the same frames observed by the receiver, potentially obtaining a session key to impersonate an original node. Additionally, there is a need for a robust key management protocol to enhance Chaskey's effectiveness [17].

Another encryption method proposed in [18] involves utilizing AES-128 and SHA-256 for encryption at the sender side and decryption at the receiver side. AES-128 is employed to encrypt the 120 bit message and an 8 bit message counter with a 128 bit cipher key. Subsequently, SHA-256 is utilized to add an additional 256 bits to the encryption process, forming a 384 bit output packet. On the receiver side, the hash function generates a 256 bit hash, which is compared with the received 256 bit hash for sender authentication. Upon successful authentication, the AES-128 decryption core decrypts the received ciphertext using the shared cipher key. This method provides strong encryption and immunity from replay attacks. However, the keys stored on the FPGA can be vulnerable to hacking, and the use of message counters reduces throughput efficiency by decreasing the effective length of transmitted data from 128 bits to 120 bits [18].

Additionally, a method proposed in [19] aims to prevent replay attacks by integrating a random ID key generated by the receiver. The sender appends this key to each message, and the receiver authenticates it before processing the message. The sender maintains the ID key's value and increments it with each message, while the receiver accepts a slightly higher ID key value to handle potential message losses. The ID key remains valid for a certain duration, after which the receiver generates a new one and notifies the sender.

In case of delayed response from the sender, the receiver generates a new ID key and retransmits it. While effective in preventing replay attacks, this method assumes that the node itself has not been substituted. If an authentic node is replaced with a malicious one, the malicious node could still gain access to the randomly created ID [19].

The proposed work aims to bolster the security of intra-vehicular communication over CAN-FD by implementing a multi-layered security framework. By leveraging advanced hardware-based cryptographic techniques like Physical Unclonable Functions (PUF) and Elliptic Curve Cryptography (ECC), combined with robust secure enrollment and authentication processes, the objective is to establish a formidable defense mechanism against diverse cyber threats including data tampering, unauthorized access, and replay attacks. This framework endeavors to ensure the integrity, confidentiality, and authenticity of communication within the vehicular network, thereby fortifying the overall system resilience and safety.

4. Hardware Security Countermeasure for CAN-FD

Despite the existence of commendable countermeasures for securing CAN-FD protocol, a few vulnerabilities persist. Challenges like replay attacks, key management, high overhead, data flooding, and malicious data injections continue to threaten CAN-FD protocol. The proposed countermeasures utilize techniques like Physical Unclonable Function (PUF), Elliptical Curve Cryptography (ECC), Ethereum Blockchain, and Smart contract that assist in eliminating the issues still lingering around CAN-FD by providing a multi-layer security. The research outlined in this section illuminates the most effective security approaches available for securing the CAN-FD protocol. Section 5 further delves into ongoing research and necessary improvements to fully secure CAN-FD, underscoring the continual efforts required to enhance its security posture.

4.1. Secure Intra-Vehicular Communication over CANFD

Old-fashioned software-based cryptographic remedies prove ineffective when implemented in embedded systems due to their limited processing power and real-time constraints. In fact, employing such software solutions for cryptographic operations can expose the entire system vulnerable to an adversary, capable of extracting the source code and reverse engineer to obtain cryptographic algorithms and keys. Hardware-based enhancements offer a more secure approach by fully securing each connected Electronic Control Unit (ECU) and minimizing software dependency on resource-constrained devices. Furthermore, with the integration of hardware-based obfuscation methods, reverse engineering attacks aimed at extracting keys can be mitigated. The proposed method eliminates the need to store secret keys on non-volatile memory within the nodes or the ECUs, eradicating probing attacks as well [1].

The outlined framework is based on client-server architecture, where a local server node within the network is responsible for managing all connected ECU nodes. Before any communication occurs among the nodes connected over the network, all clients must undergo registration and authentication by the server. This registration process occurs within a secure and trusted environment, such as during manufacturing, assembly, or at a reputable diagnostic center dedicated to the vehicle. Section 5.1 delves into an in-depth exploration of a leading security approach for CAN-FD highlighted in the paper titled "Secure Intra-Vehicular Communication over CANFD" [1]. This section covers CAN-FD implementation on Embedded systems to simulate a working ECU as used in automobiles.

4.1.1. PUF Block

In the realm of Hardware Security, Physical Unclonable Function (PUF) is gaining significant traction as an emerging physical layer cryptographic primitive. Exploiting intrinsic manufacturing process variations, these embedded structures extract unique yet reproducible secrets through a challenge-response pair (CRP) technique. The PUF is

presented with a digital input, typically in the form of a bitstring of '0' s and '1' s, serving as the challenge.

The ensuing digital output necessitates on-chip manipulation of minor analog variations captured by the PUF, converting them into a digitized form. Two different varieties of PUFs, namely Weak and Strong, exist. A prominent difference lies in the fact that the weak PUF operates with only a limited number of challenges, generating a unique key for each challenge, while the strong PUF has a large challenge space, resulting in a unique response for the majority of applicable challenges [1].

4.1.2. Hardware Embedded DeLay PUF (HELP)

Hardware-embedded delay PUF (HELP), a security feature integrated into an on-chip functional unit, is specifically designed to secure a segment of the Advanced Encryption Standard's (AES) S-box. The functional unit's logic gate structure comprises primary inputs (PIs) and primary outputs (POs), forming a neatly composed network of interconnected wires and transistors.

The primary object of HELP is to use path delay as a measure, defined as the amount of time (Δt) it takes for introduced bit transitions on the PIs to generate through the logic gate structure and emerge on a PO. HELP utilizes a clock-strobing technique to obtain high-resolution measurements of these path delays. Referred to as PNs, the digitized path delays, are collected and stored in an on-chip block RAM (BRAM). Once a set of 4096 PNs is collected, a series of operations, executed in VHDL, are initiated to produce a bitstring and helper data. The resulting cryptographic quality of the HELP PUF establishes a reliable, unique, and unpredictable response, forming a robust foundation for applications such as the Public/Private Key Generation Block in cryptographic systems [1].

