sensors

MDPI

Article

iAKA-CIoT: An Improved Authentication and Key Agreement Scheme for Cloud Enabled Internet of Things Using Physical Unclonable Function

Kisung Park [1] and Youngho Park [2,*]

[1] Blockchain & Big Data Research Department, Electronics and Telecommunications Research Institute, Daejeon 34129, Korea
[2] School of Electronics Engineering, Kyungpook National University, Daegu 41566, Korea
* Correspondence: parkyh@knu.ac.kr

Abstract: The Internet of Things (IoT) with cloud services are important functionalities in the latest IoT systems for providing various convenient services. These cloud-enabled IoT environments collect, analyze, and monitor surrounding data, resulting in the most effective handling of large amounts of heterogeneous data. In these environments, secure authentication with a key agreement mechanism is essential to ensure user and data privacy when transmitting data between the cloud server and IoT nodes. In this study, we prove that the previous scheme contains various security threats, and hence cannot guarantee essential security requirements. To overcome these security threats, we propose an improved authentication and key agreement scheme for cloud-enabled IoT using PUF. Furthermore, we evaluate its security by performing informal, formal (mathematical), and simulation analyses using the AVISPA tool and ROR model. The performance and security properties of our scheme are subsequently compared with those of other related schemes. The comparison confirms that our scheme is suitable for a practical cloud-enabled IoT environment because it provides a superior security level and is more efficient than contemporary schemes.

Keywords: key establishment; Internet of Things (IoT); physical unclonable function; authentication

Citation: Park, K.; Park, Y. iAKA-CIoT: An Improved Authentication and Key Agreement Scheme for Cloud Enabled Internet of Things Using Physical Unclonable Function. *Sensors* **2022**, *22*, 6264. https://doi.org/10.3390/s22166264

Academic Editors: Akhilesh Tyagi and Himanshu Thapliyal

Received: 22 July 2022
Accepted: 17 August 2022
Published: 20 August 2022

1. Introduction

The Internet of Things (IoT) and advanced communication technologies are opening up a novel networking paradigm that connects various devices to a public network. By 2025, the number of IoT devices and their market size are estimated to increase to approximately 30 billion [1] and 1.6 trillion [2], respectively. With the expansion of IoT infrastructure, IoT-based smart systems can support social networks in various areas, such as telemedicine, finance, smart grids, intelligent transport systems, and businesses. In these environments, IoT devices analyze the surrounding circumstances, collect data, and send them to service providers to provide various IoT services to users. However, IoT devices generally have limited computing power and storage resources, and do not handle a large amount of heterogeneous data.

Cloud-enabled IoT is known to be the most effective system for handling massive amounts of data generated by IoT devices [3]. In cloud-enabled IoT, a cloud server (CS) has sufficient ability to handle massive amounts of data and has the required storage capability for providing services. IoT devices transfer the collected data by monitoring the surrounding circumstances to utilize the storage and computing power of the CS. Thus, the CS collects IoT data and analyzes it to provide cost-effective and convenient services. Cloud-enabled IoT with communication technologies has become extremely important in human life, and thus the need for security and privacy has become essential for users. This is because of the various sensitive information that IoT data contains, such as health, finance, location, and behavior. Moreover, the Internet is an open channel that causes severe

security issues. An adversary can easily forge or intercept data transmitted in an open channel and access user data stored in the CS. Therefore, it is necessary to authenticate entities that attempt to access data.

Numerous authenticated key agreement (AKA) schemes have been presented to guarantee user privacy and data security [4–11]. However, these schemes do not resist physical capture attacks using differential power analysis because IoT devices are not equipped with tamper-proof modules. Although some schemes assume that the devices in their scheme are equipped with tamper-proof modules, they do not present detailed tamper-resistant techniques to prevent physical capture attacks.

A physically unclonable function (PUF) [12] is a novel solution for preventing physical capture attacks targeting devices. The PUF module extracts the unique value corresponding to the inputs from an integrated circuit (IC) that is deployed during the manufacturing process. It has strong and valuable properties, such as tamper-proofing, unpredictable results, and low power consumption, which can be applied to lightweight authentication and identification protocols. In recent years, several PUF-based AKA schemes have been proposed [13–15] to ensure the security of the physical layer. In PUF-based AKA schemes, the PUF module can generate the secret value using challenge-response methods from IC which has different physical characteristics. After performing a fuzzy extractor for correcting the noise of a PUF value, it can be applied for AKA schemes as a secret parameter.

In this paper, we cryptanalyze the security flaws of previous schemes and propose an improved AKA scheme for cloud-enabled IoT using a challenge-response-based PUF, called iAKA-CIoT. Additionally, we analyze its security using formal (mathematical) and informal analyses, and conduct a comparative analysis on iAKA-CIoT and other contemporary schemes. Finally, we simulate our scheme to prove that it is secure against potential attacks.

Motivations and Contributions

The main goal of this study was to propose an improved AKA scheme for IoT using PUF to overcome the security threat of the previous scheme [6,10,11,16,17]. In the previous schemes, an attacker can easily disguise a legal user and compute a correct session key between the participants of the protocol. Moreover, the devices of their scheme can be easily compromised by an attacker using physical capture attack. In our AKA scheme, an adversary cannot compromise the IoT devices because they are protected by PUF modules. We perform informal and formal (mathematical) security analysis, which proves that our scheme meets the essential security requirements and session key security in a threat model. We also conducted a formal simulation analysis using the "automated validation of internet security protocols and applications" (AVISPA) [18] to prove its security and suitability for deployment in an open channel. Finally, the comparative analysis is carried out to evaluate performances and security properties compared with the related schemes.

The organization of this paper is as follows. Sections 2–4 discuss the related works, preliminaries and a review of the scheme proposed by Bhuarya et al., respectively. Section 5 presents the security weaknesses of the aforementioned scheme. In Section 6, we propose an improved AKA scheme for IoT using PUF to overcome the security weaknesses of previous schemes. Subsequently, we present the formal-, informal-security and simulation analyses in Section 7. Section 8 presents a comparative analysis of the related schemes. Finally, the conclusion is furnished in Section 9.

2. Related Works

In the last decade, several studies have been conducted to guarantee user and data privacy in IoT [4–11,13,14]. In 2014, Islam and Biwas [4] proposed a multi-factor authentication method using elliptic curve cryptosystems (ECCs) to provide secure communication for cloud computing. However, Sarvabhatla and Vorugunti [5] showed that the scheme proposed by Islam and Biwas did not prevent offline password guessing, replay, and user impersonation, and subsequently presented an enhanced ECC-based authentication scheme. However, their scheme is inefficient owing to its high computational cost. In 2015,

Kalra and Sood [5] proposed an AKA scheme for cloud-enabled IoT using an ECC. However, in 2017, Kumari et al. [6] showed the security flaws of the Kalra and Sood Schemes and presented an AKA scheme using ECC to resolve these issues. Chaudhry et al. [7] and Chang et al. [8] simultaneously proposed an ECC-based remote user AKA scheme to provide secure mutual AKA. However, in 2019, Mo et al. [9] identified that the scheme proposed by Chaudhry et al. [7] did not resist smart-card loss attacks. Karuppiah et al. [10] proposed a remote AKA for cloud environments. However, Bhuarya et al. [11] pointed out that the aforementioned scheme did not prevent a password-guessing attack and did not achieve user anonymity and secure mutual authentication (SMA). Bhuarya et al. [11] crypt-analyzed the scheme proposed by Kumari et al. [6] and proposed an improved ECC-based AKA for cloud-based IoT. In 2022, Qureshi and Munir [13] also proposed a PUF-based robust authentication and key agreement scheme, and Wang et al. [14] proposed PUF-based authentication scheme with blockchain for wireless sensor network to prevent physical capture attacks. Although many schemes have been proposed, they do not prevent physical capture attacks or have a high communication cost while others simply do not consider them at all, which causes critical security issues.

3. Preliminaries

3.1. Threat Model

We adopted the Dolev–Yao (DY) threat model [19] to evaluate the security of the cryptographic protocols, including the assumptions proposed by Bhuarya et al. According to the adopted model, an adversary can control all messages transmitted in a public network. Additionally, an adversary can easily guess the identity or password but cannot guess them simultaneously in polynomial time. Moreover, an adversary cannot speculate on the secret parameters (secret key, nonce, random number, etc.) in polynomial time because of its large size. Finally, an adversary can obtain data stored in embedded devices that are not equipped with detailed tamper-proof techniques [20–22].

3.2. Physical Unclonable Function

A PUF [12] is a physically unclonable one-way function constructed from a semiconductor as an integrated circuit. PUF is based on challenge-response methods and allows for the identification and authentication of the user. In PUF, c is a challenge and is the input, and its unique response r is illustrated as $r = PUF(c)$. Although the same input is provided, PUF returns an inconsistent output. The PUF has following properties:

(1) Unclonable: There is no function $PUF'(c)$ satisfying $PUF(c) = PUF(c)$, and the probability of duplicating the same result in polynomial time is negligible.
(2) Computable and unpredictable: $PUF(c) = r$ is easily computed; however, it is infeasible to correctly guess r of the $PUF()$ corresponding to c in polynomial time.

3.3. Fuzzy Extractor

The PUF response $r = PUF(c)$ is not perfect because of its susceptibility to surrounding conditions and noise. Therefore, it cannot be utilized in cryptographic protocols as a secret parameter. To correct the noise or errors, we utilize a fuzzy extractor [23,24] that can recover a uniform PUF response r. A fuzzy extractor consists of the two following functions.

