




## Article

# Lightweight and Efficient Authentication and Key Distribution Scheme for Cloud-Assisted IoT for Telemedicine

Hyang Jin Lee <sup>1</sup>, Sangjin Kook <sup>1</sup>, Keunok Kim <sup>1</sup> , Jihyeon Ryu <sup>2,\*</sup> , Hakjun Lee <sup>3</sup> , Youngsook Lee <sup>4</sup> and Dongho Won <sup>1</sup>

<sup>1</sup> Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon-si 16419, Republic of Korea; hyangjin.lee@gmail.com (H.J.L.); sangjinkook@gmail.com (S.K.); kimkeunok@gmail.com (K.K.); dhwon@security.re.kr (D.W.)

<sup>2</sup> School of Computer and Information Engineering, Kwangwoon University, Seoul-si 01897, Republic of Korea

<sup>3</sup> School of Electronic Engineering, Kumoh National Institute of Technology, Gumi-si 39177, Republic of Korea; hjlee@kumoh.ac.kr

<sup>4</sup> Department of Computer, Howon University, 64 Impi-myeon, Howondae 3-gil, Gunsan-si 54058, Republic of Korea; ysooklee@howon.ac.kr

\* Correspondence: jhryu@kw.ac.kr

**Abstract:** Medical Internet of Things (IoT) systems are crucial in monitoring the health status of patients. Recently, telemedicine services that manage patients remotely by receiving real-time health information from IoT devices attached to or carried by them have experienced significant growth. A primary concern in medical IoT services is ensuring the security of transmitted information and protecting patient privacy. To address these challenges, various authentication schemes have been proposed. We analyze the authentication scheme by Wang et al. and identified several limitations. Specifically, an attacker can exploit information stored in an IoT device to generate an illegitimate session key. Additionally, despite using a cloud center, the scheme lacks efficiency. To overcome these limitations, we propose an authentication and key distribution scheme that incorporates a physically unclonable function (PUF) and public-key computation. To enhance efficiency, computationally intensive public-key operations are performed exclusively in the cloud center. Furthermore, our scheme addresses privacy concerns by employing a temporary ID for IoT devices used to identify patients. We validate the security of our approach using the formal security analysis tool ProVerif.

**Keywords:** cloud-assisted IoT; lightweight and efficient authentication; IoT for telemedicine



Academic Editors: Raffaele Bruno and Petros S. Bithas

Received: 20 March 2025

Revised: 1 May 2025

Accepted: 2 May 2025

Published: 3 May 2025

**Citation:** Lee, H.J.; Kook, S.; Kim, K.; Ryu, J.; Lee, H.; Lee, Y.; Won, D. Lightweight and Efficient Authentication and Key Distribution Scheme for Cloud-Assisted IoT for Telemedicine. *Sensors* **2025**, *25*, 2894. <https://doi.org/10.3390/s25092894>

**Copyright:** © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Medical Internet of Things (IoT) services facilitate communication between doctors and patients for monitoring and treating health conditions [1,2]. The COVID-19 pandemic has further accelerated the global demand for telemedicine, expanding medical IoT services beyond hospitals [3–5]. While the demand for such services varies by country, they have become particularly important in nations with large elderly populations [6,7], enabling patients to remotely monitor their health and connect with distant hospitals. According to a 2025 report, the global telemedicine market is projected to reach USD 107.52 billion in 2024, USD 121.10 billion in 2025, and USD 432.1 billion by 2032 [8].

Typically, telemedicine services begin with the pre-registration of the patient, attending physician, IoT medical device, and gateway in the hospital's cloud center [9]. The patient's health data are then regularly transmitted to the hospital's cloud center via a gateway (GW)

installed at home or in a nearby common location [10]. The physician reviews these data and provides appropriate responses.

In telemedicine services, IoT devices attached to or carried by patients have significantly fewer resources than medical devices within hospitals [11]. Additionally, to collect, process, and manage large volumes of patient data from diverse locations, a high-performance cloud center is essential. Thus, telemedicine imposes additional requirements beyond those of hospital-based medical IoT environments. However, owing to the sensitivity of transmitted data, ensuring mutual authentication among participants and protecting privacy and anonymity are critical. In recent years, various studies have explored these challenges in medical IoT environments.

More recently, authentication schemes have also been proposed that use cloud servers with unlimited computing resources to process information sent from a large number of IoT devices. These authentication schemes can be used in telemedicine, where a large amount of patient information needs to be processed. In 2023, C. Wang et al. [12] proposed a user authentication scheme for cloud-assisted IoT. However, this paper describes the limitation that this authentication scheme can lead to illegitimate session key exchange if an attacker obtains information stored in the user's smart device or IoT device, which can lead to sensor node impersonation attack. In this paper, we propose a lightweight and efficient authentication scheme that overcomes the limitations of C. Wang et al. [12] and apply it to telemedicine services that used medical IoT devices with limited computing resources, and enhance privacy and anonymity. The key contributions of the proposed scheme are summarized as follows.

- We analyze C. Wang et al.'s work in [12] and find that their proposed authentication scheme is vulnerable to a stolen mobile device attack, which can lead an attacker to generate an illegitimate session key. We also show that the proposed scheme fails to achieve the goal of using cloud centers to increase efficiency by allowing IoT devices to perform public-key cryptographic computation.
- We propose an authentication scheme using cloud centers with unlimited computing resources, similar to the one proposed by C. Wang et al. in [12]. However, unlike C. Wang et al. [12], our scheme improves the efficiency of the authentication scheme by performing public-key cryptographic computation (ECC) only for users (smart devices) and cloud centers that have a certain level of computing resources.
- The proposed scheme minimizes the computation at IoT devices by using a physically unclonable function (PUF). In the scheme, the PUF's challenge–response pair is transmitted through a private channel during the IoT device registration phase, and only the PUF's challenge is transmitted through a public channel during the authentication and key distribution phases, making it resistant to PUF modeling attacks.
- The proposed scheme is resistant to user impersonation attacks, stolen-device attacks, PUF modeling attacks, etc., and provides anonymity and untraceability, mutual authentication, and forward and backward secrecy.
- The proposed scheme is verified using ProVerif, an official security analysis tool. In terms of performance, the proposed scheme achieves 35,436.18% computational savings compared to other recent studies, particularly on low-capacity sensor nodes.

The remainder of this paper is organized as follows: Section 2 discusses related works. Section 3 presents the system model for telemedicine and the attack model. Sections 4 and 5 introduce Wang's scheme and its identified vulnerabilities, respectively. Section 6 details the proposed authentication and key distribution scheme. Sections 7 and 8 provide formal and informal security analyses, along with security and efficiency evaluations. Finally, Section 9 concludes the paper.

## 2. Related Work

With the advancement of IoT technology, numerous recent studies have proposed user authentication methods to ensure secure connections between IoT devices and users.

In 2018, Wazid et al. introduced the user-authenticated key management protocol (UAKMP), a secure and lightweight three-factor remote user authentication scheme for hierarchical IoT networks comprising various nodes, such as gateway nodes, cluster head nodes, and sensing nodes [13]. UAKMP provides password update functionality, ensures anonymity, and supports offline sensing node registration. However, according to Wang et al. [12], it does not satisfy clock synchronization and fails to guarantee forward secrecy.

Several authentication schemes have also been proposed for industrial IoT (IIoT) environments [14,15]. In 2020, Srinivas et al. introduced a user authentication scheme enabling remote users to analyze data in IIoT environments [15]. Their scheme employs biometric information, smart cards, and passwords for authentication, supporting password updates and smart card revocation in cases of loss or theft. However, according to Wang et al. [12], it does not ensure user anonymity. Similarly, in 2020, Yang et al. proposed a dynamic authentication credential framework for IIoT environments [14]. Their scheme achieved faster computation by avoiding public-key cryptography; however, according to Wang et al. [12], it does not guarantee forward secrecy.

Furthermore, in 2020, Wazid et al. proposed a user authentication scheme for smart-home environments [16]. Their scheme achieved high computational efficiency by utilizing only symmetric encryption and decryption. However, according to Wang et al. [12], it fails to ensure forward secrecy.

Recently, authentication schemes targeting a broader range of service environments have been proposed. In 2022, Dai et al. [17] proposed an authentication scheme for multi-gateway sensor networks. The scheme in [17] reduces communication requirements between gateways by registering two frequently visited gateway nodes and improves authentication efficiency using ECC. In the same year, Hu et al. [18] proposed a cloud-assisted authentication scheme based on Chebyshev polynomial encryption for IIoT environments. The scheme in [18] leverages cloud-computing technology with unlimited computing resources to reduce IoT device computation time. Additionally, in [19,20], cloud-computing technology has been utilized for authentication and key sharing in vehicle networks.

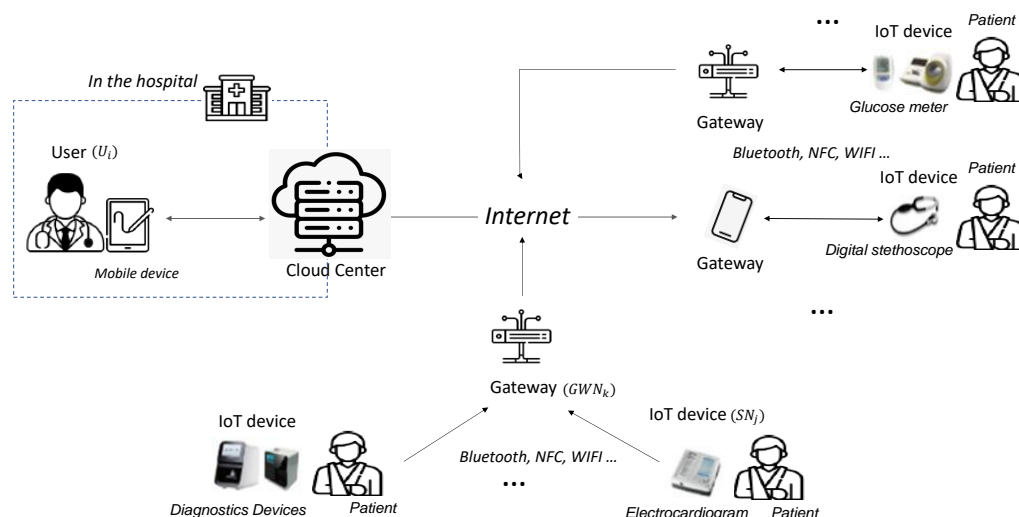
In 2023, Wang et al. [12] proposed a user authentication scheme for IoT using cloud centers. However, ref. [12] is vulnerable to stolen mobile devices and user-impersonation attacks, which may result in illegal session key exchanges. Our proposed scheme addresses the limitations of [12] and proposes a lightweight and efficient authentication and key distribution protocol.

## 3. Preliminaries

In this section, we introduce the system and attack models of the proposed scheme, along with the definitions and properties of the techniques used.

### 3.1. System Model

The proposed scheme targets a telemedicine system model, as shown in Figure 1, and consists of four entities: the user, cloud center, gateway, and IoT device. In this model, a doctor and their mobile device within a hospital are defined as the user, while a medical device that collects patient health information is classified as an IoT device. The gateway transmits data from the IoT device to the cloud center, which serves as a trusted hub for collecting, monitoring, managing, and processing patient information before transmitting it to the user. Each participant is described in detail below.



