symmetry

MDPI

Article

Enhancing MQTT-SN Security with a Lightweight PUF-Based Authentication and Encrypted Channel Establishment Scheme

Xiang Gong, Ting Kou and Yan Li *

Academy of Cyberspace Security, Gansu University of Political Science and Law, Lanzhou 730070, China; gongxiang@gsupl.edu.cn (X.G.); kouting@stu.gsupl.edu.cn (T.K.)
* Correspondence: ly6381@gsupl.edu.cn; Tel.: +86-931-7601604

Abstract: The communication of Industrial Internet of Things (IIoT) devices faces important security and privacy challenges. With the rapid increase in the number of devices, it is difficult for traditional security mechanisms to balance performance and security. Although schemes based on encryption and authentication exist, there are still difficulties in achieving lightweight security. In this paper, an authentication and key exchange scheme combining hardware security features and modern encryption technology is proposed for the MQTT-SN protocol, which is not considered security. The scheme uses Physical Unclonable Functions (PUFs) to generate unpredictable responses, and combines random numbers, time stamps, and shared keys to achieve two-way authentication and secure communication between devices and broker, effectively preventing network threats such as replay and man-in-the-middle attacks. Through verification, the proposed scheme has proved effective in terms of security and robustness, has computational and communication cost advantages compared with recent schemes, and provides higher availability.

Keywords: IIoT; MQTT-SN protocol; PUF; lightweight safety

Citation: Gong, X.; Kou, T.; Li, Y. Enhancing MQTT-SN Security with a Lightweight PUF-Based Authentication and Encrypted Channel Establishment Scheme. *Symmetry* **2024**, *16*, 1282. https://doi.org/10.3390/sym16101282

Academic Editor: Lianbo Ma

Received: 23 August 2024
Revised: 19 September 2024
Accepted: 27 September 2024
Published: 29 September 2024

1. Introduction

The IIoT stands as a pivotal element within the broader Internet of Things (IoT) ecosystem, significantly enhancing the sophistication of contemporary industrial intelligence and propelling the evolution of smart manufacturing. The IIoT is distinguished by its attributes of real-time functionality, automation, embedded software, robust security measures, and seamless information interoperability. These features are actively propelling the fourth industrial revolution, ushering in transformative changes across a spectrum of sectors, including but not limited to energy, manufacturing, urban planning, healthcare, and transportation [1].

As the convergence of industrial automation and information technology accelerates, the scope of IIoT applications is continually expanding. With this expansion comes an escalating imperative for security, encompassing protocol security, equipment security, data protection, connection integrity, and management control. Among these, protocol security emerges as the paramount concern within IIoT network security, underpinning the entire network's defense architecture [2]. The swift adoption of contemporary IIoT devices has amplified the necessity for resilient security protocols. Implementing stringent security and privacy measures at the protocol level could substantially fortify the IIoT application ecosystem. Given that the majority of IIoT devices are resource-limited, traditional security protocols are often ill-suited to their needs. Hence, there is an urgent need to devise security protocols that are specifically calibrated to the constraints and requirements of this context [3].

Within the IIoT landscape, the Machine-to-Machine (M2M) environment plays an indispensable role, enabling seamless data interchange and automated control mechanisms between machines through advanced communication technologies. This facilitates an

unobstructed device-to-device dialogue within the IIoT framework, thereby enhancing operational efficiency and fostering innovation in industrial processes.

Therefore, the M2M environment requires a reliable and efficient communication path to ensure real-time, accurate, and secure data transmission. This is crucial not only for data integrity and device interoperability but also for operational efficiency, security, and cost-effectiveness. When building and managing an M2M environment, efficient, reliable, and secure communication protocols such as Message Queuing Telemetry Transport (MQTT) and Extensible Messaging and Presence Protocol (XMPP) are typically chosen to facilitate device communication [4]. MQTT, a lightweight open messaging protocol, is specifically designed for resource-constrained network clients, particularly in low-bandwidth environments [5]. It provides reliable communication paths for M2M environments, enhances system scalability, and reduces long-distance transmission delays and bandwidth usage. To achieve a more efficient and flexible communication mode in resource-constrained environments, the MQTT for Sensor Networks (MQTT-SN) protocol, which is even lighter than MQTT, has been proposed. The MQTT-SN protocol is an M2M communication protocol designed specifically for the constrained environments of sensor networks [6].

However, the MQTT-SN protocol does not account for security, and most networks utilizing it still rely on plaintext transmission. Additionally, while standard security protocols like Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) exist, they are not designed for the constrained environments of IIoT and are unsuitable for restricted networks due to their high overhead [7]. Therefore, many scholars have introduced security mechanisms based on the MQTT-SN protocol and are working to design new protocols to ensure security.

In recent years, PUF hardware has gained significant attention for its uniqueness, unidirectionality, unpredictability, and resistance to cloning. PUF is an encryption technology that offers hardware-level security, based on the principle of utilizing the chip's physical characteristics to generate a unique identity or key for security operations like authentication, encryption, and decryption [1]. SRAM PUF, a specific implementation of PUF based on SRAM memory, leverages the deep submicron variations that naturally occur during semiconductor production to impart each transistor with a slight random electrical characteristic, thereby generating a unique and unclonable encryption root key for the device. SRAM PUF is reliable, scalable, and easily implemented, and it can be adapted to various process nodes used in IoT devices [8]. It is the only known PUF type that can be implemented simply by loading software onto a chip. In constrained environments, such as resource-limited IoT devices, SRAM PUF offers a secure and efficient method for key generation and management. As the keys are randomly extracted from within the chip, no additional hardware or memory is needed, reducing both cost and complexity. Additionally, the SRAM PUF key is generated dynamically, only when needed, and is not permanently stored in the device, further enhancing security [9].

Given this background, the integration of PUF into the security protocols of IoT devices presents a promising avenue for research and development. This paper delves into this domain, offering a novel perspective on enhancing the security of IoT communications. The main contributions of this paper are as follows.

1. This paper presents a lightweight security protocol based on MQTT-SN, utilizing lightweight PUF, XOR, hash functions, and other operations.
2. The paper employs the ProVerif tool (It is hereafter abbreviated as ProVerif), a formal verification tool grounded in the Dolev-Yao attacker model, to rigorously validate the proposed scheme, thereby substantiating its robust security profile.
3. The proposed scheme is designed with a strong emphasis on ensuring anonymity and untraceability, while also striving for scalability and resilience against DOS attacks, culminating in a high degree of usability.
4. A comparative analysis reveals that, when juxtaposed with similar schemes, the proposed scheme delivers optimal security at a reasonable cost.

5. The scheme's comprehensive fail-stop property marks a significant advancement in proactive attack detection. In the event of an attack against the protocol within the network, the system is equipped to swiftly identify the intrusion and consequently dispatch an immediate alarm to the upper management system, enabling the implementation of subsequent defensive strategies.

The structure of this paper is as follows: Section 2 offers a comprehensive review of the most pertinent literature to date. Section 3 lays down the foundational preliminaries essential for understanding the project design. Section 4 presents a meticulous exposition of the proposed scheme. Section 5 delves into both formal and informal security analyses of the scheme. Section 6 juxtaposes the proposed scheme with related work, examining its security and cost implications. Concluding the paper, Section 7 provides a thorough summary of the findings and contributions.

2. Related Work

In the domain of IIoT M2M security protocols, a critical examination of recent proposals has revealed persistent challenges. While cryptographic techniques like Diffie–Hellman, elliptic curves, and bilinear pairings are utilized, they often struggle to reconcile the competing demands of protocol accessibility and robust security. Achieving a balanced approach is essential for enhancing M2M communication security within the IIoT landscape.

Yu et al. [10] proposed a data encryption transmission algorithm, MQTT Encryption Algorithms (MQTT-EA), based on MQTT, which requires no additional hardware support, is simple to implement, and is cost-effective. Zheng et al. [11] proposed a new lightweight mutual authentication and key exchange protocol based on PUF for peer-to-peer (P2P) IoT applications. Chao et al. [12] proposed the MQTT-SE data encryption transmission algorithm, which includes a bidirectional authentication scheme based on symmetric encryption, public key, and public key certificates. Although these schemes offer security improvements, they may suffer from performance drawbacks and increase computing and communication overhead.

Wang et al. [13] proposed a PUF-based RFID security authentication protocol to guard against physical cloning. The protocol ensures communication security using two cryptographic primitives, PUF and hash functions, with all communications encrypted to guarantee information privacy and security. He et al. [14] proposed a lightweight two-party authentication and session key (SK) exchange protocol, which performs security authentication and establishes a shared SKbetween a PUF-enabled cryptosystem and a server. Xu et al. [15] combined hash functions and PUF to design a MAC algorithm, but this algorithm has high storage and communication overhead, making it less suitable for resource-limited IoT devices.

