Article

A Secure and Lightweight Three-Factor-Based Authentication Scheme for Smart Healthcare Systems †

Jihyeon Ryu [1], Dongwoo Kang [2], Hakjun Lee [2], Hyoungshick Kim [3] and Dongho Won [3,*]

1 Department of Software, Sungkyunkwan University, 2066 Seobu-ro, Jangan-gu, Suwon-si 16419, Gyeonggi-do, Korea; jhryu@security.re.kr
2 Department of Electrical and Computer Engineering, Sungkyunkwan University, 2066 Seobu-ro, Jangan-gu, Suwon-si 16419, Gyeonggi-do, Korea; dwkang@security.re.kr (D.K.); hjlee@security.re.kr (H.L.)
3 Department of Computer Engineering, Sungkyunkwan University, 2066 Seobu-ro, Jangan-gu, Suwon-si 16419, Gyeonggi-do, Korea; hyoung@skku.edu
* Correspondence: dhwon@security.re.kr; Tel.: +82-31-290-7107

† This paper is an extended version of our paper published in Jihyeon, R.; Youngsook, L.; Dongho, W. Cryptoanalysis of Lightweight and anonymous three-factor authentication and access control protocol for real-time applications in wireless sensor networks. In Proceedings of the 6th ICCST 2019, Kota Kinabalu, Malaysia, 29–30 August 2019.

Received: 18 November 2020; Accepted: 9 December 2020; Published: 12 December 2020

Abstract: Internet of Things (IoT) technology has recently been integrated with various healthcare devices to monitor patients' health status and share it with their healthcare practitioners. Since healthcare data often contain personal and sensitive information, healthcare systems must provide a secure user authentication scheme. Recently, Adavoudi-Jolfaei et al. and Sharma and Kalra proposed a lightweight protocol using hash function encryption only for user authentication on wireless sensor systems. In this paper, we found some weaknesses in target schemes. We propose a novel three-factor lightweight user authentication scheme that addresses these weaknesses and verifies the security of the proposed scheme using a formal verification tool called ProVerif. In addition, our proposed scheme outperforms other proposed symmetric encryption-based schemes or elliptic curve-based schemes.

Keywords: authentication; WSN; healthcare; IoT

1. Introduction

Digital healthcare services have recently received a considerable amount of attention as various Internet-enabled wearable devices have been deployed. Digital services can also be used to continuously monitor patients and share information with their healthcare practitioners. According to a Spyglass Consulting Group's report [1], with 100 interviewees working in medical informatics and healthcare IT technology areas, about 88% of hospitals and healthcare systems have considered adopting remote patient monitoring (RPM) as their primary business model in the future.

RPM technology is increasingly used by hospitals and medical systems [1]. We believe that continuous monitoring and fast response times are necessary for high-risk patients with chronic diseases. For example, practitioners can use a monitoring device to collect ECG signals from patients with heart-related conditions and quickly identify any suspicious changes [2–5]. Because such data (e.g., raw ECG signals) are often highly personal and sensitive, any data that have been collected in healthcare systems should be securely protected and accessible only to authorized users such as primary care physicians. Furthermore, the scheme to monitor and analyze processes should not only be safe, but also be completed in real-time because patients might otherwise be at risk.

If there is a problem with the certification of these RPM technologies, the following damage can occur. First, if hospitals do not provide fast enough authentication using a heavy enough operation, quick feedback is impossible when monitoring the patient's condition. In such cases, in a situation where urgent patients with fatal internal injuries require immediate treatment due to rapid changes, the treatment may be delayed due to the late certification speed. In the worst-case scenario, the treatment of patients can be difficult. Second, if there is a security flaw in the certification, privacy infringement of patients and medical personnel may occur. For example, if a session key is released, all medical information of the patient can be disclosed to the hacker. Therefore, in order to avoid such damage, a certification protocol is shown in Figure 1 that satisfies both high speed (i.e., lightweight computation) and safety should be used.