(1) Generation function Gen: $Gen(c) = (a, h)$, where c, a, and h are the input value, return value, and auxiliary string, respectively.
(2) Reproduction function Rep: $Rep = (c, h)$, where c and h are the noisy input value and auxiliary string, respectively. Rep can recover the correct a from c and helper string h.

4. Review of Bhuarya et al. Scheme

This section reviews the scheme proposed by Bhuarya et al. [11] to demonstrate their security limitations. The scheme consists of three phases: initial, registration, and login and authentication. The notations used in this paper are presented in Table 1.

Table 1. Notations.

Notation	Description
ED_i	Embedded device
CS	Cloud server
ID_i	Identity of ED_i
PID_i	Pseudo identity of ED_i
ID_{CS}	Identity of the CS
MX_{CS}	Master secret key of the CS
x_{cs-i}	A shared secret value between the CS and ED_i
SK_{cs-ED_i}	A session key between the CS and ED_i
rn_i	Random number chosen by entities
$\oplus$	Bit-wise XOR function
$h(\cdot)$	One-way hash function
$\|\|$	A concatenation operation

4.1. System Setup Phase

This phase is executed by the CS to set up the initial parameters for the system. The CS selects a large prime number p, elliptic curve equation $y^2 = x^3 + ax + b$ over the finite field Z_p, and elements $a, b \in Z_p$, where a, b satisfy the condition $4a^3 + 27b^2 \neq 0$. G and O are the base points of the elliptic curve and the point at infinity, respectively, where $n \cdot G = O$. The CS then generates a secret key MX_{CS} and broadcasts the initial public parameters.

4.2. Registration Phase

In this phase, embedded devices ED_i register themselves with the CS through a secure network to use the CS services. The detailed steps of this phase are as follows.

(1) ED_i chooses the identity ID_i and password PW_i. It then computes $I_i = h(ID_i||PW_i)$ and sends it to the CS via a secure channel.
(2) After receiving $\{I_i\}$, the CS selects a random number rn_s and computes a pseudo identity $PID_i = h(rn_s||ID_{CS}||I_i) \oplus ID_{CS}$ for ED_i. Afterwards, the CS computes the cookie $C_k = h(rn_s||MX_{CS}||E_t||PID_i)$, $C'_k = C_k \cdot G$, $R_i = rn_s \oplus h(MX_{cs}||PID_i)$, $A_i = h(rn_s \oplus h(MX_{CS}||PID_i) \oplus I_i \oplus C'_k)$, and $A'_i = A_i \cdot G$. The CS computes $t_i = R_i \oplus MX_{CS}$, $a_i = A_i \oplus MX_{CS}$, and expiration time $e_t = E_t$, and then stores it with PID_i and sends $\{PID_i, C_k, R_i\}$ to ED_i through a secure channel. If C_k is expired, E_t is updated to E'_t and computes a new cookie $C_k = h(rn_s||MX_{CS}||E'_t||PID_i)$.
(3) Finally, ED_i stores PID_i, R_i, C_k with I_i in a memory.

4.3. Login and Authentication Phase

In this phase, the CS and ED_i authenticate each other, which is executed via a public channel. The detailed steps of this phase are as follows.

(1) A user inputs their identity ID_i and password PW_i, and then ED_i computes $I_i^* = h(ID_i||PW_i)$ and checks if $I_i^* \stackrel{?}{=} I_i$. If it is valid, ED_i chooses a random number rn_1, a current timestamp T_1, and computes $P_1 = rn_1 \cdot G$, $P_2 = h(rn_1 \cdot C'_k)$, $E = PID_i \oplus R_i, K = h(P_1) \oplus PID_i$, and $Y = h(P_1||P_2||K||T_1)$. Then, ED_i sends the login request $\{E, P_1, Y, T_1\}$ to CS.
(2) Upon receiving the login request from ED_i, the CS checks the timestamp validity, computes $PID_i = E \oplus R_i$, and finds PID_i in the database.
(3) If it exists, the CS calculates $rn_s = R_i \oplus h(MX_{CS}||PID_i)$, $C_k = h(rn_s||MX_{CS}||E_t||PID_i)$, $K = h(P_1)$, $P_2 = h(P_1 \cdot C_k)$, and $Y^* = h(P_1||P_2||K||T_1)$, and then verifies that Y^* is equal to Y. If it is correct, the CS chooses a random number rn_2 and a current timestamp T_3 and calculates $P_3 = rn_2 \cdot G$, $P_4 = rn_2 \cdot A'_i$, and $S = h(P_3||P_4||T_3)$. Subsequently, the CS sends the response messages $\{S, P_3, T_3\}$ to ED_i.
(4) After receiving $\{S, P_3, T_3\}$ from CS, ED_i calculates $A_i = h(R_i \oplus I_i \oplus C'_k)$, $P_3^* = P_3 \cdot A_i$ and $S^* = h(P_3^*||P_4||T_3)$, and then verifies that $S^* \stackrel{?}{=} S$ and the timestamp is valid.

If this is correct, ED_i generates the session key $SK = h(rn_1 \cdot P_3||PID_i||T_4||A_i)$ and $V_i = h((rn_1 \cdot C_k')||SK)$, and then sends the messages $\{V_i, T_4\}$ to CS.

(5) The CS checks the validity of the timestamp and generates the session key $SK^* = h(rn_2 \cdot P_1||PID_i||$
$T_4||A_i)$ and $V_i^* = h((P_1 \cdot C_k||SK^*)$. Then, the CS verifies that V_i^* is equal to V_a. If it is, the CS and ED_i successfully authenticate each other.

5. Security Weaknesses of Bhuarya et al.'s Scheme

In this section, we show that the scheme proposed by Bhuarya et al. does not prevent various potential attacks, such as impersonation and man-in-the-middle. Moreover, their scheme has an incorrect authentication mechanism and does not guarantee SMA, which is an essential requirement of an AKA protocol. This analysis was performed under the DY threat model described in Section 3.1.

5.1. Impersonation Attack

Owing to the fact that the scheme does not provide detailed tamper-proof techniques, we suppose that an adversary $\mathcal{A}$ obtains the embedded device ED_i or captures it physically. Subsequently, $\mathcal{A}$ can access the data $\{PID_i, R_i, C_k, I_i\}$ stored in ED_i and perform impersonation attacks using the obtained data as follows:

(1) $\mathcal{A}$ chooses a random number rn_a and a current timestamp T_1, and computes $P_a = rn_a \cdot G$, $P_{a2} = h(rn_a \cdot C_k')$, $E_a = PID_i \oplus R_i$, $K_a = h(P_a) \oplus PID_i$, and $Y_a = h(P_a||P_{a2}||K||T_1)$. Then, $\mathcal{A}$ sends the login request $\{E_a, P_a, Y_a, T_1\}$ to CS.

(2) On receiving the login request from $\mathcal{A}$, the CS checks the timestamp validity, computes $PID_i = E_a \oplus R_i$, and finds PID_i in the database.

(3) If it exists, the CS computes $rn_s = R_i \oplus h(MX_{CS}||PID_i)$, $C_k = h(rn_s||MX_{CS}||E_t||PID_i)$, $K = h(P_a)$, $P_{a2} = h(P_a \cdot C_k)$, and $Y^* = h(P_a||P_{a2}||K||T_1)$. The CS subsequently verifies that Y^* is equal to Y. If it correct, the CS selects a random number rn_2 and a current timestamp T_3, and computes $P_3 = rn_2 \cdot G$, $P_4 = rn_2 \cdot A_i'$, $S = h(P_3||P_4||T_3)$. Afterwards, the CS sends the response messages $\{S, P_3, T_3\}$ to $\mathcal{A}$.

(4) After receiving $\{S, P_3, T_3\}$ from CS, $\mathcal{A}$ computes $A_a = h(R_i \oplus I_i \oplus C_k')$, $P_3^* = P_3 \cdot A_a$, and $S_a^* = h(P_3^*||P_4||T_3)$, and then verifies that $S_a^* \stackrel{?}{=} S$ and timestamp is valid. If it is correct, $\mathcal{A}$ computes the session key $SK = h(rn_a \cdot P_3||PID_i||T_4||A_a)$ and $V_a = h((rn_a \cdot C_k')||SK)$, and then sends the messages $\{V_a, T_4\}$ to CS.

(5) The CS checks the validity of the timestamp and computes the session key $SK^* = h(rn_2 \cdot P_a||PID_i||$
$T_4||A_a)$ and $V_a^* = h((P_a \cdot C_k||SK^*)$. Then, the CS verifies that V_a^* is equal to V_a. If it is, the CS and $\mathcal{A}$ successfully authenticate each other.

$\mathcal{A}$ can successfully generate a valid login request $\{E_a, P_a, Y_a, T_1\}$ and response messages $\{V_a, T_4\}$, showing that the aforementioned scheme does not resist impersonation attacks.

5.2. Man-in-the-Middle Attack

An adversary $\mathcal{A}$ can perform a man-in-the-middle attack as follows:

(1) $\mathcal{A}$ first intercepts the login request $\{E, P_1, Y, T_1\}$ of ED_i, and then chooses a random number rn_a and a current timestamp T_1. $\mathcal{A}$ computes $P_a = rn_a \cdot G$, $P_{a2} = h(rn_a \cdot C_k')$, $E_a = PID_i \oplus R_i$, $K_a = h(P_a) \oplus PID_i$, $Y_a = h(P_a||P_{a2}||K||T_1)$, and sends the login request $\{E_a, P_a, Y_a, T_1\}$ to CS.

(2) $\mathcal{A}$ chooses a random number rn_{a2} and computes $P_{a3} = rn_2 \cdot G$, $P_{a4} = rn_2 \cdot A_i$, and $S_a = h(P_{a3}||P_{a4}||T_3)$, where A_i is obtained by the threat model.