**Figure 1.** System model for telemedicine.

1. **User ( $U_i$ ):** The user  $U_i$  is a legitimate entity with access to the IoT device. In this scheme,  $U_i$  is a doctor who remotely monitors a patient's health through an IoT device attached to the patient. As the system is designed for telemedicine,  $U_i$  accesses the IoT device through the *Cloud Center*. Communication with the *Cloud Center* occurs via a smart device such as a smartphone or tablet.
2. **IoT Device ( $SN_j$ ):** The IoT device  $SN_j$  collects medical and health information from the patient and transmits it to  $U_i$  through the gateway  $GWN_k$  and the *Cloud Center*.  $SN_j$  is typically a low-power medical device placed in or on the patient's body, such as a glucose meter or digital stethoscope. Therefore, it is assumed that  $SN_j$  cannot perform complex, resource-intensive operations. To enhance security, the proposed method uses physically unclonable function (PUF) technology in  $SN_j$ .
3. **Gateway ( $GWN_k$ ):** The gateway  $GWN_k$  is a trusted entity responsible for transmitting information collected from  $SN_j$  to the *Cloud Center*. While  $GWN_k$  has greater computing power than  $SN_j$ , it faces limitations in handling large volumes of data transmitted by numerous IoT devices in a telemedicine environment.  $GWN_k$  mainly communicates with  $SN_j$  using Bluetooth, NFC, or Wi-Fi and connects to the *Cloud Center* via the Internet.
4. **Cloud Center (*Cloud Center*):** The *Cloud Center* is a trusted entity responsible for processing, managing, storing, and distributing patient information collected from multiple gateways to users. It plays a critical role in telemedicine services, requiring substantial computing resources to handle vast amounts of data from remotely located IoT devices. In the proposed scheme, the cloud center facilitates authentication and key distribution by securely sharing secret parameters during the registration process of  $U_i$ ,  $GWN_k$ , and  $SN_j$ . It is assumed that the *Cloud Center* is securely managed and protected from attackers. In the cloud-assisted IoT for telemedicine services proposed in this paper, the cloud center does not play the role of a simple gateway that supports mutual authentication between users and IoT devices but is proposed as a centralized scheme to perform monitoring and analysis of patient information collected from IoT devices using the computing resources of the cloud center. In particular, in telemedicine services, patients (IoT devices) are often assigned to a doctor in advance, and the doctor has to manage a large number of patients, so a centralized scheme is a more realistic service model than a decentralized or peer-to-peer scheme.

### 3.2. Attacker Model

We assume that the cryptographic primitives used in the proposed scheme are secure and the attacker has the following capabilities:

1. According to the Dolev–Yao model [21–25], the attacker can eavesdrop and intercept the messages transmitted over the public communication channel. Also, the attacker can delete, modify, and reply to any message transmitted over the public channel.
2. The attacker can physically capture the user’s smart device, gateway, or IoT device and obtain all stored information, including certain security parameters.
3. The attacker cannot access information stored in the Cloud Center.
4. The Cloud Center and Gateway each securely hold their long-term secrets, and an attacker cannot obtain them.

### 3.3. Elliptic Curve Cryptography

ECC is a public-key cryptosystem that utilizes the mathematical properties of elliptic curves to achieve secure encryption [26–28]. It provides strong security with a smaller key size compared to traditional public-key cryptosystems such as RSA, offering improved performance. The proposed scheme is based on the elliptic curve discrete logarithm problem (ECDLP), which states that given a point  $P$  and a base point  $G (\in Zp)$  on an elliptic curve  $Zp$ , it is computationally difficult to determine an integer  $d$  such that  $P = d \cdot G$ .

### 3.4. Fuzzy Extractor

A fuzzy extractor is a cryptographic technique designed to generate a stable and secure cryptographic key from biometric data or other noisy inputs [28,29]. As biometric measurements [30], such as fingerprints, may exhibit slight variations between scans, a fuzzy extractor ensures that the same cryptographic key is consistently derived despite these minor differences. It consists of two main procedures: a probabilistic generation procedure (*Gen*) and a deterministic reproduction procedure (*Rep*), characterized as follows:

- *Gen* takes a biometric input  $BIO$  and generates a random string  $\delta$  along with an auxiliary string  $\tau$ ,  $Gen(BIO) = (\delta, \tau)$ .
- *Rep* takes as input  $BIO$ , a similar  $BIO^*$ , and the auxiliary string  $\tau$  generated by *Gen*, and reconstructs the random string  $\delta$ ,  $Rep(BIO^*, \tau) = \delta$ .

### 3.5. Physically Unclonable Function, PUF

A PUF is a hardware-based security primitive that leverages inherent physical variations in semiconductor manufacturing to generate a unique, unclonable cryptographic key [31,32]. These unpredictable manufacturing variations produce distinct responses for each chip, making replication or cloning infeasible. A PUF operates based on the challenge–response pair (CRP) model, where an input challenge generates a unique response based on the physical characteristics of the hardware. When the same challenge is applied to the same chip, it consistently produces the same response, whereas different chips generate entirely different responses.

In this study, we adopt PUFs to achieve both lightweight operation and strong security for sensor nodes. PUFs generate random values based on intrinsic physical characteristics without requiring additional storage or heavy computation, unlike hardware security modules such as TPMs or symmetric key cryptographic approaches. Moreover, compared to HMAC-based schemes or traditional symmetric-key methods, PUFs offer stronger resistance against hardware cloning attacks. Due to these properties, PUFs are particularly suitable for resource-constrained medical IoT environments where computational capability and power consumption are critical concerns. PUFs are assumed to satisfy the following conditions:

1. PUFs are physically unclonable: PUFs are inherently resistant to cloning and modification. They are also highly resistant to environmental factors and aging. An identical device cannot be reproduced by cloning a PUF, and any attempt to change a device containing a PUF will modify its functionality or render it unusable.
2. PUFs exhibit one-way function properties: Given a response  $R_n$  from a specific PUF, it is computationally infeasible to derive the corresponding challenge  $C_n$ .
3. Different PUFs generate different responses: For the same challenge  $C_n$ , two different PUFs (PUF1 and PUF2) will produce distinct responses  $R_{n1}$  and  $R_{n2}$ .
4. A PUF produces a consistent response for the same challenge: A given PUF always outputs the same response  $R_n$  when presented with the same challenge  $C_n$ .
5. Low probability of response collision among multiple PUFs: The likelihood of two different PUFs generating identical responses is extremely low. This property enables PUFs to serve as unique identifiers for distinguishing a large number of IoT devices.

The proposed scheme applies PUFs to the IoT device  $SN_j$ . Using PUF properties,  $SN_j$  uses PUF during the registration phase with the Cloud Center and  $GWN_k$ , as well as during the authentication and session key distribution phase with  $GWN_k$ .

#### 4. Review of Wang et al.'s Scheme

In this section, the scheme introduced by Wang et al. [12] is described. The scheme consists of four participants: the user, cloud center, gateway, and IoT device (sensor node). The user and IoT device perform mutual authentication to share a session key using the cloud center. The scheme is divided into four phases: gateway and IoT device registration, user registration, login, and authentication. The details are as follows:

##### 4.1. Gateway and IoT Device Registration Phase

This phase comprises the registration of the gateway and IoT device with the cloud center. The details are as follows:

##### 4.1.1. Gateway Registration

To register with the cloud center  $CloCen$ , the gateway  $GWN_k$  follows these steps:

1.  $GWN_k$  sends a registration request to  $CloCen$  with  $GID_k$ .
2.  $CloCen$  computes  $X_{G_k} = h(x \parallel GID_k)$  using its secret key  $x$  and sends the message  $GID_k, X_{G_k}$  to  $GWN_k$ .
3.  $GWN_k$  stores  $X_{G_k}$ .

##### 4.1.2. IoT Device Registration

To register with  $GWN_k$ , the IoT device  $S_j$  follows these steps:

1.  $GWN_k$  sends a registration request to  $CloCen$  with  $SID_j$ .
2.  $GWN_k$  validates  $SID_j$ . If it is valid,  $GWN_k$  computes  $X_{S_j} = h(SID_j \parallel x_{G_k})$ , using  $GWN_k$ 's secret key  $x_{G_k}$ .  $GWN_k$  then sends  $SID_j, X_{S_j}$  to  $S_j$ .
3.  $S_j$  keeps  $X_{S_j}$  as its private key.

##### 4.2. User Registration Phase

1.  $U_i$  inputs their identity  $ID_i$ , password  $PW_i$ , and biometric information  $Bio_i$  into their smart mobile device. The device selects a random number  $a'$ , and computes  $Gen(Bio_i) = (\delta_i, \tau_i)$ ,  $RPW_i = h(PW_i \parallel \delta_i \parallel a')$ . The registration request  $\{ID_i, RPW_i\}$  is then sent to  $CloCen$ .



2. *CloCen* selects a timestamp  $Tr_{gi}$  and a random number  $a_i$ , then computes  $k_i = h(ID_i \parallel y \parallel Tr_{gi})$ ,  $B'_i = h(RPW_i \parallel ID_i) \oplus k_i$ . The *CloCen* stores  $\{ID_i, Tr_{gi}, a_i, Honeylist = NULL\}$  in its database and sends  $\{B'_i, B'_i \oplus a_i, Y, P\}$  to  $U_i$ .
3. Upon receiving  $\{B'_i, B'_i \oplus a_i, Y, P\}$ , the smart device selects a new random number  $a$  and calculates the following values. Finally, the smart device stores  $k_i, RPW_i^{new}, A_i, B_i, a_i$  as  $\{A_i, B_i, a, A_i \oplus a_i, \tau_i, Y, P, n0\}$ .
  - $k_i = B'_i \oplus h(RPW_i \parallel ID_i)$
  - $RPW_i^{new} = h(PW_i \parallel \delta_i \parallel a)$
  - $A_i = h(ID_i \parallel RPW_i^{new} \parallel k_i) \bmod n0$
  - $B_i = h(RPW_i^{new} \parallel ID_i) \oplus k_i$
  - $a_i = B'_i \oplus (B'_i \oplus a_i)$

#### 4.3. Login Phase

If  $U_i$  wants to access an IoT device, it initiates a login request to  $GWN_k$  as follows:

1.  $U_i$  enters  $ID_i^*, PW_i^*, Bio_i^*$  into the smart device, which then computes  $\delta_i^* = Rep(Bio_i^*, \tau_i)$ ,  $RPW_i^* = h(PW_i^* \parallel \delta_i^* \parallel a)$ ,  $k_i^* = B_i \oplus RPW_i^*$ ,  $A_i^* = h(ID_i^* \parallel RPW_i^* \parallel k_i^*) \bmod n0$ . The device then compares  $A_i^*$  with  $A_i$  to verify the authenticity of  $U_i$ . If  $A_i^*$  is not equal to  $A_i$ , the login request is rejected.
2. If  $A_i^*$  is equal to  $A_i$ , the smart device selects a random number  $r_i$  and computes  $a_i^*, M1, M2, M3, M4$ , and  $M5$ . The device then sends  $M2, M3, M4, M5$  to *CloCen*.
  - $a_i^* = (A_i \oplus a_i) \oplus A_i$
  - $M1 = r_i \cdot Y, M2 = r_i \cdot P$
  - $M3 = h(M2 \parallel M1) \oplus (ID_i^* \parallel a_i^*)$
  - $M4 = h(M1 \parallel M2 \parallel M3) \oplus SID_j$
  - $M5 = h(k_i^* \parallel ID_i^* \parallel M1 \parallel M2 \parallel SID_j)$

#### 4.4. Authentication Phase

1. To verify the  $U_i$ , *CloCen* computes  $M1' = y \cdot M2$ ,  $ID_i' \parallel a_i' = M3 \oplus h(M2 \parallel M1')$ . It then retrieves  $\{Tr_{gi}, a_i\}$  using  $ID_i'$ . If  $a_i$  is equal to  $a_i'$ , *CloCen* computes  $k_i' = h(ID_i' \parallel y \parallel Tr_{gi})$ ,  $SID_j' = M4 \oplus h(M1 \parallel M2 \parallel M3)$ ,  $M5' = h(k_i' \parallel ID_i' \parallel M1' \parallel M2 \parallel SID_j')$ . It then verifies  $U_i$  via  $M5'$ . If  $M5'$  is equal to  $M5$ , *CloCen* accepts the authenticity of  $U_i$ .
2. *CloCen* inserts  $k_i$  into *Honeylist* if there are fewer than 10 items. If the *Honey – list* exceeds 10,  $U_i$ 's account is suspended until re-registration. *CloCen* then determines the gateway  $GWN_k$  to which  $S_j$  belongs, selects a random number  $r$ , and computes  $X'_{G_k}, M6, M7$ , and  $M8$  as shown below and sends  $M2, M6, M7, M8$  to the gateway node  $GWN_k$ .
  - $X'_{G_k} = h(x \parallel GID_k)$
  - $M6 = h(X'_{G_k} \parallel M2) \oplus r$
  - $M7 = h(M6 \parallel r \parallel X'_{G_k}) \oplus SID_j'$
  - $M8 = h(M2 \parallel M6 \parallel M7 \parallel r \parallel SID_j' \parallel X'_{G_k})$
3.  $GWN_k$  computes  $r' = M6 \oplus h(X_{G_k} \parallel M2)$ ,  $SID_j'' = M7 \oplus h(M6 \parallel r' \parallel X_{G_k})$ ,  $M8' = h(M2 \parallel M6 \parallel M7 \parallel r' \parallel SID_j'' \parallel X_{G_k})$ , and then verifies whether  $M8' = M8$ . If  $M8'$  is equal to  $M8$ ,  $GWN_k$  computes  $X'_{S_j}, M9, M10$  as shown below, and sends  $M2, M9, M10$ .
  - $X'_{S_j} = h(x_{G_k} \parallel SID_j'')$
  - $M9 = h(X'_{S_j} \parallel M2 \oplus r_g)$  (where  $r_g$  is a random number chosen by  $GWN_k$ )
  - $M10 = h(M2 \parallel M9 \parallel r_g \parallel SID_j'' \parallel X'_{S_j})$

