electronics

Article

Enhancing Communication Security an In-Vehicle Wireless Sensor Network

Algimantas Venčkauskas *,†, Marius Taparauskas, Šarūnas Grigaliūnas † and Rasa Brūzgienė

Department of Computer Sciences, Kaunas University of Technology, Studentu Str. 50, 51368 Kaunas, Lithuania; taparauskasmarius1998@gmail.com (M.T.); sarunas.grigaliunas@ktu.lt (Š.G.); rasa.bruzgiene@ktu.lt (R.B.)

* Correspondence: algimantas.venckauskas@ktu.lt

† These authors contributed equally to this work.

Abstract: Confronting the challenges of securing communication in-vehicle wireless sensor networks demands innovative solutions, particularly as vehicles become more interconnected. This paper proposes a tailored communication security framework for in-vehicle wireless sensor networks, addressing both scientific and technical challenges through effective encryption methods. It segments the local vehicle network into independent subsystems communicating via encrypted and authenticated tunnels, enhancing automotive system safety and integrity. The authors introduce a process for periodic cryptographic key exchanges, ensuring secure communication and confidentiality in key generation without disclosing parameters. Additionally, an authentication technique utilizing the sender's message authentication code secures communication tunnels, significantly advancing automotive cybersecurity and interconnectivity protection. Through a series of steps, including key generation, sending, and cryptographic key exchange, energy costs were investigated and compared with DTLS and TLS methods. For cryptographic security, testing against brute-force attacks and analysis of potential vulnerabilities in the AES-CBC 128 encryption algorithm, HMAC authentication, and HKDF key derivation function were carried out. Additionally, an evaluation of the memory resource consumption of the DTLS and TLS protocols was compared with the proposed solution. This work is crucial for mitigating risks associated with in-vehicle communication compromises within smart cities.

Keywords: in-vehicle WSN; communication security; cryptographic key; MAC; DTLS; TLS

Citation: Venčkauskas, A.; Taparauskas, M.; Grigaliūnas, Š.; Brūzgienė, R. Enhancing Communication Security an In-Vehicle Wireless Sensor Network. *Electronics* **2024**, *13*, 1003. https://doi.org/10.3390/electronics13061003

Academic Editor: Christos J. Bouras

Received: 14 February 2024
Revised: 25 February 2024
Accepted: 1 March 2024
Published: 7 March 2024

1. Introduction

Today's emphasis on cybersecurity in the realm of vehicles cannot be overstated. As we witness the rapid evolution and integration of computer technologies into vehicles, we are also seeing the creation of complex networks that manage crucial systems. These systems, which include aspects like braking [1], lighting [2], and engine control [3], rely heavily on data from various sensors. The threat of compromising these in-vehicle technologies is not just a theoretical risk; it has real-world implications for the safety of both those inside the vehicle and pedestrians alike. Moreover, the advent of wireless sensors [4] has opened new doors for hackers, making our vehicles more vulnerable to cyber-attacks. Adding to this complexity is the ability of smart devices to connect to a vehicle's network, which introduces yet another risk for potential breaches, endangering the privacy of users.

The adoption of wireless sensor networks (WSNs) in vehicle systems represents a transformative development in the automotive sector, broadening the scope of functionalities from simple monitoring and control to facilitating sophisticated driver assistance and self-driving features. Yet, this fusion of technology introduces significant security concerns that demand thorough attention to safeguard the systems' safety, privacy, and dependability. The critical role of encryption in automotive WSNs is highlighted by the imperative to guard sensitive data, uphold the integrity of the data being exchanged, and secure communication across networked devices [5,6].

In the context of smart cities, such vehicles play a pivotal role in enhancing urban mobility, reducing traffic congestion, and improving environmental sustainability [7]. Secure communication within these vehicle systems and their interaction with the urban infrastructure is paramount, as it not only ensures the efficient operation of transportation networks but also underpins the safety and privacy of the city's inhabitants. Secure in-vehicle communication systems are integral to the realization of fully connected and automated urban environments, where vehicles and city systems operate in harmony to optimize city life [8].

Securing data through encryption is essential. As vehicles constantly collect and share sensitive information, including their location [9], speed, and the driver's habits, it is imperative to implement strict protocols to protect these data from falling into the wrong hands. Moreover, keeping the data intact as they are transmitted is critical for the smooth operation of vehicle systems. Employing encryption and digital signatures plays a key role in ensuring the data are not altered during their journey, protecting against attacks aimed at tampering with the data, which could put vehicle safety at risk. The complexity and scalability of networks, particularly with the emergence of Vehicle-to-Everything (V2X) systems, compound the difficulty of managing encryption keys and protocols [10]. Additionally, the real-time data transmission required by many automotive applications necessitates efficient encryption processes that do not introduce undue latency.

Nevertheless, integrating encryption within automotive wireless sensor networks presents significant challenges. The computational capabilities and resources of sensor nodes in these networks are often limited, making it difficult to implement strong encryption techniques without negatively impacting system performance. The sensor nodes must endure harsher conditions than typical stationary or portable computers, including higher temperatures and vibrations, with the added constraint of vehicles having relatively modest computing systems and power supplies. These compact and power-limited components constrain processing capabilities, increasing the vulnerability of vehicle security systems. In environments where resources are scarce, employing long cryptographic keys or sophisticated algorithms can significantly slow down system operations, potentially to a detrimental extent. Moreover, vehicle networks can be compromised through various means, such as wireless communications or internal vehicle hardware, posing a risk to user safety. The manipulation of data or signals by compromised sensors can interfere with vehicle functionality, endangering both passengers and pedestrians [11].

Moreover, it is important to note that the network within a vehicle does not undergo regular updates like standard computer systems do. Consequently, if new threats emerge or existing vulnerabilities are discovered, addressing these issues through software updates may not be feasible. Additionally, the in-vehicle network's complexity, composed of numerous subsystems each with their own security flaws, further complicates the situation. With the increasing prevalence of consumers' smart devices, there is a potential for these gadgets to connect to the vehicle's internal network, thus introducing new security challenges [12]. Inadequate security measures for communications between user devices and the vehicle's network can lead to breaches, either externally or through the user's own device. Such access does not limit itself to specific functionalities but might extend over the entire in-vehicle network. This poses a risk not only to the user's private information but also to their physical well-being, should control over the vehicle's sensor-controlled systems be compromised.

The use of wireless sensor networks for in-vehicle communication introduces various security issues. The data from sensors are transmitted omnidirectionally, meaning that with the right equipment and proximity, an unauthorized individual could intercept this information. This risk escalates if the sensor data are unencrypted, as reverse engineering could then be employed to disrupt the system's operations by injecting false data masquerading as legitimate sensor input. Furthermore, if the network's subsystems are interconnected without adequate segregation, compromising one part could lead to a breach of the entire system. Additionally, a system that connects to any device without verifying

its authenticity is vulnerable to intrusion by external devices, posing a significant threat to the network's integrity.

Confronting these challenges demands innovative solutions. With vehicles increasingly becoming interconnected, ensuring the security of WSN communications through effective encryption is not merely a technical requirement but a scientific problem in ensuring automotive safety and reliability. To overcome most concerns, this paper aims to develop a tailored communication security framework for an in-vehicle wireless sensor network. This framework will tackle the scientific and technical challenges of securing WSN communications through effective encryption methods. By segmenting the local vehicle network into independent subsystems that communicate via encrypted and authenticated tunnels, this work seeks to ensure the safety and integrity of automotive systems, address critical concerns in automotive cybersecurity, and contribute to the advancement of vehicle interconnectivity and protection.

The main authors' contribution in this work is as follows:

- We proposed a process for the periodic exchange of cryptographic keys to ensure secure communication between different subsystems;
- The framework ensures confidentiality in the creation of cryptographic keys by not disclosing the parameters involved in their generation. When a new session key is transmitted, it is done so in encrypted form, safeguarding it across the network;
- An authentication technique is used for separate, independent network subsystems that is facilitated by the use of the sender's message authentication code (MAC). This ensures authentication within the communication tunnel between parties, enhancing security and integrity.

Implementing these contributions is paramount to protecting against the risks associated with the compromise of in-vehicle communication, which endangers the well-being of both vehicle occupants and pedestrians.

The structure of this paper is as follows. Section 2 provides a review of the existing scientific works that analyzes security techniques in wireless sensor networks and in-vehicle WSNs. Section 3 details the proposed secure communication framework in-vehicle wireless sensor network. The scenario for the experiments and the research methodologies are provided in Section 4. The validation of the proposed framework along with an analysis of the validation results is presented in Section 5. Discussions on the importance, limitations of our proposed framework as well as directions for future works can be found in Section 6. Finally, Section 7 concludes the paper by summarizing the key findings.

2. Related Works

Given the importance of wireless sensor networks in the automotive industry, particularly in facilitating communication between a number of sensors and control units within a vehicle, safety issues are paramount.

Researchers have explored numerous innovative approaches to address the intricate challenges of achieving robust, efficient, and flexible security in wireless sensor networks. One significant advancement is the integration of blockchain technology into WSNs, as detailed in [13]. This method leverages blockchain's decentralized nature and resistance to tampering to enhance data security and integrity within WSNs. The research combines blockchain technology with data transmission to create a very secure network architecture for small-scale wireless sensor networks. Each network has a central node for gathering data, known as a "mobile database", which relies on embedded microcontrollers for data handling. By incorporating the decentralized and secure features of blockchain, the study aims to enhance the protection of communications within these networks. Nevertheless, this approach faces challenges related to high computational and energy demands, especially in settings with limited resources, potentially affecting the system's practical efficiency and scalability.

A research article featured in [14] delves into improving how smart solar power systems are managed and connected by applying physical layer security within WSNs. The study highlights the crucial role of protecting the physical layer from unauthorized access and eavesdropping, suggesting that robust security can be achieved through the use of specialized equipment and methods. However, the complexity and the need for particular hardware present significant challenges to the widespread adoption of these systems.

The study referenced as [15] introduces a self-adjusting approach for optimizing the coverage of wireless sensor networks. This technique dynamically adapts to enhance intrusion tolerance by factoring in trust metrics. Its natural capacity to adjust to varying circumstances helps maintain its integrity, guaranteeing secure and reliable network coverage even under malicious attacks. Nonetheless, its reliance on trust metrics introduces a vulnerability to attacks aimed at manipulating these values, indicating areas that require further refinement.

The research documented in [16] introduces a specialized technique in symmetric cryptography aimed at bolstering security within wireless sensor networks. This form of cryptography is noted for its high efficiency and simplicity, making it particularly well-suited for use in WSN nodes that operate with limited resources. Despite its advantages, the technique faces challenges related to the complexity of key management and distribution, especially when deployed on a large scale.

Additionally, another study in [17] discusses a secure and energy-efficient authentication strategy for WSNs, grounded in symmetric cryptography. This strategy skillfully balances the demands of security and energy conservation. It underscores the inherent challenges in securing WSNs, such as their susceptibility to physical attacks and the complexities of wireless communication. The study proposes a lightweight authentication protocol that efficiently conserves energy while ensuring secure communication between nodes.

Another noteworthy contribution comes from research published in [18], which delves into the characteristics and detection methods of Distributed Denial of Service (DDoS) attacks on WSNs, with a focus on vehicular networks. This study sheds light on the evolving nature of cyber threats and underscores the necessity for flexible and strong security measures to protect against these risks.

Lastly, the research in [19] investigates the practical challenges and solutions in deploying monitoring systems based on WSNs. It offers insight into the limitations and hurdles faced by WSNs, such as limited resources and susceptibility to external factors, providing valuable perspectives on the practical considerations for implementing secure sensor networks. The authors of the study in [20] conducted a statistical analysis on the network security issues faced by IT organisations, providing concrete evidence of the broader impact that security challenges in wireless sensor networks have on the business world.