(3) $\mathcal{A}$ intercepts the response messages $\{S, P_3, T_3\}$ of the CS, and then computes $SK = h(rn_a \cdot P_3||PID_i||T_4||A_a)$ and $V_a = h((rn_a \cdot C_k')||SK)$. Finally, $\mathcal{A}$ sends $\{V_a, T_4\}$ and $\{S_a, P_{a3}, T_3\}$ to the CS and ED_i, respectively.

(4) After receiving $\{V_a, T_4\}$ and $\{S_a, P_{a3}, T_3\}$, the CS and ED_i generates the session key using received messages.

$\mathcal{A}$ can successfully establish the session key SK using rn_a and rn_{a2}, which shows that the aforementioned scheme does not prevent man-in-the-middle attacks.

5.3. Correctness of Authentication Mechanism

In the login and authentication phase of the scheme, the CS computes $\{S, P_3, T_3\}$ and sends it to ED_i. Subsequently, ED_i computes $S^* = h(P_3^*||P_4||T_3)$ and verifies that $S \stackrel{?}{=} S^*$ to authenticate the CS. However, ED_i cannot authenticate the CS because S is not equal to S^* as follows:

$$\begin{aligned} S &= h(P_3||P_4||T_3) = h(rn_2 \cdot G||P_4||T_3) \\ S^* &= h(P_3^*||P_4||T_3) = h(P_3 \cdot A_i||P_4||T_3) \\ &= h(rn_2 \cdot G \cdot A_i||P_4||T_3) \\ &= h(rn_2 \cdot G \cdot h(rn_s \oplus h(MX_{cs}||PID_i) \oplus I_i \oplus C_k')||P_4||T_3) \\ &\therefore S \neq S^* \end{aligned}$$

5.4. Secure Mutual Authentication

In Sections 5.1 and 5.2, we proved that the scheme proposed by Bhuarya et al. does not resist impersonation and man-in-the-middle attacks. Moreover, we proved that their scheme contains an incorrect authentication mechanism, which causes the authentication process to be aborted. Therefore, the scheme does not ensure SMA.

6. Proposed Scheme

This section presents an improved AKA scheme for IoT using PUF, which comprises three phases: system setup, registration, and login and authentication. In our scheme, embedded devices are tamper-proof devices that use a PUF to protect the data stored in memory. The embedded devices register their identities with the CS, authenticate them, and establish the session key to each other. After completing the AKA phase, ED_i can use the various services offered by the CS.

6.1. System Setup Phase

The CS sets up the initial parameters related to the elliptic curve, which is identical to the Bhuarya et al. scheme. The CS then generates a secret key MX_{CS} and broadcasts the initial public parameters.

6.2. Embedded Device Registration Phase

This phase is shown in Figure 1, and the detailed steps are as follows:

(1) User selects identity ID_i, password PW_i, challenge c_i, and random number rn_i for ED_i, and then computes $PID_i = h(ID_i||PW_i||rn_i)$, $RPW_i = (ID_i||PW_i) \oplus rn_i$, and $CV_i = c_i \oplus rn_i \oplus h(PID_i||RPW_i||ID_i)$. ED_i calculates $res_i = PUF(c_i)$ and $(a_i, h_i) = Gen(res_i)$ using the PUF and fuzzy extractor. Afterwards, ED_i computes $h_i' = h_i \oplus h(a_i||RPW_i||rn_i)$ and sends $\{PID_i\}$ to the CS via a secure channel.

(2) On receiving the registration request from ED_i, the CS chooses a random number x_{cs-ED_i} for ED_i, and then computes $SID_i = h(PID_i||S_{cs-ED_i}$ and $S_{ED_i} = h(PID_i||rn_{cs}||x_{cs-ED_i})$. The CS stores SID_i with $\{PID_i, S_{ED_i}\}$ in a secure database and sends $\{SID_i, S_{ED_i}\}$ to ED_i through a secure channel.

(3) After receiving $\{SID_i, S_{ED_i}\}$ from the CS, ED_i computes $K_i = S_{ED_i} \oplus h(PID_i||rn_i||a_i)$ and $Ver_i = h(PID_i||S_{ED_i}||rn_i||a_i)$, and stores $\{SID_i, RPW_i, CV_i, h_i', K_i, Ver_i\}$ in memory.

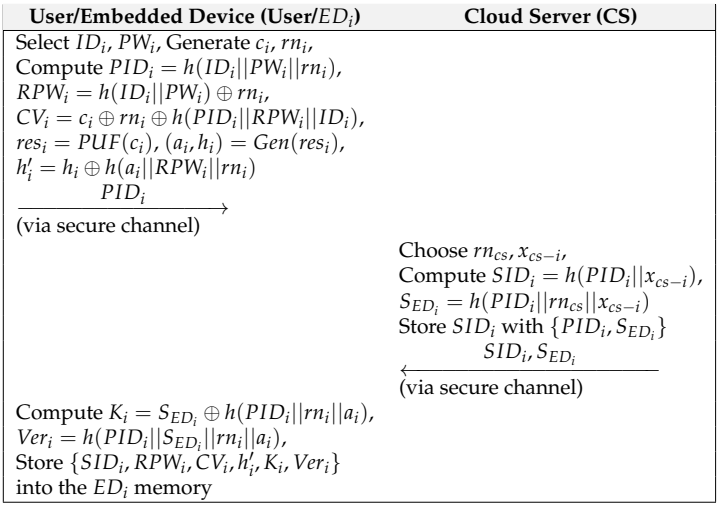

Figure 1. Registration Phase for Our Scheme.

6.3. Authentication and Key Agreement Phase

This phase is shown in Figure 2, and the detailed steps are as follows:

(1) User inputs the identity ID_i with password PW_i to ED_i, and then ED_i computes $h_{ID||PW_i}$, $rn_i = h_{ID||PW_i} \oplus RPW_i$, $PID_i = h(ID_i||PW_i||rn_i)$, $c_i = CV_i \oplus rn_i \oplus h(PID_i||RPW_i||ID_i)$, $res_i = PUF(c_i)$, $h_i = h'_i \oplus h(c_i||PID_i||ID_i)$, $a_i = Rep(res_i, h_i)$, $S_{ED_i} = K_i \oplus h(PID_i||rn_i||a_i)$ and $Ver_i^* = h(PID_i||S_{ED_i}||rn_i||a_i)$. ED_i checks whether $Ver_i^* \stackrel{?}{=} Ver_i$. If it is correct, ED_i chooses a random number rn_1 and a current timestamp T_1; otherwise, it aborts the connection. ED_i computes $R_1 = rn_1 \cdot P$, $M_1 = R_1 \oplus h(PID_i||S_{ED_i}||T_1)$, and $V_1 = h(M_1||R_1||S_{ED_i}||PID_i||ID_{cs}||T_1)$, and then sends $\{SID_i, M_1, V_1, T_1\}$ to the CS.

(2) On receiving the login request from ED_i, the CS checks the timestamp validity and finds $\{PID_i, S_{ED_i}\}$ using SID_i from a secure database. The CS computes $h(PID_i||S_{ED_i}||T_1)$, $R_1 = M_1 \oplus h(PID_i||S_{ED_i}||T_1)$ and $V_1^* = h(M_1||R_1||S_{ED_i}||PID_i||ID_{cs}||T_1)$, and then verifies that V_1^* is equal to V_1.

(3) If it is equal, the CS generates computes a random number rn_2 and a current timestamp T_2; otherwise, aborts the connection. The CS calculates $R_2 = rn_2 \cdot P$, $M_2 = R_2 \oplus h(PID_i||S_{ED_i}||T_2)$, the session key $SK_{cs-ED_i} = rn_2 \cdot R_1$, and $V_2 = h(M_2||R_2||R_1||S_{ED_i}||ID_{cs}||SK_{cs-ED_i})$. After that, the CS sends the response messages $\{M_2, V_2, T_2\}$ to ED_i.

(4) After receiving $\{M_2, V_2, T_2\}$ from the CS, ED_i checks timestamp validity and computes $h(PID_i||S_{ED_i}||T_2)$, $R_2 = M_2 \oplus h(PID_i||S_{ED_i}||T_2)$, the session key $SK_{ED_i-cs} = rn_1 \cdot R_2$, and $V_2^* = h(M_2||R_2||R_1||S_{ED_i}||ID_{cs}||SK_{ED_i-cs})$. Then, ED_i checks whether $V_2^* \stackrel{?}{=} V_2$. If it is verified, ED_i generates a current timestamp T_3 and computes $V_3 = h(SK_{ED_i-cs}||R_1||R_2||S_{ED_i}||T_3$. ED_i sends the verification messages $\{V_3, T_3\}$ to the CS.

(5) On receiving $\{V_3, T_3\}$ to ED_i, the CS computes $V_3^* = h(SK_{cs-ED_i}||R_1||R_2||S_{ED_i}||T_3$ and checks its validity. If it is verified, the CS and ED_i successfully authenticate each other.

