sensors

MDPI

Article

Robust and Efficient Authentication and Group–Proof Scheme Using Physical Unclonable Functions for Wearable Computing

Sungjin Yu [1,2] and Youngho Park [2,*]

[1] Electronics and Telecommunications Research Institute, Daejeon 34129, Republic of Korea; sj.yu@etri.re.kr
[2] School of Electronics and Electrical Engineering, Kyungpook National University, Daegu 41566, Republic of Korea
* Correspondence: parkyh@knu.ac.kr

Abstract: Wearable computing has garnered a lot of attention due to its various advantages, including automatic recognition and categorization of human actions from sensor data. However, wearable computing environments can be fragile to cyber security attacks since adversaries attempt to block, delete, or intercept the exchanged information via insecure communication channels. In addition to cyber security attacks, wearable sensor devices cannot resist physical threats since they are batched in unattended circumstances. Furthermore, existing schemes are not suited for resource-constrained wearable sensor devices with regard to communication and computational costs and are inefficient regarding the verification of multiple sensor devices simultaneously. Thus, we designed an efficient and robust authentication and group–proof scheme using physical unclonable functions (PUFs) for wearable computing, denoted as AGPS-PUFs, to provide high-security and cost-effective efficiency compared to the previous schemes. We evaluated the security of the AGPS-PUF using a formal security analysis, including the ROR Oracle model and AVISPA. We carried out the testbed experiments using MIRACL on Raspberry PI4 and then presented a comparative analysis of the performance between the AGPS-PUF scheme and the previous schemes. Consequently, the AGPS-PUF offers superior security and efficiency than existing schemes and can be applied to practical wearable computing environments.

Keywords: physical unclonable function (PUF); privacy-preserving; authentication; group proof; wearable computing

Citation: Yu, S.; Park, Y. Robust and Efficient Authentication and Group–Proof Scheme Using Physical Unclonable Functions for Wearable Computing. *Sensors* **2023**, *23*, 5747. https://doi.org/10.3390/s23125747

Academic Editors: Chao-Yang Lee, Neng-Chung Wang and Ming-Fong Tsai

Received: 29 May 2023
Revised: 15 June 2023
Accepted: 19 June 2023
Published: 20 June 2023

1. Introduction

With the development of "mobile and 5G communication" technologies, wearable computing is emerging as a new ubiquitous technology within the Internet of Things (IoT) and it has garnered a lot of attention from both scientific and academic communities [1–3]. The wearable devices are integrated into various types of accessories and clothing and provide useful application services in various fields, including "military, healthcare, and industry". In particular, sustainable wearable computing technology offers innovative healthcare opportunities, which give new methods to medical professionals to treat patients. For instance, wearable computing-based healthcare systems reduce healthcare costs and provide various medical services, including "monitoring, medical consultation, and emergency treatment" [4].

In these environments, wearable devices collect medical data, including "asthma level, blood pressure, electrocardiogram, body temperature" from the patients, and then transmit the corresponding data to the paired mobile terminal. The mobile terminal transmits the received data to the trusted cloud server, and authorized medical professionals remotely connect to the trusted cloud server and precisely monitor, analyze, and diagnose the health data of patients stored within the server. However, despite the numerous advantages of wearable computing, there are several difficulties and challenges that need to

be addressed [5]. In wearable computing environments, serious security and privacy issues may arise since the messages are transmitted via an insecure channel [6]. If the collected data from the wearable devices are exposed, an adversary can obtain the sensitive information of legitimate patients and may attempt potential cyber security attacks. Hence, adversaries can bring many unexpected threats and jeopardize the patients' lives by transmitting false medical diagnoses, such as "treatments and medications". In addition to cyber security attacks, wearable devices cannot prevent physical threats since they are deployed in hostile and unattended circumstances. Furthermore, considering the resource limitations of wearable devices, it is suitable to adopt lightweight cryptographic primitives, such as "hash functions and symmetric key cryptography that require low computation and communication costs [7]". In wearable computing environments, it is essential to identify whether data collected from multiple wearable devices belong to the same authorized user. Thus, a lightweight privacy-preserving authentication and group–proof scheme is indispensable to ensure simultaneous identification and secure communication in wearable computing environments.

Recently, Guo et al. [8] presented an "anonymous authenticated key agreement and group–proof protocol for wearable computing" to provide secure communication and simultaneous identification. Guo et al. claimed that their scheme was protected against physical/cyber security attacks, including "physical wearable device capture, impersonation, and forgery" attacks, and guaranteed "secure mutual authentication and untraceability". Unfortunately, we prove that Guo et al.'s scheme was not protected against security attacks, such as "session key disclosure, man-in-the-middle (MITM), and impersonation" attacks, and it does not offer several security properties, including "untraceability and mutual authentication". Hence, we present a "new efficient and robust authentication and group–proof scheme using physical unclonable functions (PUF) for wearable computing", denoted as AGPS-PUFs, to address the security issues of Guo et al.'s scheme [8].

1.1. Motivations

The main purpose of this paper is to identify and improve the security problems of Guo et al.'s scheme based on the threat model presented by them. This paper proves that their protocol [8] is not protected against lethal security attacks and does not offer sensitive security features in wearable computing environments. Guo et al. [8] designed a high-security-supported cryptographic and efficient group–proof scheme for wearable computing. However, they should have examined their protocol from the point of view that we analyzed and proved. This fact motivated us to design a "new efficient and robust authentication and group–proof scheme using PUF for wearable computing". This scheme is resilient to lethal security attacks and drawbacks that exist in wearable computing environments while guaranteeing security functionalities.

1.2. Research Contributions

This section introduces the main contribution of the AGPS-PUF.

- The AGPS-PUF is specifically designed to improve the security vulnerabilities of Guo et al.'s scheme and offers reliable authentication and maintenance for wearable computing. The AGPS-PUF carries out mutual authentication between a mobile user and wearable devices through a trusted entity known as the cloud server. The PUF enables wearable devices to resist tampering, including physical security attacks.
- We propose the protocol and demonstrate its effectiveness and security strengths via informal and formal security analyses. We exploited the well-known "AVISPA simulation" [9] and "ROR Oracle model" [10].
- We prove that the AGPS-PUF offers efficient performance in terms of security functionalities and overheads, as compared to previous schemes explored in the literature.

1.3. Paper Outlines

The rest of the paper is organized as follows. Section 2 presents the related works for wearable computing environments. Section 3 introduces the preliminaries. In Sections 4 and 5, we review Guo et al.'s scheme [8] and then prove the security shortcomings of Guo et al.'s scheme. Section 6 designs a "new PUF-based privacy-preserving authentication and group–proof scheme for wearable computing" to resolve the security problems of Guo et al.'s scheme. Section 7 analyzes the security of the AGPS-PUF when performing formal and informal security analyses. Section 8 introduces the testbed experiments for cryptographic operations using MIRACL crypto SDK. Section 9 analyzes the performance comparison of the AGPS-PUF with related schemes. Finally, Section 10 summarizes the future works and conclusions of this paper.

2. Related Works

Over the last few years, many authentication and key agreement (AKA) schemes have been presented for wearable computing to ensure privacy for legitimate users [11–13]. The public key cryptosystem (PKC)-based AKA schemes consist of three mechanisms: "traditional PKC scheme [14], identity-based PKC scheme [15], and certificateless PKC scheme [16]". The traditional PKC scheme faces problems in managing user certificates and needs high computing capabilities, so it is not applicable to wearable computing environments with constrained resources. Identity-based PKC schemes deal with the difficulty of certificate management; however, they are presented for server–client environments. The certificateless PKC scheme enhances the key escrow problem of the identity-based PKC scheme and prevents certificate management and delivery problems from the traditional PKC scheme [17]. However, these existing PKC-based AKA schemes [14–16] are not suitable for wearable computing environments because they utilize PKC, such as elliptic curve cryptography (ECC) and bilinear pairing, which require high communication and computation overheads.

The design of a lightweight AKA scheme for wearable computing environments has garnered a lot of attention due to the efficiency problem of the PKC-based AKA scheme and constrained resources for IoT and sensor devices. The lightweight AKA scheme has two main features: "password-based two-factor AKA scheme or password and biometric-based three-factor AKA scheme". These AKA schemes utilize lightweight cryptographic primitives, including the "one-way hash function, XOR operation, and symmetric key cryptography". Recently, numerous lightweight AKA schemes [18–20] were designed for wearable computing environments to provide useful services with lightweight properties. Li et al. [21] proposed a "secure AKA scheme with user anonymity and lightweight for healthcare applications" in wireless medical sensor networks (WMSN). Unfortunately, Das et al. [22] demonstrated that Li et al.'s scheme [21] is insecure to "privileged insider and sensor node capture" attacks and fails to ensure "user anonymity". Wu et al. [23] presented an "enhanced two-factor assisted AKA scheme in WMSN environments". Wu et al. [23] claimed that their protocol is resilient to lethal security attacks and offers the necessary security features. Unfortunately, Srinivas et al. [24] proved that their scheme [23] is not resistant to lethal security attacks, such as "stolen smart card, offline password guessing, user impersonation, and denial of service (DoS)" attacks. Srinivas et al. [24] proposed an "efficient and reliable AKA scheme for healthcare services with WMSN" to address the security weaknesses of Wu et al.'s scheme [23]. Amin et al. [25] designed a "lightweight and anonymous two-factor based AKA scheme" to provide secure patient data in patient monitoring systems for WMSN. Unfortunately, Ali et al. [26] analyzed Amin et al.'s scheme [25] and found that it does not prevent "known-session key temporary information, user impersonation, and offline password guessing" attacks. Ali et al. [26] presented an "enhanced biometric-based three-factor AKA scheme for healthcare monitoring in WMSN" to resolve the security shortcomings of Amin et al.'s scheme [25]. Gupta et al. [27] designed a "lightweight AKA scheme for wearable devices with user anonymity". Gupta et al.'s scheme [27] has high scalability because the wearable sensing device registration phase

does not need a secure channel. However, Hajian et al. [28] proved that Gupta et al.'s scheme [27] is not resistant to lethal security attacks, including "compromise sensing device, desynchronization, and privileged insider" attacks. Hajian et al. [28] proposed a "scalable and lightweight three-factor based AKA scheme with user-friendly and anonymous for wearable sensing devices" to improve the security problems of Gupta et al.'s scheme [27]. However, Yu et al. [29] pointed out that their protocol [28] is still not resistant to "mobile device stolen", "session key disclosure, MITM, impersonation" attacks and does not guarantee "mutual authentication". Unfortunately, these lightweight AKA schemes for wearable computing do not identify whether the collected data from multiple wearable devices belong to the same authorized user.

Guo et al. [8] designed an "anonymous and lightweight AKA and group–proof scheme for wearable computing", which can verify that multiple wearable devices belong to the same user. Guo et al. [8] claimed that their protocol ensures secure data transmission between each entity and is resilient to lethal security attacks. However, based on the threat model presented by them, we have proven that Guo et al.'s scheme [8] is vulnerable to lethal security threats, such as "impersonation, MITM, and session key disclosure" attacks, and does not offer several security properties, such as "untraceability and mutual authentication". In addition to cyber security attacks, wearable devices may be fragile to physical threats since they are batched in insecure circumstances. Therefore, we propose an "efficient and robust authentication and group-proof scheme using the PUF for wearable computing" to supplement the security functionalities and address the security shortcomings of Guo et al.'s scheme [8].

3. Preliminaries

The following provides an overview of the preliminaries.

3.1. Threat Model

We introduce the adversary capabilities based on the "Dolev-Yao (DY) model" [30,31].

- An adversary (henceforth denoted as $\mathcal{A}$) can "resend, eavesdrop, block, and delete" the exchanged messages over an insecure channel.
- $\mathcal{A}$ can steal the mobile device (MD) and the wearable device (WD) of the legitimate user. However, $\mathcal{A}$ cannot simultaneously capture the MD and WD of the legitimate user. The cloud server and registration center are trusted authorities and cannot be compromised by $\mathcal{A}$.
- $\mathcal{A}$ can extract the secret information stored in the captured MD or WD by performing the "power-analysis attacks" [32] and "physical capture attacks" [33].

3.2. PUF

The PUF [34,35] is a physical circuit that manufactures an output of a physical microstructure. The PUF does not store a private key in the smart device and it is extremely difficult to clone the circuit. The PUF utilizes an input/output bit string pair, denoted as the challenge/response pair. Even if various challenges occur in the PUF circuit, each has a unique output response. The PUF preserves smart devices in IoMT-enabled TMIS environments from side-channel and tampering threats. The PUF is expressed through a process denoted as $R = PUF(C)$, where C and R are the challenge/response. The following are several properties of the PUF.

- The PUF is easy to implement and evaluate.
- The PUF relies on the system's physical microstructure.
- Any attempt to tamper with a smart device that contains the PUF will update the behavior of the PUF and, thus, destroy it [36].

Figure 1 shows a "PUF-based key generator procedure". As shown in Figure 1, the PUF generates strong extractors for a private secret key based on various functions, including "encode, decode, and key derivation" functions. Thus, the PUF makes it impossible for

attackers to perform lethal physical threats. Moreover, these properties combine to make a "good solution for the robust and efficient authentication of lightweight devices in wearable computing environments".

