electronics

Article

PEASE: A PUF-Based Efficient Authentication and Session Establishment Protocol for Machine-to-Machine Communication in Industrial IoT

Xiang Gong [1], Tao Feng [1,*] and Maher Albettar [2]

[1] School of Computer and Communication, Lanzhou University of Technology, Lanzhou 730050, China
[2] Department of Electrical and Computer Engineering, Concordia University Montreal, Montréal, QC H3G 1M8, Canada
* Correspondence: fengt@lut.edu.cn

Abstract: Machine-to-machine (M2M) communication is one of the critical technologies of the industrial Internet of Things (IoT), which consists of sensors, actuators at the edge, and servers. In order to solve the security and availability problems regarding communication between edge devices with constrained resources and servers in M2M communication, in this study we proposed an authentication and session establishment protocol based on physical unclonable functions (PUFs). The scheme does not require clock synchronization among the devices, and it circumvents the situation where the authentication phase has to use a high computational overhead fuzzy extractor due to PUF noise. The protocol contains two message interactions, which provide strong security and availability while being lightweight. The security modelling is based on CPN Tools, which verifies security attributes and attack resistance in the authentication phase. After considering the design of the fuzzy extractor and scalability, the proposed scheme significantly reduces the computational overhead by more than 93.83% in the authentication phase compared with other schemes using PUFs. Meanwhile, under the guarantee of availability, the communication overhead is maintained at a balanced and reasonable level, at least 19.67% lower than the solution using XOR, hashing, or an elliptic curve.

Keywords: M2M; authentication; security protocol; Industrial Internet of Things; CPN Tools; PUF

check for updates

Citation: Gong, X.; Feng, T.; Albettar, M. PEASE: A PUF-Based Efficient Authentication and Session Establishment Protocol for Machine-to-Machine Communication in Industrial IoT. *Electronics* **2022**, *11*, 3920. https://doi.org/10.3390/electronics11233920

Academic Editor: Adão Silva

Received: 27 October 2022
Accepted: 25 November 2022
Published: 27 November 2022

1. Introduction

The Industrial Internet of Things (IIoT) is a subset of the Internet of Things (IoT), an intelligent and highly interconnected network of various industrial components [1] that achieve higher productivity and lower operating costs through real-time monitoring, automatic management, and control of industrial processes, assets, and operating hours; applications on production lines are addressed, where machines can communicate with each other. They can monitor each other and distribute workload, detect wear and tear, prevent failures, guarantee continuous production, and provide real-time production data [2]. Originally, the IIoT was proposed to differentiate between industrial and consumer-facing applications, aiming to combine manufacturing and the IoT. Compared with the general IoT, the IIoT needs higher stability, availability, and security [3]. However, Gartner reports that 20% of IoT enterprises have suffered at least one cyber-attack [4]. In recent years, the IIoT has seen an increasing number of applications in smart cities, oil and gas refineries, manufacturing and agriculture, and so on. It poses a considerable challenge to security and privacy.

Sensors are the source of perceiving all "things" in the IIoT, located at the entire network architecture's bottom (sensing layer), and connected to the intelligent gateway of edge measurement to transmit data to the cloud computing network with powerful processing capabilities [1]; sensors construct a ubiquitous architecture in the current IoT. In addition, intelligent edge measurement devices usually instruct the actuators to perform target actions, such as opening/closing switches, adjusting valves, and so on; it creates a closed loop between perception and control [5]. Intelligent sensors and actuators all need to communicate with their connected industrial PCs (IPCs), servers, or intelligent gateways, which is the generalized machine-to-machine (M2M) communication [6]. The network schematic of M2M communication is a use case of the industrial Internet unattended scenario; adopting low-cost resource-constrained devices facilitates ubiquitous monitoring of devices, but it also brings security and privacy challenges.

M2M communication is a crucial technology for future IIoT applications, which always attracts the attention of many malicious users [7]. As the ability of attackers continues to improve, various new attacks emerge one after another; some IIoT networks can even directly control the physical behavior of intelligent devices, and some trivial damage behaviors may also cause massive disasters [8]. Therefore, the M2M communication environment is also one of the leading security battlefields. In addition to installing firewalls, intrusion detection (IDS), and other macro strategies, the security protection of communication protocols is also essential. One of the effective ways to protect data is to transmit it in encrypted form, which requires a feasible authentication and key exchange protocol [9]. Through the practical and feasible protocol design by cryptography, the sensing layer nodes and the upper-layer devices authenticate each other and establish a secure session, fundamentally protecting the security of M2M, which indicates that the M2M authentication and session establishment process is crucial.

Moreover, considering the cost and volume constraints of the devices, low-cost, lightweight devices at the edge-side nodes are always preferred [10,11]. It means that in the framework of M2M, the lightweight design of the sensing layer is one of the primary considerations when considering the security mechanism. However, in the current standard implementation scheme, IT Security protocols are often used in the IoT, such as TLS (Transport Layer Security) and DTLS (Datagram-TLS) [8]. These protocols are well-known to play an essential role in Internet security; but due to the extremely high computational and communication overhead, these protocols have poor adaptability in the limited environment of the IoT [12]; even copying Internet security protocols risks in IT networks to the IoT [13]. There are almost no standard lightweight security protocols specifically designed for industrial environments. Therefore, the design of lightweight IoT protocols has become a prevalent issue; many researchers are trying to improve existing or design new IoT security protocols to break this status quo.

Physical unclonable functions (PUFs) [14] are a kind of lightweight and low-cost hardware cryptographic primitive that exploits the unique physical properties of the device and is known as hardware fingerprinting technology; it is characterized by uniqueness and unpredictability, and has recently been widely used in secure key storage and agreement schemes of the IIoT [15,16]. PUFs have become an effective means of key storage in security protection equipment.

PUFs provide a robust authentication and cryptographic method between devices. Unfortunately, the PUFs commonly used today are noisy in reality; the response of the PUFs will be affected by the temperature, voltage fluctuation, chip aging, and other factors; the same input will produce slightly different outputs, but for cryptography, it must rely on precisely reproducible keys. Therefore, some error correction techniques should be implemented in hardware to obtain a reliable PUF response while keeping the light overhead; a fuzzy extractor [17] is often used in IoT devices to process the results, compute the results computed by PUF, and consistent outputs are obtained each time. However, the high computational costs brought by the fuzzy extractor have to be considered in the constrained environment.

Based on the above, we propose a PUF-based efficient authentication and session establishment protocol (PEASE). The authentication phase of the protocol is based on Hash, XOR, and PUF operations; it has the characteristics of low computational and communication overhead; realizes mutual authentication, session key agreement, and device identity confidentiality; it can resist many known attacks, such as replay, man-in-the-middle (MitM), key compromised impersonation (KCI) and desynchronization and so on (see Section 6.1 for details). CPN Tools is used to model and verify its security based on formal analysis. Overall, the contributions of this paper are as follows:

- Explored the defects in related works and researched the reasons and solutions.
- Proposed a lightweight and secure IIoT–M2M anonymous mutual authentication and session establishment protocol.
- Applied the PUF to the M2M scenario and reduced the overhead fuzzy extractors during the authentication phase.
- Demonstrated the simplified CPN Tools security protocol formal analysis method.
- Achieved a comparison with related works regarding of computational and communication costs.

The remaining sections are organized: In Section 2, the relevant existing works are reviewed. The network architecture, design goal, and attack model are introduced in Section 3. Section 4 introduces the proposed scheme's specific design; Section 4 models and analyzes the protocol and the attacker. Section 5 presents a security analysis of the protocol. Section 7 compares PEASE protocol with similar protocols regarding security and performance. In the end, a conclusion of the paper is given in Section 8.

2. Related Works

Over the past few years, some interesting authentication protocols for M2M scenarios in the IIoT have been proposed. However, some of the works still have security problems or design flaws.

Esfahani et al. [18] have proposed a lightweight authentication mechanism for M2M communication in the IIoT environment based only on Hash and XOR operations. Their mechanism has low computational, communication, and storage overhead and was claimed to be resistant to various attacks; then, several weaknesses have been identified by [19,20]: the lack of resistance to tampering, impersonation, and replay attacks. Later, Lara et al. [21], inspired by [18], proposed a lightweight authentication and key distribution (LAKD) scheme that uses only Hash, XOR, and addition and subtraction construction; however, this protocol [21] does not explicitly state how to keep synchronization between the sensor and the gateway; so it cannot resist desynchronization attacks.

Sadhukhan et al. [22] found that the scheme of Zhang et al. [23] has the problem of massive overhead and user impersonation attacks, and proposed a lightweight smart grid communication authentication solution based on the elliptic curve [22]. Their scheme is said to outperform the schemes of [23] concerning security strength and computational costs. However, due to the use of the CA certificate (which is sent when requested), the scheme's computational and communication overhead is still high, which is unsuitable for the constrained IoT environment. In addition, the content of the CA certificate is unchanged; the malicious user who intercepts the protocol request can use it to link to any corresponding device, so the protocol does not provide unlinkability.

Alzahrani et al. [24] proved that the scheme of Noureddine et al. [25] does not have anonymity and does not provide security against known temporary information, and they proposed an authentication solution for privacy protection in the context of IoT [24]. The protocol uses PUF to encrypt the transmitted data, trying to resist the physical capture attack on the device. The authors claim that their protocol provides perfect forward security and many other security attributes; on the other hand, their work was later proved by Chen et al. [26] to be incapable of defending against insider privilege attacks, device capture attacks, temporary information leakage attacks, and other known attacks. Chen et al. proposed an enhanced AKA protocol for IoT environment [26] and claimed to satisfy the security attributes and protocol resistance unsatisfied in [24]; however, the scheme [26] does not satisfy unlinkability, as the parameter SID is sent in each request, and the PUF inputs are the same every time, which may cause multiple outputs of the function due to the noise, resulting in different session keys computed by the two parties.

Panda et al. [27] proposed a secure and lightweight authentication protocol (SLAP). Their scheme is claimed to be the first protocol that satisfies all the security attributes required for M2M communication. It uses only symmetric cryptographic operations, XOR, and hashing operations. However, the AS (authentication server) plays the trusted third party (TTP) role and knows all the pre-shared information of both parties; as described in [28], honest and curious TTP is a risk. Moreover, the protocol does not provide resistance against KCI attacks since its high dependence on the pre-shared key (PSK), in the case of key compromise, malicious users can obtain all sensors' IDs and impersonate controllers. For the same reason, the scheme cannot support perfect forward secrecy.

