mathematics

MDPI

Article

Enhancing the Security: A Lightweight Authentication and Key Agreement Protocol for Smart Medical Services in the IoHT

Tsu-Yang Wu [1,2], Liyang Wang [1] and Chien-Ming Chen [2,*]

[1] College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China; wutsuyang@gmail.com (T.-Y.W.); liyang2021@sdust.edu.cn (L.W.)
[2] School of Artificial Intelligence (School of Future Technology), Nanjing University of Information Science & Technology, Nanjing 210044, China
* Correspondence: chienmingchen@ieee.org

Abstract: The Internet of Things (IoT) has witnessed significant growth with advancements in Internet and wireless technologies. In the medical field, the Internet of Health Things (IoHT) has emerged as an extension of the IoT, enabling the exchange of remote data and real-time monitoring of patients' health conditions. Through the IoHT, doctors can promptly provide diagnoses and treatment for patients. As patient data are transmitted over public channels, security issues may arise, necessitating security mechanisms. Recently, Amintoosi et al. proposed an authentication protocol for smart medical services in the IoHT. However, their protocol exhibited security weaknesses, including vulnerabilities to privileged insider attacks. To address the security concerns, we propose an enhanced authentication and key agreement protocol. The security of our protocol is rigorously analyzed using the Real-Or-Random model, informal security analysis, and the AVISPA tool. Finally, the results of our analysis demonstrate that our proposed protocol ensures sufficient security while maintaining a performance level similar to existing protocols.

Keywords: IoHT; authentication; key agreement; lightweight; cryptanalysis

check for updates

Citation: Wu, T.-Y.; Wang, L.; Chen, C.-M. Enhancing the Security: A Lightweight Authentication and Key Agreement Protocol for Smart Medical Services in the IoHT. *Mathematics* **2023**, *11*, 3701. https://doi.org/10.3390/math11173701

Academic Editor: Daniel-Ioan Curiac

Received: 6 August 2023
Revised: 22 August 2023
Accepted: 25 August 2023
Published: 28 August 2023

1. Introduction

The Internet of Things (IoT) [1–3] is a technology that enables the collection of real-time data and the connection of devices, thus serving as an infrastructure in people's lives. With the advancements in the Internet, mobile communication, and wireless technology, the IoT can be applied to various environments, including smart home [4], smart grid [5], Internet of Vehicles [6,7], and artificial intelligence [8,9]. These environments take advantage of the information-gathering features of the IoT to solve problems existing in real life, so as to bring more benefits and convenience to people's lives.

The Internet Health of Things (IoHT) [10–12] is an extension of the IoT specifically focused on healthcare. It combines modern communication and medical information technology to create a new mode of health management. The IoHT enables the real-time monitoring of patients' health data, reducing the repetitive workload for medical staff. Simultaneously, it allows medical professionals to provide timely diagnosis and treatment based on the collected data, as well as deliver preventive or proactive healthcare services at a lower cost. The architecture of the IoHT, as illustrated in Figure 1, includes three main entities: users (doctors/nurses), gateway, and sensor nodes. Sensor nodes are distributed among patients and are responsible for collecting various health data, such as electrocardiogram readings, body temperature measurements, and blood oxygen saturation levels. The gateway serves as a semi-trusted entity that facilitates the real-time transmission of the collected data between the sensor nodes and the users. Users are medical staff (doctors/nurses) who have the ability to access patients' health data, and use the collected data to analyze the condition and provide appropriate diagnosis and treatment plans for patients.

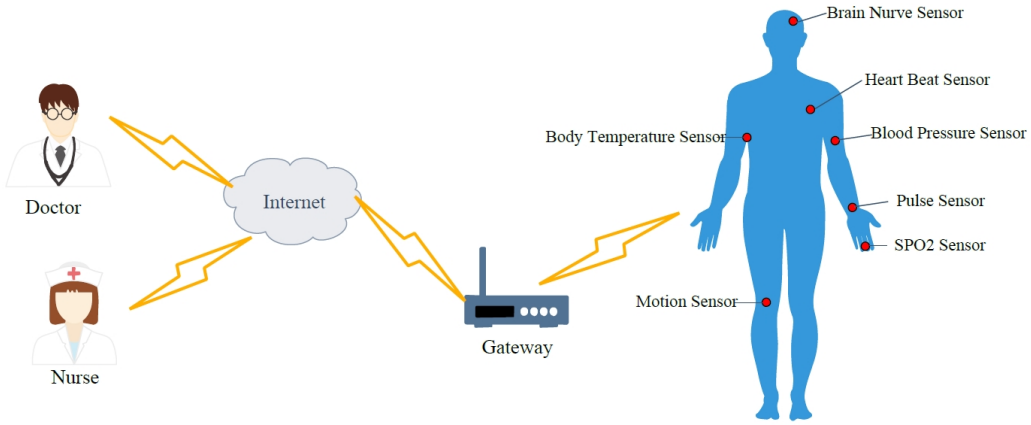


Figure 1. The architecture of IoHT.

The security of medical data in IoHT is of utmost importance due to the sensitivity of the information involved. If patient health data and diagnostic reports are stolen by attackers through public channels, it can lead to privacy breaches and potential security attacks such as impersonation [13,14], replay [15,16], and man-in-the-middle (MITM) [17,18] attacks. To address these security concerns, authentication and key agreement (AKA) techniques can be employed to achieve mutual authentication [19,20] between communication entities and establish session keys, ensuring secure communication in IoHT. According to Diffie et al.'s study [21], AKA protocols should follow some general principles when being designed. These principles include the ideas that authentication and key exchange need to be linked together, asymmetry should be exhibited in the protocol, messages should avoid being used repeatedly to prevent replay attacks, entities should incorporate appropriate random numbers into encryption operations, etc.

In recent years, several AKA protocols have been proposed specifically for healthcare applications based on IoT. Challa et al. [22] put forward a secure AKA protocol for medical wireless sensor networks based on elliptic curve cryptography (ECC) in 2018. Unfortunately, Soni et al. [23] found their protocol violated user anonymity and was subjected to session key disclosure attacks. As a result, Soni et al. proposed an enhanced AKA protocol specifically designed for healthcare systems. Unfortunately, Xu et al. [24] demonstrated that this enhanced protocol violated perfect forward secrecy (PFS) and could not resist offline password guessing (OPG) and sensor node capture (SNC) attacks. Qiu et al. [25] put forward a robust AKA protocol based on a telecare medicine system. However, Shamshad et al. [20] found that this protocol was vulnerable to privileged insider (PI) and OPG attacks. Consequently, Shamshad et al. devised a security-enhanced authentication protocol for healthcare services. Sharma and Kalra [26] presented a secure AKA protocol based on IoHT, demonstrating its resilience against multiple security attacks. Unfortunately, Azrour et al. [27] discovered that this protocol suffered from impersonation and OPG attacks. Similarly, Azrour et al. proposed an enhanced protocol for remote healthcare services based on cloud-based IoT. Aghili et al. [28] developed a lightweight AKA protocol for an e-health system based on IoT. However, Amintoosi et al. [29] demonstrated that this protocol violated PFS and was susceptible to SNC and impersonation attacks.

In 2020, Merabet et al. [30] introduced a novel mutual authentication protocol based on IoHT, ensuring secure communication between machines and the cloud. Kumari et al. [31] proposed an efficient AKA protocol for smart healthcare and cloud environments, utilizing ECC. However, Wu et al. [32] demonstrated that their protocol suffered from several security vulnerabilities, including impersonation, known session specific temporary information (KSSTI), and desynchronization attacks. Subsequently, Wu et al. proposed an alternative AKA protocol for smart healthcare, addressing the identified security issues. Hajian et al. [33] devised an attack-resilient protocol for Medical Internet of Things (MIoT) applications. Unfortunately, Yu et al. [17] found that this protocol was susceptible to

MITM, impersonation, and session key disclosure (SKD) attacks. Consequently, Yu et al. proposed an enhanced AKA protocol specifically designed for the MIoT environment, seeking to improve its security. Alladi et al. [34] designed a two-way AKA protocol for the healthcare environment, incorporating physical unclonable functions to enhance data security. Shuai et al. [35] put forward a robust AKA protocol for a private healthcare system, incorporating three factors to strengthen security. However, Xie et al. [36] identified that their protocol violated PFS and was vulnerable to PI attacks. Similarly, Xie et al. proposed a privacy-protected AKA protocol for IoT environments. Agrahari et al. [37] devised an AKA protocol for healthcare monitoring systems, ensuring the security of patient data during transmission. Al-Saggaf et al. [38] proposed a two-factor AKA protocol based on IoHT, utilizing quantum computing for enhanced security.

According to previous research, ensuring the security of medical data and user privacy in the IoHT is crucial. In light of this, Amintoosi et al. [29] proposed an authentication protocol for smart medical services, which not only achieves mutual authentication between communication entities, but also facilitates the establishment of session keys between them to ensure secure communication. However, during our investigation, we identified security vulnerabilities in their protocol. To address these security concerns, we have developed an enhanced AKA protocol specifically tailored for the IoHT environment. Our protocol aims to provide robust security measures to ensure the secure transmission of medical data and protect user privacy. The main contributions of our paper are summarized as follows:

(1) We conducted a thorough review of Amintoosi et al.'s protocol and identified certain security weaknesses, particularly PI attacks.
(2) In response to the identified weaknesses, we propose an enhanced AKA protocol for smart medical services in the IoHT. Our protocol utilizes lightweight primitives and facilitates the establishment of session keys between doctors and sensor nodes with the assistance of gateways, ensuring secure communication.
(3) To validate the security of our proposed protocol, we conducted a rigorous analysis using the Real-Or-Random (ROR) model, informal security analysis, and the automated validation of Internet security protocols and applications (AVISPA) tool.
(4) Finally, we compare the performance and security of our proposed protocol with existing protocols. The comparison results demonstrate that our proposed protocol offers sufficient security with comparable performance to other protocols in the IoHT environment.

The structure of this paper is organized as follows. In Section 2, we review and analyze Amintoosi et al.'s protocol. We present the specific process and design details of the proposed enhanced security AKA protocol in the IoHT in Section 3. In Section 4, we demonstrate the security of our protocol through the ROR model, informal security analysis, and the AVISPA tool. A comparison of the proposed protocol with existing AKA protocols in the IoHT is involved in Section 5 and the conclusion is made in Section 6.

2. Review and Cryptanalysis of Amintoosi et al.'s Protocol [29]

2.1. Review of Amintoosi et al.'s Protocol [29]

Here, we only review the "registration" and the "login and authentication" phases of Amintoosi et al.'s protocol. Their protocol involves user, medical server, and sensor node. The notations used in this paper are shown in Table 1.

Table 1. Notations.

Notations	Description
U_i	i-th user
ID_i, TID_i	U_i's identity and pseudo-identity
PW_i	Password of U_i
MS	Medical server
SID	MS's identity
s	Private key of MS
GWN	Gateway node
k	GWN's private key
S_j	j-th sensor
IDS_j	S_j's identity
X_j	Secret key of S_j
SK	Session key
T_i	Timestamp
$a_i, b_i, c_j, r_i, r_j, r_s$	The random numbers
$\oplus$	Bitwise XOR
$h(.)$	Secure-hash function
$\|\|$	Concatenation operation

2.1.1. Registration

The registration phase is divided into two phases, which are the user and sensor node registration phases.

User registration phase. The process of user registration is depicted in Figure 2, with the specific steps outlined as follows.

(1) User U_i chooses ID_i, PW_i, and a_i, and calculates $UM_i = h(ID_i \parallel PW_i \parallel a_i)$. Next, U_i sends $\{UM_i, ID_i\}$ to MS via a secure channel.

(2) On receiving the $\{UM_i, ID_i\}$, MS firstly searches for the ID_i stored in the database. If the ID_i exists, the U_i should be asked to send a new ID_i. Otherwise, MS selects b_i to compute $TID_i = h(b_i \parallel ID_i)$, $UN_i = h(b_i \parallel ID_i \parallel TID_i)$, $UO_i = h(SID \parallel s \parallel b_i)$, $UP_i = UO_i \oplus UM_i$, $UQ_i = h(UN_i \parallel UO_i)$, and $i = i + 1$. Then, MS stores $\{b_i, UP_i, UQ_i\}$ in smart card, and stores $\{UP_i, UN_i, UQ_i, ID_i\}$ in its database. Finally, MS transmits smart card to U_i.