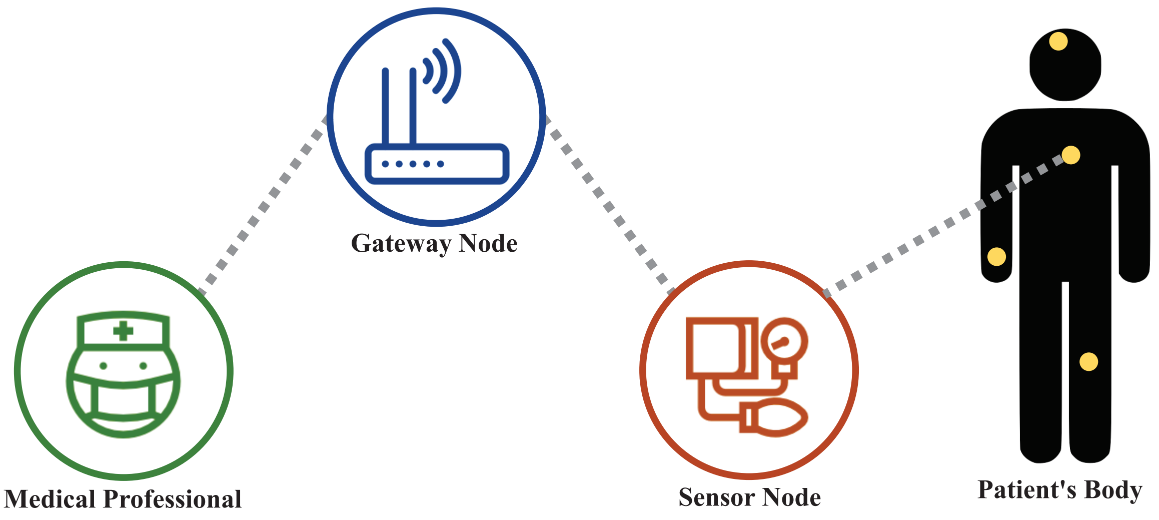

Figure 1. The Architecture of User Authentication for Digital Healthcare Services.

Many security protocols (e.g., [6–9]) have been developed to satisfy these security and performance requirements. Among those protocols, Sharma and Kalra's scheme [6] is specifically designed to improve the protocol's efficiency. Unlike existing protocols that require expensive cryptographic operations or three authentication factors [7,9,10], Sharma and Kalra's scheme [6] uses hash functions with only two authentication factors. Similarly, Adavoudi-Jolfaei et al. [11] also proposed a protocol using hash functions only. Sharma and Kala's scheme [6] and Adavoudi-Jolfei et al.'s scheme [11] are the most recently written lightweight authentication protocols that can keep high speed and safety, the conditions required by the healthcare system.

In this paper, we confirmed that Adavoudi-Jolfaei et al.'s scheme [11] has a severe vulnerability. We also demonstrate that Sharma and Kalra's scheme [6] has a serious design error. Therefore, we propose a new scheme to fix the weaknesses of these target schemes. We formally verify the security of the new protocol using ProVerif, an automatic cryptographic protocol verifier. In summary, this paper presents the following contribution:

- We demonstrate that Sharma and Kalra's scheme [6] has a serious design error: mutual authentication between a practitioner and a sensor cannot be ensured in their original protocol.
- We confirm that Adavoudi-Jolfaei et al.'s scheme [11] and Sharma and Kalra's scheme have a severe vulnerability. We find that Adavoudi-Jolfaei et al.'s scheme [11] is vulnerable to user impersonation attack and session key attack. We also find that Sharma and Kalra's scheme [6] is vulnerable to password guessing attack, stealing the session key and sensor impersonation attack.

- We propose a scheme for smart healthcare systems. Our new scheme resists privileged insider attack, outsider attack, offline ID guessing attack, online id guessing attack, session key disclosure attack, practitioner impersonation attack, and sensor impersonation attack. We provide security proofs.
- We formally verify the security of the new protocol using ProVerif, an automatic cryptographic protocol verifier.
- We show the performance analysis of the proposed scheme. We compared the proposed scheme with that of Chen et al. [12], Renuka et al. [13], and Li et al. [14] to show how efficient our proposed scheme is.

The remainder of this paper is arranged as follows: Section 2 describes related work. Section 3 introduces the preliminary knowledge necessary to understand the scheme by Sharma and Kalra and Adavoudi-Jolfaei et al. The target schemes are briefly described in Section 4. Section 5 discusses several weaknesses of the target schemes. We propose our improved scheme in Section 6 and show the security analysis of the proposed scheme in Section 7. In Section 8, we show the performance analysis of the proposed scheme. Finally, Section 9 concludes the paper.