In another piece of research [21] the optimization of access strategies for security in C-V2X (Cellular Vehicle-to-Everything) compute offloading networks was explored. This study highlights the difficulties arising from incomplete channel state information (CSI), which plays a crucial role in understanding how to ensure secure and efficient offloading in vehicular networks. Such advancements are critical for supporting the real-time data exchange and processing demands of modern vehicular systems.

Furthermore, the application of Artificial Intelligence (AI) in enhancing security is examined in the study [22], where the use of deep Q-learning to bolster the security of cellular V2X communications is investigated. This research marks a significant step forward in employing AI to bolster defence mechanisms against sophisticated cyber threats, thereby ensuring the integrity and reliability of vehicular communication systems.

Enhancing communication security in in-vehicle WSNs calls for interdisciplinary research efforts that span automotive engineering, cybersecurity, and information technology. A comprehensive approach that considers the technical, functional, safety, and privacy aspects of security is essential for developing effective solutions. Addressing the challenges of enhancing communication security for in-vehicle WSNs demands a proactive, adaptive research technique that anticipates future challenges, explores the integration of novel tech-

nologies, and promotes interdisciplinary collaboration. Traditional security frameworks need to be changed and improved to work with the changing and limited environment of in-vehicle WSNs. One way to do this could be to create lightweight encryption methods and effective key management systems.

While existing research, such as the integration of blockchain technology and physical layer security, aims to enhance data security and network integrity, these methods often face challenges related to high computational and energy demands, limiting their practical application in resource-constrained environments like vehicles. In contrast, the proposed framework focuses on the efficient management of cryptographic keys and network segmentation to ensure secure communication between subsystems without imposing significant resource overheads. Our approach addresses the unique requirements of in-vehicle networks, offering a tailored solution that balances security with the constraints of embedded automotive systems.

3. Proposed Secure Communication Framework In-Vehicle

The proposed framework is designed to address the weaknesses of today's common protocols in IoT systems. The Datagram Transport Layer Security (DTLS) protocol, which is based on the TLS protocol and is widely used today, requires X.509 certificates in order to authenticate the server and the client in a secure and reliable way. In embedded IoT systems, this causes several problems:

- Memory resources: Each device in the network needs a certificate to successfully establish communication, so sensors and their subsystems must maintain sufficient memory. This increases manufacturing costs and device complexity.
- Certificate management: Since every device on the network must have a certificate, the system must implement certificate management, revocation, generation of new certificates, and more.
- Speed: Because certificates use public key cryptography, one private key and one public key are used for each device on the system network. This increases the number of steps required to perform cryptographic functions. Also, in public key cryptography, keys are by default longer, which increases the time required for encryption and decryption. Key management and additional cryptographic functions require more computing power than symmetric cryptography. For these reasons, resource constraints in embedded systems, in particular in vehicle embedded systems, can lead to network slowdowns and packet delays. Also, due to the increased number of functions, embedded system devices may run out of memory resources and use up battery power faster.

With regard to other protocols, such as MQTT, it is noticeable that X.509 certificates are also used for security, so the same problems remain. While the proposed protocol addresses the problem of wireless network security for embedded systems in a resource-constrained vehicular environment, the resource issues listed above make the protocols in widespread use today ineffective.

The proposed solution consists of one main system module, which is responsible for the operation of the system, the collection of data from the sensors wirelessly, several different sensor subsystems, and one key management module responsible for generating and updating session keys. The initial vision of the proposed solution is as follows in Figure 1.

The sensor subsystem consists of a subsystem controller wired to a wireless communication transceiver, which enables the controller to communicate with the sensors and the host system controller (Figure 2). The sensor also includes a controller and a wireless communication transmitter and receiver.

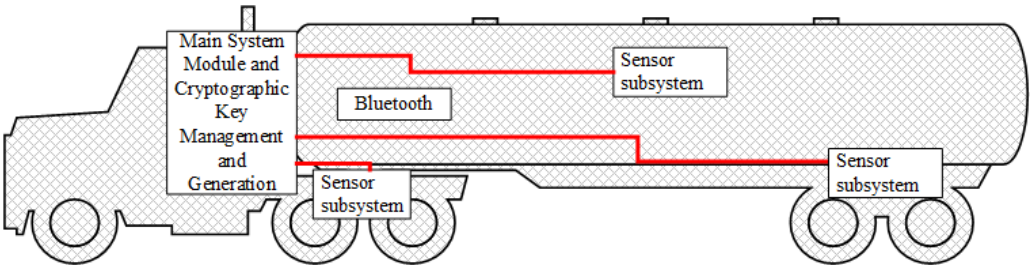

Figure 1. The general concept of the communication security framework for in-vehicle wireless sensor network.

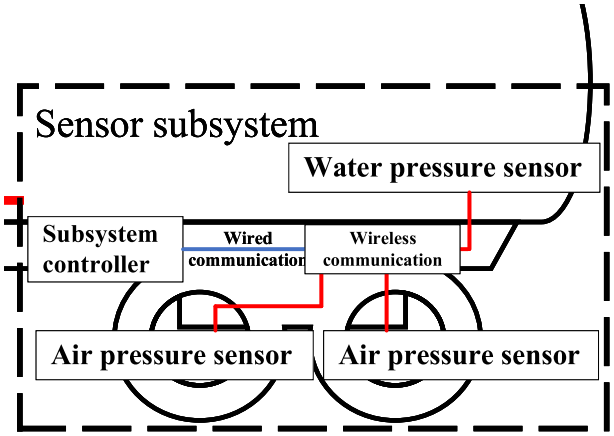

Figure 2. The wireless communication system.

The controller in this case manages the communication transceiver, the sensor data, and the communication with the subsystem controller itself (Figure 3). The main module of the system consists of a master controller wired to a cryptographic key management controller and a wireless communication transceiver.

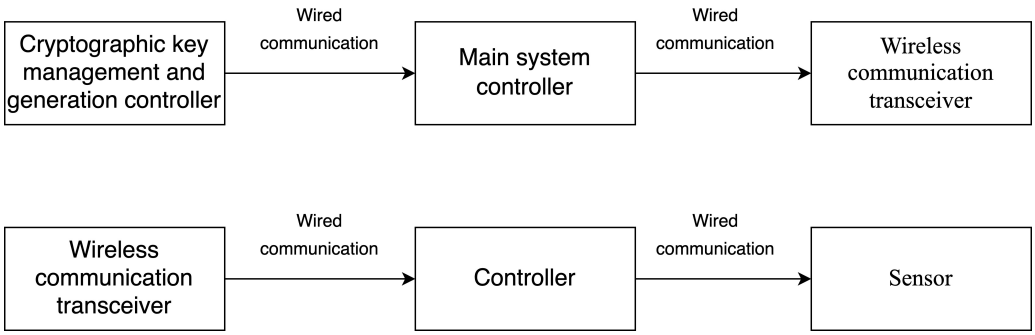

Figure 3. A diagram of the flow of a wireless communication system.

The cryptographic key management and generation controller is responsible for generating and managing the cryptographic keys that are used to secure the communication between devices. The wired communication controller is responsible for managing the wired communication between the system and other devices [23]. The wireless communication transceiver is responsible for transmitting and receiving wireless signals. The sensor is a device that detects changes in the environment and sends this information to the controller. The controller is a device that processes the information from the sensor and other devices and takes appropriate action [24].

The proposed framework for enhancing communication security in an in-vehicle wireless sensor network aims to address the vulnerabilities present in current IoT systems. One of the key aspects of the framework involves the utilization of the Datagram Transport Layer Security (DTLS) protocol. DTLS is a derivative of the Transport Layer Security (TLS)

protocol, specifically designed to operate in datagram transport environments. Unlike TLS, which is connection-oriented, DTLS is connectionless and provides security at the transport layer for datagram protocols such as User Datagram Protocol (UDP).

3.1. Authentication Technique

In the proposed solution, a controller is responsible for the management of cryptographic keys. The controller must periodically generate new session keys for each subsystem of the system and provide them to the host system controller, which stores and forwards these keys to the appropriate subsystem.

The session keys are generated each time by recalculating random numbers. When session keys are generated for a subsystem, they are encrypted with the system master key, which is pre-installed in each part of the system. In addition, a MAC value is calculated for the session keys, for which the cryptographic key management controller expects a response. As a result, when a subsystem of the system receives a packet containing cryptographic session keys, the subsystem controller shall check the MAC value received. If the MAC value is authenticated, the controller shall calculate a new MAC response value confirming successful receipt of the cryptographic keys. The MAC value shall be sent to the cryptographic key management controller, which shall verify the received MAC response value, and if it is successfully authenticated, the key exchange with the corresponding subsystem is complete. If the MAC response message is not successfully authenticated, the cryptographic key management controller shall generate new session keys, and the cycle shall repeat until the received MAC response message is authenticated (Figure 4).

Each subsystem shall check the MAC values of the received packets and drop the packet if the MAC value is not successfully authenticated. In addition, each subsystem shall perform decryption of the MAC value upon successful authentication, perform encryption prior to sending the data, and perform MAC counting upon successful encryption.

In order to ensure that the periodic change of cryptographic session keys is performed without delays, it has been chosen to generate the keys before the initialization of the key exchange. With this decision, the system will always store the new session keys generated, and they can be sent immediately. As the network load can be high depending on the usage of the system, this solution will not interfere with data transmission.

3.2. Parameters for Secure Key Generation, Transmission and Communication

In the proposed solution, the cryptographic key management controller is responsible for the generation and transmission of keys, but to do so securely requires key generation parameters that are not publicly available or transmitted over the network. This is done by defining the parameters necessary to ensure secure key generation, transmission, and communication:

1. Identification: Each part of the system and subsystem that communicates with the system network must have its own identification value in order to differentiate between subsystems.
2. Message counter: This is a counter storing the number of messages sent, used as a value to be concatenated with the session keys to synchronize communication between the two subsystems and to override the encryption of the data sent by the session. The reason for changing the encryption is that, when sending the same data, the encryption algorithm will send the same encrypted bit sequence, which an attacker could use, but changing the encryption based on the message counter means that the same unencrypted plaintext message will be encrypted differently each time.
3. Master key: The master key is used to derive all session keys periodically. This key does not change, is not sent, and does not encrypt data. The key itself shall be generated and stored at the time of installation of the cryptographic key management controller on the system or at the time of change. This key shall be stored by all controllers communicating in the system and shall be used to encrypt and decrypt session keys.

4. Session key: Each subsystem communicating with another part of the system must have a session key for that communication tunnel. A system shall not have two identical session keys that are used by more than one subsystem. One subsystem may use the same session key with all sensors in the subsystem, but the session key must be different when communicating with another subsystem. The session keys themselves will be changed periodically to protect against possible brute force attacks to discover session keys.
5. Unencrypted text: Each part of the system prepares the data for transmission in unencrypted text.
6. Encrypted text: Each part of the system must encrypt all data sent with the session keys stored at the time before sending the data.
7. Key flow is a key flow used to encrypt data using Boolean algebra functions. The stream is generated from the session key.
8. MAC value (Message Authentication Code): This is used to authenticate the sides of the communication tunnel. The MAC value must be enumerated for each packet sent.
9. Parameters for key generation: Each time a key is generated, random parameters are created and used to derive the key from the master key.