Wang et al. [16] proposed an efficient anonymous identity authentication protocol for lightweight IoT devices. The protocol is based on PUF technology and implements security attributes such as anonymity, confidentiality, untraceability, forward security, and resistance to modeling attacks. The authors validated the protocol's security using formal security models and ProVerif, demonstrating that it offers significant advantages in terms of computational, storage, and communication overhead, making it ideal for resource-constrained IoT devices.

Ma et al. [17] proposed PUF-RAKE, a lightweight and highly reliable authentication and key establishment protocol based on PUF. By employing error correction and dynamic CRP obfuscation techniques, the protocol enables efficient device authentication while preventing masquerading, brute force, replay, and modeling attacks. By implementing error correction on the server side and utilizing a lightweight stream authentication mechanism, PUF-Rake significantly reduces resource usage while maintaining security, making it particularly suitable for resource-constrained IoT devices.

Bian et al. [18] proposed Bio-AKA, a two-factor user authentication and key protocol that combines PUF with fingerprint biometrics. By employing PUF technology and a fuzzy

extractor, the scheme enables secure user authentication and key negotiation without a password, protects user anonymity and privacy, and prevents fingerprint information leakage.

Many existing solutions in this area struggle with excessive performance overhead and high computational and communication costs, particularly in resource-constrained environments. In this paper, we introduce PUF technology, combined with hash functions and XOR operations, to achieve lightweight two-way authentication and secure communication between devices. This approach reduces computational and communication costs while enhancing privacy protection and resistance to attacks, effectively addressing the limitations of previous solutions.

Additionally, this paper provides a comparative analysis of the aforementioned works [10–18] in the subsequent chapters. In summary, the diverse solutions proposed in recent years for IIoT M2M security protocols have notably enhanced security measures, especially for lightweight IIoT devices with limited resources. Nonetheless, these schemes still face challenges related to performance, computational overhead, and practical application. Therefore, further research and optimization are needed to balance security and availability, and to enhance their applicability in real-world scenarios.

3. Preliminaries

In this Section, we examine the necessity of using the MQTT-SN protocol in the context of IIoT and its associated challenges in resource-constrained networks. Next, we introduce the selection criteria for PUFs and their application in IIoT devices, with a focus on addressing the jitter problem in PUF outputs [19]. Finally, we briefly discuss the ProVerif, emphasizing its role in analyzing and verifying security protocols. This discussion lays the foundation for subsequent research, aiming to enhance the security and reliability of protocol design in IIoT systems.

3.1. IIoT MQTT-SN Environment and Constraints

The expansion of the IIoT model introduces challenges in quickly detecting and assessing attacks on co-existing systems. One of the most common methods used by cybercriminals is exploiting vulnerabilities in communication protocols, which can enable them to access, modify, and delete data, or even disable devices or entire infrastructure. In the context of IIoT, the MQTT protocol is widely used due to its portability, enabling resource-constrained devices to communicate with each other [20]. To enhance its effectiveness, a lightweight version of the protocol, MQTT-SN, has emerged. MQTT-SN is suitable for Wireless Sensor Networks (WSNs), networks that do not support TCP/IP, and those with limited bandwidth. In these constrained network environments, MQTT-SN extends MQTT's functionality, including features such as subject name registration, gateway discovery, sleep and wake mechanisms, and different Quality of Service (QoS) levels [21]. The architecture diagram of the MQTT-SN protocol is shown in Figure 1.

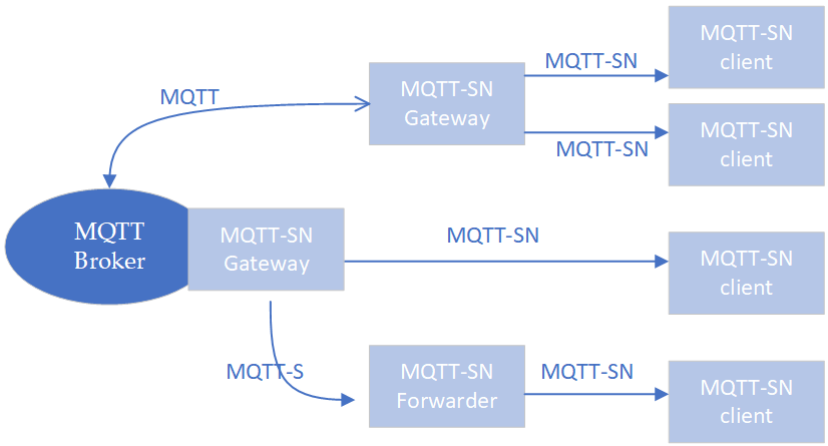


Figure 1. MQTT-SN protocol architecture diagram.

In the IIoT field, MQTT-SN can adapt to various network environments by utilizing different underlying protocols, including but not limited to 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks) [22]. 6LoWPAN is a network protocol designed for low-power wireless personal area networks, such as IEEE 802.15.4 standard devices [23]. It enables these devices to communicate over IPv6, efficiently using limited bandwidth and adapting to scenarios requiring low power consumption and high network reliability [24]. When 6LoWPAN is used as the underlying protocol, it introduces a fragmentation mechanism to support larger packet transfers, as it was designed to accommodate the small packet structure of the IEEE 802.15.4 standard, which typically has a maximum payload of 127 bytes [25]. However, to ensure efficient communication, the maximum data size for a single packet without fragmentation is usually limited to the maximum frame size of IEEE 802.15.4. After subtracting the security header, MAC layer header, and 6LoWPAN header, the remaining data available for applications are approximately 40 to 100 bytes, depending on header usage and security settings. This limitation is why MQTT-SN is designed to reduce message size and optimize data transfer for resource-constrained network environments.

In the IIoT field, where MQTT-SN and 6LoWPAN are used as low-level communication protocols, restricted devices typically exhibit characteristics such as low power consumption, limited processing power, small storage space, low bandwidth connections, and simplified communication protocols [26]. According to RFC 7228, restricted devices are roughly categorized into Class 0 (C0), Class 1 (C1), and Class 2 (C2) devices. In this field, restricted devices mainly refer to Class 1 and Class 2 devices, which can support 6LoWPAN and MQTT-SN/MQTT communication, either directly or through appropriate adaptation layers [27]. These devices can effectively participate in the IIoT network, performing data collection, processing, and communication tasks despite their limited resources. Since the protocol discussed in this article utilizes PUF and requires a long-term private key and secure storage area, it is not suitable for extremely restricted devices, such as Class 0 devices.

3.2. Selection and Description of PUFs

PUF is a security technology based on the unique physical characteristics of hardware. It leverages the minute physical variations that occur during device manufacturing to generate a unique and irreproducible "fingerprint" for device authentication and key generation. PUF technology is inherently unpredictable and unique, making it highly effective against counterfeiting and cloning attacks, which presents significant advantages for resource-constrained IoT devices. In this study, we utilize PUF technology to generate unpredictable responses, combined with hash functions and XOR operations, to implement a lightweight two-way authentication and secure communication protocol. This solution not only effectively counters forgery and replay attacks, but also significantly reduces computational and communication overhead while enhancing the device's privacy protection capabilities.

PUFs are widely used in the security field, particularly in device authentication and key generation. Based on their physical characteristics and implementation methods, PUFs can be categorized into silicon-based PUFs, optical PUFs, acoustic PUFs, magnetic PUFs, and biological PUFs. Silicon-based PUFs further include SRAM PUFs, array PUFs, and flip-chip PUFs [28]. In the IIoT, a simple and feasible PUF implementation is based on SRAM. This type of PUF leverages the random behavior of SRAM when the storage unit is not initialized to generate a unique fingerprint. The advantages of SRAM PUFs include easy implementation and high resistance to replication, but they require a certain amount of storage space to generate sufficient entropy. However, this PUF can be implemented using existing inherent equipment [29].

However, regardless of the type of PUF, it is susceptible to environmental changes (temperature, voltage, humidity, etc.) and equipment aging, which can lead to output errors. Therefore, in practical applications, minimizing output jitter is crucial. Several commonly

used methods can significantly reduce PUF output jitter and the likelihood of errors. These methods include the following: first, designing a robust PUF structure to maintain output consistency under different operating conditions; second, calibrating the environment and operating conditions to record the PUF output under standard conditions before use; third, applying error correction coding (such as Hamming code, BCH code, or Reed-Solomon code) to ensure that the original PUF output can be reconstructed even if an error occurs in the PUF response; additionally, reliability can be further improved through response filtering and post-processing algorithms, such as conformance testing to remove unstable bits and retain only those that remain consistent across multiple measurements; finally, by performing multiple measurements and statistical analysis of the PUF, the response with the least variation is selected as the effective output, reducing the impact of output jitter.

3.3. ProVerif Tool

ProVerif is a powerful automated tool for analyzing and validating security protocols, based on the formalized pi-calculus model. It can automatically verify security attributes such as confidentiality, authentication, and forward security by simulating the behavior of potential attackers. ProVerif helps detect security vulnerabilities in protocols and analyzes how they perform under different attacker models [30].