User/Embedded Device (User/ED_i)	Cloud Server (CS)
Input ID_i, PW_i,	
Compute $rn_i = h(ID_i \| \| PW_i) \oplus RPW_i$	
$PID_i = h(ID_i \| \| PW_i \| \| rn_i)$,	
$c_i = CV_i \oplus rn_i \oplus h(PID_i \| \| RPW_i \| \| ID_i)$,	
$res_i = PUF(c_i)$, $a_i = Rep(res_i, h_i)$,	
$S_{ED_i} = K_i \oplus h(PID_i \| \| rn_i \| \| a_i)$,	
$Ver_i^* = h(PID_i \| \| S_{ED_i} \| \| rn_i \| \| a_i)$,	
Verify $Ver_i \stackrel{?}{=} Ver_i^*$	
Select rn_i, T_1, Compute	
$R_1 = rn_1 \cdot P$, $M_1 = R_1 \oplus h(PID_i \| \| S_{ED_i} \| \| T_1)$,	
$V_1 = h(M_1 \| \| R_1 \| \| S_{ED_i} \| \| PID_i \| \| T_1)$	
$SID_i, M_1, V_1, T_1 \longrightarrow$	
	Retrieve PID_i, S_{ED_i} using SID_i,
	Compute $R_1 = h(PID_i \| \| S_{ED_i} \| \| T_1)$,
	$V_1^* = h(M_1 \| \| R_1 \| \| S_{ED_i} \| \| PID_i \| \| T_1)$
	Verify $V_1 \stackrel{?}{=} V_1^*$
	Choose rn_2, T_2,
	Compute $R_2 = rn_2 \cdot P$, $M_2 = R_2 \oplus h(PID_i \| \| S_{ED_i} \| \| T_2)$,
	$SK_{cs-ED_i} = rn_2 \cdot R_1$
	$V_2 = h(M_2 \| \| R_2 \| \| R_1 \| \| S_{ED_i} \| \| T_2 \| \| SK_{cs-ED_i})$
	$M_2, V_2, T_2 \longrightarrow$
Compute $R_2 = M_2 \oplus h(PID_i \| \| S_{ED_i} \| \| T_2)$,	
$SK_{ED_i-cs} = rn_1 \cdot R_2$,	
$V_2^* = h(M_2 \| \| R_2 \| \| R_1 \| \| S_{ED_i} \| \| T_2 \| \| SK_{ED_i-cs})$,	
Verify $V_2 \stackrel{?}{=} V_2^*$,	
Choose T_3,	
Compute $V_3 = h(SK_{ED_i-cs} \| \| R_1 \| \| R_2 \| \| S_{ED_i} \| \| T_3)$	
$V_3, T_3 \longrightarrow$	
	Compute $V_3^* = h(SK_{cs-ED_i} \| \| R_1 \| \| R_2 \| \| S_{ED_i} \| \| T_3)$
	Verify $V_3 \stackrel{?}{=} V_3^*$

Figure 2. Login and Authentication Phase for Our Scheme.

7. Security Analysis

In this section, we prove that iAKA-CIoT ensures the session key security (SKS) using the real-or-random (RoR) model [25]. We also perform an informal analysis and simulation analysis using the AVISPA verification tool [18] to demonstrate that our scheme is secure against various potential attacks.

7.1. Formal Security Analysis Using ROR Model

We prove that our scheme achieves SKS using an ROR model-based mathematical formal proof [26–28]. We first discuss the fundamental concept and queries of the ROR model before conducting the formal analysis.

- Participants: Let $\Pi_{ED}^{inst_1}$ and $\Pi_{CS}^{inst_2}$ be the instance $inst_1$ and $inst_2$ of the ED and CS, respectively.
- Accepted state: After completing the message exchanging process, the oracle $\Pi^{i}nst$ transfers a this state. Let the current session identifier be sid_c of Π^{inst} should all the messages be arranged in order.
- Partnering: When $\Pi_{ED}^{inst_1}$ and $\Pi_{CS}^{inst_2}$ have the same sid_c and the accepted state, and each oracle completes the AKA, partners ($\Pi_{ED}^{inst_1}$ and $\Pi_{CS}^{inst_2}$) are defined.
- Freshness: To carry out the formal proof, $\Pi_{ED}^{inst_1}$ and $\Pi_{CS}^{inst_2}$ as instances are deemed fresh if the session key between the ED and CS is presently not revealed to adversary A.
- Attacker: Under our enhanced threat model Section 3.1, A can completely control the public network and send the ROR queries shown in Table 2 to destroy the SKS.
- Semantic Security: A tries to find a correct session key from a random number utilizing the ROR queries. If A correctly guesses a bit c, A wins this game and breaks the semantic security of the scheme. Let $Adv_P = |2Pr[Succ] - 1|$ be the advantage in breaking the session key of scheme $\mathcal{P}$, where Win is the event of the winning game by A.

- Random oracle: All participant entities can use a random oracle as a collision resistant one-way hash function *Hash*.

Table 2. Queries with their descriptions.

Queries	Descriptions
$Execute(\Pi_{ED}^{inst_1}, \Pi_{CS}^{inst_2})$	A can perform an eavesdropping attack using this query under the threat model
$CorruptED(\Pi_{ED}^{inst_1})$	A can perform device stolen attacks using it to retrieve the data stored in ED_i.
$Send(\Pi^{inst}, M)$	A can send messages and receive its response from the oracle P^{inst} using it.
$Test(\Pi^{inst})$	Under this query, A guesses the probabilistic result for an unbiased coin c. When the freshness of the session key SK is established by P^{inst} and A, A guesses SK by sending a *Test* query to the oracle. If $c = c$ or $c = 1$, A obtain an arbitrary number or the correct SK, respectively; otherwise, obtains the NULL ($\perp$).

Now, we prove that our scheme ensures SKS using the following Definitions 1 and 2 and Theorem 1.

Definition 1. *Elliptic curve discrete logarithm problem (ECDLP): Given P and Q, it is computationally intractable to find integer a such that $Q = a \cdot P$, where $a \in Z_p^*$.*

Definition 2. *Elliptic curve decision Diffie–Hellman problem (ECDDHP): Given P, xP, and yP, it is computationally difficult to compute $x \cdot y \cdot P$, where $x, y \in Z_p^*$.*

Theorem 1. *Let an adversary run in polynomial time t as A, and let the advantage of A in breaking the SKS be $Adv_{\mathcal{P}}^{A}$. Then,*

$$Adv_{\mathcal{P}}^{A} \leq \frac{q_h^2}{2|Hash|} + \frac{q_{puf}^2}{2|PUF|} + max\left\{C', q_s^{s'}, \frac{q_s}{2^{lenf}}, \frac{q_s}{2^{lenp}}\right\} \quad (1)$$

where q_h, Hash, and $Adv^{ECDLP}(t)$ is the number of Hash queries, a collision-resistant hash function Hash, and an advantage in breaking ECDLP, respectively.

The formal proofs consisting of four games $G_i (i = 0, 1, 2)$ using the ROR model are as follows:

- Game G_0: A first tosses the coin c and obtains its result at the beginning of this game. Its winning advantage is:

$$Adv_{\mathcal{P}}^{A} = |2.Pr[Succ_0] - 1|, \quad (2)$$

  where *Succ* is the event of A winning the game.
- Game G_1: Under this game, Attacker A performs an eavesdropping attack using the $Execute(\Pi_{ED}^{inst_1}, \Pi_{CS}^{inst_2})$ query. A first intercepts the transmitted messages $\{SID_i, M_1, V_1, T_1\}$, $\{M_2, V_2, T_2\}$, and $\{V_3, T_3\}$ to break the SKS. Then, A executes the $Test(\Pi^t)$ query to guess whether the output of the query is equal to SK or any arbitrary number. However, the winning probability of G_1 does not increase because A does not compute the session key $SK_{ED_i-cs} = rn_1 \cdot rn_2 \cdot P$ without breaking the ECDLP and ECDDHP. Thus, we obtain:

$$Pr[Succ_1] = Pr[Succ_0] \quad (3)$$

- Game G_2: Attacker A performs an active attack using $Send(\Pi^{inst}, M)$ and $Hash$ queries. A attempts to guess the correct message digest collision to mislead a participant entity using several $Hash$ queries. However, in our scheme, all transmitted messages are secured because A does not break the $Hash$ oracle in polynomial time. Moreover, A cannot compute the correct messages without the pseudo-identity PID_i, secret value S_{ED_i}, and tamper-proof value a_i. Thus, according to the birthday paradox [29],

$$|Pr[Succ_1] - Pr[Succ_2| \leq \frac{q_h^2}{2|Hash|} \tag{4}$$

- Game G_3: Attacker A performs a final attack and can obtain $\{SID_i, RPW_i, CV_i, h_i', K_i, Ver_i\}$ stored in the memory of ED_i using $CorruptED(\Pi_{ED}^{inst_1})$. However, A does not compute the valid login request messages $\{SID_i, M_1, V_1, T_1\}$ without knowing $\{ID_i, PW_i, a_i\}$, where $M_1 = R_1 \oplus h(PID_i||S_{ED_i}||T_1)$ and $V_1 = h(M_1||R_1||S_{ED_i}||PID_i||ID_{cs}||T_1)$. Since A does not know ID_i, rn_i, PID_i and a_i, A cannot correctly guess PW_i using $Send(\Pi^{inst}, M)$. Moreover, a_i is only generated by the secure PUF function with a fuzzy extractor, which is defined in Section 3.2, and A does not distinguish between the PUF values and those of the noise without help of fuzzy extractor because the guessing probability of fuzzy extractor values $lenf$ and $lenp$ is approximately $\frac{1}{2^{lenf}}$ and $\frac{1}{2^{lenp}}$, respectively. Therefore, from the PUF simulation and Zipf's law on passwords [30],

$$|Pr[Succ_1] - Pr[Succ_2| \leq \frac{q_{puf}^2}{2|PUF|} + max\left\{C', q_s^{s'}, \frac{q_s}{2^{lenf}}, \frac{q_s}{2^{lenp}}\right\} \tag{5}$$

After simulating all the games (G_0, G_1, G_2, G_3), A attempts to guess the correct c using the $Test$ query. Therefore,

$$Adv_{\mathcal{P},G_3}^A = \frac{1}{2} \tag{6}$$

We can obtain the following results using Equations (2), (3) and (6).