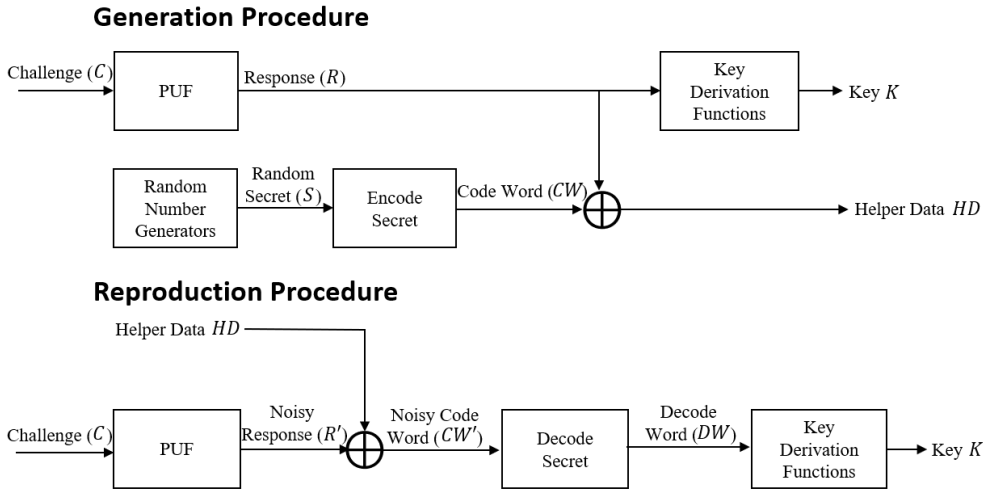

Figure 1. Key generator mechanism of the PUF.

3.3. System Model

This section introduces an overview of the system model (see Figure 2) of this paper. The system model for wearable computing is composed of four entities: registration center, cloud server, mobile users, and wearable devices.

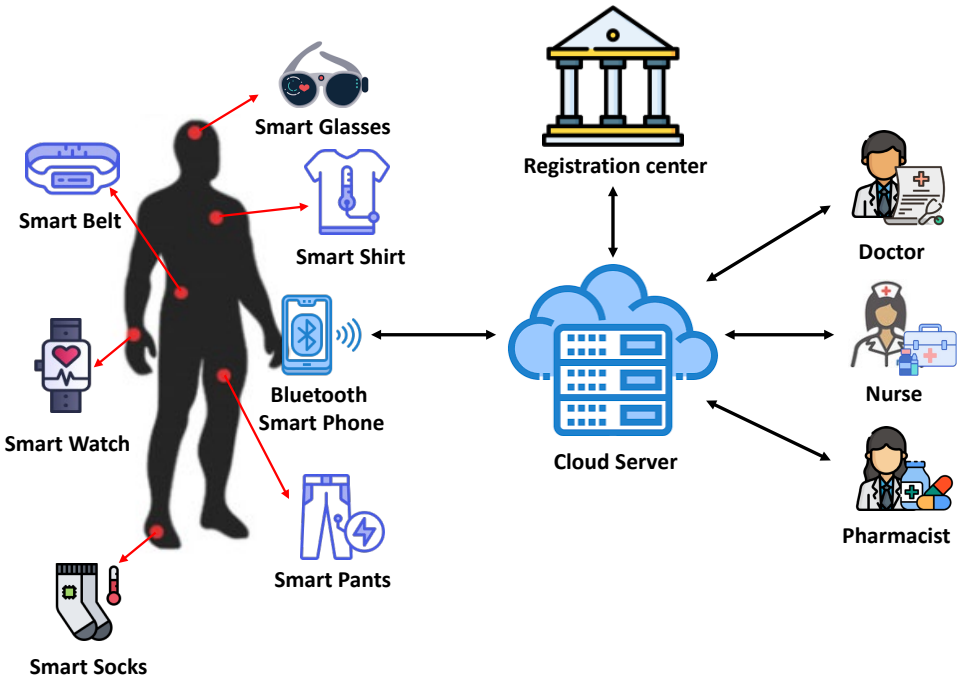

Figure 2. System model for wearable computing.

- Registration center: This entity is a trusted authority that registers wearable devices and mobile users in a secure channel. Moreover, the registration center sets the secret credentials of each wearable device before being batched in wearable computing environments.

- Cloud server: This entity is also a trusted authority. The cloud server stores and shares the health data of legitimate patients and has computational and storage capabilities to manage patients' health data.
- Mobile users: They have a mobile terminal and wear wearable devices to analyze the health status of the patients. The mobile terminal receives health data from the wearable devices, and then sends the received data to the cloud server through wireless communications. Moreover, remote authorized users access the cloud server to analyze the patients' data and provide accurate medical diagnoses based on the stored physiological data.
- Wearable device: Wearable devices track and collect health data from corresponding body parts of patients. Then, the collected data are transmitted to the paired mobile terminal via Bluetooth.

4. Review of Guo et al.'s Scheme

This section introduces the reviews of Guo et al.'s scheme [8]. Table 1 shows the notations utilized in this article.

Table 1. Notations.

Symbol	Meaning
U_i	ith user
WD_j	jth wearable device
CS	Cloud server
RC	Registration center
$ID_U, ID_{WD}, ID_{CS}, ID_{RC}$	Real identity of U_i, WD_j, CS, and RC
PW_i	Password of U_i
C_U^x, R_U^x	Challenge/response of U_i
C_{WD}^x, R_{WD}^x	Challenge/response of WD_j
$R_i, RN_i, n_i,$	Random nonce
$TID_U, TID_{WD},$	Temporary identity of U_i and WD_j
ΔT_i	Maximum transmission delay
T_i and TS_i	Timestamp
MK	A master private key of CS
SK_{CS-U}	A session key for U_i and CS
SK_{WD-U}	A session key for U_i and WD_j
$Enc_K(\cdot)/Dec_K(\cdot)$	Encryption/decryption
$h(\cdot)$	Hash function
$\oplus$	XOR function
$\|\|$	Concatenation

4.1. System Setup Phase

In this section, RC denotes the secret credentials of each WD_j.

SP-1: RC select s a master private key MK for CS.

SP-2: RC chooses a unique identity ID_{WD} for each WD_j and computes the pseudo-identity $PID_{WD} = h(ID_{WD}||MK)$. After that, RC generates a temporary identity TID_{WD} for each WD_j.

SP-3: RC stores $\{ID_{WD}, TID_j^{old} = null, TID_j^{new} = TID_{WD}\}$ in CS's secure database and then stores $\{PID_{WD}, TID_{WD}, h(\cdot)\}$ in the memory of WD_j.

4.2. User Registration Phase

In this phase, U_i registers with RC and obtains certain secret information to utilize later for authentication.

URP-1: U_i selects ID_U and PW_i at MD_i and WD_j. After that, MD_i generates a random number RN_i and computes $HID_U = h(ID_U||RN_i)$ and $HPW_i = h(PW_i||RN_i)$. Then, MD_i sends $\{HID_U, HPW_i\}$ to RC over a secure channel.

URP-2: RC selects a temporary identity TID_U and a random number r_i. After that, RC calculates $Auth^* = h(HID_U||HPW_i||r_i)$, $A_i = r_i \oplus h(HID_U \oplus HPW_i)$, $B_i = h(TID_U||r_i||MK)$, $C_i = r_i \oplus h(TID_U||MK)$. Finally, RC stores $\{TID_j^{old} = null, TID_j^{new} = TID_{WD}, C_i\}$ in CS's secure database and then sends $\{Auth^*, TID_U, A_i, B_i\}$ to MD_i over a secure channel.

URP-3: MD_i computes $HAuth = h(Auth^*||HPW_i \oplus RN_i)$, $D_i = RN_i \oplus A_i$, $E_i = B_i \oplus h(TID_U \oplus HPW_i \oplus RN_i)$, and $F_i = RN_i \oplus h(ID_U||PW_i)$. Finally, MD_i stores $\{TID_U, HAuth, D_i, E_i, F_i, h(\cdot)\}$ in its memory.

4.3. Login and Authentication Phase

In this phase, all participants authenticate each other and establish a common session key.

LAP-1: U_i first inputs ID_U and PW_i into MD_i. After that, MD_i computes $RN_i = F_i \oplus h(ID_U||PW_i)$, $HID_i^* = h(ID_U||RN_i)$, $HPW_i^* = h(PW_i||RN_i)$, $R_i^* = A_i \oplus h(HID_i^* \oplus HPW_i^*) = D_i \oplus RN_i \oplus h(HID_i^* \oplus HPW_i^*)$, $Auth^* = h(HID_i^*||HPW_i^*||R_i^*)$, and $HAuth^* = h(Auth^* \oplus HPW_i^* \oplus RN_i)$, and checks $HAuth^* \stackrel{?}{=} HAuth$. If it matches, MD_i generates a random nonce n_1 and a current timestamp T_1 and then transmits $Msg_1 = \{n_1, T_1\}$ to WD_j over an insecure channel.

LAP-2: WD_j verifies the freshness of $|T_2 - T_1| \leq \Delta T_i$, where T_2 is the current timestamp and ΔT_1 is the maximum transmission delay for the message to be transmitted between MD_i and WD_j. If they match, WD_j selects n_2 and computes $M_1 = h(TID_{WD}||PID_{WD}) \oplus n_2$ to transmit the secret parameters securely, and $M_2 = h(M_1||T_2||n_1||n_2)$ to verify the authorized entity. After that, WD_j transmits $Msg_2 = \{M_1, M_2, TID_{WD}, T_2\}$ to MD_i.

LAP-3: MD_i checks the freshness of $|T_3 - T_2| \leq \Delta T_i$. If the condition is met, MD_i selects n_3 and computes $B_i = E_i \oplus h(TID_U \oplus HPW_i \oplus RN_i)$, $M_3 = n_3 \oplus h(TID_U||r_i||T_3)$ to transmit the random nonce securely, $M_4 = h(TID_U||B_i||n_3||T_3)$ to verify the authorized entity, $M_5 = h(TID_U||TID_{WD}) \oplus n_1$ to transmit the secret parameters securely, and $M_6 = h(TID_U||TID_{WD}||M_2||M_4||n_3||T_3)$ to verify the authorized entity, and sends $Msg_3 = \{M_1, M_2, M_3, M_4, M_5, M_6, TID_U, TID_{WD}, T_2, T_3\}$ to CS over an insecure channel.

LAP-4: CS verifies the freshness of $|T_4 - T_3| \leq \Delta T_i$. If it matches, CS retrieves TID_U in the database. There are three scenarios for TID_U. The first scenario is $TID_i^{old} = TID_U$, indicating that CS and U_i did not correctly update the temporary identity of U_i in the previous session. The second scenario is $TID_i^{new} = TID_U$, indicating that CS and U_i correctly updated the temporary identity of U_i in the previous session. In the third scenario, there is no matching of TID_U in the CS database, and the authentication phase is terminated. For the first two scenarios, CS obtains $\{C_i\}$, corresponding to TID_U in its database. After that, CS computes $r_i^* = C_i \oplus h(TID_U||MK)$, $B_i^* = h(TID_U||r_i^*||MK)$, $n_3^* = M_3 \oplus h(TID_U||r_i^*||T_3)$, $M_4^* = h(TID_U||B_i^*||n_3^*||T_3)$, and $M_6^* = h(TID_U||TID_{WD}||M_2||M_4^*||n_3^*||T_3)$, and verifies $M_4^* \stackrel{?}{=} M_4$ and $M_6^* \stackrel{?}{=} M_6$. If they are not equal, the authentication phase is terminated. Otherwise, CS successfully

authenticates U_i and then updates the temporary identity of U_i. For the second scenario, U_i's new temporary identity remains unchanged for the time being and is updated later in the session.

LAP-5: CS retrieves TID_{WD} in its database. Similar to LAP-4, there are three scenarios: $TID_j^{old} = TID_{WD}$, $TID_j^{new} = TID_{WD}$, or TID_{WD} cannot be found in the CS database. In the first two scenarios, CS obtains $\{ID_{WD}\}$, corresponding to TID_{WD} in its database, and then computes $PID_{WD} = h(ID_{WD}||MK)$, $n_2^* = h(TID_{WD}||PID_{WD}) \oplus M_1$, $n_1^* = h(TID_U \oplus TID_{WD}) \oplus M_5$, and $M_2^* = h(M_1||T_2||n_1^*||n_2^*)$, and checks $M_2^* \stackrel{?}{=} M_2$. If it matches, CS successfully authenticates WD_j. Then, CS updates the temporary identity of WD_j as it updates U_i's temporary identity.

LAP-6: CS selects n_4 and timestamp T_4. After that, CS computes $M_7 = B_i \oplus T_4 = h(TID_U||r_i||MK) \oplus T_4$ to transmit the secret parameters securely, $M_8 = n_4 \oplus h(TID_U||r_i||n_3||T_4)$ to transmit the secret parameters securely, $SK_{CS-U} = h(TID_U||M_7||n_3||n_4||T_3||T_4)$, $M_9 = h(M_4||M_8||SK_{CS-U}||T_3||T_4)$ to verify the authorized entity, $SK_{WD-U} = h(TID_U||TID_{WD}||h(ID_{WD}||MK)||n_2||n_4||T_4)$, $M_{10} = h(M_4||M_8||SK_{WD-U}||T_3||T_4)$to verify the authorized entity, and $M_{11} = SK_{WD-U} \oplus h(B_i||n_1||n_3||n_4||T_3||T_4)$. CS selects the new temporary identities TID_i^{new} and TID_j^{new} for U_i and WD_j, and then changes $(TID_i^{old} = TID_U, TID_i^{new} = TID_i^{new})$ and $(TID_j^{old} = TID_{WD}, TID_j^{new} = TID_j^{new})$ in its database. Then, CS calculates $TID_i^* = TID_i^{new} \oplus h(TID_U||n_4||SK_{CS-U})$ and $TID_j^* = TID_j^{new} \oplus h(TID_{WD}||n_4||SK_{WD-U})$ and transmits $Msg_4 = \{M_8, M_9, M_{10}, M_{11}, TID_i^*, TID_j^*, T_4\}$ to MD_i over an insecure channel.