2. Related Work

Various user authentication schemes have been proposed for smart healthcare applications.

Hu et al. [15] proposed a real-time hardware and software-based healthcare monitoring system for cardiac patients in 2007. The proposed scheme focuses on efficiency improvements but lacks adequate security protection. Malasri et al. [16] proposed an authentication scheme for wireless mote-based medical sensor networks using an ECC (elliptic curve cryptography) system in 2009. However, the scheme cannot withstand denial-of-service and relay nodes attacks. Furthermore, ECC may be too expensive for embedded devices in the medical domain.

In 2012, Kumar et al. [17] proposed a two-factor user authentication scheme for wireless medical sensors to monitor patients' health status. However, Khan and Kumari [18] found that Kumar et al.'s scheme is vulnerable to security attacks. The scheme included the use of a smart card to enhance the security of the protocol, but the user information stored on the smart card can end up leaked if the smart card is stolen. Khan and Kumari proposed an improved scheme to fix the security flaws of the previous scheme in 2014. Li et al. [19] and Wu et al. [20] each analyzed the scheme presented by Khan and Kumari [18]. They discovered that Khan and Kumari's scheme is not secure against offline password guessing attacks, as it does not identify invalid input and user impersonation attacks. Li et al. [19] and Wu et al. [20] proposed their respective improved schemes, which employ a smart card to overcome the security flaws of Khan and Kumari [18]. Hossain et al. [21] proposed an IoT-based ECG health monitoring service framework in the cloud. They presented a framework for secure transmission of patients data from different sensors to the cloud in a wireless environment.

Recently, Sharma and Kalra [6] proposed an authentication scheme for cloud-IoT-based remote patient healthcare monitoring services. It is efficient because only the hash function is used for system encryption. Several papers briefly addressed security flaws of Sharma and Karla's scheme [22,23]. However, they only mentioned briefly that privileged insider attacks are possible. However, we have pinpointed the structural problems of Sharma and Kalra's scheme, and showed that password guessing attack, stealing the session key attack, and sensor impersonation attack are possible in the case of privileged insider attack.

In 2016, Gope et al. [24] proposed a novel two-factor lightweight anonymous authentication protocol in WSNs (wireless sensor networks) that uses a database to overcome prior vulnerabilities. However, Adavoudi-Jolfaei et al. [11] argue that protocol is vulnerable to side-channel attacks because of the use of 2-factors, and that the session keys are also vulnerable. To overcome these drawbacks, in 2019, Adavoudi-Jolfaei et al. [11]. proposed a new 3-factor authentication protocol in WSN. Unfortunately, Shin and Kwon found user collusion attacks, desynchronization attack and no sensor node anonymity. In addition, through our prior research, we found more weaknesses that user impersonation attack and session key attack are able to take advantage of.

3. Preliminaries

This section introduces the hash function, fuzzy extractor and threat model used in this paper.

3.1. Hash Function

Data convert an arbitrary value to a fixed-length value through a hash function. This is useful for fast and safe search functions. The hash function has the following properties [25].

- `Preimage-resistance` It is computationally impossible to use the output of any hash value to find the input that results in this value, i.e., to find any preimage a' such that $h(a') = b$ when given any b for which a corresponding input is not known.
- `2nd-preimage-resistance` For any input, when there is an output for the hash function, it is computationally impossible to find another input value with this output, i.e., to find a 2nd-preimage $a' \neq a$ such that $h(a) = h(a')$.
- `Collision resistance` It is computationally infeasible to find two different inputs with the same hashing result, i.e., any two distinct inputs a, a', which hash to the same output, such that $h(a) = h(a')$.

3.2. Fuzzy Extractor

Biometric information should be treated as sensitive. Since biometric information is unique to the user, it is convenient to use, but difficult to handle. In general, biometrics cannot be recognized equally each time. Therefore, a fuzzy extractor is used to recognize varied biometric information within a certain tolerance range. The fuzzy extractor can obtain a unique string using error tolerance. The fuzzy extractor operates through two procedures (Gen, Rep), as follows [26,27]:

$$Gen\ (B) \rightarrow \langle \alpha, \beta \rangle \tag{1}$$

$$Rep\ (B^*, \beta) = \alpha \tag{2}$$

Gen and Rep are a probabilistic generation function and a deterministic reproduction function, respectively. Gen returns a factored out string $\alpha \in \{0,1\}^k$ for the input biometrics B and a co-adjutant string $\beta \in \{0,1\}^*$. Rep is a function that restores β to α, and any vector B^* close to B.