4. The IoT device  $S_j$  computes  $r'_g = M9 \oplus h(X_{S_j} \parallel M2)$ ,  $M10' = h(M2 \parallel M9 \parallel r'_g \parallel SID_j \parallel X_{S_j})$ , and compares the values of  $M10'$  and  $M10$ . If  $M10'$  is equal to  $M10$ ,  $S_j$  chooses a random number  $r_j$ , calculates  $M = r_j \cdot M2$ ,  $M11 = r_j \cdot P$ ,  $SK = h(M2 \parallel M11 \parallel M)$ ,  $M12 = h(M2 \parallel M11 \parallel r'_g \parallel X_{S_j} \parallel SID_j)$ , and responds  $\{M11, M12\}$  to the gateway  $GWN_k$ .
5.  $GWN_k$  computes  $M12' = h(M2 \parallel M11 \parallel r_g \parallel X'_{S_j} \parallel SID'_j)$  and compares  $M12'$  with  $M12$  to verify the identity of  $S_j$ . If  $M12'$  is equal to  $M12$ ,  $GWN_k$  calculates  $M13 = h(M11 \parallel M2 \parallel SID''_j \parallel r' \parallel X_{G_k})$  and sends  $\{M11, M13\}$  to  $CloCen$ .
6.  $CloCen$  computes  $M13' = h(M11 \parallel M2 \parallel SID''_j \parallel r \parallel X'_{G_k})$  to test the identity of  $GWN_k$ . If  $M13'$  is equal to  $M13$ ,  $CloCen$  computes  $M14 = h(M1' \parallel M2 \parallel ID'_i \parallel SID''_j \parallel k'_i \parallel M11)$ , and then returns  $\{M11, M14\}$  to  $U_i$ .
7. Having obtained  $CloCen$ 's reply  $\{M11, M14\}$ , the smart mobile device computes  $M14^* = h(M1 \parallel M2 \parallel ID_i \parallel SID_j \parallel k \parallel M11)$ . If  $M14^*$  is equal to  $M14$ ,  $U_i$  accepts  $SK = h(M2 \parallel M11 \parallel r_i \cdot M11)$  as his session key shared with  $S_j$ , and the authentication process finishes successfully.

## 5. Limitations of Wang et al.'s Scheme

We identified several critical limitations in Wang et al. [12]'s scheme. A detailed analysis of these vulnerabilities is provided below.

### 5.1. Stolen Mobile Device Attack

In Wang et al. [12]'s scheme, only legitimate users are allowed to log in using  $PW_i$  and  $Bio_i$  during the user login phase. However, after user  $U_i$  logs in, the messages  $M1$ ,  $M2$ ,  $M3$ ,  $M4$ , and  $M5$ , which authenticate the user to  $CloCen$ , contain only information stored in the smart device. Although Wang et al. [12]'s scheme includes  $ID_i$  as a credential known only to the user, in general, user IDs in various Internet protocols are publicly accessible for identification, or they can often be computed in polynomial time by an attacker. This implies that if an attacker gains access to the user's smart device and retrieves its stored information, they can successfully authenticate with  $CloCen$  without requiring the login phase. The following is a detailed attack scenario.

- {In step 2 of Section 4.3 Login Phase, the attacker computes  $a_i$  using the information  $A_i$  and  $A_i \oplus a_i$  stored in the stolen mobile device, and generates a random number  $r_a$ . In addition, the attacker can find the  $ID_i$  of the legitimate user in polynomial time.
- The attacker then generates the following message using the previously calculated  $a_i$ ,  $ID_i$ , and  $r_a$  to impersonate the legitimate user, and sends it to  $CloCen$ :
  - $M1_a = r_a \cdot Y$  (where  $Y$  is ECC public key of  $CloCen$ )
  - $M2_a = r_a \cdot P$  (where  $P$  is ECC base point of  $CloCen$ )
  - $M3_a = h(M2_a \parallel M1_a) \oplus (ID_i \parallel a_i)$
  - $M4_a = h(M1_a \parallel M2_a \parallel M3_a) \oplus SID_j$
  - $M5_a = h(k_i \parallel ID_i \parallel M1_a \parallel M2_a \parallel SID_j)$
- In step 2 of Section 4.4 Authentication Phase,  $CloCen$  verifies  $M1_a = y \cdot M2_a$  using its private key  $y$ . If  $M1_a$  is equal to  $y \cdot M2_a$ , then it finds  $ID_i$  through  $ID_i \parallel a_i = M3_a \oplus h(M2_a \parallel M1_a)$ .
- The attacker can be authenticated as a legitimate user because all subsequent steps use the information stored inside  $CloCen$  regarding  $ID_i$  to perform subsequent authentication procedures.



### 5.2. Illegitimate Session Key Exchange

If the attacker knows  $SID_j$  and obtains  $X_{S_j}$  from a captured IoT device, they can manipulate the session key exchange, as shown below.

- In step 4 of Section 4.4 Authentication Phase, the attacker chooses a random number  $r_{ja}$  and calculates  $M_a$ ,  $M11_a$  using the transmitted message  $M2$  as follows.
  - $M_a = r_{ja} \cdot M2$
  - $M11_a = r_{ja} \cdot P$  (where  $P$  is the ECC base point of  $CloCen$ )
- The attacker uses the previously computed  $M_a$ ,  $M11_a$ , and the transmitted message  $M2$  to generate an illegitimate session key  $SK_a$ . The attacker also generates additional information  $M12_a$  so that the gateway and user can generate the same session key, and sends  $M11_a, M12_a$  to  $GWN_k$ .
  - $SK_a = h(M2 \parallel M11_a \parallel M_a)$
  - $M12_a = h(M2 \parallel M11_a \parallel r'_g \parallel X_{S_j} \parallel SID_j)$  (where  $r'_g$  is a random number generated by the gateway in step 3 of Section 4.4 Authentication Phase, and  $X_{S_j}$  is  $h(SID_j \parallel x_{G_k})$ . And  $x_{G_k}$  is the long-term secret of  $GWN_k$ .)
- In step 5 of Section 4.4 Authentication Phase,  $GWN_k$  checks if  $M12_a$  is equal to  $h(M2 \parallel M11_a \parallel r_g \parallel X_{S_j} \parallel SID_j)$  using  $M11_a$  and  $M12_a$  sent by the attacker and the information  $M2, r_g, X_{S_j}$ , and  $SID_j$  that it knows, where  $X_{S_j}$  is  $h(SID_j \parallel x_{G_k})$ . If it is equal,  $GWN_k$  computes  $M13_a = h(M11_a \parallel M2 \parallel SID_j \parallel r \parallel X_{G_k})$  and sends  $\{M11_a, M13_a\}$  to  $CloCen$ .
- In step 6 of Section 4.4 Authentication Phase,  $CloCen$  checks whether  $M13_a$  is equal to  $h(M11_a \parallel M2 \parallel SID_j \parallel r \parallel X_{G_k})$  using  $M11_a$  and  $M13_a$  sent from  $GWN_k$  and the information  $M2, SID_j, r$ , and  $X_{G_k}$  that it knows, where  $X_{G_k}$  is  $h(x \parallel GID_k)$ , and  $x$  is the secret long-term key of  $CloCen$ . If it is equal,  $CloCen$  computes  $M14_a = h(M1 \parallel M2 \parallel ID_i \parallel SID_j \parallel k_i \parallel M11_a)$  and sends  $\{M11_a, M14_a\}$ .
- In step 7 of Section 4.4 Authentication Phase,  $U_i$  uses  $M11_a$  and  $M14_a$  sent by  $CloCen$  and the information  $M1, M2, ID_i, SID_j, k_i$ , and  $M11_a$ , knowing to check whether  $M14_a$  matches  $h(M1 \parallel M2 \parallel ID_i \parallel SID_j \parallel k_i \parallel M11_a)$ . If it matches, the user generates a session key  $SK_a$  as follows.
  - $SK_a = h(M2 \parallel M11_a \parallel r_i \cdot M11_a)$  (where,  $r_i \cdot M11_a = r_{ja} \cdot M2 = M_a$ )

Through this process, the illegally generated session key  $SK_a$  is exchanged between  $U_i$  and  $SID_j$  by the attacker.

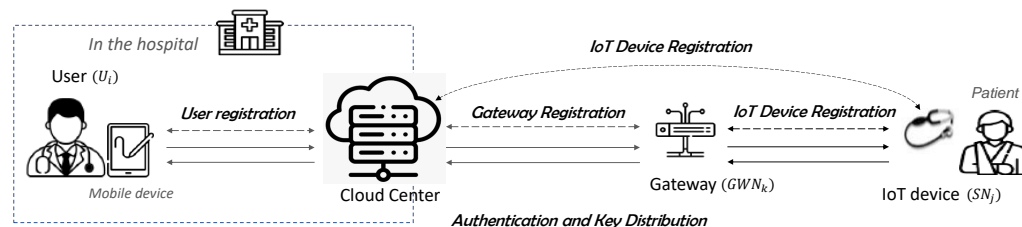
### 5.3. Inefficiency

A key contribution of Wang et al.'s scheme is the use of cloud-computing technology to overcome the computational and storage limitations of gateways and IoT devices. Reducing the computational load on the gateway or IoT devices significantly impacts the overall efficiency of the authentication scheme. However, in Wang et al.'s scheme, both the gateway and IoT devices perform computationally intensive ECC operations. Therefore, it is unclear whether the scheme effectively reduces the computational load on these resource-constrained devices compared to other authentication schemes that rely on symmetric key algorithms, despite the incorporation of cloud-computing technology.

## 6. Proposed Scheme

In this section, we propose a secure and efficient authentication and key distribution scheme to establish a secure telemedicine environment by addressing the vulnerabilities in Wang et al.'s scheme [12] described in Section 5 and integrating a cloud-assisted IoT framework into the medical domain.

The proposed scheme is applied to the system model shown in Figure 2. In this model, a doctor in a hospital serves as the user ( $U_i$ ), while a medical IoT device (sensor node,  $SN_j$ ) attached to or carried by the patient monitors the patient's condition outside the hospital. The patient's health data are transmitted from the IoT device to the cloud center through a gateway ( $GWN_k$ ), enabling the user to access the transmitted information, assess the patient's condition, and provide appropriate prescriptions remotely.



**Figure 2.** Proposed authentication scheme.

To use the telemedicine service, the patient must first register the user  $U_i$ , who is the doctor in charge, along with the medical IoT device  $SN_j$ , and the gateway  $GWN_k$  with the *Cloud Center*. Therefore, in our proposed scheme, it is assumed that the cloud center is aware of the user  $UID_i$  and gateway  $GID_k$  based on the IoT device  $SID_j$  before the registration phase. In addition, in the proposed scheme, the user  $U_i$  establishes a session key with the *Cloud Center*, while  $GID_k$  and  $SID_j$  share their respective session keys with the *Cloud Center* through mutual authentication. Consequently, direct authentication between  $UID_i$ ,  $GID_k$ , and  $SID_j$  is not required. The notation used in the proposed scheme is provided in Table 1.