Recently, Alshareeda et al. [29] proposed a chaotic map-based conditional privacy-preserving authentication protocol called CM-CPPA. This scheme stores the private key of trusted authority (TA) on the tamper-proof device (TPD) of all vehicles, which makes them vulnerable to side-channel attacks; a more powerful attacker model should have been adopted to guide the design. Hence, it does not have forward (backward) secrecy and resistance to attacks such as KCI, privileged-insider, and so on. Later, the authors realized the problem and adopted an improved threat model to enhance the channel protection method of the scheme so that the improved scheme [30] has more robust security. However, since the scheme adopts a calculation based on Chebyshev polynomials and aims at the vehicular network scenario, its application in the constrained M2M communication environment described in this paper is still limited.

In addition, according to the description of protocol scalability in [28] (Feature P1), we believe that the schemes given in the design of [18,21,27] lack scalability. That means sensors and gateways in M2M have a many-to-one relationship; the gateway does not have a direct identity index due to using pseudonyms after receiving the request, and cannot determine which sensor the message is sent from, so it cannot find the corresponding calculation parameters; the subsequent calculation must exhaustively traverse the entire registered client table and compute the verification one by one. This defect reduces the availability of the scheme given in [18,21,27], and its performance analysis becomes meaningless due to the missing cost of exhaustive searches.

Furthermore, a significant problem in distributed systems is clock synchronization [31]; regardless of whether there is a global clock server in the network, the device must have an additional secure channel to maintain clock synchronization, which increases the computational and communication overheads of the device; it provides malicious users with another attack concept, which modifies the device time by trying to attack the time synchronization protocol; once modified, the device will be automatically denied service according to the above protocol rules [21,22,26,27]. On the contrary, if there is no global synchronous clock server and no additional protocol, even without malicious users, equipment in operation after a period between devices may have a slight clock error. So, a device based on protocols that need to be clock-synchronized will be disabled.

Nevertheless, due to changes in temperature, power flicker, and so on, PUF for constrained IoT environments (e.g., SRAM-PUF) may be affected by noise and output jitter; in the design of PUF-based applications, if it is necessary to obtain accurate output from the same input multiple times, they should adopt a fuzzy extractor to ensure its uniqueness. The fuzzy extractor consists of the generation function and the recovery function; according to the energy efficiency data [32], the overhead imposed by fuzzy extractors on constrained systems is not to be underestimated. The work of [24,26] uses a fixed PUF input in each round of the authentication phase, but the authors did not consider the expenditure calculation of the fuzzy extractor, which causes much lower availability of these schemes in reality. Therefore, we believe that to adapt to the IoT-constrained environment if PUFs are adopted, the challenges inputted would not be the same each time; otherwise, the fuzzy extractor has to be considered to unify the output of the PUF.

3. Preliminaries

It can be seen from the related works that each of them has several unsolved problems; the M2M security protocol for IIoT edge measurement must provide comprehensive security while ensuring it is lightweight. In this section, the preliminaries of our work are introduced in the following subsections.

3.1. Network Architecture

Figure 1 shows the architecture focused on in our work for IIoT M2M communication. In this paper, sensors, actuators, and other edge nodes are uniformly referred to as Devices. The upper-layer machines that need to respond directly to the Device, such as intelligent gateways, IPCs, or servers, are collectively referred to as Supervisors; when authentication is prescribed, the Device$_i$ (the ith device) initiates the message, and the Supervisor responds; the relationship between Devices and Supervisor is many-to-one.

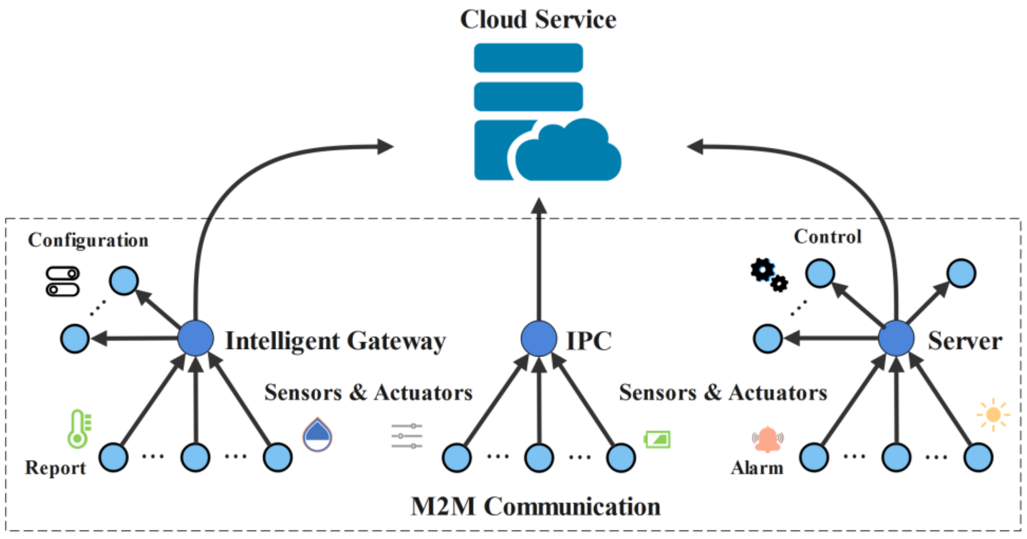

Figure 1. Illustration of M2M communication in IIoT.

Meanwhile, the architecture and terms of this paper are explained as follows.

- Devices are resource-constrained, while the Supervisor is relatively unconstrained.
- Device$_i$ initiates the protocol request directly to the Supervisor, Device$_i$ does not accept any request, and the reception of the response is realized within a specific reception window.
- Device$_i$ and Supervisor have their master key and short-term keys derived from the master key, respectively. The master key and other keys are stored in the trusted platform module TPM of the device and supervisor; the short-term key can be updated periodically, and it is generated by PUF or other methods each time, which prevents the cloning of the key. However, this paper does not discuss the TPM and the specific derivation method of the initial key.
- A fuzzy extractor consists of generation ($FE.Gen$) and reproduction ($FE.Rec$) functions. The $FE.Gen$ takes the PUFs response R as input and outputs key K and helper data hd, i.e.,$(K, hd) = FE.Gen(R)$. The $FE.Rec$ takes another sample with noised R' and the helper data hd as input and outputs the key K', i.e., $K' = FE.Gen(R', hd)$; $K' = K$ in the case that the Hamming distance is sufficiently close.
- The computing capabilities of devices include random number generation, hash calculation, XOR calculation, and PUF calculation.
- Supervisor includes capabilities for random number generation, hash calculation, XOR calculation, fuzzy extractor generation, and reproduction.
- The storage of a Device is divided into random access memory (RAM) and non-volatile memory (NVM, i.e., EEPROM).

3.2. Design Goal

Our work should achieve the following objectives:

- Efficient authentication: To satisfy the device's constraints environment, the solution must avoid high-cost computing solutions. It should be done with only PUF, hash, and XOR operations.
- Security: The scheme should provide reasonable security, including availability, confidentiality, integrity, forward security, and so on.
- Pseudonym Identity and unlinkable: A potential attacker in a public channel cannot obtain a device's true identity and has no way to link to a specific source by tracing any parts in different messages.
- Scalability: Provides responders with the ability to index the requester's identity in its database, ensuring protocol availability.
- Eliminate PUFs noise: Due to the noise problem of PUF at present, the scheme of the fuzzy extractor should be added to the proposed scheme. However, the overhead the fuzzy extractor brings cannot be ignored either; it is necessary to consider how to avoid the high cost.
- Without clock synchronization: The scheme cannot require devices in the M2M environment to realize clock synchronization.
- Resistance to known attacks: The solution should have strong attack resist capabilities to provide security protection in the network of potentially powerful attackers and resistance to various known attacks, including KCI, internal authorization attacks, DoS attacks, and so on, as will be explained in the following subsection.

3.3. Attack Model

The Dolev–Yao (DY) model [33] creates a formal verification method based on symbolic analysis; it is used to find logical flaws that may be contained in security protocols. This method has been widely used in the past decades to find vulnerabilities effectively at the design stage; the attacker $\mathcal{A}$ described by the DY model can be summarized as having the following capabilities:

- $\mathcal{A}$ has complete control over the network, and can eavesdrop, block, and intercept any message on the network; $\mathcal{A}$ has recorded all messages previously sent between honest entities.
- $\mathcal{A}$ is able to send and resend messages at will and combine and decompose messages.
- $\mathcal{A}$ can perform any encryption operation specified in the protocol and decrypt the encrypted message when the decryption key is known.
- $\mathcal{A}$ is a legal system member, meaning $\mathcal{A}$ is registered with the system and has all the security parameters.

Furthermore, with the increasing ability of attackers, the "compromised communication parties" should also be considered [34]. Similar to the extended Canetti–Krawczyk (eCK) security model, we consider the case where an attacker obtains various system secrets separately to make the protocol as secure as possible, one of the following capabilities against the authentication phase of PEASE protocol should be added to $\mathcal{A}$ when different attacks are verified.

- When $\mathcal{A}$ breaks the random number generator, $\mathcal{A}$ can directly obtain the operands r_1, r_2, C_i^{new} and n used for the current round key generation between honest entities.
- When $\mathcal{A}$ exposes the non-volatile memory by capturing the Device$_i$, $\mathcal{A}$ can directly obtain AID_i, C_i and S_i. In addition, if the Supervisor is compromised, $\mathcal{A}$ can get all the values in the Supervisor database;
- When verifying the forward secrecy or KCI, $\mathcal{A}$ directly obtains the keys DK_i and DK_S of the Device$_i$ and Supervisor, as well as the session key SK of the current and previous rounds. In this case, $\mathcal{A}$ obtains the Ri value that will be used in the next handshake by analyzing the RAM of the Device$_i$.

4. Proposed Scheme

This section presents the proposed PEASE protocol. Table 1 tabulates the notations and definitions used in these phases of our work.

Table 1. Notations and Descriptions.

