PUFTAP-IoT: PUF-Based Three-Factor Authentication Protocol in IoT Environment Focused on Sensing Devices

JoonYoung Lee, JiHyeon Oh, DeokKyu Kwon, MyeongHyun Kim, SungJin Yu, Nam-Su Jho, and Youngho Park

Abstract: In IoT-based environments, smart services can be provided to users under various environments, such as smart homes, smart factories, smart cities, smart transportation, and healthcare, by utilizing sensing devices. Nevertheless, a series of security problems may arise because of the nature of the wireless channel in the Wireless Sensor Network (WSN) for utilizing IoT services. Authentication and key agreements are essential elements for providing secure services in WSNs. Accordingly, two-factor and three-factor-based authentication protocol research is being actively conducted. However, IoT service users can be vulnerable to ID/password pair guessing attacks by setting easy-to-remember identities and passwords. In addition, sensors and sensing devices deployed in IoT environments are vulnerable to capture attacks. To address this issue, in this paper, we analyze the protocols of Chunka et al., Amintoosi et al., and Hajian et al. and describe their security vulnerabilities. Moreover, this paper introduces PUF and honey list techniques with three-factor authentication to design protocols resistant to ID/password pair guessing, brute-force, and capture attacks. Accordingly, we introduce PUFTAP-IoT, which can provide secure services in the IoT environment. To prove the security of PUFTAP-IoT, we perform formal analyses through Burrows Abadi Needham (BAN) logic, Real-Or-Random (ROR) model, and scyther simulation tools. In addition, we demonstrate the efficiency of the protocol compared with other authentication protocols in terms of security, computational cost, and communication cost, showing that it can provide secure services in IoT environments.

Keywords: IoT; WSN; PUF; biometrics; honey list; authentication; BAN logic; ROR model; scyther

1. Introduction

The rapid development of wireless networks and the Internet of Things (IoT) has created opportunities to communicate with things over the Internet. Wireless sensor networks (WSN), a combination of wireless networks and IoT sensors, are garnering increasing attention worldwide as an exciting new paradigm of IoT in various fields, such as smart home, smart city, smart transportation, and smart agriculture [1–3]. In this IoT-based environment, data are collected through various sensors and sensing devices, and users can access them through a gateway node. Through WSN, users can use convenient services in real-time through IoT devices in an IoT-based environment. For example, with their IoT devices, users can remotely operate the lights in their house or sprinklers in their garden.

However, because this convenient service is provided through a wireless network, it is vulnerable to illegal access by malicious attackers [4,5]. This can harm the convenience of IoT, such as invasions of user privacy and eavesdropping on privacy. Malicious attackers can also be insiders or outsiders seeking to breach network security and falsify data integrity. Moreover, problems of node and link failures (i.e., cascading failures) can occur due to the
limitations of resources and energy of IoT equipment [6,7]. To this end, the development of lightweight protocols that provide secure communication between nodes and that can overcome resource and energy limitations is ongoing.

Key agreement and authentication protocols are an integral part of addressing security vulnerabilities in WSN and IoT environments and are being studied continuously. Two-factor-based authentication protocols consisting of passwords and smart cards have been proposed for secure communication in IoT-based environments [8–18]. However, these two-factor-based authentication protocols are also vulnerable to smart card theft and guessing attacks, among other attacks. In addition, some researchers have argued that, in two-factor authentication protocols, an attacker can guess an ID/password pair as users create easy-to-remember ID/password pair for convenience [19–21]. Therefore, the researchers argued that attackers can guess an ID/password pair within polynomial time. Accordingly, to respond to various attacks, including ID/password pair guessing attacks, three-factor-based authentication protocols have been proposed, involving user’s biometric information [22–28].

Although the three-factor-based authentication protocol is more secure than the two-factor-based authentication protocol, some researchers have found that the three-factor authentication protocols proposed in WSN and IoT environments are also not secure against multiple attacks. Although three-factor-based authentication protocols can defend against ID/password pair guessing attacks, they are still vulnerable to attacks that can be performed with values obtained through device capture attacks. Additionally, the three-factor authentication protocol is still vulnerable to replay, impersonation, and session key disclosure attacks.

In this paper, we analyze the security of two-factor-based and three-factor-based authentication protocols to discover their vulnerabilities. Chunka et al.’s protocol [16] is vulnerable to known session-specific temporary information, ID/password pair guessing, and impersonation attacks. The protocol of Amintoosi et al. [18] is also vulnerable to ID/password pair guessing attacks, thus allowing impersonation attacks. The protocol of Hajian et al. [27] is vulnerable to device physical capture attacks, and through these attacks, device impersonation and session key disclosure attacks are possible.

This paper introduces the Physical Unclonable Function (PUF) [29], which can strengthen security against device capture attacks, and honey list [30,31], which can prevent off-line guessing and brute-force attacks from solving the vulnerabilities of three-factor-based authentication protocols. With honey list, the authentication protocol can be secure, even if two of three factors of the authentication protocol are leaked. In addition, we configure the authentication protocol with XOR and hash functions for real-time communication of sensing devices and prevention of system down.

Therefore, this study aims to solve the security vulnerabilities of the two-factor and three-factor-based WSN authentication protocols [16,18,27]. In addition, we propose PUFTAP-IoT, a secure protocol for IoT-based environments using the three factors that are safe against various attacks in the IoT environment.

We adopt two technologies for a secure protocol for sensing devices in the IoT environment. We also invoke honey list technology to defend against online-guessing and brute-force attacks and consider PUF to be safe against takeover attacks of sensors and sensing devices. The contributions of this paper are as follows:

- We prove the vulnerabilities of protocols by Chunka et al. [16] and Amintoosi et al. [18], which are two-factor authentication protocols, and Hajian et al. [27], which is a three-factor authentication protocol.
- PUFTAP-IoT adopts PUF [29] and honey list [30,31] technology to be safe against various attacks. In addition, to solve the resource problem of sensors and sensing devices, only XOR and hash functions excluding elliptic curve cryptography (ECC) functions are used to lighten the protocol.
- Informal (non-mathematical) analysis and formal analysis are performed to prove the security of the proposed PUFTAP-IoT. Formal analysis uses the widely adopted
Burrows Abadi Needham (BAN) logic [32] and Real-Or-Random (ROR) model [33]. We also use the scyther simulation tool [34] to show that PUFTAP-IoT is secure in networks over public channels.

- We compare PUFTAP-IoT with other authentication protocols in terms of computation cost, communication cost, and security to analyze its efficiency.

The remainder of this paper is organized as follows: Section 2 reviews two-factor and three-factor-based authentication protocols in IoT and WSN environments. Section 3 outlines the proposed system model, attacker model, PUF, fuzzy extraction, and honey list. We analyze the protocols of Zou et al., Amintoosi et al., and Hajian et al. to demonstrate security vulnerabilities. Section 5 describes PUFTAP-IoT, and the safety of PUFTAP-IoT is analyzed in Section 6. We also analyze the efficiency of the protocol in Section 7. Finally, Section 8 concludes the paper.

2. Related Works

Lamport [35] first proposed a password-based authentication protocol in 1981. Since then, many related studies on password-based, two-factor authentication protocols have been proposed in various network environments to protect users’ privacy. In 2009, Das [8] proposed a two-factor authentication concept using a smart card with password in an IoT-based WSN environment. Das argued that the proposed scheme has a security advantage in that it uses only a hash function to reduce communication overhead and resist various attacks. However, He et al. [9] proved [8]’s authentication protocol is vulnerable to insider attacks along with impersonation attacks in 2010. In addition, He et al. presented an improved protocol as a countermeasure against these attacks. Unfortunately, it was found by Kumar and Lee [10] that He et al.’s protocol also does not guarantee mutual authentication and cannot generate a session key. Turkanović et al. proposed a new authentication and key agreement method in the WSN environment, focusing on heterogeneous IoT. The proposed scheme allows users to negotiate session keys securely with sensor nodes using the authentication protocol. However, Amin and Biswas [12] demonstrated that the protocol of Turkanović et al. is not secure against impersonation, identity guessing, and password guessing attacks. Moreover, they showed that their scheme has an inefficient authentication phase. Amin and Biswas proposed a protocol that compensated for these problems. However, Wu et al. [13] found that the protocol of Amin and Biswas are also vulnerable to sensor capture and guessing and spoofing attacks. Shuai et al. [14] suggested an authentication protocol for smart homes in 2019. In their protocol, they use Elliptic Curve Cryptography (ECC) for efficient and anonymous authentication. They demonstrate that their protocol is secure against a variety of attacks, including desynchronization and verification table stolen attacks. However, Zou et al. [15] proved Shuai et al.’s protocol is insecure against perfect forward secrecy, node capture attack, and impersonation attacks. Moreover, they proposed more secure user authentication schemes for smart homes. In 2021, Chunka et al. [16] point out the problems with the authentication protocol for WSN environment proposed by Kalra and Sood [17]. They pointed out that the protocol proposed by Kalra and Sood is vulnerable to sensor node capture attacks and cannot provide perfect forward secrecy. In 2022, Amintoosi et al. [18] proposed a two-authentication-based authentication and key agreement protocol to ensure the privacy and security of patients’ health-related data. They claim that their protocol is safe from various attacks and is a lightweight protocol using only hash and XOR functions.

According to [19,20], people tend to choose ID/password pairs that are easy to remember. As a result, ID and password pairs are chosen from a small dictionary space. This allows an attacker to guess a user’s ID and password in polynomial time [21]. Many researchers have proposed a secure three-factor authentication scheme to prevent simultaneous ID and password pair guessing attacks.

In 2016, Amin et al. [22] proposed a three-factor authentication protocol for WSN. They designed an anonymity-preserving authentication scheme for WSN and proved that their proposed protocol is secure against multiple attacks and is more efficient than
other protocols. However, Jiang et al. [23] showed that the protocol of Amin et al. is insecure against replay attacks and does not provide complete forward secrecy. To solve this security flaw, Jiang et al. presented an authentication protocol based on the Rabin cryptosystem for WSN. However, Ostad-Sharif et al. [24] demonstrate that the Jiang et al.’s protocol also does not provide perfect forward secrecy. In 2019, Mo et al. [25] proposed a secure three-factor-based key agreement and user authentication protocol for WSN. They presented a protocol based on ECC. They demonstrated that their protocol is able to provide security against untraceability and user anonymity. However, Yu and Park [26] pointed out that the protocol of Mo et al. is not safe for impersonation, replay, and session key disclosure attacks. Unfortunately, Hajian et al. [27] proved that the protocol proposed by Ostad-Sharif et al. [24] and Yu et al. [26] is also vulnerable to some attacks. To prevent security problems, Hajian et al. proposed a lightweight authentication protocol for IoT environments. They argued that the proposed protocol can defend against multiple attacks. In 2019, Amintoosi et al. [18] pointed out the security vulnerabilities of the authentication protocol for e-health proposed by Aghili et al. [28]. They proposed a lightweight authentication protocol for smart healthcare services that solves the security vulnerabilities of Aghili et al.’s protocol.