4.1.3. Bit Generation

During enrollment, every ECU is assigned the responsibility of producing a unique, reproducible response or key as its private key. This is accomplished using seed and configuration parameters. By utilizing this response along with stored configuration parameters as inputs to the Public/Private Key Generation Block, a public key is generated. This public key is then stored on the Non-Volatile Memory (NVM) of the ECU. Additionally, helper data stored aids in the regeneration of the same response for a given challenge and public key. Subsequently, PUF-based authentication is integrated within the vehicular network [1].

4.1.4. Public/Private Key Generation Block

In hardware, the Elliptic Curve Cryptography Diffie-Hellman (ECDH) based asymmetric key exchange and encryption engine is implemented. ECDH, suitable for resource-constrained devices, provides security comparable to RSA with reduced requirements. The ECC algorithm operates in Galois Field (GF), especially Prime Fields (GF(Fp)) and Binary Fields (GF(F₂^m)), following NIST recommended curves.

Utilizing the PUF response as a private key, the ECC algorithm generates public keys for each ECU during the one-time enrollment process. Every time the vehicle's ignition is activated, a process referred to as authentication occurs, generating public and private keys for each ECU [1].

4.1.5. Authentication and Shared Key Exchange

During startup, each connected node is required to authenticate itself with the server. This is achieved by sending a packet containing its ID as the target identifier and its public key, encrypted with the shared key, utilizing Elliptic Curve Cryptography (ECC) in the payload. Upon receipt of the packet, the server decrypts the payload to compare the public key with the sender ID sent in the arbitration bits. Once the ECU is authenticated, the server shares the shared secret keys of other ECUs with the requesting node for communication with other connected nodes across the network. The authentication process is timed, and

failure to authenticate within the allocated time results in node blacklisting for the session, triggering an alert on the user dashboard. To uphold security, private keys for all nodes are dynamically generated at runtime, not stored, never leave the node, and are erased at the end of the session [1].

4.1.6. Key Storage

Key Storage consists of non-volatile memory (NVM) to store:

- Seed, challenge, and helper data for unique, random, but reproducible PUF response generation.
- Public Key of the server for authentication.
- Configuration parameters for public/private key generation.

The framework enables private key generation dynamically for each and every ECU, thereby eliminating the need for storage. Private and shared session keys are stored temporarily on volatile memory. For every transceiver-receiver pair, elements stored on the volatile key storage and separated from the untrusted apps and CAN bus interface using TrustZone, are:

- Shared secret key generated for encryption of every pair of ECUs
- Public Key of the communicating node [1]

4.1.7. Hardware Access Isolation

After generating shared keys for every pair of nodes, encryption and decryption are executed using either AES-128 or AES-256 engine, determined by system designers. Messages are encrypted at the sender's end and decrypted at the receiver's end using the shared key [1].

For hardware access isolation, TrustZone is utilized, to ensure secure cryptographic IPs are exclusively accessible through the secure world. To facilitate communication between ECU node and the secure framework, a set of Application Programming Interface (APIs) are implemented which offer various functions described below.

- *generatePubKeyAuth()*—Function for authentication message generation for the server, that contains the public key of the ECU, encrypted using a shared encryption key.
- *addNode()*—Function to pass the encrypted public key of the authenticated node to the secure framework for decryption and storage, once the server responds with the authenticated ECU.
- *generateMessage()*—Function to pass ECU ID of the receiver ECU and the message to be transmitted. Additional functionality includes instructing the framework to encrypt the message.
- *decryptMessage()*—Function called, using sender's node ID and encrypted message as an argument, for message decryption.

4.1.8. Encryption/Decryption Block

The encryption/decryption block is the final component of this framework, responsible for utilizing the shared secret key to encrypt or decrypt any message using the Elliptic Curve Integrated Encryption Scheme (ECIES). It ensures that all messages are encrypted before transmission and decrypted after being received by the node. Figure 4 provides a visual representation of the constituent elements in this framework.

This framework has been rigorously tested on Xilinx Kintex KC705 FPGA Evaluation Board, a certified automotive device. The PUF block integrates a Hardware Embedded Delay-based PUF which leverages delay variations within the functional unit. For key generation, an ECC engine operates on a binary field of size 2163. AES-128 has been employed for the encryption and decryption block to facilitate message encryption and decryption. Table 1 below illustrates the analysis of block speeds at various system clock rates.

Ultimately, the generated keys are stored in a volatile memory present on-chip. Upon initialization, the block RAM possesses a width of 128 bits and contains 512 elements. This

results in an 8KB BRAM overhead for key storage. Table 2 below outlines the overhead associated with standard CAN connection speeds.

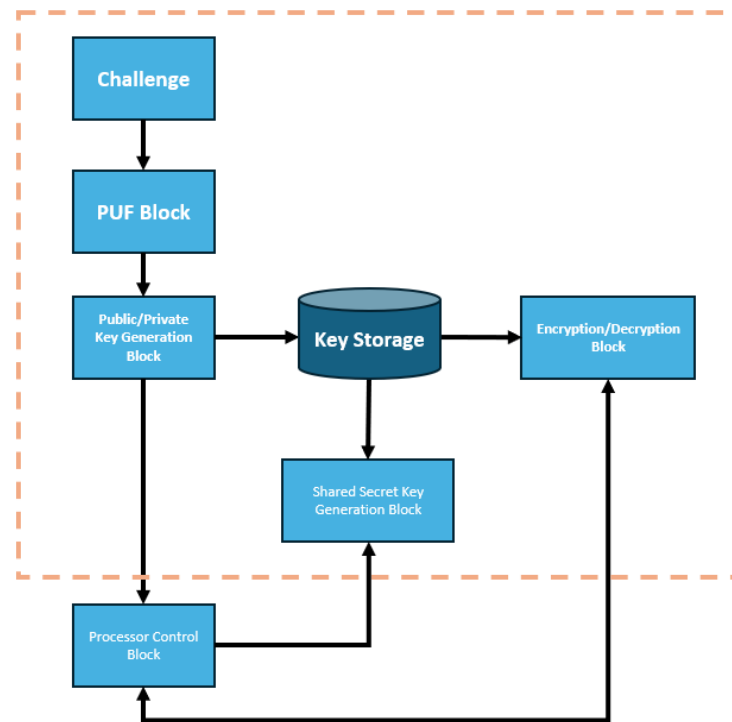


Figure 4. Secure Block Implementation at Each Client Node [1].