The following functions are required for key generation, encryption, and communication security:

- Session key: Each subsystem communicating with another part of the system must have a session key for that communication tunnel. A system cannot use two session keys that are identical in more than one subsystem. One subsystem may use the same session key with all sensors in the subsystem, but the session key must be different when communicating with another subsystem. The session keys themselves will be changed periodically to protect against possible brute force attacks to discover session keys.
- Unencrypted text: Each part of the system prepares the data for transmission in unencrypted text.
- Encrypted text: Each part of the system must encrypt all data sent with the session keys stored at the time before sending the data.
- Key flow is a key flow used to encrypt data using Boolean algebra functions. The stream is generated from the session key.
- MAC value (Message Authentication Code): This is used to authenticate the sides of the communication tunnel. The MAC value must be enumerated for each packet sent.
- Parameters for key generation: Each time a key is generated, random parameters are created and used to derive the key from the master key.

The defined parameters and functions are used to periodically generate cryptographic keys for all devices in the system network. Each device in the subsystem communicates with only one controller in the subsystem, so that devices in the subsystem may use the same cryptographic key during the life cycle of that cryptographic key. After a defined period of time, this key shall be replaced on all devices in the subsystem. This solution saves memory resources for the subsystem controller, as it eliminates the need to store separate keys for each subsystem device. In addition, the subsystem controller shall have another cryptographic key to establish communication with the next device in the network chain. This separates the subnetwork of one subsystem device from the subnetwork of several subsystems. Cryptographic keys are changed periodically to prevent a possible brute-force attack on the cryptographic key. Since cryptographic keys are used in the network instead of passwords, they are longer, which means that a brute force attack takes longer to succeed. Therefore, the success of a brute force attack on the former key does not pose a threat to the system when cryptographic keys are changed periodically. Also, the addition of salt during encryption solves the problem of sending the same unchanged data. In this case, the ciphertext is unchanged, but using a salt value based on the packet counter when encrypting the same data value results in an unequal result.

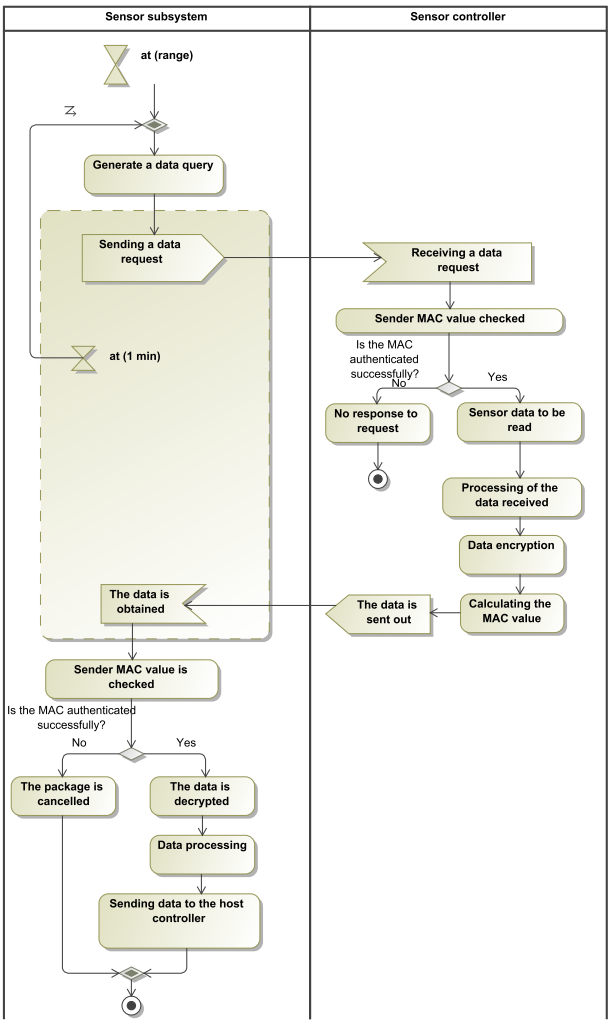

Figure 4. The algorithm for the management of the authentication.

3.3. Architecture for the Secure Communication In-Vehicle

In the proposed solution, the core module of the system is responsible for processing and storing the data received from all subsystems (Figure 5). This module includes a cryptographic key management controller, a master system controller, a wireless communication transceiver, and a memory module for the database. The database is designed to store all the information needed to operate the system. Since the database is accessible only by the host controller and the cryptographic key management controller and does not communicate with external objects, it can store the cryptographic keys of all the subsystems and the parameters necessary for key generation.

The cryptographic key management controller can only access the database by sending commands to the host controller, which checks the requests to prevent incorrect requests from being executed and takes action on them. The wireless transceiver is designed to communicate with all subsystems of the system. The host controller must check the MAC values of each received packet to determine whether the sender is authenticated to the system. If no errors are found, the controller will decrypt the received data and record it in a database. As in other parts of the system, the cryptographic keys shall be re-generated each time the system is started.

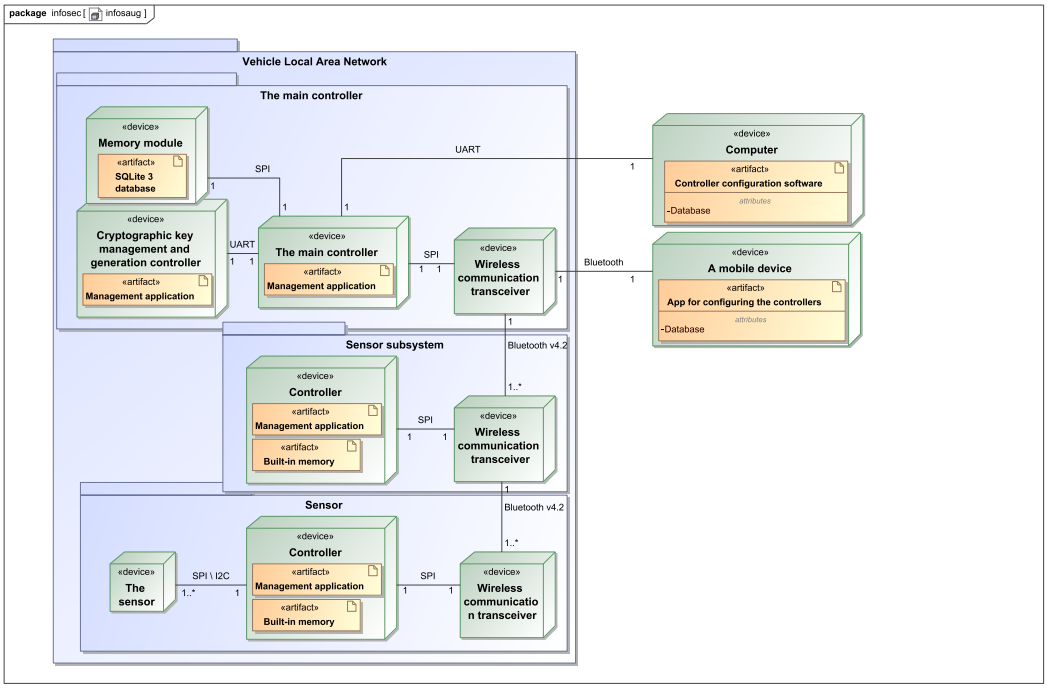

Figure 5. The architecture of the communication between different subsystems.

4. Experimental Case Study

4.1. Design of the Prototype of the Proposed Framework

The proposed solution outperforms the DTLS protocol analyzed in terms of memory consumption because it uses symmetric cryptographic keys instead of certificates. The proposed solution is superior in terms of energy consumption and computational power because the cryptographic keys are more fragile, which requires less computation for cryptographic functions. Also, the architecture of the communication process is simpler, as there is no need to implement certificate management, renewal, or revocation functions. Also, the DTLS protocol is designed for standard use in an open network to ensure security of communication over the Internet. The vehicle's local network does not require access to the Internet, so the proposed solution is superior in limiting communication. This allows it to be adapted to weaker devices in terms of computing power, which also have less memory. However, it should be noted that this solution is not suitable for communication over the Internet or for systems that have at least one subsystem or device outside the local network. Also, the proposed solution is not suitable for a frequently changing network architecture, where new devices may arrive at unexpected times, since all network devices must have a fixed cryptographic key stored in memory, which is used for the exchange of session keys. As this key is non-public, new devices cannot be added to the system without knowing this key, so new devices can only be added during system configuration. In addition, for each message sent over the system network, a MAC value is calculated to authenticate the sender of the message. The CMAC function for this function shall be an efficient and computationally lightweight function for the calculation of MAC values.

For the management of each subsystem of the main system, the ESP32 microcontroller was chosen. This controller was chosen because of its wide range of applications, the variety of ESP32 controllers, and the technical specifications:

1. Cryptographic hardware acceleration for AES, digest functions, and other cryptographic functions.
2. Bluetooth Classic, Low Energy Controller, and Baseband—supports Bluetooth v4.2 and the newer, low-energy BLE version. Also, more than one client connection or server is supported.
3. Dual-core 32-bit LX6 microprocessor supporting 240 MHz.

4. 520 KB SRAM, 448 KB ROM, and 16 KB RTC SRAM.
5. Thirty-four programmable GPIOs.
6. SPI, I2C, I2S, and UART serial communication protocols are supported.

In programming environments, the Microsoft Visual Studio Code text editor was used due to its status as an open-source application that offers an extensive library of plug-ins. Espressif IDF is the primary plugin utilized in the prototype's implementation. The Espressif IDF plugin furnishes and deploys every essential library and tool into the Visual Studio Code text editor, including the Buildroot tool for programming, modifying, deactivating, or activating device technical specifications, serial port software configurations, and configurations of supported devices. Buildroot is used to enable and configure the numerous Bluetooth interfaces, cryptographic libraries, and other modules that the project employs on the controller hardware and in the Espressif IDF environment. By properly configuring this utility, it exclusively compiles and utilizes modules that are explicitly enabled. As a result, resource-constrained systems are not burdened with inactive modules and libraries that consume memory and processing power. Additionally, the majority of required modules are disabled in a typical device configuration, rendering the Buildroot utility indispensable for the implementation of the prototype.

The following modules were enabled, and the following settings were configured using Buildroot:

- Flash memory was configured at 2 MB.
- Only the Bluetooth BLE version was enabled, which uses less power and supports multiple client interfaces.
- The number of Bluetooth connections was set to 3. Each available communication increases the amount of memory used. Since the prototype decided to use a maximum of three communications in the controller, it was decided to use fewer resources.
- The Bluetooth BlueDroid implementation was enabled, which is based on the Android Bluetooth implementation, enabling easy network scanning for devices, easier network data transfer, and other features. The most important BlueDroid feature for the prototype implementation is the support for GATT clients and GATT servers (also called Bluetooth Smart). Each GATT client or server profile allows for new communication.
- The processor speed of the ESP32 controller was set to 240 MHz, which increases the computing power.
- Enabled MBEDTLS library modules: CRYPTO, CMAC, AES, SHA, SHA512, MD5, HKDF.

4.2. Experimental Scenario for the Testing of the Proposed Solution

In order to perform the experiments properly, four devices need to be switched on: one master controller, which is a Bluetooth client, and three network devices, which are Bluetooth servers. When the client device is switched on, it waits for the other three server devices to connect, and work continues only when all devices have successfully connected. When the connection to all devices is successful, the client device performs cryptographic session key generation and exchange:

1. A new cryptographic key is computed from the master key using the HKDF key derivation function.
2. The cryptographic key is encrypted with the AES-CBC algorithm.
3. For an encrypted cryptographic key, the message authentication value (MAC) is calculated using the HMAC function.
4. A new message is created, starting with a message authentication value (MAC), followed by a message code, the code of the device to which the message is sent, and an encrypted cryptographic key.
5. The message is sent to the network device for which it is intended.
6. We wait for a response from the network device, after which the authentication value in the message is checked. If this value is successfully verified, the application will consider the cryptographic key to have been successfully sent.