However, we prove that some schemes [16,18,27] are vulnerable to security attacks. We found that Chunka et al. [16] protocol is vulnerable to known session-specific temporary information, ID/password pair guessing, and impersonation attacks. Additionally, we prove that Amintoosi et al.’s protocol [18] cannot withstand identity and password guessing attacks and smart card stolen attacks. Finally, Hajian et al.’s protocol [27] is vulnerable to device capture and session key disclosure attacks.

3. Preliminaries

This section introduces the PUFTAP-IoT system model and an adversary model for security analysis of authentication protocols. In addition, we briefly describe PUF, fuzzy extraction, and honey_list, which are the security technologies adopted in the proposed IoT-TFBAP.

3.1. The Proposed System Model

The system model of PUFTAP-IoT is shown in Figure 1. PUFTAP-IoT consists of following three entities:

- **User**: The user requests communication to the gateway to use the sensing device. Only registered users can use IoT services by requesting communication to the gateway.
- **Sensing device**: Sensing devices are smart devices deployed in various IoT environments. Examples in Figure 1 include smart agriculture, vehicles, smart doors, and smart watches. They collect data and provide it to users, and users can use the data to execute any commands they want. Sensing devices also have limited computational power.
- **Gateway**: All service users and sensing devices must be registered with the gateway. A gateway is a trusted entity that is responsible for the process and regulates authentication requests between users and sensing devices.

![Figure 1. PUFTAP-IoT’s system model.](image-url)
controls communication between users and sensing devices. PUFTAP-IoT consists of a registration phase, login and authentication phase, and password and biometrics update phase. In the registration phase, users and sensing devices are registered with the gateway through secure channels. During the login and authentication phase, the user, gateway, and sensing device authenticate each other and generate a session key for communication. In the future, the user can safely communicate with the sensing device using this session key. In the password and biometrics update phase, users can update their passwords and biometrics if desired. To defend against malicious adversaries’ ID/password guessing attacks and brute-force attacks, the gateway creates and stores honey list. In addition, the sensing devices have built-in PUF technology to protect them from physical capture attacks.

3.2. The Adversary Model
We adopt the “Dolev-Yao (DY) adversary model” [36] to analyze the proposed protocols [15,18,27] and the IoT-TFBAP. The DY adversary model is a widely adopted model to analyze the security of wireless networks and assumes the following:

- The adversary can learn messages by intercepting messages delivered over insecure, public wireless channels. Through the learned message, the adversary can create a valid message and insert and modify it.
- The adversary can obtain stored values by stealing a valid user’s smart card and sensing device [37].
- The adversary can guess the user’s ID/password pair in polynomial time [21].
- The adversary can perform guessing, impersonation, known session-specific temporary information, and session key disclosure attacks using the acquired values.

3.3. Physical Unclonable Function
We adopt PUF technology to securely store secret parameters in the sensing device. PUF can be described as “the representation of the unique, non-replicable, instance-specific functionality of a physical entity” [29]. The randomness and uncertainty in integrated circuit fabrication is less likely to create duplicates, making PUFs increasingly visible in the security realm. PUF receives the challenge C and obtains its response R through the physical properties of C and the integrated chip (IC). Since both the accepted C and the generated R are strings of bits, PUF is expressed as \( R = PUF(C) \) and can be considered as a one-way function. In an ideal situation, a one-to-one correspondence exists between a challenge–response pair and a PUF, where if a challenge is assigned to the same PUF multiple times, the generated response is the same, and when the same challenge is given to different PUFs, the response obtained is different. PUF also has the following characteristics:

- It is impossible to clone PUF to create the same device [38].
- Any attempt to change the device containing the PUF will change the PUF’s behavior and destroy the PUF [39].
- In real-world manufacturing circuits, the difference between mapping input and output functions is fixed and unpredictable. In this respect, the hardware is equivalent to a one-way function [40].

However, due to environmental and circuit noise, PUFs always output varying responses with some margin of error in C_s. To solve this problem, PUF is being applied with fuzzy extractor technology [41].

3.4. Fuzzy Extraction
To solve the problem of noisy PUF, we introduce fuzzy extraction technology [41]. Moreover, we can use fuzzy extraction to solve the noise that can occur in the biometric input. The fuzzy extractor consists of the GEN function and the REP function.

The GEN function is for generating key information corresponding to the entered value. Entering the data \( D_i \) into the GEN function outputs the secret key data \( R_i \), which
is a uniform random string. The GEN function also outputs the string $P_i$, which helps to remove the noise and recover the key value.

The REP function restores the secret key $R_i$. Enter the data $D_i$ and the helper string $P_i$ into the REP function. At this time, $D_i$ may generate noise. For this, $P_i$ helps to output the correct $R_i$. To recover the same $R_i$, the metric space distance between $D_i$ and $D_j$ must be within the specified tolerance.

3.5. Honey List

Assume that attackers attempt to obtain useful data by performing brute-force and online-guessing attacks. In this case, honey_list prevents the algorithm “Honey Encryption (HE)” [30,31] from attempting to obtain data by guessing the password. If an adversary attempts attacks with the wrong password, HE uses an algorithm to generate fake valid messages, “Honey words”. [42] has more details on the honey word generation algorithm.

Various methods have been used to resist brute-force or online-guessing attacks using honey_list at the login and authentication phase. Out of all of them, PUFTAP-IoT calls honey_list by adopting the following method. If an attacker tries to login using the guessing password, the login proceeds as usual, but the gateway monitors the attacker’s login source for intrusion detection. The gateway also kills the session “when the number of entries in honey_list exceeds a predefined threshold” and notifies the user to update their password.

4. Cryptanalysis of Authentication Protocols

This section shows the analysis of various authentication protocols using sensor or sensing devices in an IoT environment. A review of each protocol is omitted, and for convenience of explanation, $S$ (sensor) of Chunka et al. and Amintoosi et al. and $S$ (sensing device) of Hajian et al. are all denoted as $SD$ (sensing device). The rest of the notation is the same as that of each authentication protocol. Table 1 shows the notations used in this paper.

Table 1. Notation.

| Notations | Meanings |
|-----------|----------|
| $U_i$     | i-th user |
| $SD_j$    | j-th sensing device |
| GW        | Gateway node |
| SC        | Smartcard |
| $ID_i$    | Identity of $U_i$ |
| $SID_j$   | Identity of $SD_j$ |
| $PW_i$    | Password of $U_i$ |
| $HPW_i$   | The hidden password of i-th user |
| $Bi$      | Biometrics of $U_i$ |
| $PUF$     | The Physical Unclonable Function |
| $C_j, R_j$| The challenge/response pair |
| $GEN, REP$| Generation and reproduction algorithm of fuzzy extractor |
| $K_{gw}$  | Secret key of GW |
| $R_x, N_x$| Random nonces |
| $T_x$     | Timestamps |
| $HID_i, PSID_j$ | Pseudo-identity of $U_i$ and $SD_j$ |
| $THID_j$  | Temporary user identity $U_i$ |
| $Skey$    | Session key |
| $||$       | Data concatenation operator |
| $\oplus$  | Bitwise exclusive-or operator |
| $h(*)$    | Collision-resistant one-way hash function |

4.1. Cryptanalysis of Chunka et al.’s Protocol

We prove that Chunka et al.’s protocol [16] is not safe against known session-specific temporary information attacks and does not provide perfect forward secrecy.
4.1. Known Session-Specific Temporary Information Attacks

Suppose that the adversary \( Adv \) obtains a session-specific temporary information \( r_1 \). Then, \( Adv \) is able to compute the legitimate session key. The detailed steps are as follows:

**Step 1:** \( Adv \) computes \( h(a_i \oplus k) = r_1 \oplus MID_i \), since \( MID_i \) is public parameter. Then, \( Adv \) can obtain \( P_i \), where \( P_i \) is obtained through an insecure channel.

**Step 2:** \( Adv \) computes \( h(P_i||h(a_i \oplus k)||r_1) = E_j \oplus h(r_1 \oplus r_2 \oplus r_3) \) via \( E_j \), which is transmitted to the public channel.

**Step 3:** Finally, \( Adv \) can compute the legitimate session key \( SK = h(r_1 \oplus r_2 \oplus r_3)||h(P_i||h(a_i \oplus k)||r_1) \).

4.1.2. Off-Line Guessing Attacks

According to the adversary model in Section 3.2, the adversary \( Adv \) can guess the ID/PW pair in polynomial time. The detailed steps are as follows:

**Step 1:** \( Adv \) is able to obtain values \( \{X_i, Z_i, Q_i, R_i, h(\cdot)\} \) stored on the smart card via smart card stolen attacks. Then, \( Adv \) picks \( ID_a/PW_a \) and computes \( r_a = X_i \oplus h(ID_a||PW_a) \).

**Step 2:** \( Adv \) calculates \( Z_i = h(ID_a||PW_a||r_a) \) and checks if \( Z_i = Z_i \).

**Step 3:** If they are the same, \( Adv \) has successfully guessed the correct ID/password pair for the user. Otherwise, \( Adv \) repeats Steps 1 and 2.

4.1.3. Impersonation Attacks

After off-line guessing attacks, \( Adv \) can impersonate the valid user. The detailed steps are as follows.

**Step 1:** Through guessing attacks, \( Adv \) computes \( h(ID_i||r) \). Then, \( Adv \) can compute \( h(a_i) = Q_i \oplus h(ID_i||r) \) and \( h(a_i \oplus k) = R_i \oplus h(ID_i||r) \) because \( R_i \) and \( Q_i \) are values stored in the smart card.

**Step 2:** Then, \( Adv \) generates a random nonce \( r_1 \) and computes \( Mid_a = h(a_i \oplus k) \oplus r_a \), \( N_a = h(h(a_i \oplus k)||h(a_i)||r_a) \).

**Step 3:** Finally, \( Adv \) sends the message \( \{P_i, MID_i, N_i\} \). Thus, \( Adv \) can impersonate the legitimate user.

4.2. Cryptanalysis of Amintoosi et al.’s Protocol

This section shows that Amintoosi et al.’s protocol [18] is not secure to smart card stolen, off-line guessing, and impersonation attacks.

4.2.1. Off-line Guessing Attacks

The adversary \( Adv \) can obtain the sensitive information stored in the smart card. Then, \( Adv \) can guess the ID/password pair in polynomial time. The detailed steps are as follows:

**Step 1:** \( Adv \) can obtain values \( \{b_i, A_i, B_i, a_i\} \) stored on the smart card. Then, \( Adv \) picks \( ID_a/PW_a \) and computes \( p_a = h(ID_a||PW_a||a_i) \), \( N_a = A_a \oplus p_a \), and \( bID_a = h(b_i||ID_a) \).