Table 1. AES-128 and ECC Operation Speeds on Different Clock Rates.

	AES-128	ECC Operation
At 60 MHZ	0.36666 μ s	23.8 μ s
At 100 MHZ	0.220 μ s	14.28 μ s
At 200 MHZ	0.11 μ s	7.14 μ s

Table 2. Overhead Overview at Standard CAN Connection Speeds with CAN-FD Speed of 8 Mbps in Normal Operation.

	125 Kbps	500 Kbps	1 Mbps
ECU Sending Encrypted Message to Server	256 μ s	76 μ s	24.2 μ s
Server's Reply	257.37 μ s	77.375 μ s	47.375 μ s

4.1.9. System Overview

Figure 4 delineates the design flow for each secured ECU. Every block synergistically collaborates to provide node enrollment within the system and to authenticate each individual node [1].

The enrollment process involves the following steps:

- The enrollment process begins with the storage of a challenge C for the PUF and a predefined set of parameters for the ECC curve during the manufacturing stage.
- All ECUs on the bus are prompted with the provided challenge and parameters, resulting in the generation of a distinctive and reproducible key, which functions as a private key.
- By utilizing the private key and some parameters as input to the private/public key generation block, a public key is generated.

- During each vehicle startup, the public keys of all ECUs are retained in a database for subsequent verification purposes.
- Should the generated public key of an ECU not align the anticipated value, the enrollment of that ECU fails.

The authentication process involves the following steps:

- Authentication ensues after the enrollment process, utilizing the PUF response, parameters, and public/private keys obtained during enrollment.
- At the sender node, the shared secret key is generated using the sender's public key and private key acquired during enrollment, employing the Shared Key Generation Block.
- The message is encrypted using the shared secret key and the receiver's public key.
- The encrypted message is transmitted, with the node ID of the current node set as the CAN message arbitration ID.
- Upon reception of the message, it is decrypted using the receiver's shared secret key.
- Concurrently, the public key of the sender's node is sent to the receiver's node for decryption, and the public key is verified from the database to ensure authentication.
- In summary, the enrollment process occurs every time a vehicle turns on, while authentication and encryption take place each time a message needs to be transmitted or received.

4.2. Blockchain Based Distributed Key Provisioning and Secure Communication over CAN FD

Bryson Shannon spearheads a blockchain-based distributed key provisioning and secure communication framework over CAN-FD, leveraging the Elliptic Curve Integrated Encryption Scheme for securing CAN-FD communication. This method harnesses Elliptic Curve Cryptography, which relies on the elliptic curve equation $y^2 = x^3 + ax + b$. The parameters a and b shape the elliptical curve, subsequently utilized in public and private key generation. This section provides a comprehensive examination of a prominent security strategy for CAN-FD [20].

4.2.1. Elliptic Curve Diffie-Helman

Elliptic Curve Diffie-Helman (ECDH) serves as the mechanism for generating secret keys employed in encrypting and decrypting messages between senders and receivers. In the scenario where Alice wishes to send a message to Bob, she encrypts it using her secret key along with Bob's public key. The public and private keys are derived from agreed-upon domain parameters on ECC. Alice and Bob exchange their public key over an insecure channel since the exposure of the public key does not aid the attacker in any way. Following this exchange, Alice computes a shared key using her private key and Bob's public key, which is then used to encrypt the message. Conversely, if Bob wants to decrypt Alice's message, he uses his private key and Alice's public key for decryption.

4.2.2. Elliptic Curve Integrated Encryption Scheme

The subsequent technique utilized is referred to as the Elliptic Curve Integrated Encryption Scheme (ECIES), serving as the methodology for encrypting and decrypting messages. In a scenario where Alice wants to transmit a message to Bob, she initiates communication by generating private and public keys using Elliptic Curve Cryptography. Alice utilizes her private key and Bob's public key to establish a shared key through a key agreement protocol. This shared key along with an optional parameter serves as input to the key derivation function. The resultant output consists of a MAC key and an encryption key. Subsequently, the encryption key is employed to encrypt the plaintext message. The MAC key, encrypted message, and an optional parameter are used as inputs in the hash function to generate a MAC tag. Alice dispatches her MAC tag, encrypted message, and public key to Bob. Refer to Figure 5 for the encryption process.

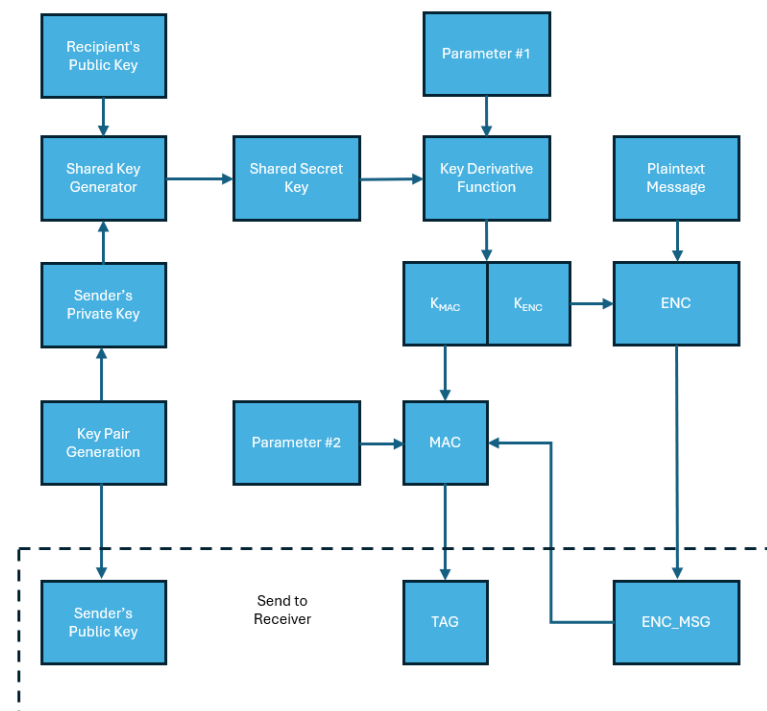


Figure 5. Sender Preparing to send the encrypted message along with MAC Tag and sender public key to help receiver decrypt the message.