7. Steps 1 to 5 are repeated until the cryptographic keys have been sent to all networked devices. The programme shall not wait for the message marked with action number 6 to arrive but shall immediately start work on re-generating the cryptographic key for the next network device. Action number 6 works asynchronously.
8. Once all the cryptographic keys have been allocated, a data request message is generated and encrypted with the AES-CBC algorithm.
9. The authentication value (MAC) is calculated for this message.
10. A new message is created, starting with the authentication value (MAC) of the message, followed by the message code, the code of the device to which the message is being sent, and the encrypted text of the data request.
11. The message is sent to the network device for which it is intended.
12. We wait for a response from the network device, after which the authentication value in the message is checked. If this value is successfully verified, the application counts the successful receipt of data from the server.
13. Steps 8 to 11 are repeated until the data request messages have been sent to all connected devices on the network. The application shall not wait for the message marked with action number 12 to arrive but shall immediately start again to create a data request for another network device. Action number 12 works asynchronously.
14. After one minute, repeat the steps from step 1.

All Bluetooth network server devices work in one application. The application itself does not initiate any actions and only responds to messages received from the client device:

1. When a message is received, the message code is checked, indicating the type of message received.
2. The authentication value (MAC) of the message is checked, if this value is not successfully authenticated, the message is deleted. If the message is authenticated, the message code is used.
3. If the message is a cryptographic key exchange message:
    - The cryptographic key is decrypted using the AES-CBC algorithm and the system master cryptographic key.
    - The cryptographic key is stored.
    - Text with a success value is generated.
4. If the message is a data request:
    - The data request is decrypted using the AES-CBC algorithm and the session cryptographic key.
    - Text is generated with a random number and a unit of measure.
5. The text is encrypted with the AES-CBC algorithm using the session key.
6. The authentication value (MAC) is calculated for the encrypted text.
7. A new message is created with the ciphertext, which starts with the authentication value of the message, followed by the reply code of the message, the code of the device sending the reply, and the ciphertext.

4.3. Realized System Data

As shown in Figure 6 in the first column, the "HKDF" tag is the printed computed cryptographic session key, which is further encrypted. The result of the encryption is printed with the "AES_ENC" tag. The encrypted cryptographic session key has to be sent, resulting in a packet that is printed with the "SIUNCIA" flag. The first two lines consist of the enumerated authentication value; in the third line, the first two digits are the message type code and the code of the device to which the message belongs. After these code values, the encrypted cryptographic session key is provided in the data stream. The device code "00" is used to indicate in red the data used to calculate and send the cryptographic session key. See Figure 6; the same steps for device code "01" are shown in blue.

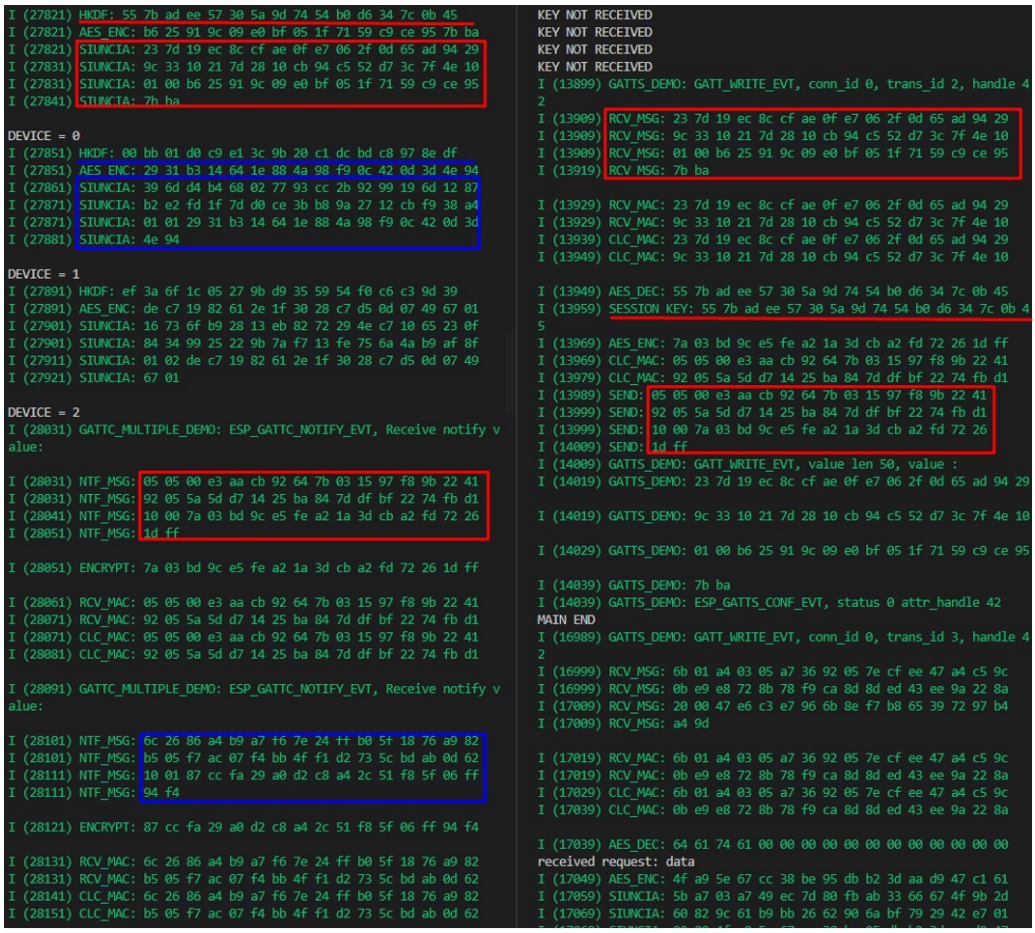

Figure 6. Key generation and distribution for the master controller to the first controller of the sensor subsystem.

In the second column, Figure 6, the message received by the first device with the code "00" containing the encrypted cryptographic session key is marked "RCV_MSG" and highlighted in red. A comparison with the data sent in the first column shows that the message matches. Next, in the second column, the authentication value is compared. In the first step, the authentication message is extracted from the received message and printed in the terminal with the tag "RCV_MAC". After the programme has calculated the comparable authentication value, which is calculated using the encrypted cryptographic key from the received message, this value is printed in the terminal with the flag "CLC_MAC". If these values match, the application decrypts the cryptographic session key and prints the result with the flags "AES_DEC" and "SESSION_KEY", when the cryptographic session key is successfully stored.

Next, a success message is generated and encrypted, and the result is printed with the tag "AES_ENC". For this encrypted message, the authentication value is computed and printed with the tag "CLC_MAC". When the programme has created a data message, it will be printed with the "SEND" flag. The first two lines of the message shall consist of the authentication value, then the message code and the code "00" of the device sending the message. This message is received and printed (see Figure 6) in the first column using the tag "NTF_MSG". Next, the application performs the same steps to check the authenticity of the message. In Figure 7 in the second column, the same steps are performed as in Figure 6 in the second column, but for a different device with the code "01". The results of this device are marked in blue.

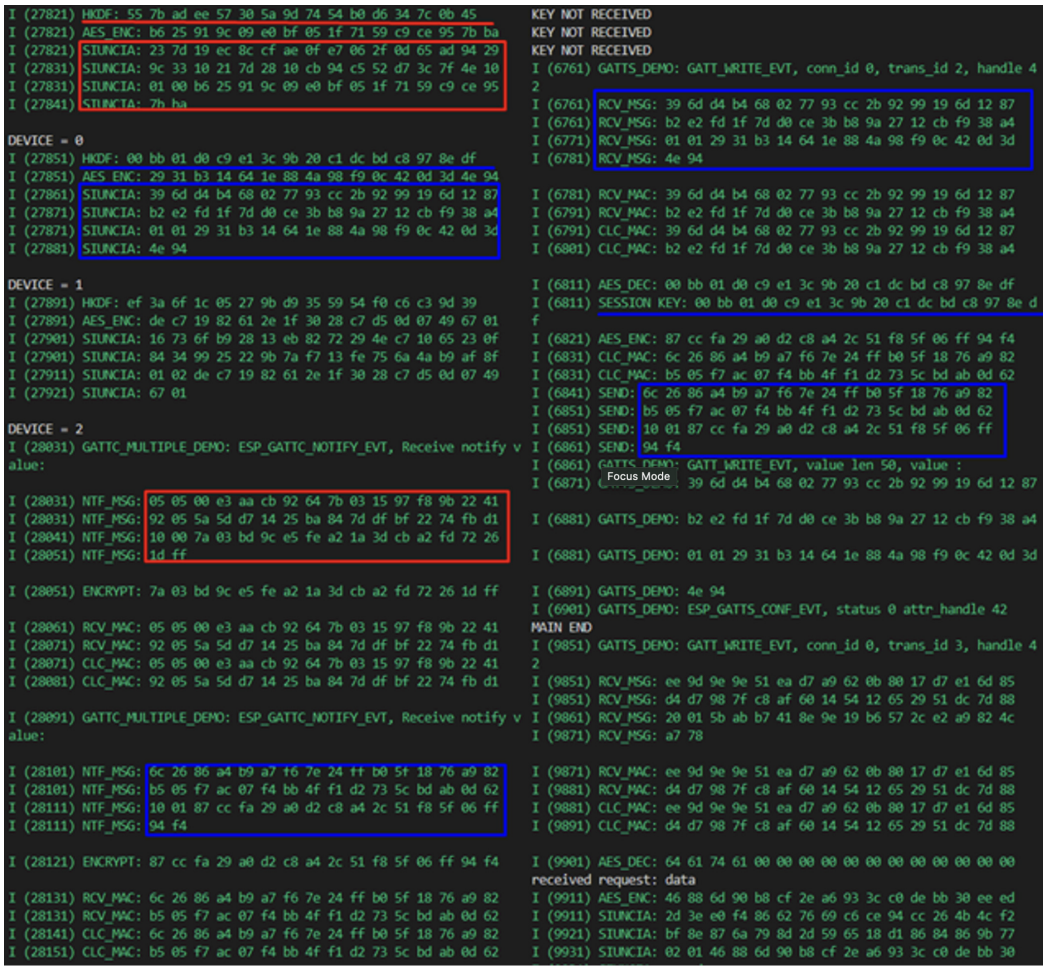

Figure 7. Key generation and distribution for the master controller to the second controller of the sensor subsystem.

The data query is shown in Figure 8. The first column shows the sending of data requests to all devices on the network. The red colour indicates a message to a device with the code "00". The row marked "AES_ENC" prints the result of the encryption of the text "date", followed by "SIUNCIA", which prints the whole message with the authentication value counted, the message code, the code of the device to which the message is addressed, and the encrypted text. In the second column of the figure, the tag "RCV_MSG" prints the message received with the device code "00".

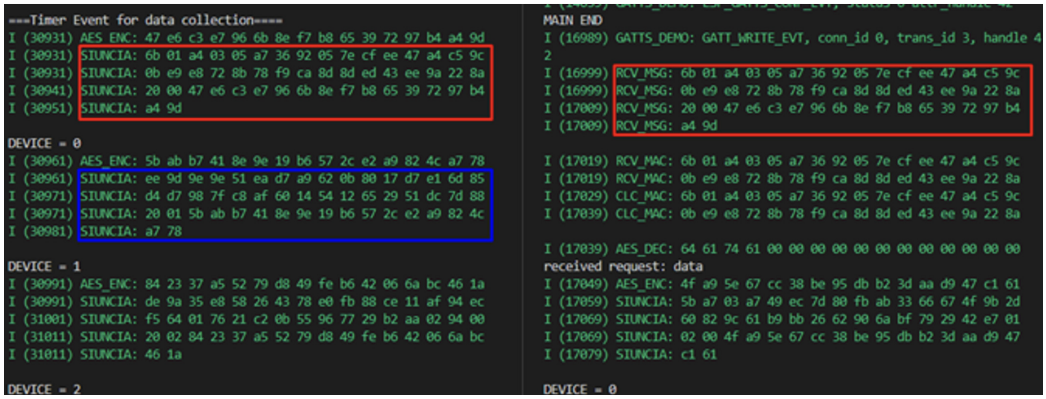

Figure 8. Master controller data requests to the first controller of the sensor subsystem.