Notations	Description
AID_i	The pseudonym of Device_i
DK_i, DK_S	Device_i and Supervisor's respective keys
C_i	The PUF challenges of Device_i
R_i	The PUF response of Device_i
S_i	Shared secret between Device_i and Supervisor
T_i	The current timestamp obtained by Supervisor
iV_i	The shared secret used for initialization phase
ΔT_1, ΔT_2	The respective minimum time interval for initialization and authentication phases
T_i^{Re}, T_i^{Auth}	Device_i initialization and authentication timestamps
IR_i, ER_S	The respective encrypted storage variable R_i of Device_i and Supervisor
r_1, r_2, n	Random nonce
Hr_i^{old}	The hash string of the nonce r_1 used in previous round
$PUF(.)$	Physical unclonable functions
$FE.Gen(.)$, $FE.Rec(.)$	Generating and reproducing functions of the fuzzy extractor
$h(.)$	One-way hash function
X^{new}	The superscript new for a parameter X indicates that the parameter will be used in the next round of computation
X^{old}	The superscript old of a parameter X indicates that the parameter has been used in the previous round
X'	The $'$ after the parameter X indicates that the parameter is derived from the value received from the network
$X*$	The * after the parameter X indicates that the parameter was taken from the database
$\perp$	Null value
$\|\|$	The concatenation operation
$\oplus$	The exclusive-OR operation

4.1. PEASE Protocol

The protocol has four phases: (1) device registration phase, (2) device initialization phase, (3) authentication phase, and (4) device logout phase.

4.1.1. Registration

This phase is carried out by the Devices and the Supervisor over a secure channel. In this phase, the Devices and the Supervisor exchange the secret; if there is no exception, this phase is executed only once in the whole life cycle of the device. Figure 2 illustrates the steps of the device registration phase, and the steps are described as follows:

Step 1: Initially, Device_i randomly generates the first pseudonym AID_i, challenge C_i, and secret S_i. Compute $R_i = PUF(C_i)$, $A_1 = (R_i||S_i) \oplus DK_i$. Send AID_i, A_1 to Supervisor.

Step 2: Supervisor sends $A_2 = A_1{}' \oplus DK_s$ to Device_i.

Step 3: Device_i sends $A_3 = A_2{}' \oplus DK_i$ to Supervisor.

Step 4: Supervisor generates a random iV_i, gets the current timestamp T_i, computes $(R_i||S_i) = A_3 \oplus DK_S$. Send $A_4 = h(R_i) \oplus h(S_i) \oplus iV_i$ to Device_i. Create an entry for Device_i in the login device table format $< AID_i,\ AID_i^{old},\ ER_i,\ ER_i^{old},\ S_i,\ S_i^{old},\ HV_i,\ Hr_i^{old},\ T_i^{Re},\ T_i^{Auth} >$ in the database with the initial values of $< AID_i,\ \perp,\ (R_i \oplus DK_S),\ \perp,\ S_i,\ \perp,\ (iV_i \oplus DK_S),\ \perp,\ T_i,\ T_i >$.

Step 5: After receiving, Device_i computes $iV_i{}' = h(R_i) \oplus h(S_i) \oplus A_4{}'$, and stores AID_i, C_i, S_i and $EV_i = iV_i{}' \oplus DK_i$ in NVM.

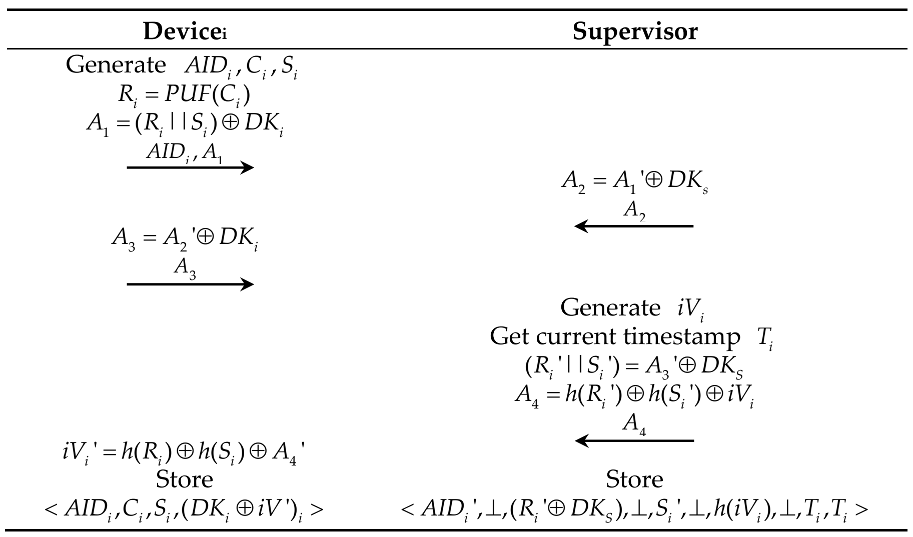
Device$_i$
Supervisor
Generate AID_i, C_i, S_i
$R_i = PUF(C_i)$
$A_1 = (R_i || S_i) \oplus DK_i$
AID_i, A_1
$A_2 = A_1' \oplus DK_s$
A_2
$A_3 = A_2' \oplus DK_i$
A_3
Generate iV_i
Get current timestamp T_i
$(R_i' || S_i') = A_3' \oplus DK_S$
$A_4 = h(R_i') \oplus h(S_i') \oplus iV_i$
A_4
$iV_i' = h(R_i) \oplus h(S_i) \oplus A_4'$
Store
$< AID_i, C_i, S_i, (DK_i \oplus iV')_i >$
Store
$< AID_i', \perp, (R_i' \oplus DK_S), \perp, S_i', \perp, h(iV_i), \perp, T_i, T_i >$

Figure 2. PEASE protocol steps in the registration phase (secure channel).

4.1.2. Initialization

When the Device$_i$ is turned on for the first time after the registration is completed; alternatively, when the Device$_i$ is restarted due to maintenance or battery replacement, the Device$_i$'s RAM will load the context for authentication and session establishment. In these cases, the steps shown in Figure 3 will be executed. This phase is carried out in a public channel, the detailed steps are as follows:

Step 1: The Device$_i$ computes $R_i = PUF(C_i), iV_i = EV_i \oplus DK_i$, $B_1 = h(h(S_i)||h(iV_i)) \oplus R_i$ and $B_2 = h(AID_i||S_i||h(iV_i)||R_i)$. Send initialization request message $\{B_1, B_2, AID_i\}$ to Supervisor.

Step 2: Next, upon receiving the initialization request message, the Supervisor first locates AID_i' in its database and subsequently reads and loads ER_i^*, S_i^*, HV_i, T_i (ER_i, S_i, HV_i, T_i when $AID_i' = AID_i$ or ER_i^{old}, S_i^{old}, HV_i, T_i when $AID_i' = AID_i^{old}$) into its RAM. Hereafter, the Supervisor obtains the current timestamp T_i and verifies $T_i - T_i^{Re} > \Delta T_1$. If the verification is successful, the Supervisor updates T_i^{Re} with T_i, computes $R_i' = B_1 \oplus h(h(S_i^*)||HV_i)$, and verifies $B_2' = h(AID_i||S_i^*||HV_i||R_i')$; if successful, it determines whether $R_i' = ER_i^* \oplus DK_S$ is true. If so, send $B_3 = h(R_i'||S_i^*||h(iV_i))$ to Devices, and the step processes are finished. If it is false, compute $(k_i', hd_i') = FE.Gen(R_i')$, $(k_i, hd_i) = FE.Gen(ER_i^* \oplus DK_S)$, $XR_i = FE.Rec(R_i', hd_i')$, and $YR_i = FE.Rec(ER_i^* \oplus DK_S, hd_i)$, then verify $XR_i = YR_i$. If successful, update ER_i^* with $R_i' \oplus DK_S$ and send $B_3 = h(R_i'||S_i^*||h(iV_i))$ to Devices.

Step 3: After receiving the Message B_3, the Device$_i$ verifies $B_3' = h(R_i||S_i||iV_i)$,and if successful, computes $IR_i = R_i^{new} \oplus DK_i$, then clears all the RAM's temp values.

4.1.3. Initialization

In this phase, the Device$_i$ and the Supervisor will communicate following the steps shown in Figure 4. At the end of this phase, Device$_i$ and Supervisor authenticate each other and share a key (SK) for encrypted communication in the session.

Device$_i$	**Supervisor**
$R_i = PUF(C_i)$	
$iV_i = EV_i \oplus DK_i$	
$B_1 = h(h(S_i) \| \| h(iV_i)) \oplus R_i$	
$B_2 = h(AID_i \| \| S_i \| \| h(iV_i) \| \| R_i)$	
$\xrightarrow{B_1, B_2, AID_i}$	
	Find ER_i^* (ER_i / ER_i^{old}),
	S_i^* (S_i / S_i^{old}), HV_i, T_i^{Re} with AID_i' $\boxed{CP1}$
	Get current timestamp T_i
	Check $T_i - T_i^{Re} > \Delta T_1$ $\boxed{CP2}$
	Update T_i^{Re} with T_i
	$R_i' = B_1 \oplus h(h(S_i^*) \| \| HV_i)$
	Check $B_2' \overset{?}{=} h(AID_i \| \| S_i^* \| \| HV_i \| \| R_i')$ $\boxed{CP3}$
	if $R_i' \neq ER_i^* \oplus DK_S$ {
	$(k_i', hd_i') = FE.Gen(R_i')$
	$(k_i, hd_i) = FE.Gen(ER_i^* \oplus DK_S)$
	$XR_i = FE.Rec(R_i', hd_i')$
	$YR_i = FE.Rec(ER_i^* \oplus DK_S, hd_i)$
	Check $XR_i \overset{?}{=} YR_i$ $\boxed{CP4}$
	Update ER_i^* with $R_i' \oplus DK_S$ }
	$B_3 = h(R_i' \| \| S_i^* \| \| h(iV_i))$
	$\xleftarrow{B_3}$
Check $B_3' \overset{?}{=} h(R_i \| \| S_i \| \| iV_i)$	
$IR_i = R_i \oplus DK_i$	

Figure 3. PEASE protocol steps in the initialization phase. (CP1-4 stand for checkpoints for attack detection; this will be described in Section 4.2).