LAP-7: MD_i verifies the freshness of $|T_5 - T_4| \leq \Delta T_i$. If it matches, U_i calculates $n_4 = M_8 \oplus h(TID_U||n_3||T_4)$, $M_7^* = B_i^* \oplus T_4 = E_i \oplus h(TID_U \oplus HPW_i \oplus RN_i) \oplus T_4$, $SK_{CS-U}^* = h(TID_U||M_7^*||n_3||n_4||T_3||T_4)$, $SK_{WD-U}^* = M_{11} \oplus h(B_i||n_1||n_3||n_4||T_3||T_4)$, $M_9^* = h(M_4||M_8||SK_{CS-U}^*||T_3||T_4)$, and $M_{10}^* = h(M_4||M_8||SK_{WD-U}^*||T_3||T_4)$, and then checks whether $M_9^* \stackrel{?}{=} M_9$ and $M_{10}^* \stackrel{?}{=} M_{10}$. If they are valid, MD_i authenticates CS. After that, MD_i stores the session keys, SK_{CS-U}^* and SK_{WD-U}^*, and the new temporary identity, $TID_i^{new} = TID_i^* \oplus h(TID_U||n_4||SK_{CS-U}^*)$.

LAP-8: MD_i selects n_5 and computes $M_{12} = n_4 \oplus h(TID_{WD}||n_1||T_4)$ to transmit the secret parameters securely, $M_{13} = n_5 \oplus h(TID_{WD}||n_1||T_5)$ to transmit the secret parameters securely, $M_{14} = h(TID_U||SK_{WD-U}||n_5||T_5)$ to verify the authorized entity, and then transmits $Msg_5 = \{TID_U, TID_j^*, M_{12}, M_{13}, M_{14}, T_4, T_5\}$ to WD_j.

LAP-9: WD_j verifies the freshness $|T_6 - T_5| \leq \Delta T_i$. If it matches, WD_j computes $n_4 = M_{12} \oplus h(TID_{WD}||n_1||T_4)$, $n_5 = M_{13} \oplus h(TID_{WD}||n_1||T_5)$, $SK_{WD-U}^* = h(TID_U||TID_{WD}||h(ID_{WD}||MK)||n_2||n_4||T_4)$, and $M_{14}^* = h(TID_U||SK_{WD-U}||n_5||T_5)$, and then checks whether $M_{14}^* \stackrel{?}{=} M_{14}$. If it matches, WD_j authenticates U_i successfully. Finally, WD_j stores a session key SK_{WD-U}^* and a new temporary identity $TID_j^* = TID_j^{new} \oplus h(TID_{WD}||n_4||SK_{WD-U}^*)$.

5. Security Flaws of Guo et al.'s Scheme

In this section, we prove that Guo et al.'s scheme [8] is not protected against the lethal security threats and cannot offer several security functionalities.

5.1. Impersonation Attack

According to Section 3.1, $\mathcal{A}$ can extract the secret credentials $\{PID_{WD}, TID_{WD}\}$ stored in WD_j. Moreover, $\mathcal{A}$ can intercept, block, modify, replay, and delete the exchanged messages over an insecure channel. In this attack, $\mathcal{A}$ attempts to impersonate a legitimate entity.

- **Step 1:** $\mathcal{A}$ first calculates $n_2 = M_1 \oplus h(TID_{WD}||PID_{WD})$ and a new random nonce n_2^A. After that, $\mathcal{A}$ computes $M_1^A = h(TID_{WD}||PID_{WD}) \oplus n_2^A$ and $M_2^A = h(M_1^A||T_2||n_1||n_2^A)$. After that, $\mathcal{A}$ transmits the message $\{M_1^A, M_2^A, TID_{WD}, T_2\}$ to CS via MD_i.

- **Step 2:** After receiving the message, CS retrieves TID_{WD} in its database and then obtains $\{ID_{WD}\}$, corresponding to TID_{WD} in its database. Then, CS calculates $PID_{WD} = h(ID_{WD}||MK)$, $n_2^{A*} = h(TID_{WD}||PID_{WD}) \oplus M_1^A$, $n_1^* = h(TID_U \oplus TID_{WD}) \oplus M_5$, and $M_2^{A*} = h(M_1^A||T_2||n_1^*||n_2^{A*})$, and checks $M_2^{A*} \stackrel{?}{=} M_2^A$. If it matches, CS authenticates $\mathcal{A}$, successfully.
- **Step 3:** CS generates a random nonce n_4 and timestamp T_4. After that, CS computes $M_7 = B_i \oplus T_4 = h(TID_U||r_i||MK) \oplus T_4$, $M_8 = n_4 \oplus h(TID_U||r_i||n_3||T_4)$, $SK_{CS-U} = h(TID_U||M_7||n_3||n_4||T_3||T_4)$, $M_9 = h(M_4||M_8||SK_{CS-U}||T_3||T_4)$, $SK_{WD-U}^A = h(TID_U||TID_{WD}||h(ID_{WD}||MK)||n_2^A||n_4||T_4)$, $M_{10}^A = h(M_4||M_8||SK_{WD-U}^A||T_3||T_4)$, and $M_{11}^A = SK_{WD-U}^A \oplus h(B_i||n_1||n_3||n_4||T_3||T_4)$. CS selects the new temporary identities, TID_i^{new} and TID_j^{new} for U_i and WD_j, and then changes $(TID_i^{old} = TID_U, TID_i^{new} = TID_i^{new})$, and $(TID_j^{old} = TID_{WD}, TID_j^{new} = TID_j^{new})$ in its database. Then, CS calculates $TID_i^* = TID_i^{new} \oplus h(TID_U||n_4||SK_{CS-U})$ and $TID_j^{A*} = TID_j^{new} \oplus h(TID_{WD}||n_4||SK_{WD-U}^A)$ and transmits $\{M_8, M_9, M_{10}^A, M_{11}^A, TID_i^*, TID_j^{A*}, T_4\}$ to MD_i over an open channel.
- **Step 4:** Upon receiving the message, MD_i verifies the freshness of $|T_5 - T_4| \leq \Delta T_i$. If it matches, U_i calculates $n_4 = M_8 \oplus h(TID_U||n_3||T_4)$, $M_7^* = B_i^* \oplus T_4 = E_i \oplus h(TID_U \oplus HPW_i \oplus RN_i) \oplus T_4$, $SK_{CS-U}^* = h(TID_U||M_7^*||n_3||n_4||T_3||T_4)$, $SK_{WD-U}^{A*} = M_{11}^A \oplus h(B_i||n_1||n_3||n_4||T_3||T_4)$, $M_9^* = h(M_4||M_8||SK_{CS-U}^*||T_3||T_4)$, and $M_{10}^{A*} = h(M_4||M_8||SK_{WD-U}^{A*}||T_3||T_4)$, and then checks whether $M_9^* \stackrel{?}{=} M_9$ and $M_{10}^{A*} \stackrel{?}{=} M_{10}^A$. If they are valid, MD_i authenticates CS. After that, MD_i stores the session keys SK_{CS-U}^* and SK_{WD-U}^{A*} and the new temporary identity $TID_i^{new} = TID_i^* \oplus h(TID_U||n_4||SK_{CS-U}^*)$.
- **Step 5:** Then, MD_i selects n_5 and computes $M_{12} = n_4 \oplus h(TID_{WD}||n_1||T_4)$, $M_{13} = n_5 \oplus h(TID_{WD}||n_1||T_5)$, $M_{14}^A = h(TID_U||SK_{WD-U}^A||n_5||T_5)$, and then transmits $\{TID_U, TID_j^{A*}, M_{12}, M_{13}, M_{14}^A, T_4, T_5\}$ to WD_j.
- **Step 6:** After eavesdropping on the message, $\{TID_U, TID_j^{A*}, M_{12}, M_{13}, M_{14}^A, T_4, T_5\}$, $\mathcal{A}$ calculates $n_4 = M_{12} \oplus h(TID_{WD}||n_1||T_4)$, $n_5 = M_{13} \oplus h(TID_{WD}||n_1||T_5)$, $SK_{WD-U}^{A*} = h(TID_U||TID_{WD}||h(ID_{WD}||MK)||n_2^A||n_4||T_4)$, and $M_{14}^{A*} = h(TID_U||SK_{WD-U}^A||n_5||T_5)$. Note that $h(ID_{WD}||MK)$, included in the session key, is the same as PID_{WD}. Finally, $\mathcal{A}$ stores a session key SK_{WD-U}^{A*} and a new temporary identity $TID_j^{A*} = TID_j^{new} \oplus h(TID_{WD}||n_4||SK_{WD-U}^{A*})$.

Consequently, their scheme is not resistant to impersonation attacks since $\mathcal{A}$ can impersonate the legitimate WD_j.

5.2. MITM Attack

Based on the threat model, $\mathcal{A}$ can extract the secret parameters $\{PID_{WD}, TID_{WD}\}$ stored in WD_j. Furthermore, $\mathcal{A}$ can block, intercept, modify, replay, and delete the transmitted messages via an open channel.

- **Step 1:** After eavesdropping on the message $\{n_1, T_1\}$ via a public channel, $\mathcal{A}$ first calculates $n_2 = M_1 \oplus h(TID_{WD}||PID_{WD})$ and $M_2 = h(M_1||T_2||n_1||n_2)$. After that, $\mathcal{A}$ transmits $\{M_1, M_2, TID_{WD}, T_2\}$.
- **Step 2:** After eavesdropping on the message $\{TID_U, TID_j^*, M_{12}, M_{13}, M_{14}, T_4, T_5\}$ via a public channel, $\mathcal{A}$ computes $n_4 = M_{12} \oplus h(TID_{WD}||n_1||T_4)$ and $n_5 = M_{13} \oplus h(TID_{WD}||n_1||T_5)$.
- **Step 3:** $\mathcal{A}$ calculates a session key $SK_{WD-U}^* = h(TID_U||TID_{WD}||h(ID_{WD}||MK)||n_2||n_4||T_4)$, where $h(ID_{WD}||MK)$, included in the session key, is the same as PID_{WD}. Finally, $\mathcal{A}$ successfully calculates $M_{14}^* = h(TID_U||SK_{WD-U}||n_5||T_5)$ and then verifies $M_{14}^* \stackrel{?}{=} M_{14}$. Hence, their scheme is not protected against this attack.

5.3. Session Key Disclosure Attack

Based on Section 5.2, $\mathcal{A}$ extracts $n_4 = M_{12} \oplus h(TID_{WD}||n_1||T_4)$ and $n_5 = M_{13} \oplus h(TID_{WD}||n_1||T_5)$, and then computes a session key $SK^*_{WD-U} = h(TID_U||TID_{WD}||h(ID_{WD}||MK)||n_2||n_4||T_4)$ successfully. As a result, $\mathcal{A}$ can successfully obtain a common session key SK^*_{WD-U} between legitimate U_i and WD_j. Thus, Guo et al.'s scheme is insecure to this attack.

5.4. Mutual Authentication

In Guo et al.'s scheme, they claimed to provide mutual authentication between the entities. Unfortunately, according to Sections 5.2 and 5.3, $\mathcal{A}$ can successfully generate the sensitive messages, $M_{10} = h(M_4||M_8||SK_{WD-U}||T_3||T_4)$ and $M_{14} = h(TID_U||SK^A_{WD-U}||n_5||T_5)$, for mutual authentication. Thus, Guo et al.'s scheme cannot guarantee secure mutual authentication between the legitimate U_i and WD_j.

5.5. Untraceability

Guo et al. claimed that their protocol achieved untraceability. However, according to Sections 5.2 and 5.3, $\mathcal{A}$ calculates the random nonces $n_4 = M_{12} \oplus h(TID_{WD}||n_1||T_4)$ and $n_5 = M_{13} \oplus h(TID_{WD}||n_1||T_5)$ and then computes a session key $SK^*_{WD-U} = h(TID_U||TID_{WD}||h(ID_{WD}||MK)||n_2||n_4||T_4)$. After that, $\mathcal{A}$ successfully calculates a new temporary identity $TID^{new}_j = TID^*_j \oplus h(TID_{WD}||n_4||SK^*_{WD-U})$. Thus, Guo et al.'s scheme does not achieve untraceability because $\mathcal{A}$ can trace the authorized WD_j through their new temporary identity.

6. Proposed Scheme

The existing related schemes for wearable computing are not protected against potential security attacks. Thus, we propose a "robust and efficient authentication and group–proof scheme using the PUF for wearable computing (AGPS-PUF)" to improve the security flaws of the existing schemes. The AGPS-PUF is resilient to cyber/physical security attacks and provides necessary security functionalities. The AGPS-PUF consists of six phases: (1) system setup, (2) registration, (3) login and authentication, (4) group proof, and (5) password update. We show the overall flowchart during the AKA phase of the AGPS-PUF, as shown in Figure 3.

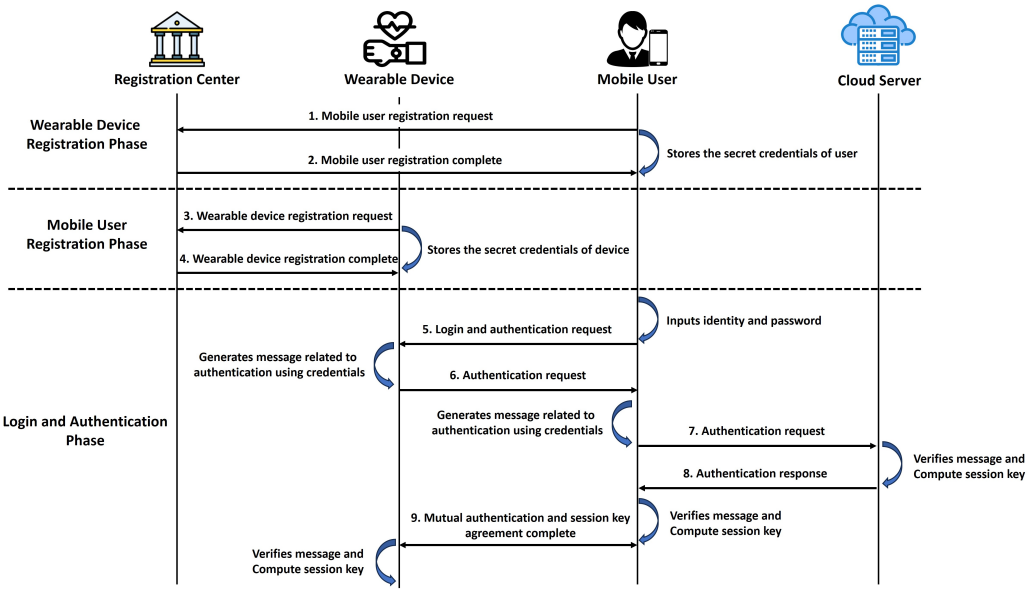

Figure 3. The overall flowchart during the AKA phase.