The advantage of this tool lies in its automated processing capability, enabling rapid verification of complex protocol designs and greatly reducing the effort and error likelihood associated with manual analysis. ProVerif is widely used in academic research and practical engineering to analyze network security protocols, such as TLS and IPsec, ensuring they are designed and implemented to withstand various known attacks.

Given its powerful verification capabilities, ProVerif is an indispensable tool in secure protocol design and analysis, particularly in scenarios requiring a high degree of security.

3.4. Security Mechanism

The fail-stop mechanism is a protective strategy designed to immediately terminate the current operation or protocol execution if the system detects an anomaly or potential security threat, preventing further damage or information disclosure. The core idea of this mechanism is to ensure that the system does not further expose its vulnerabilities or allow attackers to exploit them by halting operations when it cannot determine whether it is safe to continue. This mechanism activates when the system detects errors, abnormal behavior, or potential attacks, such as replay attacks, man-in-the-middle attacks, desynchronization attacks, or other security threats, causing the system to immediately stop the current process. This prevents erroneous data from being further processed and blocks attackers from exploiting erroneous states.

To ensure that the protocol can respond promptly to potential attacks or abnormal conditions, this paper introduces the fail-stop mechanism. This mechanism prevents further damage or information leakage by halting protocol operations as soon as an error or abnormal behavior is detected. The fail-stop mechanism is particularly effective against threats such as denial-of-service (DoS) attacks and replay attacks, providing an additional layer of security to the system by aborting operations promptly.

4. Proposed Protocol Scheme

Before presenting the protocol scheme, this Section outlines the basic assumptions and required environment for the protocol design. This Section not only provides the necessary background to understand how the protocol works but also clarifies its applicability and limitations. The protocol environment is then described in detail, including the participants, required hardware, and assumptions about operating conditions.

4.1. Environment and Assumptions

In the communication model based on the MQTT-SN protocol presented in this paper, the main roles are the Device and the Broker. The Device typically refers to an endpoint

in an IoT environment, such as a sensor, actuator, or any other smart device capable of connecting to the network. The Broker, on the other hand, receives, processes, and forwards messages, acting as a server that manages connections, sessions, and message routing for all clients. Although it is somewhat equivalent to a server, the term "Broker" is used to emphasize its role in message forwarding and management, rather than as a traditional server, especially in the context of the MQTT-SN protocol [31].

In this research, the relationship between the Device and the Broker is one-to-many. The Device is resource-constrained, while the Broker is resource-unrestricted. The protocol is initiated by the Device, and the Broker responds. The protocol is divided into three phases: registration, authentication and secret key exchange, and PUF shuffling. Both parties need to store pre-shared data, including a pre-shared secret key (PSK), a pre-shared secret (S), and 30 sets of CRPs assumed for the PUF. The protocol does not include an independent logout phase; devices that have not been contacted for a long time are considered logged out. Additionally, this paper assumes that both communicating parties are equipped with long-term secret keys at the factory. The pre-shared secret, AID, and CRP messages are stored in the non-volatile memory (NVM, such as EEPROM) of the device, and in the broker's database. The PSK is stored encrypted with the long-term secret key of both parties. The derived shared secret, denoted as S, is safeguarded in the device's secure storage compartment, potentially within a Hardware Security Module (HSM), and similarly within the broker's secure environs, perhaps utilizing the broker's Trusted Platform Module (TPM) for enhanced security.

The QoS mechanism of the MQTT protocol offers varying degrees of message transmission guarantees through three levels (QoS 0, 1, 2). QoS 0 does not guarantee message delivery, QoS 1 ensures the message is delivered at least once, potentially with duplication, and QoS 2 ensures the message is delivered only once without duplication. This mechanism allows for flexible selection of message reliability and transmission overhead based on requirements in resource-constrained network environments. The QoS mechanism of the MQTT-SN protocol is cleverly embedded into multiple design stages to ensure the reliability and security of data transmission. The QoS mechanism plays a key role, particularly in ensuring message transmission and security. For example, when a message is transmitted between a device and a Broker, the QoS level determines whether the message may be sent multiple times to ensure successful delivery. In the AKE phase, the proposed protocol requires MQTT QoS Level 3, which includes an acknowledgment mechanism to ensure that the message is neither lost nor duplicated, assuming the MQTT communication link remains intact.

In this paper, the administrator must determine the bit lengths of all parameters, the frequency of CRP updates, the survival time of encrypted sessions, and the maximum error counter value based on the specific usage scenario. Under typical conditions without special security requirements, this paper assumes that the input and output of the PUF are 128-bit binary, the hash function uses SHA3-256, truncating the first 128 bits as the output, and other parameters such as the pre-shared secret key and secret S are 128 bits, while the timestamp is 64 bits. The session survival time is set to 24 h, and the number of CRPs updated at a time is 30 (implying a PUF update cycle of approximately 30 days). The error counter is capped at 5, and when it reaches this limit, corresponding actions will be taken, such as blocking the channel or triggering an alarm. This protocol assumes that all entities are equipped with an error counter for any step involving parameter verification, such as verifying the consistency of hash values, during protocol operations. With this design, the system can issue an out-of-band alarm to the upper layer when a specific error counter exceeds a predetermined threshold while implementing the fail-stop mechanism. This mechanism supports attack detection across the entire system.

Finally, the protocol assumes that the PUF has undergone the error correction processing described in Section 3.2, and the specific details of this processing are not further elaborated. Table 1 lists all the symbols used in this paper and their descriptions.

Table 1. Notations and Descriptions.

Notation	Description
N_a, N_b, $Tn_{1\sim7}$	Random numbers
AID_i	Identity identifier of the Device
PSK_i	Pre-Shared Key
CRP	Challenge-response
C-HRPs	Challenge-Hushed Response Pairs
T_1, T_i, T_r	Timestamps
S	Pre-shared secret
C	Challenge
R	Response
hR	Response encrypted using a hash function
PUF()	Physically Unclonable Function
n_i	A random set of numbers generated by the broker
ESK	Used SK for encryption
DK	Device's long-term private key
BK	Broker's long-term private key
h()	Cryptographic hash function
$\oplus$	Bit-wise XOR operation
a | b	Cut lower b bits from a
X *	The parameter was taken from the database
X′	Received value or the value computed from received values

* indicates the updated value.

The proposed scheme consists of three main phases: the Registration phase, the Authentication and Key Exchange phase, and the PUF-Shuffling phase. The specific steps for each phase are detailed below.

4.2. Registration Phase

This phase takes place in a secure channel between the device and the Broker, and the device synchronizes the clock with the Broker. The steps are shown in Figure 2.

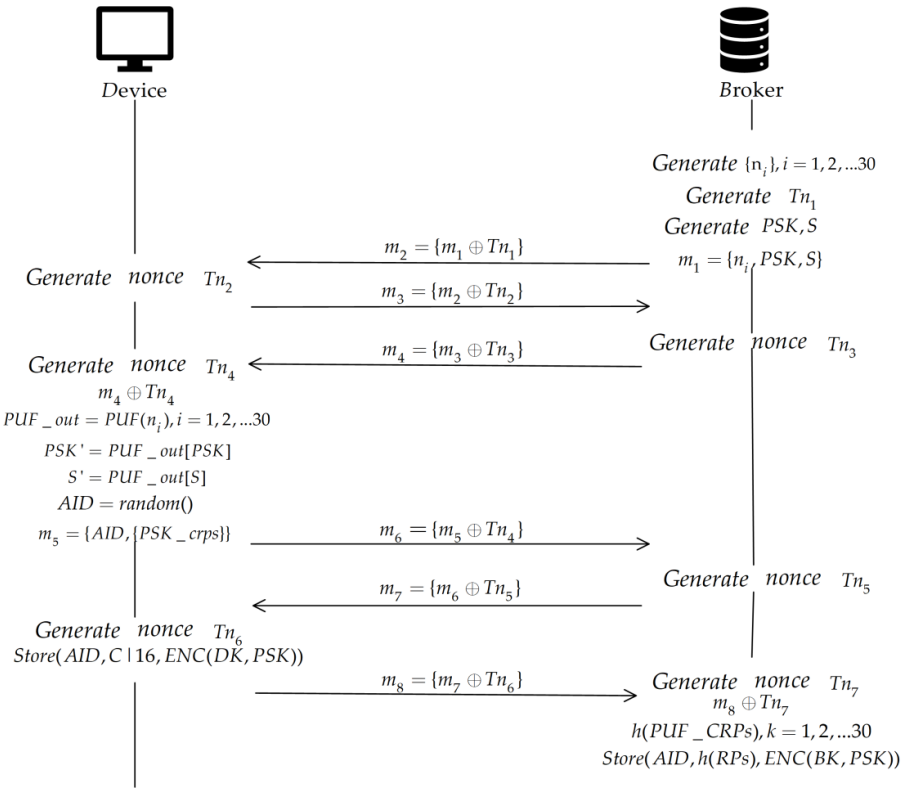


Figure 2. Registration phase (Secure channel).