**Step 2:** \( Adv \) calculates \( M_a = h(b_i||ID_a||bID_a) \) and \( B_a = h(M_a||ID_a||bID_a) \). Then, \( Adv \) checks if \( B_a = B_i \).

**Step 3:** If they are the same, \( Adv \) has successfully guessed the correct ID/password pair for the user. Otherwise, \( Adv \) repeats Steps 1 and 2.

4.2.2. Impersonation Attacks

After guessing the legitimate user’s ID/password pair, the \( Adv \) computes \( \{M1_i, M2_i, T_1, b_i, M2_i\} \) can be masquerading. The detailed steps are as follows:

**Step 1:** After off-line guessing attacks, \( Adv \) obtains valid values \( M_i \) and \( N_i \). Then, \( Adv \) can compute \( d_i = M1_i \oplus h(M_i||N_i||T_1) \) to obtain \( d_i \), where \( M1_i \) is transmitted to the public channel.

**Step 2:** Then, \( Adv \) also can compute \( M2_i = h(M_i||N_i||d_i) \).

**Step 3:** Therefore, \( Adv \) can compute \( M1_i, T_1, b_i, \) and \( M2_i \). This means that \( Adv \) can impersonate the valid user. So, we can say that Amintoosi et al.’s protocol is not secure against impersonation attacks.
4.3. Cryptanalysis of Hajian et al.’s Protocol

In this section, we show that Hajian et al.’s protocol [27] is vulnerable to device capture attacks, device impersonation attacks, and session key disclosure attacks.

4.3.1. Device Impersonation Attacks

The adversary $Adv$ can obtain the $\{SID,j,x_j,f_j\}$ stored in $SD$ through a device capture attack. After that, $Adv$ can impersonate as a valid $SD$ by generating a message using the obtained values. After the device capture attacks, the detailed steps of the $Adv$’s device impersonation attack are as follows:

**Step 1:** $Adv$ obtains the values $\{M_2,T_1\}$ via the message sent to the public channel. Then, $Adv$ can obtain $K_j$ through computing $K_j = h(x_j || f_j) \oplus M_2$.

**Step 2:** $Adv$ can compute the legitimate $V_2 = h(SID_j || (x_j \oplus f_j) || (K_j || T_1))$. Finally, $Adv$ can compute the valid response message $\{M_2, V_2\}$. Thus, we can say that $Adv$ can conduct device impersonation attacks.

4.3.2. Session Key Disclosure Attacks

After $Adv$ conducts device impersonation attacks, $Adv$ obtains $x_j, f_j$, and $K_j$. $Adv$ can calculate the session key using these values. Therefore, an attacker can perform session key disclosure attacks, and the detailed steps are as follows:

**Step 1:** $Adv$ can learn values $M_1$ and $M_5$ through the message sent over the open channel. Then, $Adv$ can compute $M_5 \oplus h(T_1 || f_j \oplus x_j) \oplus M_1 = (K'_i || TID^{new}_i)$.

**Step 2:** Then, $Adv$ can obtain $K'_i$ and $TID^{new}_i$.

**Step 3:** Therefore, $Adv$ can compute the session key $SK_{ij} = h(K'_i \oplus K_j || SID_j || TID^{new}_i)$. Thus, we can say that Hajian et al.’s protocol is not secure against session key disclosure attacks.

5. The Proposed PUFTAP-IoT

In this section, we describe the proposed PUFTAP-IoT. In the proposed protocol, we adopt PUF technology to withstand device capture attacks. Additionally, we also apply the user’s biometrics and honey list to prevent online-guessing and brute-force attacks. Accordingly, our protocol is observed to be secure against various attacks. Finally, we propose a lightweight protocol using XOR and hash functions to consider the resource limitations of sensing devices and to prevent system down.

5.1. Registration Phase

In order for a service user to use IoT services through a sensing device in an IoT environment, first, he/she must register his/her information in the gateway. Moreover, the sensing device also registers its information in the gateway. The registration phase for service users and sensing devices is shown in Figure 2, and the detailed registration phase is described below.
5.1.1. Service User Registration Phase

Service users create their own information through ID, password, and biometric information and register it with the gateway, and the gateway issues a smart card. Here are the detailed steps:

**Step 1:** The service user $U_i$ inputs his/her identity ID, password PW, and imprints $B_i$. Then, $U_i$ generates $α$ and $R_u$ and computes $Gen(B_i) = (R_i, P_i)$, $HID_i = h(ID_i || PW_i || R_u || R_i)$, $HPW_i = h(ID_i || PW_i || B_i)$, $(HID_i, HPW_i) ⊕ α$. Then, $U_i$ sends $\langle HID_i, HPW_i \oplus α \rangle$ to the gateway GW through a secure channel.

**Step 2:** After receiving the registration request message, GW checks the uniqueness of $HID_i$. If it has the uniqueness, GW generates a random nonce $R_{gw}$ and computes $A_i = h(HID_i || K_{gw} || R_{gw})$, $B_i = A_i ⊕ (HPW_i ⊕ α)$, $C_i = h(A_i || HID_i)$. Then, GW generates temporary service user identity $THID_i$. Stores $\{HID_i, THID_i\}, R_{gw}, honey_list = null$ $\langle C_i, THID_i \rangle$ (via secure channel) GW issues the smart card $SC = \langle B_i, C_i, THID_i \rangle$ to $U_i$ via a secure channel.

**Step 3:** $U_i$ computes $L_i = h(ID_i || PW_i || R_i) ⊕ R_u$, $B'_i = B_i ⊕ α = A_i ⊕ HPW_i$, $C'_i = h(C_i || HPW_i)$. Then, $U_i$ deletes $B_i$ and $C_i$ in SC and stores $L_i, B'_i, C'_i$ in SC.

5.1.2. Sensing Device Registration Phase

The sensing device $SD_j$ utilizes the PUF function for registration and registers its own information with GW. The detailed registration steps are as follows:

**Step 1:** $SD_j$ picks its identity $SID_j$ and challenge $C_j$. $SD_j$ generates a random nonce $R_{sd}$ and computes $Req_j = SID_j \oplus h(R_{sd})$, $R_j = PUF(C_j)$, $Gen(R_j) = \langle SDR_j, SDP_j \rangle$, $HSID_j = h(SID_j || SDR_j)$, $⟨ Req_j, R_{sd}, HSID_j,C_j ⟩$ (via secure channel) $\langle PSID_j, S1_j \rangle$ (via secure channel) Stores $\{HSID_j, PSID_j, S1_j, SDR_j \}$ $\langle PSID_j, S1_j \rangle$ (via secure channel) $\langle PSID_j, S1_j \rangle$ (via secure channel). $SD_j$ issues the smart card SC to $U_i$ via a secure channel.

**Step 2:** After receiving the registration request message, GW checks the uniqueness of $HID_i$. If it has the uniqueness, GW generates a random nonce $R_{gw}$ and $GW$ computes $A_i = h(HID_i || K_{gw} || R_{gw})$, $B_i = A_i ⊕ (HPW_i ⊕ α)$, and $C_i = h(A_i || HID_i)$. Then, GW generates the temporary service user’s identity $THID_j$ and stores $\{HID_j, THID_j\}, R_{gw}, honey_list = null$ in its secure database. GW issues the smart card SC to $U_i$ via a secure channel.

**Step 3:** $SD_j$ computes $L_i = h(ID_i || PW_i || R_i) ⊕ R_u$, $B'_i = B_i ⊕ α = A_i ⊕ HPW_i$, and $C'_i = h(C_i || HPW_i)$. Then, $U_i$ deletes $B_i$ and $C_i$ in SC and stores $L_i, B'_i, C'_i$ in SC.
Step 2: GW computes $SID_i = Req_i \oplus h(R_{sd})$ and GW generates a random secret key $RK_i$. GW also computes $PSID_i = h(HSID_i||RK_i)$ and $SI = h(PSID_i||h(K_{gw}||RK_i))$. Finally, GW stores \{(HSID_i, PSID_i), RK_i, C_i\} in its database and transmits \{(PSID_i, SI_i)\} to SD_i through a closed channels.

Step 3: After receiving the message, SD_i stores \{SID_i, PSID_i, SI_i, SDP_i\}.

5.2. Login and Authentication Phase

$U_i$ sends an authentication request message to GW after login through his/her smart card and credential information. After confirming this, GW sends an authentication message to the corresponding SD_i and each entity authenticates the response message. When authentication is completed, $U_i$, GW, and SD_i agree on a session key $Sky_i$ and secure communication can be guaranteed later through $Sky_i$. In addition, $U_i$ and GW update $THID_i$ to $THID_{i,\text{new}}$ when authentication and key agreement are successful. The detailed formula is as follows, and the entire steps are summarized in Figure 3:

![Figure 3. Login and authentication phase.](image-url)

**Step 1:** The service user $U_i$ inserts SC and inputs $ID_i, PW_i$, and $B_i$. Then, SC computes $Rep(B_i, P_i) = R_i, HID_i = h(ID_i||R_i), R_a = L_i \oplus h(ID_i||PW_i||R_i), HPW_i = h(ID_i||PW_i||R_a||R_i), A_1 = B_i' \oplus HPW_i$, and $C_i' = h(h(A_1|||HID_i|||HPW_i))$. SC checks $C_i = C_i'$. If it is correct, SC generates a random nonce $N_a$ and timestamp $T_1$. After that, SC computes $Msg_a = h(h(N_a||A_1)|||HID_i|||PSID_i), V_i = h(N_a||A_1) \oplus h(HID_i|||T_1). U_i$ sends the message $\{Msg_a, V_i, T_i, THID_i, PSID_{i,\text{new}}\}$ through an open channel.

**Step 2:** When GW receives the request message, GW checks $|T_1 - T_i| < \Delta T$. GW retrieves $HID_i$ corresponding to $THID_i$, and GW computes $A_1 = h(HID_i||K_{gw}|||R_{gw}), h(N_a||A_1) = h(HID_i||A_1|||T_1) \oplus V_i, Msg_a^* = h(h(N_a||A_1)|||HID_i|||PSID_i).$ GW checks if $Msg_a = Msg_a^*$. If it is not same, GW inserts $A_1^*$ into $\text{honey\_list}$. Otherwise, GW retrieves $(C_i, RK_i)$ corresponding to $PSID_i$. GW generates a random nonce $N_b$ and timestamp $T_2$ and computes $SI = h(PSID_i||h(K_{gw}||RK_i)), V_2 = C_i \oplus h(PSID_i||SI_i), V_3 = h(h(N_b||A_1)||h(N_b||SI_i) \oplus h(HSID_i||C_i||SI_i), and $Msg_b = h(h(h(N_b||A_1)||h(N_b||SI_i))|||T_2|| HSID_i||C_i |||SI_i).$ After computing, GW sends $\{Msg_a, V_2, V_3, THID_i, PSID_{i,\text{new}}\}$ to $SD_i$ via an open wireless channel.