(3) When U_i receives the smart card, $\{a_i\}$ is added to it.

User U_i		Medical server MS
Select ID_i, PW_i, a_i		
Calculate		
$UM_i = h(ID_i \parallel PW_i \parallel a_i)$		
	$\{UM_i, ID_i\}$ $\longrightarrow$	
		Select b_i and compute
		$TID_i = h(b_i \parallel ID_i)$
		$UN_i = h(b_i \parallel ID_i \parallel TID_i)$
		$UO_i = h(SID \parallel s \parallel b_i)$
		$UP_i = UO_i \oplus UM_i$
		$UQ_i = h(UN_i \parallel UO_i)$
		$i = i + 1$
		Store $\{b_i, UP_i, UQ_i\}$ in smart card
		Store $\{UP_i, UN_i, UQ_i, ID_i\}$ in its database
	$\longleftarrow$ smart card	
Add $\{a_i\}$ in smart card		
Smart card $\{b_i, UP_i, UQ_i, a_i\}$		

Figure 2. U_i's registration phase of Amintoosi et al.'s protocol.

Sensor registration phase. Figure 3 depicts the sensor registration process, and the subsequent detailed steps are as follows.

(1) Sensor S_j selects IDS_j and c_j to calculate $SM_j = h(IDS_j \parallel X_j \parallel c_j)$, and sends $\{SM_j, c_j, IDS_j\}$ to MS via secure channel.

(2) When MS receives the $\{SM_j, c_j, IDS_j\}$, it computes $SN_j = h(IDS_j \parallel s \parallel c_j)$, and $j = j + 1$. Then, MS stores $\{SM_j, IDS_j, c_j\}$ in database, and transmits $\{SN_j\}$ to S_j.

(3) S_j receives the $\{SN_j\}$, and stores $\{SN_j, c_j\}$ in its memory.

Sensor S_j		Medical server MS
Select IDS_j, c_j		
Compute $SM_j = h(IDS_j \parallel X_j \parallel c_j)$		
	$\{SM_j, c_j, IDS_j\} \rightarrow$	
		$SN_j = h(IDS_j \parallel s \parallel c_j)$
		$j = j + 1$
		Store $\{SM_j, IDS_j, c_j\}$ in database
	$\leftarrow \{SN_j\}$	
Store $\{SN_j, c_j\}$ in memory		

Figure 3. S_j's registration phase of Amintoosi et al.'s protocol.

2.1.2. Login and Authentication Phase

The login and authentication phase process is illustrated in Figure 4, as shown below in the specific steps.

(1) First, U_i inputs ID_i^*, PW_i^*, computes $UM_i^* = h(ID_i^* \parallel PW^* \parallel a_i)$, $UO_i^* = UP_i \oplus UM_i^*$, $TID_i^* = h(b_i \parallel ID_i^*)$, $UN_i^* = h(b_i \parallel ID_i^* \parallel TID_i^*)$, $UQ_i^* = h(UN_i^* \parallel UO_i^* \parallel UM_i^*)$ and checks $UQ_i^* \stackrel{?}{=} UQ_i$. If it holds, U_i chooses r_i and T_1 to compute $W_1 = h(UN_i \parallel UO_i \parallel T_1) \oplus r_i$, $V_1 = h(UN_i \parallel UO_i \parallel r_i)$. Finally, U_i transmits message $M_1 = \{W_1, T_1, b_i, V_1\}$ to MS via public channel.

(2) After MS receives M_1, it verifies freshness of T_1 by calculating $|T_1 - T_c| \leq \Delta T$. Then, MS computes $UO_i = h(SID \parallel s \parallel b_i)$, $r_i = h(UN_i \parallel UO_i \parallel T_1) \oplus W_1$, $V_1^* = h(UN_i \parallel UO_i \parallel r_i)$ and checks $V_1^* \stackrel{?}{=} V_1$. If the two values do not correspond, the authentication process is suspended. Otherwise, MS selects r_s, and calculates $W_2 = h(r_i \parallel IDS_j \parallel c_j) \oplus r_s$, $SN_j = h(IDS_j \parallel s \parallel c_j)$, $V_2 = h(SN_j \parallel SM_j \parallel r_s)$. Finally, MS retrieves T_2 and transmits the message $M_2 = \{r_i, W_2, V_2, T_2\}$ to S_j.

(3) On S_j receiving the $\{r_i, W_2, V_2, T_2\}$, it first verifies T_2. Next, S_j computes $r_s = W_2 \oplus h(r_i \parallel IDS_j \parallel c_j)$, $SM_j^* = h(IDS_j \parallel X_j \parallel c_j)$, $V_2^* = h(SN_j \parallel SM_j^* \parallel r_s)$ and checks $V_2^* \stackrel{?}{=} V_2$. If the two values are equal, S_j chooses r_j, and computes $SK = h(r_s \parallel IDS_j \parallel r_j)$, $W_3 = h(SM_j \parallel c_j \parallel SN_j) \oplus r_j$, $V_3 = h(SK \parallel SN_j \parallel T_3)$. At last, S_j transmits the message $M_3 = \{W_3, V_3, T_3\}$ to MS.

(4) MS verifies the T_3 after receiving the M_3. Next, MS computes $UM_i = UO_i \oplus UP_i$, $r_j = h(SM_j \parallel c_j \parallel SN_j) \oplus W_3$, $SK = h(r_s \parallel IDS_j \parallel r_j)$, $V_3^* = h(SK \parallel SN_j \parallel T_3)$ and checks $V_3^* \stackrel{?}{=} V_3$. If the two values are equal, the MS selects a timestamp T_4 and computes $W_4 = UM_i \oplus r_j$, $W_5 = UM_i \oplus r_s$, $V_4 = h(W_4 \parallel W_5 \parallel UM_i \parallel T_4)$. Next, MS transmits message $M_4 = \{W_4, W_5, V_4, T_4\}$ to U_i.

(5) U_i verifies the T_4 after receiving the M_4. If it is fresh, U_i computes $V_4^* = h(W_3 \parallel W_4 \parallel UM_i \parallel T_4)$ and checks $V_4^* \stackrel{?}{=} V_4$. If the two values are equal, U_i computes $r_j = UM_i \oplus W_4$, $r_s = UM_i \oplus W_5$, and then computes $SK = h(r_s \parallel IDS_j \parallel r_j)$.

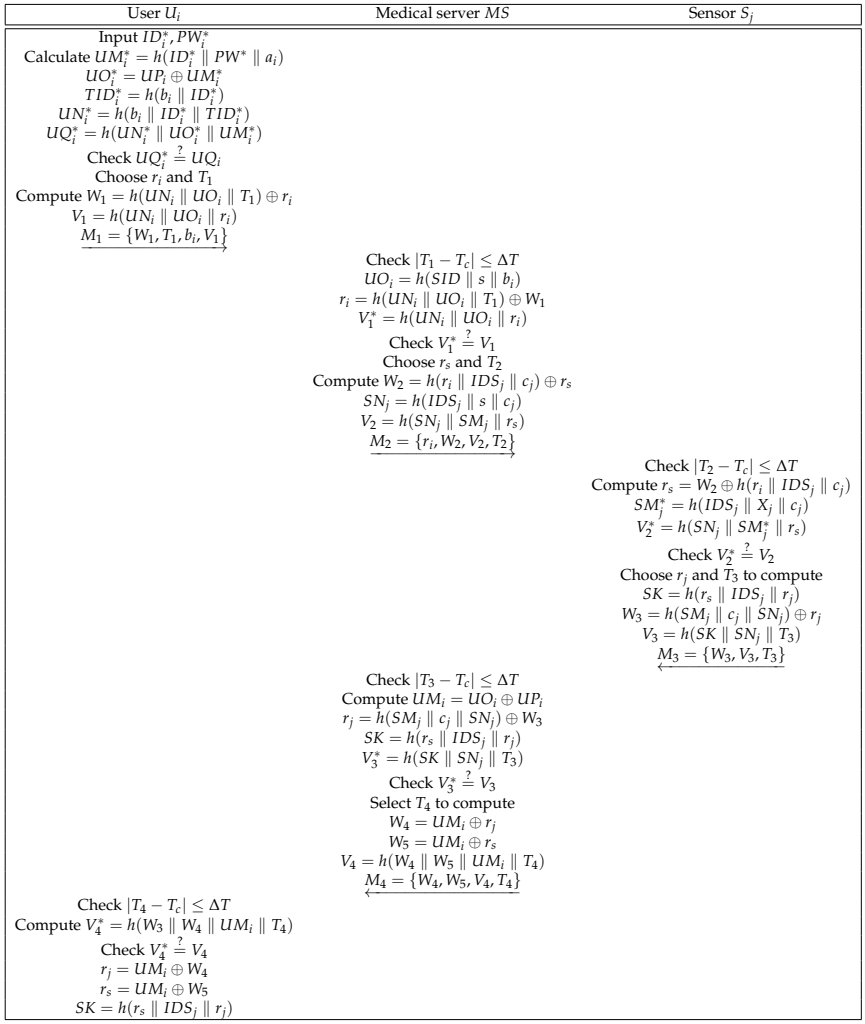


Figure 4. Authentication phase of Amintoosi et al.'s protocol.

2.2. Cryptanalysis of Amintoosi et al.'s Protocol

In this section, we point out that Amintoosi et al.'s protocol has certain security weaknesses, particularly PI attacks.

Attacker Model. According to the Dolev-Yao (DY) [39] and Canetti and Krawczyk (CK) [40] models, we define the following capabilities for an attacker ($\mathcal{A}$) to follow.

(1) $\mathcal{A}$ possesses the capability to intercept, monitor, and manipulate messages that are transmitted through the public channel.
(2) The medical server may have a malicious insider named $\mathcal{A}$ who can acquire data from the database.
(3) $\mathcal{A}$ can utilize power analysis to obtain the data in the user's smart card or smart device.
(4) $\mathcal{A}$ can obtain temporary information value and long-term key.

2.2.1. Privileged Insider Attacks

Assume $\mathcal{A}$ obtains the data $\{SM_j, IDS_j, c_j\}$ from MS. Through the following steps, $\mathcal{A}$ can compute the SK successfully. The process of the attack method is depicted in Figure 5, showing only the important portion. The parts marked in red indicate the data and messages obtained by $\mathcal{A}$, while the red boxes represent $\mathcal{A}$'s computational steps.

(1) $\mathcal{A}$ can eavesdrop on the messages $M_2 = \{r_i, W_2, V_2, T_2\}$, and $M_3 = \{W_3, V_3, T_3\}$ on public channel.

(2) Next, $\mathcal{A}$ can compute $r_s = W_2 \oplus h(r_i \parallel IDS_j \parallel c_j)$ and $r_j = h(SM_j \parallel c_j \parallel SN_j) \oplus W_3$, respectively.
(3) At last, $\mathcal{A}$ can compute $SK = h(r_s \parallel IDS_j \parallel r_j)$.

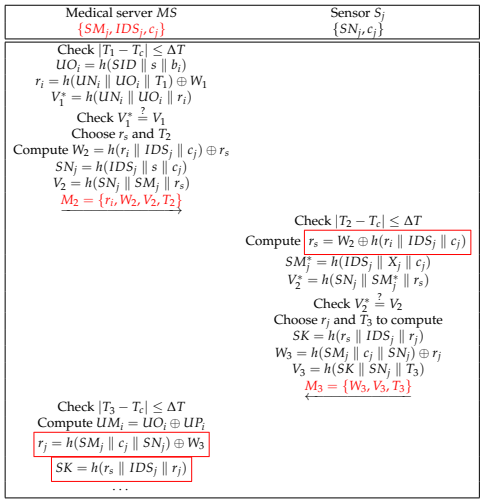


Figure 5. PI attacks in Amintoosi et al.'s protocol.

2.2.2. Incorrectness of *SK*

In the authentication phase of Amintoosi et al.'s protocol [29], *MS* first transmits the M_4 to U_i. On receiving the M_4, U_i calculates numbers r_j and r_s to establish the *SK*, where $SK = h(r_s \parallel IDS_j \parallel r_j)$. The IDS_j is stored in the database of *MS*, and the *MS* does not transmit the value IDS_j to U_i. Thus, the U_i cannot know the value IDS_j, and cannot establish the *SK*.