Step 1: Device ← Broker, $m_2 = \{m_1 \oplus Tn_1\}$.

The Broker generates a random number group n_i, pre-shared key PSK, and random number S containing 30 pieces of information and packages them as m_1, then generates

a random number with the same length as m_1, performs an XOR operation with it, and finally sends the obtained result $m_2 = m_1 \oplus Tn_1$ to the Device.

Step 2: Device $\rightarrow$ Broker, $m_3 = \{m_2 \oplus Tn_2\}$.

Upon receiving m_2, the Device generates a new random number of the same length as it and sends its XOR result $m_3 = m_2 \oplus Tn_2$ back to the Broker over the secure channel.

Step 3: Device $\leftarrow$ Broker, $m_4 = \{m_3 \oplus Tn_3\}$.

Upon receiving m_3, the Broker generates a new random number of the same length as it and sends their XOR result $m_4 = m_3 \oplus Tn_3$ back to the Device over the secure channel.

Step 4: Device $\rightarrow$ Broker, $m_6 = \{m_5 \oplus Tn_4\}$**.**

The Device receives m_4, generates a new random number of the same length, obtains their XOR result m_5 and sends it back to the Broker. The Device computes the PUF response sequence based on the received random number n_i. And use these PUF responses to generate a new key PSK 'and a random number S' Then, the Device generates a random identifier AID and sends the message m_6 containing the identifier AID and the XOR result of PSK 'and S' to the Broker through a secure channel.

Step 5: Device $\leftarrow$ Broker, $m_7 = \{m_6 \oplus Tn_5\}$.

Upon receiving m_6, the Broker generates a new random number Tn_5 and sends the XOR result $m_7 = m_6 \oplus Tn_5$ of message m_6 and Tn_5 back to the Device through the secure channel.

Step 6: Device $\rightarrow$ Broker, $m_8 = \{m_7 \oplus Tn_6\}$.

The Device generates a new random number Tn_6 and sends the received message m_7 with the XOR result $m_8 = m_7 \oplus Tn_6$ of Tn_6 to the Broker over a secure channel.

Step 7: Broker.

The Broker computes the hash of the PUF CRPs and generates the encrypted key information $ENC(BK, PSK)$.

At the end of this phase, the device and the Broker will each maintain a record of the other. Specifically, the device stores <AID, C | 32, DK $\oplus$ PSK> in NVM and <S> in a secure storage area. The Broker stores <AID, C-HRPs, BK $\oplus$ PSK> in its database and <S> in a secure storage area.

4.3. Authentication and Key Exchange Phase

This phase occurs in an open channel between the device and the Broker. To ensure communication security in an insecure environment, the protocol employs encryption and authentication mechanisms. The steps are illustrated in Figure 3.

Step 1: Device $\rightarrow$ Broker, $msg1 = \{AID_i, B_1, B_2, T_1\}$.

The device generates a random number N; obtains the current timestamp T_i; calculates $B_i = N \oplus h(PSK_i, S_i, T_i)$, where $\oplus$ represents XOR operation, h is a hash function, PSK_i is a pre-shared key, and S_i is the state of the device; sends $msg1 = \{AID_i, B_1, B_2, T_1\}$ to the broker.

Step 2: Device $\leftarrow$ Broker, $msg2 = \{C_i, B_3, B_4, T_2\}$.

The broker verifies the validity of the timestamp by obtaining the current timestamp T_r and checks $T_r - T_i$ whether it is smaller than the ΔT. Uses the device AID_i to find relevant data in the database; then, calculates $N_i\prime = B_i \oplus h(PSK_i, S_i, T_i)$, uses the resulting sum, $AID_i\prime$ and $T_i\prime$ hash and checks whether the result is equal with $B_2\prime$; if not, the protocol is terminated, the error count is increased by 1, and the system determines whether to send an alarm to the upper layer. Otherwise, then select a challenge-response to CRP from the database and generate a random number N_b; calculate $B_3 = N_b \oplus h(PSK_i, S_i, T_2)$ and $B_4 = h(C, hR, N_a, N_b, T_2)$; at this point, CRP is marked in the database as used, and $msg2 = \{C_i, B_3, B_4, T_2\}$ is sent to the device.

Step 3: Device $\rightarrow$ Broker, $msg3 = \{B_5, T_3\}$.

The device obtains the current timestamp T_i again and checks $T_r - T_i$ if it is less than ΔT; if not, terminate the protocol operation. Otherwise, if the time is valid, the device removes C | 16 (timeout challenge) from the challenge manager CM and makes sure N_i' is new. After the device calculates $N_b' = B_r' \oplus h(PSK_i, S_i, T_i')$ and checks the value of B_r,

where $N_b' = B_r' \oplus h(PSK_i, S_i, T_i')$, $R' = PUF(C')$; the device then computes the new AID_i, PSK_i, and SK; device generation $B_s = h(h(R'), S_i, N_b, T_i)$; brings new AID_i; finally, the device sends $msg3 = \{B_5, T_3\}$ to the broker.

After calculating the new AID_i, PSK_i, and S_i, the device must also store the previous values of AID_i, PSK_i, and S_i. This ensures that synchronization with the Broker can be restored in case of a power outage, system restart, or desynchronization attack. The specific recovery process is as follows: First, the device attempts to send msg1, generated using the new synchronization data, to the Broker and waits for a response. If no response is received after three attempts, the device then tries to send msg1 generated from the previous synchronization data. If there is still no response after another three attempts, the device repeats the process after a waiting period, continuing until synchronization is successfully restored.

Step 4: Device ← Broker, $msg4 = \{B_6, T_4\}$.

The broker obtains the current timestamp T_r and checks $T_r - T_i'$ if it is less than ΔT; if not, terminate the protocol operation. Otherwise, calculate $B_s' = h(hR, S_i, N_b, T_i')$; update $AID_i = h(AID_i, N_a \oplus N_b, hR')$ in the database; then, calculate; calculate $PSK_i = h(N_i, PSK_i, hR')$ and $S_i = h(N_a, N_b, S_i)$; calculate the session key $SK_i = h(h(N_a, N_b) \oplus hR')$; check whether CRP in the database is less than 2; if B_s' is received, then PUF shuffling is performed. Last, send $msg4 = \{B_6, T_4\}$ to the device.

Step 5: Next steps (optional).

Once the broker sends msg4, it means that the server finds C-HRPs exhausted. Shuffling is required. Thus, the PUF shuffling phase is entered.

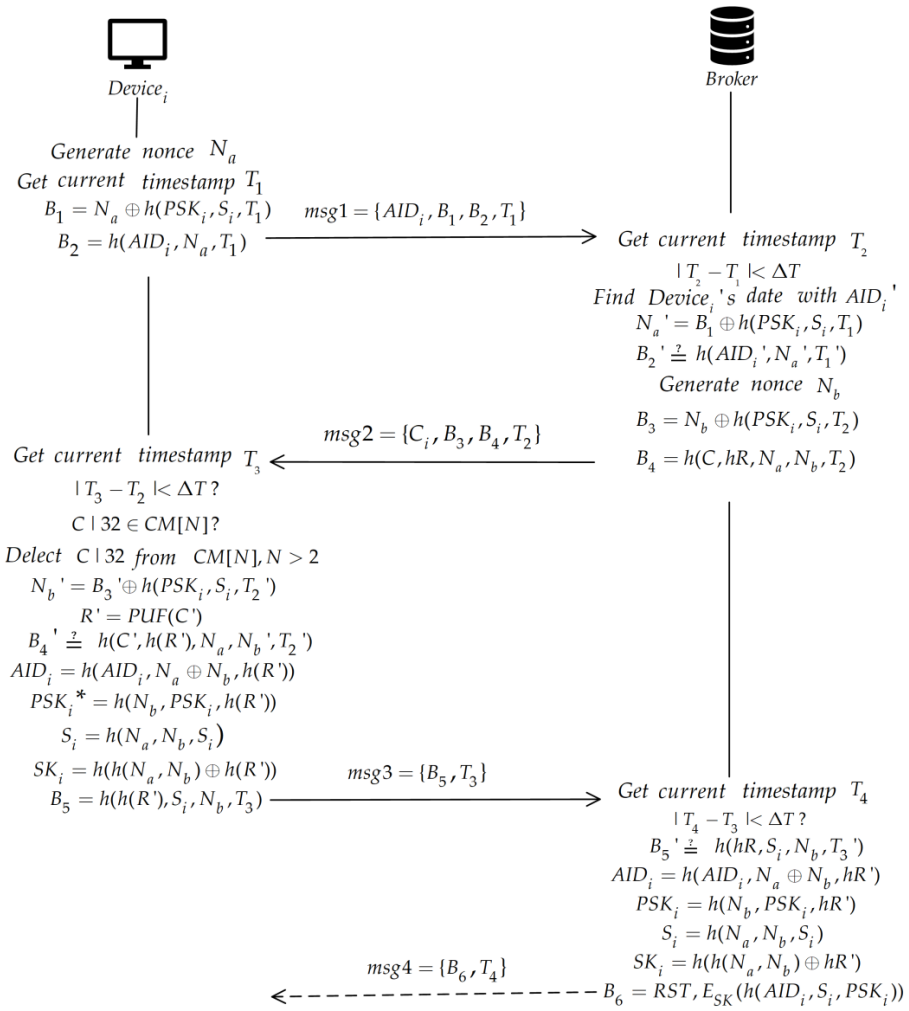


Figure 3. Authentication and Key Exchange phase (Unsecure channel).