**Step 3:** $SD_i$ checks $|T_2 - T_i| < \Delta T$. Then, $SD_i$ computes $V_i = V_2 \oplus h(PSID_i||SI_i), PUF(C_i) = R_i, Rep(R_i, SDP_i) = SDP_i, HSID_i = h(SID_i||SDR_i), K_{gw} = h(h(N_a||A_1)||h(N_a||SI_i)) \oplus V_3 \oplus h(HSID_i||C_i||SI_i), and$ $Msg_b^* = h(K_{gw}||T_2||HSID_i||C_i||SI_i). Then, SD_i checks $Msg_b = Msg_b^*$. If it is the same, $SD_i$ generates a random nonce $N_{sd}$ and
timestamp $T_3$ and $SD_j$ computes a session key $Skey = h(N_x||K_{SD})$. $SD_j$ also computes $V_4 = Skey \oplus h(HSID_i||SI_j||C_j||T_3)$ and $MS_g = h(C_j||HSID_i||Skey)$. After that, $SD_j$ sends the response message $\langle Msg_3, V_4, T_3 \rangle$ to GW.

**Step 4:** GW computes the session key $Skey = V_4 \oplus (HSID_i||SI_j||C_j||T_3)$, and computes $MS_g = h(C_j||HSID_i||Skey)$. Then, GW checks if $MS_g = MS_g$. If it verifies, GW computes $THID_{new} = h(N_x||A_j||N_x||THID_i)$, $V_5 = Skey \oplus h(N_x||A_j||HID_j)$, $V_6 = THID_{new} \oplus h(HID_i||THID_i||h(N_x||A_j))$, and $MS_g = h(Skey||THID_{new})$. After computing, GW transmits $(Msg_4, V_5, V_6)$ to $UI$ through an insecure channel.

**Step 5:** After receiving the response message, $UI$ computes a session key $Skey = V_5 \oplus h(N_x||A_j)||HID_j)$. Additionally, $UI$ also computes $V_6 = THID_{new} \oplus h(HID_i||THID_i||h(N_x||A_j))$ and $MS_g = h(Skey||THID_{new})$. Then, $UI$ checks if $MS_g = MS_g$. If it is correct, the session key is authentic, and $UI$ and GW update $THID_{new}$.

5.3. Service User Password and Biometrics Update Phase

Assume that the service user $UI_i$ wants to use SC to change to a new password and biometrics. Specifically, this phase runs locally without any additional connections to GW, reducing computation and communication overhead. The following steps are the password and biometrics update process:

**Step 1:** The service user $UI_i$ inputs his/her identity $ID$ and password $PW_i$ and imprints his/her biometrics $B_i$. Then, SC computes $Rep(B_i, PW_i) = R_i$, $HID_i = h(ID_i||R_i)$, $R_u = L_i \oplus h(ID_i||PW_i||R_i)$, $HPW_i = h(ID_i||PW_i||R_i)$, $A_i = B_i \oplus HPW_i$, and $C_i = h(A_i||HID_i||HPW_i)$. SC checks $C_i = C_i$. If it is valid, SC asks $UI_i$ to enter the new password and biometrics.

**Step 2:** $UI_i$ enters a new password $PW_{new}$ and new biometrics $B_{new}$. SC proceeds to compute parameters $GEN(B_{new}, PW_{new}) = (R_{new}, PW_{new})$, $HPW_{new} = h(ID_i||PW_{new}||R_{new})$, $L_{new} = h(ID_i||PW_{new}||R_{new}) \oplus R_u$, $B_{new} = B_i \oplus HPW_i \oplus HPW_{new}$, and $C_{new} = h(C_i||HPW_i)$. Then, SC replaces $L_i, B_i, and C_i$ with $L_{new}, B_{new}, and C_{new}$.

6. Security Analysis

In this section, we analyze the security of PUFTAP-IoT. We first show that the protocol is safe against various attacks through informal analysis. In addition, we prove that mutual authentication and session key agreement of the protocol can be safely achieved through the universally used BAN logic and ROR model. Finally, we demonstrate the security of PUFTAP-IoT on a wireless network using the scyther simulation tool.

6.1. Informal Security Analysis

Here, we perform an informal (non-mathematical) security analysis to show that PUFTAP-IoT is safe against various attacks and also provides various security features.

6.1.1. Offline and Online-Guessing Attacks

Assume that the adversary $Adv$ obtains the $UI_i$’s SC and attempts an offline-guessing attack using parameters $\{L_i, B_i, C_i, THID_i\}$ in SC. However, since $Adv$ is the value that $R_i$ should be calculated as, the biometric of $UI_i$, $R_u = L_i \oplus h(ID_i||PW_i||R_i)$ could not be calculated. Moreover, $Adv$ tries an online-guessing attack for obtaining $UI_i$’s sensitive information. Unfortunately, the attacker does not know if the correct ID and password were guessed because of the $honey\_list$ stored on the gateway system. Moreover, PUFTAP-IoT is safe from online-guessing attacks because the session is terminated when the threshold of $honey\_list$ is exceeded. Therefore, PUFTAP-IoT is safe against offline- and online-guessing attacks.

6.1.2. Service User Anonymity

If $Adv$ steals $UI_i$’s SC and obtains values stored in SC, $Adv$ tries to obtain $UI_i$’s real identity, pseudo-identity or temporary identity. However, $Adv$ cannot obtain the ID of $UI_i$ and $HID_i$ because $HID_i$ is masked by the hash function and $R_i$. Although $THID_i$ is transmitted through the public channel, $THID_{new}$ is updated by GW when authentication
and key agreement are successful. In addition, $THID_{\text{new}}$ is masked with $N_u$ and $N_g$, and these values change every session. Therefore, PUFTAP-IoT can safely guarantee the anonymity of service users.

6.1.3. Impersonation Attack

In order for $Adv$ to disguise $U_i$, $GW$, and $SD_j$, $Adv$ must be able to compute the messages sent to the public channel. Messages sent from PUFTAP-IoT to public channels change per session due to random values $N_u$, $N_g$, and $N_{sd}$ and timestamps. In addition, $THID_j$ is also updated to $THID_{\text{new}}$ when the authentication is successful, so $Adv$ cannot calculate the correct message. Therefore, PUFTAP-IoT is resistant to impersonation attacks.

6.1.4. Sensing Device Physical Capture Attack

When $Adv$ performs a physical capture attack on $SD_j$, $Adv$ can obtain $\{SID_j, PSID_j, SI_j, SDP_j\}$ stored in $SD_j$. However, $Adv$ cannot calculate the correct session key through these parameters. In order for $Adv$ to calculate the session key, $K_{gw}(=h(h(N_u||A_i)||h(N_g||SI_i))) = V_3 \oplus h(HSID_j||C_j||SI_j)$ must be calculated. However, since $Adv$ cannot obtain $R_{ij}$, $Adv$ cannot compute $SDR_j$. Therefore, $Adv$ is not able to compute $HSID_j = h(SID_j||SDR_j)$. This is because $R_i$ is a value created by the PUF function, and the PUF is a function that is a physically unclonable circuit and cannot be duplicated. Therefore, PUFTAP-IoT is safe against sensing device physical capture attacks.

6.1.5. Replay and Man-in-the-Middle Attack

We assume that $Adv$ obtains messages transmitted over a public channel and information of $U_i$’s SC and $SD_j$. However, $Adv$ cannot compute $U_i$’s valid message as mentioned in Section 6.1.3. Additionally, $Adv$ also cannot generate $SD_j$’s valid messages according to Section 6.1.4. Additionally, every message changes with $N_u$, $N_g$, and $N_{sd}$ and timestamps every session. Therefore, we can say that PUFTAP-IoT is secure against replay and man-in-the-middle attacks.

6.1.6. Stolen Verifier Attack

Suppose $Adv$ obtains the $GW$ verification tables $\{HID_j, THID_j, R_{gw}, honey\_list = null\}$ and $\{HSID_j, PSID_j, RK_j, C_j\}$ to compute the session key $Skey$ or perform impersonation attacks. However, $Adv$ cannot compute $A_i = h(HID_j||K_{gw}||R_{gw})$ and $SI_j = h(PSID_j||h(K_{gw}||RK_j)$ without $GW$’s secret key $K_{gw}$. Furthermore, due to the nature of the PUF function, $Adv$ cannot compute $R_j = PUF(C_j)$. Therefore, $Adv$ cannot perform session key and impersonation attacks. Accordingly, we can say that PUFTAP-IoT is resistant to stolen verifier attacks.

6.1.7. Perfect Forward Secrecy

Assuming that $GW$’s secret key $K_{gw}$ is leaked to $Adv$, $Adv$ can try to calculate $Skey$ through $K_{gw}$. However, since $A_i$ and $SI_j$ are masked with $K_{gw}$ as well as $R_{gw}$ and $RK_j$ which are secret keys generated for each entity, $Adv$ cannot compute $A_i$ and $SI_j$. Therefore, since $Adv$ cannot compute valid $Skey$, PUFTAP-IoT can guarantee perfect forward secrecy.

6.1.8. Session-Specific Random Number Leakage Attack

Assume that $N_u$, $N_g$, and $N_{sd}$, which are random nonces generated in the session, were leaked to $Adv$. With these values, $Adv$ will try to compute $Skey$. However, $Adv$ cannot compute the session key $Skey = h(N_{sd}||h(h(N_u||A_i)||h(N_g||SI_j)))$. To calculate a valid $Skey$, $A_i$ and $SI_j$ must be calculated, but as mentioned in the Sections above, $A_i$ and $SI_j$ cannot be calculated by $Adv$. Therefore, PUFTAP-IoT is safe against session-specific random number leakage attacks.
6.1.9. Session Key Disclosure Attack

Adv wants to compute the Skey for obtaining sensitive information. However, as discussed in Sections 6.1.6–6.1.8, Adv cannot compute the valid Skey because of the computationally infeasible problem. Thus, PUFTAP-IoT prevents session key disclosure attacks.

6.1.10. Mutual Authentication

In PUFTAP-IoT, all entities authenticate each other by verifying messages containing $Msg_1$, $Msg_2$, $Msg_3$, and $Msg_4$. Moreover, these messages are changed with random numbers and current timestamps. After all entities authenticate each other, they compute the same Skey. Thus, PUFTAP-IoT guarantees mutual authentication.