**Table 1.** Notation.

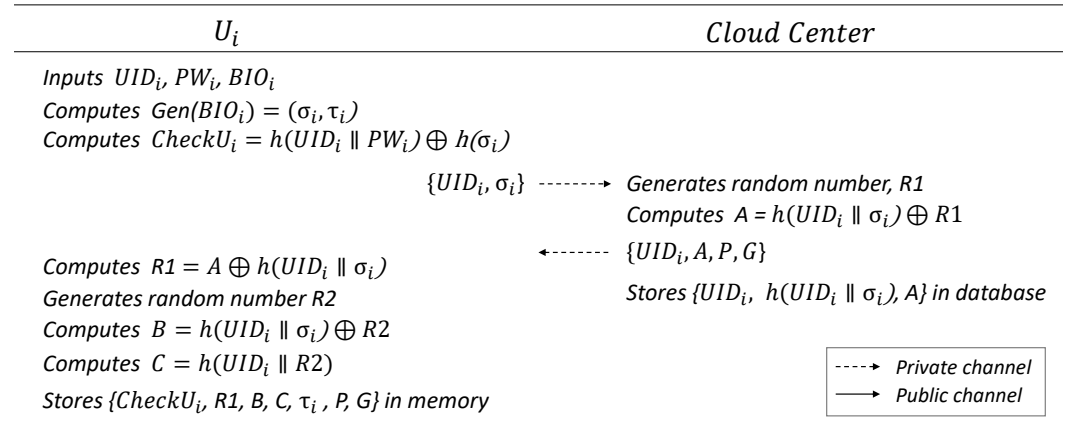
Notation	Description
$U_i, SN_j, GWN_k$	$i$ -th user, $j$ -th IoT device (sensor), $k$ -th medical gateway
$UID_i, PW_i$	$i$ -th user's identity and password
$SID_j, GID_k$	$j$ -th IoT device's identity, $k$ -th medical gateway's identity
$T.SID_j$	Temporary identity of $j$ -th IoT device (sensor)
$SC$	Long-term secret information of cloud center
$SC_k$	Long-term secret information of $k$ -th medical gateway
$P, d, G$	ECC-based public key, private key, base point of cloud center
$(Cn_c, Rn_c)$	Challenge–response pair of PUF for cloud center
$(Cn_g, Rn_g)$	Challenge–response pair of PUF for gateway
$SK$	Session key
$R_1, R_2, \dots$	Random number
$T_1, T_2, \dots$	Time stamp

### 6.1. User Registration Phase

In this phase,  $U_i$  enters a user ID  $UID_i$ , password  $PW_i$ , and biometric information  $BIO_i$  on the mobile device to initiate a legitimate login and register validation information with the *Cloud Center* for authentication. The details are illustrated in Figure 3.

1.  $U_i$  inputs  $UID_i$ ,  $PW_i$ , and  $BIO_i$  into the mobile device, computes  $Gen(BIO_i) = (\delta_i, \tau_i)$ , and generates  $CheckU_i = h(UID_i \parallel PW_i \parallel \tau_i) \oplus h(\delta_i)$  as additional validation information to prove legitimacy.  $U_i$  then sends  $\{UID_i, \delta_i\}$  to the *Cloud Center* through a secure private channel.
2. The *Cloud Center* generates a random number  $R1$  and computes  $A = h(UID_i \parallel \delta_i)$ . It then sends  $\{UID_i, A, P, G\}$  to  $U_i$ , where  $G$  is the base point of the shared ECC

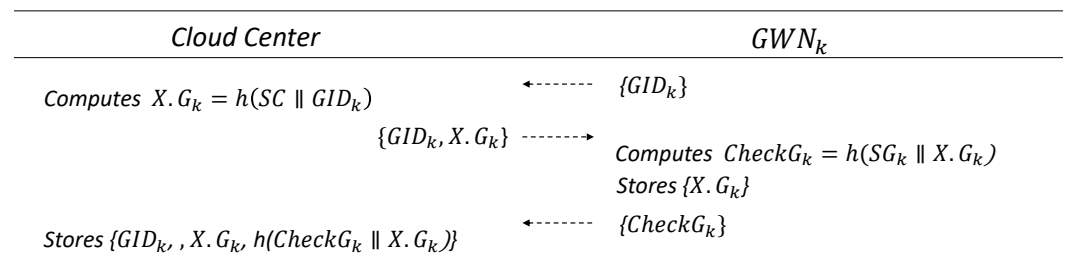
- algorithm, and  $P$  is the public key of the *Cloud Center*. After registering  $U_i$ , the *Cloud Center* stores  $\{UID_i, h(UID_i / parallel \delta_i), A\}$  in its database.
- $U_i$  computes  $R1 = A \oplus h(UID_i \parallel \delta_i)$  and generates a random number  $R2$ . It then computes  $B = h(UID_i \parallel \delta_i) \oplus R2$ ,  $C = hH(UID_i \parallel R2)$  and stores  $\{CheckU_i, R1, B, C, \tau_i, P, G\}$  in the mobile device's memory.



**Figure 3.** User registration phase.

#### 6.2. Gateway Registration Phase

In this phase,  $GWN_k$  and the *Cloud Center* complete the registration process using their respective long-term secrets,  $S_C$  and  $S_{G_k}$ , which are securely held by  $GWN_k$  and *Cloud Center*, respectively. It is assumed that the *Cloud Center* is already aware of  $GID_k$  and  $SID_j$ , through prior offline registration. The details are illustrated in Figure 4.

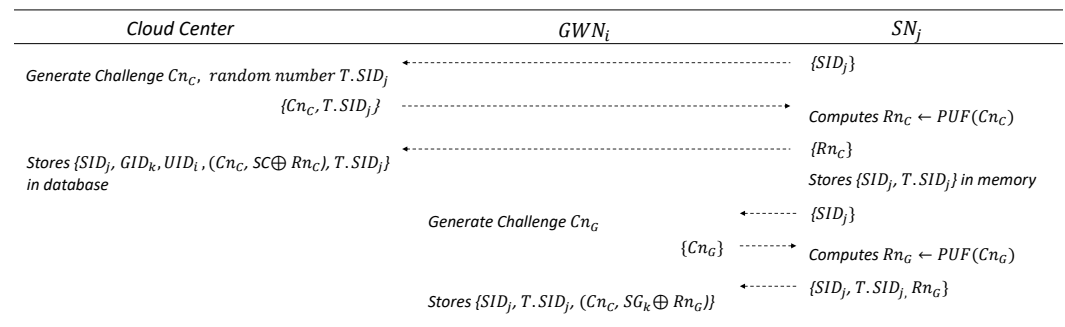


**Figure 4.** Gateway registration phase.

- $GWN_k$  sends a registration request message containing  $GID_k$  to *Cloud Center*.
- Cloud Center* computes  $X_{G_k} = h(S_C \parallel GID_k)$  and sends  $\{GID_k, X_{G_k}\}$  to  $GWN_k$ .
- $GWN_k$  computes  $CheckG_k = h(S_{G_k} \parallel X_{G_k})$  and sends it to the cloud center. It then stores  $X_{G_k}$  in memory.
- $GWN_k$  stores  $\{GID_k, X_{G_k}, h(CheckG_k \parallel X_{G_k})\}$  in its database.

#### 6.3. IoT Device Registration Phase

In this phase,  $SN_j$  registers with *Cloud Center* and  $GWN_k$ , respectively. The details are illustrated in Figure 5.



**Figure 5.** IoT device (sensor) registration phase.

### 6.3.1. IoT Device Registration to Cloud Center

1.  $SN_j$  sends a registration request message containing  $SID_j$  to *Cloud Center*.
2. *Cloud Center* identifies  $SN_j$  using  $SID_j$  and generates a challenge  $Cn_C$  along with a random number  $T.SID_j$ , where  $T.SID_j$  is a randomly generated temporary ID for  $SID_j$ . The *Cloud Center* then sends  $\{Cn_C, T.SID_j\}$  to  $SN_j$ .
3.  $SN_j$  generates a response  $Rn_C$  for  $Cn_C$  and sends it to the *Cloud Center*. After sending the response,  $SN_j$  stores  $\{SID_j, T.SID_j\}$ .

### 6.3.2. IoT Device Registration to Gateway

1.  $SN_j$  sends a registration request message containing  $SID_j$  to  $GWN_k$ .
2.  $GWN_k$  generates a challenge  $Cn_G$  and sends it to  $SN_j$ .
3.  $SN_j$  computes the response  $Rn_G \leftarrow PUF(Cn_G)$  and sends  $\{SID_j, T.SID_j, Rn_G\}$  to  $GWN_k$ .
4.  $GWN_k$  stores  $\{SID_j, T.SID_j, (Cn_G, SG_k \oplus Rn_G)\}$  in its database.

## 6.4. Authentication and Key Distribution Phase

In the proposed scheme, authentication and key distribution among the four participants are managed centrally by the *Cloud Center*. User  $U_i$  does not communicate directly with  $GWN_k$  or  $SN_j$ ; instead, all authentication and key exchange processes are facilitated through the cloud center. The details are illustrated in Figures 6 and 7.

1. User  $U_i$  begins the authentication process by entering  $UID_i^*$ ,  $PW_i^*$ , and  $BIO_i^*$  on the mobile device, which then computes  $\delta_i^* = Rep(BIO_i^*, \tau_i)$ . The device verifies whether  $h(UID_i^* \parallel PW_i^*) \oplus h(\delta_i^*)$  is equal to  $CheckU_i$  stored on it. If the check passes, the device computes  $R2^* = B \oplus h(UID_i^* \parallel \delta_i^*)$  and further verifies whether  $h(UID_i^* \parallel R2^*)$  is equal to  $C$ . If successful,  $U_i$  is logged into the mobile device. The device then generates a random number  $R3$  and timestamp  $T1$ , computing  $M1, M2, M3, M4$ , and  $M5$  as follows:
  - $M1 = R3 \cdot P, M2 = R3 \cdot G, M3 = h(M1 \parallel M2) \oplus R1 \oplus h(UID_i^* \parallel \delta_i^*)$
  - $M4 = h(M1 \parallel M2 \parallel M3) \oplus SID_j$
  - $M5 = h(h(UID_i \parallel \delta_i) \oplus R1 \parallel IM1 \parallel M2 \parallel SID_j))$
 Finally,  $U_i$  sends  $\{UID_i, M2, M3, M4, M5, T1\}$  to *Cloud Center*.
2. Upon receiving the message, the *Cloud Center* first verifies the validity of the timestamp  $T1$ . If it is invalid, the protocol terminates; otherwise, the *Cloud Center* retrieves  $UID_i$  and its associated information from its database. It then computes  $M1^* = d \cdot M2, R1^* = M3 \oplus h(M1^* \parallel M2) \oplus h(UID_i \parallel \delta_i)$ , checking whether  $A \oplus h(UID_i \parallel \delta_i)$  matches  $R1^*$ . If this condition is met, the *Cloud Center* computes  $SID_j^* = M4 \oplus h(M1^* \parallel M2 \parallel M3)$  and verifies whether it matches  $SID_j$ . To authenticate  $U_i$ , it further checks whether  $h(h(UID_i \parallel \delta_i) \oplus R1^* \parallel M1^* \parallel M2 \parallel SID_j)$  matches  $M5$ . If successful, the user authentication is complete. Then *Cloud Center* sends the

messages  $\{T.SID_j, M6, M7, M8, T2, Cn_C\}$  to  $GWN_k$ . The *Cloud Center* then identifies the target gateway  $GID_k$ , generates a random session key  $SK$ , and a timestamp  $T2$ , and computes the following:  $M6$ ,  $M7$ , and  $M8$ .

- $M6 = SK \oplus h(X_{G_k}) \oplus h(CheckG_k \parallel X_{G_k})$
- $M7 = h(SK \parallel M6 \parallel X_{G_k}) \oplus T.SID_j$
- $M8 = h(SK \parallel M6 \parallel X_{G_k} \parallel T.SID_j \parallel T2)$

Finally, the *Cloud Center* sends  $\{T.SID_j, M6, M7, M8, T2, Cn_C\}$  to  $GWN_k$ .