3. The Proposed Protocol

In response to the identified weaknesses of Amintoosi et al.'s protocol, we propose an enhanced AKA protocol in the IoHT (shown in Figure 1). The entities involved in the protocol include U_i, *GWN*, and S_j. Here, we use *GWN* to replace the *MS* in Amintioosi et al.'s protocol, because the functions of the *MS* and the *GWN* are the same, and the *GWN* is commonly used in the IoHT environment. The initialization, registration, and login and authentication phases are included in our proposed protocol.

3.1. Initialization and Registration Phases

3.1.1. Initialization Phase

The smart device, gateway, and sensor nodes need to write basic arithmetic functions, such as $h(.)$, $\oplus$, and $||$. Here, *GWN* is a semi-trusted entity, which means that it possesses the ability to engage in misconduct, yet lacks the capacity to collaborate with other entities. Moreover, the *GWN* chooses k as its private key, and is responsible for the pre-deployment of the sensor nodes. The sensor pre-deployment process is shown in Figure 6. The specific steps are described below.

(1) S_j chooses its IDS_j and a random number c_j, and sends $\{IDS_j, c_j\}$ to *GWN* via a secure channel.
(2) When *GWN* receives the $\{IDS_j, c_j\}$, it calculates $SM_j = h(IDS_j \parallel c_j \parallel k)$, $SN_j = h(IDS_j \parallel k) \oplus SM_j$. Then, *GWN* stores $\{IDS_j, SN_j\}$ in its database. Finally, *GWN* transmits $\{SM_j\}$ to S_j.
(3) On receiving $\{SM_j\}$, S_j computes $SO_j = h(IDS_j \parallel c_j) \oplus SM_j$. Next, S_j stores $\{c_j, SO_j\}$ in its memory.

Sensor S_j		Gateway node GWN
Select IDS_j, c_j		
	$\{IDS_j, c_j\}$ →	
		Compute $SM_j = h(IDS_j \parallel c_j \parallel k)$ $SN_j = h(IDS_j \parallel k) \oplus SM_j$ Store $\{IDS_j, SN_j\}$ in database
	← $\{SM_j\}$	
$SO_j = h(IDS_j \parallel c_j) \oplus SM_j$ Store $\{c_j, SO_j\}$ in memory		

Figure 6. Pre-deployment of sensor node.

3.1.2. Doctor Registration Phase

In this phase, doctors need to register with the GWN to become legitimate users U_i. The doctor registration process is described in Figure 7, and the specific steps are as follows.

(1) First, U_i chooses ID_i, PW_i, a_i, and calculates $TID_i = h(ID_i \parallel PW_i \parallel a_i)$. Next, U_i transmits $\{TID_i, a_i\}$ to GWN via secure channel.

(2) When GWN receives the $\{TID_i, a_i\}$, it chooses b_i to compute $UM_i = h(TID_i \parallel b_i \parallel a_i)$, $UN_i = a_i \oplus h(b_i \parallel k)$. Then GWN stores $\{TID_i, b_i, UN_i\}$ in database, and transmits $\{UM_i\}$ to U_i.

(3) On receiving $\{UM_i\}$, U_i calculates $RPW_i = h(PW_i \parallel a_i)$, $UO_i = a_i \oplus h(ID_i \parallel RPW_i)$, $UP_i = h(TID_i \parallel RPW_i \parallel a_i)$, $UQ_i = UM_i \oplus h(a_i \parallel RPW_i)$. Finally, U_i stores $\{UO_i, UP_i, UQ_i\}$ in smart device.

Doctor U_i		Gateway node GWN
Select ID_i, PW_i, a_i Compute $TID_i = h(ID_i \parallel PW_i \parallel a_i)$		
	$\{TID_i, a_i\}$ →	
		Select b_i and compute $UM_i = h(TID_i \parallel k \parallel a_i)$ $UN_i = a_i \oplus h(b_i \parallel k)$ Store $\{TID_i, b_i, UN_i\}$ in database
	← $\{UM_i\}$	
$RPW_i = h(PW_i \parallel a_i)$ $UO_i = a_i \oplus h(ID_i \parallel RPW_i)$ $UP_i = h(TID_i \parallel RPW_i \parallel a_i)$ $UQ_i = UM_i \oplus h(a_i \parallel RPW_i)$ Store $\{UO_i, UP_i, UQ_i\}$ in smart device		

Figure 7. Doctor registration phase.

3.2. Login and Authentication Phase

In this section, the U_i, GWN, and S_j achieve mutual authentication, and the U_i and S_j successfully establish a SK with the assistance of the GWN. The login and authentication process is depicted in Figure 8, and the detailed login and authentication steps are as follows.

(1) First, U_i inputs ID_i, PW_i, and calculates $a_i^* = UO_i \oplus h(ID_i^* \parallel PW_i^*)$, $TID_i^* = h(ID_i^* \parallel PW_i^* \parallel a_i^*)$, $RPW_i^* = h(PW_i^* \parallel a_i)$, $UP_i^* = h(TID_i^* \parallel RPW_i^* \parallel a_i^*)$. Then, U_i checks $UP_i^* \stackrel{?}{=} UP_i$. If it is not equal, U_i login fails. Otherwise, U_i computes $UM_i = UQ_i \oplus h(a_i \parallel RPW_i)$, and chooses r_i and its IDS_j. Next, U_i calculates $R_i = h(ID_i \parallel a_i \parallel r_i)$, $W_1 = R_i \oplus h(a_i \parallel UM_i)$, $W_2 = IDS_j \oplus h(UM_i \parallel R_i)$, and retrieves the T_1 to compute $V_1 = h(TID_i \parallel IDS_j \parallel R_i \parallel T_1)$. Finally, U_i sends message $M_1 = \{TID_i, W_1, W_2, V_1, T_1\}$ to GWN via public channel.

(2) Following the receipt of message M_1, GWN initially verifies that timestamp T_1 is fresh. Next, GWN retrieves $\{b_i, UN_i\}$ from the database using TID_i and calculates $a_i = UN_i \oplus h(b_i \parallel k)$, $UM_i = h(TID_i \parallel k \parallel a_i)$, $R_i = W_1 \oplus h(a_i \parallel UM_i)$, $IDS_j = W_2 \oplus h(UM_i \parallel R_i)$, $V_1^* = h(TID_i \parallel IDS_j \parallel R_i \parallel T_1)$ and checks $V_1^* \stackrel{?}{=} V_1$.

If they are equal, GWN retrieves $\{SN_i\}$ according to IDS_j and computes $SM_j = SN_j \oplus h(IDS_j \parallel k)$, $W_3 = R_i \oplus h(IDS_j \parallel SM_j)$. At last, GWN retrieves the current timestamp T_2 to compute $V_2 = h(TID_i \parallel R_i \parallel SM_j \parallel T_2)$ and transmits message $M_2 = \{TID_i, W_3, V_2, T_2\}$ to S_j.

(3) When S_j receives the M_2, it checks freshness of T_2 by computing $|T_2 - T_c| \leq \Delta T$. Then, S_j calculates $SM_j = SO_j \oplus h(IDS_j \parallel c_j)$, $R_i = W_3 \oplus h(IDS_j \parallel SM_j)$, $V_2^* = h(TID_i \parallel R_i \parallel SM_j \parallel T_2)$ and checks $V_2^* \stackrel{?}{=} V_2$. If it holds, S_j chooses r_j to calculate $R_j = h(IDS_j \parallel c_j \parallel r_j)$, $SK = h(R_i \parallel R_j)$, $W_4 = R_j \oplus h(IDS_j \parallel SM_j)$. Finally, S_j retrieves T_3 to compute $V_3 = h(R_j \parallel SM_j \parallel T_3)$ and transmits message $M_3 = \{W_4, V_3, T_3\}$ to GWN.

(4) When GWN receives the M_3, it verifies the freshness of T_3. Next, GWN computes $R_j = W_4 \oplus h(IDS_j \parallel SM_j)$, $V_3^* = h(R_j \parallel SM_j \parallel T_3)$ and checks $V_3^* \stackrel{?}{=} V_3$. If $V_3^* = V_3$, GWN computes $W_5 = R_j \oplus h(a_i \parallel R_i)$ and retrieves current timestamp T_4 to calculate $V_4 = h(R_i \parallel UM_i \parallel R_j \parallel T_4)$. Finally, GWN sends message $M_4 = \{W_5, V_4, T_4\}$ to U_i.

(5) U_i verifies freshness of the T_4 after receiving M_4. Then, U_i computes $R_j = W_5 \oplus h(a_i \parallel R_i)$, $V_4^* = h(R_i \parallel UM_i \parallel R_j \parallel T_4)$ and checks $V_4^* \stackrel{?}{=} V_4$. If it holds, U_i computes $SK = h(R_i \parallel R_j)$, which means that the U_i and S_j successfully establish a SK with the assistance of the GWN.

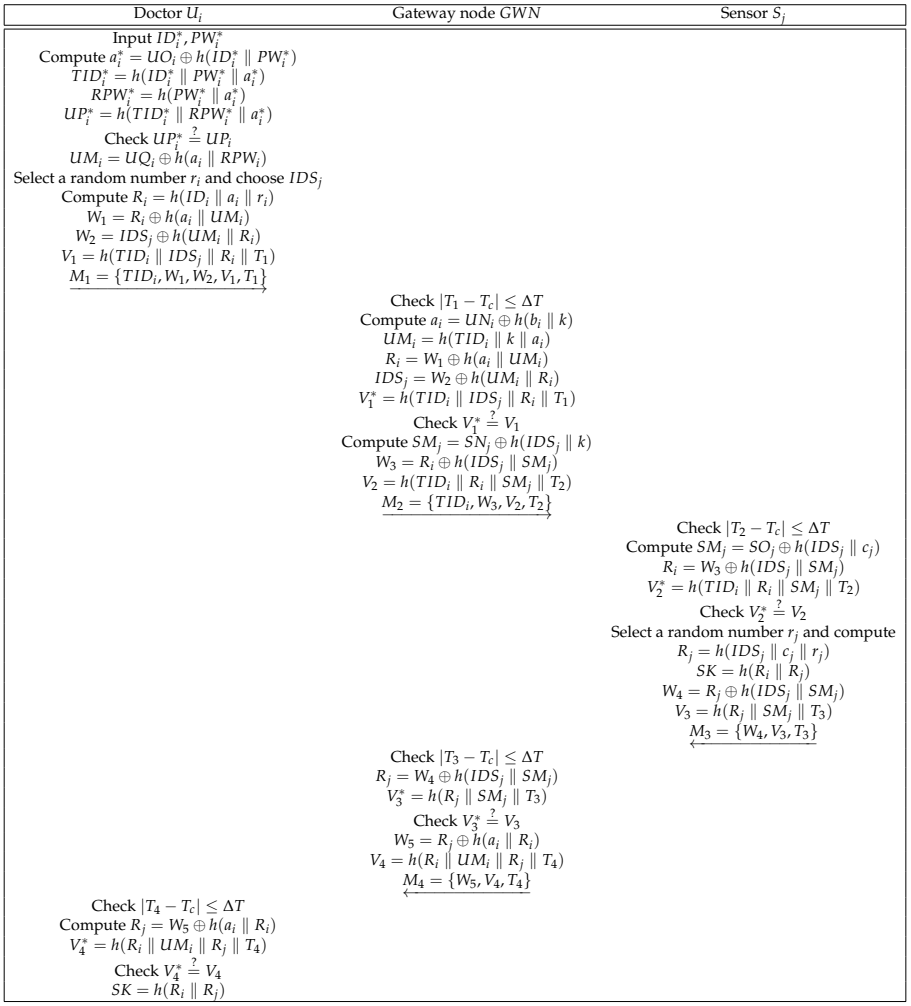


Figure 8. Login and authentication phase.

4. Security Analysis

4.1. Formal Security Analysis

We show the security of our protocol using the well-known ROR model [41–43]. Real attacks are simulated in this model through a series of rounds of games.