4.2.3. Decryption Process

During the Decryption process, Bob receives from Alice the MAC tag, encrypted message, public key, and any optional parameters. Using his private key and Alice's public key, Bob establishes a shared key. The shared key, combined with an optional parameter key is used to derive both a decryption key and a MAC, which in turn generates a MAC tag. By comparing the MAC tag generated by Alice with the one computed by Bob, any potential tampering with the message can be detected. Finally, the decryption key is applied to decrypt the message. The decryption process is summarized in Figure 6 below.

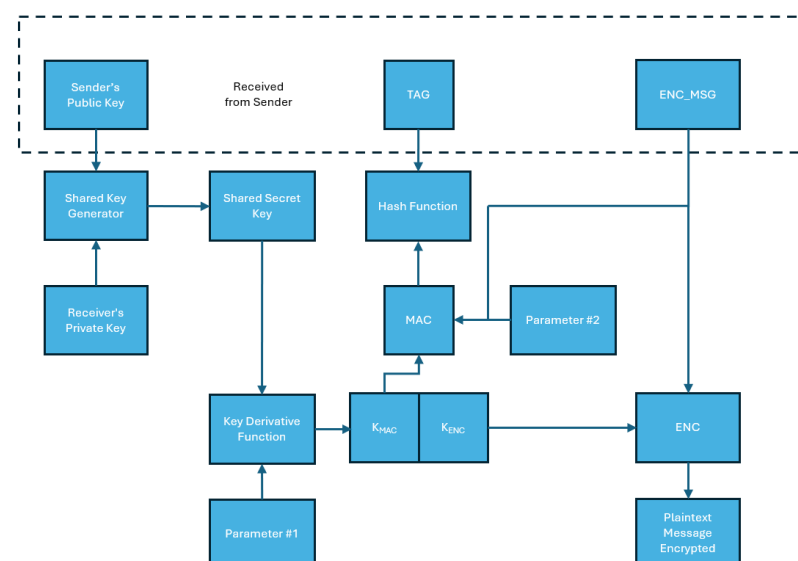


Figure 6. Decrypting a Message from the Sender.

4.2.4. Blockchain and Smart Contract

Blockchain is a revolutionary technology that serves as a decentralized and distributed peer-to-peer (P2P) network. At its core, it comprises a series of blocks interconnected through cryptographic hashing functions, forming a secure and immutable ledger of transactions. Each block in the chain contains essential information such as transaction data, timestamps, nonce, and hashes of the previous block, ensuring and maintaining the integrity and traceability of the entire system [21].

One of blockchain's fundamental principles is decentralization, meaning that the ledger is replicated across multiple nodes in the network. This redundancy fosters resilience and prevents a single point of failure. Additionally, blockchain operates on a consensus mechanism, where participating nodes collectively validate and agree upon the transactions, eliminating the need for a central authority.

The Ethereum blockchain, a prominent example of blockchain technology, goes beyond simple transaction recording. It supports the deployment of smart contracts, self-executing agreements with predefined conditions encoded directly onto the blockchain. These contracts enable automated and trustless execution of transactions, significantly reducing the reliance on intermediaries and third-party services.

Within the Ethereum ecosystem, nodes play a crucial role as individual participants responsible for maintaining a copy of the blockchain ledger and contributing to the consensus process. Blocks, the building blocks of the blockchain, consist of transaction data and metadata encapsulated within a block header. This header contains essential information like the block's hash, timestamp, and references to the previous block, ensuring the chronological integrity of the chain.

Smart contracts deployed on the Ethereum blockchain provide a powerful mechanism for enforcing rules and facilitating transactions without the need for intermediaries. By executing code autonomously, these contracts enable a wide range of applications, from decentralized finance to supply chain management, with unprecedented transparency and efficiency.

In the realm of security, blockchain technology offers robust solutions for key provisioning and authentication. By leveraging smart contracts and distributed ledgers, secure frameworks can be established to validate keys and prevent unauthorized access. Ethereum's Externally Owned Accounts (EOA) and contract accounts further bolster security by enabling controlled access to resources and executing predefined actions based on cryptographic signatures.

Figure 7 illustrates the structure of blockchain technology depicting a sequence of interconnected blocks, each representing a unit of data or a transaction. These blocks are linked together in chronological order, forming an immutable chain. Additionally, the diagram highlights the decentralized nature of the network, showcasing multiple nodes participating in the validation and maintenance of the blockchain ledger. Smart contracts may also be depicted, illustrating their role in automating and enforcing predefined rules within the blockchain ecosystem.

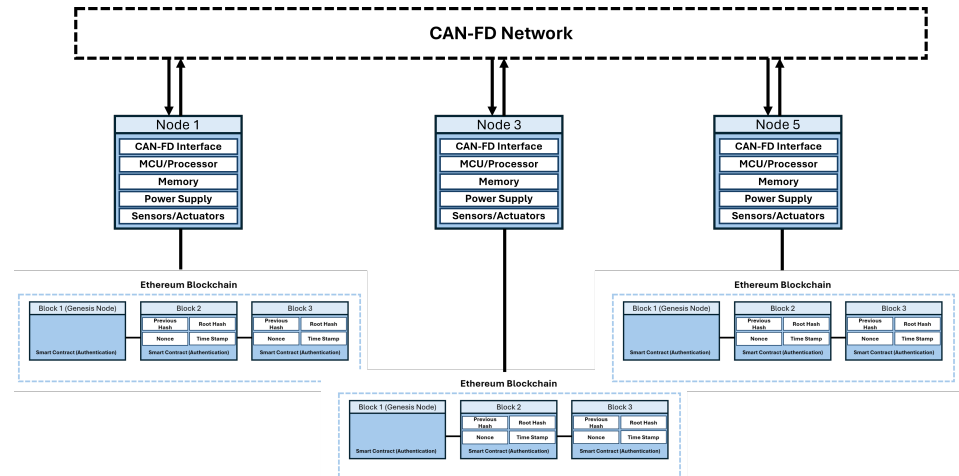


Figure 7. CAN-FD Blockchain Integration Architecture [6].