4.4. PUF Shuffling Phase

This phase occurs in an insecure channel between the device and the broker. These steps are shown in Figure 4.

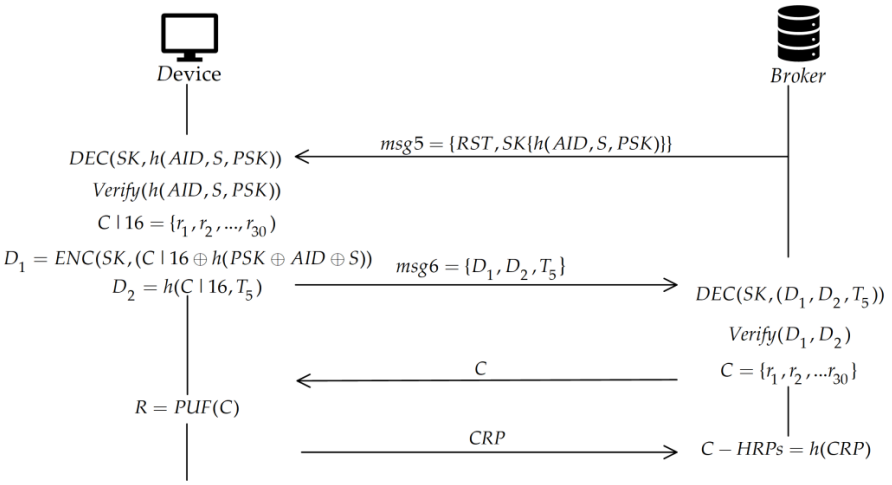


Figure 4. PUF shuffling phase (Unsecure channel).

Step 1: When the time since the last CRP update exceeds a certain threshold (e.g., 25 days), or the number of CRPs stored by the broker is less, after completing the AKE phase above, the broker sends an optional response $msg5 = \{RST, SK\{h(AID, S, PSK)\}\}$.

Step 2: After receiving msg5, the device decrypts the received information with SK to verify whether the result is equal to $h(AID, S, PSK)$; if they are equal, enter the PUF update phase, generate 30 16-bit random numbers, and save each 16-bit number as an element to the array C | 16. On SET, $D_1 = ENC(SK, (C|16 \oplus h(PSK \oplus AID \oplus S))$, $D_2 = h(C|16, T_5)$ (On SET is a circular completion of lack of bits) is calculated, and $msg6 = \{D_1, D_2, T_5\}$ is sent to Broker.

Step 3: After receiving it, the Broker decrypts it, verifies it, generates 30 112-bit random numbers, concatenates them one by one to C | 16, forms a complete C, and then sends it to the device.

After receiving it, the device inputs C one by one into the PUF to obtain R, which is spliced into CRP and sent to the Broker.

Upon receipt, the Broker validates, hashes all the responses in the new CRP, and overwrites the resulting CRPs into the database.

At this point, the PUF shuffling phase is complete.

5. Security Verification

5.1. Formal Security Verification

In this paper, the Dolev–Yao attacker model is used for formal analysis, assuming that the attacker possesses powerful capabilities, such as controlling the network, replaying messages, and having comprehensive knowledge of the protocol, to test the protocol's security in harsh environments. By simulating various attacker operations, the Dolev–Yao model can evaluate the security attributes of protocols in different scenarios. To further verify the security of the protocol, this paper introduces the formal verification tool ProVerif, which can automatically analyze the protocol, detect its resistance to various attacks, and ensure that the designed security protocol is effective and reliable in practice.

ProVerif is a powerful automated tool for formal verification and analysis of security protocol attributes. Developed by Bruno Blanchet, ProVerif employs model checking and theorem-proving methods based on Horn clauses. It can handle an infinite number of sessions and complex messaging patterns, including encryption and decryption operations. By modeling the behavior of both the protocol and the attacker, ProVerif generates a series

of queries to detect potential security vulnerabilities and verify attributes such as confidentiality, authentication, and integrity. Its automation capabilities and wide applicability make it a popular choice in academic research and the design and evaluation of practical security protocols. ProVerif can also effectively verify the robustness of protocols against various attack modes, ensuring resistance to known attack methods. The verification results are shown in Figure 5.

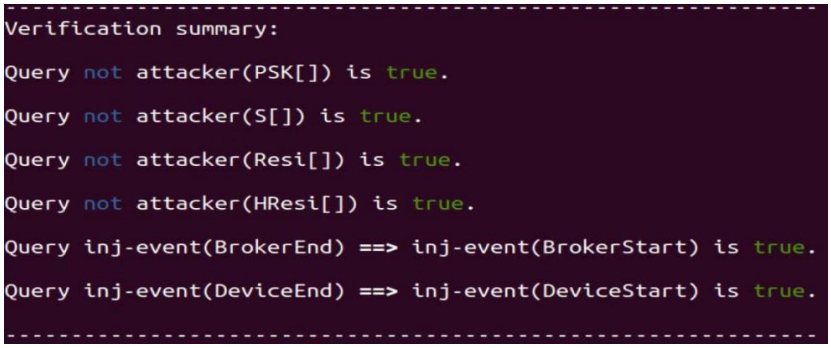


Figure 5. Verification result.

The result of this verification is "true" showing some results of this query:

1. Query not attacker (PSK []) is true: indicates that the pre-shared key (PSK) is not obtained by the attacker.
2. Query not attacker (S []) is true: the secret S is not obtained by the attacker.
3. Query not attacker (Resi []) is true: the response value Resi is not obtained by the attacker.
4. Query not attacker (HResi []) is true: indicates that the HResi response value is not obtained by the attacker.
5. Query inj-event (BrokerEnd) ==> inj-event (BrokerStart) is true: Indicates that if a BrokerEnd event occurs, a BrokerStart event must also occur. This is a causal verification, and the result is true, indicating that the causal relationship is valid.
6. Query inj-event (DeviceEnd) ==> inj-event (DeviceStart) is true: If the DeviceEnd event occurs, the DeviceStart event must also occur. This is a causal verification, and the result is true, indicating that the causal relationship is valid.

The significance of these verification results is that the security and event-triggering logic of the system are guaranteed, with all key security properties and event logic validated in the verification. The formal verification using the ProVerif demonstrates that this protocol maintains all required security properties under the Dolev–Yao attacker model.

5.2. Informal Security Verification

Informal security verification aims to evaluate and demonstrate the security of a protocol through logical reasoning and intuitive interpretation. This method does not rely on complex mathematical proofs but instead focuses on a step-by-step analysis of the protocol's various steps and defense mechanisms, ensuring that each step is protected against potential attacks. This verification approach helps identify and correct security vulnerabilities at an early stage of protocol design, providing easy-to-understand security assessment results. The following are the informal security validations for the proposed scheme.

5.2.1. Confidentiality

Once the device is authenticated and a secure connection to the server is established, all communication data are encrypted using a SK. This ensures that even if the data are intercepted in transit, a third party without the key cannot interpret the content, thereby guaranteeing the confidentiality of the data. To further enhance security, the system periodically updates the SK. By automatically changing the key after each communication session

or at regular intervals, the security of new communications remains unaffected even if the old key is compromised. Additionally, the critical data are encrypted, preventing attackers from obtaining the plaintext of the critical data, thereby ensuring the confidentiality of the protocol.

5.2.2. Prevent Replay Attacks

Timestamps and random numbers are employed in the protocol to prevent replay attacks. In each communication, the device and broker generate new random numbers, and each message includes the current timestamp. The broker verifies that the difference between the timestamp in the message and the current time is within the allowed time window, thereby preventing replay of old messages. Additionally, this protocol uses invalid flags and delete operations to prevent attackers from using invalid or outdated data in replay attacks by checking the validity of entries and deleting those that are invalid.

5.2.3. Prevent Man-in-the-Middle Attacks

The two parties in the communication share a PSK and multiple C-HRPs. These shared data enable both parties to verify each other's identity during communication. The uniqueness of PUF technology ensures that each device's response is unique and unpredictable, making it impossible for an attacker to spoof the identity authentication of either party.