4.1.1. Security Model

Three entities are included in our proposed protocol: U_i, GWN, and S_j. We use $I^x_{U_i}$ to represent the x-th user instance, I^y_{GWN} represents the y-th gateway instance, and $I^z_{S_j}$ represents the z-th sensor node instance. Here, we define that $\mathcal{A}$ has certain capabilities in different games, but needs to follow the following queries.

(1) $Execute(\mathcal{E})$: This query means that $\mathcal{A}$ can intercept messages on the public channel, where $\mathcal{E} = \{I^x_{U_i}, I^y_{GWN}, I^z_{S_j}\}$.
(2) $Send(\mathcal{E}, M_i)$: $\mathcal{A}$ is able to acquire the response from $\mathcal{E}$ subsequent to transmitting message M_i to $\mathcal{E}$.
(3) $Hash(string)$: $\mathcal{A}$ may enter a $string$ to obtain its hash value by performing this query.
(4) $Corrupt(\mathcal{E})$: This query gives $\mathcal{A}$ access to the long-term key or temporary information of $\mathcal{E}$.
(5) $Test(\mathcal{E})$: The $\mathcal{A}$ would verify the validity of the SK by flipping a coin c. When $c = 1$, $\mathcal{A}$ obtains the SK. Otherwise, $\mathcal{A}$ obtains the random string.

4.1.2. Security Proof

Theorem 1. *The advantage that $\mathcal{A}$ breaking the proposed protocol ($\mathcal{P}$) in polynomial time ξ is $Adv^{\mathcal{P}}_{\mathcal{A}}(\xi) \leq \frac{q_h^2}{|Hash|} + 2C' \cdot q_s^{s'}$ under ROR model. Here, q_h, q_s, $|Hash|$ denote the hash query, send query, and the space of the hash function, respectively. In addition, C' and s' are two constants.*

Proof. We define four games GM_0-GM_3 to prove the proposed protocol's security, and these games simulate the real process of $\mathcal{A}$ attacking the protocol. Here, $Succ^{GM_i}_{A}(\xi)$ indicates that the $\mathcal{A}$ wins the i-th game, and $Adv^{\mathcal{P}}_{\mathcal{A}}$ is defined as the advantage of $\mathcal{A}$ breaking the protocol. The $\mathcal{A}$ simulates detailed queries as shown in Table 2. The following are the detailed processes in the proof.

GM_0: In GM_0, the $\mathcal{A}$ performs real attacks to break the proposed protocol. The $\mathcal{A}$ starts the game by flipping the c. Hence, we have

$$Adv^{\mathcal{P}}_{\mathcal{A}}(\xi) = |2Pr[Succ^{GM_0}_{\mathcal{A}}(\xi)] - 1|. \tag{1}$$

GM_1: In GM_1, $\mathcal{A}$ can eavesdrop on the transmitted messages $M_1 = \{TID_i, W_1, W_2, V_1, T_1\}$, $M_2 = \{TID_i, W_3, V_2, T_2\}$, $M_3 = \{W_4, V_3, T_3\}$ and $M_4 = \{W_5, V_4, T_4\}$ by executing the $Execute()$ query. After GM_1, $\mathcal{A}$ validates the $SK = h(R_i \parallel R_j)$ through executing the $Test()$ query. Since $\mathcal{A}$ cannot obtain the values R_i and R_j, $\mathcal{A}$ cannot compute the SK. Therefore, the result of GM_1 is no different from GM_0.

$$Pr[Succ^{GM_1}_{\mathcal{A}}(\xi)] = Pr[Succ^{GM_0}_{\mathcal{A}}(\xi)]. \tag{2}$$

GM_2: The $Send()$ and $Hash()$ queries are added to GM_2. The $\mathcal{A}$ wants to tamper with the eavesdropped messages, but the authentication values V_1, V_2, V_3, and V_4 in the message are composed of private values and are protected by hash function. Thus, since $\mathcal{A}$ cannot obtain the private value and cannot crack the hash function, the intercepted message cannot be tampered with. Furthermore, no hash collision occurs because each session's random numbers are distinct. Hence, in accordance with the birthday paradox, we have

$$|Pr[Succ^{GM_2}_{\mathcal{A}}(\xi)] - Pr[Succ^{GM_1}_{\mathcal{A}}(\xi)]| \leq \frac{q_h^2}{2|Hash|}. \tag{3}$$

GM_3: In GM_3, $\mathcal{A}$ obtains the data $\{UO_i, UP_i, UQ_i\}$ in the smart device by executing the Corrupt ($I^x_{U_i}$) query. Then, $\mathcal{A}$ utilizes these data and intercepted messages to attempt to deduce the correct PW_i. Since $\mathcal{A}$ cannot obtain the values RPW_i and a_i, $\mathcal{A}$ cannot compute correct UP_i and cannot obtain the PW_i, where $UP_i = h(TID_i \parallel RPW_i \parallel a_i)$. From Zipf's law [44], we can obtain

$$|Pr[Succ_{\mathcal{A}}^{GM_3}(\xi)] - Pr[Succ_{\mathcal{A}}^{GM_2}(\xi)]| \leq C' \cdot q_{send}^{s'}. \tag{4}$$

Finally, $\mathcal{A}$ wants to win the game by guessing bit c to obtain the correct SK. Thus, we can obtain

$$Pr[Succ_{\mathcal{A}}^{GM_3}(\xi)] = \frac{1}{2}. \tag{5}$$

According to GM_0 to GM_3, we have

$$\begin{aligned} \frac{Adv_{\mathcal{A}}^{\mathcal{P}}(\xi)}{2} &= |Pr[Succ_{\mathcal{A}}^{GM_0}(\xi)] - \frac{1}{2}| \\ &= |Pr[Succ_{\mathcal{A}}^{GM_0}(\xi)] - Pr[Succ_{\mathcal{A}}^{GM_3}(\xi)]| \\ &= |Pr[Succ_{\mathcal{A}}^{GM_1}(\xi)] - Pr[Succ_{\mathcal{A}}^{GM_3}(\xi)]| \\ &\leq \sum_{i=0}^{2} |Pr[Succ_{\mathcal{A}}^{GM_{i+1}}(\xi)] - Pr[Succ_{\mathcal{A}}^{GM_i}(\xi)]| \\ &= \frac{q_h^2}{2|Hash|} + C' \cdot q_{send}^{s'}. \end{aligned} \tag{6}$$

Thus, we can obtain

$$Adv_{\mathcal{A}}^{\mathcal{P}}(\xi) \leq \frac{q_h^2}{|Hash|} + 2C' \cdot q_{send}^{s'}. \tag{7}$$

□

Table 2. Simulation of queries.

Query	Description
$Send(\mathcal{E}, M_i)$	For $Send(I^x_{U_i}, \text{start})$. Assume $I^x_{U_i}$ is in a normal state and selects r_i, IDS_j, and T_1 to compute $R_i = h(ID_i \parallel a_i \parallel r_i), W_1 = R_i \oplus h(a_i \parallel UM_i), W_2 = IDS_j \oplus h(UM_i \parallel R_i)$, $V_1 = h(TID_i \parallel IDS_j \parallel R_i \parallel T_1)$. Next, the query returns the $M_1 = \{PID_i, W_1, W_2, V_1, T_1\}$.
	On $Send(I^y_{GWN}, (PID_i, W_1, W_2, V_1, T_1))$. Assume that I^y_{GWN} computes UM_i, R_i, IDS_j and checks V_1 in a normal state. Next, I^y_{GWN} calculates SM_j, W_3, V_2. Then, I^y_{GWN} selects T_2. The query is answered by $M_2 = \{TID_i, W_3, V_2, T_2\}$.
	For $Send(I^z_{S_j}, (TID_i, W_3, V_2, T_2))$. On receiving the message $\{TID_i, W_3, V_2, T_2\}$, $I^z_{S_j}$ computes SM_j, R_i, and checks the V_2. Then, $I^z_{S_j}$ calculates R_j, SK, W_4, V_3. Next, $I^z_{S_j}$ returns the output $M_3 = \{W_4, V_3, T_3\}$.
	For $Send(I^y_{GWN}, (W_4, V_3, T_3))$. Assume that I^y_{GWN} computes R_j, and checks V_3 in a normal state. If the V_3 holds, I^y_{GWN} calculates W_5, V_4 and selects T_4. Then, the query returns the $M_4 = \{W_5, V_4, T_4\}$.
	On $Send(I^x_{U_i}, W_5, V_4, T_4)$. Upon receiving the message (W_5, V_4, T_4), $I^x_{U_i}$ computes R_j and checks V_4. If the V_4 is correct, $I^x_{U_i}$ computes SK, which means that the $I^x_{U_i}$ accepts and terminates.
$Execute(\mathcal{E})$	Continue to use $Send$ queries to simulate the process for $Execute(\mathcal{E})$. $(TID_i, W_1, W_2, V_1, T_1) \longleftarrow Send(I^x_{U_i}, \text{start})$, $(TID_i, W_3, V_2, T_2) \longleftarrow Send(I^y_{GWN}, (TID_i, W_1, W_2, V_1, T_1))$, $(W_4, V_3, T_3) \longleftarrow Send(I^z_{S_j}, (TID_i, W_3, V_2, T_2))$, $(W_5, V_4, T_4) \longleftarrow Send(I^y_{GWN}, (W_4, V_3, T_3))$. The query returns $(TID_i, W_1, W_2, V_1, T_1)$, (TID_i, W_3, V_2, T_2), (W_4, V_3, T_3) and (W_5, V_4, T_4).
$Corrupt(\mathcal{E})$	If the $I^x_{U_i}$ is accepted, this query outputs $\{UO_i, UP_i, UQ_i\}$ in the smart device.
$Test(\mathcal{E})$	Flip the coin c. If the result is 1, the SK will be returned. Otherwise, a random string of the same length as SK will be returned.

4.2. Informal Security Analysis

4.2.1. Perfect Forward Secrecy (PFS)

We use two methods to show that the proposed protocol ensures PFS.

Method 1: Suppose $\mathcal{A}$ can obtain the k of GWN, and attempts to calculate the SK. First, $\mathcal{A}$ needs to calculate the value R_i, IDS_j and R_j, where $R_i = W_1 \oplus h(a_i \parallel UM_i)$ and $R_j = W_4 \oplus h(IDS_j \parallel SM_j)$. Then $\mathcal{A}$ uses these values to calculate the SK. Since $\mathcal{A}$ cannot obtain a_i, UM_i and SM_j, $\mathcal{A}$ cannot calculate the SK.

Method 2: We use Ge et al.'s method [45] to demonstrate that $\mathcal{A}$ cannot calculate the SK. The specific proof steps are as follows.

(1) First, the composition of the session key requires variables $\{R_i, R_j\}$, where $SK = h(R_i \parallel R_j)$. Based on the rules of Ge et al. [45], we add these variables around SK and use arrows to point to SK. Then, we proceed step by step to analyze the newly added variables. For example, the composition of R_i requires $\{r_i, ID_i, a_i\}$ or $\{a_i, W_1, UM_i\}$ or $\{W_3, IDS_j, SM_j\}$.

(2) Then, coloring is employed to denote nodes that involve long-term secrets or are transmitted over public channels. These nodes are k, W_1, W_3, W_4, W_5, which means that $\mathcal{A}$ can obtain these variables.

(3) Finally, we remove the incoming edges of all colored nodes, and judge whether the proposed protocol ensures PFS through the remaining nodes. From Figure 9, we can see that the $\mathcal{A}$ does not have the required variables to compute the SK.

Thus, our proposed protocol ensures PFS.

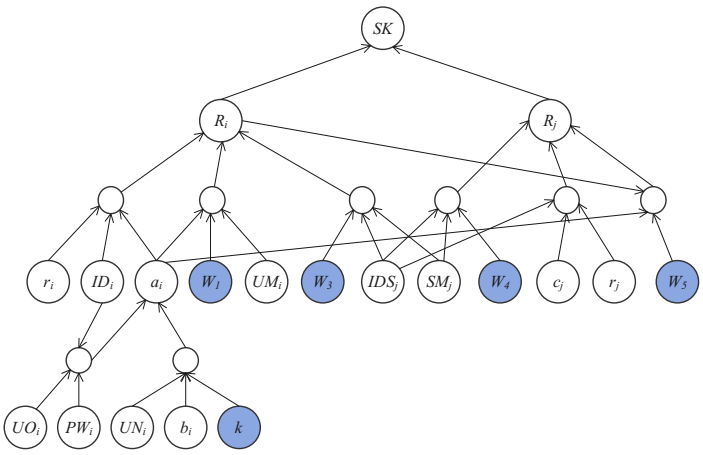


Figure 9. The verification result of our protocol for PFS using Ge et al.'s method [45].