4.3. Secure Communication over CANBus

This work utilizes lightweight Physical Cloneable Function (PUF) based authentication encryption in conjunction with Elliptic Curve Cryptography and other methodologies to bolster the security of CANBus communication. The author has introduced a novel hardware-enriched electronic control unit design aimed at fortifying the security of the nodes on the CANBus network. This proposed framework adopts a server and client architecture. The server oversees all of the clients present on the network. Prior to initiating communication with other nodes, clients on the network undergo authentication by the server. The server and client undergo a registration process referred to as secure registration overseen by a trusted manufacturer during the construction of the vehicle to ensure the original trustworthiness and security of all components. Section 4.1 discusses the implementation of the PUF block on FPGA, with the objective of replicating this approach on an actual ECU using FPGA technology. The research covered in this section is derived from another novel framework titled “Secure Communication over CANBus” [22].

4.3.1. Physical Unclonable Function Block

The overall implementation is divided into five major blocks. The first block is dedicated to the Physical Unclonable Function (PUF) that is responsible for generating private and public keys for the ECU. PUF is a cryptographic methodology implemented on a physical layer. To better describe a PUF consider a scenario where two identical Integrated Circuit (IC) chips are created. Despite their identical characteristics and performance, inherent manufacturing variations lead to distinct responses. PUF’s utilize these process variations to generate reproducible and unpredictable keys. PUF’s are referred to as challenge-response pairs i.e., the challenge serves as the input to the PUF, which is generally a digital input like ‘0’ or ‘1’ and the response serves as the output which is also in the form of digital output. During the vehicle manufacturing process, all the ECU are challenged using a seed and some configuration parameters. This results in ECUs generating a unique private key that can be reproduced with the same challenge. All the helper data is stored within the Non-Volatile Memory (NVM) of the ECU for future key regeneration. The generated public key is securely stored in a server. Prior to utilization, all client nodes undergo enrollment and authentication processes to ensure secure operation.

4.3.2. Private/Public Key Generation Block

The second block is a Private/Public key Generation module, which facilitates asymmetric encryption and an Elliptic Curve Diffie-Hellman key exchange. For resource-constrained devices like the ECU, Elliptic Curve Cryptography is an optimal choice due to its low overhead requirements accompanied by security levels akin to RSA. This block utilizes elliptic curves endorsed by the National Institute of Standard and Technology

(NIST) that have been deemed safe and secure. Through ECC, public and private keys are generated after the registration is successful, a process initiated each time the vehicle is powered on. Any nodes failing to register during the registration process are automatically blacklisted for communication during the entirety of that session. To enhance security, private keys for all nodes are generated solely at startup and are not stored thereafter.

4.3.3. Shared Key Generation Block

The shared key Generation Block facilitates the generation of shared key enabling encrypted communication between the two nodes. Leveraging the Elliptic Curve Diffie-Hellman (ECDH) algorithm, the sender's private key, and the receiver's public key are combined to generate a shared secret key. All generated keys are securely stored in volatile memory to mitigate the risk of compromise.

4.3.4. Key Storage Block

The key storage block comprises of a non-volatile memory used to store seeds, challenges, and helper data for the PUFs data essential for PUFs to deliver random and reproducible responses. Additionally, Public Keys are also stored for server authentication and parameters necessary for generating public and private key pairs using ECC. In contrast, volatile memory deployed to stored shared secret key generated using ECDH used for encryption and decryption, private key for all nodes, and public key of the communicating node. This framework eliminates the necessity for the private key to be stored in the non-volatile memory as it can be generated on a need basis.

4.3.5. Encryption/Decryption Block

The final block used to complete this framework is the encryption/decryption block. This block utilizes the shared secret key to encrypt or decrypt any message using Elliptic Curve Integrated Encryption Scheme (ECIES). It also ensures that all messages are encrypted before being sent out and being decrypted after being received by the node [3]. Refer to Figure 2 to observe the building blocks present in this framework.

4.3.6. Arbiter PUF Based Alternate Framework Implementation

The Arbiter Physically Unclonable Function (PUF) stands out as a robust design with multiple Challenge-Response Pairs (CRPs) integrated, utilizing process variations in equivalent delay lines for response bit generation. In the realm of FPGA implementation, the conventional Arbiter PUF faces challenges as a result of place limitations and route capabilities. The use of the Arbiter PUF, as opposed to a HELP PUF, involves a similar implementation with a few minor changes. The performance and Security analysis outlined in the aforementioned "Performance and Security Analysis: FPGA CAN-FD Implementation" section can also be observed with this implementation. Implementation of the framework described in Section 4.1 utilizing an Arbiter PUF, has also been conducted and verified with favorable outcomes [23].

5. Evaluation of Security Measures and Result Analysis

5.1. Security and Performance Analysis: Embedded CAN-FD Implementation

The proposed design outlined in Section 5.1 utilizes a Raspberry-Pi with a CAN-FD shield, featuring two CAN-FD nodes operating at 200 MHz, each equipped with one receiver and transmitter. Analysis is performed based on a system clock speed of 200 MHz, corresponding to CAN speeds of 1 Mbps and 8 Mbps for CAN-FD. Tests are conducted to measure the transmission time required for data exchange. At 1 Mbps, the transmission time for one bit is approximately 1 μ s, whereas at 8 Mbps, it reduces to 0.125 μ s, excluding considerations like bit stuffing. For instance, transmitting 30 bits on a 1 Mbps connection typically takes around 30 μ s. These findings underscore the practical implications of system configuration on data transmission efficiency.