6.2. BAN Logic

For proving that PUFTAP-IoT is able to provide secure authentication, we conduct BAN logic [32]. The notations used in BAN logic are shown in Table 2, and the five rules used are as follows [43–45]:

1. Jurisdiction rule:
   \[
   \begin{align*}
   &\chi \equiv \omega \Rightarrow \epsilon, \quad \chi \equiv \omega \equiv \epsilon \\
   \therefore &\chi \equiv \epsilon
   \end{align*}
   \]

2. Message meaning rule:
   \[
   \begin{align*}
   &\chi \equiv \chi \leftrightarrow \omega, \quad \chi \leftrightarrow \{\epsilon\}_{\chi} \\
   \Rightarrow &\chi \equiv B \sim \epsilon
   \end{align*}
   \]

3. Nonce verification rule:
   \[
   \begin{align*}
   &\chi \equiv \#(\epsilon), \quad \chi \equiv \omega \sim \epsilon \\
   \therefore &\chi \equiv \omega \equiv \epsilon
   \end{align*}
   \]

4. Belief rule:
   \[
   \begin{align*}
   &\chi \equiv \{\epsilon, F\} \\
   \therefore &\chi \equiv \epsilon
   \end{align*}
   \]

5. Freshness rule:
   \[
   \begin{align*}
   &\chi \equiv \#(\epsilon) \\
   \therefore &\chi \equiv \#(\epsilon, F)
   \end{align*}
   \]

Table 2. The basic notations of BAN logic.

| Notations | Description |
|-----------|-------------|
| $Skey$    | The session key in PUFTAP-IoT |
| $\#$     | The statement $S$ is fresh |
| $\chi \equiv \epsilon$ | $\chi$ believes the statement $\epsilon$ |
| $\chi < \epsilon$ | $\chi$ sees the statement $\epsilon$ |
| $\chi \sim \epsilon$ | $\chi$ once said $\epsilon$ |
| $\epsilon \sim \epsilon$ | $\epsilon$ is combined with formula $F$ |
| $\{\epsilon\}_{Key}$ | Encrypt $\epsilon$ with $Key$ |
| $\chi \Rightarrow \epsilon$ | $\chi$ controls $\epsilon$ |
| $Key \leftrightarrow \omega$ | $\chi$ and $\omega$ shard and use $Key$ for communication |

To implement BAN logic, we describe logical rules, goals, assumptions, and ideal forms, thereby proving that PUFTAP-IoT provides secure mutual authentication.
6.2.1. Goals

In order to prove that secure mutual authentication is achieved, the following goals must be achieved:

**Goal 1:** \( U_i \equiv (U_i \overset{\text{Skey}}{\leftrightarrow} GW) \)

**Goal 2:** \( U_i \equiv GW \equiv (U_i \overset{\text{Skey}}{\leftrightarrow} GW) \)

**Goal 3:** \( GW \equiv (U_i \overset{\text{Skey}}{\leftrightarrow} GW) \)

**Goal 4:** \( GW \equiv U_i \equiv (U_i \overset{\text{Skey}}{\leftrightarrow} GW) \)

**Goal 5:** \( SD_j \equiv (SD_j \overset{\text{Skey}}{\leftrightarrow} GW) \)

**Goal 6:** \( SD_j \equiv GW \equiv (SD_j \overset{\text{Skey}}{\leftrightarrow} GW) \)

**Goal 7:** \( GW \equiv (SD_j \overset{\text{Skey}}{\leftrightarrow} GW) \)

**Goal 8:** \( GW \equiv SD_j \equiv (SD_j \overset{\text{Skey}}{\leftrightarrow} GW) \)

6.2.2. Idealized Forms

The idealized forms are:

**M_1:** \( U_i \rightarrow GW : \{h(N_u || A_i) \}_HID_i \)

**M_2:** \( GW \rightarrow SD_j : \{h(h(N_u || A_i) || h(N_g || SI_j)) \}_SI_j \)

**M_3:** \( SD_j \rightarrow GW : \{h(N_{sd} || K_{gs}) \}_SI_j \)

**M_4:** \( GW \rightarrow U_i : \{h(N_{sd} || K_{gs}) \}_HID_i \)

6.2.3. Assumptions

The following assumptions are generated for the initial state of PUFTAP-IoT to achieve the BAN logic proof:

**A_1:** \( GW \equiv (U_i^{\text{HID}_i} \rightarrow GW) \)

**A_2:** \( GW \equiv #(N_u) \)

**A_3:** \( SD_j \equiv GW \overset{SI_j}{\leftrightarrow} SD_j \)

**A_4:** \( SD_j \equiv #(N_g) \)

**A_5:** \( GW \equiv GW \overset{SI_j}{\leftrightarrow} SD_j \)

**A_6:** \( GW \equiv #(N_{sd}) \)

**A_7:** \( U_i \equiv U_i^{\text{HID}_i} \rightarrow GW \)

**A_8:** \( U_i \equiv #(N_g) \)

**A_9:** \( U_i \equiv GW \rightarrow (U_i \overset{\text{Skey}}{\leftrightarrow} GW) \)

**A_{10}:** \( GW \equiv U_i \rightarrow (U_i \overset{\text{Skey}}{\leftrightarrow} GW) \)

**A_{11}:** \( SD_j \equiv GW \rightarrow (SD_j \overset{\text{Skey}}{\leftrightarrow} GW) \)

**A_{12}:** \( GW \equiv SD_j \rightarrow (SD_j \overset{\text{Skey}}{\leftrightarrow} GW) \)
6.2.4. Proof

The main proof using the rules and assumptions of BAN logic is:

**Step 1:** $S_1$ can be obtained from $M_1$.

$$ S_1 : GW \preccurlyeq \{h(N_u||A_i)\}_{HID_i} $$

**Step 2:** $S_2$ can be obtained by applying the MMR with $A_1$.

$$ S_2 : GW \equiv U_i \sim \{h(N_u||A_i)\}_{HID_i} $$

**Step 3:** $S_3$ can be gained from the FR with $S_2$ and $A_2$.

$$ S_3 : GW \equiv \#(h(N_u||A_i)) $$

**Step 4:** $S_4$ can be acquired by applying the NVR with $S_2$ and $S_3$.

$$ S_4 : GW \equiv U_i \equiv (h(N_u||A_i)) $$

**Step 5:** $S_5$ can be obtained from $M_2$.

$$ S_5 : SD_j \preccurlyeq \{h(h(N_u||A_i)||h(N_g||SI_j))\}_{SI_j} $$

**Step 6:** $S_6$ can be gained from MMR with $S_5$ and $A_3$.

$$ S_6 : SD_j \equiv GW \sim \{h(h(N_u||A_i)||h(N_g||SI_j))\}_{SI_j} $$

**Step 7:** $S_7$ can be obtained by applying FR with $S_6$ and $A_4$.

$$ S_7 : SD_j \equiv \#(h(h(N_u||A_i)||h(N_g||SI_j))) $$

**Step 8:** $S_8$ can be obtained from NVR with $S_6$ and $S_7$.

$$ S_8 : SD_j \equiv GW \equiv (h(h(N_u||A_i)||h(N_g||SI_j))) $$

**Step 9:** $S_9$ can be obtained from $M_3$.

$$ S_9 : GW \preccurlyeq \{h(N_{sd}||K_{gs})\}_{SI_j} $$

**Step 10:** $S_{10}$ can be gained from MMR with $S_9$ and $A_5$.

$$ S_{10} : GW \equiv SD_j \sim \{h(N_{sd}||K_{gs})\}_{SI_j} $$

**Step 11:** $S_{11}$ can be obtained by applying FR with $S_{10}$ and $A_6$.

$$ S_{11} : GW \equiv \#(h(N_{sd}||K_{gs})) $$

**Step 12:** $S_{12}$ can be obtained from NVR with $S_{10}$ and $S_{11}$.

$$ S_{12} : GW \equiv SD_j \equiv (h(N_{sd}||K_{gs})) $$

**Step 13:** $S_{13}$ can be obtained from $M_4$.

$$ S_{13} : U_i \preccurlyeq \{h(N_{sd}||K_{gs})\}_{HID_i} $$
Step 14: $S_{14}$ can be obtained from $MMR$ with $S_{13}$ and $A_7$.

$$S_{14} : U_i \equiv GW| \sim \{ h(N_{sd}||K_{gs}) \}_{HID_i}$$

Step 15: $S_{15}$ can be obtained from $FR$ with $S_{14}$ and $A_8$, since $K_{gs} = h(h(N_a||A_i)||h(N_{gs}||S_I))$.

$$S_{15} : U_i \equiv \#(h(N_{sd}||K_{gs}))$$

Step 16: $S_{16}$ can be obtained from $NVR$ with $S_{14}$ and $S_{15}$.

$$S_{16} : U_i \equiv GW| \equiv (h(N_{sd}||K_{gs}))$$

Step 17: $S_{17}$ and $S_{18}$ can be obtained from $S_8$ and $S_{12}$ since $Skey = h(N_{sd}||K_{gs})$.

$$S_{17} : SD_j| \equiv GW| \equiv (SD_j \xrightarrow{Skey} GW) \quad (Goal \ 6)$$

$$S_{18} : GW| \equiv SD_j| \equiv (SD_j \xrightarrow{Skey} GW) \quad (Goal \ 8)$$

Step 18: $S_{19}$ and $S_{20}$ can be obtained by applying $JR$ from $S_{17}$, $S_{18}$, $A_{11}$, and $A_{12}$.

$$S_{19} : SD_j| \equiv GW| \equiv (SD_j \xrightarrow{Skey} GW) \quad (Goal \ 5)$$

$$S_{20} : GW| \equiv SD_j| \equiv (SD_j \xrightarrow{Skey} GW) \quad (Goal \ 7)$$

Step 19: $S_{21}$ and $S_{22}$ can be obtained from $S_4$ and $S_{16}$ since $Skey = h(N_{sd}||K_{gs})$.

$$S_{21} : U_i| \equiv GW| \equiv (U_i \xrightarrow{Skey} GW) \quad (Goal \ 2)$$

$$S_{22} : GW| \equiv U_i| \equiv (U_i \xrightarrow{Skey} GW) \quad (Goal \ 4)$$

Step 20: $S_{23}$ and $S_{24}$ can be obtained by applying $JR$ from $S_{21}$, $S_{22}$, $A_9$, and $A_{10}$.

$$S_{23} : U_i| \equiv (U_i \xrightarrow{Skey} GW) \quad (Goal \ 1)$$

$$S_{24} : GW| \equiv (U_i \xrightarrow{Skey} GW) \quad (Goal \ 3)$$

Therefore, we prove that PUFTAP-IoT can satisfy all goals of BAN logic. Accordingly, it can be said that PUFTAP-IoT can guarantee secure mutual authentication.