4.2.2. Privileged Insider (PI) Attacks

Suppose that $\mathcal{A}$ is an insider in the gateway and has access to data $\{TID_i, b_i, UN_i\}$ and $\{IDS_j, SN_j\}$ in its database. Then, $\mathcal{A}$ attempts to compute the values R_i and R_j using these data, where $R_i = W_1 \oplus h(a_i \parallel UM_i)$ and $R_j = W_4 \oplus h(IDS_j \parallel SM_j)$. Because a_i, UM_i and SM_j are confidential to $\mathcal{A}$, the $\mathcal{A}$ cannot compute R_i and R_j, and then cannot calculate the SK. Therefore, our protocol prevents PI attacks.

4.2.3. Sensor Node Capture (SNC) Attacks

Assume $\mathcal{A}$ can capture the $\{c_j, SO_j\}$ in the memory of the S_j, and attempt to calculate the values R_i and R_j. However, since $\mathcal{A}$ cannot obtain IDS_j, SM_j, and R_j, $\mathcal{A}$ cannot compute R_i and R_j, and thus the $\mathcal{A}$ does not obtain the correct SK. So our protocol can withstand SNC attacks.

4.2.4. Offline Password Guessing (OPG) Attacks

Suppose $\mathcal{A}$ obtains the data $\{UO_i, UP_i, UQ_i\}$ from a smart device and tries to enumerate the correct password using a password dictionary. Since the $\mathcal{A}$ cannot obtain the RPW_i

and a_i, and does not calculate the correct value UP_i, where $UP_i = h(TID_i \parallel RPW_i \parallel a_i)$, $\mathcal{A}$ cannot obtain the correct PW_i. Thus, our protocol can prevent OPG attacks.

4.2.5. Session Key Disclosure (SKD) Attacks

$\mathcal{A}$ can only obtain the private values R_i and R_j in order to compute the $SK = \{R_i \parallel R_j\}$. However, $\mathcal{A}$ cannot obtain the ID_i, a_i and r_i, so R_i cannot be calculated, where $R_i = h(ID_i \parallel a_i \parallel r_i)$. Similarly, the $\mathcal{A}$ cannot obtain IDS_j, c_j and r_j, and cannot calculate R_j, where $R_j = h(IDS_j \parallel c_j \parallel r_j)$. Thus, the correct SK remains undisclosed to $\mathcal{A}$. The proposed protocol is immune to SKD attacks.

4.2.6. Correctness of SK

In our proposed protocol, the entities involved in establishing the session key include U_i, GWN, and S_j. The required values for the SK are R_i and R_j, where $SK = h(R_i \parallel R_j)$, $R_i = h(ID_i \parallel a_i \parallel r_i)$ and $R_j = h(IDS_j \parallel c_j \parallel r_j)$. The U_i transmits the computed R_i to the GWN, which securely forwards it to the S_j. Upon receiving R_i, the S_j independently computes R_j to establish the SK. Similarly, the S_j transmits R_j to the GWN, which then forwards this value to the U_i. When receiving R_j, the U_i is able to successfully establish the SK. Therefore, our protocol ensures the correctness of SK.

4.2.7. Man-In-The-Middle (MITM) Attacks

Assume that $\mathcal{A}$ can intercept messages M_1, M_2, M_3 and M_4. Here, we take $M_1 = \{TID_i, W_1, W_2, V_1, T_1\}$ as an example. The $\mathcal{A}$ attempts to tamper with the authentication value V_1, where $V_1 = h(TID_i \parallel IDS_j \parallel R_i \parallel T_1)$. However, due to the fact that values IDS_j and R_j are confidential to the $\mathcal{A}$, $\mathcal{A}$ cannot calculate the V_1. Thus, the request message sent by $\mathcal{A}$ cannot be authenticated by the GWN. Similarly, $\mathcal{A}$ cannot obtain private values to tamper with messages M_2, M_3, and M_4. Thus, it is impossible for MITM attacks to break our protocol.

4.2.8. Mutual Authentication

In the proposed protocol, entities verify each other's legitimacy by the authentication values $\{V_1, V_2, V_3, V_4\}$, where $V_1 = h(TID_i \parallel IDS_j \parallel R_i \parallel T_1)$, $V_2 = h(TID_i \parallel R_i \parallel SM_j \parallel T_2)$, $V_3 = h(R_j \parallel SM_j \parallel T_2)$, and $V_4 = h(R_i \parallel UM_i \parallel R_j \parallel T_4)$. Here, the GWN is to determine the legitimacy of the U_i by verifying V_1. The S_j judges the legitimacy of the GWN by verifying V_2. The GWN is used to determine the legitimacy of the S_j by verifying V_3. The U_i is to determine the legitimacy of the GWN by verifying V_4. Since the message sent by one entity to another entity can be verified, our protocol can achieve mutual authentication.

4.3. AVISPA

The AVISPA [46] is an instrument for formal verification that automatically analyzes the cryptographic protocol's security. AVISPA is based on the DY model, which allows $\mathcal{A}$ to have attack capabilities during the simulation, and it uses High-Level Protocol Specification Language (HLPSL). In this paper, AVISPA is used to simulate the whole process of the proposed protocol.

We define the role specification for U_i, GWN and S_j as shown in Figure 10a–c, respectively. Additionally, the role specifications for the session, goal, and environment are shown in Figure 10d. Here, we take the role of U_i as an example to explain. In the registration and authentication phases, it is essential for the user to recognize the involvement of three agents: "user agent (UA), gateway agent (GA), and sensor agent (SA)". "(SND, RCV)" represent the send and receive channels, where "(dy)" means that the channel follows the DY model. "RCV(start)" indicates that the entire protocol starts running. "RCV(H(H(IDi.PWi.Ai').K.Ai')-SKuaga)" indicates that the user receives the message $\{UM_i\}$ transmitted from the gateway. The "SKuaga" encrypts transmitted messages, and this indicates that the message is transmitted via secure channel. Further-

more, "SND(H(IDi.PWi.Ai').W1'.W2'.V1'.T1')" signifies that the user transmits the message $\{TID_i, W_1, W_2, V_1, T_1\}$ to the gateway via a public channel. In "State 3", it becomes evident that the user has successfully established a session key with the sensor. Finally, we use the widely recognized On-the-Fly Model-Checker (OFMC) and Constraint Logic-based Attack Searcher (CL-AtSe) backends to verify the security of the proposed protocol, and the simulated results are depicted in Figure 11. It can be clearly seen that whether it is in the results of OFMC or CL-AtSe backend, the summary display is "SAFE", which means that our proposed protocol can resist replay and MITM attacks.

```
role doctor(UA,GA,SA:agent, SKuaga:symmetric_key,H:hash_func,SND,
  RCV: channel(dy))

played_by UA
def=
local State: nat,
   IDi,PWi,Ai,TIDi,Bi,K,UMi,UNi,RPWi,UOi,UPi,UQi,Ri,
   Rii,W1,W2,W3,V1,V2,T1,T2: text,
   IDSj,Cj,SMj,SNj,SOj,Rj,Rjj,V3,V4,W4,W5,T3,T4: text,
   SKij,SKji: text
const sp1,sp2,sp3,sp4,sp5,ua_ga_ri,ga_ua_rj,ga_sa_ri,sa_ga_rj: protocol_id
init State:=0
transition
1.State =0/\RCV(start)=|>
  State':= 1/\Ai':= new()
       /\TIDi':= H(IDi.PWi.Ai')
       /\SND({TIDi'}_SKuaga)
       /\secret({PWi},sp1,{UA})
       /\secret({TIDi'},sp2,{UA,GA})

2.State = 1/\RCV({H(H(IDi.PWi.Ai').K.Ai')}_SKuaga)=|>
  State':= 2/\ RPWi':= H(PWi,Ai')/\ UOi':=xor(Ai',H(IDi.H(PWi,Ai')))
       /\UPi':=H(H(IDi.PWi.Ai').H(PWi,Ai').Ai')
       /\UQi':= xor(H(H(IDi.PWi.Ai').K.Ai'),H(Ai'.H(PWi,Ai')))
       /\ Ri':= new() /\ T1':= new() /\Rii':= H(IDi.Ai'.Ri')
       /\ W1':= xor(H(IDi.Ai'.Ri'),H(Ai'.H(H(IDi.PWi.Ai').K.Ai')))
       /\ W2':= xor(IDSj,H(H(H(IDi.PWi.Ai').K.Ai').H(IDi.Ai'.Ri')))
       /\ V1':= H(H(IDi.PWi.Ai').IDSj.H(IDi.Ai'.Ri').T1')
       /\ SND(H(IDi.PWi.Ai').W1'.W2'.V1'.T1')
       /\ witness(UA,GA,ua_ga_ri,Ri')

3.State = 2/\RCV(xor(H(IDSj.Cj'.Rj'), H(Ai'.H(IDi.Ai'.Ri'))).
  H(H(IDi.Ai'.Ri').H(H(IDi.PWi.Ai').K.Ai').H(IDSj.Cj'.Rj').T4').T4')=|>
  State':=3/\SKij':= H(H(IDi.Ai'.Ri').H(IDSj.Cj'.Rj'))
       /\request(GA,UA,ga_ua_rj,Rj')
end role
```

(**a**)

```
role gateway(UA,GA,SA : agent, SKuaga,SKsaga: symmetric_key,H: hash_func,
SND, RCV: channel(dy))

played_by GA
def=
local State: nat,
   IDi,PWi,Ai,TIDi,Bi,K,UMi,UNi,RPWi,UOi,UPi,UQi,Ri,
   Rii,W1,W2,W3,V1,V2,T1,T2: text,
   IDSj,Cj,SMj,SNj,SOj,Rj,Rjj,V3,V4,W4,W5,T3,T4: text,
   SKij,SKji: text
const sp1,sp2,sp3,sp4,sp5,ua_ga_ri,ga_ua_rj,ga_sa_ri,sa_ga_rj: protocol_id
init State:=0
transition
1.State =0/\RCV({H(IDi.PWi.Ai').Ai'}_SKuaga)=|>State':= 1 /\ Bi':=new()
       /\ UMi' :=H(H(IDi.PWi.Ai).K.Ai)
       /\UNi':= xor(Ai,H(Bi'.K))
       /\SND({UMi'}_SKuaga)
       /\secret({Bi',K},sp3,{GA})

2.State = 1/\RCV({IDSj.Cj'}_SKsaga)=|>State':= 2/\SMj':=H(IDSj.Cj.K)
       /\SNj':= xor(H(IDSj.K),H(IDSj.Cj.K))
       /\SND({SMj'}_SKsaga)
       /\secret({K},sp4,{GA}) /\secret({Cj,SMj'},sp5,{SA,GA})

3.State = 2/\ RCV(H(IDi.PWi.Ai').xor(H(IDi.Ai'.Ri'),H(Ai'.H(H(IDi.PWi.Ai').
K.Ai'))).xor(IDSj,H(H(H(IDi.PWi.Ai').K.Ai').H(IDi.Ai'.Ri'))).H(H(IDi.PWi.Ai').
IDSj.H(IDi.Ai'.Ri').T1'))=|>State':= 3 /\T2' :=new()/\Cj' :=new()
       /\SMj':=xor(xor(H(IDSj.K),H(IDSj.Cj.K)),H(IDSj.K))
       /\W3':=xor(H(IDi.Ai.Ri),H(IDSj.xor(xor(H(IDSj.K),H(IDSj.Cj.K)),H(IDSj.K))))
       /\V2':=h(H(IDi.PWi.Ai).H(IDi.Ai.Ri).xor(xor(H(IDSj.K),H(IDSj.Cj.K)),
       H(IDSj.K)).T2')/\ SND(H(IDi.PWi.Ai).W3'.V2'.T2')
       /\ witness(GA,SA, ga_sa_ri, Ri')
       /\ request(UA,GA, ua_ga_ri, Ri')

4.State = 3/\ RCV(xor(H(IDSj.Cj'.Rj'),H(IDSj.xor(xor(H(IDSj.K),H(IDSj.Cj'.K)),
H(IDSj.K)))).H(IDSj.Cj'.Rj').xor(xor(H(IDSj.K),H(IDSj.Cj'.K)),H(IDSj.K)).T3')
=|>State':= 4 /\T4' :=new()
       /\ W5':=xor(H(IDSj.Cj.Rj),H(Ai,H(IDi.Ai.Ri)))
       /\ V4':=H(H(IDi.Ai.Ri).H(H(IDi.PWi.Ai').K.Ai').H(IDSj.Cj.Rj).T4')
       /\ SND(W5',V4',T4')
       /\ witness(GA,UA, ga_ua_rj,Rj')
       /\ request(SA,GA, sa_ga_rj,Rj')
end role
```