3.3. Threat Model

Based on the work of Dolev and Yao [28] and other previous research [10,29], we employ a threat model with the following assumptions.

- An attacker can steal a smart device with the user's identity.
- An attacker can eavesdrop on a public channel. An attacker can steal the message between the user and the gateway node or between the gateway node and the sensor node.
- An attacker can extract the information stored in the smart device as a side-channel attack.

4. Review of Target Protocols

This section describes the target protocols.

4.1. Review of Adavoudi-Jolfaei et al.'s Protocol

This section describes the protocol developed by Adavoudi-Jolfaei et al. [11]. The scheme consists of four phases: registration, login, authentication, and password change. The notation for the target paper [11] is shown in Table 1.

Table 1. Notations.

Notations	Description
U	The user
P	The practitioner who is a medical professional
GWN	Gateway node
SN	Sensor node
U_{id}	U's identity
U_{psw}	U's password
U_b	U's biometric information
AU_{id}	U's disposable identity
SU_{id}	U's shadow identity
APM	Set of access rights mask for U
G	Group identity set of U
B_p	Biometric information of the practitioner P
ID_p	Identity of the practitioner P
PW_p	Password of the practitioner P
$Mask(PW_p)$	Masked password of the user P
GWN_{id}	GWN's identity
w	GWN's private key
KEM_{ug}	The secret emergency key between U and GWN
Sk_{ug}	The secret key between U and GWN
SN_{id}	SN's identity
Sk_{gs}	The secret key between GWN and SN
SK	The session key between U and SN
SC	Smart card or smart device
DB	Database
Ts_{ug}	Timestamp sequence
$Mask(PW_p)$	Masked password of the user P
K	Secret key of GWN
$h(\cdot)$	One-way hash function
T_x	The x^{th} timestamp
T_c	Current timestamp
Gen	A probabilistic generation function
Rep	A deterministic reproduction function
ΔT	Maximum transmission delay
$\oplus$	XOR operation
$\|\|$	Concatenation operation

4.1.1. Registration Phase

In the registration phase, U and GWN in the private channel exchange secret information about SC. When the user authenticates, this allows confidential information to be stored in the database used by SC and GWN.

1. User U chooses his/her identity U_{id} and sends the registration request U_{id} and personal credential to the gateway node GWN in the secure channel.
2. The gateway node GWN generates a random number n_g, a unique random number used to identify a particular access group G_j, a random number user access privilege mask APM_j and random sequence number Ts_{ug}. Then, the created variables are grouped as $G = \{G_1, G_2, ...\}$, $APM = \{APM_1, APM_2, ...\}$. After obtaining the registration request from user U, GWN calculates as follows:

$$Sk_{ug} = h\ (U_{id} \parallel n_g) \oplus GW_{id}$$
$$sid_j = h(U_{id} \parallel r_j \parallel Sk_{ug})$$
$$SU_{id} = \{sid_1, sid_2, ...\}$$
$$KEM_{ug_j} = h\ (U_{id} \parallel sid_j \parallel r'_j)$$
$$G = \{G_1, G_2, ...\}$$
$$APM = \{APM_1, APM_2, ...\}$$

$U_{id}^{\#} = U_{id} \oplus h\,(GWN_{id} \parallel w \parallel Ts_{ug})$
$Sk_{ug}^{\#} = Sk_{ug} \oplus h\,(GWN_{id} \parallel U_{id} \parallel w)$
$G_j^{\#} = G_j \oplus h\,(GWN_{id} \parallel U_{id} \parallel w)$
$APM_j^{\#} = APM_j \oplus h\,(GWN_{id} \parallel U_{id} \parallel w)$
$Sk_{gs}^{\#} = Sk_{gs} \oplus h\,(GWN_{id} \parallel w \parallel SN_{id})$
$KEM_{ug}^{\#} = KEM_{ug} \oplus h\,(GWN_{id} \parallel U_{id} \parallel w)$ using its secret key w.