$$\begin{aligned} \frac{1}{2}.Adv_{\mathcal{P}}^A &= |Pr[Succ_0] - \frac{1}{2}| \\ &= |Pr[Succ_1] - \frac{1}{2}| \\ &= |Pr[Succ_1] - Pr[Succ_3]| \end{aligned} \tag{7}$$

Then, we can gain the following results using (5)–(7):

$$\begin{aligned} \frac{1}{2}.Adv_{\mathcal{P}}^A &= |Pr[Succ_1] - Pr[Succ_3]| \\ &\leq |Pr[Succ_1] - Pr[Succ_2]| \\ &+ |Pr[Succ_2] - Pr[Succ_3]| \\ &\leq \frac{q_h^2}{2|Hash|} + \frac{q_{puf}^2}{2|PUF|} + max\left\{C', q_s^{s'}, \frac{q_s}{2^{lenf}}, \frac{q_s}{2^{lenp}}\right\} \end{aligned} \tag{8}$$

Finally, we acquire the final goal by multiplying both sides of (8) by two.

$$Adv_{\mathcal{P}}^A \leq \frac{q_h^2}{2|Hash|} + \frac{q_{puf}^2}{2|PUF|} + max\left\{C', q_s^{s'}, \frac{q_s}{2^{lenf}}, \frac{q_s}{2^{lenp}}\right\} \tag{9}$$

7.2. Informal Security Analysis

This section demonstrates that our scheme is secure against various potential attacks, such as impersonation, man-in-the-middle, replay, physical capture, and offline password guessing. In addition, we demonstrate that it guarantees SMA and anonymity.

7.2.1. Impersonation Attack

Under our threat model, an adversary $\mathcal{A}$ can acquire the exchanged messages in a public network and extract the stored data $\{SID_i, RPW_i, CV_i, h_i', K_i, Ver_i\}$ from the memory of ED_i. However, $\mathcal{A}$ cannot attempt to impersonate a legitimate ED_i because $\mathcal{A}$ does not successfully generate the login request $\{SID_i, M_1, V_1, T_1\}$ and verification messages $\{V_3, T_3\}$ without knowing ID_i, PW_i, S_{ED_i} and a_i. Therefore, iAKA-CIoT is secure against impersonation attacks.

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

When $\mathcal{A}$ tries to perform a man-in-the-middle attack, $\mathcal{A}$ should obtain $\{R_1, R_2\}$ and compute the response messages $\{M_2, V_2\}$ and $\{V_3\}$. However, $\mathcal{A}$ cannot obtain R_1 and R_2 without obtaining $h(PID_i||rn_i||a_i)$. Moreover, all response messages include a timestamp and are masked by a collision-resistant hash function, which makes it difficult to find original messages in polynomial time. Therefore, iAKA-CIoT resists man-in-the-middle and replay attacks.

7.2.3. Physical Capture Attack

After obtaining the data $\{SID_i, RPW_i, CV_i, h_i', K_i, Ver_i\}$ stored in the memory of ED_i' using a physical capture attack, the data do not help compute the session key SK because the PUF response a_i is only generated by ED_i and $\mathcal{A}$ cannot retrieve S_{ED_i} from K_i. Therefore, our scheme protects against physical-capture attacks.

7.2.4. Offline Password Guessing Attack

We assume that $\mathcal{A}$ attempts to guess the password of the user by using intercepted messages and extracting data. $\mathcal{A}$ must know the real identity ID_i, random number rn_i, pseudo identity PID_i and secure parameter S_{ED_i}. However, $\mathcal{A}$ does not know these values because it is masked by a collision-resistant hash function, and $\mathcal{A}$ cannot simultaneously guess two or three parameters in polynomial time. Therefore, iAKA-CIoT is secure against offline password-guessing attacks.

7.2.5. Secure Mutual Authentication and Anonymity

In the AKA phase of our scheme, the CS and ED_i verify the login request $V_1 \stackrel{?}{=} V_1^*$ and response messages $V_2 \stackrel{?}{=} V_2^*$ by using PID_i and S_{ED_i}. According to previous analyses (Sections 7.2.1–7.2.3), $\mathcal{A}$ does not compute verification messages V_1 and V_2 without obtaining $\{PID_i, a_i, ID_i, PW_i, S_{ED_i}\}$. Moreover, in our scheme, the user utilizes the pseudo identity PID_i for the AKA phase, and $\mathcal{A}$ cannot obtain the real identity ID_i of the user. Therefore, our scheme achieves SMA and anonymity.

7.2.6. Denial-of-Service Attack

After receiving exchanged messages between CS and ED_i, they should perform verification procedures to prove validity of these messages $\{Ver_i, V_1, V_3\}$. If it is not valid, the AKA procedure is immediately aborted. It can mitigate denial of service (DoS)/distributed denial of service (DDoS) attacks because $\{Ver_i, V_1, V_3\}$ has freshness which includes timestamp and random number, and can be generated by a legitimate entities.

7.3. Simulation Analysis Using AVISPA Tool

In this section, we discuss the simulation of our scheme by using the AVISPA simulation tool to prove its security [18,31]. AVISPA is a well-known formal simulation tool for evaluating the security of protocols, whereby it verifies that a protocol resists man-in-the-middle and replay attacks. First, we define the security properties of our scheme by using a high-level protocol specification language (HLPSL) [32]. The defined HLPSL code was transformed into an intermediate format using the HLPSL2IF translator. This simulation was executed under the four back-ends model [33]; "on-the-fly model checker"

(OFMC); "tree automata based on a protocol analyzer" (TA4SP); "SAT-based model checker" (SATMC), and "constraint logic-based attack searcher" (CL-AtSE). The procedure of this simulation is shown in Figure 3 and the concept of HLPSL is presented in [31,32].

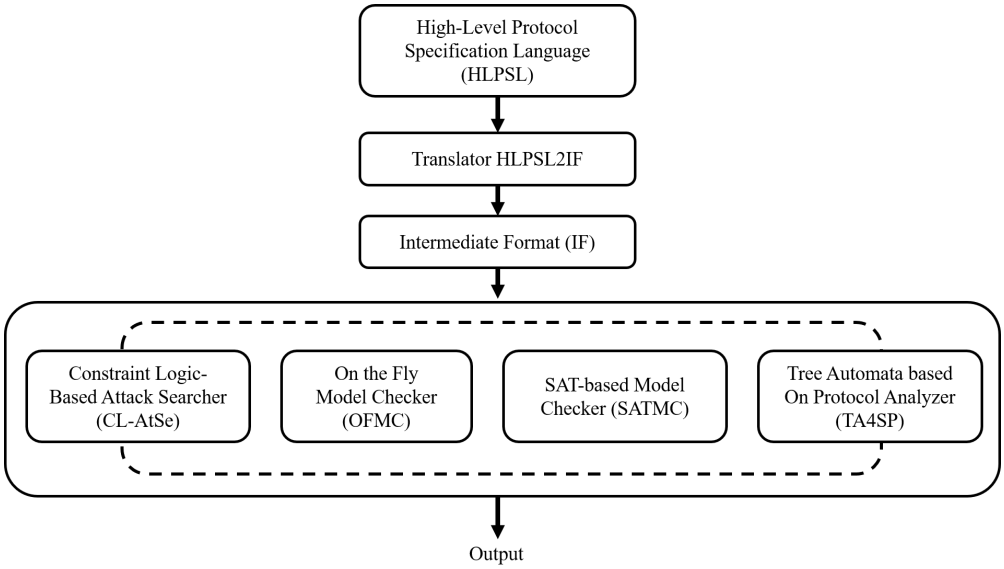

Figure 3. The Procedures of AVISPA Simulation.

7.3.1. HLPSL Specifications

We simulated the defined HLPSL by considering the ED_i and CS AKA phase. There are two basic roles (CS, ED_i), and their HLPSL descriptions are presented in Figures 4 and 5. A session with the environment is defined in Figure 6.

```
%%%%% Cloud server
role cloudsv(US,CS:agent,SKus:symmetric_key,H,PUF,Mul:hash_func,SN,RC:channel(dy))
played_by CS
def=
local State: nat,
IDi,PWi,Ri,Xcs,SIDi,Sedi,Rcs,Rr1,P,T1,IDcs,Rr2,T2,R2,M2,SK,V2,T3:text
const sp3,sp4,cs_us_r1,us_cs_r2: protocol_id
init State:=1
transition

1. State = 1 /\ RC({H(IDi.PWi.Ri')}_SKus) =|>
State':=2
/\ Xcs':=new()
/\ SIDi':=H(H(IDi.PWi.Ri').Xcs')
/\ Sedi':=H(H(IDi.PWi.Ri').Rcs.Xcs')
/\ SN({SIDi'.Sedi'}_SKus)
/\ secret({Sedi',SIDi'},sp3,{US,CS})
/\ secret({Xcs'},sp4,CS)

2. State =2
/\ RC(H(H(IDi.PWi.Ri').Xcs').xor(Mul(Rr1'.P),H(H(IDi.PWi.Ri').H(H(IDi.PWi.Ri').Rcs.Xcs').T1')).H(xor(Mul(Rr1'
.P),H(H(IDi.PWi.Ri').H(H(IDi.PWi.Ri').Rcs.Xcs').T1')).Mul(Rr1'.P).H(H(IDi.PWi.Ri').Rcs.Xcs').H(IDi.PWi.Ri').I
Dcs.T1').T1') =|>
State':=4
/\ Rr2':=new() /\ T2':=new()
/\ R2':=Mul(Rr2'.P)
/\ M2':=xor(R2',H(H(IDi.PWi.Ri').H(H(IDi.PWi.Ri').Rcs.Xcs').T2'))
/\ SK':=Mul(Rr2'.Mul(Rr1'.P))
/\ V2':=H(M2'.R2'.Mul(Rr1'.P).H(H(IDi.PWi.Ri').Rcs.Xcs').IDcs.SK')
/\ SN(M2'.V2'.T2')
/\ witness(US,CS,us_cs_r2,Rr2')
/\ request(CS,US,cs_us_r1,Rr1')

3. State = 4 /\ RC(H(Mul(Rr2'.Mul(Rr1'.P)).Mul(Rr1'.P).Mul(Rr2'.P).H(H(IDi.PWi.Ri').Rcs.Xcs').T3').T3')=|>
State':=6

end role
```