(**b**)

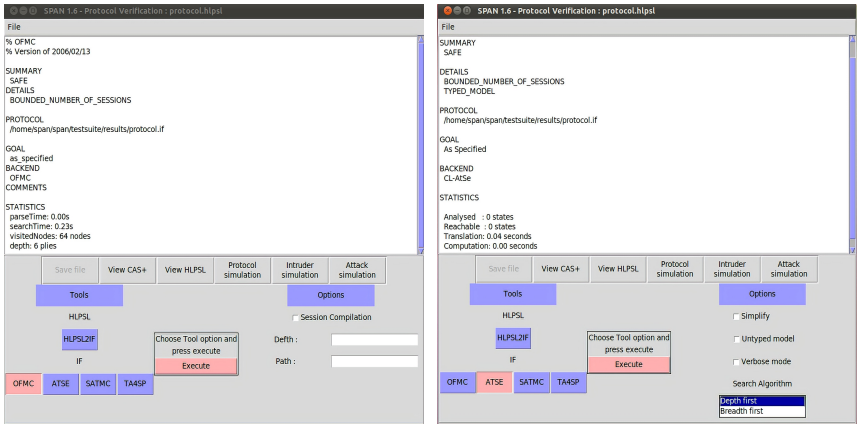
```
role sensor(UA,GA,SA : agent,SKuasa,SKsaga : symmetric_key, H:hash_func,SND,
RCV : channel(dy))

played_by SA
def=
local State: nat,
   IDi,PWi,Ai,TIDi,Bi,K,UMi,UNi,RPWi,UOi,UPi,UQi,Ri,Rii,
   W1,W2,W3,V1,V2,T1,T2: text,
   IDSj,Cj,SMj,SNj,SOj,Rj,Rjj,V3,V4,W4,W5,T3,T4: text,
   SKij,SKji: text
const sp1,sp2,sp3,sp4,sp5,ua_ga_ri,ga_ua_rj,ga_sa_ri,sa_ga_rj: protocol_id
init State:=0
transition
1.State=0/\ RCV(start)=|>
   State':= 1/\ Cj':=new()
       /\ SND({IDSj.Cj'}_SKsaga)

2.State = 1/\ RCV({H(IDSj.Cj'.K)}_SKsaga)=|>State':= 2
       /\ SOj':= xor(H(IDSj.Cj'),H(IDSj.Cj'.K))

3.State = 2/\ RCV(H(IDi.PWi.Ai').xor(H(IDi.Ai'.Ri'),H(IDSj.xor(xor(H(IDSj.K),
H(IDSj.Cj'.K)),H(IDSj.K)))).H(H(IDi.PWi.Ai').H(IDi.Ai'.Ri').xor(xor(H(IDSj.K),
H(IDSj.Cj'.K)),H(IDSj.K)).T2').T2')=|>State':= 3
       /\ Rj':= new() /\ T3':= new()/\Rjj':= H(IDSj.Cj'.Rj')
       /\ SKji':=H(H(IDi.Ai'.Ri').H(IDSj.Cj'.Rj'))
       /\ W4':= xor(H(IDSj.Cj'.Rj'),H(IDSj.xor(xor(H(IDSj.K),
       H(IDSj.Cj'.K)),H(IDSj.K))))
       /\ V3':=H(H(IDSj.Cj'.Rj').xor(xor(H(IDSj.K),H(IDSj.Cj'.K)),H(IDSj.K)).T3')
       /\ SND(W4',V3',T3')
       /\ witness(SA,GA, sa_ga_rj,Rj')
       /\ request(GA,SA, ga_sa_ri,Ri')
end role
```

(**c**)

```
role session(UA,GA,SA : agent,SKuaga,SKsaga : symmetric_key,
H:hash_func)
def=
local SN1,SN2,SN3, RV1, RV2,RV3: channel(dy)
composition
doctor(UA,GA,SA,SKuaga, H,SN1,RV1)
   /\gateway(UA,GA, SA, SKuaga, SKsaga, H, SN2,RV2)
   /\sensor(UA,GA,SA,SKuaga,SKsaga, H, SN3, RV3)
end role
role environment()
def=
   const ua, ga, sa : agent,
   skuaga,sksaga: symmetric_key,
   h: hash_func,
   idi,idsj,tidi: text,
   ua_ga_ri,ga_ua_rj,ga_sa_ri,sa_ga_rj: protocol_id,
   sp1,sp2,sp3,sp4,sp5: protocol_id
   intruder_knowledge = {ua,ga,sa,idi,tidi,idsj,h}
composition
session(ua,ga,sa,skuaga, sksaga,h)/\session(i,ga,sa,skuaga, sksaga,h)
   /\session(ua,i,sa, skuaga, sksaga,h)
   /\session(ua,ga,i, skuaga, sksaga,h)
end role
goal
   secrecy_of sp1, sp2, sp3, sp4, sp5
   authentication_on ua_ga_ri
   authentication_on ga_sa_ri
   authentication_on ga_ua_rj
   authentication_on sa_ga_rj
end goal
environment()
```

(**d**)

Figure 10. Proof of AVISPA. (**a**) Role specification for user. (**b**) Role specification for gateway. (**c**) Role specification for sensor. (**d**) Role specification for session, goal, environment.



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

Figure 11. The simulation results. (**a**) Result using OFMC backend. (**b**) Result using ATSE backend.

5. Security and Performance Comparisons

We compare the security and performance of our proposed protocol to five IoHT authentication protocols [23,29,33,35,47].

5.1. Security Comparisons

In terms of security comparison, ✓ means that the protocol is resistant to that attack, while × means that the protocol does not satisfy that security property. The primary security properties include S1, mutual authentication; S2, PFS; S3, PI attacks; S4, OPG attacks; S5, SKD attacks; S6, SNC attacks; S7, MITM attacks; S8, correctness of SK.

The security comparison results are presented in Table 3. It is clear that our protocol and Wu et al.'s protocol [47] satisfies all security properties. However, Soni et al.'s protocol [23] violated PFS and suffered from OPG and SNC attacks. Hajian et al.'s protocol [33] failed to provide mutual authentication, leaving it vulnerable to SKD and MITM attacks. Similarly, Shuai et al.'s protocol [35] also violated PFS and suffered from PI attacks. Amintoosi et al.'s protocol [29], like the others, exhibited security weaknesses, specifically being susceptible to PI attacks and unable to ensure the correctness of SK.

Table 3. Security comparison results.

Security Properties	Soni et al. [23]	Hajian et al. [33]	Shuai et al. [35]	Amintoosi et al. [29]	Wu et al. [47]	Ours
S1	✓	× [17]	✓	✓	✓	✓
S2	× [24]	✓	× [36]	✓	✓	✓
S3	✓	✓	× [36]	×	✓	✓
S4	× [24]	✓	✓	✓	✓	✓
S5	✓	× [17]	✓	✓	✓	✓
S6	× [24]	✓	✓	✓	✓	✓
S7	✓	× [17]	✓	✓	✓	✓
S8	✓	✓	✓	×	✓	✓

5.2. Performance Comparisons

The protocol compares three aspects of communication, computational, and storage costs in performance comparison. When comparing communication and computational costs, we exclusively consider the login and authentication phases of the protocols. On the other hand, in the comparison of storage costs, our focus is solely on the registration phase.

5.2.1. Computational Cost Comparisons

For the computational cost, we use three different devices to obtain the runtime of the cryptographic primitives. The configurations of these three experimental devices are shown in Table 4, where we denote that the laptop simulates U_i, the desktop computer simulates GWN, and the Xiaomi mobile phone MI 8 simulates S_j. The software we use is

IntelliJ idea 2020.3, and we use the Java language and cryptographic library JPBC-2.0.0 [48] to write programs. In addition, since the cost of $\oplus$ and $\|$ in the protocol is too small, its computational size is ignored. The times of various operations are displayed in Table 5, where the running time of the operation runs 20 times in the software and takes the average value of the results. In addition, since the running time of the hash function and the fuzzy extraction are similar, we take one of them to calculate. The results of the comparison are presented in Table 6, and more clearly shown in Figure 12.

Table 4. Configuration of simulated devices.

	Lenovo Laptop	Desktop Computer	MI 8
Operating System	Windows 10	Windows 10	Android system
CPU	Intel(R) Core(TM) i7-6700HQ CPU @ 2.60 GHz	Intel(R) Core(TM) i5-9500 CPU @ 3.00 GHz	Qualcomm Snapdragon 845
Running Memory	8 GB	16 GB	6 GB

Table 5. Running time of operations.

Definitions	Operations	U_i (ms)	GWN (ms)	S_j (ms)
T_{pm}	Point scalar multiplication	0.4326	0.3672	0.5543
T_{sd}	Symmetric key encryption/decryption	0.1864	0.1482	0.2458
T_h	Hash function	0.0032	0.0028	0.0043

Table 6. Computational cost comparison.

Protocols	U_i (ms)	GWN (ms)	S_j (ms)
Soni et al. [23]	$T_f + 12T_h + 3T_{pm} \approx 1.3394$	$11T_h + 3T_{pm} \approx 1.1324$	$5T_h \approx 0.0215$
Hajian et al. [33]	$12T_h \approx 0.0384$	$7T_h \approx 0.0196$	$9T_h \approx 0.0387$
Shuai et al. [35]	$T_f + 7T_h + 2T_{sd} \approx 0.3984$	$10T_h + 2T_{sd} \approx 0.3244$	$4T_h \approx 0.0172$
Amintoosi et al. [29]	$8T_h \approx 0.0256$	$10T_h \approx 0.0280$	$6T_h \approx 0.0258$
Wu et al. [47]	$T_f + 15T_h \approx 0.0512$	$21T_h \approx 0.0588$	$9T_h \approx 0.0387$
Ours	$12T_h \approx 0.0384$	$12T_h \approx 0.0336$	$7T_h \approx 0.0301$

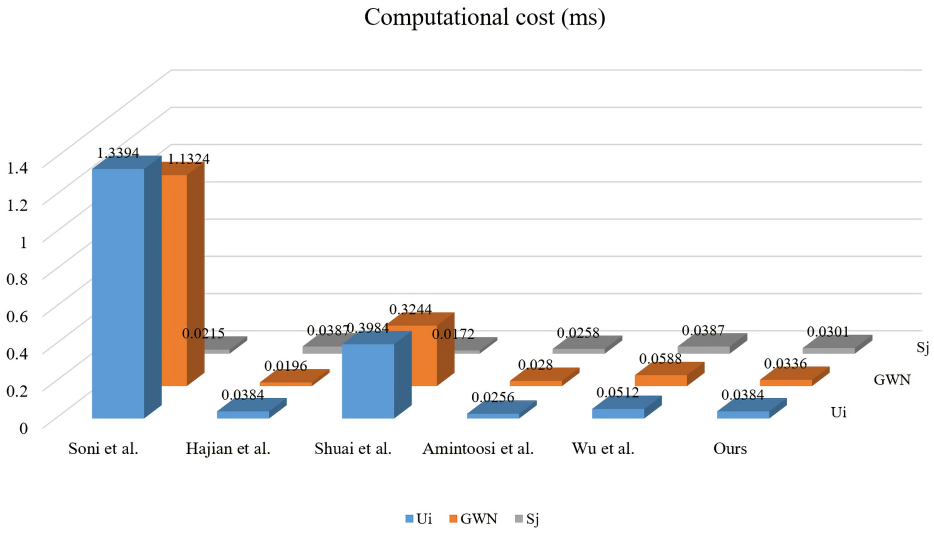


Figure 12. The comparison of computational cost [23,29,33,35,47].