The data are saved $\langle Ts_{ug}, (SU_{id}, KEM_{ug}^{\#}), Sk_{ug}^{\#}, Sk_{gs}^{\#}, U_{id}^{\#}, G^{\#}, APM^{\#}\rangle$ in DB.GWN sends $\langle Sk_{ug}, (SU_{id}, KEM_{ug}), Ts_{ug}, G_U, h\,(\cdot)\rangle$ to user U in SC.

3. After user U takes SC from the GWN, chooses his/her U_{id}, password U_{psw}, imprints the biometric U_b and then computes as follows:

   $Gen(U_b) = (RS_U, P_U)$
   $Sk_{ug}^{*} = Sk_{ug} \oplus h\,(h\,(U_{id}) \oplus h\,(U_{psw}) \oplus h\,(RS_U))$
   $KEM_{ug}^{*} = KEM_{ug} \oplus h\,(h\,(U_{id}) \oplus h\,(U_{psw}) \oplus h\,(RS_U))$
   $SU_{id}^{*} = SU_{id} \oplus h\,(h\,(U_{id}) \oplus h\,(U_{psw}) \oplus h\,(RS_U))$
   $G^{*} = G \oplus h\,(h\,(U_{id}) \oplus h\,(U_{psw}) \oplus h\,(RS_U)),\ f_U^{*} = h\,(h\,(Sk_{ug}) \oplus h\,(U_{id}) \oplus h\,(U_{psw}) \oplus h\,(RS_U))$

   Moreover, save the data $\langle Sk_{ug}^{*}, f_U^{*}, (SU_{id}^{*}, KEM_{ug}^{*}), Tsug, G^{*}, P_U, Gen\,(\cdot), Rep\,(\cdot), h\,(\cdot)\rangle$ in SC.

4.1.2. Login Phase

In the login phase, the user enters his/her confidential information into the smart card and requests login.

1. U inserts the smart card and enters U_{id}, U_{psw} and U_b. The smart card computes $RS_U = Rep\,(U_b, P_U)$, $Sk_{ug} = Sk_{ug}^{*} \oplus h\,(h\,(U_{id}) \oplus h\,(U_{psw}) \oplus h\,(RS_U))$ and checks the condition $f_U = h\,(h\,(Sk_{ug}) \oplus h\,(U_{psw}) \oplus h\,(U_{id}) \oplus h\,(RS_U)) \overset{?}{=} f_U^{*}$. If it holds, the smart card ensures that the user successfully passes the verification process. Otherwise, this phase terminates immediately.
2. After successful verification, user U generates random number N_u and the system computes as follows:

   $N_x = Sk_{ug} \oplus N_u$
   $G = G^{*} \oplus h\,(h\,(U_{id}) \oplus h\,(U_{psw}) \oplus h\,(RS_U))$
   $AU_{id} = h\,(U_{id} \parallel Sk_{ug} \parallel N_u \parallel Ts_{ug})$
   $G_j^{'} = G_j \oplus N_u$
   $V_1 = h\,(AU_{id} \parallel G_j^{'} \parallel Sk_{ug} \parallel N_x \parallel SN_{id})$

   If there is a loss of synchronization, user U selects one of the unused pair of $(sid_j^{*}, KEM_{ug_j}^{*})$ from $(SU_{id}^{*}, KEM_{ug}^{*})$ and surrenders his/her U_{id}, U_{psw}, RS_U and computes $sid_j = sid_j^{*} \oplus h\,(h\,(U_{id}) \oplus h\,(U_{psw}) \oplus h\,(RS_U))$, $KEM_{ug} = KEM_{ug}^{*} \oplus h\,(h\,(U_{id}) \oplus h\,(U_{psw}) \oplus h\,(RS_U))$, $AU_{id} = sid_j$ and $Sk_{ug} = KEM_{ug_j}$.
3. U sends the login request messages $M_{A_1} = \{AU_{id}, G_j^{'}, N_x, Ts_{ug}(ifreq), SN_{id}, V_1\}$ to GWN.

4.1.3. Authentication Phase

In the authentication phase, GWN verifies U with the login message received from U, and sends a new message containing secret information to SN. SN and U share their keys and exchange the secret information.