6.3. ROR

We use the ROR model [33] to describe the semantic security of PUFTAP-IoT. We demonstrate that session key security can be guaranteed through the ROR model [46–48]. This section briefly describes the ROR model and presents a proof of the protocol’s session key security in Theorem 1. PUFTAP-IoT in the ROR model has three participants $P_i$, which are service user $P_{UI}^i$, gateway mathcalP_{GW}^i$, and sensing device $P_{SD}^i$. Additionally, for each participant, $th$ represents an instance of the running participant. We assume that an attacker $Adv$ can modify, remove, insert or learn messages sent during communication. In the ROR model, various queries are defined to simulate real-world attacks, Execute, CorruptSC, Reveal, Send, and Test queries. A detailed description of the query follows:

- **Execute($P_{UI}^i, P_{GW}^i, P_{SD}^i$)**: $Adv$ conducts Execute query to obtain messages sent over insecure channels between $U_i$, GW, and SD$J$.
- **CorruptSC($P_{UI}^i$)**: CorruptSC indicates that $Adv$ can obtain information stored in the smart card of $U_i$. 

Theorem 1. \( \text{GM} \) is that \( \text{Adv} \) returns the session key \( \text{Skey} \) between \( \mathcal{P}_1 \), \( \mathcal{P}_2 \), and \( \mathcal{P}_3 \). \( \text{Adv} \) is safe if \( \text{Adv} \) reveals \( \text{Skey} \) using the \( \text{Reveal}(\mathcal{P}) \) query.

**Send(\( \mathcal{P}_i \), \( M \)):** Send query allows \( \text{Adv} \) to transmit the \( M \) message to \( \mathcal{P}_i \) and receive a response.

**Test(\( \mathcal{P}_i \)):** A fair coin \( f_c \) is tossed before the game starts, and the result is known only to \( \text{Adv} \). \( \text{Adv} \) uses this result to determine Test query. If \( \text{Adv} \) conducts this query and \( \text{Skey} \) is fresh, \( \mathcal{P}_i \) will return \( \text{Skey} \) for \( f_c = 1 \) or a random nonce for \( f_c = 0 \). Otherwise, \( \mathcal{P}_i \) returns a null \((\bot)\).

After \( \text{Adv} \) conducts Test query on participants, \( \text{Adv} \) has to separate resulting values. \( \text{Adv} \) checks the consistency of the random bit \( f_c \) through the output of the Test query. For \( \text{Adv} \) to win the game, the guessed bit \( f_c^\prime \) must equal \( f_c \). Additionally, the collision-resistant one-way hash function \( h(\cdot) \) is accessible to all participants. Model \( h(\cdot) \) is a random oracle \( \text{Hash} \).

**Security Proof**

**Theorem 1.** \( \text{Adv} \) can obtain information by breaking the session key security. Mark the advantage of \( \text{Adv} \) running in polynomial time as \( \text{Adv}_t \). Then, we obtain:

\[
\text{Adv}_t \leq \frac{q_h^2}{|\text{Hash}|} + \frac{q_p^2}{|\text{PUF}|} + 2 \max\{ C \cdot q_s^+ \cdot \frac{q_s}{2D} \}
\]

Here, \( q_h, q_p, \) and \( q_s \) are the number of Hash, PUF, and Send queries, respectively. \(|\text{Hash}|\) and \(|\text{PUF}|\) are the range space of the hash function \( h(\cdot) \) and PUF function \( \text{PUF}(\cdot) \), respectively. In addition, \( C \) and \( s \) denote Zipf’s parameters, and \( 1_D \) is the number of bits in the biometric \( B_i \) of \( U_i \).

**Proof.** We run four sequence games \( \text{GM}_i \) to prove session key security, where \( i \in [0, 4] \). \( \text{Succ}_{\text{Adv},i} \) represents the event that \( \text{Adv} \) wins \( \text{GM}_i \) by correctly guessing any bit \( f_c \). The advantage of \( \text{Adv} \) winning the game \( \text{GM}_i \) is denoted by \( \text{Pr}[\text{Succ}_{\text{Adv},\text{GM}_i}] \). Each game is described below.

- **GM_0:** \( \text{Adv} \) can execute a real attack on PUFTAP-IoT through this game. \( \text{Adv} \) selects \( f_c \) at the beginning of \( \text{GM}_0 \). Then, according to this game, we obtain:

\[
\text{Adv}_t = |2\text{Pr}[\text{Succ}_{\text{Adv},\text{GM}_0}] - 1|
\]

- **GM_1:** \( \text{Adv} \) conducts the \( \text{Execute}(\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3) \) query through this game and eavesdrops transmitted messages \(< \text{Msg}_8, V_1, T_1, \text{THID}_j, \text{PSID}_j >, < \text{Msg}_9, V_2, T_2 >, < \text{Msg}_3, V_3 >, \) and \(< \text{Msg}_4, V_5, V_6 >. \text{Adv} \) then checks whether the derived \( \text{Skey} \) is real to execute \( \text{Reveal} \) and Test queries. In PUFTAP-IoT, the session key consists of \( \text{Skey} = h(\text{N}_d || K_{\text{ps}}) \). To derive \( \text{Skey} \), \( \text{Adv} \) must know the ID and random numbers of \( U_i, \text{GW}, \) and \( \text{SD}_j \). As a result, \( \text{Adv} \) never increases the probability of winning \( \text{GM}_1 \). Thus, \( \text{GM}_0 \) and \( \text{GM}_1 \) can be considered indistinguishable, and we obtain:

\[
\text{Pr}[\text{Succ}_{\text{Adv},\text{GM}_1}] = \text{Pr}[\text{Succ}_{\text{Adv},\text{GM}_0}]
\]

- **GM_2:** To obtain \( \text{Skey} \), \( \text{Adv} \) conducts \( \text{Hash} \) and \( \text{Send} \) queries. \( \text{Adv} \) can modify exchanged messages to carry out active attacks. However, all exchanged messages are protected using the one-way hash function \( h(\cdot) \) and consist of secret credentials and random numbers. Moreover, it is difficult for \( \text{Adv} \) to derive secret credentials and a random nonce because it is a computationally infeasible problem depending on the properties of \( h(\cdot) \). So, using the birthday paradox, we obtain:

\[
|\text{Pr}[\text{Succ}_{\text{Adv},\text{GM}_2}] - \text{Pr}[\text{Succ}_{\text{Adv},\text{GM}_0}]| \leq \frac{q_h^2}{2|\text{Hash}|}
\]
• **GM3**: It is similar to GM2. \( \text{Adv} \) conducts \( \text{Send} \) and PUF queries. As described in Section 3.3, PUF(\( \cdot \)) has security properties. So, we can obtain the following result:

\[
\left| \Pr[\text{Succ}_{\text{Adv},\text{GM3}}] - \Pr[\text{Succ}_{\text{Adv},\text{GM2}}] \right| \leq \frac{q_p^2}{2|\text{PUF}|} \tag{4}
\]

• **GM4**: In GM4, \( \text{Adv} \) can try to obtain \( \text{Skey} \) with the CorruptSC query. By the CorruptSC query, \( \text{Adv} \) can extract sensitive values \( \{L_i, B'_i, C'_i, \text{THID}\} \) stored on the smart card of \( U_i \). \( L_i, B'_i, \) and \( C'_i \) are expressed as \( L_i = h(ID_i || PW_i || R_i) \oplus R_{i'} \) \( B'_i = B_i \oplus \alpha = A_i \oplus \text{HPW}, \) and \( C'_i = h(C_i || \text{HPW}) \). Since \( \text{Adv} \) has no knowledge of identity \( ID \) and password \( PW_i \), \( \text{Adv} \) must guess these parameters from the extracted values. However, it is computationally infeasible for \( \text{Adv} \) to guess \( ID, \) password, and \( B_i \) simultaneously. Thus, GM3 and GM4 are indistinguishable. By utilizing Zipf’s law, we can obtain:

\[
\left| \Pr[\text{Succ}_{\text{Adv},\text{GM4}}] - \Pr[\text{Succ}_{\text{Adv},\text{GM3}}] \right| \leq \max\{ C \cdot q_{\text{send}} \frac{q_s}{2|\text{GM}|} \} \tag{5}
\]

Now that all the games were run, \( \text{Adv} \) has to guess the bit to win the game. Thus, we can obtain the following results:

\[
\Pr[\text{Succ}_{\text{Adv},\text{GM4}}] = \frac{1}{2} \tag{6}
\]

From Equations (1) and (2), we obtain the result as follows:

\[
\frac{1}{2} \text{Adv} = \Pr[\text{Succ}_{\text{Adv},\text{GM0}} - \frac{1}{2}] = \Pr[\text{Succ}_{\text{Adv},\text{GM1}} - \frac{1}{2}]. \tag{7}
\]

With Equations (5) and (6), we derive the below equation:

\[
\frac{1}{2} \text{Adv} = \Pr[\text{Succ}_{\text{Adv},\text{GM1}}] - \Pr[\text{Succ}_{\text{Adv},\text{GM4}}]. \tag{8}
\]

Using the trigonometric inequality, we can obtain the results of Equations (4), (5), and (7).

\[
\frac{1}{2} \text{Adv} = \left| \Pr[\text{Succ}_{\text{Adv},\text{GM1}}] - \Pr[\text{Succ}_{\text{Adv},\text{GM4}}] \right|
\leq \left| \Pr[\text{Succ}_{\text{Adv},\text{GM1}}] - \Pr[\text{Succ}_{\text{Adv},\text{GM3}}] \right|
+ \left| \Pr[\text{Succ}_{\text{Adv},\text{GM3}}] - \Pr[\text{Succ}_{\text{Adv},\text{GM4}}] \right|
\leq \frac{q_p^2}{2|\text{Hash}|} + \frac{q_p^2}{2|\text{PUF}|} + \max\{ C \cdot q_{\text{send}} \frac{q_s}{2|\text{GM}|} \} \tag{9}
\]

Finally, multiply both sides of Equation (8) by 2 to obtain the desired result.

\[
\text{Adv} \leq \frac{q_p^2}{|\text{Hash}|} + \frac{q_p^2}{|\text{PUF}|} + 2\max\{ C \cdot q_{\text{send}} \frac{q_s}{2|\text{GM}|} \} \tag{10}
\]

Therefore, we prove Theorem 1. \( \square \)

### 6.4. Scyther Tool Simulation

In this section, we simulate IoT-PUFTAP using the scyther tool [34]. The scyther tool is a push-button tool to verify and analyze various security protocols. It supports unbounded model checking and multi-protocol analysis and provides a graphical user interface (GUI)
to trace security vulnerabilities [49]. We validated the proposed protocol using the scyther tool according to the specifications below:

- Scyther tool checks security attack classes and possible protocol behaviors of the proposed protocol based on a pattern refinement algorithm.
- Scyther tool traces the most efficient and optimal security attacks through the “Find best attacks” setting.
- Scyther tool analyzes the security of the proposed protocol using claim events, including “Secret”, “Alive”, “Weakagree”, “Niagree”, and “Nisynch”.
- To support multiple executions of protocols in the scyther tool, the “Maximum number of run” and “Maximum number of patterns per claim” parameters are set to 5 and 10, respectively.