6.1. System Setup Phase

In this section, RC first sets the secret credentials for each WD_j. The following are detailed descriptions:

SP-1: RC selects a master private key MK for CS.

SP-2: RC chooses a unique identity ID_{WD} for each WD_j and then generates a temporary identity TID_{WD} for each WD_j.

SP-3: RC stores $\{ID_{WD}, TID_{WD}\}$ in CS's secure database and then stores the secret credentials $\{ID_{WD}, TID_{WD}, h(\cdot)\}$ in the memory of WD_j.

6.2. Registration Phase

This phase consists of two parts: U_i and WD_j registration phases.

6.2.1. User Registration Phase

In this phase, U_i registers within RC and then obtains the secret credentials from RC.

URP-1: U_i chooses unique ID_U and PW_U in MD_i. After that, MD_i selects a random number a_i and generates a set of (C_U^x, R_U^x) based on the PUF to ensure the unique physical properties of the device. Then, MD_i computes $HID_U = h(ID_U||a_i)$ and $HPW_i = h(PW_U||a_i||R_U^x)$ and then transmits $\{HID_U, HPW_i, (C_U^x, R_U^x)\}$ to RC over a secure channel.

URP-2: RC generates a temporary identity TID_U and computes $X_i = h(TID_U||MK||R_U^x)$, $X_{UW} = h(ID_{RC}||MK)$, $A_i = (X_i||X_{UW}) \oplus h(HID_U \oplus h(HPW_i||R_U^x))$, $B_i = h(HPW_i||TID_U||X_i)$, $C_i = X_i \oplus h(ID_{CS}||TID_U||MK)$, Finally, RC stores $\{TID_U, (C_U^x, R_U^x), C_i\}$ in CS's database and then sends $\{A_i, B_i, TID_U\}$ to MD_i over a secure channel.

URP-3: Finally, MD_i computes $D_i = a_i \oplus h(ID_U||PW_U||R_U^x)$ and stores $\{A_i, B_i, D_i, TID_U\}$ in its memory.

6.2.2. Wearable Device Registration Phase

In this phase, WD_j registers within RC and then obtains the secret credentials from RC.

WDRP 1: WD_j generates a random number b_j and a set (C_{WD}^x, R_{WD}^x) under the PUF to ensure the unique physical properties of the device. After that, WD_j calculates $Q_j = b_j \oplus h(ID_{WD}||R_{WD}^x)$ and $W_j = h(ID_{WD}||b_j||R_{WD}^x)$. After that, WD_j sends $\{TID_{WD}, Q_j, W_j, (C_{WD}^x, R_{WD}^x)\}$ to RC.

WDRP 2: RC retrieves the corresponding ID_{WD} stored in the database using TID_{WD}. After that, RC computes $b_j^* = Q_j \oplus h(ID_{WD}||R_{WD}^x)$, and $W_j^* = h(ID_{WD}^*||b_j^*||R_{WD}^x)$, and verifies $W_j^* \stackrel{?}{=} W_j$. If it is invalid, CS terminates WD_j's registration request; otherwise, RC computes $PID_{WD} = h(TID_{WD}||MK)$, $Z_j = h(TID_{WD}||MK||R_{WD}^x)$, $X_{UW} = h(ID_{RC}||MK)$, $E_j = (X_{UW}||Z_j||PID_{WD}) \oplus h(ID_{WD}||TID_{WD}||R_{WD}^x||b_j)$, and $Y_j = Z_j \oplus h(ID_{CS}||TID_{WD}||MK)$. After that, RC stores $\{Y_j, (C_{WD}^x, R_{WD}^x)\}$ in CS's secure database and then transmits $\{E_j\}$ to WD_j.

WDRP 3: Finally, WD_j computes $O_j = b_j \oplus h(R_{WD}^x \oplus TID_{WD} \oplus ID_{WD})$ and then stores $\{E_j, O_j\}$ in memory.

6.3. Login and Authentication Phase

The registered U_i and WD_j should establish a common session key with the help of CS to use reliable medical services. This phase is illustrated in Figure 4.

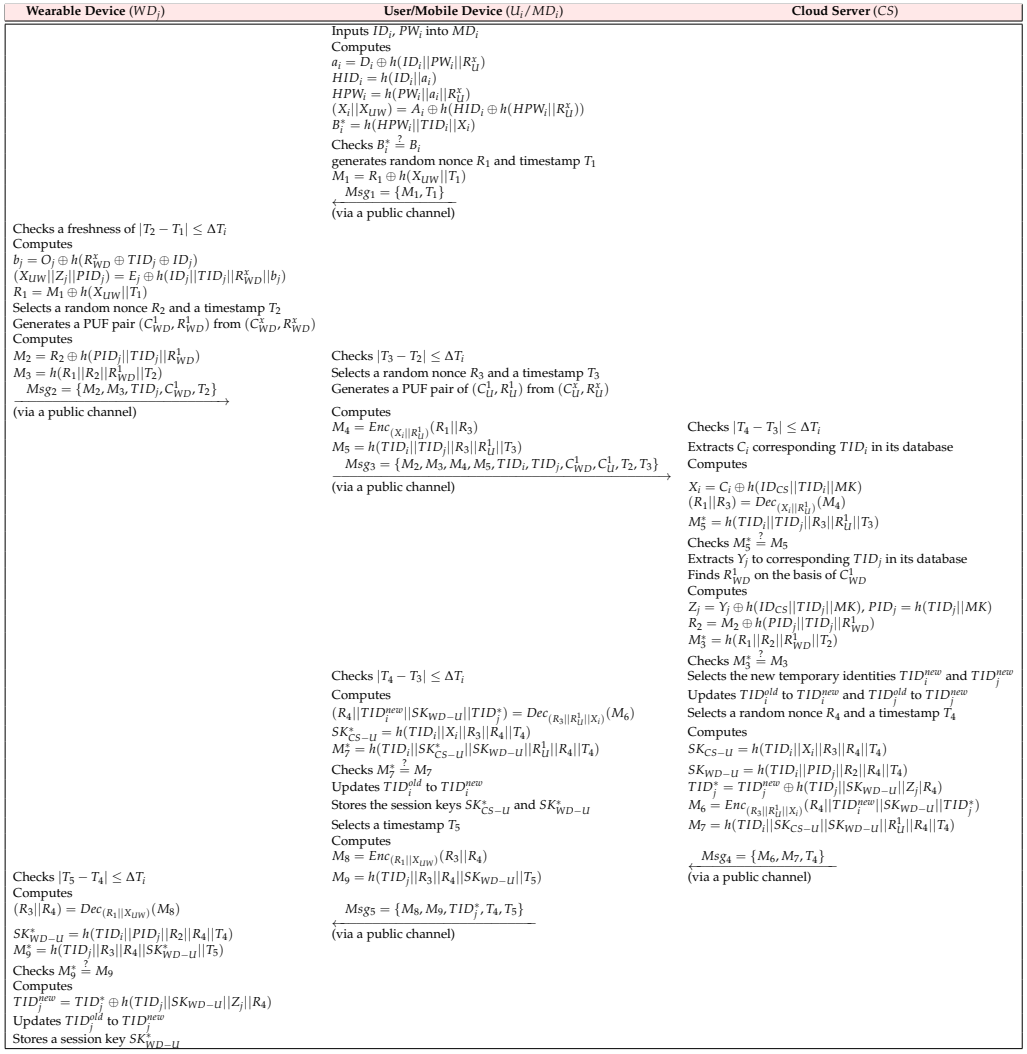

Figure 4. Login and Authentication Phase of the AGPS-PUF.

LAP-1: U_i first inputs a unique identity ID_U and password PW_i into MD_i. After that, MD_i calculates $a_i = D_i \oplus h(ID_U||PW_i||R_U^x)$, $HID_U = h(ID_U||a_i)$, $HPW_i = h(PW_i||a_i||R_U^x)$, $(X_i||X_{UW}) = A_i \oplus h(HID_U \oplus h(HPW_i||R_U^x))$, and $B_i^* = h(HPW_i||TID_U||X_i)$ and then checks $B_i^* \stackrel{?}{=} B_i$. If it matches, MD_i generates a random nonce R_1 and a timestamp T_1. Then, MD_i computes $M_1 = R_1 \oplus h(X_{UW}||T_1)$ to make the masked random nonce and transmits $Msg_1 = \{M_1, T_1\}$ to WD_j via an insecure channel.

LAP-2: WD_j checks the freshness of $|T_2 - T_1| \leq \Delta T_i$, where T_2 is the current timestamp and ΔT_1 is the maximum transmission delay for the message to be transmitted between MD_i and WD_j. If it matches, WD_j calculates $b_j = O_j \oplus h(R_{WD}^x \oplus TID_{WD} \oplus ID_{WD})$, $(X_{UW}||Z_j||PID_{WD}) = E_j \oplus h(ID_{WD}||TID_{WD}||R_{WD}^x||b_j)$, and $R_1 = M_1 \oplus h(X_{UW}||T_1)$. Then, WD_j selects R_2 and T_2. After that, WD_j chooses a pair of (C_{WD}^1, R_{WD}^1) from the preloaded CRPs (C_{WD}^x, R_{WD}^x) to ensure the unique physical properties of the device and computes $M_2 = R_2 \oplus h(PID_{WD}||TID_{WD}||R_{WD}^1)$ to make the masked random nonce, and $M_3 = h(R_1||R_2||R_{WD}^1||T_2)$ to verify the authorized entity, and then transmits $Msg_2 = \{M_2, M_3, TID_{WD}, C_{WD}^1, T_2\}$ to MD_i.

LAP-3: After receiving the message, MD_i verifies the freshness of $|T_3 - T_2| \leq \Delta T_i$. If it matches, MD_i generates R_3 and a timestamp T_3 and chooses a pair of (C_U^1, R_U^1) from the preloaded CRPs (C_U^x, R_U^x) to ensure the unique physical properties of the device. After that, MD_i decrypts $M_4 = Enc_{(X_i||R_U^1)}(R_1||R_3)$ to obtain the random nonce and calculates $M_5 = h(TID_U||TID_{WD}||R_3||R_U^1||T_3)$ to verify the authorized

entity, and then transmits $Msg_3 = \{M_2, M_3, M_4, M_5, TID_U, TID_{WD}, C^1_{WD}, C^1_U, T_2, T_3\}$ to CS through a public channel.

LAP-4: After receiving Msg_3 from MD_i, CS checks the freshness of $|T_4 - T_3| \leq \Delta T_i$. If it matches, CS finds R^1_U on the basis of C^1_U. After that, CS extracts C_i corresponding to TID_U in its database. CS decrypts $(R_1||R_3) = Dec_{(X_i||R^1_U)}(M_4)$, and computes $X_i = C_i \oplus h(ID_{CS}||TID_U||MK)$, and $M^*_5 = h(TID_U||TID_{WD}||R_3||R^1_U||T_3)$, and verifies $M^*_5 \stackrel{?}{=} M_5$. If it matches, CS aborts the current session; otherwise, CS extracts Y_j to the corresponding TID_{WD} in its database. Then, CS finds R^1_{WD} on the basis of C^1_{WD} and computes $Z_j = Y_j \oplus h(ID_{CS}||TID_{WD}||MK)$, $PID_{WD} = h(TID_{WD}||MK)$, $R_2 = M_2 \oplus h(PID_{WD}||TID_{WD}||R^1_{WD})$, and $M^*_3 = h(R_1||R_2||R^1_{WD}$ $||T_2)$, and verifies $M^*_3 \stackrel{?}{=} M_3$. If it matches, CS selects the new temporary identities, TID^{new}_i and TID^{new}_j for U_i and WD_j, and updates TID^{old}_i to TID^{new}_i and TID^{old}_j to TID^{new}_j in its database. CS generates R_4 and T_4 and computes $SK_{CS-U} = h(TID_U||X_i||R_3||R_4||T_4)$, $SK_{WD-U} = h(TID_U||PID_{WD}||R_2||R_4||T_4)$, $TID^*_j = TID^{new}_j \oplus h(TID_{WD}||$ $SK_{WD-U}||Z_j|R_4)$, $M_6 = Enc_{(R_3||R^1_U||X_i)}(R_4||TID^{new}_i||SK_{WD-U}||TID^*_j)$to transmit the secret parameters securely, and $M_7 = h(TID_U||SK_{CS-U}||SK_{WD-U}||R^1_U||R_4||T_4)$ to verify the authorized entity. Finally, CS sends $Msg_4 = \{M_6, M_7, T_4\}$ to MD_i.

LAP-5: MD_i verifies the freshness of $|T_4 - T_3| \leq \Delta T_i$. If it matches, MD_i decrypts $(R_4||TID^{new}_i||SK_{WD-U}||TID^*_j) = Dec_{(R_3||R^1_U||X_i)}(M_6)$ and computes $SK^*_{CS-U} = h$ $(TID_U||X_i||R_3||R_4||T_4)$, and $M^*_7 = h(TID_U||SK^*_{CS-U}||SK_{WD-U}||R^1_U||R_4||T_4)$, and verifies $M^*_7 \stackrel{?}{=} M_7$. If it is not equal, MD_i terminates the current session; otherwise, MD_i updates a new temporary identity TID^{old}_i to TID^{new}_i and stores the session keys SK^*_{CS-U} and SK^*_{WD-U}. After that, MD_i generates a timestamp T_5 and computes $M_8 = Enc_{(R_1||X_{UW})}(R_3||R_4)$ to transmit the secret parameters securely, and $M_9 = h(TID_{WD}||R_3||R_4||SK_{WD-U}||T_5)$ to verify the authorized entity, and then transmits $Msg_5 = \{M_8, M_9, TID^*_j, T_4, T_5\}$ to WD_j over a public channel.