Node	Gateway
Generate r_1, C_i^{new}	
$R_i^{new} = PUF(C_i^{new})$	
$R_i = IR_i \oplus DK_i$	
$D_1 = S_i \oplus R_i \oplus r_1$	
$D_2 = R_i^{new} \oplus h(r_1 \| \| S_i)$	
$D_3 = h(r_1 \| \| R_i^{new} \| \| R_i)$	
$\xrightarrow{AID_i, D_1, D_2, D_3}$	
	Find ER_i^* (ER_i / ER_i^{old}),
	S_i^* (S_i / S_i^{old}), r_i^{old}, T_i^{Auth} with AID_i' $\boxed{CP1}$
	Get current timestamp T_i
	Check $\Delta T_2 < T_i - T_i^{Auth}$ $\boxed{CP2}$
	$R_i = ER_i^* \oplus DK_S$
	$r_1' = D_1' \oplus S_i^* \oplus R_i$
	Check $h(r_1') \neq Hr_i^{old}$ $\boxed{CP3}$
	$R_i^{new} = D_2' \oplus h(r_1' \| \| S_i^*)$
	Check $D_3' = h(r_1' \| \| R_i^{new} \| \| R_i)$ $\boxed{CP4}$
	Check $R_i^{new} \neq R_i$
	Generate two nonces r_2, n
	$D_4 = h(S_i^* \| \| R_i \| \| r_1') \oplus r_2$
	$D_5 = n \oplus h(r_2 \| \| R_i^{new})$
	$D_6 = h(n \| \| S_i^* \| \| r_2)$
	$SK = h(R_i^{new} \| \| R_i \| \| r_1' \| \| r_2)$
	$(AID_i^{new}, S_i^{new}) = h(SK \| \| n \| \| S_i^*)$
	Update database
	$\xleftarrow{D_4, D_5, D_6}$
$r_2' = h(S_i \| \| R_i \| \| r_1) \oplus D_4$	
$n' = h(r_2' \| \| R_i^{new}) \oplus D_5$	
Check $D_6' \overset{?}{=} h(n' \| \| S_i \| \| r_2')$	
$SK = h(R_i^{new} \| \| R_i \| \| r_1 \| \| r_2')$	
$(AID_i^{new}, S_i^{new}) = h(SK \| \| n' \| \| S_i)$	
$IR_i = R_i^{new} \oplus DK_i$	
Update NVM	

Figure 4. PEASE protocol steps in the authentication phase (unsecure channel). (CP1-4 stand for checkpoints for attack detection; this will be described in Section 4.2).

This phase takes place over the public channel and has three steps:

Step 1: When the Device$_i$ intends to communicate with the Supervisor, the Device$_i$ generate two random numbers r_1, C_i^{new}, then computes $R_i^{new} = PUF(C_i^{new})$, $R_i = IR_i \oplus DK_i$, $D_1 = S_i \oplus R_i \oplus r_1$, $D_2 = R_i^{new} \oplus h(r_1||S_i)$, $D_3 = h(r_1||R_i^{new}||R_i)$. Send message $D_3 = h(r_1||R_i^{new}||R_i)$. $M_1 = AID_i, D_1, D_2, D_3$ as a request to the Supervisor.

Step 2: Next, upon receiving the request message M_1, the Supervisor first locates AID_i' in its database and subsequently reads and loads ER_i^*, S_i^*, HV_i, T_i (ER_i, S_i, HV_i, T_i when $AID_i' = AID_i$ or ER_i^{old}, S_i^{old}, HV_i, T_i when $AID_i\prime = AID_i^{old}$) into its RAM. Then, the Supervisor gets the current timestamp T_i and verifies $T_i - T_i^{Re} > \Delta T_1$. If the verification is successful, computes $R_i = ER_i^* \oplus DK_S$, $r_1' = D_1' \oplus S_i^* \oplus R_i$. verifies $h(r_1') \neq Hr_i^{old}$, if successful, next computes $R_i^{new} = D_2' \oplus h(r_1'||S_i^*)$, $D_3 = h(r_1'||R_i^{new}||R_i)$ and then verifies $D_3 = D_3'$, $R_i^{new} \neq R_i$. If successful, generate two random numbers r_2 and n, computes $D_4 = h(S_i^*||R_i||r_1') \oplus r_2$, $D_5 = n \oplus h(r_2||R_i^{new})$, $D_6 = h(n||S_i^*||r_2)$, $SK = h(R_i^{new}||R_i||r_1'||r_2)$ and $(AID_i^{new}, S_i^{new}) = h(SK||n||S_i^*)$, where AID_i^{new} and S_i^{new} are obtained by splitting the hash string equally. Finally, computes $ER_i^{new} = R_i^{new} \oplus EK$, update field $< AID_i,\ AID_i^{old},\ ER_i,\ ER_i^{old},\ S_i,\ S_i^{old},\ Hr_i^{old},\ T_i^{auth} >$ with corresponding values $< AID_i^{new},\ AID_i',\ ER_i^{new},\ ER_i^*,\ S_i^{new},\ S_i^*,\ h(r_1'),\ T_i >$ in database. Send $M_2 = D_4, D_5, D_6$ to Device$_i$.

Step 3: After receiving M_2 Device$_i$ computes $r_2' = h(S_i||R_i||r_1) \oplus D_4$ and $n' = h(r_2'||R_i^{new}) \oplus D_5$, then verifies $D_6' = h(n_2'||S_i||r_2')$, if successful, next computes $SK = h(R_i^{new}||R_i||r_1||r_2')$, $IR_i = R_i^{new} \oplus DK_i$ and $(AID_i^{new}, S_i^{new}) = h(SK||n'||S_i)$. Replace the corresponding values with AID_i^{new}, C_i^{new} and S_i^{new} in the NVM. At this point, the protocol authentication phase was completed, and both parties mutually authenticated and established a secure channel with SK as the subsequent symmetric encryption key.

4.1.4. Device Logout

During operation, the Devices are allowed to apply for logout from the Supervisor through a public channel, if the Device fails or other reasons require active logout, the Device$_i$ sends a unique format of M_1 to the Supervisor, which is $R_i^{new} = R_i$ in the M_1 message; the rest is the same as the Device$_i$ calculation in the authentication phases. When the Supervisor receives the request and determines it is a logout request, it records the event in the logout device table and deletes the corresponding data entry from the login device table.

4.2. Attack Detection

Verifications of the Supervisor in the steps described above are referred to as checkpoints in the following (represented as CP1-CP4 in the Figures 3 and 4); if any of the verifications fail, the Supervisor records the event and does not perform any more computations. In addition, it can provide the system with the capability of protocol-level attack detection. Next, we illustrate the situation when the verification of these checkpoints fails.

According to Table 2, the countermeasures can be set for specific exceptions at the implementation. All errors should be handled by the supervisor or alarmed to the upper layer for decision-making. Suppose multiple verification errors occur in a short period; in that case, measures such as temporarily blocking the channel can be taken to reduce the impact of attacks on the system.

Table 2. Checkpoint validation failure description.

Checkpoint	Initialization Phase	Authentication Phase
CP1	No record for $AID_i{}'$ was found. An unsynchronized device attempted the request. Impersonation and replay attacks are possible	No record for $AID_i{}'$ was found. An unsynchronized device attempted the request. Impersonation and replay attacks are possible
CP2	Attempt to initialize in a short period, the system may be under replay attack	If authentication is attempted in a short period of time, the system may be under replay attack
CP3	The message is incomplete or tampered with. May suffer from tampering, impersonation attacks	Duplicate messages are received and the system suffers from replay attack
CP4	The $R_i{}'$ is incorrect, and the device fails. It may be attacked by device capture and information is lost.	The message is incomplete or tampered with. May suffer from tampering, impersonation attacks.

5. Security Verification

We used CPN Tools to model the protocol and the attacker. State space was used to verify the security, improving and simplifying the work method [34]. The modeling was carried out top-down, but considering that too many layers may affect the overall readability, we no longer layered the same entity in the modeling. The model mainly focuses on the protocol's authentication and session establishment process, including a top-level model and three second-layer model subpages. The CPN Tools model files of this paper are found in the supplementary material, which contains all CPN models and the definition of color sets and functions. Therefore, the definitions of color sets and functions are not listed below.

The top-level model is shown in Figure 5.

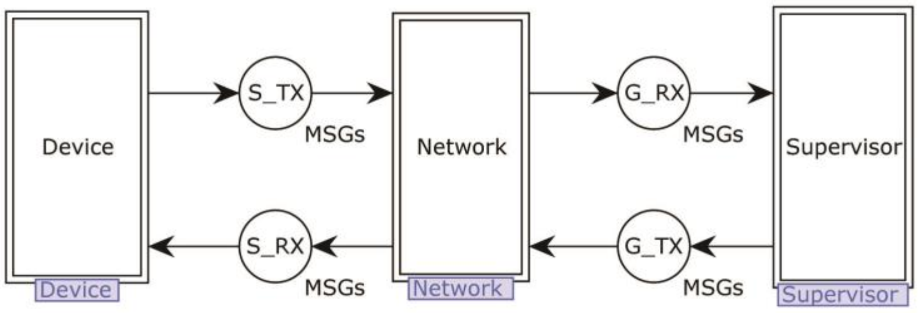

Figure 5. Top-level model of PEASE protocol.

The top-level model consists of three alternative transitions; according to the protocol rules, the Device sends requests, and the Supervisor processes and responds; the whole process consists of one interaction.

5.1. Baseline

The security verification baseline would first be established before modeling the attacker; that means we modeled the protocol in regular operation (without an attacker). The modeling calls the XOR and hash functions provided in SML/NJ library supported by CPN Tools, and all parameter color set types are STRING.

Figure 6 shows the Device model subpage.

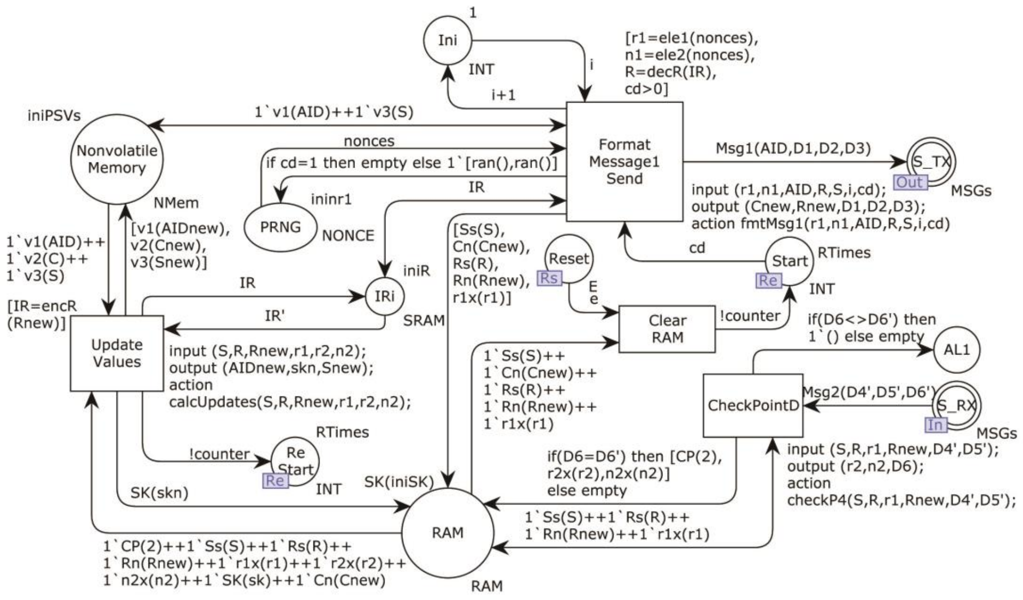

Figure 6. The Device model subpage.