The computational costs of a few U_i in each protocol are illustrated in Figure 13a. Soni et al.'s protocol [23] utilizes point scalar multiplication and fuzzy extractor, and Shuai et al.'s protocol [35] relies on symmetric key encryption/decryption. As a result, both of them incur relatively high computational costs for U_i compared to the other protocols in

the comparison. On the other hand, the computational costs of U_i in the remaining protocols show little variation and are relatively lower compared to Soni et al.'s and Shuai et al.'s protocols. Figure 13b depicts the computational costs of a few S_j in each protocol. In our proposed protocol, the computational costs of S_j are higher than in some other protocols, but still lower than the costs in Hajian et al. [33] and Wu et al. [47]. It is worth noting that Hajian et al.'s protocol is the same as that in Wu et al.'s work. Overall, Soni et al.'s and Shuai et al.'s protocols have relatively higher computational costs for U_i due to their specific cryptographic operations.

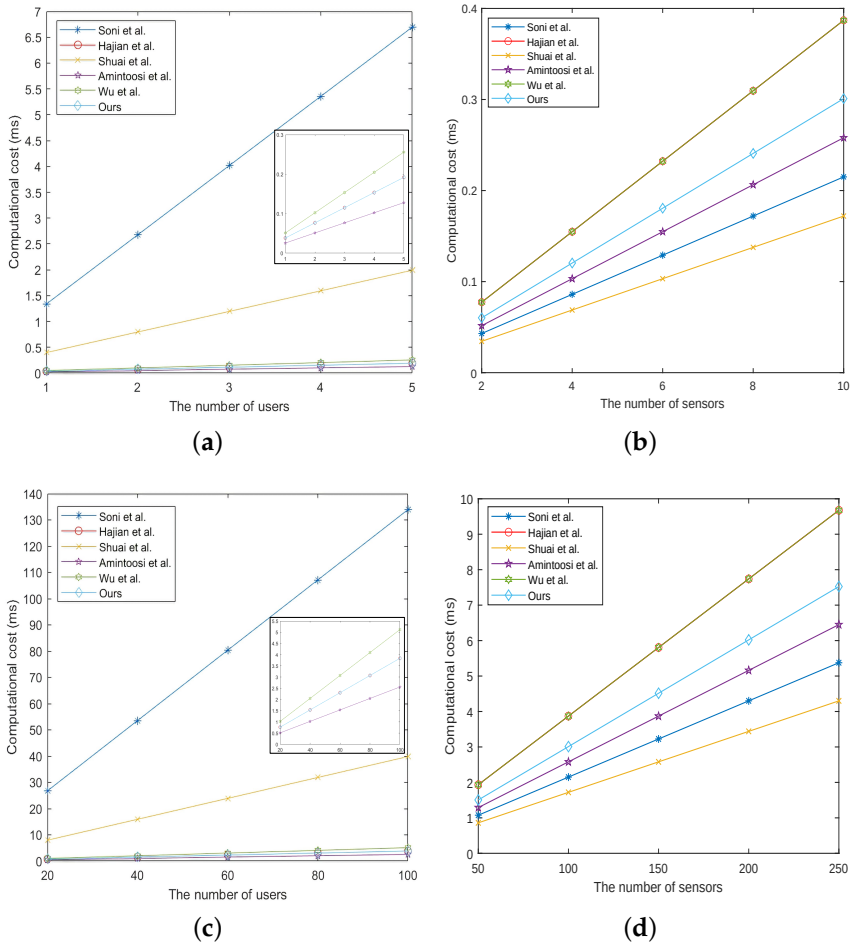


Figure 13. The computational costs for users and sensors [23,29,33,35,47]. (**a**) Computational cost with few users. (**b**) Computational cost with few sensors. (**c**) Computational cost during user surge. (**d**) Computational cost during sensor surge.

To verify the scalability of the protocol, we gradually increased the number of U_i from 20 to 100, while simultaneously increasing the number of S_j from 50 to 250. The results of computational cost as the counts of U_i and S_j surged are presented in Figure 13c,d, respectively. The results demonstrate that our protocol can maintain reasonable computational costs as the quantity of entities grows, ensuring the protocol retains stable performance and efficiency. As a result, our proposed protocol can guarantee scalability.

5.2.2. Communication Cost Comparisons

In the comparison of communication costs, the lengths of the identity, timestamp, hash function, random number, point multiplication, and symmetrically encrypted ciphertext are defined to be 160, 32, 256, 128, 320, and 256 bits, respectively. Here, the communication cost is illustrated using our protocol as an example. The messages $M_1 = \{TID_i, W_1, W_2, V_1, T_1\}$, $M_2 = \{TID_i, W_3, V_2, T_2\}$, $M_3 = \{W_4, V_3, T_3\}$, and $M_4 = \{W_5, V_4, T_4\}$, in which TID_i

is identity, $\{W_1, W_2, W_3, W_4, W_5\}$ are random numbers, $\{V_1, V_2, V_3, V_4\}$ are hash values, $\{T_1, T_2, T_3, T_4\}$ are timestamps. Based on the above calculation, our proposed protocol's communication cost is $2 \times 160 + 5 \times 128 + 4 \times 256 + 4 \times 32 = 2112$ bits. Soni et al.'s protocol [23] is $7 \times 128 + 4 \times 256 + 5 \times 32 + 320 = 2400$ bits, Hajian et al.'s protocol [33] is $4 \times 160 + 3 \times 128 + 8 \times 256 + 7 \times 32 = 3296$ bits, Shuai et al.'s protocol [35] is $160 + 128 + 6 \times 256 + 4 \times 32 = 1952$ bits, Amintoosi et al.'s protocol [29] is $6 \times 128 + 5 \times 256 + 4 \times 32 = 2176$ bits, and Wu et al.'s protocol [47] is $11 \times 128 + 4 \times 256 + 4 \times 32 = 2560$ bits. Based on the data presented in Table 7 and Figure 14, it is evident that our proposed protocol exhibits a slightly higher communication cost compared to Shuai et al.'s protocol [35]. However, our proposed protocol still maintains lower communication costs compared to Soni et al. [23], Hajian et al. [33], Wu et al. [47], and Amintoosi et al. [29].

Table 7. Communication cost comparisons.

Protocols	Rounds	Communication Cost
Soni et al. [23]	4	2400 bits
Hajian et al. [33]	5	3296 bits
Shuai et al. [35]	4	1952 bits
Amintoosi et al. [29]	4	2176 bits
Wu et al. [47]	4	2560 bits
Ours	4	2112 bits

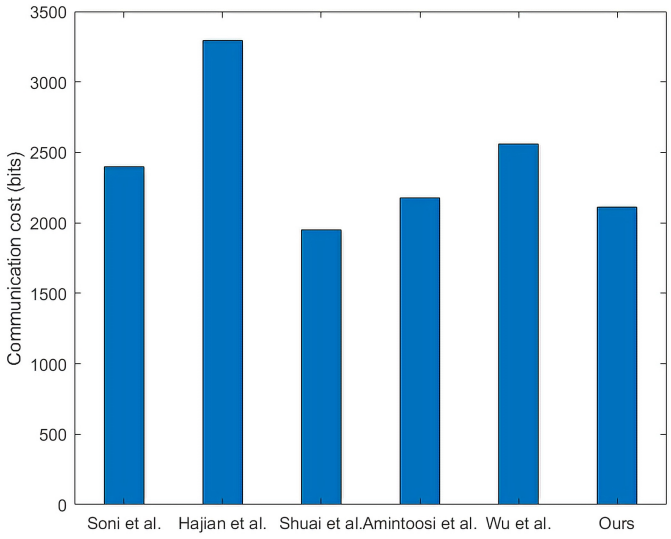


Figure 14. Comparisons of communication cost [23,29,33,35,47].

5.2.3. Storage Cost Comparisons

In the comparison of storage costs, the lengths required for various parameters are consistent with the assumptions in Section 5.2.2. Here, we take the registration phases of our proposed protocol as an example. The storage costs required for U_i, GWN, and S_j are $128 \times 2 + 256 = 512$ bits, $128 \times 3 + 160 \times 2 = 704$ bits, and $128 \times 2 = 256$ bits, respectively. The storage cost required for our proposed protocol is 1472 bits. The total storage costs for each protocol are presented in Table 8. From Figure 15, it is evident that Hajian et al.'s protocol [33] demands the highest storage costs. In contrast, our proposed protocol requires the minimum storage costs.

Table 8. Storage cost comparisons.

Protocols	Storage Cost
Soni et al. [23]	1632 bits
Hajian et al. [33]	2336 bits
Shuai et al. [35]	1728 bits
Amintoosi et al. [29]	2208 bits
Wu et al. [47]	1792 bits
Ours	1472 bits

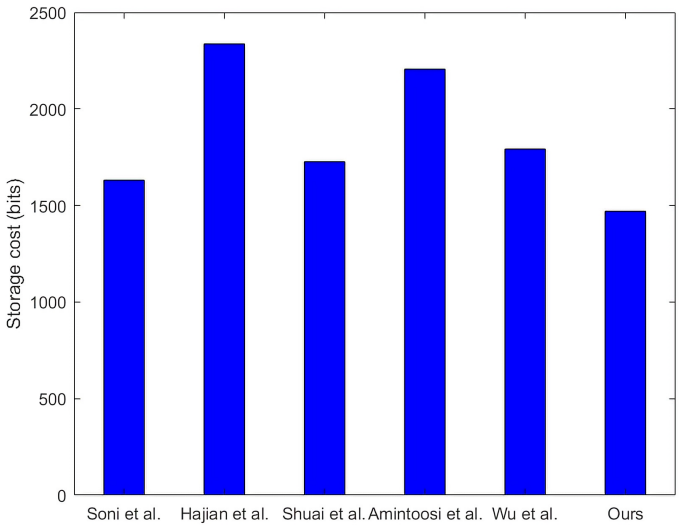


Figure 15. Comparisons of storage cost [23,29,33,35,47].

Based on the security and performance comparison results, we can confidently assert the following:

1. Security comparison: Our proposed protocol, along with Wu et al.'s protocol, demonstrates the ability to withstand all known attacks. In contrast, other protocols in the comparison exhibit varying degrees of vulnerability to certain attacks.
2. Performance comparison: Despite having the same security level of as Wu et al.'s protocol, our protocol outperforms theirs in terms of computational and storage costs, while also possessing scalability. Additionally, while our computational cost is slightly higher compared to Amintoosi et al.'s protocol, our communication and storage costs are lower than theirs.

6. Conclusions

In this paper, we emphasized the significance of ensuring secure data transmission within the IoHT environments. We conducted a comprehensive review of the AKA protocols employed in the IoHT context. Subsequently, we thoroughly analyzed Amintoosi et al.'s protocol and identified various security weaknesses, notably PI attacks. In response to these issues, we proposed an enhanced AKA protocol specifically tailored for the IoHT environment. Then, we subjected it to rigorous security analysis using the ROR model, informal security analysis, and the AVISPA tool. Finally, we compared the security and performance aspects of our proposed protocol with existing protocols. The comparison results revealed that our protocol outperforms other protocols in terms of security while maintaining a comparable level of performance, thereby enhancing the feasibility of its practical application. The potential challenge lies in the slightly higher computational and communication costs of the proposed protocol, but this is acceptable in practical applications. Consequently, in future research, we will focus on further enhancing the security and performance of AKA protocols in the IoHT to address evolving needs.

Author Contributions: Conceptualization, T.-Y.W.; methodology, T.-Y.W. and L.W.; software and security analysis, C.-M.C. and L.W.; investigation, C.-M.C.; writing—original draft preparation, T.-Y.W., L.W. and C.-M.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported by Natural Science Foundation of Shandong Province, China (Grant no. ZR202111230202).

Data Availability Statement: The data are included in the article.