5.2.4. Key Compromise Impersonation (KCI) Attack

The non-clonability of the PUF in this protocol ensures that even if the private key is compromised, an attacker cannot replicate the device's unique response. Random number generation and verification ensure the uniqueness and security of each communication, preventing attackers from exploiting the leaked key. In the proposed protocol, secrets are stored in different areas. Even if an attacker gains access to the long-term private keys of both the device and the Broker and successfully recovers the encrypted PSK, they still cannot calculate critical data, such as the SK and the AID used in the next round, because they cannot access the secret S stored in the secure area. Therefore, the proposed protocol can effectively resist KCI attacks and maintain the security of communication.

5.2.5. Perfect Forward Security

Perfect forward security ensures that even if a session key is compromised in the future, the contents of previous communications remain secure. This protocol ensures both forward and backward security in the communication process through a multi-level key protection mechanism. First, the SK is calculated based on various factors such as dynamically generated random numbers, timestamps, and PUF responses in each communication. Therefore, even if the SK in one communication is compromised by an attacker, the previous and subsequent communication content will not be affected because the key used in each communication is independent.

5.2.6. Prevent Data Tampering

Preventing data tampering is crucial to ensuring the integrity and consistency of communication data. This paper achieves this goal through a multi-layer design, including encryption to protect transmitted data, hash verification to detect data integrity, and the use of timestamps and random numbers to ensure data timeliness and uniqueness. Additionally, an NVM update mechanism was designed to ensure data consistency after the device is powered down or restarted, further preventing data tampering. These protocol-level designs jointly ensure the security and reliability of data during communication.

5.2.7. Privileged Insider Attack

Preventing insider authorization attacks involves protecting against system insiders who may abuse their privileges to perform unauthorized operations or access sensitive

information. In the proposed scheme, even if authorized personnel obtain all CRPs stored in plaintext in the TA database during the AKE phase, they still cannot calculate other critical data because they cannot access the PSK and secret S. As a result, they cannot intervene in or influence the negotiation process between any entities. Furthermore, even if an authorized attacker attempts to obtain sensitive information by eavesdropping on the secret channel during the registration phase, they will not succeed. This is because, during the registration phase, the proposed protocol uses XOR operations to encrypt each transmitted data, ensuring that no plaintext is transmitted over the channel. Even if the attacker eavesdrops, they cannot obtain all the parameters needed to calculate the critical data. Therefore, the proposed scheme demonstrates strong resistance to insider authorization attacks.

5.2.8. Mutual Authentication

During AKE, the Broker authenticates the device's identity by sending a PUF challenge and verifying the device's PUF response. In turn, the device verifies the Broker's identity by checking the PSK sent by the Broker and the parameters calculated using secret S. This two-way verification mechanism ensures the authenticity of both parties' identities, achieving the security goal of two-way authentication.

5.2.9. Device Anonymity and Untraceability

Each time a device authenticates, a one-time temporary identity AID is generated based on the random challenge of the current session. This ensures that even if multiple communications from the same device are intercepted, an attacker cannot correlate them to identify the specific device. Additionally, during the authentication process, the protocol verifies identity through the device's PUF response without directly conveying any fixed identity information. Because PUF responses are physically based and unique, they inherently provide a layer of anonymity to the device.

Untraceability refers to the ability to protect a device's identity from being identified and tracked by an attacker analyzing communication data across multiple interactions. In the proposed protocol, all parameters transmitted over the insecure channel change with each round of communication, preventing an attacker from tracing the same device by eavesdropping on fixed parameters during different rounds of protocol execution. This mechanism enables the protocol to achieve untraceability, effectively protecting the device's privacy and preventing its identity from being tracked and identified during communication.

5.2.10. Resist Modeling Attack

A modeling attack involves collecting the input and output data of a device to build a model that predicts and falsifies the response of the PUF. In this paper, in the proposed protocol, the PUF response is neither transmitted in plaintext over the channel nor directly stored in the Broker's database. Instead, the Broker's database stores the hashed PUF response data. As a result, even if an attacker eavesdrops on the communication channel or gains access to all the data in the database, they still cannot retrieve the original PUF's CRPs of the device. Due to the uniqueness and unpredictability of the PUF response, and the fact that it is never exposed in plaintext, an attacker cannot fabricate the device's PUF response by modeling the acquired data. This design greatly enhances the protocol's resistance to modeling attacks, ensures the security of device identity authentication, and further improves the overall protection level of the system. By leveraging this mechanism, the proposed protocol effectively prevents attackers from reconstructing and mimicking the device's behavior through eavesdropping or data leakage, thereby ensuring the reliability and security of the device authentication process.

5.2.11. Resist Desynchronization Attack

An anti-desynchronization attack aims to prevent an attacker from interfering or tampering with data during communication, which could result in inconsistent information between the device and server, rendering the protocol invalid. This paper primarily addresses the data update issue in the third step of the protocol. If a desynchronization attack occurs, the device and server may store inconsistent keys, random numbers, or other critical data, leading to failures in subsequent authentication or communication processes.

The QoS mechanism of the MQTT-SN protocol enables the device to detect desynchronization in a timely manner. Specifically, if the device does not receive an ACK response after sending msg3, it recognizes that msg3 may not have been delivered correctly and immediately attempts to resend the AKE request with updated data. If the server remains unresponsive after more than three retries, the device will attempt to resend the AKE request using parameters such as PID and S from the previous round. Through this mechanism, even if desynchronization occurs between the device and the Broker due to msg3 loss, the protocol can restore synchronization using the steps described above, ensuring continuous and reliable communication. This design not only effectively addresses potential desynchronization attacks but also maintains data consistency between the device and the server during communication, thereby enhancing the protocol's robustness and security.

5.2.12. Denial of Service (DoS) Resistance

DoS resistance is a security measure designed to prevent system resources from being exhausted by detecting and blocking a large number of invalid or malicious requests, thereby ensuring the availability and stability of services. In general, security protocols alone are typically insufficient to resist DoS attacks directly and often require the combination of additional software and hardware defense measures. However, the proposed protocol makes a proactive attempt to resist DoS attacks. Specifically, the protocol introduces a fail-stop mechanism, which halts its operation as soon as an error is detected. This design endows the protocol with a certain level of "attack-awareness." For example, during the AKE phase, if the protocol receives msg1 with constant parameters multiple times, it can detect that a replay attack may be occurring.

To further mitigate the impact of DoS attacks, the protocol tracks the frequency of errors of the same type. If the same type of error occurs frequently within a short period, the protocol will temporarily block the channel, reducing the risk of DoS attacks caused by a large volume of error messages. By actively sensing and responding to abnormal behaviors at the protocol level, the protocol can mitigate the impact of DoS attacks and enhance the system's overall resilience. Although this design cannot completely prevent DoS attacks, it enhances the protocol's robustness and its ability to resist attacks.

5.2.13. Intrusion Detection

In this paper, we explore and implement an intrusion detection mechanism within the security protocol design, aiming to enhance communication security in the IIoT environment. Specifically, the protocol achieves monitoring and defense against potential intrusions through multi-level security measures. First, the protocol employs a timestamp and random number mechanism to ensure the uniqueness and timeliness of each communication, allowing for the detection of abnormal behaviors such as replay attacks or data tampering. Second, the protocol includes an error counter that triggers security measures when abnormal operations are detected repeatedly. These measures may include blocking further communication or renegotiating security parameters to prevent potential intrusions.

Additionally, the PUF mechanism in the protocol is not only used for security authentication and key generation but also for error correction to identify and exclude abnormal outputs caused by physical attacks or environmental interference. Together, these measures form a dynamic and sensitive intrusion detection system, enabling the protocol to respond to and address various potential security threats in a timely manner, thereby enhancing the overall protection and robustness of the system.

5.2.14. Desynchronization Attack

A desynchronization attack is a method that disrupts the communication process between a device and a server, causing their states or data to become unsynchronized. This can result in failed subsequent communications or authentication attempts. Generally, desynchronization does not occur easily because the QoS mechanism of MQTT-SN ensures reliable message delivery. However, in the AKE phase of the proposed protocol, if the device does not receive an ACK acknowledgment for msg3, it will recognize the situation and immediately recompute msg1 using the last known synchronization data, thereby restarting the handshake process. If the device fails to receive msg3 multiple times consecutively, it can reasonably infer that a desynchronization attack may have occurred. In this case, the device can pause further computation and transmission of msg1 and consider reporting the anomaly to the upper layer.

Additionally, if the device fails to receive msg3 due to a power outage or an unsuccessful restart and is uncertain whether the Broker has updated the synchronization parameters, it will first attempt to compute msg1 with the latest synchronization data and send a new request. If there is no response from the Broker after three consecutive attempts, the device will then recompute and resend msg1 using the previous synchronization data. If no response is received after three more attempts, the device may conclude that the channel is blocked and repeat the process after a delay. This approach ensures that the device can withstand active or passive desynchronization attacks by utilizing a recovery synchronization mechanism in the event of such attacks or other abnormal conditions, thereby maintaining the protocol's robustness and security.