1. After receiving the login request messages M_{A_1} from user U, GWN first checks the validity of the transaction sequence number Ts_{ug}. GWN computes as follows:

   $N_u = Sk_{ug} \oplus N_x$
   $G_j = G_j^{'} \oplus N_u$

$h\ (GWN_{id} \parallel U_{id} \parallel w) = G_j^{\#} \oplus G_j$
$APM_j = APM_j^{\#} \oplus h\ (GWN_{id} \parallel U_{id} \parallel w)$ that $G_j^{\#}$ and $APM_j^{\#}$ are in DB.

Then, GWN calculates $AU_{id} = h\ (U_{id} \parallel Sk_{ug} \parallel N_U \parallel Ts_{ug})$, $V_1 = h\ (AU_{id} \parallel G_j^{'} \parallel Sk_{ug} \parallel N_x \parallel SN_{id})$ and checks if AU_{id} and V_1 are valid. If the verification of AU_{id} is successful, then calculation continues. Otherwise, GWN terminates the session. GWN generates a session key SK and time stamp T and calculates as follows:

$SK^{'} = h\ (Sk_{gs}) \oplus SK$, $APM_j^{'} = h\ Sk_{gs} \oplus APM_j$ and
$V_2 = h\ (AU_{id} \parallel APM_j^{'} \parallel SK^{'} \parallel T \parallel Sk_{gs})$

Finally, GWN sends the messages $M_{A_2} = \{AU_{id}, APM_j^{'}, SK^{'}, T, V_2\}$ to the sensor node SN.

2. Upon receiving the message M_{A_2}, SN assess the validity of T. If it is not valid, SN disconnects the session. If it is valid, SN also verifies $V_2 \overset{?}{=} h\ (AU_{id} \parallel APM_j^{'} \parallel SK^{'} \parallel T \parallel Sk_{gs})$. If this condition is not satisfied, SN disconnects the session. If it is satisfied, SN computes as follows:

   $APM_j = APM_j^{'} \oplus h\ (Sk_{gs})$ and generates a new time stamp $T^{'}$.
   $SK = h\ (Sk_{gs}) \oplus SK^{'}$, $V_3 = h\ (SK \parallel Sk_{gs} \parallel SN_{id} \parallel T^{'})$
   $K_{gs_{new}} = h\ (Sk_{gs} \parallel SN_{id})$ and
   $Sk_{gs} = K_{gs_{new}}$

   Finally, SN transmits $M_{A_3} = \{T^{'}, SN_{id}, V_3\}$ to GWN.

3. The gateway node GWN checks that the time stamp T' and $V_3 \overset{?}{=} h\ (SK \parallel Sk_{gs} \parallel SN_{id} \parallel T^{'})$. If not, it terminates the connection. GWN generates a random number $Ts_{ug_{new}}$ and calculates as follows:

   $Ts = h\ (Sk_{ug} \parallel U_{id} \parallel N_U)$
   $SK^{''} = h\ (Sk_{ug} \parallel U_{id} \parallel N_U) \oplus SK$
   $V_4 = h\ (SK^{''} \parallel N_U \parallel Ts \parallel Sk_{ug})$
   $K_{ug_{new}} = h\ (Sk_{ug} \parallel U_{id} \parallel Ts_{ug_{new}})$
   $Sk_{ug} = K_{ug_{new}}$
   $K_{gs_{new}} = h(\ Sk_{gs} \parallel SN_{id})$

   GWN updates $Sk_{ug} = K_{ug_{new}}$ and $Sk_{gs} = K_{gs_{new}}$. If GWN cannot find Ts_{ug} in M_{A_1}, GWN generates a random number $K_{ug_{new}}$ and calculates $x = h\ (U_{id} \parallel KEM_{ug_j}) \oplus K_{ug_{new}}$. Then, GWN updates $Sk_{ug} = K_{ug_{new}}$ and then sends the messages $M_{A_4} = \{SK^{''}, Ts, V_4, x\}$ to the user U.
4. When user U obtains the message $V_4 = h\ (SK^{''} \parallel N_U \parallel Ts \parallel Sk_{ug})$, the protocol checks its validity. If there is no abnormality, the system proceeds to the next step or ends the session. Furthermore, U computes $SK = h\ Sk_{ug} \parallel U_{id} \parallel N_U) \oplus SK^{''}$, $Ts_{ug_{new}} = h\ (Sk_{ug} \parallel U_{id} \parallel N_U) \oplus Ts$, $K_{ug_{new}} = h\ (Sk_{ug} \parallel U_{id} \parallel Ts_{ug_{new}}$ and then updates $Sk_{ug} = K_{ug_{new}}$ and $Ts_{ug} = Ts_{ug_{new}}$.
5. U and SN have successfully shared SK. SN responds to user U's query according to APM_j stored for user U using session key SK. Finally, at the end of this phase, SN removes APM_j from storage for security reasons.