The authentication value "RCV_MAC" contained in this message is compared with the newly calculated ciphertext authentication value "CLC_MAC" by the application. If these values match, the ciphertext of the message is decrypted, and the result is printed with the tag "AES_DEC" (marked in yellow) and the result is displayed in the terminal with the "ASCII" code after the tag "received request". The sample sensor data are then encrypted, the authentication value is computed, and a message printed with the tag "SIUNCIA" is generated. In the second one, Figure 9 in column 22, the same steps are carried out with device code "01".

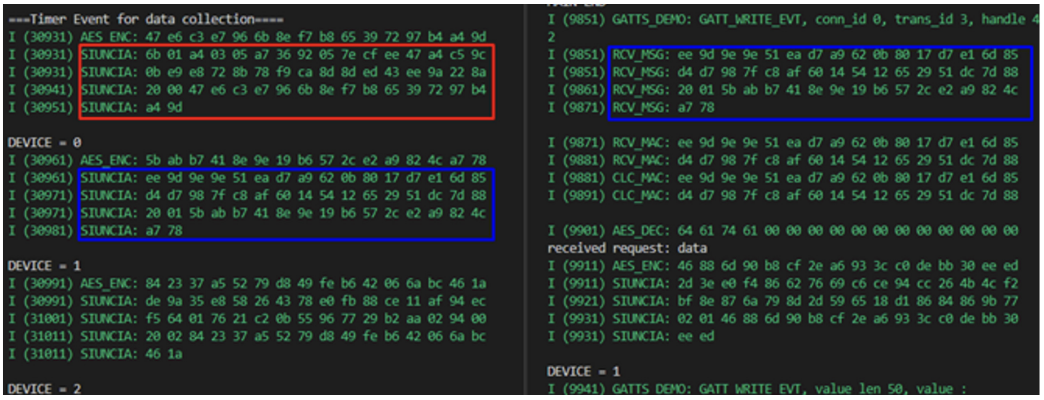

Figure 9. Data requests from the master controller to the second controller of the sensor subsystem.

The reception of data sent by these devices is shown in Figure 10 in the first column. The data received from the device with the code "00" are marked in red (Figure 11).

The data received from device code "01" are shown in blue. The messages received are printed with the flag "NTF_MSG". These messages are authenticated with the authentication values "RCV_MAC" and "CLC_MAC". Next, the data are decrypted and printed with the tags "AES_DEC" and "received message".

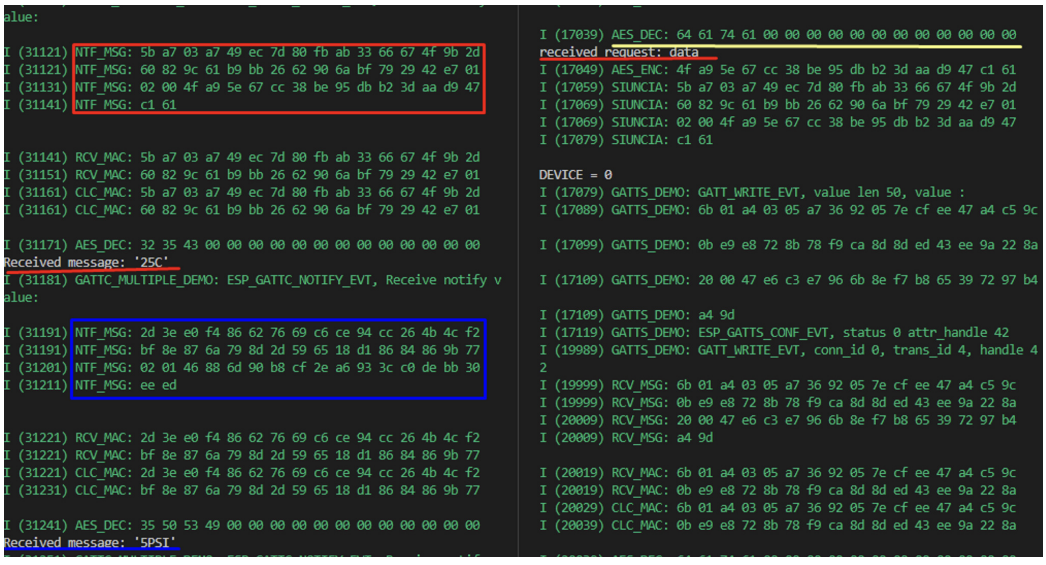

Figure 10. Master controller data request response message from the first subsystem controller.

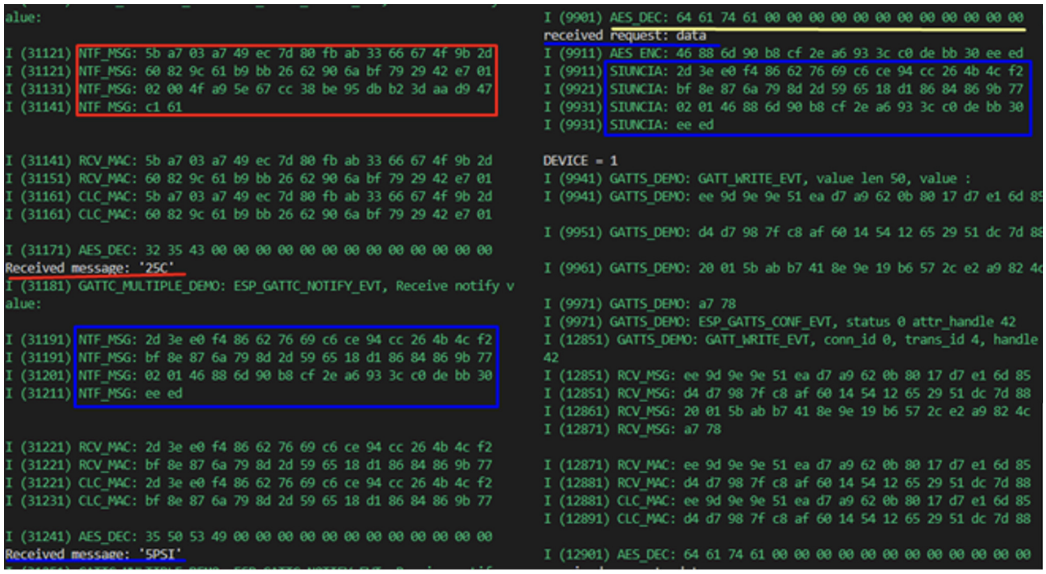

Figure 11. Master controller data request response message from the second subsystem controller.

After a set period of time, a new cryptographic session key calculation and distribution are performed, as shown in Figures 12 and 13. All data on the terminal also have a data printout time since the device was switched on. These data are measured in microseconds and printed at the beginning of the line in brackets.

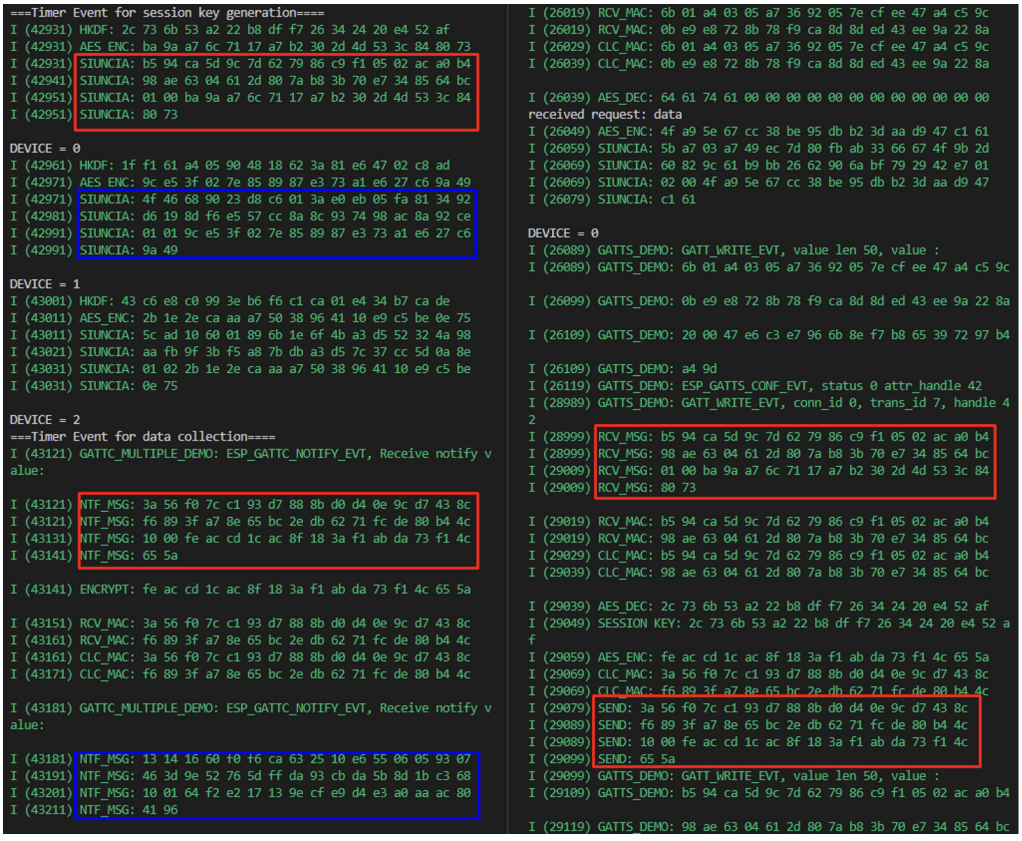

Figure 12. Master controller periodic cryptographic session key generation time event—first device data.

Due to the detailed documentation of the ESP microcontroller system and its programming environment, the security programmes for the prototype implementation of the project are developed using the internal libraries of the Espressif tool since most of the cryptographic algorithms are implemented in the MBEDTLS library.

It was observed that the most challenging part of the implementation of the secure method is the secure key exchange between devices and the secure message exchange between devices. In order to implement it properly, a new packet is created for each message with all the security information and encrypted data. This packet is successfully sent to another device on the network after being encapsulated in a Bluetooth communication packet.