In this chapter, we first conduct a comprehensive evaluation of the proposed protocol through formal security verification and informal security analysis. These analyses not only confirm the protocol's effectiveness in defending against various attacks (such as desynchronization attacks and DoS attacks) but also demonstrate its robustness and adaptability in practical application scenarios. Based on this, we will further conduct a detailed comparison of the protocol's security and performance to evaluate its feasibility and superiority in practical applications. This comparison will help clarify the protocol's performance under different security requirements and provide an important reference for future optimization and application.

6. Security Analysis and Performance Comparison

In this chapter, we conduct a comprehensive analysis and comparison of the security and performance of the protocols. First, the protocol's effectiveness against various attacks is evaluated. Second, we compare the protocol with existing similar protocols to analyze its performance in terms of computational overhead, communication efficiency, and overall security. Through this analysis, we aim to verify the feasibility and superiority of the protocol's design for practical application in the IIoT environment, providing a reference for further optimization and expansion.

6.1. Security Analysis

In this Section, we compare the security of the protocol with similar schemes and evaluate its reliability and resistance to attacks in practical applications by analyzing each security feature individually. The security comparison is presented in Table 2.

The comparison results above demonstrate that our protocol offers superior security compared to similar schemes.

Table 2. Security comparison.

Security Property	Scheme [19]	Scheme [18]	Scheme [17]	Scheme [16]	Our Scheme
Resist replay attacks	√	√	√	√	√
Resistance to man-in-the-middle attacks	√	√	√	√	√
Resist KCI attacks	×	×	×	×	√
Modeling attack resistance	×	×	√	×	√
Resist desynchronization attacks	√	√	√	×	√
Resisting DoS attacks	√	√	×	×	√
Perfect forward safety	√	√	√	√	√
Device anonymity	√	√	×	×	√
Untraceability	√	√	×	×	√
Expandability	×	×	√	×	√

6.2. Performance Comparison

In this Section, we analyze and compare the performance of the proposed scheme, focusing primarily on computing overhead and communication overhead.

6.2.1. Computation and Communication Cost Evaluation

We also compare the proposed protocol with a similar protocol [17–20] and assume that the length of the parameters is shown in Table 3.

Table 3. Each parameter conforms to and length.

Parameters	Conforms	Length (bit)
Identification	ID	64
Random number	n	128
Timestamp	T	32
Challenge	C	128
Response	R	128
Hash calculation result	H	128
XOR calculation result	XOR	The longest between two operands
Elliptic curve cryptography	ECC	160

Generally, when elliptic curve points are used as calculation parameters in a security scheme, only the X-coordinate value is utilized as the operand. This approach is also employed in this paper.

In addition, the symbols and time required for each cryptographic operation are shown in Table 4.

Table 4. Symbols and running time of each operation.

Arithmetic	Symbols	Time (ms)
Hash computation	T_h	0.03
Random number	T_R	0.035
Symmetric encryption/decryption algorithms	T_{sk}	0.075
PUF module	T_p	0.15
MASK/UNMASK functions	T_M	0.686
Helper functions	T_F	0.036
Run time for error correction function recovery	$T_{FE.REP}$	2.85

Therefore, the total computational cost of the protocol in this paper is 0.58 ms, and the communication overhead of both parties is 608 bits (76 bytes). The comparison results with other protocols proposed in the literature are shown in Tables 5 and 6.

Table 5. Comparison of the computational cost of similar schemes.

Schemes	Initiator	Responder	Total
Scheme [18]	$4T_h + 2T_p + 2T_{sk}$	$2T_h + T_p + 2T_{sk}$	$6T_h + 3T_p + 4T_{sk}$
Scheme [17]	$2T_p + 8T_F + 2T_R$	$8T_F + 2T_R$	$2T_p + 16T_F + 4T_R$
Scheme [19]	$5T_M + T_p + 3T_R$	$5T_M + 2T_p + T_R$	$10T_M + 3T_p + 4T_R$
Scheme [16]	$11T_h + T_p + T_R + T_{FE.REP}$	$10T_h + T_p + T_{FE.REP}$	$21T_h + 2T_p + T_R + 2T_{FE.REP}$
Our scheme	$6T_h + T_p + T_R$	$6T_h + T_R$	$12T_h + T_p + 2T_R$

Table 6. Comparison of the communication overhead of similar schemes.

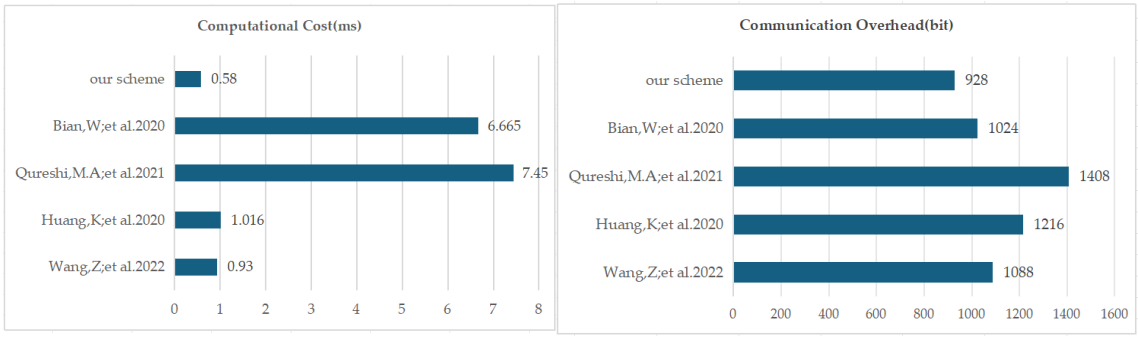
Schemes	Communication Overhead	Number
Scheme [18]	$\oplus \mid +2 \mid H \mid$	3
Scheme [17]	$\oplus \mid$	7
Scheme [19]	$\oplus \mid +4 \mid ECC \mid$	4
Scheme [16]	$\oplus \mid +3 \mid H \mid$	3
Our scheme	$\oplus \mid +3 \mid H \mid +3 \mid T \mid + \mid C \mid$	3

Table 5 shows a comparison of the computational costs of similar schemes, while Table 6 shows a comparison of the corresponding communication costs and the number of interactions. Through the analysis of these two sets of data, the performance of each scheme can be evaluated more comprehensively.

Analysis of Tables 5 and 6 reveals that this scheme offers significant advantages in both computation and communication overhead. Compared to other schemes, this scheme not only maintains low computing costs but also significantly reduces communication costs and interaction times. This balance makes the scheme more efficient in practical applications, particularly in scenarios sensitive to resource consumption and requiring efficient communication. Therefore, this scheme demonstrates excellent overall performance and is well-suited for prioritization in various application scenarios.

6.2.2. Computation versus Communication Cost Comparison

The comparison of communication and computation costs for each scheme is illustrated in Figure 6. This protocol incurs the lowest costs in both categories while also offering superior security and applicability.



(**a**) (**b**)

Figure 6. Calculation cost and communication overhead comparison. (**a**) The comparison of computation costs for each scheme. (**b**) The comparison of communication overhead for each scheme [16–19].

In summary, this paper presents a comprehensive security analysis and performance comparison of the proposed M2M security protocols for IIoT. These analyses reveal the strengths and weaknesses of each scheme in defending against various attack types, such as replay attacks, modeling attacks, and side-channel attacks, and also evaluate their performance in terms of computation, storage, and communication overhead. Based on

the data in the chart, our scheme performs well in terms of computational overhead, with a time of only 0.58 milliseconds, which is at least 43% faster than other similar schemes. In terms of communication overhead, our scheme reduces the communication cost by about 10% compared to the closest competing scheme. These evaluations provide a more complete understanding of the applicability and limitations of these protocols in real-world applications.

7. Conclusions

This paper proposes an authenticated and encrypted channel establishment scheme based on lightweight PUF technology to enhance the security of the MQTT-SN protocol in the IoT environment. By integrating hardware security features and modern encryption technology, the scheme effectively addresses the security and privacy challenges in IoT device communication and demonstrates significant advantages.

First, this paper designs a lightweight security protocol that uses PUF technology to generate unpredictable responses, combines XOR operations and hash functions to achieve mutual authentication and secure communication between the device and the Broker, and maintains low computation and communication overhead in resource-constrained environments. Second, formal verification is conducted using the ProVerif tool, combined with informal analysis, to demonstrate the protocol's effectiveness in resisting common security threats such as replay attacks, man-in-the-middle attacks, and impersonation attacks. Third, the scheme emphasizes the anonymity and untraceability of the device, ensuring that it cannot be identified or tracked by an attacker across different communication rounds, thereby effectively protecting the device's privacy. Finally, the comprehensive fail-stop feature enhances the protocol's attack awareness, allowing the system to quickly respond and send an alarm signal when an attack is detected, enabling timely defensive measures.