The model includes the whole process of Device$_i$ sending requests and processing responses. The simulation execution of the model starts with the transition Format_Mssage1_Send, which takes the data from place Nonvolatlle_Memory and IRi. It combines the message according to the M_1 rule, then sends the message to the network by the interface place S_TX.

When receiving the response, the interface place S_RX obtains the M_2 token, the transition CheckPointD fires, checks whether D6 and D6′ are equal, and terminates the operation if not satisfied. If it is satisfied, the values of r2 and n2 are calculated. Then the transition Update_Values is triggered, which computes SK into memory, computes AID, Sinew, and updates the values in Nonvolatlle_Memory.

The Supervisor model page is shown in Figure 7.

The model simulates the process of the Supervisor receiving a request and sending a response according to the protocol rules. First, the interface place G_RX obtains the token; if the received AID′ has the corresponding AID value in the database, the transition Synchronized is triggered, and the corresponding Si and ER values in the place Database are extracted and put into place G_RAM as the AID and ER values of the current round; if the received AID′ has the corresponding AIDold value in the Database, the transition Non_Synchronized is triggered, and the corresponding Siold and ERiold values in the database are extracted and put into memory as the AID, S and ER values for the current round calculation. After that, transition Checkpoint3 is fired, which checks whether r1 calculated from received D1 is equal to the last received r1 value in the record, and aborts the run. If not, it calculates R and fires transition CheckPoint4, which checks whether D3′ and D3 are equal and whether R and Rnew are equal. The equality of D3 and D3′ is the condition for the protocol to continue to run, while the equality of R and Rnew is the token obtained by the place LO on behalf of the device logout. Then, if there is no interruption, the transition Calulate_update_values is fired. It calculates and updates the database according to the rules. Finally, the Fomat_Message2_and_Send transition is triggered, which calculates the values of D4, D5, D6 and formats them as Message2 to be sent to the interface place G_TX.

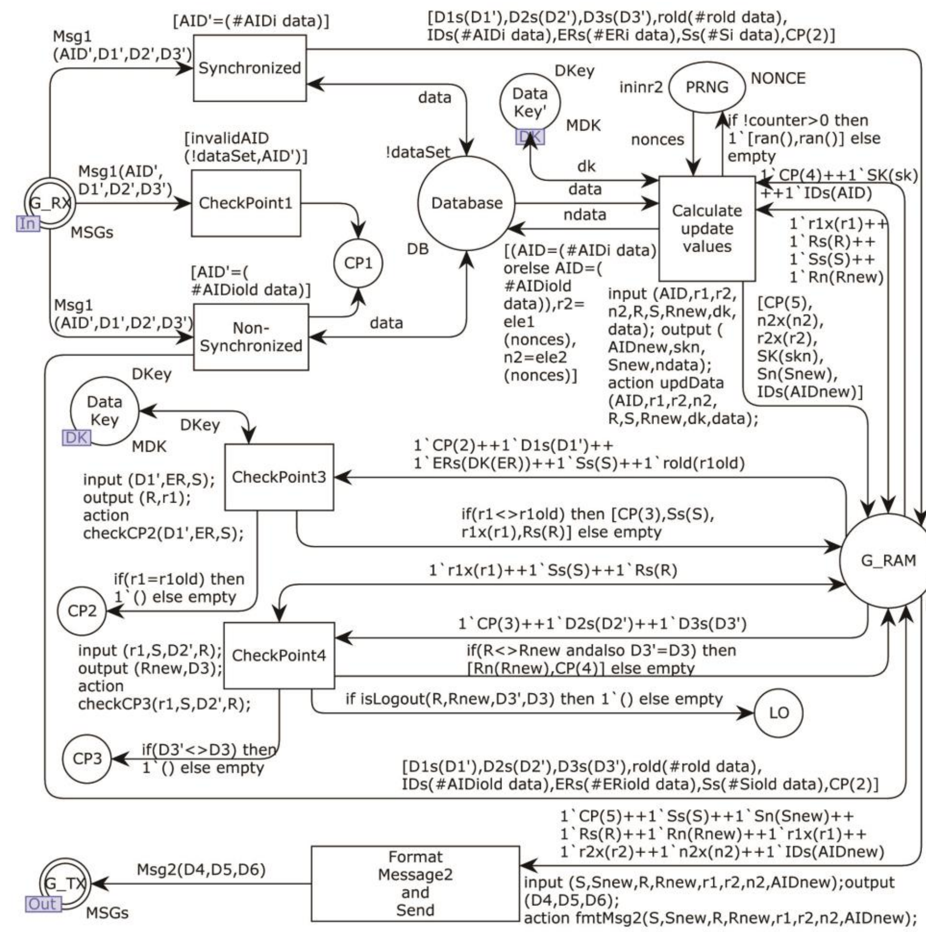

Figure 7. The Supervisor model subpage.

The model is executed step-by-step, strictly, according to the protocol rules, and at most, one transition is enabled at any time. Currently, there is no attacker, and the network is reliable. Through the state space computation, the number of nodes is 11, the number of arcs is 10, and the dead marking is 1, indicating that there is only one possible path. It lays the foundation for the following attacker modeling.

5.2. Attacker Modeling

According to the DY model, the attacker completely controls the network, so we describe the DY attacker at the network in the model. Since the attacker of DY is also one of the legitimate members, the attacker has a legitimate shared secret with Supervisor; the replace transition network in Figure 5 is the carrier of the attacker model.

Moreover, in this article, we adopt the method of incremental verification. First, validate the M_1 message; after receiving the M_1 sent by Supervisor, the attacker calculates and combines M_1, which conforms to the protocol rules at will according to the data owned by the knowledge base; then sends it to the Supervisor, sets a breakpoint when the attacker receives M_2, computes the state space, and searches for the legitimacy of all messages responded to by the Supervisor.

After that, the verification of M_2 is carried out. In this case, to reduce the amount of useless state space, the attacker's processing of M_1 will only send messages that the Supervisor can respond to in the previous verification. Instead of setting breakpoints, the model continues to search through the state space to find all messages that do not terminate due to the failure of Device CheckPoint4. Finally, it verifies and determines their validity.

In the two-step verification process, if an illegal M_1 is found to be responded to, the Supervisor entity rule in the protocol is proved to have a design flaw. If an illegal M_2 is responded to, it verifies that the design of the Device entity rule of the protocol is defective; when a design defect is found, the attacker's attack trace can be extracted by setting the observation place.

The model subpage of the attacker is shown in Figure 8.

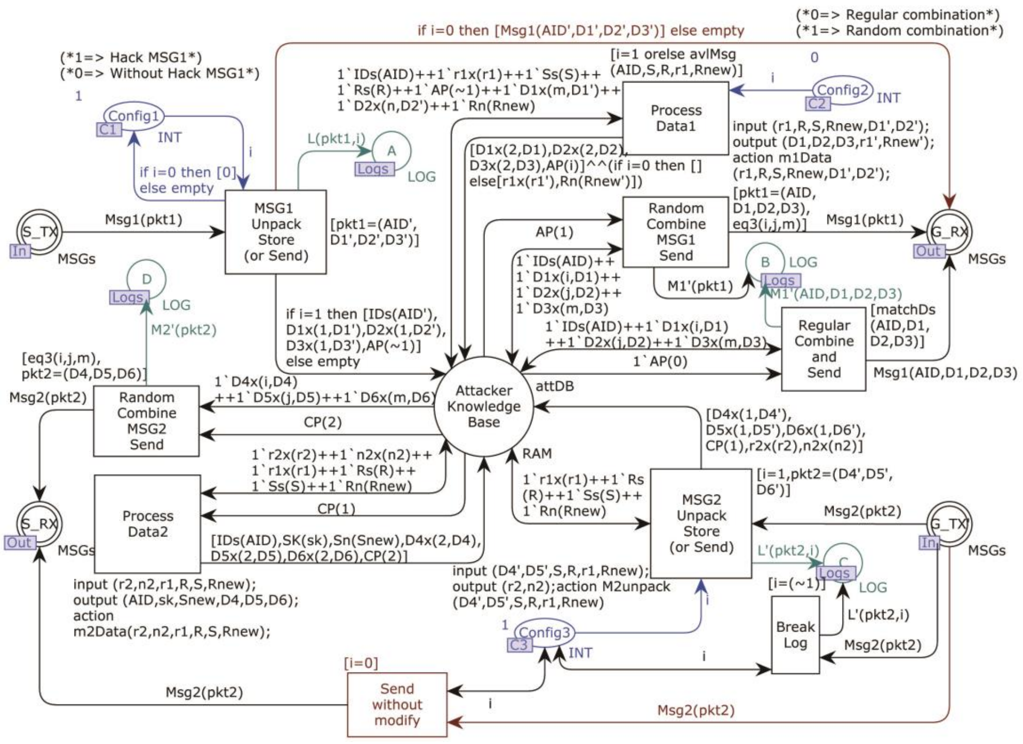

Figure 8. The Attacker model subpage.

There are two parts to the model; the upper and lower parts simulate the processing of M_1 and M_2, respectively; the two parts are connected by the place Attacker_Knowledge_Base in the middle; transitions are triggered in turn, and the transitions appearing in the model are explained in the following order.

For M_1, after receiving the token of the place S_TX, the attacker decides the route through Config1. See Table 3 for details; here, the Config value is 1, which triggers transition MSG1_Unpack_Store_(or Send) to split it and store it in the Attacker_Knowledge_Base. The role of the transition Process_Data1 is then to calculate the possible atomic data and store them in the knowledge base. This step simulates all possible computational attempts by the attacker and is indispensable. Next, the fusion place Config2 decides whether the Random_Combine_MSG1_Send transition should perform the randomly selected data combination M_1 and send it to G_RX. If no random combination is performed, only valid messages that the Supervisor can respond to are synthesized by the transition Regular_Combine_and_Send and sent to G_RX. It is to prevent the state space explosion caused by the random composition of M_1 during the verification of M_2.