LAP-6: WD_j checks the freshness of $|T_5 - T_4| \leq \Delta T_i$. If it matches, WD_j computes $(R_3||R_4) = Dec_{(R_1||X_{UW})}(M_8)$, $SK^*_{WD-U} = h(TID_U||PID_{WD}||R_2||R_4||T_4)$, and $M^*_9 =$ $h(TID_{WD}||R_3||R_4||SK^*_{WD-U}||T_5)$, and verifies $M^*_9 \stackrel{?}{=} M_9$. If it matches, WD_j authenticates U_i, successfully and then calculates $TID^{new}_j = TID^*_j \oplus h(TID_{WD}||$ $SK_{WD-U}||Z_j||R_4)$. Finally, WD_j updates a new temporary identity TID^{old}_j to TID^{new}_j and stores a session key SK^*_{WD-U}.

6.4. Group–Proof Generation and Verification Phases

After the authentication process is executed successfully, U_i generates a group proof for multiple WD_j by MD_i, indicating that WD_j belongs to the same U_i and then sends the group proof to CS for verification. This phase is illustrated in Figure 5.

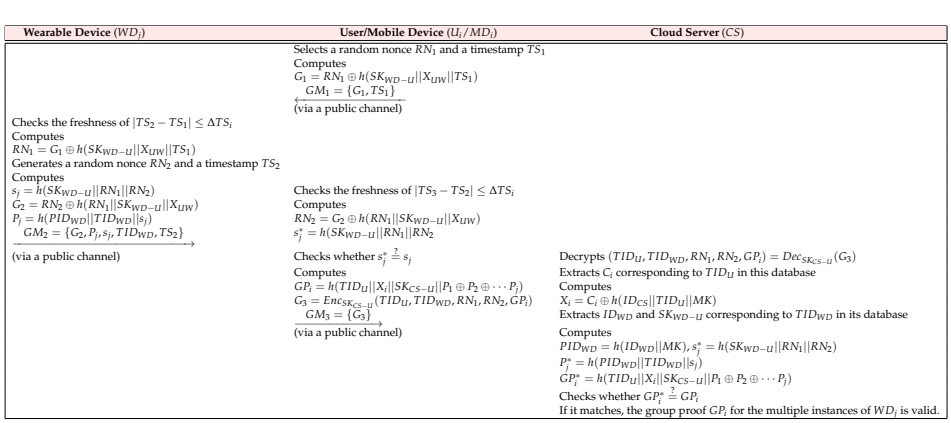

Figure 5. Group–proof generation and verification phase of the AGPS-PUF.

GPGV 1: MD_i for authorized U_i selects RN_1 and TS_1. After that, MD_i computes $G_1 = RN_1 \oplus h(SK_{WD-U}||X_{UW}||TS_1)$ and then sends $GM_1 = \{G_1, TS_1\}$ to WD_j over a public channel.

GPGV 2: WD_j verifies the freshness of $|TS_2 - TS_1| \leq \Delta TS_i$. If it matches, WD_j calculates $RN_1 = G_1 \oplus h(SK_{WD-U}||X_{UW}||TS_1)$ and generates a random nonce RN_2 and a timestamp TS_2. After that, WD_j computes $s_j = h(SK_{WD-U}||RN_1||RN_2)$, $G_2 = RN_2 \oplus h(RN_1||SK_{WD-U}||X_{UW})$, and $P_j = h(PID_{WD}||TID_{WD}||s_j)$, and then transmits $GM_2 = \{G_2, P_j, s_j, TID_{WD}, TS_2\}$ to MD_i.

GPGV 3: MD_i checks the freshness of $|TS_3 - TS_2| \leq \Delta TS_i$. If it matches, MD_i computes $RN_2 = G_2 \oplus h(RN_1||SK_{WD-U}||X_{UW})$ and $s_j^* = h(SK_{WD-U}||RN_1||RN_2$, and checks whether $s_j^* \stackrel{?}{=} s_j$. If it matches, MD_i generates the group proof $GP_i = h(TID_U||X_i||SK_{CS-U}||P_1 \oplus P_2 \oplus \cdots P_j)$ for all wearable devices. Finally, WD_j encrypts $G_3 = Enc_{SK_{CS-U}}(TID_U, TID_{WD}, RN_1, RN_2, GP_i)$ using a session key SK_{CS-U} and then transmits $GM_3 = \{G_3\}$ to CS over an open channel.

GPGV 4: CS decrypts $(TID_U, TID_{WD}, RN_1, RN_2, GP_i) = Dec_{SK_{CS-U}}(G_3)$ using a session key SK_{CS-U}. CS extracts C_i corresponding to TID_U in this database, computes $X_i = C_i \oplus h(ID_{CS}||TID_U||MK)$, and then extracts ID_{WD} and SK_{WD-U}, corresponding to TID_{WD} in its database. After that, CS computes $PID_{WD} = h(ID_{WD}||MK)$, $s_j^* = h(SK_{WD-U}||RN_1||RN_2)$, $P_j^* = h(PID_{WD}||TID_{WD}||s_j)$, and $GP_i^* = h(TID_U||X_i||SK_{CS-U}||P_1 \oplus P_2 \oplus \cdots P_j)$ and then checks whether $GP_i^* \stackrel{?}{=} GP_i$. If it matches, CS successfully verifies that the multiple instances of WD_j belong to the same U_i through the group proof.

6.5. Password Update Phase

If U_i wishes to obtain a new PW_i, U_i can freely update their old PW_i without interacting with RC.

PUP-1: U_i inputs ID_U and an old password PW_i^{old} in MD_i.

PUP-2: MD_i chooses a set of (C_U^x, R_U^x), and computes $a_i = D_i \oplus h(ID_U||PW_i||R_U^x)$, $HID_U = h(ID_U||a_i)$, $HPW_i = h(PW_i||a_i||R_U^x)$, $(X_i||X_{UW}) = A_i \oplus h(HID_U \oplus h(HPW_i||R_U^x)$, and $B_i^* = h(HID_U||TID_U||X_i)$, and verifies whether $B_i^* \stackrel{?}{=} B_i$. If the condition is met, U_i is prompted to choose a new password.

PUP-3: U_i inputs a new PW_i^{new} and computes $HPW_i^{new} = h(PW_i^{new}||a_i||R_U^x)$, $A_i^{new} = (X_i||X_{UW}) \oplus h(HID_U \oplus h(HPW_i^{new}||R_U^x))$, $B_i^{new} = h(HPW_i^{new}||TID_U||X_i)$, $D_i^{new} = a_i \oplus h(ID_U||PW_i^{new}||R_U^x)$. Finally, MD_i replaces $\{A_i, B_i, D_i\}$ with $\{A_i^{new}, B_i^{new}, D_i^{new}\}$. As a result, MD_i contains $\{A_i^{new}, B_i, D_i^{new}, TID_U\}$.

7. Security Analysis

The following introduces the informal/formal security analyses.

7.1. Informal Security Analysis

We demonstrate that the AGPS-PUF can prevent"lethal security attacks" and allow "anonymity, untraceability, and mutual authentication".

7.1.1. Impersonation Attack

This attack means that $\mathcal{A}$ attempts to impersonate the legitimate user by eavesdropping on the exchanged data over an open channel. In this case, $\mathcal{A}$ should generate the authentication messages $\{Msg_1, Msg_2, Msg_3\}$, and $\{Msg_4, Msg_5\}$. However, it is difficult to generate the sensitive messages since $\mathcal{A}$ cannot obtain the "random nonces $\{R_1, R_3\}$" and "secret credential $\{X_i, X_{UW}\}$". Consequently, the AGPS-PUF is resistant to impersonation attacks because $\mathcal{A}$ cannot successfully generate the sensitive messages of the legitimate user.

7.1.2. MITM Attack

According to Section 3.1, $\mathcal{A}$ can inject, modify, eavesdrop, intercept, delete, and block the exchanged messages, $\{Msg_1, Msg_2, Msg_3, Msg_4, Msg_5\}$, in the bidirectional communication between U_i, WD_j, and CS, and then attempt to obtain sensitive information from legitimate entities. However, $\mathcal{A}$ cannot generate sensitive messages since all messages are masked with the PUF responses, $\{R_U^1, R_{WD}^1\}$, and fresh random nonces $\{R_1, R_2, R_3\}$, by using "XOR" and "hash" functions. Hence, the AGPS-PUF is secure against MITM attacks since $\mathcal{A}$ cannot obtain sensitive information from legitimate entities.

7.1.3. Session Key Disclosure Attack

Based on the information presented in Section 3.1, $\mathcal{A}$ can steal the MD and then extract the secret information $\{A_i, B_i, D_i, TID_U\}$ stored in the memory. In the AGPS-PUF, $\mathcal{A}$ should obtain the real identities $\{ID_U, ID_{WD}, ID_{CS}\}$ and random nonces $\{R_2, R_3, R_4\}$ to calculate the session keys, $SK_{CS-U} = h(TID_U||X_i||R_3||R_4||T_4)$ and $SK_{WD-U} = h(TID_U||PID_{WD}||R_2||R_4||T_4)$. However, it is impossible for $\mathcal{A}$ to obtain the common session keys, SK_{CS-U} and SK_{WD-U}, since the random nonces and real identities are preserved with secret parameters $\{X_i, X_{UW}, Z_j\}$, and the PUF parameters $\{R_U^1, R_{WD}^1\}$, using cryptographic primitives. Hence, the AGPS-PUF resists session key disclosure attacks.

7.1.4. Replay Attack

$\mathcal{A}$ eavesdrops on the transmitted messages $\{Msg_1, Msg_2, Msg_3, Msg_4, Msg_5\}$ during the AKA phase and then attempts to authenticate with other parties by transmitting the intercepted data in the previous session. A solution to prevent replay attacks, such as the existing schemes [37,38], is to add random nonces and timestamps to the information exchanged so that the data are unique for each authentication phase. Thus, the AGPS-PUF verifies the freshness of T_i. Moreover, the data are masked with $\{R_1, R_2, R_3, R_4\}$. Therefore, even if $\mathcal{A}$ selects and sends valid authentication messages to legitimate entities, the AGPS-PUF is secure against replay attacks since the current timestamp freshness is incorrect.

7.1.5. Physical Wearable Device Capture Attack

Assume that WD_js are physically captured by $\mathcal{A}$ and then extract $\{E_j, O_j\}$ in WD_j's memory, where $E_j = (X_{UW}||Z_j||PID_{WD}) \oplus h(ID_{WD}||TID_{WD}||R_{WD}^x||b_j)$ and $O_j = b_j \oplus h(R_{WD}^x \oplus TID_{WD} \oplus ID_{WD})$. However, $\mathcal{A}$ does not successfully compute SK_{WD-U}^* between U_i and WD_j without the knowledge of $\{R_2, R_4\}$ and the secret credentials $\{X_{UW}\}$. In addition, the PUF pairs $\{(C_{WD}^1, R_{WD}^1)\}$ are distinct, independent, and secure for all batched WD_j. Hence, the AGPS-PUF is resilient against physical wearable device capture attacks since the PUF output depends on the inherent physical fluctuations of the IC chip.

7.1.6. Stolen Verifier Attack

In this attack, $\mathcal{A}$ extracts and learns the secret parameters related to U_i and WD_j, which are stored in the database of CS, and it then attempts to masquerade as a legitimate entity. However, even if $\mathcal{A}$ obtains the stored parameters $\{TID_U, (C_U^x, R_U^x), C_i\}$ for U_i and $\{Y_j, (C_{WD}^x, R_{WD}^x)\}$ for WD_j, $\mathcal{A}$ cannot calculate the common session keys $\{SK_{WD-U}, SK_{CS-U}\}$, and impersonate a legitimate entity. Unfortunately, $\mathcal{A}$ does not obtain the secret credentials $\{C_i, Y_j\}$ that are masked with RC's master secret key MK by performing the cryptographic primitives. Furthermore, PUF pairs (C_U^x, R_U^x), and (C_{WD}^x, R_{WD}^x) for U_i and WD_j are computationally infeasible for $\mathcal{A}$ to derive the fresh PUF because the PUF output depends on the inherent physical fluctuations of the IC. Hence, the AGPS-PUF is resistant to stolen verifier attacks.

7.1.7. Offline Password-Guessing Attack

Referring to the information presented in Section 3.1, we assume that $\mathcal{A}$ can intercept the transmitted information and then extract the secret credentials stored in the MD. Then, $\mathcal{A}$ attempts to use these attacks to guess U_i's real PW_i. However, PW_i is composed as

$MPW_i = h(PW_i||a_i||R_U^x)$. Therefore, it is impossible for $\mathcal{A}$ to correctly guess PW_i without knowledge of the random number a_i and the PUF response value R_U^x. As a result, the offline password-guessing attack is not feasible in the AGPS-PUF.

7.1.8. Desynchronization Attack

In the AGPS-PUF, the temporary identities, TID_U and TID_{WD}, are assigned to U_i and WD_j during the AKA phase and then tables are maintained from CS. Since both the old temporary identities, i.e., TID_i^{old} and TID_j^{old}, are stored, if the last acknowledgment messages are blocked or lost due to time delay, there will always be consistent temporary identities between U_i, WD_j, and CS. Thus, the AGPS-PUF is resistant to desynchronization attacks.

7.1.9. Privileged Insider Attack

In this attack, $\mathcal{A}$ is a privileged insider of the proposed system. Hence, we assume that $\mathcal{A}$ is able to obtain the request message $\{HPW_i, HPW_i, (C_U^x, R_U^x)\}$ from the remote user U_i. However, the secret credentials, $\{X_i, Z_j, X_{UW}\}$ of M_i, and WD_j, are computationally infeasible for $\mathcal{A}$ without knowledge of the master private key MK and identity ID_{CS}. Thus, the AGPS-PUF can prevent privileged insider attacks because $\mathcal{A}$ cannot correctly generate the sensitive information of U_i and WD_j.