To simulate IoT-PUFTAP, we write code in “Security Protocol Description Language (SPDL)”. After that, the scyther tool simulates the “Secret”, “Alive”, “Weakagree”, “Niagree”, and “Nisynch” claim events under the DY model. Note that the claim event “Secret” means that the parameter can ensure confidentiality during the authentication phase. The claim event “Alive” denotes that the participants are alive and running the protocol in same session. “Weakagree” can be satisfied when participants actually communicate with a legal participant. “Niagree” can be guaranteed when participants agree on the exchanged parameters. “Nisynch” is the non-injective synchronization claim event, which means that messages are exchanged from legal participants in appropriate sequence. We conducted simulation on a Ubuntu 20.04.2 LTS virtual machine with an Intel Core i3-8100 3.60 GHz CPU and 16.0 GM of RAM.

6.4.1. Scyther Framework

Figure 4 shows the basic framework of the scyther tool. Firstly, we describe the proposed protocol into the scyther GUI according to the syntax of SPDL. Then, the scyther command-line tool performs the security validation using claim events. Finally, the command-line tool outputs the summary reports and trace class graphs. When the protocol satisfies each claim event, the result window displays the “OK” message and “No attacks” comment.

![Figure 4. Basic framework of the scyther tool.](image-url)

6.4.2. SPDL Specification

Figure 5 shows the PUFTAP-IoT written in SPDL code. In PUFTAP-IoT, there are three roles: user $UI$, gateway $GWN$, and sensing device $SDJ$. The user $UI$ sends an authentication request message $\{Msg_1, V_1, T_1, THID_i, PSID_j\}$ to the $GWN$ using the send$_1$ function. Then, $GWN$ receives the message using the recv$_1$ function and sends $\{Msg_2, V_2, V_3, T_2\}$ to the $SDJ$. The $SDJ$ computes the session key $Skey$ and returns $\{Msg_3, V_4, T_3\}$. The $GWN$ transmits $\{Msg_4, V_5, V_6\}$ to the $UI$. 
Figure 5. PUFTAP-IoT written in SPDL code.

6.4.3. Simulation Result

Figure 6 indicates the simulation result of PUFTAP-IoT. The result shows that each role is not exposed to security attacks and ensures the “Secret”, “Alive”, “Weakagree”, “Nisynch” claim events. Therefore, we can demonstrate that PUFTAP-IoT can resist security vulnerabilities and ensure mutual authentication between each participant.

Figure 6. Scyther tool simulation result of PUFTAP-IoT.

7. Efficiency Analysis

In this section, we compare computation cost, communication cost, and security aspects with other relevant papers to prove the efficiency of PUFTAP-IoT.

7.1. Security and Functionality Features Comparison

In this section, we compare PUFTAP-IoT with the related existing protocols in terms of speculating, replay and man-in-the-middle, spoofing, guessing, known session-specific
temporary information (KSSTI), device capture, device impersonation, and session key disclosure attacks and security features such as anonymity, forward secrecy, and secure mutual authentication. Table 3 indicates that the existing protocols do not meet all security requirements. On the other hand, PUFTAP-IoT meets all essential security requirements for communication in IoT environment.

Table 3. Security properties comparison.

| Security Properties       | PUFTAP-IoT | Chunka et al. [16] | Amintoosi et al. [18] | Hajian et al. [27] |
|---------------------------|------------|--------------------|-----------------------|--------------------|
| Replay attack             | o          | o                  | o                     | o                  |
| Man-in-the-middle attack  | o          | o                  | o                     | o                  |
| Guessing attack           | o          | x                  | x                     | o                  |
| Impersonation attack      | o          | x                  | x                     | x                  |
| KSSTI attack              | o          | x                  | o                     | x                  |
| Smart card stolen attack  | o          | o                  | x                     | o                  |
| Device capture attack     | o          | o                  | o                     | x                  |
| Anonymity                 | o          | x                  | x                     | o                  |
| Perfect forward secrecy   | o          | o                  | o                     | o                  |
| Using three factors       | o          | x                  | x                     | o                  |
| Using PUF                 | o          | x                  | x                     | x                  |
| Using honey_list          | o          | x                  | x                     | x                  |
| Secure mutual authentication| o          | x                  | x                     | x                  |

x: insecure against an attack; o: secure against an attack.

7.2. Computation Cost Comparison

We cited [50,51] to compare and analyze computation cost with other authentication protocols. Accordingly, we hypothesized notations and times for cryptographic functions and functional functions as follows: \(T_h\), \(T_{rg}\), \(T_{pu f}\), and \(T_f\) as the execution time needed for hash function, random nonce generation, PUF function, and fuzzy extraction, where \(T_h\), \(T_{rg}\), \(T_{pu f}\), and \(T_f\) are 0.23 ms, 53.9 ms, 12 ms, and 2.68 ms, respectively. Table 4 briefly shows the comparison results.

Table 4. Computation cost of login and authentication phase.

| Protocol                  | User                  | Gateway/Sever         | Sensing Device/Sensor | Total Cost            |
|---------------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Chunka et al. [16]        | \(6T_h + 1T_{rg}\)    | \(5T_h + 1T_{rg}\)   | \(10T_h + 1T_{rg}\)  | \(21T_h + 3T_{rg}\)  |
| Amintoosi et al. [18]     | \(8T_h + 1T_{rg}\)    | \(10T_h + 1T_{rg}\)  | \(10T_h + 1T_{rg}\)  | \(23T_h + 3T_{rg}\)  |
| Hajian et al. [27]        | \(12T_h + 1T_{pu f} + 1T_f\) | \(10T_h\)  | \(5T_h + 1T_{pu f}\) | \(27T_h + 2T_{pu f} + 1T_f\)  |
| Ours                      | \(11T_h + 1T_{pu f} + 1T_f\) | \(16T_h + 1T_{rg}\) | \(7T_h + 1T_{pu f} + 1T_f\) | \(34T_h + 3T_{pu f} + 1T_f\) |

7.3. Communication Cost Comparison

In this section, we compare the cost of communication with other authentication protocols. We assume that each value according to [52]: SHA-1 hash digest, entities’ identity, random nonce, and timestamp is 160, 160, 128, and 32 bits, respectively. Based on this assumption, the communication cost of PUFTAP-IoT is analyzed. Messages \(\{Msg_1, V_j, T_j, THID_j, PSID_j\}\), \(\{Msg_2, V_2, V_3, T_2\}\), \(\{Msg_3, V_4, T_3\}\), and \(\{Msg_4, V_5, V_6\}\) require \((160 + 160 + 32 + 160 + 160 = 592)\), \((160 + 160 + 160 + 32 = 512)\), \((160 + 160 + 32 = 352)\), and \((160 + 160 + 160 = 480)\) bits, respectively. Thus, the total communication cost requires \(592 + 512 + 253 + 480 = 1837\) bits. Table 5 is the analysis of the communication cost consumption of different protocols.

Table 5. Communication cost of login and authentication phase.

| Protocol                  | Total Communication Costs | No. of Messages |
|---------------------------|---------------------------|-----------------|
| Chunka et al. [16]        | 2560 bits                 | 4               |
| Amintoosi et al. [18]     | 1664 bits                 | 4               |
| Hajian et al. [27]        | 2144 bits                 | 5               |
| Ours                      | 1837 bits                 | 4               |
7.4. Results of Comparison

The results of the comparative analysis of PUFTAP-IoT and other papers in terms of security, computation cost, and communication cost are as follows. Although PUFTAP-IoT has a higher computational cost compared with authentication protocols in other papers, the communication cost is similar or lower. Moreover, from a security point of view, PUFTAP-IoT is safe against a variety of attacks and can provide security for guessing, brute-force, and device capture attacks using three-factor, PUF, honey list, etc. Therefore, PUFTAP-IoT can provide very secure services to service users in the IoT environment, even though the computation cost is higher than other authentication protocols.

8. Conclusions

With the development of WSN and IoT, areas using IoT are gradually expanding. Therefore, a secure authentication protocol is required to provide secure IoT services. In this paper, we analyze the security vulnerabilities of two-factor and three-factor authentication protocols in various IoT-based environments. To compensate for the security vulnerabilities of these protocols, we proposed PUFTAP-IoT, which applied PUF, honey list, and three-element technology. We used BAN logic to prove that PUFTAP-IoT can provide secure mutual authentication. We also demonstrated that PUFTAP-IoT can achieve Sean key security through the ROR model. In addition, the scyther simulation tool was used to show that the proposed protocol is safe against various attacks in a wireless network environment. In addition, as a result of the performance analysis of the protocol, it was found that it provides a more secure service in the IoT environment compared with other authentication protocols. In conclusion, PUFTAP-IoT is safer for real-world applications in IoT environments than other related technologies. In the future, based on the proposed protocol, we will analyze the network delay and throughput of the protocol through programming and simulation and apply the developed protocol to the real environment to develop better protocols.

Author Contributions: Conceptualization, J.L.; formal analysis, J.L., D.K. and J.O.; methodology, J.L. and Y.P.; software M.K., D.K. and J.O.; validation, S.Y. and N.-S.J.; writing—original draft, J.L.; writing—review and editing, S.Y. and N.-S.J.; supervision, Y.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by the National Research Foundation of Korea (NRF) funded by the Ministry of Education under grant 2020R1I1A3058605 and in part by the Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean Government (20ZR1300, Core Technology Research on Trust Data Connectome).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Zhang, Y.; Zhao, H.; Xiang, Y.; Huang, X.; Chen, X. A key agreement scheme for smart homes using the secret mismatch problem. *IEEE Internet Things J.*, 2019, 6, 10251–10260. [CrossRef]

2. Rashid, B.; Rehmani, M.H. Applications of wireless sensor networks for urban areas: A survey. *J. Netw. Comput. Appl.*, 2016, 60, 192–219. [CrossRef]

3. Pierce, F.J.; Elliott, T.V. Regional and on-farm wireless sensor networks for agricultural systems in Eastern Washington. *Comput. Electron. Agric.*, 2008, 61, 32–43. [CrossRef]

4. Wazid, M.; Bagga, P.; Das, A.K.; Shetty, S.; Rodrigues, J.J.P.C.; Park, Y. AKM-IoV: Authenticated key management protocol in fog computing-based Internet of vehicles deployment. *IEEE Internet Things J.*, 2019, 6, 8804–8817. [CrossRef]

5. Kwon, D.; Yu, S.; Lee, J.; Son, S.; Park, Y. WSN-SLAP: Secure and lightweight mutual authentication protocol for wireless sensor networks. *Sensors*, 2021, 21, 936. [CrossRef]

6. Fu, X.; Wang, Y.; Yang, Y.; Postolache, O. Analysis on cascading reliability of edge-assisted Internet of Things. *Reliab. Eng. Syst. Saf.*, 2022, 223, 108463. [CrossRef]
7. Fu, X.; Pace, P.; Aloï, G.; Li, W.; Fortino, G. Cascade Failures Analysis of Internet of Things under Global/Local Routing Mode. *IEEE Sensors J.* 2021, 22, 1705–1719. [CrossRef]

8. Das, M.L. Two-factor user authentication in wireless sensor networks. *IEEE Trans. Wirel. Commun.* 2009, 8, 1086–1090. [CrossRef]

9. He, D.; Gao, Y.; Chan, S.; Chen, C.; Bu, J. An enhanced two-factor user authentication scheme in wireless sensor networks. *Ad Hoc Sensor Wirel. Netw.* 2010, 10, 361–371.