For M_2, similar to the process of M_1, Config3 will decide the route after the token is obtained by the interface place G_TX. As detailed in Table 3, it is specified by the Config value of 1. Transition MSG2_Unpack_Store_(or Send) unpacks the received M_2 and stores it in the knowledge base. Transition Process_Data2 calculates the value in the existing knowledge base according to the protocol rules. It returns the result to the knowledge base. Next, the transition Random_Combine_MSG2_Send randomly extracts the required atomic messages from the knowledge base and calculates M_2 according to the protocol rules. Finally, it is sent to S_RX, the receiving interface of the Device.

Table 3. Description the values of places Config 1~3.

Place	Value	Process	Description
Config 1	0	*M*1 is sent without processed	Baseline test
	1	The subsequent steps are performed for *M*1	Enable the security verification of *M*1
Config 2	0	Only the legal *M*1 is assembled	The useless state space is reduced and used when verifying *M*2
	1	Take random values to combine *M*1	Verification of the security of *M*1
Config 3	0	*M*2 is sent without processed	Baseline test
	1	The subsequent steps are enabled for *M*2	Enable the security verification of *M*2
	~1	Set a breakpoint to stop the run	Block the execution of subsequent steps used when verifying *M*1.

5.3. State Space Analysis

According to the extended DY attacker described in Section 3.3, we divide the attacker's initial knowledge base into four cases. In the model, the switch is made by changing the global variable AttIniCf in the palette, corresponding to four values, which are:

0: It means that the knowledge of the DY attacker's initial place is not extended, and the attacker has all the capabilities of the DY attacker. The knowledge base includes M_1 and M_2 of the previous interaction between Device$_\text{i}$ and Supervisor and all atomic data after splitting. Since the attacker is a legitimate entity, the library also contains his own key, the shared secret, and all atomic data from the previous round with Supervisor.

1: Based on 0, the fresh random numbers used by Device$_\text{i}$ and Supervisor for current authentication and key agreement are added to the attacker's initial knowledge base.

2: Add the value of the current record in the NVM of Device$_\text{i}$ and all the data related to Device$_\text{i}$ stored in the Supervisor database to the attacker's initial knowledge base based on 0.

3: Based on 0, SK and the key of the last authentication round between Device$_\text{i}$ and Supervisor, and R stored securely between them, are added to the attacker's initial knowledge base.

The state space is calculated for each of the eight possible cases consisting of the two configuration items above. The written SML code searches the node satisfying the condition. The method used to judge the legitimacy of a message is to first calculate all possible legal M_1 and M_2 from the initial value and put them into two lists (legalMsg1 and legalMsg2), respectively. M_1 and M_2 extracted from the filtering results of the node search are compared with the two lists in turn; if there is a corresponding item, the message is considered legal; otherwise, the message is considered illegal. Eventually, we obtain the results shown in Table 4. (The calculated state space text files and the verification results screenshots of this paper were placed in the Supplementary Material).

From the results, we can see that PEASE protocol is secure against the extended DY attacker. The protocol has the ability to make both sides of the communication aware of attacks in the network and terminate all malicious messages to continue running. The attacker in the model made all possible attempts and created a considerable state space. However, the honest entity running according to the protocol rules did not appear to respond or process any messages tampered with by the attacker; instead, all the exceptions that appear are caught by the entity; it also provides an effective way for supervisors to find attacks in the network. In the following, the security attributes of the protocol are analyzed by combining the model-checking results.

Table 4. State space search results.

AttIniCf	Message [1]	Entity [2]	Node [3]	DM [4]	Filtered [5]	No. of IM [6]
0	$M1$	Supervisor	1651	384	3/40	0
	$M2$	Device_i	7747	3074	1/512	0
1	$M1$	Supervisor	2491	576	5/64	0
	$M2$	Device_i	38,123	15,194	1/2592	0
2	$M1$	Supervisor	2467	576	3/56	0
	$M2$	Device_i	17,363	6914	1/1152	0
3	$M1$	Supervisor	2467	576	3/56	0
	$M2$	Device_i	17,363	6914	1/1152	0

[1] The verified message; [2] The verified entity; [3] The number of nodes in state space; [4] The number of dead markings in state space; [5] The node message is responded to or processed (unique/total); [6] The number of illegal messages found.

6. Security and Features Analysis

In this section, we discuss the security of PEASE protocols, including security attributes and attack resistance, as well as some other features.

6.1. Security Analysis

6.1.1. Confidentiality

Typically, the parameters used in each run of PEASE protocol are updated. Shannon's theory proves that if at least one item in the XOR operation is random, then a simple XOR encryption guarantees security. For $\mathcal{A}$, the parameters of the intercepted messages change randomly every round. Therefore, the protocol effectively ensures the confidentiality of data transmission by simple XOR encryption and reduces the overheads.

6.1.2. Mutual Authentication

Before PEASE protocol key agreement, the Device and Supervisor authenticate with the shared secret S_i and the response R_i of PUF, respectively. $\mathcal{A}$ cannot obtain the secrets S_i and R_i, and cannot forge the identity. Therefore, $\mathcal{A}$ cannot authenticate.

6.1.3. Device Anonymity and Unlinkability

In the protocol, the real identity of the device is not used. Under normal circumstances, all the pseudonym and the transmitted data are changed in each round; $\mathcal{A}$ cannot link to a specific device by intercepting $M1$ or $M2$. Therefore, PEASE protocol provides strong anonymity and unlinkability.

6.1.4. Perfect Forward and Backward Secrecy

There is no connection between the session key SK generated by the protocol each time, and the key material in the protocol is based on random numbers; if $\mathcal{A}$ obtains the keys of the Device and Supervisor, the keys are only used to store data, not to encrypt the data transmission; so even if $\mathcal{A}$ eavesdropped and recorded multiple interaction messages, $\mathcal{A}$ could not calculate any previous or subsequent session keys. Therefore, PEASE protocol satisfies perfect forward security.

6.1.5. Resistance to the Replay Attack

Supervisor stores the last random value r_i^{old}. If $\mathcal{A}$ replays the last request directly, it will be rejected. However, if $\mathcal{A}$ replays the previous message, the $AID_i{}'$ in the message will not be accepted because the $AID_i{}'$ in the message no longer exists in the Supervisor database; $\mathcal{A}$ cannot perform replay attacks against PEASE protocol, nor can it impersonate through replay attacks.

6.1.6. Resistance to Man-in-the-Middle Attack

Because of the protocol's mutual authentication between the Device and Supervisor, it can resist the replay attack; $\mathcal{A}$ cannot impersonate any honest entity by tampering or replaying the intercepted message so that PEASE protocol can resist a MitM attack.

6.1.7. Resistance to the Key Compromise Impersonation Attack

$\mathcal{A}$ obtains the Device keys DK_i and DK_S and wants to impersonate either Device or Supervisor to authenticate the other. However, in PEASE protocol, the key is only used for the secure storage of parameters. The message encrypted with the key is not transmitted in the message sent through the insecure channel; only the data encrypted by XOR is transmitted in the interactive message. $\mathcal{A}$ cannot decrypt or forge any ciphertext containing information after getting the keys; PEASE protocol can resist KCI attacks.

6.1.8. Resistance to the Known Session-Specific Temporary Information (KSSTI) Attack

In the process of security model checking in Section 5, we added the random numbers r_1, r_2, n and C_i^{new} that Device$_\text{i}$ and Supervisor will use to $\mathcal{A}$'s initial knowledge base. However, the verification result shows that data loss has not caused the key material to leak. It is because the key material must include the parameter R_i^{new} calculated by $PUF(C_i^{new})$ and the encrypted saved R_i in addition to the random number participation, but $\mathcal{A}$ cannot obtain the secret parameter S_i and the key DK_S. Therefore, $\mathcal{A}$ cannot obtain R_i^{new} and R_i; PEASE protocol can resist the KSSTI attacks.

6.1.9. Resistance Device Capture Attacks

$\mathcal{A}$ captures a Device and obtains the data and keys in the NVM utilizing channel test analysis or other methods. However, $\mathcal{A}$ cannot obtain it due to the unclonable nature of the PUF, and cannot obtain r_2 and n. At the same time, the authentication between the devices has nothing to do with the key generation material, so $\mathcal{A}$ cannot get any other Device's information. Therefore, the protocol can resist device capture attacks.

6.1.10. Resistance to Verifier Loss Attacks

When $\mathcal{A}$ obtains the Supervisor's entire database, $\mathcal{A}$ still cannot get the latest R_i because it is secretly stored by XOR and encrypted using the short-term key. In each round of authentication, the R_i used by the two sides of the communication is generated by the Device$_\text{i}$ through the PUF through random numbers. Therefore, $\mathcal{A}$ cannot predict R_i^{new} in the case of having the verifier. Therefore, verifier loss does not affect the protocol.

6.1.11. Resistance to the Privileged Insider Attack

The model checking in Section 5 illustrates that the protocol is resistant to insider privilege attacks. Because our attacker is also a legitimate entity, $\mathcal{A}$ could get all the data of the NVM in the Device and all the data in the Supervisor database. However, the attacker still could not get the session key of the honest entity; the reason is that in a Device benefiting from the unclonable property of PUF, the corresponding PUF output R_i cannot be calculated even if the value of C_i is obtained. In Supervisor, the value of R_i is stored encrypted with the key DK_S. If an insider $\mathcal{A}$ intervenes in the protocol operation during the registration phase, $\mathcal{A}$ cannot obtain secrets because the information transmitted during the registration phase is XOR encrypted with the respective keys of Device$_\text{i}$ and Supervisor. $\mathcal{A}$ cannot obtain the shared secrets without the keys of both parties.

6.1.12. Resistance to the Desynchronization Attack

In PEASE protocol, the two sides include many shared secret parameters that change each time. Hence, the anti-desynchronization attack is one of the primary attacks the protocol considers preventing. By some means, $\mathcal{A}$ prevents the Device$_\text{i}$ from receiving the response M_2; in this case, the shared secret in Supervisor is updated, not Device$_\text{i}$. However, the Supervisor in PEASE protocol keeps the previous parameters of the current and previous rounds, both of which are acceptable when requested. Therefore, $\mathcal{A}$ cannot implement a desynchronization attack by obstructing M_2 reception. If the Device request uses a corrupted AID more than once, the Supervisor's CheckPoint1 will record this and alarm the system or higher layers about the decision.

6.1.13. Resistance to the DoS Attack

As mentioned in 3.1, because devices do not accept authentication requests, DoS attacks are mainly launched by $\mathcal{A}$ against Supervisors; the four checkpoints in the authentication phase block all abnormal requests; if the number of false captures of a checkpoint exceeds a specific value in a short period, the system can take corresponding measures, such as actively temporarily blocking the receiver port corresponding to the device to resist DoS attacks. Therefore, PEASE protocol can resist DoS attacks.