4.1.4. Password and Biometrics Change Phase

The protocol uses the following steps to change the user's password:

1. U puts his/her smart card into the terminal and inserts U_{id}, previous password U_{psw} and previous biometric U_b. U then inputs the new password U_{psw}^* and new biometric U_b^*.
2. The smart card computes $RS_U = Rep\ (U_b, P_U)$ and retrieves Sk_{ug}, KEM_{ug}, SU_{id}, G and f_U. The smart card continues to compute as follows:

   $Sk_{ug} = Sk_{ug}^* \oplus h\ (h\ (U_{id}) \oplus h\ (U_{psw}) \oplus h\ (RS_U))$
   $KM_{ug} = KEM_{ug}^* \oplus h\ (h\ (U_{id}) \oplus h\ (U_{psw}) \oplus h\ (RS_U))$

$SU_{id} = SU_{id}^{*} \oplus h\ (h\ (U_{id}) \oplus h\ (U_{psw}) \oplus h\ (RS_{U}))$
$G = G^{*} \oplus h\ (h\ (U_{id}) \oplus h\ (U_{psw}) \oplus h\ (RS_{U}))$
$f_{U} = f_{U}^{*} \oplus h\ (h\ (Sk_{ug}) \oplus h\ (U_{psw}) \oplus h\ (U_{id}) \oplus h\ (RS_{U}))$

3. The smart card computes $Gen\ (U_{b}^{*})$, Sk_{ug}^{**}, SU_{id}^{**}, KEM_{ug}^{**}, G^{**} and f_{U}^{**}, as shown below.

   $Gen\ (U_{b}^{*}) = (RS_{U}^{*}, P_{U}^{*})$
   $Sk_{ug}^{**} = Sk_{ug} \oplus h\ (h\ (U_{id}) \oplus h\ (U_{psw}^{*}) \oplus h\ (RS_{U}^{*}))$
   $SU_{id}^{**} = SU_{id} \oplus h\ (h\ (U_{id}) \oplus h\ (U_{psw}^{*}) \oplus h\ (RS_{U}^{*}))$
   $KEM_{ug}^{**} = KEM_{ug} \oplus h\ (h\ (U_{id}) \oplus h\ (U_{psw}^{*}) \oplus h\ (RS_{U}^{*}))$
   $G^{**} = G \oplus h\ (h\ (U_{id}) \oplus h\ (U_{psw}^{*}) \oplus h\ (RS_{U}^{*}))$
   $f_{U}^{**} = h\ (h\ (Sk_{ug}) \oplus h\ (U_{psw}^{*}) \oplus h\ (U_{id}) \oplus h\ (RS_{U}^{*}))$
4. Finally, the smart card replaces Sk_{ug}^{*} with Sk_{ug}^{**}, SU_{id}^{*} with SU_{id}^{**}, KEM_{ug}^{*} with KEM_{ug}^{**}, G^{*} with G^{**}, f_{U}^{*} with f_{U}^{**} and P_{U} with P_{U}^{*} .

4.2. Review of Sharma and Kalra's Scheme

This section briefly describes Sharma and Kalra's scheme. The notation of the scheme is summarized in Table 1. Sharma and Kalra's scheme consists of five different phases:

1. `Setup Phase`: The registration center sets up the parameters.
2. `Registration Phase`: The practitioner registers with his/her identity and password.
3. `Login Phase`: The practitioner logs in with his/her identity, password and smart device.
4. `Authentication Phase`: The practitioner and sensor node mutually authenticate.
5. `Password Change Phase`: The practitioner inputs identity, password and smart device, and changes his/her old password to the new password.