Figure 4. HLPSL Description: CS Role.

```
%%%%% User
role user(US,CS:agent,SKus:symmetric_key,H,PUF,Mul:hash_func,SN,RC:channel(dy))
played_by US
def=
local State: nat,
IDi,PWi,Ci,Ri,PIDi,RPWi,CVi,Resi,Hi,Xcs,Rcs,Rr1,T1,R1,P,M1,V1,IDcs,Rr2,T2,T3,V3:text
const sp1,sp2,cs_us_r1,us_cs_r2: protocol_id
init State:=0
transition

1. State = 0 /\ RC(start) =|>
State':=1
/\ Ri':=new() /\ PIDi':=H(IDi.PWi.Ri')
/\ RPWi':=xor((IDi.PWi),Ri')
/\ CVi':=xor(xor(Ci,Ri'),H(PIDi'.RPWi'.IDi))
/\ Resi':=PUF(Ci)
/\ Hi':=xor(Resi',H(Resi'.RPWi'.Ri'))
/\ SN({PIDi'}_SKus)
/\ secret({IDi,PWi,Ri'},sp1,{US})
/\ secret({PIDi'},sp2,{US,CS})

2. State = 1 /\ RC({H(H(IDi.PWi.Ri').Xcs').H(H(IDi.PWi.Ri').Rcs.Xcs')}_SKus) =|>
State':=3
/\ Rr1':=new() /\ T1':=new()
/\ R1':=Mul(Rr1'.P)
/\ M1':=xor(R1',H(H(IDi.PWi.Ri').H(H(IDi.PWi.Ri').Rcs.Xcs').T1'))
/\ V1':=H(M1'.R1'.H(H(IDi.PWi.Ri').Rcs.Xcs').H(IDi.PWi.Ri').IDcs.T1')
/\ SN(H(H(IDi.PWi.Ri').Xcs').M1'.V1'.T1')
/\ witness(CS,US,cs_us_r1,Rr1')

3. State = 3 /\ RC(xor(Mul(Rr2'.P),H(H(IDi.PWi.Ri').H(H(IDi.PWi.Ri').Rcs.Xcs').T2')).H(xor(Mul(Rr2'.P),
H(H(IDi.PWi.Ri').H(H(IDi.PWi.Ri').Rcs.Xcs').T2')).Mul(Rr2'.P).Mul(Rr1'.P).H(H(IDi.PWi.Ri').Rcs.Xcs').IDcs.Mul(Rr2'.Mul(Rr1'.P))).T2') =|>
State':=5
/\ T3':=new()
/\ V3':=H(Mul(Rr2'.Mul(Rr1'.P)).Mul(Rr1'.P).Mul(Rr2'.P).H(H(IDi.PWi.Ri').Rcs.Xcs').T3')
/\ SN(V3'.T3')
/\ request(US,CS,us_cs_r2,Rr2')

end role
```

Figure 5. HLPSL Description: ED_i role.

```
role session(US,CS:agent,SKus:symmetric_key,H,PUF,Mul:hash_func)
def=
local
SN1,SN2,RC1,RC2:channel(dy)
composition
user(US,CS,SKus,H,PUF,Mul,SN1,RC1)
/\ cloudsv(US,CS,SKus,H,PUF,Mul,SN2,RC2)
end role

role environment()
def=
const
skus:symmetric_key,
h,puf,mul:hash_func,
us,cs:agent,
idi,idcs,sidi:text,
sp1,sp2,sp3,sp4,cs_us_r1,us_cs_r2:protocol_id
intruder_knowledge = {us,cs,idi,idcs,sidi,h,puf,mul}
composition
session(us,cs,skus,h,puf,mul)
/\ session(i,cs,skus,h,puf,mul)
/\ session(us,i,skus,h,puf,mul)
end role

goal
secrecy_of sp1,sp2,sp3,sp4
authentication_on cs_us_r1
authentication_on us_cs_r2
end goal

environment()
```

Figure 6. HLPSL Description: Session and Environment.

7.3.2. Simulation Results

Figures 7 shows the results of the AVISPA simulation, which presents the simulation summary "SAFE". In the CL-AtSe results, the translation time was 0.01 s. For the OFMC results, the search depth was four when 16 nodes were explored in 0.02 s. Therefore, our scheme prevents man-in-the-middle and replay attacks.

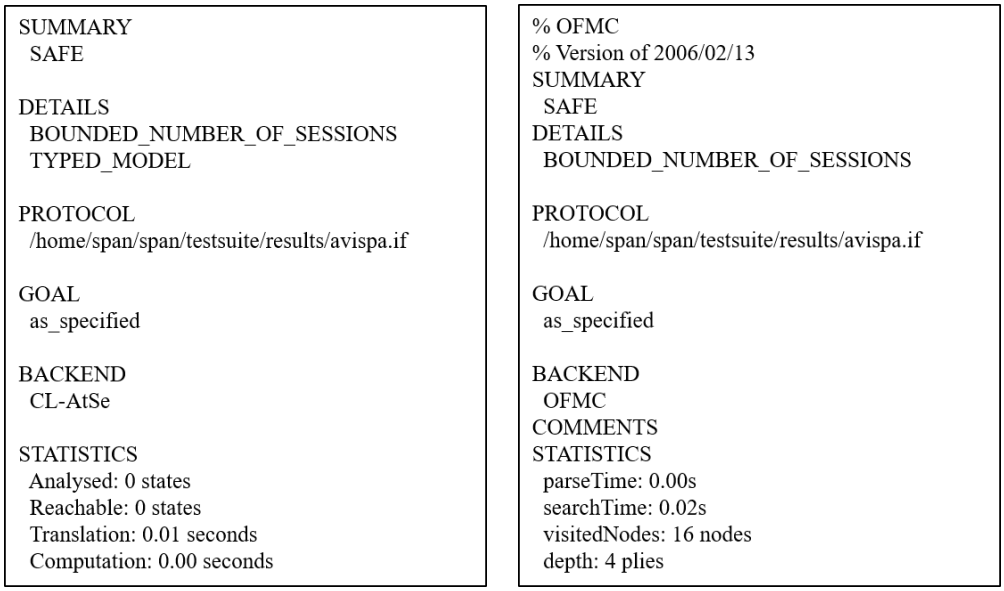

Figure 7. Simulation Result of AVISP Simulation (Summary: SAFE). (**a**) Result of CL-AtSe. (**b**) Result of OFMC.

8. Comparative Analysis

This section presents a comparative analysis of our scheme on the security property, communication, and computation cost with other related schemes [6,10,11,16,17].

8.1. Security Property

We compared the security properties of our scheme with those of the contemporary schemes. Table 3 shows that the previous schemes cannot resist security attacks, achieve anonymity, or SMA. In contrast, we demonstrate that iAKA-CIoT can prevent potential security attacks and guarantee essential security requirements. Therefore, our scheme is more secure than the aforementioned schemes [6,10,11,13,14,16,17].

Table 3. A Comparative Summary: Security Properties.

Properties	[6]	[10]	[16]	[17]	[11]	[13]	[14]	Ours
SP_1	×	×	×	×	×	√	√	√
SP_2	×	×	×	×	×	√	×	√
SP_3	√	×	×	√	√	N/A	√	√
SP_4	×	√	√	√	√	√	√	√
SP_5	×	×	×	×	×	√	×	√
SP_6	√	√	√	×	√	×	×	√
SP_7	×	√	×	×	×	√	×	√
SP_8	√	×	×	×	√	×	√	√

√: supports the security property; ×: does not support the security property; N/A: not applicable; SP_1: physical capture attack; SP_2: impersonation attack; SP_3: offline password guessing attack; SP_4: replay attack; SP_5: mutual authentication; SP_6: user anonymity; SP_7: formal (mathematical) proof; SP_8: formal simulation proof.

We demonstrated that Bhuarya et al. [11] is insecure against physical capture attacks in Section 5. We also proved that other related schemes [6,10,16] does not prevent physical capture attacks to highlight our contributions. The detailed processes of AKA schemes refer to [6,10,16,17].

In [6], an adversary $\mathcal{A}$ can extract the data $\{Pid_i, C'_k\}$ stored in the embedded device ED_i, and then $\mathcal{A}$ selects a random number and computes $P_1 = r_1 \cdot G$, $P_2 = h(r_1 \cdot C'_k)$. Finally $\mathcal{A}$ can successfully generate the login request messages $\{P_1, P_2, Pid_i\}$ without knowing any other information.

In [10], $\mathcal{A}$ can obtain the parameters $\{B_i, N_i, N_t\}$ and $\{W_i, V_i, T_u\}$ from the user's smart card and open channel. Then, $\mathcal{A}$ tries to obtain the PW_i by executing offline password guessing attacks [34]. $\mathcal{A}$ chooses $\{ID_i^*, PW_i^*\}$, and computes $k^* = N_i \oplus h(ID_i^* \oplus PW_i^*)$, $A_i^* = B_i \oplus h(ID_i^*||h(PW_i^*||k^*)$, $(r_i \oplus k)^* = W_i \oplus h(T_u \oplus A_i^*)$ and $V_i^* = h(ID_i^*||A_i^*||W_i||(r_i \oplus k)^*||T_u$. If $V_i$is equal to V_i^*, $\mathcal{A}$ successfully guesses the correct PW_i and can correctly generate valid login request.