7.1.10. Mutual Authentication

In the AGPS-PUF, all participants successfully perform secure mutual authentication. After obtaining the authentication request messages, $\{M_3, M_5\}$, CS_j check whether $M_5^* \stackrel{?}{=} h(TID_U||TID_{WD}||R_3||R_U^1||T_3)$ to verify the authenticity and integrity of the received message. If it matches, CS is authenticated with U_i. CS then verifies whether $M_3^* \stackrel{?}{=} h(R_1||R_2||R_{WD}^1||T_2)$ to verify the authenticity and integrity of the received message. If it matches, CS is authenticated with WD_j. Upon receiving the authentication message, $\{M_7\}$, U_i checks $M_7^* \stackrel{?}{=} h(TID_U||SK_{CS-U}^*||SK_{WD-U}||R_U^1||R_4||T_4)$ to verify the authenticity and integrity of the received message. If it matches, U_i authenticates CS. After receiving the authentication confirmation message, $\{M_9\}$, WD_j verifies $M_9^* \stackrel{?}{=} h(TID_{WD}||R_3||R_4||SK_{WD-U}^*||T_5)$ to verify the authenticity and integrity of the received message. If it is valid, WD_j authenticates U_i. Thus, the AGPS-PUF successfully allows secure mutual authentication and integrity between U_i, WD_j, and CS.

7.1.11. Anonymity and Untraceability

Assume that $\mathcal{A}$ intercepts the transmitted messages during the AKA phase. However, it is impossible for $\mathcal{A}$ to obtain U_i's identity ID_U and pseudo-identity HID_U and WD_j's identity ID_{WD} and pseudo-identity PID_{WD} without knowledge, such as random nonces, the PUF secret value, and secret credentials. Hence, the AGPS-PUF provides anonymity for U_i and WD_j. Furthermore, $\mathcal{A}$ cannot track the legitimate U_i since all messages are unique and dynamic using timestamps, random nonces, and temporary identities in each session. Moreover, the temporary identities, TID_U and TID_{WD} of U_i and WD_j, are updated as TID_i^{new} and TID_j^{new} in each session. Hence, 3P-AGPS guarantees untraceability for U_i and WD_j.

7.1.12. Perfect Forward Secrecy (PFS)

The PFS security indicates that SK will not be exposed to $\mathcal{A}$ even if a long-term secret key is compromised. In the AGPS-PUF, if CS's long-term secret key MK is compromised, $\mathcal{A}$ cannot compute the session keys, SK_{CS-U} and SK_{WD-U}, because $\mathcal{A}$ does not have knowledge of the secret credentials $\{X_{UW}, X_i\}$, the PUF secret value $\{R_U^x, R_{WD}^x\}$, and real identities $\{ID_U, ID_{WD}\}$. Consequently, the AGPS-PUF is resistant to PFS.

7.2. Formal Analysis through ROR Oracle Model

We utilize a formal proof, denoted as the ROR Oracle model, to prove the session key (SK) security. We define the queries required for the ROR Oracle model [10].

In the AGPS-PUF, there are three participants: the mobile user Γ_U, the wearable device Γ_{WD}, and the cloud server Γ_{CS}. Let $\Gamma_U^{t_1}$ be the instance t_1 of a participant U, $\Gamma_{WD}^{t_2}$ be the instance t_2 of a participant WD, and $\Gamma_{CS}^{t_3}$ be the instance t_3 of a participant CS. In Table 2, we present the descriptions for each query, including "$CorruptMD(\cdot)$,$Execute(\cdot)$, $Test(\cdot)$, $Send(\cdot)$, and $Reveal(\cdot)$ for ROR Oracle model".

Table 2. Query and purpose.

Query	Purpose
$Send(\Gamma^t, Msg)$	Based on this query, $\mathcal{A}$ can transmit the message Msg to the Γ^t, and obtain the response message accordingly.
$CorruptMD(\Gamma_U^{t_1})$	This query indicates as the mobile device stolen attacks, where $\mathcal{A}$ can extract the secret credentials stored in MD.
$CorruptWD(\Gamma_{WD}^{t_3})$	This query indicates as the physical capture attacks where $\mathcal{A}$ can obtain the secret parameters stored in WD.
$Execute(\Gamma_U^{t_1}, \Gamma_{WD}^{t_2}, \Gamma_{CS}^{t_3},)$	Based on $Execute(\cdot)$, $\mathcal{A}$ performs the passive/active attacks by eavesdropping the exchanged messages between each entity over a insecure channel.
$Reveal(\Gamma^t)$	Based on this query, $\mathcal{A}$ reveals a SK generated between $\Gamma_U^{t_1}$ and $\Gamma_{WD}^{t_2}$ using $Reveal(\cdot)$ query.
$Test(\Gamma^t)$	An unbiased coin c is tossed prior to game start. If $\mathcal{A}$ gets $c = 1$ under the $Test(\cdot)$, it indicates a SK between $\Gamma_U^{t_1}$ and $\Gamma_{WD}^{t_2}$ is fresh. If $\mathcal{A}$ gets the $c = 0$, it indicates SK is not fresh; otherwise, $\mathcal{A}$ gets a null value ($\perp$).

Theorem 1. *Let* $Adv_{\mathcal{A}}^{AGPS-PUF}$ *be the advantage that* $Adv_{\mathcal{A}}^{AGPS-PUF}$ *is able to break the SK security of the AGPS-PUF. Hence, we derive the following* $Adv_{\mathcal{A}}^{AGPS-PUF} \leq \frac{q_h^2}{|Hash|} + \frac{q_P^2}{|PUF|} + 2\{C \cdot q_{send}^s, \frac{q_s}{2^{l_1}}, \frac{q_s}{2^{l_2}}\}$

Proof. The $PUF(\cdot)$ range space, $h(\cdot)$ query number, $Send(\cdot)$ query number, and $Hash$ range space indicate q_P, q_h, q_{send}, and $Hash$. Furthermore, the Zipf credentials [39] indicate C, l_m, s, and l_n.

Proof: We present the five games GM_i ($i \in [0, 4]$). We indicate that $Adv_{\mathcal{A},GM_i}^{AGPS-PUF}$ is the probability of $\mathcal{A}$ to win GM_i. All games are described in detail as follows.

Game: GM_0: $\mathcal{A}$ executes a real attack in AGPS-PUF. Hence, $\mathcal{A}$ picks a random bit c at the beginning of GM_0. We obtain the following Equation (1) as

$$Adv_{\mathcal{A}}^{AGPS-PUF} = |2 \cdot Adv_{\mathcal{A},GM_0}^{AGPS-PUF} - 1| \tag{1}$$

Game GM_1: GM_1 indicates that $\mathcal{A}$ executes an "eavesdropping attack, in which the transmitted messages are intercepted between U, WD, and CS performing $Execute(\cdot)$ query". In GM_1, $\mathcal{A}$ carries out "$Test(\cdot)/Reveal(\cdot)$ queries" to compromise SK. The results of the $Test(\cdot)/Reveal(\cdot)$ queries determine whether $\mathcal{A}$ obtains SK_{CS-U} and SK_{WD-U}. To compromise SK, $\mathcal{A}$ requires the random nonces $\{R_2, R_3, R_4\}$, and PUF values. Therefore, $\mathcal{A}$ is not able to increase the winning probability of GM_1. We can derive Equation (2) as

$$Adv_{\mathcal{A},GM_1}^{AGPS-PUF} = Adv_{\mathcal{A},GM_0}^{AGPS-PUF} \tag{2}$$

Game GM_2: This game indicates that $\mathcal{A}$ executes a "real attack" based on "$Send(\cdot)$ and $Hash$" queries. $\mathcal{A}$ transmits the modified messages to participants and acts as a legal user so that it is able to guess the outcomes of the "$Send(\cdot)$ query". Moreover, $\mathcal{A}$ aims to

find collisions for the hash oracle and attempts to copy messages that are expected to be authenticated by the entities. Because the random nonce, timestamp, temporary secret, and identity are configured using hash functions in each message, running "$Send(\cdot)$ and $Hash$ queries" cannot cause a conflict. We can deduce that the probability of aborting the game is bounded by $\frac{q_h^2}{2|Hash|}$. It is worth noting that this may happen when processing $Send()$ query; the game is aborted with a probability determined by the birthday paradox [40]. The probability of finding collisions in the hash oracle $Hash$, as per the square of the birthday paradox, is the probability, and the two games, GM_1 and GM_2, are indistinguishable, unless one of the above rules causes the game to abort. Thus, we can have Equation (3) as

$$|Adv_{\mathcal{A},GM_2}^{AGPS-PUF} - Adv_{\mathcal{A},GM_1}^{AGPS-PUF}| \leq \frac{q_h^2}{2|Hash|} \tag{3}$$

Game GM_3: This game is executed in the analogy as presented in GM_2. By using the "analogous argument" described in GM_2, we can derive Equation (4) as

$$|Adv_{\mathcal{A},GM_3}^{AGPS-PUF} - Adv_{\mathcal{A},GM_2}^{AGPS-PUF}| \leq \frac{q_P^2}{2|PUF|} \tag{4}$$

Game GM_4: In this game, $\mathcal{A}$ attempts to extract $\{A_i, B_i, D_i, TID_U\}$ in the MD's memory by using the "differential power analysis" with $CourruptWD(\cdot)$ and $CourruptMD(\cdot)$ queries. Note that $A_i = (X_i||X_{UW}) \oplus h(HID_U \oplus h(HPW_i||R_U^x))$, $B_i = h(HPW_i||TID_U||X_i)$, $D_i = a_i \oplus h(ID_U||PW_i||R_U^x)$, and TID_U. Moreover, $\mathcal{A}$ can obtain the secret credentials $\{E_j, O_j\}$ in the WD's memory using physical capture attacks. Note that, $E_j = (X_{UW}||Z_j||PID_{WD}) \oplus h(ID_{WD}||TID_{WD}||R_{WD}^x||b_j)$ and $O_j = b_j \oplus h(R_{WD}^x \oplus TID_{WD} \oplus ID_{WD})$. However, GM_3 is computationally infeasible for $\mathcal{A}$ to compromise the PW_i of the legitimate U_i over the $Send(\cdot)$ query without a_i and R_U^x. Moreover, $\mathcal{A}$ should guess the parameters from the extracted data because $\mathcal{A}$ does not have knowledge of the "password", "biometric", and "PUF secret". Moreover, it is computationally impossible to guess the "biometric", "password", and "PUF secret". In conclusion, GM_3 and GM_4 are "indistinguishable". We obtain Equation (5) as follows:

$$|Adv_{\mathcal{A},GM_4}^{AGPS-PUF} - Adv_{\mathcal{A},GM_3}^{AGPS-PUF}| \leq \{C \cdot q_{send}^s, \frac{q_s}{2^{l_b}}\} \tag{5}$$

Based on the execution of $GM_0 - GM_4$, $\mathcal{A}$ attempts to guess the "bit c to win the games by performing $Test(\cdot)$ query". We can obtain Equation (6) as follows:

$$Adv_{\mathcal{A},GM_4}^{AGPS-PUF} = \frac{1}{2} \tag{6}$$

Based on the "Formulas (1), (2), and (6)", we obtain Equation (7) as follows:

$$\begin{aligned} \frac{1}{2} Adv_{\mathcal{A}}^{AGPS-PUF} &= |Adv_{\mathcal{A},GM_0}^{AGPS-PUF} - \frac{1}{2}| \\ &= |Adv_{\mathcal{A},GM_1}^{AGPS-PUF} - \frac{1}{2}| \\ &= |Adv_{\mathcal{A},GM_1}^{AGPS-PUF} - Adv_{\mathcal{A},GM_4}^{AGPS-PUF}| \end{aligned} \tag{7}$$

Based on the "triangular inequality with the Formulas (3), (4), (5) and (7)", we obtain Equation (8) as follows:

$$\begin{aligned}\frac{1}{2}Adv_{\mathcal{A}}^{AGPS-PUF} &= |Adv_{\mathcal{A},GM_1}^{AGPS-PUF} - Adv_{\mathcal{A},GM_4}^{AGPS-PUF}| \\ &\leq |Adv_{\mathcal{A},GM_1}^{AGPS-PUF} - Adv_{\mathcal{A},GM_3}^{AGPS-PUF}| \\ &\quad + |Adv_{\mathcal{A},GM_3}^{AGPS-PUF} - Adv_{\mathcal{A},GM_4}^{AGPS-PUF}| \\ &\leq |Adv_{\mathcal{A},GM_1}^{AGPS-PUF} - Adv_{\mathcal{A},GM_2}^{AGPS-PUF}| \\ &\quad + |Adv_{\mathcal{A},GM_2}^{AGPS-PUF} - Adv_{\mathcal{A},GM_3}^{AGPS-PUF}| \\ &\quad + |Adv_{\mathcal{A},GM_3}^{AGPS-PUF} - Adv_{\mathcal{A},GM_4}^{AGPS-PUF}| \\ &\leq \frac{q_h^2}{2|Hash|} + \frac{q_P^2}{2|PUF|} + \{C \cdot q_{send}^s, \frac{q_s}{2^{l_1}}, \frac{q_s}{2^{l_2}}\}. \end{aligned} \tag{8}$$

Finally, by multiplying both sides of Equation (8) by a factor of 2, we can obtain the following: $Adv_{\mathcal{A}}^{AGPS-PUF} \leq \frac{q_h^2}{|Hash|} + \frac{q_P^2}{|PUF|} + 2\{C \cdot q_{send}^s, \frac{q_s}{2^{l_1}}, \frac{q_s}{2^{l_2}}\}$ □

7.3. Formal Analysis through AVISPA Simulation

This simulation proves the formal security robustness of the cryptographic protocol against MITM and replay attacks. We implement the security simulation and demonstrate the security result. We first need to implement the AGPS-PUF as a programming language HLPSL [41]. After that, this simulation starts analyzing the intermediate format (IF) over the four backends: "On-the-Fly Model Checker (OFMC)", "Constraint Logic-based Attack Searcher (CL-AtSe)", "SAT-based Model-Checker (SATMC)", and "Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP)". Since TA4SP and SATMC backends do not implement XOR operations, the simulation results of the AGPS-PUF under these backends become inconclusive; thus, the results based on TA4SP and SATMC backends have been ignored.