During authentication, the generation of public and private key pairs requires a point multiplication operation that takes around 7.14 μ s. Another point multiplication is required for the generation of a shared encryption key using a stored public key that takes around 110 ns. Therefore, the total time (Minimum) required for booting up and sending the shared key to the server is:

$$(7.14 \mu\text{s} \times 2) + 0.11 \mu\text{s} + (30 \mu\text{s} + (64 + 16) \times 0.125 \mu\text{s}) = 24.42 \mu\text{s}$$

For every server-authenticated node, a two-part message is sent by the server to all other connected nodes. 1st part contains the authenticated node's ID and the 2nd consists of the encrypted public key of the authenticated node for establishing shared key-based communication. The transmission time of these messages is:

$$30 \mu\text{s} + (11 + 128) \times 0.125 \mu\text{s} = 47.375 \mu\text{s}$$

The repeated process for node authentication includes encryption of the public key of an authorized node, within the message, 110 ns of decryption, and finally, the generation of the shared key that requires an additional 7.14 μ s by the ECC unit [1].

A single CAN-FD message needs encryption for a frame that allows up to 64 bytes and therefore requires only a single frame transmission.

This framework secures device-level and CAN bus data security by relying on a secure server for the enrollment of legitimate ECUs. HELP PUF is used for the detection and prevention of any invasive attacks with the potential of compromising key generation during boot-up. Isolation techniques safeguard keys from malicious access via CAN and prevent unauthorized code execution [1].

It also ensures secure data exchange by storing only the server's public key on ECU nodes, preventing probing attacks. In case of enrollment failure, retrieval of other secret keys is blocked, enhancing resilience against compromise. Malicious node connection attempts are swiftly thwarted due to authentication failure, as the server's records lack the public key of the malicious node. Moreover, any potential impact from compromised public keys is limited to public keys, safeguarding communication integrity.

5.1.1. Security and Performance Analysis: FPGA CAN-FD Implementation

The proposed FPGA based framework has undergone testing, leveraging Xilinx technology, with CAN-FD successfully implemented on the Kintex KC705 FPGA Evaluation Board, an accredited automotive device. Notably, the framework incorporates a Hardware Embedded Delay-based PUF which operates on delay variations present in the functional unit. For key generation, an ECC engine operates on a binary field of 2163. Additionally, AES-128 encryption and decryption blocks have been integrated to facilitate secure message transmission. Table 1 provides a detailed analysis of block speeds across varying system clock rates [1], showcasing the framework's performance under different configuration.

The keys are stored in volatile memory on-chip, with the block RAM having a width of 128 bits and containing 512 elements upon initialization. The key storage area overhead is 8KB BRAM. Table 2 illustrates the overhead for standard CAN connection speeds [1].

5.1.2. Key Exchange Protocol Execution: Findings and Outcomes

The experimental setup comprises NXP boards equipped with Dual CAN FD Shields, connected to a National Instruments CAN breakout box using female-to-female DB9 cables, thereby establishing a CAN FD bus network with three nodes. To align with the circuit design of a CAN bus network, the termination resistor of the second node is removed. The microcontroller board is capable of operating on an embedded operating system. Upon invocation of a board, both public and private keys are generated, and each node accesses the blockchain to retrieve keys of other nodes and communicate with shared keys on the bus, transmitting encrypted messages on the network. For Alice, the process involves maintaining a list of active party members (or nodes) on the CAN bus, enabling them to

send a request to each member via blockchain and smart contract on the bus to execute the key exchange protocol alongside her public key. The blockchain algorithm starts with Elliptic Curve Cryptography, secure key provisioning, and then validation as mentioned in Section 4.2. Figure 8 before shows successful communication between two nodes via the blockchain framework as mentioned in Section 4.2.

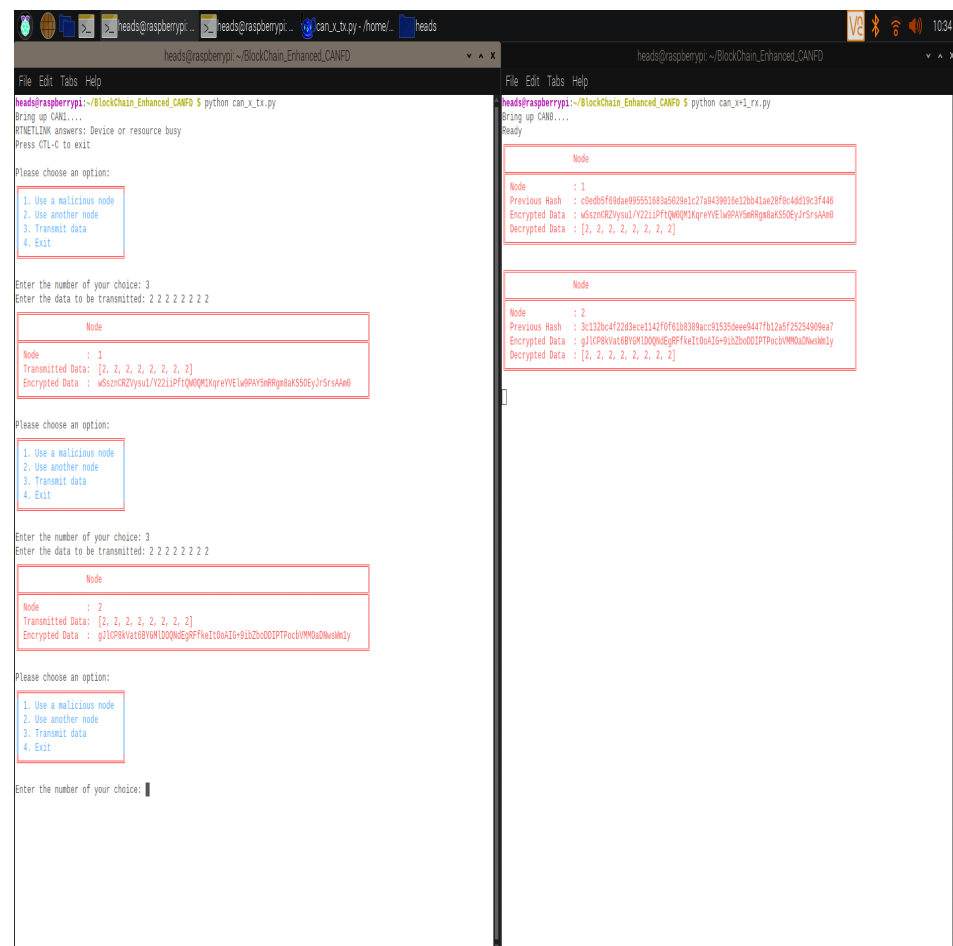


Figure 8. CAN-FD Communication via Blockchain [6].