In [16], we assumed that $\mathcal{A}$ is a dishonest registered participant in the system. Then, $\mathcal{A}$ can extract the data $\{AID_A, BID_A, r_A\}$ from smart card and can impersonate a legitimate user U_a using it. In their scheme, $\mathcal{A}$ can establish the session key of any legitimate user by betraying a trusted server [7].

Therefore, the aforementioned schemes are insecure against physical capture attacks because they stored secret data as plaintext, which causes critical security issues.

8.2. Computation, Communication and Storage Costs

In this analysis, we consider the AKA phase for protocols. Tables 4–6 compare the computation, communication and storage costs between our scheme and other related schemes, which is shown in Figures 8, 9 and 10, respectively.

Table 4. A Comparative Analysis: Computational Cost.

Scheme	Login Procedure	Authentication Procedure	Total Costs
Kumari et al. [6]	$4T_h + 4T_{em}$	$3T_h + 4T_{em}$	$7T_h + 8T_{em} \approx 0.357 + 22.784 = 23.141$ ms
Karuppiah et al. [10]	$10T_h + 3T_{modexp}$	$4T_h$	$14T_h + 3T_{modexp} \approx 0.714 + 9.438 = 10.152$ ms
Huang et al. [16]	$12T_h + 5T_{em}$	$5T_h + 1T_{em}$	$17T_h + 6T_{em} \approx 0.867 + 17.088 = 17.955$ ms
Jiang et al. [17]	$6T_h + 5T_{modexp}$	$3T_h$	$9T_h + 5T_{modexp} \approx 0.459 + 15.73 = 16.189$ ms
Bhuarya et al. [11]	$10T_h + 5T_{em}$	$6T_h + 4T_{em}$	$16T_h + 9T_{em} \approx 0.816 + 25.632 = 26.448$ ms
Qureshi and Munir [13]	$2T_{aes}$	$2T_{em}$	$2T_{aes} + 2T_{em} \approx 0.024 + 5.696 + 1 = 5.72$ ms
Wang et al. [14]	$6T_h$	$7T_h$	$13T_h \approx 0.663$ ms
Ours	$9T_h + 1T_{em}$	$6T_h + 3T_{em}$	$15T_h + 4T_{em} \approx 0.765 + 11.392 = 12.157$ ms

Table 5. A Comparative Analysis: Communication Cost.

Scheme	Handshake	Total Costs
Kumari et al. [6]	3	1760 bits
Karuppiah et al. [10]	2	2848 bits
Huang et al. [16]	3	1600 bits
Jiang et al. [17]	2	1984 bits
Bhuarya et al. [11]	3	1760 bits
Qureshi and Munir [13]	7	2400 bits
Wang et al. [14]	5	3200 bits
Ours	3	1760 bits

Table 6. A Comparative Analysis: Storage Cost.

Scheme	Total Costs
Kumari et al. [6]	480 bits
Karuppiah et al. [10]	3712 bits
Huang et al. [16]	320 bits
Jiang et al. [17]	640 bits
Bhuarya et al. [11]	640 bits
Qureshi and Munir [13]	800 bits
Wang et al. [14]	960 bits
Ours	960 bits

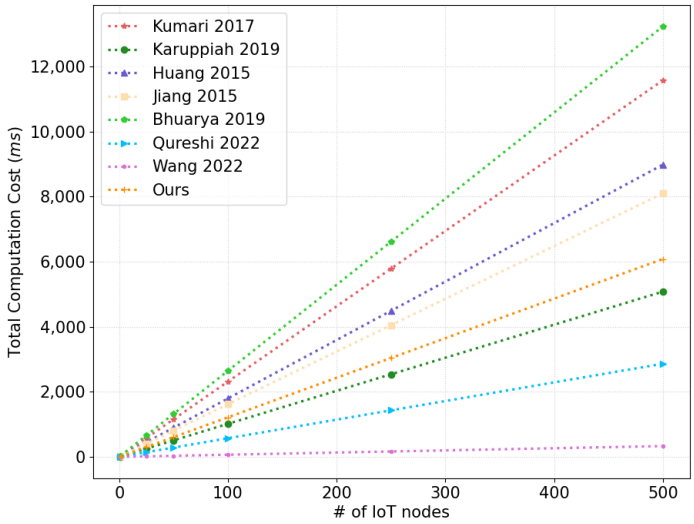

Figure 8. A Comparative Analysis: Computational Cost (Figure).

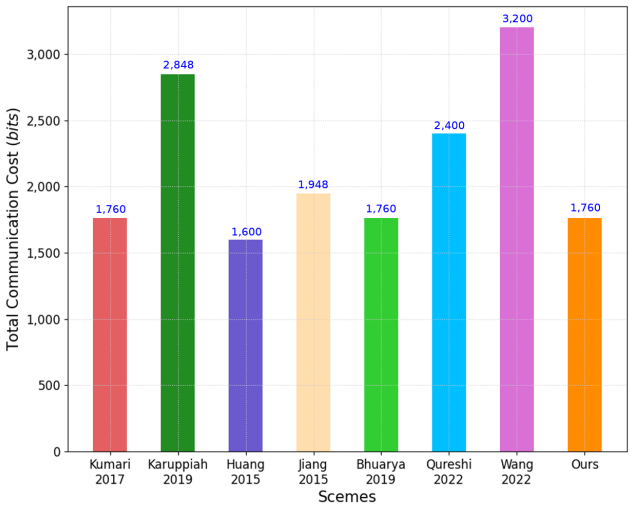

Figure 9. A Comparative Analysis: Communication Cost (Figure).

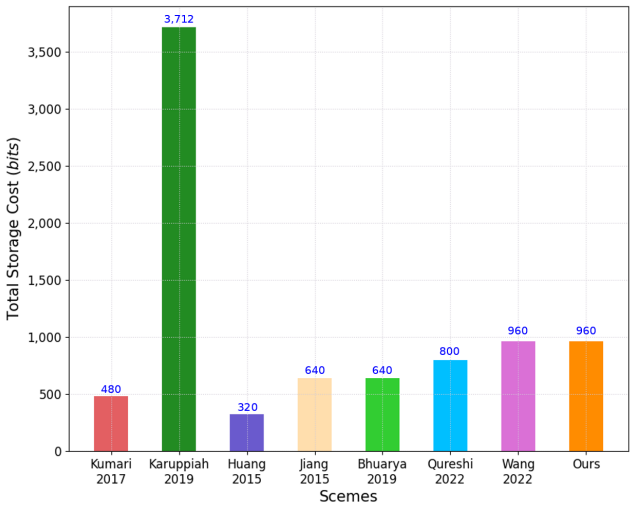

Figure 10. A Comparative Analysis: Storage Cost (Figure).

The computation cost analysis was performed using Raspberry PI 4B with Linux Ubuntu 18.04.4 LTS with 64-bits, 8 GB, and MIRACL library. We utilized the average values for each cryptographic primitive, which was run 100 times to measure its execution cost. To evaluate the computational cost of iAKA-CIoT compared with other schemes, we considered four cryptographic primitives, and their execution costs are presented in Table 7.

Table 7. Execution cost (milliseconds).

Operation	Max. Time (ms)	Min. Time (ms)	Average Time (ms)
T_{em}	2.920	2.766	2.848
T_h	0.142	0.022	0.051
T_{modexp}	4.649	1.746	3.146
T_{aes}	0.021	0.011	0.012

T_{em}: elliptic curve scalar multiplication; T_h: hash function; T_{modexp}: modular exponentiation; T_{aes}: AES-256.

Our scheme requires the total cost $15T_h + 4T_{em} \approx 0.765 + 11.392 = 12.157$ ms, whereas the total cost for other schemes are as follows: that in Kumari et al. [6] required $7T_h + 8T_{em} \approx 0.357 + 22.784 = 23.141$ ms; that in Karuppiah et al. [10] required $0.714 + 9.438 = 10.152$ ms; that in Huang et al. [16] required $17T_h + 6T_{em} \approx 0.867 + 17.088 = 17.955$ ms; that in Jiang et al. [17] required $9T_h + 5T_{modexp} \approx 0.459 + 15.73 = 16.189$ ms; that in Bhuarya et al. [11] required $16T_h + 9T_{em} \approx 0.816 + 25.632 = 26.448$ ms; that in Qureshi and Munir [13] required $2T_{aes} + 2T_{em} \approx 0.024 + 5.696 = 5.72$ ms; that in Wang et al. [14] required $13T_h \approx 0.663$ ms.

For the comparison of communication costs, we defined the message length of the parameters. The one-way hash function, identity, timestamp, and random number are 160 bits. The elliptic curve point and modular exponentiation are 320 and 1024 bits, respectively. In our scheme, the exchanged messages $\{SID_i, M_1, V_1, T_1\}$, $\{M_2, V_2, T_2\}$, and $\{V_3, T_3\}$ needs $160 + 320 + 160 + 160 = 800$ bits, $320 + 160 + 160 = 640$ bits, and $160 + 160 = 320$ bits, respectively. Thus, the total communication cost for our scheme was $800 + 640 + 320 = 1760$. Kumari et al. [6], Karuppiah et al. [10], Huang et al. [16], Jiang et al. [17], Bhuarya et al. [11], Qureshi and Munir [13] and Wang et al. [14] required 1760, 2848, 1600, 1984, 1760, 2400, and 3200, respectively.