We simulated the AGPS-PUF using the "Security Protocol ANimator (SPAN) [**?**]" for AVISPA. It is worth noting that AVISPA implements the DY model and that an intruder participates in the protocol execution with a concrete session. The specification roles of the WD, U, and CS are implemented using HLPSL, such as sessions, security goals, and environments. In Figure 6, the HLPSL specification of the protocol is converted into the IF by using the HLPSL2IF translator. After that, the IF is converted to the output format (OF) by feeding it to one of the four backends. The OF contains the following:

- SUMMARY: It refers to whether the tested security protocol is safe or unsafe, or whether the analysis is inconclusive.
- DETAILS: It explains why the analysis is inconclusive, why the tested security protocol is safe, or under what conditions the test applications or security protocols may be exploitable to the attack.
- PROTOCOL: It refers to the HLPSL specification of the target security protocol in the IF.
- GOAL: It demonstrates the goal of the analysis, which is performed by AVISPA using HLPSL specifications.
- BACKEND: It is the name of the backend that is utilized for the analysis of SATMC, CL-AtSe, OFMC, or TA4SP.
- STATISTICS: It includes the trace of any potential vulnerability in the target security protocol, along with several useful statistics and related comments.

In the simulation based on AVISPA backends, two verifications were performed: (1) checking for replay attacks and (2) DY model-based MITM attacks. When checking for replay attacks on the AGPS-PUF, both OFMC and CL-AtSe check if the legitimate participants can execute the specified protocols by performing a search for a passive intruder. Moreover, both OFMC and CL-AtSe backends are used to check whether any MITM attacks are possible by an intruder in the DY model. The SPAN simulation results

demonstrate the security attacks and intruder simulations over a web-based GUI (graphical user interface). Moreover, the implementation results obtained using the CL-AtSe and OFMC backends are presented in Figure 7. According to the simulation results under the OFMC and CL-AtSe in Figure 7, the SAFE output shows that the AGPS-PUF is safe based on the specified security goals. Consequently, we demonstrate that the AGPS-PUF is protected from replay and MITM attacks.

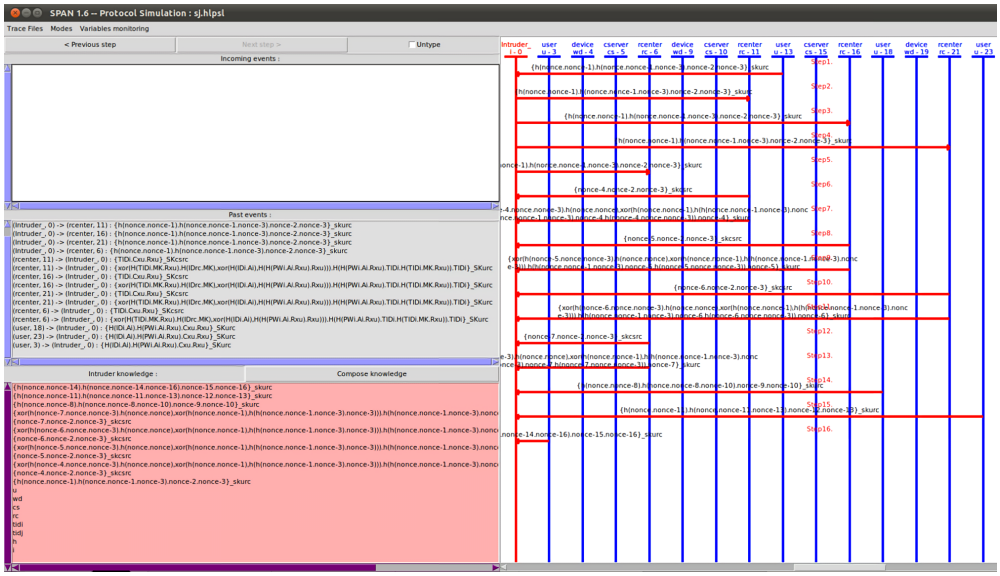

Figure 6. AVISPA simulation results based on SPAN.

SUMMARY **SAFE** DETAILS BOUNDED_NUMBER_OF_SESSIONS TYPED_MODEL PROTOCOL /home/span/span/test_suite/results/sj.if GOAL As_specified BACK_END OFMC STATISTICS ParseTime: 0.01 s SearchTime: 1.12 s VisitedNodes: 1432 nodes Depth: 8 plies	SUMMARY **SAFE** DETAILS BOUNDED_NUMBER_OF_SESSIONS TYPED_MODEL PROTOCOL /home/span/span/test_suite/results/sj.if GOAL As_specified BACK_END CL-AtSe STATISTICS Analysed : 13 states Reachable : 13 states Translation: 0.81 seconds Computation: 0.02 seconds

Figure 7. AVISPA results based on OFMC and CL-AtSe.

8. Testbed Experiments Using MIRACL

We present the testbed experiments to estimate the execution times required for essential cryptographic operations utilized in the AGPS-PUF and existing related schemes. We used the well-known "MIRACL crypto SDK [42]", which is a C/C++-based programming software library.

We used the two platforms to estimate the execution times required for cryptographic operations. T_{sed}, T_{ecpm}, T_{me}, and T_h evaluate the execution times required for "a AES encryption and decryption", "an ECC scalar point multiplication", "a modular exponentiation", and "a SHA-256 hash function".

- **Platform 1:** This platform is used to calculate the execution times for the *MD* and *WD* settings on MIRACL, as follows: "Model: Raspberry PI 4B, with "OS: Ubuntu 20.04.2 LTS", "Processor: 1.5 GHz Quad-core", "CPU: 64-bit". Each operation was run

1000 times on the same setup and we observed the average, maximum, and minimum times. The results of this platform are tabulated in Table 3.

Table 3. Execution times (in milliseconds) based on the MIRACL library, obtained using a Raspberry Pi 4.

Operation	Min. Time (ms)	Max. Time (ms)	Average Time (ms)
T_{ecpm}	2.766	2.920	2.848
T_h	0.274	0.643	0.309
T_{me}	0.178	0.493	0.228
T_{sed}	0.011	0.021	0.012

- **Platform 2:** This platform was used to calculate the execution time for the *CS* server setting as follows: "OS: Ubuntu 18.04.4 LTS, Processor: Intel Core i5-10400 @2.9 GHz, Six-core, CPU: 64-bits". All primitives were run 1000 times on the same setup and we observed the average, maximum, and minimum times. The results of this platform are tabulated in Table 4.

Table 4. Execution times (in milliseconds) based on the MIRACL library for a server.

Operation	Min. Time (ms)	Max. Time (ms)	Average Time (ms)
T_{ecpm}	0.472	2.737	0.522
T_h	0.024	0.149	0.055
T_{me}	0.022	0.082	0.040
T_{sed}	0.001	0.002	0.001

9. Performance Comparison

This section presents the "performance comparison analysis" of the AGPS-PUF and existing related schemes for wearable computing [8,21,23,25,26,28].

9.1. Computation Costs

We discuss the comparative computation costs of the AGPS-PUF with the existing related schemes [8,21,23,25,26,28] during the AKA phase. We used the" testbed experimental results for the Raspberry PI 4 and server setting in Section 8". With the information presented in Table 3, we utilized the analysis results of the average time for each operation under U_i and WD_j.

We calculated the execution times for the *MD* and *WD* settings on MIRACL as follows: "Model: Raspberry PI 4B, with "OS: Ubuntu 20.04.2 LTS", "Processor: 1.5 GHz Quad-core", "CPU: 64-bit". As seen in Table 3, we present "$T_{ecpm} \approx 2.848$ ms, $T_h \approx 0.309$ ms, $T_{me} \approx 0.228$ ms and $T_{sed} \approx 0.012$ ms". Moreover, we calculated the execution times for the *CS* server setting as follows: "OS: Ubuntu 18.04.4 LTS, Processor: Intel Core i5-10400 @2.9 GHz, Six-core, CPU: 64-bits". As seen in Table 4, we utilized the analysis results for the average time of each operation under *CS*. In scenario 2, we present "$T_{ecpm} \approx 0.522$ ms, $T_h \approx 0.055$ ms, $T_{me} \approx 0.040$ ms and $T_{sed} \approx 0.001$ ms". We prove the performance results for the comparative computational costs in Table 5 and Figure 8. Consequently, the AGPS-PUF offers the necessary security requirements and features while maintaining similar costs compared to previous schemes [8,25,26,28]. Hence, the AGPS-PUF is suitable for practical wearable computing environments.

Table 5. Comparison between computational costs.

Scheme	User	Gateway/Server	Wearable Device	Total Computation Cost
Li et al. [21]	$6T_h + 2T_{sed}$	$7T_h + 6T_{sed}$	$5T_h + 2T_{sed}$	$18T_h + 10T_{sed}$
Wu et al. [23]	$10T_h + 2T_{sed}$	$6T_h + 5T_{sed}$	$4T_h + T_{sed}$	$20T_h + 8T_{sed}$
Amin et al. [25]	$12T_h$	$18T_h$	$6T_h$	$36T_h$
Ali et al. [26]	$12T_h + 2T_{sed}$	$16T_h + 3T_{sed}$	$7T_h + 5_{sed}$	$35T_h + 10T_{sed}$
Hajian et al. [28]	$13T_h$	$7T_h$	$9T_h$	$29T_h$
Guo et al. [8]	$21T_h$	$18T_h$	$7T_h$	$46T_h$
AGPS-PUF	$10T_h + 3T_{sed}$	$10T_h + 2T_{sed}$	$8T_h + T_{sed}$	$28T_h + 6T_{sed}$

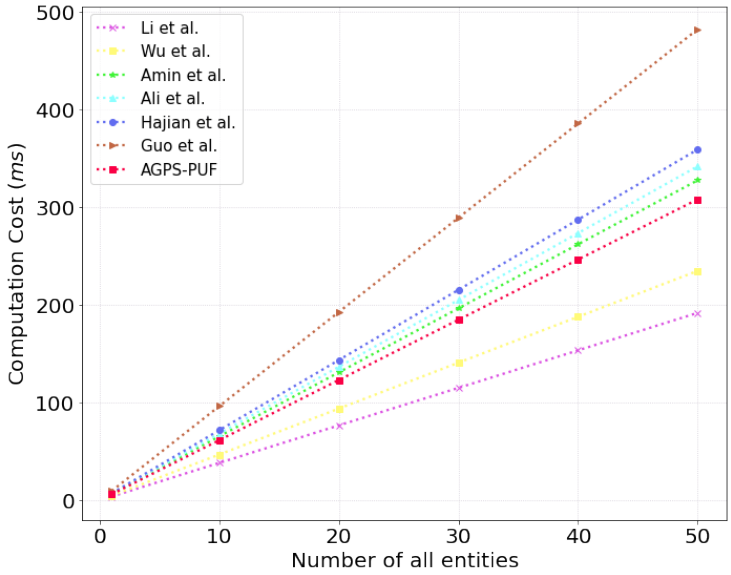

Figure 8. Computational cost comparison of all entities.

9.2. Communication Costs

We discuss the comparative communication costs of the AGPS-PUF and existing related schemes [8,21,23,25,26,28]. Referring to [8], we assume that the bits for the timestamp, the PUF challenge, identity, random nonce, symmetric encryption/decryption, and hash digest are 32, 64, 128, 128, 128, and 256 bits, respectively. During the AKA phase of the AGPS-PUF, the exchanged messages $\{M_1, T_1\}$, $\{M_2, M_3, TID_{WD}, C^1_{WD}, T_2\}$, $\{M_2, M_3, M_4, M_5, TID_U, TID_{WD}, C^1_{WD}, C^1_U, T_2, T_3\}$, $\{M_6, M_7, T_4\}$, and $\{M_8, M_9, TID^*_j, T_4, T_5\}$ require "(256 + 32 = 288 bits), (256 + 256 + 128 + 64 + 32 = 736 bits), (256 + 256 + 128 + 256 + 128 + 128 + 64 + 64 + 32 + 32 = 1344), (128 + 256 + 32 = 416 bits), and (256 + 256 + 256 + 32 + 32 = 832 bits)". Consequently, the AGPS-PUF has similar costs compared with previous schemes, as presented in Table 6 and Figure 9, since transmitting fewer bits minimizes the network latency and number of collisions.

Table 6. Comparison between communication costs.

Scheme	Communication Cost for WD_j	Total Cost	Number of Messages
[21]	2112 bits	4352 bits	4 messages
[23]	2304 bits	2816 bits	3 messages
[25]	1280 bits	4096 bits	5 messages
[26]	1952 bits	4128 bits	4 messages
[28]	1504 bits	3552 bits	5 messages
[8]	1920 bits	5088 bits	5 messages
AGPS-PUF	1568 bits	3616 bits	5 messages

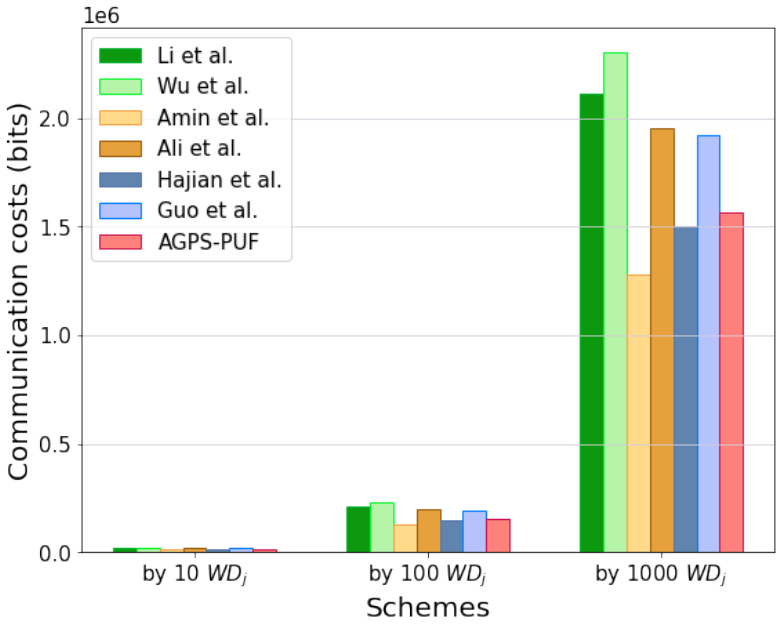

Figure 9. Communication cost comparison.