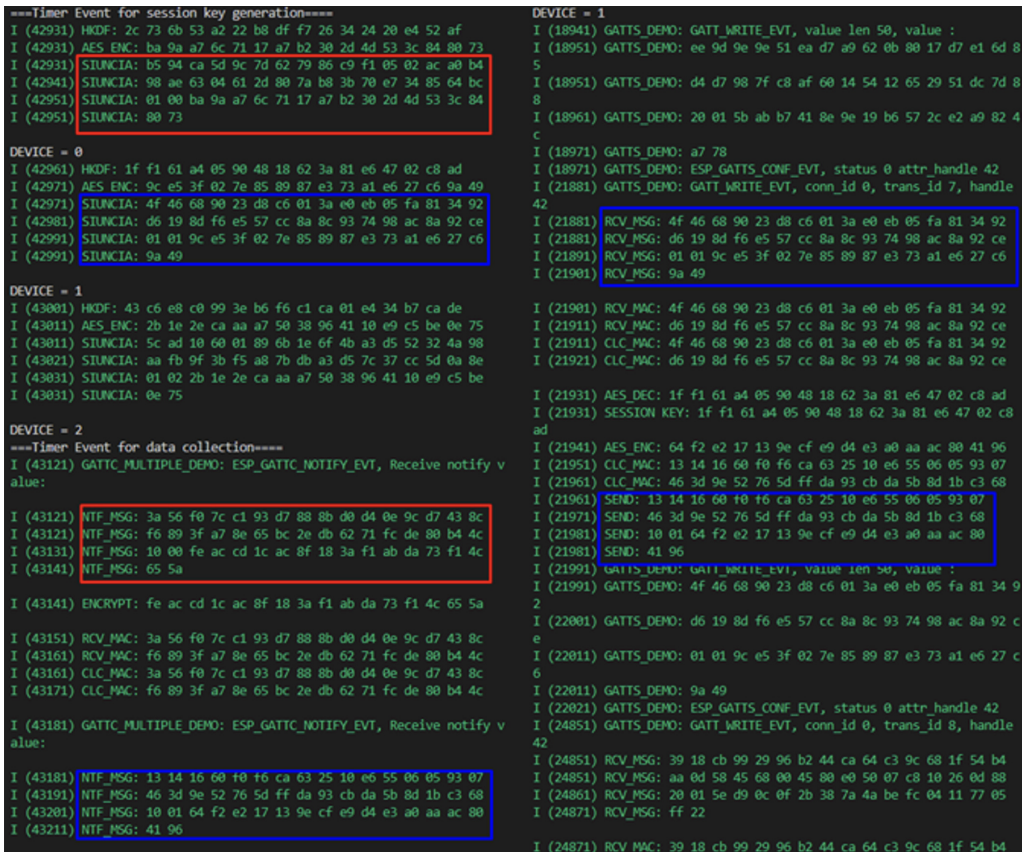

Figure 13. Master controller periodic cryptographic session key generation time event—second device data.

As the hardware chosen is more than strong enough in terms of computing power, all cryptographic functions are executed at a very high speed, and no delays or slowdowns were observed.

Also, it has been observed that the HMAC value sent in each message is always received undistorted in the communication tunnel and is always correct after verification by the device that received the message from the other device in the network. For this reason, all messages are processed successfully.

4.4. Methodology for Testing the Proposed Solution and Its Security

This subsection describes precisely the steps and methodology for the research of power consumption, time rate, memory resource consumption, and cryptographic security of the proposed solution.

Energy efficiency is an important parameter, as the application of the proposed solution is for resource-constrained embedded systems that can be used in vehicles. Therefore, excessive costs may cause system or data interference. In order to properly measure the energy cost of the proposed solution, the following steps are taken:

1. The device in which the proposed solution is installed is connected to the test electrical circuit. This circuit consists of a constant 5-volt power source whose earth is connected to the earth ground of the device.

2. The 5 volt connector of the constant power supply is connected to the 5 volt connector of the device using a multimeter, which becomes part of the electrical circuit. The device under test is connected only to a power source and an ammeter (a multimeter configured to measure the electric current in amperes).
3. A multimeter is configured to measure the electric current in an electrical circuit with milli-ampere accuracy.
4. The energy consumption of the device before running the test programme is measured.
5. The parameters marked under "Energy efficiency" are tested.
6. The measurements are taken with a video camera so that you can see the exact steps and results.
7. The multimeter is connected to a computer to collect all the intermediate data from which the energy consumption graphs are drawn.

A total of five measurements were carried out and the results of all steps are shown in Table 1.

Table 1. Energy costs of the proposed solution.

Step No.	Measurement 1	Measurement 2	Measurement 3	Measurement 4	Measurement 5	Average
1	50 mA	51 mA	50 mA	53 mA	52 mA	51 mA
2	77 mA	78 mA	77 mA	76 mA	77 mA	77 mA
3	79 mA	79 mA	78 mA	80 mA	79 mA	79 mA
4	68 mA	69 mA	68 mA	70 mA	70 mA	69 mA
5	74 mA	75 mA	74 mA	77 mA	75 mA	75 mA
6	57 mA	58 mA	57 mA	58 mA	58 mA	58 mA
7	56 mA	56 mA	57 mA	57 mA	58 mA	57 mA

As a first step, the energy consumption of the device was measured before running the prototype application of the proposed solution. It is observed that the device consumes approximately 50 mA of current. The second step measures key generation and sending per network device. It was observed that the device used mainly 77 mA of current, and it took 266 ms. In the third step, the network topology is configured with three devices to which the client device of the prototype solution connects and performs cryptographic key exchange. It was observed that the device mainly used 79 mA of current, and it took 600 ms. In the fourth step, the network topology is configured with only one device, to which the prototype client device connects. After measuring the energy consumption when sending and receiving a message, it was observed that the device's energy consumption did not exceed a current of 68 mA and lasted 33 ms of time. In the fifth step, the network topology is configured with three network devices to which the client device connects. After measuring the energy consumption of sending and receiving messages between these devices, it was observed that the devices mainly consume 74 mA of current and that it takes 300 ms. The sixth step is to measure the energy consumption of the cryptographic key acquisition device of the server to which the client connects. It was observed that the device used a maximum current of 57 mA for a duration of 600 ms. In step 7, the energy consumption of the server device for receiving and sending messages was measured. It was observed that the device used mainly 56 mA for a duration of 300 ms. The power consumption is for connecting to other devices in the Bluetooth network; this is done before the start of the proposed solution method. When calculating how much energy the device consumed in power watts during the test, these data are included in order to compare with the energy consumption of other methods on the market. It is important to note that during the first power consumption test, the cryptographic session keys are changed every 60 s.

Cryptographic security is an important parameter as the application of the proposed solution is designed to protect the communication technology used, which in the research of

the proposed solution is Bluetooth Low Energy. In order to properly test the cryptographic security of the proposed solution, the following steps are performed:

1. We investigated whether a brute-force attack and other attacks can be used to enumerate the cryptographic session key used.
2. We analyzed possible attacks on the cryptographic technologies and techniques used:
    - Potential attacks and weaknesses in the AES-CBC 128 encryption algorithm were investigated.
    - Potential attacks and weaknesses of the HMAC authentication value calculation algorithm were investigated.
    - Potential attacks and weaknesses in the HKDF cryptographic key derivation function were investigated.

The AES-CBC-128 cipher belongs to the AES cipher family, and all AES 128 ciphers are resistant to brute force attacks because it takes too long to calculate the correct cryptographic key used. Also, in the proposed solution, each message contains a Message Authentication Characteristic (MAC) value, which provides additional security against out-of-network devices and the insertion of a hacker or data message in the network. Since each message is authenticated before the data can be decrypted, it is necessary to bypass the system authentication first. It is worth noting that the messages sent within the system are always encrypted, eliminating any possibility of attacks that need to have the textogram (the data before encryption) in order to verify the result or to look for duplicate blocks in the ciphergram. Also, each encryption uses a variable salt value that changes the ciphertext even when sending two messages with the same textogram that are encrypted with the same cryptographic session key. The use of a floating salt value, which is a counter, also helps to synchronize the operation of network devices. This lowers the likelihood of success when inserting a system-sent message containing previously read data. This feature increases the resistance against a replay attack since the data message on the network will not match the counter value between communicating devices.

The security of the system is further enhanced by changing cryptographic session keys. This cryptographic key change can be configured in different ways, but during the study, key changes are performed every minute and data messages are sent every five seconds. Given the cryptographic security of the system discussed above, it can be noted that session keys can be changed at longer time intervals. Since the energy consumption study has shown that energy consumption does not increase with the generation and exchange of new cryptographic session keys, it is concluded that key changes can be performed as frequently as tested in the study. This feature helps to further reduce the probability of key guessing. If a data message packet is captured by the system and it succeeds in calculating a cryptographic key, depending on how the cryptographic key exchange is configured in the system, the calculated key may not be used anymore, as the cryptographic keys in the system will have been changed already.

The Hashed Message Authentication Code (HMAC) hash function used to calculate the Message Authentication Value (MAC) is considered secure and reliable because it is resistant to dictionary attacks and because the calculated authentication value is extremely difficult to spoof without knowing the secret key. However, it is important that different secret keys are used to ensure security. The proposed solution increases the security that HMAC offers by having each communication tunnel communicate using unique cryptographic session keys. The suggested solution also uses a type of HMAC-SHA256. This type of encryption is safer because it needs a key that is long enough; a brute force attack cannot work with this feature. The HMAC-SHA256 digest is not reversible, so there is no point in analyzing the added authentication value in the messages sent by the system. In addition, the HMAC digest function is resistant to a length extension attack, which allows additional data to be added to the message for which the authentication value is being calculated and extends the digest function.

HKDF can work with and without the salt value, but the use of the salt value strengthens the result of the HKDF function. Therefore, in the proposed solution, the salt value,

which is a counter in the implementation of the solution, is added to the execution of the function. Also, a timestamp is added to the salt value. This timestamp is calculated from the start of the device, so a difference of one millisecond will change the cryptographic session key calculated via the HKDF function in the calculation. It is worth noting that HKDF has two parameters that change the computed key: the salt and info parameters. The counter is fed into the info parameter, and the timestamp into the salt parameter. Due to these implemented parameters, the key derivation function HKDF always generates a unique cryptographic session key.

5. Validation of the Proposed Framework

5.1. Results of the Experiments on the Time Speed of the Proposed Solution

In terms of time speed, the investigation is carried out by saving the data printed on the terminal indicating when and what actions were performed. It has also been observed that printing more than one line results in a printing time of 10 milliseconds. This time is deducted from the final result.

The "HKDF" flag indicates that the cryptographic key generation is complete, and this line also contains the generated cryptographic key. Both lines mark the same time since the start of the device: 5821 ms. Also, the same timestamp is printed in the line marked "AES_ENC", which marks the end of the encryption of the cryptographic key and prints the ciphergram of the cryptographic key. From these results, it is concluded that the generation and encryption of the cryptographic key take less than 10 ms.

It took approximately 220 ms for the client device to receive the cryptographic session key exchange response from the server device. However, the second transmission took 200 ms to generate and send the cryptographic key. The same results can be observed in the generation and sending of the data collection. The timestamp printed on the left shows that the encryption of the message does not take more than 10 ms. It took 200 ms per device to send a message and receive a reply.

Sending data requests to three different devices and printing all the results at the terminal (printing at the terminal takes 10 ms for two lines of text) did not take more than 380 ms in generating, encrypting, sending, and receiving the response from all three devices. The same amount of time was taken to compute new session keys, encrypt them, send them, and receive reply messages from all three devices. Therefore, it is concluded that the most time-consuming part of communication technology is the sending of data.

5.2. Results of the DTLS Time-Speed Experiments

The time-speed test is performed by saving the data printed on the terminal indicating when and what actions were performed. Also, as in the study of the proposed solution, it was observed that printing more than one line results in a printing time of 10 milliseconds. As shown in Figure 14 on the left, the circled number is the time taken to print a line of terminal text from the start of the device in mili-seconds. The first red frame indicates the establishment of a new session, which started at approximately 11.6 s of device operation, and the new session was successfully established at 12.5 s of device operation. With the DTLS protocol, it took 910 ms to establish a new session between the two devices.

It can be concluded that if more devices are connected to the network topology, it will take the same amount of time for each device to establish a session. The second red frame indicates the sending and receiving of one request message. The request is sent at 14.6 s of device operation, and the reply is received at 14.8 s of device operation. It took 210 ms for the device to receive the response. It was observed that establishing a communication session takes the longest time, so establishing a new session each time a message is sent would cause excessive system latency, especially when communicating with more than two network devices. Therefore, for all studies, the DTLS applications are configured to keep the session alive and send other messages using the same session. It has been observed that it takes, on average, 200 ms to send a message and receive a reply without creating a new communication session.