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

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
IoHT	Internet Health of Things
ROR	Real-Or-Random
MITM	Man-in-the-middle
AKA	Authentication and key agreement
ECC	Elliptic curve cryptography
PFS	Perfect forward secrecy
OPG	Offline password guessing
SNC	Sensor node capture
PI	Privileged insider
KSSTI	Known session specific temporary information
MIoT	Medical Internet of Things
SKD	Session key disclosure
AVISPA	Automated validation of internet security protocols and applications
HLPSL	High-Level Protocol Specification Language
OFMC	On-the-Fly Model-Checker
CL-AtSe	Constraint Logic-based Attack Searcher

References

1. Shen, S.; Yang, Y.; Liu, X. Toward data privacy preservation with ciphertext update and key rotation for IoT. *Concurr. Comput. Pract. Exp.* **2021**, *35*, e6729. [CrossRef]
2. Huang, X.; Xiong, H.; Chen, J.; Yang, M. Efficient revocable storage attribute-based encryption with arithmetic span programs in cloud-assisted internet of things. *IEEE Trans. Cloud Comput.* **2023**, *11*, 1273–1285. [CrossRef]
3. Guezzaz, A.; Benkirane, S.; Azrour, M. A novel anomaly network intrusion detection system for internet of things security. In *IoT and Smart Devices for Sustainable Environment*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 129–138. [CrossRef]
4. Wu, T.Y.; Meng, Q.; Chen, Y.C.; Kumari, S.; Chen, C.M. Toward a Secure Smart-Home IoT Access Control Scheme Based on Home Registration Approach. *Mathematics* **2023**, *11*, 2123. [CrossRef]
5. Luo, Y.; Zheng, W.m.; Chen, Y.C. An anonymous authentication and key exchange protocol in smart grid. *J. Netw. Intell.* **2021**, *6*, 206–215.
6. Chaudhry, S.A. Combating identity de-synchronization: An improved lightweight symmetric key based authentication scheme for IoV. *J. Netw. Intell.* **2021**, *6*, 12.
7. Xiong, H.; Chen, J.; Mei, Q.; Zhao, Y. Conditional privacy-preserving authentication protocol with dynamic membership updating for VANETs. *IEEE Trans. Dependable Secur. Comput.* **2022**, *19*, 2089–2104. [CrossRef]
8. Xue, X.; Chen, J. Matching biomedical ontologies through compact differential evolution algorithm with compact adaption schemes on control parameters. *Neurocomputing* **2021**, *458*, 526–534. [CrossRef]
9. Xue, X.; Huang, Q. Generative adversarial learning for optimizing ontology alignment. *Expert Syst.* **2023**, *40*, e12936. [CrossRef]
10. Xiong, H.; Qin, Z. Revocable and scalable certificateless remote authentication protocol with anonymity for wireless body area networks. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 1442–1455. [CrossRef]
11. Si-Ahmed, A.; Al-Garadi, M.A.; Boustia, N. Survey of Machine Learning based intrusion detection methods for Internet of Medical Things. *Appl. Soft Comput.* **2023**, *140*, 110227. [CrossRef]
12. Singh, A.; Chatterjee, K.; Satapathy, S.C. TrIDS: An intelligent behavioural trust based IDS for smart healthcare system. *Clust. Comput.* **2023**, *26*, 903–925. [CrossRef]
13. Nikkhah, F.; Safkhani, M. LAPCHS: A lightweight authentication protocol for cloud-based health-care systems. *Comput. Netw.* **2021**, *187*, 107833. [CrossRef]
14. Gupta, S.; Arya, P.K.; Sharma, H.K. User anonymity-based secure authentication protocol for telemedical server systems. *Int. J. Inf. Comput. Secur.* **2023**, *20*, 199–219. [CrossRef]

15. Safkhani, M.; Vasilakos, A. A new secure authentication protocol for telecare medicine information system and smart campus. *IEEE Access* **2019**, *7*, 23514–23526. [CrossRef]
16. Alzahrani, B.A.; Irshad, A.; Albeshri, A.; Alsubhi, K. A provably secure and lightweight patient-healthcare authentication protocol in wireless body area networks. *Wirel. Pers. Commun.* **2021**, *117*, 47–69. [CrossRef]
17. Yu, S.; Park, K. SALS-TMIS: Secure, Anonymous and Lightweight Privacy-Preserving Scheme for IoMT-Enabled TMIS Environments. *IEEE Access* **2022**, *10*, 60534–60549. [CrossRef]
18. Lee, J.; Oh, J.; Park, Y. A secure and anonymous authentication protocol based on three-factor wireless medical sensor networks. *Electronics* **2023**, *12*, 1368. [CrossRef]
19. Li, J.; Su, Z.; Guo, D.; Choo, K.K.R.; Ji, Y. PSL-MAAKA: Provably secure and lightweight mutual authentication and key agreement protocol for fully public channels in internet of medical things. *IEEE Internet Things J.* **2021**, *8*, 13183–13195. [CrossRef]
20. Shamshad, S.; Ayub, M.F.; Mahmood, K.; Kumari, S.; Chaudhry, S.A.; Chen, C.M. An enhanced scheme for mutual authentication for healthcare services. *Digit. Commun. Netw.* **2022**, *8*, 150–161. [CrossRef]
21. Diffie, W.; Van Oorschot, P.C.; Wiener, M.J. Authentication and authenticated key exchanges. *Des. Codes Cryptogr.* **1992**, *2*, 107–125. [CrossRef]
22. Challa, S.; Das, A.K.; Odelu, V.; Kumar, N.; Kumari, S.; Khan, M.K.; Vasilakos, A.V. An efficient ECC-based provably secure three-factor user authentication and key agreement protocol for wireless healthcare sensor networks. *Comput. Electr. Eng.* **2018**, *69*, 534–554. [CrossRef]
23. Soni, P.; Pal, A.K.; Islam, S.H. An improved three-factor authentication scheme for patient monitoring using WSN in remote health-care system. *Comput. Methods Programs Biomed.* **2019**, *182*, 105054. [CrossRef] [PubMed]
24. Xu, G.; Wang, F.; Zhang, M.; Peng, J. Efficient and provably secure anonymous user authentication scheme for patient monitoring using wireless medical sensor networks. *IEEE Access* **2020**, *8*, 47282–47294. [CrossRef]
25. Qiu, S.; Xu, G.; Ahmad, H.; Wang, L. A Robust Mutual Authentication Scheme Based on Elliptic Curve Cryptography for Telecare Medical Information Systems. *IEEE Access* **2018**, *6*, 7452–7463. [CrossRef]
26. Sharma, G.; Kalra, S. A lightweight user authentication scheme for cloud-IoT based healthcare services. *Iran. J. Sci. Technol. Trans. Electr. Eng.* **2019**, *43*, 619–636. [CrossRef]
27. Azrour, M.; Mabrouki, J.; Chaganti, R. New efficient and secured authentication protocol for remote healthcare systems in cloud-iot. *Secur. Commun. Netw.* **2021**, *2021*, 5546334. [CrossRef]
28. Aghili, S.F.; Mala, H.; Shojafar, M.; Peris-Lopez, P. LACO: Lightweight three-factor authentication, access control and ownership transfer scheme for e-health systems in IoT. *Future Gener. Comput. Syst.* **2019**, *96*, 410–424. [CrossRef]
29. Amintoosi, H.; Nikooghadam, M.; Shojafar, M.; Kumari, S.; Alazab, M. Slight: A lightweight authentication scheme for smart healthcare services. *Comput. Electr. Eng.* **2022**, *99*, 107803. [CrossRef]
30. Merabet, F.; Cherif, A.; Belkadi, M.; Blazy, O.; Conchon, E.; Sauveron, D. New efficient M2C and M2M mutual authentication protocols for IoT-based healthcare applications. *Peer-to-Peer Netw. Appl.* **2020**, *13*, 439–474. [CrossRef]
31. Kumari, A.; Kumar, V.; Abbasi, M.Y.; Kumari, S.; Chaudhary, P.; Chen, C.M. Csef: Cloud-based secure and efficient framework for smart medical system using ecc. *IEEE Access* **2020**, *8*, 107838–107852. [CrossRef]
32. Wu, T.Y.; Yang, L.; Luo, J.N.; Ming-Tai Wu, J. A provably secure authentication and key agreement protocol in cloud-based smart healthcare environments. *Secur. Commun. Netw.* **2021**, *2021*, 2299632. [CrossRef]
33. Hajian, R.; ZakeriKia, S.; Erfani, S.H.; Mirabi, M. SHAPARAK: Scalable healthcare authentication protocol with attack-resilience and anonymous key-agreement. *Comput. Netw.* **2020**, *183*, 107567. [CrossRef]
34. Alladi, T.; Chamola, V. HARCI: A two-way authentication protocol for three entity healthcare IoT networks. *IEEE J. Sel. Areas Commun.* **2020**, *39*, 361–369. [CrossRef]
35. Shuai, M.; Yu, N.; Wang, H.; Xiong, L.; Li, Y. A lightweight three-factor Anonymous authentication scheme with privacy protection for personalized healthcare applications. *J. Organ. End User Comput. (JOEUC)* **2021**, *33*, 1–18. [CrossRef]
36. Xie, Q.; Ding, Z.; Hu, B. A secure and privacy-preserving three-factor anonymous authentication scheme for wireless sensor networks in Internet of Things. *Secur. Commun. Netw.* **2021**, *2021*, 4799223. [CrossRef]
37. Agrahari, A.K.; Varma, S.; Venkatesan, S. Two factor authentication protocol for IoT based healthcare monitoring system. *J. Ambient. Intell. Humaniz. Comput.* **2022**, 1–18. [CrossRef] [PubMed]
38. Al-Saggaf, A.A.; Sheltami, T.; Alkhzaimi, H.; Ahmed, G. Lightweight two-factor-based user authentication protocol for iot-enabled healthcare ecosystem in quantum computing. *Arab. J. Sci. Eng.* **2023**, *48*, 2347–2357. [CrossRef] [PubMed]
39. Dolev, D.; Yao, A. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [CrossRef]
40. Canetti, R.; Krawczyk, H. Analysis of key-exchange protocols and their use for building secure channels. In *International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings of the EUROCRYPT 2001: Advances in Cryptology—EUROCRYPT 2001, Innsbruck, Austria, 6–10 May 2001*; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2045, pp. 453–474.
41. Abdalla, M.; Fouque, P.A.; Pointcheval, D. Password-based authenticated key exchange in the three-party setting. In *Public Key Cryptography-PKC 2005, Proceedings of the 8th International Workshop on Theory and Practice in Public Key Cryptography, Les Diablerets, Switzerland, 23–26 January 2005*; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3386, pp. 65–84.
42. Li, X.; Liu, S.; Kumari, S.; Chen, C.M. PSAP-WSN: A Provably Secure Authentication Protocol for 5G-Based Wireless Sensor Networks. *CMES-Comput. Model. Eng. Sci.* **2023**, *135*, 711–732. [CrossRef]

43. Chen, C.M.; Liu, S.; Li, X.; Islam, S.H.; Das, A.K. A provably-secure authenticated key agreement protocol for remote patient monitoring IoMT. *J. Syst. Archit.* **2023**, *136*, 102831. [CrossRef]
44. 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]
45. Ge, M.; Kumari, S.; Chen, C.M. AuthPFS: A Method to Verify Perfect Forward Secrecy in Authentication Protocols. *J. Netw. Intell.* **2022**, *7*, 734–750.
46. Armando, A.; Basin, D.; Boichut, Y.; Chevalier, Y.; Compagna, L.; Cuéllar, J.; Drielsma, P.H.; Héam, P.C.; Kouchnarenko, O.; Mantovani, J.; et al. The AVISPA tool for the automated validation of internet security protocols and applications. In Proceedings of the Computer Aided Verification: 17th International Conference, CAV 2005, Edinburgh, UK, 6–10 July 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 281–285.
47. Wu, T.Y.; Meng, Q.; Yang, L.; Kumari, S.; Pirouz, M. Amassing the Security: An Enhanced Authentication and Key Agreement-Protocol for Remote Surgery in Healthcare Environment. *CMES-Comput. Model. Eng. Sci.* **2023**, *134*, 317–341. [CrossRef]
48. De Caro, A.; Iovino, V. jPBC: Java pairing based cryptography. In Proceedings of the 2011 IEEE Symposium on Computers and Communications (ISCC), Kerkyra, Greece, 28 June–1 July 2011; pp. 850–855. [CrossRef]