The iAKA-CIoT requires a storage cost of 960 bits, whereas the storage cost for other schemes are as follows: that in Kumari et al. [6] required 480 bits; that in Karuppiah et al. [10] required 3712 bits; that in Huang et al. [16] required 320 bits; that in Jiang et al. [17] required 640 bits; that in Bhuarya et al. [11] required 640 bits; that in Qureshi and Munir [13] required 800 bits; and that in Wang et al. [14] required 960 bits.

Section 8.1 shows that the abovementioned schemes [6,10,11,13,14,16,17] are insecure against various attacks such as password guessing, impersonation, replay, and physical capture attacks. Moreover, their schemes do not provide anonymity, a formal proof analysis, or SMA. Although some schemes [13,14] can prevent physical capture attacks, their scheme has security weaknesses [15] or high communication costs. Therefore, our scheme is secure and superior for practical IoT environments.

9. Conclusions

This paper demonstrated that the Bhuarya et al. scheme had an incorrect authentication mechanism, did not resist various attacks, such as impersonation, man-in-the-middle, and physical capture attacks. We also demonstrated that it did not achieve SMA and SKS. We proposed an improved authentication and key agreement scheme for cloud-enabled IoT using PUF to resolve these security flaws. We demonstrated that iAKA-CIoT is secure against impersonation, man-in-the-middle, replay, offline-password guessing, and physical capture attacks, and achieves SMA and anonymity. Formal security proof confirmed that our scheme achieved SKS between the CS and ED using the ROR model. Moreover, we performed a formal simulation analysis using the AVISPA tool and compared it with other

related schemes using the Raspberry PI 4B with MIRACL library. Our scheme also provides superior security properties compared to the aforementioned schemes. Therefore, iAKA-CIoT is suitable for practical cloud-enabled IoT environments because it is more secure and superior than the other related schemes.

Author Contributions: Conceptualization, K.P.; Formal analysis, K.P.; Methodology, K.P.; Project administration, Y.P.; Supervision, Y.P.; Validation, K.P.; Writing—original draft, K.P.; Writing—review & editing, Y.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education(2020R1I1A3058605).

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. Statista: Internet of Things (IoT) and Non-IoT Active Device Connections Worldwide from 2010 to 2025. Available online: https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/ (accessed on 6 May 2022).
2. Statista: Forecast End-User Spending on IoT Solutions Worldwide from 2017 to 2025. Available online: https://www.statista.com/statistics/976313/global-iot-market-size/ (accessed on 26 May 2022).
3. Dizdarević, J.; Carpio, F.; Jukan, A.; Masip-Bruin, X. A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration. *ACM Comput. Surv. (CSUR)* **2019**, *51*, 1–29. [CrossRef]
4. Islam, S.H.; Biswas, G. Dynamic ID-based remote user mutual authentication scheme with smartcard using elliptic curve cryptography. *J. Electron. (China)* **2014**, *31*, 473–488. [CrossRef]
5. Sarvabhatla, M.; Vorugunti, C.S. A secure and robust dynamic ID-based mutual authentication scheme with smart card using elliptic curve cryptography. In Proceedings of the 2015 Seventh International Workshop on Signal Design and its Applications in Communications (IWSDA), Bengaluru, India, 14–18 September 2015.
6. Kumari, S.; Karuppiah, M.; Das, A.K.; Li, X.; Wu, F.; Kumar, N. A secure authentication scheme based on elliptic curve cryptography for IoT and cloud servers. *J. Supercomput.* **2017**, *74*, 6428–6453. [CrossRef]
7. Chaudhry, S.A.; Naqvi, H.; Mahmood, K.; Ahmad, H.F.; Khan, M.K. An improved remote user authentication scheme using elliptic curve cryptography. *Wirel. Pers. Commun.* **2017**, *96*, 5355–5373. [CrossRef]
8. Chang, C.-C.; Wu, H.-L.; Sun, C.-Y. Notes on "Secure authentication scheme for IoT and cloud servers". *Pervasive Mob. Comput.* **2017**, *38*, 275–278. [CrossRef]
9. Mo, J.; Hu, Z.; Chen, H.; Shen, W. An efficient and provably secure anonymous user authentication and key Agreement for mobile cloud computing. *Wirel. Commun. Mob. Comput.* **2019**, *2019*, 4520685. [CrossRef]
10. Karuppiah, M.; Das, A.K.; Li, X.; Kumari, S.; Wu, F.; Chaudhry, S.A.; Niranchana, R. Secure a remote user mutual authentication scheme with key agreements for the cloud environment. *Mob. Netw. Appl.* **2019**, *24*, 1046–1062. [CrossRef]
11. Bhuarya, P.; Chandrakar, P.; Ail, R.; Sharaff, A. An enhanced authentication scheme for Internet of Things and cloud based on elliptic curve cryptography. *Int. J. Commun. Syst.* **2019**, *34*, e4834. [CrossRef]
12. Wallrabenstein, J.R. Practical and secure IoT device authentication using physical unclonable functions. In Proceedings of the 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), Vienna, Austria, 22–24 August 2016.
13. Qureshi, M.A.; Munir, A. PUF-RAKE: A PUF-based robust and lightweight authentication and key establishment protocol. *IEEE Trans. Dependable Secur. Comput.* **2022**, *4*, 2457–2475. [CrossRef]
14. Wang, W.; Chen, Q.; Yin, Z.; Srivastava, G.; Gadekallu, T.R.; Alsolami, F.; Su, C. Blockchain and PUF-based lightweight authentication protocol for wireless medical sensor networks. *IEEE Internet Things J.* **2022**, *9*, 8883–8891. [CrossRef]
15. Yu, S.; Park, Y. A robust authentication protocol for wireless medical sensor networks using blockchain and physically unclonable functions. *IEEE Internet Things J.* 2022, *to be published*. [CrossRef]
16. Huang, B.; Khan, M.K.; Wu, L.; Muhaya, F.T.B.; He, D. An efficient remote user authentication with key agreement scheme using elliptic curve cryptography. *Wirel. Pers. Commun.* **2015**, *85*, 225–240. [CrossRef]
17. Jiang, Q.; Ma, J.; Li, G.; Li, X. Improvement of robust smart-card-based password authentication scheme. *Int. J. Commun. Syst.* **2015**, *28*, 383–393. [CrossRef]
18. AVISPA. Automated Validation of Internet Security Protocols and Applications. Available online: http://people.irisa.fr/Thomas.Genet/span/ (accessed on 8 April 2022).
19. Dolev, D.; Yao, A.C. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [CrossRef]
20. Messerges, T.S.; Dabbish, E.A.; Sloan, R.H. Examining smartcard security under the threat of power analysis attacks. *IEEE Trans. Comput.* **2002**, *51*, 541–552. [CrossRef]

21. Eisenbarth, T.; Kasper, T.; Moradi, A.; Paar, C.; Salmasizadeh, M.; Shalmani, M.T.M. On the power of power analysis in the real world: A complete break of the KEELOQ code-hopping scheme. In *Advances in Cryptology—CRYPTO*; Springer: Berlin/Heidelberg, Germany, 2008.
22. Kocher, P.; Jaffe, J.; Jun, B. Differential power analysis. In *Advances in Cryptology—CRYPTO*; Springer: Berlin/Heidelberg, Germany, 1999.
23. Dodis, Y.; Ostrovsky, R.; Reyzin, L.; Smith, A. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.* **2008**, *38*, 97–139. [CrossRef]
24. Delvaux, J.; Gu, D.; Schellekens, D.; Verbauwhede, I. Helper data algorithms for PUF-based key generation: Overview and analysis. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2015**, *34*, 889–902. [CrossRef]
25. Abdalla, M.; Fouque, P.; Pointcheval, D. Password-based authenticated key exchange in a three-party setting. In Proceedings of the 8th International Workshop on Theory and Practice in Public Key Cryptography (PKC'05), Les Diablerets, Switzerland, 23–26 January 2005.
26. Yu, S.; Lee, J.; Park, K.; Das, A.K.; Park, Y. IoV-SMAP: Secure and efficient message authentication protocol for IoV in a smart city environment. *IEEE Access* **2020**, *8*, 167875–167886. [CrossRef]
27. Park, K.; Lee, J.; Das, A.K.; Park, Y. BPPS:Blockchain-enabled privacy-preserving scheme for demand response management in smart grid environments. *IEEE Trans. Dependable Secur. Comput.* 2022, *Early Acess*. [CrossRef]
28. 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]
29. Boyko, V.; Mackenzie, P.; Patel, S. Provably secure password-authenticated key exchange using Diffie-Hellman. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques Advances in Cryptology (EUROCRYPT), Bruges, Belgium, 14–18 May 2000.
30. Wang, D.; Cheng, H.; Wang, P.; Huang, X.; Jian, G. Zipf's Law in Passwords. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 2776–2791. [CrossRef]
31. Park, K.; Noh, S.; Lee, H.; Das, A.K.; Kim, M.; Park, Y.; Wazid, M. LAKS-NVT: Provably secure and lightweight authentication and key agreement scheme without verification table in medical Internet of Things. *IEEE Access* **2020**, *8*, 119387–119404. [CrossRef]
32. Von Oheimb, D. The high-level protocol specification language, HLPSL developed in the EU project avispa. In Proceedings of the APPSEM 2005 Workshop, Tallinn, Finland, 13–15 September 2005.
33. Vigano, L. Automated Security Protocol Analysis with the AVISPA Tool. *Electron. Notes Theor. Comput. Sci.* **2006**, *155*, 61–68. [CrossRef]
34. Xu, M.; Wang, D.; Wang, Q.; Jia, Q. Understanding security failures of anonymous authentication schemes for cloud environments. *J. Syst. Archit.* **2021**, *118*, 102206–102215. [CrossRef]