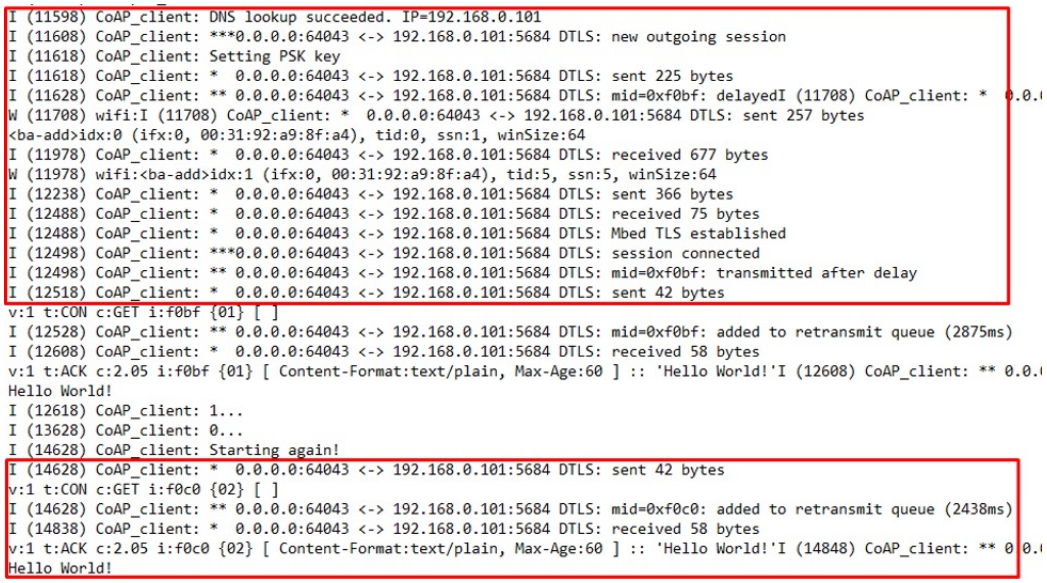

Figure 14. Time taken to set up a DTLS protocol communication session and receive a message.

5.3. Results of the DTLS Protocol Memory Consumption Study

The memory resource consumption of the DTLS client device application is shown in Figure 15. The total memory consumption of the application is labelled "Total image size" and indicates a size of 856961 bytes (856.96 Kb). The size of the programme includes the printing of the data to be analyzed in the terminal window.

The memory resource consumption of the DTLS server device application is shown in Figure 16.

The total memory consumption of the application is labelled "Total image size" and indicates a size of 848,765 bytes (848.77 Kb). The size of the application includes the printing of the data to be analyzed in the terminal window. It is noted that the DTLS server application takes 8196 bytes less than the DTLS client application.

```
Total sizes:
Used static DRAM:   43572 bytes ( 137164 remain, 24.1% used)
      .data size:   14684 bytes
      .bss  size:   28888 bytes
Used static IRAM:   83402 bytes (  47670 remain, 63.6% used)
      .text size:   82375 bytes
   .vectors size:    1027 bytes
Used Flash size :  758875 bytes
      .text     :  625259 bytes
      .rodata   :  133360 bytes
Total image size:  856961 bytes (.bin may be padded larger)
```

Figure 15. DTLS protocol client device application memory resource consumption.

```
Total sizes:
Used static DRAM:   43340 bytes ( 137396 remain, 24.0% used)
      .data size:   14692 bytes
      .bss  size:   28648 bytes
Used static IRAM:   83402 bytes (  47670 remain, 63.6% used)
      .text size:   82375 bytes
   .vectors size:    1027 bytes
Used Flash size :  750671 bytes
      .text     :  618447 bytes
      .rodata   :  131968 bytes
Total image size:  848765 bytes (.bin may be padded larger)
```

Figure 16. Memory resource consumption of the DTLS protocol server device application.

5.4. Results of the TLS Memory Resource Consumption Study

The memory resource consumption of the TLS client device application is shown in Figure 17.

```
Total sizes:
Used static DRAM:   30616 bytes ( 150120 remain, 16.9% used)
      .data size:   14400 bytes
      .bss  size:   16216 bytes
Used static IRAM:   83182 bytes (  47890 remain, 63.5% used)
      .text size:   82155 bytes
   .vectors size:    1027 bytes
Used Flash size :  720235 bytes
      .text     :  544939 bytes
      .rodata   :  175040 bytes
Total image size:  817817 bytes (.bin may be padded larger)
```

Figure 17. TLS protocol client device application memory resource consumption.

The total memory consumption of the application is labelled "Total image size" and indicates a size of 817,817 bytes (817.82 Kb). The size of the program includes the printing of the data to be analyzed in the terminal window. The memory resource consumption of the TLS server device application is shown in Figure 18.

The total memory consumption of the application is labelled "Total image size" and indicates a size of 765,757 bytes (765.76 Kb). The size of the application includes the printing of the data to be analyzed in the terminal window.

It is noted that the TLS server application takes 52,060 bytes less than the TLS client application.

```
Total sizes:
Used static DRAM:   30232 bytes ( 150504 remain, 16.7% used)
      .data size:   14400 bytes
      .bss  size:   15832 bytes
Used static IRAM:   83166 bytes (  47906 remain, 63.5% used)
      .text size:   82139 bytes
   .vectors size:    1027 bytes
Used Flash size :  668191 bytes
      .text     :  556239 bytes
      .rodata   :  111696 bytes
Total image size:  765757 bytes (.bin may be padded larger)
```

Figure 18. Memory resource consumption of the TLS protocol server device application.

5.5. Comparison of Energy Costs

Following energy cost investigations of the proposed solution using the "DTLS" and "TLS" approaches, comparisons were made (see Table 2), the results of which are shown in the graph in Figure 19. The client device's energy consumption graph shows that the proposed solution uses less energy at first when connecting to the network than both DTLS and TLS. However, during normal operation, the proposed solution uses less energy than the TLS method, which is set up to not protect the communication session.

Table 2. Comparison of the energy costs.

Methods	Energy Cost, mA
DTLS	61.4912
TLS	69.491
Proposed solution	65.354

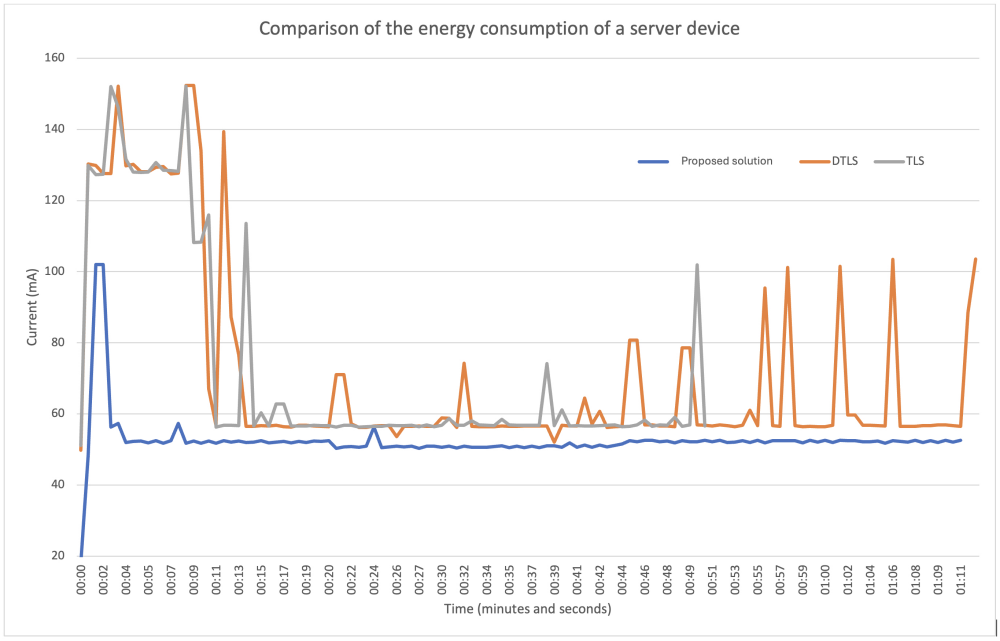

Figure 19. Server device energy comparison chart.

A comparison of the calculated average client device energy consumption of the methods shows that the DTLS client device consumes an average of 3.863 mA of current, or 0.0195 watts less than the proposed solution client device. However, the graph shows that the client device of the proposed solution always consumes approximately the same amount of power, while the DTLS device experiences a spike when the WiFi network interface needs to be upgraded. If the WiFi network does not return a response, the cost of the device will increase.

It is also worth mentioning that during the testing of the proposed solution, the client device communicated with a total of four devices, while DTLS and TLS only communicated with two devices each. The energy consumption graph of the server device shows that the initial energy consumption of the server device when connecting to the network and the lifetime energy consumption of the server device in the proposed solution are lower than those of the "DTLS" and "TLS" methods. Also, it can be seen that the energy consumption is more stable with little variation, especially when compared to the energy consumption of the "DTLS" server device. A comparison of the calculated average energy consumption of the server devices of the methods shows that the device of the proposed solution consumes the least energy. This is because the server device of the proposed solution does not perform any computation itself, but only as a result of receiving a data request from the client device.

5.6. Memory Resource Costs

The overall results of the memory resource costs for the proposed solution in comparison with DTLS and TLS methods are provided in Table 3.

Table 3. Comparison of memory consumption.

Methods	Client Application	Server Application	Costs in % in Comparison with Proposed Solution
DTLS	856.961 B	848.765 B	137.76%
TLS	817.817 B	765.757 B	127.9%
Proposed solution	622.053 B	616.113 B	100%

The memory resource consumption of the method application on the client device is higher than that of the server device, but only the memory consumption of the TLS

method shows a significant difference. Also, the server and client device applications of the proposed solution consume the least memory resources compared to the "DTLS" and "TLS" methods. It is concluded that the reduction in the size of the cryptographic keys in the proposed solution, compared to certificates, has also had a strong impact on memory resource consumption. This is because the system does not need to store certificates, certificate authority, or other parameters and functions needed to perform computations with these certificates. This not only reduces the consumption of memory resources but also increases the efficiency of the method and reduces energy consumption. However, it is worth noting that the DTLS client device consumes less energy than the client device of the proposed solution. It was observed that the DTLS approach had a lower energy and time-rate cost than the TLS approach, but the memory resource cost was the highest compared to the memory resource cost of the proposed solution and the TLS approach. The graph comparing the average memory consumption shows that the DTLS protocol uses approximately 37.76% more memory resources than the proposed solution, while the TLS method uses approximately 27.9% more. The "DTLS" method uses 7.71% more memory resources compared to the "TLS" method.