6.2. Additional Features

6.2.1. Real-Time Attack Awareness

Since checkpoints can detect every abnormal message, the system has the ability to know whether there is a potential attacker in the network.

6.2.2. No Clock Synchronization Requirement

The protocol has no transmission timestamp, so it does not need to consider establishing a global clock server and other additional equipment. There are no security risks or extra costs caused by time synchronization between communication entities.

6.2.3. Scalability

When the Supervisor receives a request, it does not need to exhaust the validation table and does not need to calculate each entry to find the appropriate parameter; the corresponding data item can be directly searched in the table using the synchronized pseudonym AID_i.

6.2.4. Synchronization Is Recovered on Restart

When the device is restarted due to equipment maintenance, battery replacement, and other reasons, the protocol's boot initialization operation will automatically resume the synchronization state with the Supervisor.

6.2.5. Avoid Fuzzy Extractors

PEASE protocol does not use a fuzzy extractor in the authentication phase, reducing the computational runtime cost caused by error recovery.

7. Security and Performance Comparison

In this section, the computational cost and communication cost of PEASE are analyzed, and the security and performance of PEASE are compared with similar related works [21,22,24,26,27].

7.1. Security Comparison

In terms of security and availability, the protocol provides security attributes and other features that candidate protocols do not provide. Table 5 shows the comparison.

Table 5. Security comparison.

Security Attribute	LAKD [21]	Sadhukhan et al. [22]	Alzahrani et al. [24]	Chen et al. [26]	SLAP [27]	PEASE
No clock synchronization	×	×	√	×	×	√
Perfect Forward and Secrecy	√	√	√	√	×	√
Anonymity and Unlinkability	√	×	√	×	√	√
Scalability	×	√	√	√	×	√
Resist KCI Attack	√	√	√	√	×	√
Resist Desynchronization Attack	×	√	√	√	√	√
Resist Privileged Insider Attack	√	√	×	√	√	√
Resist Device Capture Attack	√	√	×	√	√	√
Resist KSSTI Attack	√	√	×	√	√	√
Resist DoS Attack	√	×	√	√	√	√

7.2. Computation Overhead and Communication Overhead Comparison

In this comparison of computational overhead, we only consider various intensive computes and focus on the authentication and key establishment phase during the run-time of each protocol. Due to different constraint levels, we count the computational overhead of the Device and Supervisor separately. According to the experimental data of [32], we define various operation symbols and calculation times in Table 6.

Table 6. Operation cost symbol and execution time.

Operation	Notation	Device	Supervisor
Hash function	T_h	0.026 ms	0.011 ms
Point multiplication	T_{pm}	5.9 ms	2.6 ms
Symmetric encryption/decryption	$T^S_{e/d}$	0.079 ms	0.041 ms
Certificate verification	T^{cert}_{ver}	-	17.24 ms
Fuzzy extractor generation	T_{FEG}	1.67 ms	-
Fuzzy extractor reproduction	T_{FER}	-	2.85 ms
PUF	T_{PUF}	0.12 ms	-

The results in Table 7 show the computational overhead of each protocol. Among them, Sadhukhan's protocol [22] uses several times of elliptic curve operations and one certificate verification calculation; only XOR and hash computations are used in LAKD [21] and SLAP [27]. However, for the sake of fairness, the scheme of Chen [26] and Alzahrani [24] inevitably uses a fuzzy extractor because the same parameters need to be extracted by PUF in different rounds. In view of [24] and [26], we assume that the fuzzy generation calculation is carried out at the Device side and the fuzzy regeneration calculation is carried out at the Supervisor side.

Table 7. Comparison of computational overhead.

Title 1	Title 2	Title 3	Title 4
LAKD [21]	$8T_h \approx 0.208$ ms	$8T_h \approx 0.088$ ms	0.296 ms
Sadhukhan et al. [22]	$4T_{pm} + 3T_h + 1T^S_{e/d} \approx 23.757$ ms	$4T_{pm} + 3T_h + 1T_{e/d} + 1T^{cert}_{ver} \approx 27.714$ ms	51.471 ms
Alzahrani et al. [24]	$3T_h + 1T_{PUF} + 1T_{FEG} \approx 1.894$ ms	$5T_h + 1T_{FER} \approx 2.905$ ms	4.799 ms
Chen et al. [26]	$4T_h + 1T_{PUF} + 1T_{FEG} \approx 1.92$ ms	$5T_h + 1T_{FER} \approx 2.905$ ms	4.825 ms
SLAP [27]	$6T_h \approx 0.156$ ms	$11T_h \approx 0.121$ ms	0.277 ms
PEASE	$7T_h + 1T_{PUF} \approx 0.302$ ms	$8T_h \approx 0.088$ ms	0.39 ms

Figure 9 shows that only LAKD [21] and SLAP [27] have a lower cost than PEASE. However, considering the lack of scalability of these two schemes, the Supervisor needs to traverse the database and verify the calculation to obtain the corresponding parameters after receiving the request. Therefore, the actual calculation cost is much larger than the calculated value in the table above; PEASE, on the other hand, is the least computationally expensive because of its scalable design. Compared to other schemes using PUF, PEASE significantly reduces the computational overhead by 93.83%.

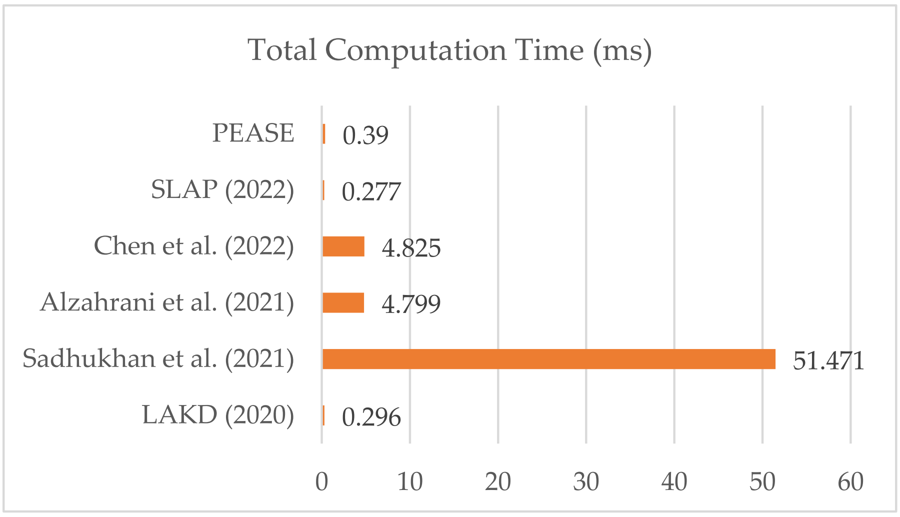

Figure 9. Performance comparison based on execution time.

In terms of communication overhead, we take the ID to be 128 bits, assuming SHA3-256 is used, 128 bits of random numbers, 320 bits of elliptic curve points (160 bits of X-axis coordinates), 160 bits of symmetric decryption and encryption blocks [27], 139 bytes of certificates [35], and 160 bits of timestamps [26]. The comparison of communication overhead during authentication is shown in Table 8.

Table 8. Comparison of communication overhead.

Scheme	Cost at Device	Cost at Supervisor	Total	No. of Messages
LAKD [21]	1344-bits	1344-bits	2688-bits	4
Sadhukhan et al. [22]	1752-bits	160-bits	1912-bits	3
Alzahrani et al. [24]	512-bits	640-bits	1152-bits	2
Chen et al. [26]	672-bits	672-bits	1344-bits	2
SLAP [27]	928-bits	1184-bits	2112-bits	2
PEASE	768-bits	768-bits	1536-bits	2

The two messages transmitted in the authentication phase of PEASE are $M_1 = AID_i$, D_1, D_2, D_3 and $M2 = D_4$, D_5, D_6, respectively; AID_i is 128 bits,$D_1 = S_i \oplus R_i \oplus r_1$ is 128 bits, and the remaining D_2, D_3, D_4, D_5 and D_6 are all 256 bits; therefore, M_1 and M_2's communication costs are both 768 bits, totaling 1536 bits.

It can be seen from Figure 10 that the communication cost of Alzahrani and Chen's protocol [24,26] is lower than that of our scheme. However, since the fuzzy extractor that cannot be omitted is not considered in their schemes, the communication cost caused by the fuzzy extractor is not calculated; hence, the communication cost calculation of the two in the above table is not referenced. Our communication cost is the lowest among the remaining schemes, at least 19.67% lower.

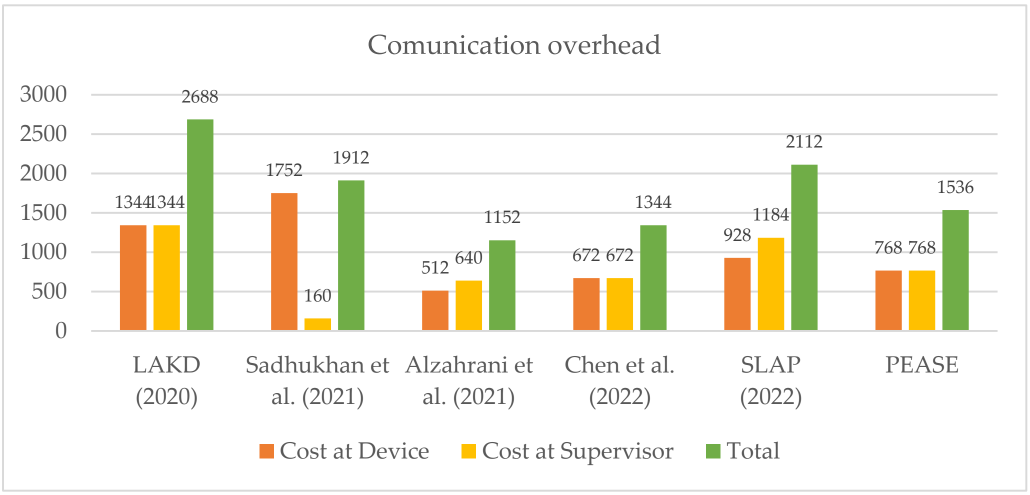

Figure 10. Performance comparison based on communication overhead.

To summarize, PEASE protocol provides significant security advantages, satisfactory performance, and high availability.