6. Future Works

Currently, the most effective strategies to mitigate security concerns involve Physical Unclonable Functions (PUFs) for their uniqueness and reproducibility, alongside encryption/decryption methods for greater efficiency, combining PUFs with cryptography to ensure unparalleled encryption and developing lightweight blockchain solutions to enhance authentication capabilities. This fusion of encryption and authentication presents a promising approach to securing FPGAs.

Due to the limited capacity of ECUs, forward compatibility and future upgrades pose significant challenges, driving the promotion of FPGAs as a standard to control CAN-FD. Extensive tests are being performed to compare ECUs and FPGAs across various CAN bus scenarios to highlight FPGA advantages such as flexibility, high performance, integration and parallel processing. However, drawbacks such as cost, power consumption and development time necessitate ongoing efforts for mitigation.

In conclusion, as the automotive industry advances, it's essential to anticipate the development of CAN XL as the potential successor to CAN-FD. While CAN XL is not yet mainstream, there is a strong likelihood that it will become the preferred choice in the future. Therefore, all current and future endeavors should be conducted with an eye towards transitioning to CAN XL. This forward-thinking approach ensures that ongoing

efforts align with the evolving landscape of automotive communication protocols and positions and facilitates seamless integration and advancement in the automotive sector.

7. Discussion, Key Takeaways and Reflections

7.1. Limitations of CAN-FD

Hardware compatibility emerges as a limitation, particularly concerning older devices designed for Classical CAN. This constraint may restrict CAN-FD's deployment in certain applications or necessitate upgrades to hardware components for compatibility.

Additionally, implementing CAN-FD in resource-constrained environments, such as embedded systems or low-power devices, may pose challenges. The heightened complexity and data rates of CAN-FD may require more processing power, memory, and energy consumption, potentially limiting its applicability in certain scenarios.

Integrating CAN-FD into existing systems or networks may also prove challenging, especially in environments with mixed CAN and CAN-FD devices. Ensuring seamless interoperability and compatibility between different protocols and hardware platforms requires careful planning and execution.

Finally, cost considerations are significant. Upgrading to CAN-FD may entail additional costs, including hardware upgrades, software development, and system testing. Organizations must carefully evaluate the cost-benefit ratio and consider factors such as performance improvements, scalability, and long-term sustainability when adopting CAN-FD.

In summary, while CAN-FD offers significant advantages in terms of data rates, payload sizes, and flexibility, addressing the associated challenges and limitations is crucial to realizing its full potential and ensuring successful deployment in automotive and other industrial applications.

Security emerges as an important concern for the CAN-FD protocol as a result of its increased data rates and larger payload sizes. The enhanced capabilities expose the protocol to potential vulnerabilities that require attention to ensure secure communication. A report highlights the various techniques hackers use, including hacking the car's engine to obtain wireless key codes, unauthorized entry via on-board diagnostics, and employing USB injectors. These methods involve manipulating the car's CAN system, ultimately obtaining access to the car or interrupting its normal functionality [24]. Addressing these security concerns and potential vulnerabilities becomes preeminent to safeguard against possible breaches and ensure the integrity of CAN-FD protocol.

7.2. Summary of Findings

To summarize this investigation reveals the efficacy of the proposed frameworks in enhancing the security of Controller Area Network (CAN) communication, striking a commendable balance between increased security and minimal operational overhead, thus preserving optimal performance. The integration of various cryptographic techniques, including Physically Unclonable Function (PUF)-based authentication, Elliptic Curve Cryptography (ECC)-based key generation, and Elliptic Curve Integrated Encryption Scheme (ECIES) for encryption/decryption, collectively establishes a robust framework to ensure communication is secured among the nodes connected over the in-vehicular network.

7.3. Closing Thoughts

In closing, this journal has underscored the critical importance of securing the CAN-FD protocol and has provided valuable insights into vulnerabilities, current security solutions, and areas for improvement. By leveraging advanced cryptographic techniques and exploring innovative approaches such as the integration of Physically Unclonable Functions (PUFs) and lightweight blockchain technology, we are advancing towards a more secure and resilient in-vehicular communication framework.

As we navigate the evolving landscape of automotive communication protocols, it's imperative to remain proactive and forward-thinking. Embracing technologies like FPGAs and anticipating the transition to CAN XL ensures that our efforts are aligned with

future advancements and industry standards. By staying committed to ongoing research, collaboration, and innovation, we can continue to enhance the security, efficiency, and reliability of automotive systems, ultimately contributing to safer and more advanced transportation solutions for the future.

8. Conclusions

The aim of this journal was to emphasize the importance of securing CAN-FD protocol, cover vulnerabilities of CAN-FD, current security solutions and their drawbacks, best countermeasures currently available to implement and most importantly provide insight and share our ongoing research in order to bring the protocol close to state-of-the-art technology. All information up until this section provides an understanding of the current vulnerabilities and current countermeasures but like most engineering solutions, there is always room for improvement. The upcoming sections discuss some major flaws that still remain in the protocol along with some improvements to be made in order to bring this technology close to the state-of-the-art by improving forward compatibility, reducing overhead and complexity, hardware improvements, and finally some conclusions.