In terms of practical performance, the proposed protocol demonstrates excellent efficiency, with a total computation cost of only 0.58 milliseconds and a communication cost of 608 bits (76 bytes). Compared to similar schemes, the proposed protocol reduces computational cost by at least 43% and communication cost by approximately 10%. These quantitative results verify the efficiency and feasibility of the proposed scheme in practical applications, particularly for resource-constrained IoT devices. In conclusion, the proposed PUF-based MQTT-SN protocol provides an efficient and secure solution for IoT communication, fully demonstrating the significant potential of PUF technology in enhancing IoT security.

While this paper has delved into the security challenges of the MQTT protocol within the M2M context of the Client/Server architecture, there are numerous uncharted territories within the IIoT that warrant further exploration. Our future endeavors will extend to encompass a broader spectrum of scenarios, including multicast communication, Device-to-Device (D2D) interactions, and the unique constraints of resource-limited IIoT settings. To this end, we intend to leverage a suite of advanced, yet lightweight, cryptographic methodologies. Our overarching objective is to cultivate a suite of security protocols that are not only robust but also adaptable to the diverse and evolving demands of the IIoT landscape.

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

Funding: This research was funded by the Gansu University of Political Science and Law Scientific Research Innovation Project 2023, grant number GZF2023XZD11.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding authors.

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

References

1. Xia, Y.; Qi, R.; Ji, S. Research on Lightweight Key Exchange Protocol based on PUFs in Industrial Internet of Things. *Comput. Appl. Softw.* **2022**, *39*, 316–321.
2. Sun, D.-Z.; Gao, Y.-N.; Tian, Y. On the Security of a PUF-Based Authentication and Key Exchange Protocol for IoT Devices. *Sensors* **2023**, *23*, 6559. [CrossRef] [PubMed]
3. Xu, H. Research on Three-Factor Anonymous Authentication and Key Negotiation Scheme Based on Biological Fuzzy Extraction Technology in Industrial Internet of Things. Master's Thesis, Central China Normal University, Wuhan, China, 2023. [CrossRef]
4. Oza, P.; Kamdar, D. A review on security approaches of MQTT protocol with respect to Internet of Things. *IJRAR* **2020**, *7*, 1–11.
5. Tian, S.; Vassilakis, V.G. On the Efficiency of a Lightweight Authentication and Privacy Preservation Scheme for MQTT. *Electronics* **2023**, *12*, 3085. [CrossRef]
6. Roldán-Gómez, J.; Carrillo-Mondéjar, J.; Castelo Gómez, J.M.; Ruiz-Villafranca, S. Security Analysis of the MQTT-SN Protocol for the Internet of Things. *Appl. Sci.* **2022**, *12*, 10991. [CrossRef]
7. Park, C.S.; Nam, H.M. Security architecture and protocols for secure MQTT-SN. *IEEE Access* **2020**, *8*, 226422–226436. [CrossRef]
8. Chen, Z.; Kong, D.; Yin, A.; Chen, Z.-F.; Zhang, P.-Y. SRAM-PUF pre-selection algorithm based on data residual time. *Acta Electron. Sin.* **2024**, *52*, 1478–1487.
9. Díaz, J.P.; Almenares, F. Integrating an optimised PUF-based authentication scheme in OSCORE. *Ad Hoc Netw.* **2023**, *140*, 103038. [CrossRef]
10. Yu, Z.Z.; Hong, H.W.; Xu, G.; Zhu, D.Q. Data Encryption Transmission Algorithm Based on MQTT. *Comput. Syst. Appl.* **2019**, *28*, 178–182. (In Chinese)
11. Zheng, Y.; Liu, W.; Gu, C.; Chang, C.-H. PUF-Based Mutual Authentication and Key Exchange Protocol for Peer-to-Peer IoT Applications. *IEEE Trans. Dependable Secur. Comput.* **2023**, *20*, 3299–3316. [CrossRef]
12. Chao, X.B.; Guo, F.; Wu, C.K. MQTT-SE Algorithm for Data Encryption Transmission. *Comput. Syst. Appl.* **2022**, *31*, 169–177. (In Chinese)
13. Wang, L.; Li, E.; Ji, Y.; Li, X. PUF-based Anti-physical Cloning RFID Security Authentication Protocol. *Netinfo Secur.* **2020**, *20*, 89–97.
14. He, Z.; Li, H.; Wan, M.; Wu, T. Authentication and session key exchange protocol based on Physical Uncolonable Function. *Comput. Eng. Appl.* **2018**, *54*, 17–21.
15. Xu, X.; Ou, Y.; Ling, J.; Jiang, X.; Wang, S. Lightweight RFID security Authentication Protocol based on PUF. *Comput. Appl. Softw.* **2014**, *31*, 302–306. (In Chinese)
16. Wang, Z.; Guo, Y.; Li, S.; Hou, S.; Deng, D. Design of efficient anonymous identity authentication protocol for lightweight IoT devices. *J. Commun.* **2022**, *43*, 49–61.
17. Qureshi, M.A.; Munir, A. PUF-RAKE: A PUF-based robust and lightweight authentication and key establishment protocol. *IEEE Trans. Dependable Secur. Comput.* **2021**, *19*, 2457–2475. [CrossRef]
18. Bian, W.; Gope, P.; Cheng, Y.; Li, Q. Bio-AKA: An efficient fingerprint based two factor user authentication and key agreement scheme. *Future Gener. Comput. Syst.* **2020**, *109*, 45–55. [CrossRef]
19. Huang, K.; Liu, Y.; Yin, X. An Ultra-Lightweight RFID Tag Ownership Transfer Protocol Based on PUF. *Acta Cryptologica Sin.* **2020**, *7*, 115–133.
20. Baranauskas, E.; Toldinas, J.; Lozinskis, B. Evaluation of the impact on energy consumption of MQTT protocol over TLS. In Proceedings of the CEUR Workshop Proceedings: IVUS 2019 International Conference on Information Technologies: Proceedings of the International Conference on Information Technologies (CEUR-WS 2019), Kaunas, Lithuania, 25 April 2019; Volume 2470, pp. 56–60.
21. Sochor, H.; Ferrarotti, F.; Ramler, R. Exploiting MQTT-SN for Distributed Reflection Denial-of-Service Attacks. In Proceedings of the International Conference on Database and Expert Systems Applications, Bratislava, Slovakia, 14–17 September 2020; Springer International Publishing: Cham, Switzerland, 2020; pp. 74–81.
22. Palmese, F.; Redondi AE, C.; Cesana, M. Adaptive quality of service control for mqtt-sn. *Sensors* **2022**, *22*, 8852. [CrossRef]
23. Krentz, K.F.; Rafiee, H.; Meinel, C. 6LoWPAN security: Adding compromise resilience to the 802.15.4 security sublayer. In Proceedings of the International Workshop on Adaptive Security, Zurich, Switzerland, 8 September 2013; pp. 1–10.
24. Hennebert, C.; Dos Santos, J. Security protocols and privacy issues into 6LoWPAN stack: A synthesis. *IEEE Internet Things J.* **2014**, *1*, 384–398. [CrossRef]
25. Glissa, G.; Meddeb, A. 6LowPSec: An end-to-end security protocol for 6LoWPAN. *Ad Hoc Netw.* **2019**, *82*, 100–112. [CrossRef]
26. Potrino, G.; De Rango, F.; Santamaria, A.F. Modeling and evaluation of a new IoT security system for mitigating DoS attacks to the MQTT broker. In Proceedings of the 2019 IEEE Wireless Communications and Networking Conference (WCNC), Marrakech, Morocco, 15–19 April 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6.
27. Sadio, O.; Ngom, I.; Lishou, C. Lightweight security scheme for mqtt/mqtt-sn protocol. In Proceedings of the 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS), Granada, Spain, 22–25 October 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 119–123.
28. Chatterjee, U.; Chakraborty, R.S.; Mukhopadhyay, D. A PUF-based secure communication protocol for IoT. *ACM Trans. Embed. Comput. Syst. (TECS)* **2017**, *16*, 1–25. [CrossRef]

29. Halak, B.; Zwolinski, M.; Mispan, M.S. Overview of PUF-based hardware security solutions for the Internet of Things. In Proceedings of the 2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS), Abu Dhabi, United Arab Emirates, 16–19 October 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–4.
30. Tang, Z.; Li, X. Formal description of Dolev-Yao Attacker Model. *Comput. Eng. Sci.* **2010**, *32*, 36–38.
31. Gong, X. Industrial IoT AKE Protocol Design and Safety Formalized Validation Method Research. Ph.D. Thesis, Lanzhou University of Technology, Lanzhou, China, 2023.