8. Conclusions

We have presented in this paper some of the drawbacks of the recently proposed M2M protocols for the IIoT. The reasons for these defects are analyzed and listed; after learning these lessons, an M2M authentication and session establishment protocol for the IIoT is proposed. The protocol is lightweight, and the required operations are only XOR, hashing, and PUF; the output of PUF has some noise; in reality, it needs a fuzzy extractor with a high computational cost for error correction; however, we circumvent the higher overhead caused by using fuzzy extractors by not reusing the same challenge in the authentication phase.

In addition, considering the difficulty of clock synchronization in constrained distributed scenarios, PEASE protocol is designed not to require time synchronization between devices; it improves the feasibility while reducing the overhead and the possibility of being exploited by attackers. CPN Tools is used to model checking the protocol's security. The protocol's security is verified by simulating an extended DY attacker. A comparison with the recent related work in security and performance shows that PEASE protocol outperforms the existing candidate schemes in terms of security, performance, and availability.

The limitation of this paper is that it does not regard peer-to-peer (P2P) communication security protection, so it cannot meet all communication scenarios of the IoT. In future work, we will consider the requirement for P2P communication in the IoT and try to build an available, stable, and secure P2P cryptographic protection protocol scheme with the assistance of CPN modeling.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/electronics11233920/s1, Folder S1: State space and verification results; File S1: ModelVerification.cpn. Folder S1 contains all state space reports and screenshots of state space searches. File S1 is the complete CPN model file for this article, it can be opened using CPN Tools 4.0.1.

Author Contributions: Conceptualization, X.G. and T.F.; methodology, X.G.; validation, X.G., T.F. and M.A.; formal analysis, X.G.; investigation, X.G. and M.A.; resources, X.G. and T.F.; writing—original draft preparation, X.G.; writing—review and editing, X.G. and M.A.; supervision, T.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under grant number 62162039 and the National Natural Science Foundation of China under grant number 61762060.

Data Availability Statement: Not applicable.

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

References

1. Schiller, E.; Aidoo, A.; Fuhrer, J.; Stahl, J.; Ziörjen, M.; Stiller, B. Landscape of IoT security. *Comput. Sci. Rev.* **2022**, *44*, 100467. [CrossRef]
2. Sisinni, E.; Saifullah, A.; Han, S.; Jennehag, U.; Gidlund, M. Industrial internet of things: Challenges, opportunities, and directions. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4724–4734. [CrossRef]
3. Misra, S.; Roy, C.; Sauter, T.; Mukherjee, A.; Maiti, J. Industrial Internet of Things for Safety Management Applications: A Survey. *IEEE Access* **2022**, *10*, 83415–83439. [CrossRef]
4. Middleton, P.; Contu, R.; Pace, B.; Alaybeyi, S. Forecast: IoT Security, Worldwide. 2018. Available online: https://www.gartner.com/en/documents/3863770 (accessed on 21 July 2021).
5. Bhushan, B.; Sahoo, G. Requirements, Protocols, and Security Challenges in Wireless Sensor Networks: An Industrial Perspective. In *Handbook of Computer Networks and Cyber Security: Principles and Paradigms*; Gupta, B.B., Perez, G.M., Agrawal, D.P., Gupta, D., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 683–713.
6. Gündoğan, C.; Kietzmann, P.; Lenders, M.S.; Petersen, H.; Frey, M.; Schmidt, T.C.; Shzu-Juraschek, F.; Wählisch, M. The impact of networking protocols on massive M2M communication in the industrial IoT. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 4814–4828. [CrossRef]
7. Meng, Z.; Wu, Z.; Muvianto, C.; Gray, J. A data-oriented M2M messaging mechanism for industrial IoT applications. *IEEE Internet Things J.* **2016**, *4*, 236–246. [CrossRef]
8. Khan, M.N.; Rao, A.; Camtepe, S. Lightweight cryptographic protocols for IoT-constrained devices: A survey. *IEEE Internet Things J.* **2020**, *8*, 4132–4156. [CrossRef]
9. Vinoth, R.; Deborah, L.J.; Vijayakumar, P.; Kumar, N. Secure multifactor authenticated key agreement scheme for industrial IoT. *IEEE Internet Things J.* **2020**, *8*, 3801–3811. [CrossRef]
10. Singh, S.; Sharma, P.K.; Moon, S.Y.; Park, J.H. Advanced lightweight encryption algorithms for IoT devices: Survey, challenges and solutions. *J. Ambient Intell Humaniz. Comput.* **2017**, 1–18. [CrossRef]
11. Sabri, C.; Kriaa, L.; Azzouz, S.L. Comparison of IoT constrained devices operating systems: A survey. In Proceedings of the 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), Hammamet, Tunisia, 3 November 2017; pp. 369–375.
12. Restuccia, G.; Tschofenig, H.; Baccelli, E. Low-power IoT communication security: On the performance of DTLS and TLS 1.3. In Proceedings of the 2020 9th IFIP International Conference on Performance Evaluation and Modeling in Wireless Networks (PEMWN), Berlin, Germany, 1–3 December 2020; pp. 1–6.
13. Arvind, S.; Narayanan, V.A. An overview of security in coap: Attack and analysis. In Proceedings of the 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), Coimbatore, India, 15–16 March 2019; pp. 655–660.
14. Pappu, R.; Recht, B.; Taylor, J.; Gershenfeld, N. Physical one-way functions. *Science* **2002**, *297*, 2026–2030. [CrossRef]
15. Braeken, A. PUF based authentication protocol for IoT. *Symmetry* **2018**, *10*, 352. [CrossRef]
16. Idriss, T.A.; Idriss, H.A.; Bayoumi, M.A. A lightweight puf-based authentication protocol using secret pattern recognition for constrained iot devices. *IEEE Access* **2021**, *9*, 80546–80558. [CrossRef]
17. Dodis, Y.; Ostrovsky, R.; Reyzin, L.; Smith, A. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In *International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin, Germany, 2004.
18. Esfahani, A.; Mantas, G.; Matischek, R.; Saghezchi, F.B.; Rodriguez, J.; Bicaku, A.; Maksuti, S.; Tauber, M.G.; Schmittner, C.; Bastos, J. A lightweight authentication mechanism for M2M communications in industrial IoT environment. *IEEE Internet Things J.* **2017**, *6*, 288–296. [CrossRef]
19. Aghili, S.F.; Mala, H. Breaking a lightweight M2M authentication protocol for communications in IIoT environment. *Cryptol. ePrint Arch.* 2018. Available online: https://eprint.iacr.org/2018/891 (accessed on 14 July 2022).
20. Adeel, A.; Ali, M.; Khan, A.N.; Khalid, T.; Rehman, F.; Jararweh, Y.; Shuja, J. A multi-attack resilient lightweight IoT authentication scheme. *Trans. Emerg. Telecommun. Technol.* **2022**, *33*, e3676. [CrossRef]
21. Lara, E.; Aguilar, L.; Sanchez, M.A.; García, J.A. Lightweight authentication protocol for M2M communications of resource-constrained devices in industrial Internet of Things. *Sensors* **2020**, *20*, 501. [CrossRef] [PubMed]
22. Sadhukhan, D.; Ray, S.; Obaidat, M.S.; Dasgupta, M. A secure and privacy preserving lightweight authentication scheme for smart-grid communication using elliptic curve cryptography. *J. Syst. Archit.* **2021**, *114*, 101938. [CrossRef]
23. Zhang, L.; Zhao, L.; Yin, S.; Chi, C.-H.; Liu, R.; Zhang, Y. A lightweight authentication scheme with privacy protection for smart grid communications. *Future Gener. Comput. Syst.* **2019**, *100*, 770–778. [CrossRef]
24. Alzahrani, B.A.; Mahmood, K. Provable privacy preserving authentication solution for internet of things environment. *IEEE Access* **2021**, *9*, 82857–82865. [CrossRef]
25. Chikouche, N.; Cayrel, P.-L.; Mboup, E.H.M.; Boidje, B.O. A privacy-preserving code-based authentication protocol for Internet of Things. *J. Supercomput.* **2019**, *75*, 8231–8261. [CrossRef]

26. Chen, C.-M.; Li, X.; Liu, S.; Wu, M.-E.; Kumari, S. Enhanced authentication protocol for the Internet of Things environment. *Secur. Commun. Netw.* **2022**, *2022*, 1–13. [CrossRef]
27. Panda, S.; Mondal, S.; Kumar, N. SLAP: A Secure and Lightweight Authentication Protocol for machine-to-machine communication in industry 4.0. *Comput. Electr. Eng.* **2022**, *98*, 107669. [CrossRef]
28. Braeken, A. Public key versus symmetric key cryptography in client–server authentication protocols. *Int. J. Inf. Secur.* **2022**, *21*, 103–114. [CrossRef]
29. Al-Shareeda, M.A.; Manickam, S.; Mohammed, B.A.; Al-Mekhlafi, Z.G.; Qtaish, A.; Alzahrani, A.J.; Alshammari, G.; Sallam, A.A.; Almekhlafi, K. Cm-cppa: Chaotic map-based conditional privacy-preserving authentication scheme in 5g-enabled vehicular networks. *Sensors* **2022**, *22*, 5026. [CrossRef] [PubMed]
30. Al-Shareeda, M.A.; Manickam, S.; Mohammed, B.A.; Al-Mekhlafi, Z.G.; Qtaish, A.; Alzahrani, A.J.; Alshammari, G.; Sallam, A.A.; Almekhlafi, K. Chebyshev polynomial-based scheme for resisting side-channel attacks in 5g-enabled vehicular networks. *Appl. Sci.* **2022**, *12*, 5939. [CrossRef]
31. Garg, V.K.; Mittal, N. Time and State in Asynchronous Distributed Systems. In *Wiley Encyclopedia of Computer Science and Engineering*; Wiley: Hoboken, NJ, USA, 2007. [CrossRef]
32. Gope, P. PMAKE: Privacy-aware multi-factor authenticated key establishment scheme for advance metering infrastructure in smart grid. *Comput. Commun.* **2020**, *152*, 338–344. [CrossRef]
33. Dolev, D.; Yao, A. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [CrossRef]
34. Gong, X.; Feng, T. Lightweight Anonymous Authentication and Key Agreement Protocol Based on CoAP of Internet of Things. *Sensors* **2022**, *22*, 7191. [CrossRef]
35. Mattsson, J.P.; Selander, G.; Raza, S.; Höglund, J.; Furuhed, M. CBOR Encoded X.509 Certificates (C509 Certificates). Internet Engineering Task Force. January 2022. Available online: https://datatracker.ietf.org/doc/draft-ietf-cose-cbor-encoded-cert/ (accessed on 17 July 2022).