3. Upon receiving the message from the *Cloud Center*,  $GWN_k$  first verifies the validity of timestamp  $T2$ . If invalid, the protocol terminates; otherwise,  $GWN_k$  computes  $SK^* = M6 \oplus h(X_{G_k}) \oplus h(CheckG_k \parallel X_{G_k})$ ,  $T.SID_j^* = M7 \oplus h(SK^* \parallel M6 \parallel X_{G_k})$  and verifies whether  $h(SK^* \parallel M6 \parallel X_{G_k} \parallel T.SID_j^* \parallel T2)$  matches  $M8$ . If successful,  $GWN_k$  generates timestamp  $T3$ , and computes  $Rn_g$ ,  $M9$ , and  $M10$  as follows:

- $Rn_g = (S_{G_k} \oplus Rn_g) \oplus S_{G_k}$
- $M9 = SK \oplus h(Rn_g)$
- $M10 = h(SK \parallel T.SID_j \parallel Rn_g \parallel T3)$

$GWN_k$  then sends  $\{GID_k, M9, M10, T3, Cn_C, Cn_G\}$  to  $SN_j$ .

4. Upon receiving the message,  $SN_j$  verifies the validity of timestamp  $T3$ . If invalid, the protocol terminates; otherwise,  $SN_j$  computes  $Rn_G \leftarrow PUF(Cn_G)$ ,  $SK^* = M9 \oplus h(Rn_G)$  and verifies whether  $h(SK^* \parallel T.SID_j \parallel Rn_G \parallel T3)$  matches  $M10$ . If successful,  $SN_j$  generates timestamp  $T4$  and computes  $Rn_C$ ,  $M11$ , and  $M12$  as follows:

- $Rn_C \leftarrow PUF(Cn_C)$
- $M11 = h(SK^* \parallel h(Rn_g) \parallel T4)$
- $M12 = h(SK^* \parallel h(Rn_C))$

$SN_j$  then sends the message  $\{T.SID_j, M11, M12, T4\}$  to  $GWN_k$ .

5.  $GWN_k$  verifies the validity of timestamp  $T4$  and verifies whether  $h(SK \parallel h(Rn_G) \parallel T4)$  matches  $M11$ . If valid,  $GWN_k$  generates timestamp  $T5$  and computes  $M13 = h(SK \parallel M12) \oplus h(T5 \parallel X_{G_k})$ , then sends  $\{T.SID_j, M12, M13, T5\}$  to the *Cloud Center*.
6. Upon receiving the message, the *Cloud Center* verifies the validity of timestamp  $T5$ . If valid, it retrieves  $SID_j$ ,  $GID_k$ , and  $UID_i$  from the database regarding  $T.SID_j$  and computes  $M12^* = h(SK \parallel h(Rn_C))$ ,  $M13^* = h(SK \parallel M12^*) \oplus h(T5 \parallel X_{G_k})$ . The *Cloud Center* then verifies whether  $M12$  and  $M13$  match  $M12^*$  and  $M13^*$ , respectively. If successful, it generates timestamp  $T6$  and computes  $M14$ ,  $M15$ , and  $M16$  as follows:

- $M14 = SK \oplus h(M1 \parallel R1 \parallel T6) \oplus h(UID_i \parallel \delta_i)$
- $M15 = SK \cdot d$
- $M16 = h(SK \parallel M1 \parallel M15 \parallel T6)$

The *Cloud Center* sends the message  $\{M14, M15, M16, T6\}$  to  $U_i$ .

7.  $U_i$  verifies the validity of timestamp  $T6$ . If valid,  $U_i$  computes  $SK^*$  and confirms the integrity of the shared  $SK^*$  by performing the following steps:

- Computing  $SK^* = M14 \oplus h(M1 \parallel R1 \parallel T6) \oplus h(UID_i^* \parallel \delta_i^*)$ .
- Verifying  $SK^* \cdot P$  matches  $M15 \cdot G$ .
- Calculating  $h(SK^* \parallel M1 \parallel M15 \parallel T6)$  and checking whether it is equal to  $M16$ .

If all conditions hold, the session key is securely shared among the participants.

8. Finally, at the end of the session, the *Cloud Center*,  $GWN_k$ , and  $SN_j$  update  $T.SID_j$  as follows:

- $T.SID_j^{new} = T.SID_j \oplus M12$
- $T.SID_j = T.SID_j^{new}$

$U_i$	Cloud Center	$GW N_k$	$SN_j$
Inputs $UID_i^*$ , $PW_i^*$ , $BIO_i^*$ Computes $Rep(BIO_i^*, \tau_i) = \sigma_i^*$ Checks $CheckU_i = ? h(UID_i^* \parallel PW_i^*) \oplus h(\sigma_i^*)$ Computes $R2^* = B \oplus h(UID_i^* \parallel \sigma_i^*)$ Checks $C = ? h(UID_i^* \parallel R2^*)$ Generates random number $R3$ , timestamp $T1$ Computes $M1 = R3 \cdot P$ , $M2 = R3 \cdot G$ , $M3 = h(M1 \parallel M2) \oplus R1 \oplus h(UID_i^* \parallel \sigma_i^*)$ $M4 = h(M1 \parallel M2 \parallel M3) \oplus SID_j$ , $M5 = h(h(UID_i^* \parallel \sigma_i^*) \oplus R1 \parallel M1 \parallel M2 \parallel SID_j)$ $\{UID_i, M2, M3, M4, M5, T1\} \longrightarrow$ Validity check $T1$ Computes $M1^* = d \cdot M2$ , $R1^* = M3 \oplus h(M1^* \parallel M2) \oplus h(UID_i \parallel \sigma_i)$ Checks $R1^* = ? A \oplus h(UID_i \parallel \sigma_i)$ Computes $SID_j^* = M4 \oplus h(M1^* \parallel M2 \parallel M3)$ Checks $SID_j^* = ? SID_j$ Checks $M5 = ? h(h(UID_i \parallel \sigma_i) \oplus R1^* \parallel M1^* \parallel M2 \parallel SID_j)$ Generates random session key $SK$ , timestamp $T2$ Computes $M6 = SK \oplus h(X \cdot G_k) \oplus h(CheckG_k \parallel X \cdot G_k)$ $M7 = h(SK \parallel M6 \parallel X \cdot G_k) \oplus T \cdot SID_j$ , $M8 = h(SK \parallel M6 \parallel X \cdot G_k \parallel T \cdot SID_j \parallel T2)$ $\{T \cdot SID_j, M6, M7, M8, Cn_c, T2, \} \longrightarrow$ Validity check $T2$ Computes $SK^* = M6 \oplus h(X \cdot G_k) \oplus h(CheckG_k \parallel X \cdot G_k)$ $T \cdot SID_j^* = M7 \oplus h(SK^* \parallel M6 \parallel X \cdot G_k)$ Check $M8 = ? h(SK^* \parallel M6 \parallel X \cdot G_k \parallel T \cdot SID_j^* \parallel T2)$ Generates timestamp $T3$ Computes $Rn_G = (SG_k \oplus Rn_G) \oplus SG_k$ $M9 = SK^* \oplus h(Rn_G)$ , $M10 = h(SK^* \parallel T \cdot SID_j \parallel Rn_G \parallel T3)$ $\{GID_j, M9, M10, Cn_c, Cn_g, T3\} \longrightarrow$ Validity check $T2$			

Figure 6. Authentication and key distribution phase (Steps 1–3).

$U_i$	Cloud Center	$GW N_k$	$SN_j$
		$\{GID_j, M9, M10, Cn_c, Cn_g, T3\} \longrightarrow$ Validity check $T3$ Computes $Rn_G \leftarrow PUF(Cn_G)$ Computes $SK^* = M9 \oplus h(Rn_G)$ Check $M10 = ? h(SK^* \parallel T \cdot SID_j \parallel Rn_G \parallel T3)$ Generates timestamp $T4$ Computes $Rn_c \leftarrow PUF(Cn_C)$ $M11 = h(SK^* \parallel h(Rn_c) \parallel T4)$ , $M12 = h(SK^* \parallel h(Rn_c))$ $\leftarrow \{T \cdot SID_j, M11, M12, T4\}$	
		Validity check $T4$ Check $M11 = ? h(SK^* \parallel h(Rn_c) \parallel T4)$ Generates timestamp $T5$ Computes $M13 = h(SK^* \parallel M12) \oplus h(T5 \parallel X \cdot G_k)$ , $\leftarrow \{T \cdot SID_j, M12, M13, T5\}$	
	Validity check $T5$ Computes $M12^* = h(SK \parallel h(Rn_c))$ , $M13^* = h(SK \parallel M12^*) \oplus h(T5 \parallel X \cdot G_k)$ Check $M12^* = ? M12$ , $M13^* = ? M13$ Generates timestamp $T6$ Computes $M14 = SK \oplus h(M1 \parallel R1 \parallel T6) \oplus h(UID_i \parallel \sigma_i)$ $M15 = SK \cdot d$ , $M16 = h(SK \parallel M1 \parallel M15 \parallel T6)$ $\leftarrow \{M14, M15, M16, T6\}$		
Validity check $T6$ Computes $SK^* = M14 \oplus h(M1 \parallel R1 \parallel T6) \oplus h(UID_i^* \parallel \sigma_i^*)$ Check $SK^* \cdot P = ? M15 \cdot G$ Computes $M16^* = h(SK^* \parallel M1 \parallel M15 \parallel T6)$ , Check $M16^* = ? M16$			
Cloud Center, $GW N_k$ , $SN_j$ updates $T \cdot SID_j^* = T \cdot SID_j \oplus M12$ , and sets $T \cdot SID_j = T \cdot SID_j^*$ at the end of the session			

Figure 7. Authentication and key distribution phase (Steps 4–7).

### 6.5. Password Update Phase

At the end of the session,  $U_i$  inputs  $UID_i^*$ ,  $PW_i^*$ , and  $BIO_i^*$  into the smart device to initiate the password update process. The device computes  $CheckU_i^* = h(UID_i^* \parallel PW_i^*) / plush(\delta_i^*)$  and verifies whether  $CheckU_i^*$  matches the stored  $CheckU_i$ . If the values match,  $U_i$  inputs  $PW_i^{new}$  as the new password and computes  $CheckU_i^{new} = h(UID_i \parallel$



$PW_i^n) / oplush(\delta_i)$ . Finally, the password update is completed when  $U_i$  updates  $CheckU_i$  to  $CheckU_i^{new}$  and stores it in the device's memory.

## 7. Security Analysis of Proposed Scheme

In this section, we present both formal and informal security analyses. The details are as follows:

### 7.1. Formal Security Analysis

In this paper, the ProVerif tool is used to formally analyze the security of the proposed authentication protocol [33–37]. ProVerif, developed by Blanchet et al. [38], is widely used for verifying security properties such as confidentiality and authentication in cryptographic protocols. The proposed protocol is modeled in Applied Pi-Calculus, and its resistance to attacks is evaluated under the Dolev–Yao model [21].

The details of each code are presented in Tables 2–8. Table 2 describes the variables, events, channels, and function declarations in the ProVerif code. Table 3 presents the user registration and authentication phases. Table 4 illustrates the operation process of the cloud center, while Table 5 outlines the gateway's operation process. Table 6 describes the operation process of the sensor node. Table 7 lists the queries used to verify the overall operation, and Table 8 presents the results obtained when the queries from Table 7 were executed.

**Table 2.** Definitions of channels, variables, and other related parameters.

---

```
(*—channels—*1)
free privateChannel: channel [private].
free publicChannel: channel.
(*—constants—*)
free UserSigma: bitstring [private].
free UserPassword: bitstring [private].
free UserID: bitstring [private].
free GatewayKey: bitstring [private].
free GatewayID: bitstring.
free ServerID: bitstring.
(*—shared key—*)
free SharedKey: bitstring [private].
(*—functions—*)
fun xor(bitstring, bitstring): bitstring.
fun concat(bitstring, bitstring): bitstring.
fun h(bitstring): bitstring.
fun scalar_mult(bitstring, bitstring): bitstring.
(*—events—*)
event startUser(bitstring).
event endUser(bitstring).
event startCloudCenter().
event endCloudCenter().
event startGateway(bitstring).
event endGateway(bitstring).
event startSensorNode(bitstring).
event endSensorNode(bitstring).
```

---

<sup>1</sup> In ProVerif, the (\*...\*) notation indicates a comment.