Author Contributions: Conceptualization, Resources, Data curation, writing—original draft preparation, writing—review and editing by N.L. Data curation, writing—original draft preparation, writing—review and editing by N.T. Data curation, Supervision, and writing—review and editing by F.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

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

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
ECU	Electronic Control Units
CAN	Controller Area Network
CAN-FD	Controller Area Network-Flexible Data Rate
WCTT	Worst-Case Transmission Time
PUF	Physical Unclonable Function
ECC	Elliptical Curve Cryptography
ECDH	Elliptical Curve Diffie-Helman
AES	Advanced Encryption Standard

References

1. Siddiqui, A.S.; Lee, C.-C.; Che, W.; Plusquellic, J.; Saqib, F. Secure intra-vehicular communication over CANFD. In Proceedings of the 2017 Asian Hardware Oriented Security and Trust Symposium, Beijing, China, 19–20 October 2017; pp. 97–102.
2. Oladimeji, D.; Rasheed, A.; Varol, C.; Baza, M.; Alshahrani, H.; Baz, A. CANAttack: Assessing Vulnerabilities within Controller Area Network. *Sensors* **2023**, *23*, 8223. [CrossRef] [PubMed]
3. Wired. Available online: <https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/> (accessed on 25 February 2024).
4. 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, Oakland, CA, USA, 16–19 May 2010; pp. 447–462.
5. National Instruments. Available online: <https://www.ni.com/en/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/understanding-can-with-flexible-data-rate--can-fd-.html> (accessed on 25 February 2024).
6. Agrawal, M.; Huang, T.; Zhou, J.; Change, D. CAN-FD-Sec: Improving Security of CAN-FD Protocol. In *Security and Safety Interplay of Intelligent Software Systems, Proceedings of the ESORICS 2018 International Workshops, ISSA 2018 and CSITS 2018, Barcelona, Spain, 6–7 September 2018*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 77–93.
7. CAN and CAN FD Protocol. Available online: <https://www.youtube.com/watch?v=2Mhqwt2xTxk> (accessed on 8 April 2024).

8. Xie, Y.; Huang, P.; He, Y. Comparison between CAN and CAN FD: A Quantified Approach. In Proceedings of the 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC), Guangzhou, China, 12–15 December 2017; pp. 1399–1403.
9. AutoPi. Available online: <https://www.autopi.io/blog/can-fd-explained> (accessed on 10 April 2024).
10. Xie, G.; Yang, L.T.; Liu, Y.; Luo, H.; Peng, X.; Li, R. Security Enhancement for Real-Time Independent In-Vehicle CAN-FD Messages in Vehicular Networks. *IEEE Trans. Veh. Technol.* **2021**, *70*, 5244–5253. [[CrossRef](#)]
11. Alfardus, A.; Rawat D.B. Evaluation of CAN Bus Security Vulnerabilities and Potential Solutions. In Proceedings of the 2023 Sixth International Conference of Women in Data Science at Prince Sultan University (WiDS PSU), Riyadh, Saudi Arabia, 14–15 March 2023; pp. 90–97.
12. Syed Ahamed, M.F.; Tewolde, G.; Kwon, J. Software-in-the-Loop Modeling and Simulation Framework for Autonomous Vehicles. In Proceedings of the 2018 IEEE International Conference on Electro/Information Technology (EIT), Rochester, MI, USA, 3–5 May 2018; pp. 0305–0310.
13. 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, San Francisco, CA, USA, 10–12 August 2011; pp. 1–16.
14. Palanca, A.; Evenchick, D.E.; Maggi, F.; Zanero, S. A Stealth, Selective, Link-Layer Denial-of-Service Attack Against Automotive Networks. In *Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA 2017*; Springer: Cham, Switzerland, 2017; Volume 10327.
15. Jia, J.; Xu, Y.; Wu, Y.; Xu, Y.; Fan, Y.; Zhang D. Intelligent Connected Vehicle CAN-FD Bus Network Security Protocol. In Proceedings of the 2023 International Conference on Mobile Internet, Cloud Computing and Information Security, Nanjing, China, 7–9 April 2023; pp. 57–62.
16. Woo, S.; Jo, H.J.; Wu, Y.; Lee, D.H. A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 993–1006. [[CrossRef](#)]
17. Carel, G.; Isshiki, R.; Kusaka, T.; Nogami, Y.; Araki, S. Design of a Message Authentication Protocol for CAN FD Based on Chaskey Lightweight MAC. In Proceedings of the 2018 Sixth International Symposium on Computing and Networking Workshops, Takayama, Japan, 27–30 November 2018; pp. 267–271.
18. Lagnf, F.M.E.; Ganesan, S. Securing CAN FD by implementing AES-128, SHA256, and Message Counter based on FPGA. In Proceedings of the IEEE International Conference on Electro Information Technology, Mt. Pleasant, MI, USA, 14–15 May 2021; pp. 091–096.
19. Happel, A. Secure Communication for CAN FD. Available online: https://cdn.vector.com/cms/content/know-how/_technicalarticles/Security_CAN_Newsletter_201411_PressArticle_EN.pdf (accessed on 15 May 2024).
20. Shannon, B.; Etikala, S.; Gui, Y.; Siddiqui, A.S.; Saqib, F. Blockchain Based Distributed Key Provisioning and Secure Communication over CAN FD. In Proceedings of the 2019 IEEE Computer Society Annual Symposium on VLSI, Miami, FL, USA, 15–17 July 2019; pp. 638–644.
21. Sapra, R.; Dhaliwal, P. Blockchain: The new era of Technology. In Proceedings of the 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC), Solan, India, 20–22 December 2018; pp. 495–499.
22. Siddiqui, A.S.; Gui, Y.; Plusquellic, J.; Saqib, F. Secure communication over CANBus. In Proceedings of the 2017 IEEE 60th International Midwest Symposium on Circuits and Systems, Boston, MA, USA, 6–9 August 2017; pp. 1264–1267.
23. Siddiqui, A.S.; Gui, Y.; Plusquellic, J.; Saqib, F. A Secure Communication Framework for ECUs. *Adv. Sci. Technol. Eng. Syst. J.* **2017**, *2*, 1307–1313. [[CrossRef](#)]
24. CISO. Available online: <https://ciso.economictimes.indiatimes.com/news/vulnerabilities-exploits/from-hacking-wireless-keys-to-hijacking-the-network-car-theft-becomes-high-tech/102965321> (accessed on 25 February 2024).

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.