10. Kumar, P.; Lee, H.J. Cryptanalysis on two user authentication protocols using smart card for wireless sensor networks. In Proceedings of the Wireless Advanced, London, UK, 20–22 June 2011; pp. 241–245.

11. Turkanović, M.; Brumen, B.; Hölbl, M. A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the Internet of Things notion. *Ad Hoc Netw.* 2014, 20, 96–112. [CrossRef]

12. Amin, R.; Biswas, G.P. A secure light weight scheme for user authentication and key agreement in multi-gateway based wireless sensor networks. *Ad Hoc Netw.* 2016, 36, 58–80. [CrossRef]

13. Wu, F.; Xu, L.; Kumari, S.; Li, X.; Shen, J.; Choo, K.R.; Wazid, M.; Das, A.K. An efficient authentication and key agreement scheme for multi-gateway wireless sensor networks in IoT deployment. *J. Netw. Comput. Appl.* 2017, 81, 72–85. [CrossRef]

14. Shuai, M.; Yu, N.; Wang, H.; Xiong, L. Anonymous authentication scheme for smart home environment with provable security. *Comput. Secur.* 2019, 86, 132–146. [CrossRef]

15. Zou, S.; Cao, Q.; Wang, C.; Huang, Z.; Xu, G. A Robust Two-Factor User Authentication Scheme-Based ECC for Smart Home in IoT. *IEEE Syst. J.* 2021, 16, 4938–4949. [CrossRef]

16. Chunka, C.; Banerjee, S.; Goswami, R.S. An efficient user authentication and session key agreement in wireless sensor network using smart card. *Wirel. Pers. Commun.* 2021, 117, 1361–1385. [CrossRef]

17. Kalra, S.; Sood, S.K. Advanced password based authentication scheme for wireless sensor networks. *J. Inf. Secur. Appl.* 2015, 20, 37–46. [CrossRef]

18. Amintooosi, H.; Nikooghadam, M.; Shojaifar, M.; Kumari, S.; Alazab, M. Slight: A lightweight authentication scheme for smart healthcare services. *Comput. Electr. Eng.* 2022, 99, 107803. [CrossRef]

19. He, D.; Kumar, N.; Chen J.; Lee, C.-C.; Chilamkurthi, N.; Yeo, S.-S. Robust anonymous authentication protocol for health-care applications using wireless medical sensor networks. *Multimedia Syst.* 2015, 21, 49–60. [CrossRef]

20. Wu, F.; Xu, L.; Kumari, S.; Li, X. An improved and anonymous two-factor authentication protocol for health-care applications with wireless medical sensor networks. *Multimedia Syst.* 2017, 23, 195–205. [CrossRef]

21. Wang, C.; Xu, G.; Li, W. A secure and anonymous two-factor authentication protocol in multiserver environment. *Secur. Commun. Netw.* 2018, 2018, 1–15. [CrossRef]

22. Amin, R.; Islam, S.H.; Biswas, G.; Khan, M.K.; Leng, L.; Kumar, N. Design of an anonymity-preserving three-factor authenticated key exchange protocol for wireless sensor networks. *Comput. Netw.* 2016, 101, 42–62. [CrossRef]

23. Jiang, Q.; Zeadally, S.; Ma, J.; He, D. Lightweight three-factor authentication and key agreement protocol for internet-integrated wireless sensor networks. *IEEE Access* 2017, 5, 3376–3392. [CrossRef]

24. Ostad-Sharif, A.; Arshad, H.; Nikooghadam, M.; Abbasinezhad-Mood, D. Three party secure data transmission in IoT networks through design of a lightweight authenticated key agreement scheme. *Future Gener. Comput. Syst.* 2019, 100, 882–892. [CrossRef]

25. Mo, J.; Chen, H. A lightweight secure user authentication and key agreement protocol for wireless sensor networks. *Secur. Commun. Netw.* 2019, 2019, 1–17. [CrossRef]

26. Yu, S.; Park, Y. SLUA-WSN: Secure and lightweight three-factor-based user authentication protocol for wireless sensor networks. *Sensors* 2020, 20, 4143. [CrossRef]

27. Hajar, E.; Erfani, S.H.; Kumari, S. A lightweight authentication and key agreement protocol for heterogeneous IoT with special attention to sensing devices and gateway. *J. Supercomput.* 2022, 1–43. [CrossRef]

28. Aghili, S.F.; Mala, H.; Shojaifar, M.; Peris-Lopez, P. LACO: Lightweight three-factor authentication, access control and ownership transfer scheme for e-health systems in IoT. *Future Gener. Comput. Syst.* 2019, 96, 410–424. [CrossRef]

29. Maes, R. Physically unclonable functions: Properties. In *Physically Unclonable Functions*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 49–80.

30. Juels, A.; Ristenpart, T. Honey encryption: Encryption beyond the brute-force barrier. *IEEE Secur. Privacy* 2014, 12, 59–62. [CrossRef]

31. Juels, A.; Ristenpart, T. Honey encryption: Security beyond the brute-force bound. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, 11–15 May 2014; pp. 293–310.

32. Burrows, M.; Abadi, M.; Needham, R. A logic of authentication. *ACM Trans. Comput. Syst.* 1990, 8, 18–36.

33. Abdalla, M.; Fouque, P.-A.; Pointcheval, D. Password-based authenticated key exchange in the three-party setting. In *Lecture Notes in Computer Science, Proceedings of the 8th International Workshop on Theory and Practice in Public Key Cryptography (PKC’05), Les Diablerets, Switzerland*, 23–26 January 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 65–84.

34. Scyther Tool—Cas Cremer. Available online: https://people.cispa.io/cas.cremers/scyther/ (accessed on 23 July 2022).

35. Lamport, L. Password authentication with insecure communication. *Commun. ACM* 1981, 24, 770–772. [CrossRef]

36. Dolev, D.; Yao, A.C. On the security of public key protocols. *IEEE Trans. Inf. Theory* 1983, 29, 198–208. [CrossRef]

37. Kocher, P.; Jaffe, J.; Jun, B. Differential power analysis. In *Advances in Cryptology*; Springer Science+Business Media: Berlin, Germany; New York, NY, USA, 1999; pp. 388–397.
38. Aman, M.N.; Chua, K.C.; Sikdar, B. Mutual authentication in IoT systems using physical unclonable functions. *IEEE Internet Things J.* 2017, 4, 1327–1340. [CrossRef]
39. Friksen, K.B.; Blantonm, M.; Atallahm, M.J. Robust authentication using physically unclonable functions. In Proceedings of the International Conference on Information Security, Pisa, Italy, 7–9 September 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 262–277.
40. Chatterjee, U.; Chakraborty, R.S.; Mukhopadhyay, D. A PUF-based secure communication protocol for IoT. *ACM Trans. Embedded Comput. Syst.* 2017, 16, 1–25. [CrossRef]
41. Dodis, Y.; Reyzin, L.; Smith, A. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Lecture Notes in Computer Science, *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland*, 2–6 May 2004; Springer: Berlin/Heidelberg, Germany, 2004; pp. 523–540.
42. Juels, A.; Rivest, R.L. Honeywords: Making password cracking detectable. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, Berlin, Germany, 4–8 November 2013; pp. 145–160.
43. Lee, J.; Yu, S.; Park, K.; Park, Y.; Park, Y. Secure three-factor authentication protocol for multi-gateway IoT environments. *Sensors* 2019, 19, 2358. [CrossRef]
44. Son, S.; Lee, J.; Park, Y.; Park, Y.; Das, A.K. Design of blockchain-based lightweight V2I handover authentication protocol for VANET. *IEEE Trans. Netw. Sci. Eng.* 2022, 9, 1346–1358. [CrossRef]
45. Oh, J.; Yu, S.; Lee, J.; Son, S.; Kim, M.; Park, Y. A secure and lightweight authentication protocol for IoT-based smart homes. *Sensors* 2021, 21, 1488. [CrossRef]
46. Yu, S.; Park, Y. A Robust Authentication Protocol for Wireless Medical Sensor Networks Using Blockchain and Physically Unclonable Functions. *IEEE Internet Things J.* 2022. [CrossRef]
47. Kim, M.; Lee, J.; Oh, J.; Park, K.; Park, Y.; Park, K. Blockchain based energy trading scheme for vehicle-to-vehicle using decentralized identifiers. *Appl. Energy* 2022, 322, 119445. [CrossRef]
48. Lee, J.; Kim, G.; Das, A.K.; Park, Y. Secure and efficient honey list-based authentication protocol for vehicular ad hoc networks. *IEEE Trans. Netw. Sci. Eng.* 2021, 8, 2412–2425. [CrossRef]
49. Cremers, C.J. The scyther tool: Verification, falsification, and analysis of security protocols. In Proceedings of the International Conference on Computer Aided Verification, Princeton, NJ, USA, 7–14 July 2008; Springer: Berlin/Heidelberg, Germany, 2008; pp. 414–418.
50. Kilinc, H.H.; Yanik, T. A survey of SIP authentication and key agreement schemes. *IEEE Commun. Surv. Tutor.* 2013, 16, 1005–1023. [CrossRef]
51. Gope, P.; Sikdar, B. Lightweight and privacy-preserving two-factor authentication scheme for IoT devices. *IEEE Internet Things J.* 2019, 6, 580–589. [CrossRef]
52. Banerjee, S.; Odelu, V.; Das, A.K.; Chattopadhyay, S.; Rodrigues, J.J.; Park, Y. Physically secure lightweight anonymous user authentication protocol for internet of things using physically unclonable functions. *IEEE Access* 2019, 7, 85627–85644. [CrossRef]