**Table 3.** User process.

---

```

(*—User Process—*)
let UserProcess = let checkUserID = xor(h(concat(UserID, UserPassword)), h(UserSigma)) in
out(privateChannel, (UserID, UserSigma));
in(privateChannel, (receivedUserID: bitstring, receivedA: bitstring, receivedP: bitstring, receivedG:
bitstring));
if receivedUserID = UserID then
let computedR1 = xor(receivedA, h(concat(UserID, UserSigma))) in
new R2: bitstring;
let computedB = xor(h(concat(UserID, UserSigma)), R2) in
let computedC = h(concat(UserID, R2)) in
event startUser(UserID);
new inputUserID:bitstring;
new inputUserPassword:bitstring;
new inputUserSigma:bitstring;
if checkUserID = xor(h(concat(inputUserID, inputUserPassword)), h(inputUserSigma)) then
let computedR2 = xor(computedB, h(xor(inputUserID, inputUserSigma))) in
if computedC = h(concat(inputUserSigma, computedR2)) then
new R3:bitstring;
new T1:bitstring;
new SensorID:bitstring;
let computedM1 = scalar_mult(R3, receivedP) in
let computedM2 = scalar_mult(R3, receivedG) in
let computedM3 = xor(xor(h(concat(computedM1, computedM2)), computedR1),
h(concat(inputUserID, inputUserSigma))) in
let computedM4 = xor(h(xor(xor(computedM1, computedM2), computedM3)), SensorID) in
let computedM5 = h(concat(concat(concat(computedR1, computedM1), computedM2), SensorID)) in
out(publicChannel, (inputUserID, computedM2, computedM3, computedM4, computedM5, T1 ));
in(publicChannel, (receivedM14: bitstring, receivedM15: bitstring, receivedM16: bitstring,
receivedT6: bitstring));
let SharedKey = xor(xor(receivedM14, h(concat(concat(computedM1, computedR1), receivedT6))),
h(concat(inputUserID, inputUserSigma))) in
if scalar_mult(SharedKey, receivedP) = scalar_mult(computedM5, receivedG) then
let computedM16 = h(concat(concat(concat(SharedKey, computedM1), receivedM15), receivedT6)) in
if computedM16 = receivedM16 then
event endUser(UserID).

```

---

This study conducted an automated formal verification using the ProVerif tool to evaluate the security and safety of the designed protocol.

The ProVerif queries were configured as follows:

- query `inj-event(endUser(idi)) ==> inj-event(startUser(idi))` verifies that the user's authentication completion event occurs only if the authentication initiation event has occurred (mutual authentication for users).
- query `inj-event(endCloudCenter()) ==> inj-event(startCloudCenter())` verifies the correct execution of the cloud center's authentication process.
- query `inj-event(endGateway(gwi)) ==> inj-event(startGateway(gwi))` checks the correctness of the gateway's authentication.
- query `inj-event(endSensorNode(snj)) ==> inj-event(startSensorNode(snj))` checks the correctness of the sensor node's authentication.
- query `attacker(SharedKey)` verifies the confidentiality of the session key, ensuring it is not accessible to the attacker.

**Table 4.** CloudCenter process.

---

```

(*—CloudCenter Process—*)
let CloudCenterProcess = in(privateChannel, (receivedUserID:bitstring, receivedSigma:bitstring));
new R1:bitstring;
new P:bitstring;
new G:bitstring;
let computedA = xor(h(concat(receivedUserID,receivedSigma)),R1) in
out(privateChannel, (receivedUserID, computedA, P, G));
in(privateChannel, (receivedGatewayID:bitstring));
new SC:bitstring;
let computedXG=h(concat(SC,receivedGatewayID)) in
out(privateChannel, (receivedGatewayID, computedXG));
in(privateChannel, (receivedCheckG:bitstring));
in(privateChannel, (receivedSensorID:bitstring));
new Challenge:bitstring;
new TempSensorID:bitstring;
out(privateChannel, (Challenge,TempSensorID));
in(privateChannel,(receivedRNC:bitstring));
event startCloudCenter();
in(publicChannel, (receivedUserID:bitstring, receivedM2:bitstring, receivedM3:bitstring,
receivedM4:bitstring, receivedM5:bitstring, receivedT1:bitstring));
new d:bitstring;
let computedM1=scalar_mult(d, receivedM2) in
let computedR1=xor(xor(receivedM3,h(concat(computedM1,receivedM2))),h(concat(receivedUserID,receivedSigma
))) in
if computedR1= xor(computedA, h(concat(receivedUserID, receivedSigma))) then
let computedSensorID = xor(receivedM4, h(concat(concat(computedM1,receivedM2),receivedM3))) in
if computedSensorID=receivedSensorID then
if receivedM5=h(concat(concat(concat(xor(h(concat(receivedUserID,receivedSigma)),
computedR1),computedM1),receivedM2),receivedSensorID)) then
new SK:bitstring;
new T2:bitstring;
let computedM6 = xor(xor(SK,h(computedXG)),h(concat(receivedCheckG,computedXG))) in
let computedM7 = xor(h(concat(concat(SK,computedM6),computedXG)),TempSensorID) in
let computedM8 =h(concat(concat(concat(concat(SK,computedM6),computedXG),TempSensorID),T2)) in
out(publicChannel,(TempSensorID,computedM6, computedM7, computedM8, Challenge, T2));
in(publicChannel, (receivedTempSensorID:bitstring, receivedM12:bitstring, receivedM13:bitstring,
receivedT5:bitstring));
let computedM12= h(concat(SK,h(receivedRNC))) in
let computedM13= xor(h(concat(SK,computedM12)),h(concat(receivedT5,computedXG))) in
if computedM12=receivedM12 then
if computedM13=receivedM13 then
new T6:bitstring;
let computedM14 =
xor(xor(SK,h(concat(concat(computedM1,computedR1),T6))),h(concat(receivedUserID,receivedSigma))) in
let computedM15 =scalar_mult(SK, d) in
let computedM16 = h(concat(concat(concat(SK,computedM1),computedM15),T6)) in
out(publicChannel, (computedM14, computedM15,computedM16, T6));
event endCloudCenter().

```

---

**Table 5.** Gateway process.

---

```

(*—Gateway Process—*) let GatewayProcess = new GatewayID:bitstring;
out(publicChannel, (GatewayID));
in(publicChannel, (receivedGatewayID:bitstring, receivedXG:bitstring));
new SG:bitstring;
let computedCheckG = h(concat(SG, receivedXG)) in
out(publicChannel, (computedCheckG));
in(publicChannel, (receivedSensorID:bitstring));
new CNg:bitstring;
out(publicChannel, (CNg));
in(publicChannel, (receivedSensorID:bitstring, receivedTempSensorID:bitstring, receivedRN:bitstring));
event startGateway(GatewayID);
in(publicChannel, (receivedTempSensorID:bitstring, receivedM6:bitstring, receivedM7:bitstring,
receivedM8:bitstring, receivedCNc:bitstring, receivedT2:bitstring));
let computedSK = xor(xor(receivedM6, h(receivedXG)), h(concat(computedCheckG, receivedXG))) in
let TempSensorID = xor(receivedM7, h(concat(concat(computedSK, receivedM6), receivedXG))) in
if receivedM8 =
h(concat(concat(concat(concat(computedSK, receivedM6), receivedXG), receivedTempSensorID), receivedT2)) then
new T3:bitstring;
let computedRNg = xor(xor(SG, receivedRN), SG) in
let computedM9 = xor(computedSK, h(receivedRN)) in
let computedM10 = h(concat(concat(concat(computedSK, receivedTempSensorID), receivedRN), T3)) in
out(publicChannel, (GatewayID, computedM9, computedM10, receivedCNc, CNg, T3 ));
in(publicChannel, (receivedTempSensorID:bitstring, receivedM11:bitstring, receivedM12:bitstring,
receivedT4:bitstring));
if receivedM11 = h(concat(concat(computedSK, h(receivedRN)), receivedT4)) then
new T5:bitstring;
let computedM13 = xor(h(concat(computedSK, receivedM12)), h(concat(T5, receivedXG))) in
out(publicChannel, (receivedTempSensorID, receivedM12, computedM13, T5));
event endGateway(GatewayID).

```

---

**Table 6.** SensorNode process.

---

```

(*—SensorNode Process—*)
let SensorNodeProcess = new SensorID:bitstring;
out(publicChannel, (SensorID));
in(publicChannel, (receivedCn:bitstring, receivedTempSensorID:bitstring));
new RNc:bitstring;
out(publicChannel, (RNc));
out(publicChannel, (SensorID));
in(publicChannel, (CNg:bitstring));
new RNg:bitstring;
out(publicChannel, (SensorID, receivedTempSensorID, RNg ));
event startSensorNode(SensorID);
in(publicChannel, (receivedGatewayID:bitstring, receivedM9:bitstring, receivedM10:bitstring,
receivedCNc:bitstring, receivedCNg:bitstring, receivedT3:bitstring));
new RNg:bitstring;
let computedSK = xor(receivedM9, h(RNg)) in
if receivedM10 = h(concat(concat(concat(computedSK, receivedTempSensorID), RNg), receivedT3)) then
new T4:bitstring;
new RNc:bitstring;
let computedM11 = h(concat(concat(computedSK, h(RNg)), T4)) in
let computedM12 = h(concat(computedSK, h(RNc))) in
out(publicChannel, (receivedTempSensorID, computedM11, computedM12, T4 ));
event endSensorNode(SensorID).

```

---

**Table 7.** Queries and main process.

(*—queries—*)
query idi:bitstring; inj-event(endUser(idi)) ==> inj-event(startUser(idi)).
query inj-event(endCloudCenter()) ==> inj-event(startCloudCenter()).
query gwi:bitstring; inj-event(endGateway(gwi)) ==> inj-event(startGateway(gwi)).
query snj:bitstring; inj-event(endSensorNode(snj)) ==> inj-event(startSensorNode(snj)).
query attacker(SharedKey).
(*—process—*)
process ((!UserProcess)   (!CloudCenterProcess)   (!GatewayProcess)   (!SensorNodeProcess))

**Table 8.** Result.

Verification summary:
Query inj-event(endUser(idi)) ==> inj-event(startUser(idi)) is true.
Query inj-event(endCloudCenter) ==> inj-event(startCloudCenter) is true.
Query inj-event(endGateway(gwi)) ==> inj-event(startGateway(gwi)) is true.
Query inj-event(endSensorNode(snj)) ==> inj-event(startSensorNode(snj)) is true.
Query not attacker(SharedKey[]) is true.

The verification results show that all authentication and confidentiality queries returned true. This indicates that the proposed protocol successfully guarantees mutual authentication among the user, cloud center, gateway, and sensor node, and that the session key remains confidential against potential attackers. The analysis results obtained from ProVerif show that all specified security queries were proven to be true, confirming that the protocol successfully meets its intended security requirements. Specifically, the verification demonstrated that a user's termination event always occurs after the corresponding initiation event, clearly ensuring user authentication. Additionally, the termination event at the cloud center was verified to occur strictly following its initiation event, thereby validating the reliability of the system's communication flow. Furthermore, the gateway and sensor nodes were also confirmed to maintain correct sequential consistency between their respective initiation and termination events, securing the integrity of message exchanges. Moreover, the analysis mathematically verified that an attacker could not access or compromise the primary shared keys, highlighting the strong confidentiality provided by the protocol. Based on these comprehensive results, it can be concluded that the proposed protocol effectively fulfills critical security requirements such as authentication, confidentiality, and integrity.