9.3. Security Functionality Comparison

This section compares the "security functionalities" of the AGPS-PUF with the existing related schemes for wearable computing [8,21,23,25,26,28]. In Table 7, we show that some existing schemes for wearable computing are not fully protected and may be fragile to different potential security attacks. Thus, the security protocols must be designed in such a way that they must be robust against lethal security attacks. In contrast, the AGPS-PUF is resilient to lethal security attacks, and guarantees the necessary security requirements and functionalities, including "mutual authentication, PFS, anonymity, and untraceability". Thus, the AGPS-PUF provides more security functionalities when compared to the existing related schemes for wearable computing [8,21,23,25,26,28].

Table 7. Comparative study on security features.

Security Features	[21]	[23]	[25]	[26]	[28]	[8]	Our
SF_1	o	o	o	o	x	x	o
SF_2	o	x	x	o	o	o	o
SF_3	o	x	x	o	x	x	o
SF_4	x	NA	NA	NA	o	x	o
SF_5	NA	x	x	NA	x	o	o
SF_6	o	o	o	o	o	o	o
SF_7	o	o	x	o	x	x	o
SF_8	o	o	o	o	o	o	o
SF_9	x	x	x	o	o	o	o
SF_{10}	o	o	o	o	x	x	o
SF_{11}	x	x	x	o	o	o	o
SF_{12}	o	o	o	o	o	o	o
SF_{13}	o	x	o	o	o	x	o
SF_{14}	x	o	o	o	o	x	o
SF_{15}	NA	NA	NA	NA	NA	o	o

o: "protection of security features"; x: "non-protection of security features"; NA: "not applicable"; SF_1: "MITM attack"; SF_2: "offline password-guessing attack"; SF_3: "impersonation attack"; SF_4: "sensor physical capture attack"; SF_5: "mobile device stolen attack"; SF_6: "stolen verifier attack"; SF_7: "session key disclosure attack"; SF_8: "replay attack"; SF_9: "privileged insider attack"; SF_{10}: "mutual authentication"; SF_{11}: "user anonymity"; SF_{12}: "PFS"; SF_{13}: "untraceability"; SF_{14}: "formal (simulation) analysis"; SF_{15}: "group proof".

10. Conclusions

We prove that Guo et al.'s scheme is not protected against session key disclosure, MITM, and impersonation attacks, and it does not offer security requirements and features such as mutual authentication and untraceability. Hence, we designed an efficient and robust authentication and group–proof scheme using the PUF for wearable computing to address the security issues of Guo et al.'s scheme. We demonstrate the session key security of the AGPS-PUF by performing formal security under the ROR Oracle model analysis and show that the AGPS-PUF is resistant to replay and MITM attacks by using the AVISPA simulation analysis. Furthermore, we present the testbed experiments of the AGPS-PUF using MIRACL crypto SDK based on Raspberry PI 4. We demonstrate the performance comparison of the AGPS-PUF and the existing related schemes for wearable computing with respect to computation costs, communication costs, and security features. Thus, the AGPS-PUF ensured a higher security level than the existing related scheme in wearable computing environments and provided similar computational and communication costs to the existing related schemes for wearable computing. Thus, the AGPS-PUF is suitable for practical wearable computing environments, as it offers more effective efficiency and superior security compared to existing related schemes for wearable computing.

Author Contributions: Conceptualization, S.Y.; methodology, S.Y.; validation, S.Y.; formal analysis, S.Y.; writing—original draft preparation, S.Y.; writing—review and editing, Y.P.; supervision, Y.P.; project administration, Y.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korean government (MSIT) (no. 2022-0-01019, Development of eSIM security platform technology for edge devices to expand the eSIM ecosystem).

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: Not applicable

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

References

1. Roggen, D.; Magnenat, S.; Waibel, M.; Troster, G. Wearable Computing. *IEEE Robot. Autom. Mag.* **2011**, *18*, 83–95. [CrossRef]
2. Sun, H.; Zhang, Z.; Hu, R.Q.; Qian, Y. Wearable Communications in 5G: Challenges and Enabbling Technologies. *IEEE Veh. Technol. Mag.* **2018**, *13*, 100–109. [CrossRef]
3. Abbas, G.; Tanveer, M.; Abbas, Z.H.; Waqas, M.; Baker, T. A Secure Remote User Authentication Scheme for 6LoWPAN-based Internet of Things. *PLoS ONE* **2021**, *16*, e0258279. [CrossRef] [PubMed]
4. Majumder, S.; Mondal, T.; Deen, M.J. Wearable Sensors for Remote Health Monitoring. *Sensors* **2017**, *17*, 130. [CrossRef] [PubMed]
5. Seneviratne, S.; Hu, Y.; Nguyen, T.; Lan, G.; Khalifa, S.; Thilakarathna, K.; Hassan, M.; Seneviratne, A. A Survey of Wearable Devices and Challenges. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2573–2620. [CrossRef]
6. Wang, S.; Bie, R.; Zhao, F.; Zhang, N.; Cheng, X.; Choi, H.A. Security in Wearable Communications. *IEEE Netw.* **2016**, *30*, 61–67. [CrossRef]
7. Zhang, Y.; Deng, R.H.; Han, G.; Zheng, D. Secure Smart Health with Privacy-aware Aggregate Authentication and Access Control in Internet of Things *J. Netw. Comput. Appl.* **2018**, *123*, 889–100. [CrossRef]
8. Guo, Y.; Zhang, Z.; Guo, Y. Anonymous Authenticated Key Agreement and Group Proof Protocol for Wearable Computing. *IEEE Trans. Mob. Comput.* **2022**, *21*, 2718–2731. [CrossRef]
9. AVISPA. Automated Validation of Internet Security Protocols and Applications. 2001. Available online: http://www.avispa-project.org/ (accessed on 16 March 2021).
10. Abdalla, M.; Fouque, P.A.; Pointcheval, D. Password-based authentication key exchange in the three-party setting, in Public Key Cryptography. In Proceedings of the International Workshop on Public Key Cryptography, Les Diablerets, Switzerland, 23–26 January 2005; Springer: Berlin/Heidelberg, Garmany, 2005; pp. 65–84.
11. Park, K.S.; Noh, S.K.; Lee, H.J.; Das, A.K.; Kim, M.H.; Park, Y.H.; Wazid, M. LAKS-NVT: Provably Secure and Lightweight Authentication and Key Agreement Scheme Without Verification Table in Medical Internet of Things. *IEEE Access* **2020**, *8*, 119387–119404. [CrossRef]

12. Das, A.K.; Zeadally, S.; Wazid, M. Lightweight Authentication Protocols for Wearable Devices. *Comput. Electr. Eng.* **2017**, *63*, 196–208. [CrossRef]
13. Vhaduri, S.; Poellabauer, C. Multi-Modal Biometric-Based Implicit Authentication of Wearable Device Users *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 3116–3125. [CrossRef]
14. Li, M.; Yu, S.; Lou, W.; Ren, K. Group Device Pairing Based Secure Sensor Association and Key Management for Body Area Networks. In Proceedings of the IEEE INFOCOM, San Diego, CA, USA, 14–19 March 2010; pp. 2651–2659.
15. Tan, C.C.; Wang, H.; Zhong, S.; Li, Q. IBE-Lite: A Lightweight Identity-Based Cryptography for Body Sensor Networks. *IEEE Trans. Inf. Technol. Biomed.* **2019**, *13*, 926–932. [CrossRef]
16. 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]
17. Al-Riyami, S.S.; Paterson, K.G. Certificateless Public Key Cryptography. *Lect. Notes Comput. Sci.* **2003**, *294*, 452–473.
18. Liu, W.; Liu, H.; Wan, Y.; Kong, H.; Ning, H. The Yoking-Proof-based Authentication Protocol for Cloud-assisted Wearable Devices *Pers. Ubiquitous Comput.* **2016**, *20*, 469–479. [CrossRef]
19. Das, A.K.; Wazid, M.; Kumar, N.; Khan, M.K.; Choo, K.K.R.; Park, Y.H. Design of Secure and Lightweight Authentication Protocol for Wearable Devices Environment. *IEEE J. Biomed. Health Inform.* **2018**, *22*, 1310–1322. [CrossRef]
20. Liu, H.; Yao, X.; Yang, T.; Ning, H. Cooperative Privacy Preservation for Wearable Devices in Hybrid Computing-Based Smart Health. *IEEE Internet Things J.* **2019**, *6*, 1352–1362. [CrossRef]
21. Li, X.; Niu, J.; Kumari, S.; Liao, J.; Liang, W.; Khan, M.K. A New Authentication Protocol for Healthcare Applications Using Wireless Medical Sensor Networks with User Anonymity. *Secur. Commun. Netw.* **2016**, *9*, 2643–2655. [CrossRef]
22. Das, A.K.; Sutrala, A.K.; Odelu, V.; Goswami, A. A Secure Smartcard-Based Anonymous User Authentication Scheme for Healthcare Applications Using Wireless Medical Sensor Networks. *Wirel. Pers. Commun.* **2017**, *94*, 1899–1933. [CrossRef]
23. Wu, F.; Xu, L.; Kumari, S.; Li, X. An Improved and Anonymous Two-factor Authentication Protocol for Health-care Applications with Wireless Medical Sensor Networks. *Multimed. Syst.* **2017**, *23*, 195–205. [CrossRef]
24. Srinivas, J.; Mishra, D.; Mukhopadhyay, S. A Mutual Authentication Framework for Wireless Medical Sensor Networks. *J. Med. Syst.* **2017**, *41*, 80. [CrossRef] [PubMed]
25. Amin, R.; Islam, S.K.H.; Biswas, G.P.; Khan, M.K.; Kumar, N. A Robust and Anonymous Patient Monitoring System Using Wireless Medical Sensor Networks. *Future Gener. Comput. Syst.* **2018**, *80*, 483–495. [CrossRef]
26. Ali, R.; Pal, A.K.; Kumari, S.; Sangaiah, A.K.; Li, X.; Wu, F. An Enhanced Three Factor Based Authentication Protocol Using Wireless Medical Sensor Networks for Healthcare Monitoring. *J. Ambient. Intell. Humaniz. Comput.* **2018**, *9*, 1–22. [CrossRef]
27. Gupta, A.; Tripathi, M.; Shaikh, T.J.; Sharma, A. A Lightweight Anonymous User Authentication and Key Establishment Scheme for Wearable Devices. *Comput. Netw.*, **2019**, *149*, 29–42. [CrossRef]
28. 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]
29. Yu, S.J.; Park, K.S. SLAS-TMIS: Secure, Anonymous and Lightweight Privacy-Preserving Scheme for IoMT-Enabled TMIS Environments. *IEEE Access* **2022**, *10*, 60534–60549. [CrossRef]
30. Dolev, D.; Yao A.C. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [CrossRef]
31. Yu, S.J.; Park, K.S. ISG-SLAS: Secure and Lightweight Authentication and Key Agreement Scheme for Industrial Smart Grid Using Fuzzy Extractor. *J. Syst. Archit.* **2022**, *131*, 102698. [CrossRef]
32. Kocher, P.; Jaffe, J.; Jun, B. Differential power analysis. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 1999; pp. 388–397.
33. Park, K.S.; Park, Y.H.; Park, Y.H.; Das, A.K. 2PAKEP: Provably Secure and Efficient Two-Party Authenticated Key Exchange Protocol for Mobile Environment. *IEEE Access* **2018**, *6*, 30225–30241. [CrossRef]
34. Yu, S.J.; Park, Y.H. A Robust Authentication Protocol for Wireless Medical Sensor Networks Using Blockchain and Physically Unclonable Functions. *IEEE Internet Things J.* **2022**, *9*, 20214–20228. [CrossRef]
35. Gao, Y.; Sarawi, S.F.A.; Abbott, D. Physical Unclonable Functions. *Nat. Electron.* **2020**, *3*, 81–91. [CrossRef]
36. Frikken, K.B.; Blanton, M.; Atallah, M.J. Robust Authentication Using Physically Unclonable Functions. In Proceedings of the International Conference on Information Security, Pisa, Italy, 7–9 September 2009; pp. 262–277.
37. Badshah, A.; Waqas, M.; Abbas, G.; Muhammad, F.; Abbas, Z.H.; Vimal, S.; Bilal, M. LAKA-BSG: Lightweight Authenticated Key Exchange Scheme for Blockchain-Enabled Smart Grids. *Sustain. Energy Technol. Assessments* **2022**, *52*, 102248. [CrossRef]
38. Tanveer, M.; Alasmary, H. LACP-SG: Lightweight Authentication Protocol for Smart Grids. *Sensors* **2023**, *23*, 2309. [CrossRef] [PubMed]
39. 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]
40. Boyko, V.; Mackenzie, P.; Patel, S. Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. In Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, 14–18 May 2000; Springer: Berlin/Heidelberg, Germany, 2000; pp. 156–171.

41. Oheimb, D.V. The High-Level Protocol Specification Lanuage HLPSL Developed in the EU Project AVISPA. In Proceedings of the APPSEM 2005 Workshop, Tallinn, Finland, 13 September 2005; pp. 1–17.
42. MIRACL. Cryptographic SDK: Multiprecision Integer and Rational Arithmetic Cryptographic Library. 2019. Available online: https://github.com/miracl/MIRACL (accessed on 16 April 2021).