5.7. Comparison of Experimental Results

The difference between the proposed solution and the TLS and DTLS approaches is that the proposed solution does not use asymmetric encryption to form the communication tunnel. This is done using a single-session cryptographic key, which is changed periodically. However, all three methods investigated encrypt the data with a single symmetric cryptographic key after establishing secure communication. However, the proposed solution is configured to use a 128-bit cryptographic key, while TLS and DTLS use a 2048-bit key. It can be stated that the "TLS" family of methods provides more secure data encryption, but at the additional cost of computing power resources. Although the data messages sent by DTLS and the proposed solution took the same amount of time in the investigations, using equipment weaker in terms of computing power, a longer cryptographic key may slow down the messages. Also, the proposed solution does not provide a secure communication tunnel like the DTLS and TLS methods but covers this immediately with the devices enabled in the system. In this case, the devices on the network are pre-configured to communicate with only a few nearby devices. The system is not dynamic as in "DTLS" or "TLS" methods. This factor increases the security of the system by reducing the possibility of inserting an external device into the network. The establishment of a secure communication session is also not important in the proposed solution because the first message sent from the client device is encrypted with a master key, which is not distributed over the network but is installed on the device during the system configuration. Compared to TLS or DTLS, this is done with certificates and their certificate authority, which verifies whether the certificates are authenticated. In addition, the proposed solution does not send parameters over the network, which are used to generate the cryptographic key needed to encrypt the traffic. Instead, there is a single device in the network that is responsible for the generation and distribution of cryptographic keys. It is worth noting that in the "DTLS" and "TLS" methods, authentication is carried out at the time of the establishment of the communication session based on certificates. The proposed solution provides a message authentication value (MAC), which is checked before decrypting or otherwise processing the received data.

6. Discussion

The work presented in this paper outlines significant improvements in securing communication within in-vehicle wireless sensor networks. It introduces a framework for the regular updating of cryptographic keys, which is vital for keeping communication between various subsystems secure. This limits the duration for which a compromised key can be used, thus minimizing potential security breaches.

Furthermore, the framework enhances the confidentiality of cryptographic key generation by keeping the generation parameters confidential. It also ensures that any new session keys are transmitted across the network in encrypted form, providing an additional layer of security against unauthorized access. An authentication technique is also proposed for distinct, independent network subsystems. This technique uses the sender's message authentication code (MAC) to verify communication, thereby ensuring that messages are authenticated within the communication tunnel between parties. This not only secures the communication but also maintains the integrity of the data being transmitted, offering a comprehensive solution to in-vehicle network security challenges.

The proposed solution is specifically designed for static and short-range networks that do not interface with the Internet. While primarily developed for vehicles, it is noted that the solution can also be applied to other embedded systems or Internet of Things (IoT) technologies where resources are limited. However, it is important to mention that this solution is not suitable for systems that require Internet communication but may be applicable to isolated subsystems within such environments. The advantage of the method presented in the paper over the BLE security mode is its efficiency in terms of computation, energy consumption, and memory resources. The proposed solution uses symmetric cryptographic keys instead of certificates, which requires less computation for cryptographic functions and simplifies the architecture of the communication process.

The analysis reveals that network security problems often arise due to a lack of authentication and weaknesses in the cryptography used. As embedded systems frequently operate within resource-constrained environments, it is impossible to implement the strongest and most stringent security standards and functions available in the market. Therefore, there is a need for security methods that enhance the security of embedded systems. Nonetheless, it is worth noting that such methods invariably require system restrictions, such as a static network architecture and isolation of the system from external networks or devices.

Our framework's introduction of periodic cryptographic key exchanges significantly outperforms traditional static key systems in dynamic network environments, as evidenced by improved resilience against replay and man-in-the-middle attacks. Furthermore, the segmentation of vehicle networks into independently secured subsystems presents a novel approach that enhances compartmentalization and reduces the attack surface, a method not extensively explored in prior studies. The comparison reveals that while other research has laid the foundational aspects of communication security, our study introduces more robust and adaptive mechanisms that better address the complex threat landscape of modern vehicular networks. This detailed analysis underscores the superiority of our framework in providing comprehensive security solutions, thereby setting a new benchmark for future research in the field.

The exploration of how the proposed framework can be adapted or integrated with emerging technologies such as V2X communication and blockchain is pointed out as future work in this research area. These technologies can potentially enhance security features in in-vehicle communication, but they may also introduce new challenges that need to be addressed.

7. Conclusions

The proposed communication security framework for in-vehicle wireless sensor networks is efficient and uses fewer memory resources compared to the DTLS and TLS methods because it employs weaker cryptography. It is important to emphasize that the proposed solution is designed for a local network without any communication links to external networks. The absence of communication with the external Internet reduces the risk to the network, allowing for the use of less complex cryptography. This is why the proposed solution has lower energy consumption, requires fewer memory resources, and is more time-efficient compared to the DTLS and TLS approaches.

The reduced energy consumption results from the proposed solution performing fewer computational steps in encrypting and decrypting data, calculating message authentication values, and generating new cryptographic session keys. The higher time efficiency stems from the same reason: fewer computational steps and the use of shorter cryptographic keys. Compared to DTLS and TLS, the use of shorter cryptographic keys further accelerates the process. Additionally, memory resource consumption is lower due to shorter program code, cryptographic keys, and other security parameters.

It has been noted that the proposed framework is not suitable for dynamic networks. This is because each device in the network must be individually configured, specifying the devices with which it will communicate. Cryptographic parameters and the master key must also be set during device configuration. Without these parameters, it is not possible to add an additional device to the network.

Another important consideration is that the proposed solution is not suitable for networks with an interface to the Internet, since the cryptography used does not meet the security requirements. For the same reasons as for dynamic networks, it is not an efficient solution.

Author Contributions: Conceptualization, A.V. and M.T.; methodology, Š.G.; software, M.T.; validation, A.V., M.T., Š.G. and R.B.; formal analysis, R.B.; investigation, M.T.; resources, M.T., Š.G. and R.B.; data curation, A.V.; writing—original draft preparation, Š.G. and R.B.; writing—review and editing, A.V.; visualization, S.G.; supervision, A.V.; project administration, A.V.; funding acquisition, A.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this paper are available on request from the corresponding author. The data are not publicly available due to the project not being completed.

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

References

1. Murshed, A.; Anowar, S.S. Automatic Braking System and Smart Safety Features to Avoid and Reduce Road Accident. Ph.D. Thesis, BRAC University, Dhaka, Bangladesh, 2021.
2. Wang, Y.; Cui, Y.; Chen, F.; Ren, R. An "illumination moving with the vehicle" intelligent control system of road tunnel lighting. *Sustainability* **2020**, *12*, 7314. [CrossRef]
3. Gautam, A.; Verma, G.; Qamar, S.; Shekhar, S. Vehicle pollution monitoring, control and challan system using MQ2 sensor based on Internet of things. *Wirel. Pers. Commun.* **2021**, *116*, 1071–1085. [CrossRef]
4. Bharati, S.; Podder, P.; Mondal, M.; Robel, M.R.A. Threats and countermeasures of cyber security in direct and remote vehicle communication systems. *arXiv* **2020**, arXiv:2006.08723.
5. Choudhary, D.; Pahuja, R. Deep learning approach for encryption techniques in vehicular networks. *Wirel. Pers. Commun.* **2022**, *125*, 1–27. [CrossRef]
6. Olaniyi, O.O.; Okunleye, O.J.; Olabanji, S.O.; Asonze, C.U. IoT security in the era of ubiquitous computing: A multidisciplinary approach to addressing vulnerabilities and promoting resilience. *Asian J. Res. Comput. Sci.* **2023**, *16*, 354–371. [CrossRef]
7. Menon, V.G.; Jacob, S.; Joseph, S.; Sehdev, P.; Khosravi, M.R.; Al-Turjman, F. An IoT-enabled intelligent automobile system for smart cities. *Internet Things* **2022**, *18*, 100213. [CrossRef]
8. Olufowobi, H.; Bloom, G. Connected cars: Automotive cybersecurity and privacy for smart cities. In *Smart Cities Cybersecurity and Privacy*; Elsevier: Amsterdam, The Nerthlands, 2019; pp. 227–240.
9. Xiong, Z.; Cai, Z.; Han, Q.; Alrawais, A.; Li, W. ADGAN: Protect your location privacy in camera data of auto-driving vehicles. *IEEE Trans. Ind. Inform.* **2020**, *17*, 6200–6210. [CrossRef]
10. Ghosal, A.; Conti, M. Security issues and challenges in V2X: A survey. *Comput. Netw.* **2020**, *169*, 107093. [CrossRef]
11. El-Rewini, Z.; Sadatsharan, K.; Selvaraj, D.F.; Plathottam, S.J.; Ranganathan, P. Cybersecurity challenges in vehicular communications. *Veh. Commun.* **2020**, *23*, 100214. [CrossRef]
12. Rathore, R.S.; Hewage, C.; Kaiwartya, O.; Lloret, J. In-vehicle communication cyber security: Challenges and solutions. *Sensors* **2022**, *22*, 6679. [CrossRef]
13. Hsiao, S.J.; Sung, W.T. Employing Blockchain Technology to Strengthen Security of Wireless Sensor Networks. *IEEE Access* **2021**, *9*, 72326–72341. [CrossRef]

14. Xiao, X.; Li, Y.; He, X.; Cai, Y.; Xiao, Y.; Huang, B.; Jin, X. Optimal Topology Control of Monitoring Sensor Network Based on Physical Layer Security for Smart Photovoltaic Power System. *Front. Energy Res.* **2023**, *11*, 1124700. [CrossRef]
15. Chen, Z.; Li, X.; Yang, B.; Zhang, Q. A Self-Adaptive Wireless Sensor Network Coverage Method for Intrusion Tolerance Based on Trust Value. *J. Sens.* **2015**, *2015*, 430456. [CrossRef]
16. Li, J. A Symmetric Cryptography Algorithm in Wireless Sensor Network Security. *Int. J. Online Eng. (IJOE)* **2017**, *13*, 102. [CrossRef]
17. Delgado-Mohatar, O.; Sierra, J.M.; Brankovic, L.; Fúster-Sabater, A. An energy-efficient symmetric cryptography based authentication scheme for wireless sensor networks. In Proceedings of the Information Security Theory and Practices. Security and Privacy of Pervasive Systems and Smart Devices: 4th IFIP WG 11.2 International Workshop, WISTP 2010, Passau, Germany, 12–14 April 2010; Proceedings 4; Springer: Berlin/Heidelberg, Germany, 2010; pp. 332–339.
18. Fang, X.; Fang, K.; Li, G.; Jin, X.; Zheng, L. Research on the Characteristics and Detection Methods of DDoS Attacks on Wireless Sensor Networks for Vehicle Networking. *Eng. Adv.* **2022**, *2*, 175–181. [CrossRef]
19. Miptahudin, A.; Suryani, T.; Wirawan, W. Wireless Sensor Network Based Monitoring System: Implementation, Constraints, and Solution. *JOIV Int. J. Inform. Vis.* **2022**, *6*, 778–783. [CrossRef]
20. Tetteh, A.; Essah, R.; Badhon, A.J.; Asante, Y.A.; Patrick, A.B. A Statistical Study Into Network Security Issues of IT Companies in Accra. *Asian J. Res. Comput. Sci.* **2021**, *12*, 1–13. [CrossRef]
21. Qiu, B.; Xiao, H.; Chronopoulos, A.T.; Zhou, D.; Ouyang, S. Optimal Access Scheme for Security Provisioning of C-V2x Computation Offloading Network With Imperfect CSI. *IEEE Access* **2020**, *8*, 9680–9691. [CrossRef]
22. Jameel, F.; Javed, M.A.; Zeadally, S.; Jantti, R. Secure Transmission in Cellular V2X Communications Using Deep Q-Learning. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 17167–17176. [CrossRef]
23. Abidi, B.; Jilbab, A.; Haziti, M.E. Security in Wireless Sensor Networks. *Int. J. Inform. Commun. Technol. (Ij-Ict)* **2019**, *8*, 13–15. [CrossRef]
24. Teekaraman, Y.; Manoharan, H.; Kuppusamy, R.; Urooj, S.; Alrowais, F. Energy Efficient Multi-Hop Routing Protocol for Smart Vehicle Monitoring Using Intelligent Sensor Networks. *Int. J. Distrib. Sens. Netw.* **2021**, *17*, 15501477211039134. [CrossRef]