## 7.2. Informal Security Analysis

The proposed scheme satisfies nine critical security requirements, including protection against user-impersonation attacks, stolen-device attacks, session key disclosure attacks, anonymity and untraceability, mutual authentication, replay attacks, forward secrecy attacks, man-in-the-middle attacks, and PUF modeling attacks [39–42]. A comparison of the latest studies and their corresponding security properties is summarized in Table 9. The detailed security properties of the proposed scheme are as follows:

**Table 9.** Comparison of security features.

Security Features	Wang et al. [12]	Wazid et al. [13]	Yang et al. [14]	Srinivas et al. [15]	Wazid et al. [16]	Dai et al. [17]	Hu et al. [18]	Jiang et al. [19]	Ours
A1	O	O	O	O	O	O	O	O	O
A2	O	O	O	O	O	O	O	O	O
A3	X	O	O	O	O	O	O	O	O
A4	O	O	O	O	O	O	O	O	O
A5	O	O	O	O	O	O	O	O	O
A5	O	O	O	O	O	O	O	O	O
S1	O	O	O	X	O	O	O	O	O
S2	O	O	O	O	O	O	O	O	O
S3	O	X	X	O	X	O	O	O	O

#### 7.2.1. A1: Resistance to User-Impersonation Attack

The proposed scheme effectively resists user-impersonation attacks because even if an attacker intercepts the authentication messages sent to the cloud center, they cannot generate  $h(UID_i \parallel \delta_i)$ , which is a critical parameter for authentication. As the cloud center must calculate  $R1^* = M3 \oplus h(M1^* \parallel M2)) \oplus h(UID_i \parallel \delta_i)$  to verify the user, the inability of the attacker to reconstruct  $h(UID_i \parallel \delta_i)$  prevents further progress in the authentication process. The scheme is also resistant to stolen-device attacks.

#### 7.2.2. A2: Resistance to Stolen-Device Attack

Even if an attacker physically steals a user's smart device and obtains stored information  $\{checkU_i, R1, B, C, \tau_i, P, G\}$ , they cannot generate the valid biometric information  $\delta_i$  required for login or authentication. Furthermore, for IoT devices, even if an attacker gains access to  $\{SID_j, T.SID_j\}$  and intercepts messages transmitted over public channels, they cannot generate the response  $Rn_G$  used for authentication, owing to the nature of the PUF. The scheme is also resistant to stolen-device attacks.

#### 7.2.3. A3: Resistance to Session Key Disclosure Attack

Messages  $M6$  and  $M9$ , which contain session key  $SK$  information, are transmitted over a public channel. However, an attacker cannot compute  $SK$  from these messages because they cannot access  $H(X_{G_k})$  and  $Rn_G$ , both of which are essential for deriving  $SK$ . The scheme ensures security against session key exposure attacks.

#### 7.2.4. A4: Resistance to Replay Attack

During the authentication and key distribution phase, each transmitted message includes a timestamp, and the recipient first verifies its validity. If the timestamp is invalid, the session is immediately terminated. Therefore, the proposed scheme is resistant to replay attacks.

#### 7.2.5. A5: Resistance to Man-in-the-Middle Attack

In the proposed scheme, secret information that can only be generated by each participant is used for mutual authentication between  $U_i$ , Cloud Center,  $GW N_k$ , and  $SN_j$  during the authentication and key distribution phases. For instance, even if an attacker intercepts all messages transmitted over the public channel, they may attempt to change  $M2$  and  $M3$  in a message from  $U_i$  to Cloud Center to impersonate the legitimate  $U_i$ . However, the attacker cannot generate  $h(UID_i \parallel \delta_i)$ , which is uniquely generated by  $U_i$ . Therefore, the request message sent by the attacker cannot successfully authenticate as the legitimate user. In this manner, the proposed scheme is resistant to man-in-the-middle attacks, as the authentication value is generated using secret information known only to the legitimate participants, making it impossible for an attacker to alter or forge this value.



#### 7.2.6. A6: Resistance to PUF Modeling Attack

Finally, the scheme is resistant to PUF modeling attacks, which involve collecting a large number of PUF challenge–response pairs and using machine learning techniques to predict legitimate responses. In the proposed scheme, two PUF challenge–response pairs,  $(Cn_C, Rn_C)$  and  $(Cn_G, Rn_G)$ , are transmitted over a private channel during the IoT device registration phase. However, during the authentication and key-sharing phases, only the PUF challenges  $Cn_C$  and  $Cn_R$  are transmitted over a public channel, while the responses remain private. This prevents attackers from collecting sufficient challenge–response pairs to train a predictive model, making PUF modeling attacks infeasible.

#### 7.2.7. S1: Provide Anonymity and Untraceability

Regarding anonymity and untraceability, the proposed telemedicine system model does not require user anonymity because the doctor communicates with the cloud center rather than directly with the IoT device. However, anonymity and untraceability are necessary for the patient's IoT device. The scheme achieves this by using  $T.SID_j$ , a temporary ID for  $SID_j$ , which is randomly generated for each session and updated at the end of the session to provide anonymity and untraceability.

#### 7.2.8. S2: Provide Mutual Authentication

During the authentication and key distribution phases,  $U_i$ , *Cloud Center*,  $GWN_k$ , and  $SN_j$  undergo mutual authentication. The *Cloud Center* authenticates  $U_i$  by verifying that  $R1^*$  and  $M5$  match  $A \oplus h(UID_i \parallel \delta_i)$  and  $h(h(UID_i \parallel \delta_i) \oplus R1^* \parallel M1^* \parallel M2 \parallel SID_j)$ , respectively.  $GWN_k$  authenticates the *Cloud Center* by verifying  $M8$ , while  $SN_j$  authenticates  $GWN_k$  by verifying  $M10$ . If any of these values are invalid, the session is aborted, ensuring that the proposed scheme provides mutual authentication.

#### 7.2.9. S3: Provide Forward and Backward Secrecy

The proposed scheme guarantees forward and backward secrecy by ensuring that the session key  $SK$  is randomly generated for each session. Although it is used to compute  $M12$ , it cannot be inferred owing to the one-way nature of the hash function. Therefore, even if the session key for a particular session is compromised, an attacker cannot determine the session keys for previous or future sessions. Therefore, the proposed scheme provides forward secrecy and backward secrecy.

## 8. Performance Analysis of Proposed Scheme

In this section, we compare the performance of the proposed scheme with recent studies [12–19], focusing on minimizing the computational overhead of IoT devices while evaluating our scheme against existing works.

For comparison, we utilized computational overhead measurements obtained from a prior study conducted in a CPU-based environment using an Intel Core i7-8700 (3.20 GHz) processor, Windows 10 (64-bit) OS, and 48 GB of memory, employing the Python cryptography library 3.13.2 [33]. The results are presented in Table 10.

In the study by Wang et al. [12], the user performs 8 hash function computations, 1 fuzzy extractor operation, and 3 elliptic curve multiplications, which can be expressed as  $8h + 1f + 3m$ . Based on the experimental environment, this corresponds to 2129.52  $\mu$ s. At the cloud center, 10 hash function computations and 1 elliptic curve multiplication are performed, expressed as  $10h + 1m = 533.9 \mu$ s. The gateway node performs 9 hash function computations, represented as  $9h = 1.71 \mu$ s. Additionally, the sensor node executes 4 hash function computations and 2 elliptic curve multiplications, which can be expressed as  $4h + 2m = 1064.76 \mu$ s.

**Table 10.** Computation times for each operation ( $\mu\text{s}$ ).

Symbol	Meaning	Time ( $\mu\text{s}$ )
$h$	Computation time for hash functions	0.19
$f$	Extraction time of biometric information in fuzzy extractors	532
$m$	Multiplication operation time in ECC	532
$s$	Computation time for symmetric key encryption or decryption	0.27
$c$	Chebyshev polynomial computation time	3.8

Wazid et al. [13] introduced an approach where the user carries out  $13h + 2s + 1f$ , which corresponds to 535.01  $\mu\text{s}$ . Their scheme does not involve a cloud center, so the gateway node handles  $5h + 4s$ , taking 2.03  $\mu\text{s}$ . On the sensor side,  $4h + 2s$  is performed, resulting in 1.3  $\mu\text{s}$ .

Yang et al. [14]’s method focuses on lightweight computation, requiring the user to compute  $8h$ , leading to 1.52  $\mu\text{s}$ . The gateway node processes  $14h$ , with an execution time of 2.66  $\mu\text{s}$ , while the sensor node performs  $7h$ , taking 1.33  $\mu\text{s}$ .

Srinivas et al. [15] proposed an approach where the user performs  $2c + 15h + 1f$ , consuming 542.45  $\mu\text{s}$ . Their method involves computations at the gateway node, where  $10h$  is executed, taking 1.9  $\mu\text{s}$ . The sensor node carries out  $2c + 6h$ , leading to a processing time of 8.74  $\mu\text{s}$ .

Wazid et al.’s scheme in [16] follows a slightly different structure. The user executes  $9h + 1s + 1f$ , with a computation time of 533.98  $\mu\text{s}$ . The gateway node computes  $11h + 2s$ , requiring 2.63  $\mu\text{s}$ , while the sensor node carries out  $7h + 1s$ , taking 1.6  $\mu\text{s}$ .

Dai et al. [17] introduced a model where the user must perform  $1f + 9h + 3m$ , consuming 2129.71  $\mu\text{s}$ . The gateway node handles  $11h + 1m$ , with an execution time of 534.09  $\mu\text{s}$ , and the sensor node processes  $5h + 2m$ , leading to 1064.95  $\mu\text{s}$ .

Hu et al. [18]’s scheme assigns  $15h + 1f + 2c$  computations to the user, requiring 542.45  $\mu\text{s}$ . Their method does not include cloud center processing, so the gateway node executes  $7h + 1s$ , with a computation time of 1.6  $\mu\text{s}$ . Meanwhile, the sensor node performs  $4h + 2c + 1s$ , resulting in 8.63  $\mu\text{s}$ .

Jiang et al. [19] designed a method where the user processes  $1f + 11h + 5m + 1s$ , which takes 3194.36  $\mu\text{s}$ . The cloud center, in their model, executes  $5m + 1s + 12h$ , consuming 2662.55  $\mu\text{s}$ , while the sensor node handles  $4h$ , leading to 0.76  $\mu\text{s}$ .

In our proposed scheme, as demonstrated in Table 11, we optimize the computational cost across all entities. The user performs  $9h + 1f + 4m$ , leading to 2661.71  $\mu\text{s}$ . Our approach eliminates unnecessary computations at the cloud center, processing  $2m + 17h$ , taking 1067.23  $\mu\text{s}$ . The gateway node handles  $9h$ , requiring 1.71  $\mu\text{s}$ , while the sensor node executes  $3h$ , resulting in a final processing time of 0.57  $\mu\text{s}$ . Our proposed scheme increases the computational load at the cloud center and user mobile devices, thereby reducing the computational burden on the sensor node. While the average computational cost at the sensor node in other schemes is 202.5563  $\mu\text{s}$ , our scheme reduces this to 0.57  $\mu\text{s}$ , making it approximately 35,436.18% more efficient. The overall efficiency of the authentication scheme is most affected by the load and cost of the sensor nodes. Relatively speaking, cloud centers and user mobile devices have relatively high computing resources compared to sensor nodes and gateways, so the increase in computation does not have a significant impact on the overall efficiency of the authentication scheme. Moreover, as demonstrated in Table 9, our proposed scheme ensures stronger security performance compared to other studies.

**Table 11.** Comparisons of computational costs ( $\mu$ s).

Scheme	User	Cloud Center	Gateway Node	Sensor Node
Wang et al. [12]	$8h + 1f + 3m$	$10h + 1m$	$9h$	$4h + 2m$
Wazid et al. [13]	$13h + 2s + 1f$	-	$5h + 4s$	$4h + 2s$
Yang et al. [14]	$8h$	-	$14h$	$7h$
Srinivas et al. [15]	$2c + 15h + 1f$	-	$10h$	$2c + 6h$
Wazid et al. [16]	$9h + 1s + 1f$	-	$11h + 2s$	$7h + 1s$
Dai et al. [17]	$1f + 9h + 3m$	-	$11h + 1m$	$5h + 2m$
Hu et al. [18]	$15h + 1f + 2c$	-	$7h + 1s$	$4h + 2c + 1s$
Jiang et al. [19]	$1f + 11h + 5m + 1s$	$5m + 1s + 12h$	-	$4h$
Ours	$9h + 1f + 4m$	$2m + 17h$	$9h$	$3h$

## 9. Conclusions

In this study, we analyzed Wang et al.'s authentication scheme and identified several limitations, including the vulnerability in illegal session key exchanges due to stored information in IoT devices and publicly transmitted data, as well as inefficiencies despite utilizing cloud centers. To overcome these issues, we incorporated *PUF* technology into IoT devices to prevent unauthorized session key exchanges and improve efficiency by ensuring that public-key computations are performed exclusively in the cloud center. In addition to safeguarding patient privacy and maintaining untraceability, IoT devices utilize only a temporary  $T.SID_j$  during authentication and key distribution. In this study, computationally intensive operations are offloaded to the cloud to significantly reduce the computational and energy burden on sensor nodes. This approach enables stable service provision even in resource-constrained medical IoT devices. By performing all authentication and key exchange processes through the cloud center, the proposed scheme achieves lightweight operation at both sensor nodes and gateways. The system is designed under the assumption that the cloud center possesses sufficient computational power and communication resources. In practical deployments, ensuring appropriate cloud resource provisioning and adopting efficient resource management strategies will help maintain system efficiency and stability. Through this approach, the proposed authentication and key distribution scheme is expected to maintain efficiency and robustness across diverse environments. Although the proposed authentication and key distribution scheme is designed for medical IoT environments, its structure is not application-specific, allowing for flexible adaptation to other IoT domains. For instance, the lightweight sensor node design and cloud-based authentication framework can be readily applied to smart homes, industrial IoT, and smart city environments. Therefore, the proposed scheme can maintain high efficiency and stability not only in healthcare but also in a variety of large-scale IoT deployments. Finally, we compared the security and efficiency of our scheme with recent authentication schemes [12–19] and validated its security using ProVerif, a formal security analysis tool.

**Author Contributions:** Conceptualization: H.J.L., S.K., and K.K.; methodology: H.J.L. and J.R.; software: S.K. and K.K.; validation: H.J.L., H.L., and J.R.; formal Analysis: K.K. and J.R.; writing—original draft preparation: H.J.L., S.K., and K.K.; writing—review and editing: J.R. and Y.L.; visualization: H.J.L.; supervision: J.R. and D.W.; project administration: J.R. and D.W.; funding acquisition: D.W. All authors have read and approved the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data are contained within the article.

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

## References

1. Ali, Z.; Mahmood, S.; Mansoor, K.; Daud, A.; Alharbey, R.; Bukhari, A. A lightweight and secure authentication scheme for remote monitoring of patients in IoMT. *IEEE Access* **2024**, *12*, 73004–73020. [CrossRef]
2. Manickam, N.; Ponnusamy, V. A Review of Secure Healthcare Data Analytics using Federated Machine Learning and Blockchain Technology. *IEIE Trans. Smart Process. Comput.* **2024**, *13*, 254–262. [CrossRef]
3. Batool, R.; Raza, G.M.; Khalid, U.; Kim, B.S. Automated Detection of COVID-19 in Chest Radiographs: Leveraging Machine Learning Approaches. *IEIE Trans. Smart Process. Comput.* **2024**, *13*, 572–578.
4. Lee, H.J.; Kook, S.; Kim, K.; Ryu, J.; Lee, Y.; Won, D. LAMT: Lightweight and Anonymous Authentication Scheme for Medical Internet of Things Services. *Sensors* **2025**, *25*, 821. [CrossRef]
5. Prajapat, S.; Kumar, P.; Kumar, D.; Das, A.K.; Hossain, M.S.; Rodrigues, J.J. Quantum secure authentication scheme for internet of medical things using blockchain. *IEEE Internet Things J.* **2024**, *11*, 38496–38507. [CrossRef]
6. Mookherji, S.; Vanga, O.; Prasath, R.; Das, A.K. A secure authentication protocol for remote patient monitoring in an internet-of-medical-things environment. *Secur. Priv.* **2024**, *7*, e428. [CrossRef]
7. Sachnev, V.; Suresh, M.B. An Automatic Diagnostic Tool for Autism Spectrum Disorder using Structural Magnetic Resonance Imaging and a Tailored Binary Coded Genetic Algorithm. *IEIE Trans. Smart Process. Comput.* **2024**, *13*, 236–242. [CrossRef]
8. Fortune Business Insights. Available online: <https://www.fortunebusinessinsights.com/press-release/telemedicine-market-9214> (accessed on 20 March 2025).
9. Ekambaram, D.; Ponnusamy, V. AI-assisted physical therapy for post-injury rehabilitation: Current state of the art. *IEIE Trans. Smart Process. Comput.* **2023**, *12*, 234–242. [CrossRef]
10. Rani, D.; Tripathi, S. Design of blockchain-based authentication and key agreement protocol for health data sharing in cooperative hospital network. *J. Supercomput.* **2024**, *80*, 2681–2717. [CrossRef]
11. Yadav, N.S.; Goar, V.K. IoT in Healthcare and Telemedicine: Revolutionizing Patient Care and Medical Practices. In *Scalable Modeling and Efficient Management of IoT Applications*; IGI Global: Hershey, PA, USA, 2025; pp. 19–58.
12. Wang, C.; Wang, D.; Duan, Y.; Tao, X. Secure and lightweight user authentication scheme for cloud-assisted Internet of Things. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 2961–2976. [CrossRef]
13. Wazid, M.; Das, A.K.; Odelu, V.; Kumar, N.; Conti, M.; Jo, M. Design of secure user authenticated key management protocol for generic IoT networks. *IEEE Internet Things J.* **2018**, *5*, 269–282. [CrossRef]
14. Yang, Z.; He, J.; Tian, Y.; Zhou, J. Faster authenticated key agreement with perfect forward secrecy for industrial internet-of-things. *IEEE Trans. Ind. Inform.* **2020**, *16*, 6584–6596. [CrossRef]
15. Srinivas, J.; Das, A.K.; Wazid, M.; Kumar, N. Anonymous lightweight chaotic map-based authenticated key agreement protocol for industrial Internet of Things. *IEEE Trans. Dependable Secur. Comput.* **2020**, *17*, 1133–1146. [CrossRef]
16. Wazid, M.; Das, A.K.; Odelu, V.; Kumar, N.; Susilo, W. Secure remote user authenticated key establishment protocol for smart home environment. *IEEE Trans. Dependable Secur. Comput.* **2020**, *17*, 391–406. [CrossRef]
17. Dai, C.; Xu, Z. A secure three-factor authentication scheme for multi-gateway wireless sensor networks based on elliptic curve cryptography. *Ad Hoc Netw.* **2022**, *127*, 102768. [CrossRef]
18. Hu, H.; Liao, L.; Zhao, J. Secure authentication and key agreement protocol for cloud-assisted industrial internet of things. *Electronics* **2022**, *11*, 1652. [CrossRef]
19. Jiang, Q.; Zhang, N.; Ni, J.; Ma, J.; Ma, X.; Choo, K.K.R. Unified biometric privacy preserving three-factor authentication and key agreement for cloud-assisted autonomous vehicles. *IEEE Trans. Veh. Technol.* **2020**, *69*, 9390–9401. [CrossRef]
20. Zhao, J.; Li, Q.; Gong, Y.; Zhang, K. Computation Offloading and Resource Allocation For Cloud Assisted Mobile Edge Computing in Vehicular Networks. *IEEE Trans. Veh. Technol.* **2019**, *68*, 7944–7956. [CrossRef]
21. Dolev, D.; Yao, A. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [CrossRef]
22. Degefa, F.; Ryu, J.; Kim, H.; Won, D. MES-FPMIPv6: MIH-Enabled and enhanced secure Fast Proxy Mobile IPv6 handover protocol for 5G networks. *PLoS ONE* **2022**, *17*, e0262696. [CrossRef]
23. Jung, J.; Lee, D.; Lee, H.; Won, D. Security enhanced anonymous user authenticated key agreement scheme using smart card. *J. Electron. Sci. Technol.* **2018**, *16*, 45–49.
24. Mahmood, K.; Obaidat, M.S.; Shamshad, S.; Alenazi, M.J.; Kumar, G.; Anisi, M.H.; Conti, M. Cost-effective authenticated solution (CAS) for 6G-enabled artificial intelligence of medical things (AIoMT). *IEEE Internet Things J.* **2024**, *11*, 23977–23984. [CrossRef]
25. Chen, C.M.; Xiong, Z.; Wu, T.Y.; Kumari, S.; Alenazi, M.J. Protecting Virtual Economies: A Blockchain-based Anti-Phishing Authentication Protocol for Metaverse Applications. *IEEE Internet Things J.* **2025**, early access.

26. Kapoor, V.; Abraham, V.S.; Singh, R. Elliptic curve cryptography. *Ubiquity* **2008**, *9*, 1–8. [[CrossRef](#)]
27. Maarouf, A.; Sakr, R.; Elmougy, S. An offline direct authentication scheme for the internet of medical things based on elliptic curve cryptography. *IEEE Access* **2024**, *12*, 134902–134925. [[CrossRef](#)]
28. Lee, H.; Kang, D.; Lee, Y.; Won, D. Secure Three-Factor Anonymous User Authentication Scheme for Cloud Computing Environment. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 2098530. [[CrossRef](#)]
29. Fuller, B.; Reyzin, L.; Smith, A. When are fuzzy extractors possible? *IEEE Trans. Inf. Theory* **2020**, *66*, 5282–5298. [[CrossRef](#)]
30. An, J.; Choi, S.H. End-to-end Facial Recognition Deep Learning Model Specialized for Facial Angle using Gray Image. *IEIE Trans. Smart Process. Comput.* **2024**, *13*, 534–539. [[CrossRef](#)]
31. Gao, Y.; Al-Sarawi, S.F.; Abbott, D. Physical unclonable functions. *Nat. Electron.* **2020**, *3*, 81–91. [[CrossRef](#)]
32. Aldosary, A.; Tanveer, M. PAAF-SHS: PUF and authenticated encryption based authentication framework for the IoT-enabled smart healthcare system. *Internet Things* **2024**, *26*, 101159. [[CrossRef](#)]
33. Woo, N.; Kang, T.; Ryu, J. CESA: Chebyshev-Polynomials-Based Efficient and Secure Access Authentication Scheme for Both User Equipment and Massive Machine-Type-Communication Devices Over 5G Networks. *IEEE Internet Things J.* **2025**, *early access*.
34. Kim, K.; Ryu, J.; Lee, H.; Lee, Y.; Won, D. Distributed and federated authentication schemes based on updatable smart contracts. *Electronics* **2023**, *12*, 1217. [[CrossRef](#)]
35. Lee, H.; Ryu, J.; Won, D. Secure and anonymous authentication scheme for mobile edge computing environments. *IEEE Internet Things J.* **2023**, *11*, 5798–5815. [[CrossRef](#)]
36. Ryu, J.; Lee, H.; Lee, Y.; Won, D. SMASG: Secure mobile authentication scheme for global mobility network. *IEEE Access* **2022**, *10*, 26907–26919. [[CrossRef](#)]
37. Kang, T.; Woo, N.; Ryu, J. Enhanced lightweight medical sensor networks authentication scheme based on blockchain. *IEEE Access* **2024**, *12*, 35612–35629. [[CrossRef](#)]
38. Blanchet, B.; Smyth, B.; Cheval, V.; Sylvestre, M. ProVerif 2.00: Automatic Cryptographic Protocol Verifier. *User Man. Tutor.* **2018**, *16*, 5–16.
39. Kim, J.; Moon, J.; Jung, J.; Won, D. Security analysis and improvements of session key establishment for clustered sensor networks. *J. Sens.* **2016**, *2016*, 4393721. [[CrossRef](#)]
40. Ryu, J.; Kang, D.; Won, D. Improved secure and efficient Chebyshev chaotic map-based user authentication scheme. *IEEE Access* **2022**, *10*, 15891–15910. [[CrossRef](#)]
41. Kook, S.; Kim, K.; Ryu, J.; Lee, Y.; Won, D. Lightweight Hash-Based Authentication Protocol for Smart Grids. *Sensors* **2024**, *24*, 3085. [[CrossRef](#)]
42. Miao, J.; Wang, Z.; Wu, Z.; Ning, X.; Tiwari, P. A blockchain-enabled privacy-preserving authentication management protocol for Internet of Medical Things. *Expert Syst. Appl.* **2024**, *237*, 121329. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.