4.2.1. Setup Phase

The gateway node GWN obtains its secret key K from the registration center. The center also computes and gives $Sk_{gs} = h(SN_{id} \parallel K)$ to the sensor node SN. Sk_{gs} and K are stored in GWN and SN.

4.2.2. Registration Phase

The practitioner creates his/her identity and password. He/she registers through the gateway node to receive a smart device. The detailed process is as follows:

1. Practitioner P chooses his/her ID_{p} and PW_{p} and generates random number R, computes the masked password $Mask(PW_{p}) = h(PW_{p} \parallel R)$. Finally, he/she sends the registration message $\{Mask(PW_{p}), ID_{p}\}$ to the gateway node GWN.
2. After the gateway node GWN receives the message from the practitioner, it computes variables $a = h(Mask(PW_{p}) \parallel ID_{p})$, $b = h(ID_{p} \parallel K)$, $c = h(K) \oplus h(Mask(PW_{p}) \parallel b)$ and $d = b \oplus h(Mask(PW_{p}) \parallel a)$. After calculation, GWN sends the smart device $SC = \{a, c, d\}$ to P.
3. P stores $\{a, c, d, R\}$ in SC.

4.2.3. Login Phase

When the practitioner enters his/her identity and password, the smart device checks that the practitioner is an authorized party. The procedure for doing so is as follows:

1. P inputs his/her identity ID_{p} and password PW_{p} in his/her smart device.
2. SC computes $Mask(PW_{p}) = h(PW_{p} \parallel R)$, $a^{'} = h(Mask(PW_{p}) \parallel ID_{p})$ and compares a to $a^{'}$. If the two are not the same, P fails to login.

4.2.4. Authentication Phase

We describe the mutual authentication of the practitioner's login information and the sensor node. The procedure is as follows:

1. If P successfully logs in, SC computes $b = d \oplus h(Mask(PW_p) \parallel a)$, $h(K) = c \oplus h(Mask(PW_p) \parallel b)$, $V_1 = ID_p \oplus h(h(K) \parallel T_1)$. SC selects a random nonce N, and calculates $V_2 = N \oplus h(b \parallel T_1)$, $V_3 = h(V_1 \parallel V_2 \parallel N \parallel T_1)$. Finally, SC posts the message $M_1 = \{V_1, V_2, V_3, T_1, SN_{id}\}$ to GWN.
2. GWN checks that the timestamp $|T_1 - T_c| < \Delta T$. ΔT means maximum transmission delay. If it is in range, GWN chooses a random nonce M and computes:

   $MSN_{id} = SN_{id} \oplus h(h(K) \parallel T_2)$
   $V_4 = h(Sk_{gs} \parallel T_1 \parallel T_2) \oplus M$
   $V_5 = h(SN_{id} \parallel V_4 \parallel T_1 \parallel T_2 \parallel M)$

   Finally, GWN sends the message $M_2 = \{V_1, V_2, V_3, V_4, V_5, T_1, T_2, MSN_{id}\}$ to the sensor node SN.
3. SN checks the validity of $|T_2 - T_c| < \Delta T$. If it is valid, SN continues as follows:

   $MSN_{id}' = MSN_{id} \oplus h(h(K) \parallel T_2)$, $Sk_{gs}' = h(SN_{id})$
   $M' = V_4 \oplus h(Sk_{gs}' \parallel T_1 \parallel T_2)$
   $V_5' = h(SN_{id} \parallel V_4 \parallel T_1 \parallel T_2 \parallel M')$
   $ID_p' = V_1 \oplus h(h(K) \parallel T_1)$, $b' = h(ID_p' \parallel K)$
   $N' = V_2 \oplus h(b' \parallel T_1)$
   $V_3' = h(V_1 \parallel V_2 \parallel N' \parallel T_1)$, $V_6 = M' \oplus h(b' \parallel T_3)$
   $V_7 = N' \oplus h(Sk_{gs}' \parallel T_3)$
   $V_8 = h(V_6 \parallel b' \parallel T_3)$
   $V_9 = h(V_7 \parallel Sk_{gs}' \parallel T_3)$

   Finally, SN posts the message $M_3 = \{V_6, V_7, V_8, V_9, T_3\}$ to GWN.
4. GWN checks the timestamp $|T_3 - T_c| < \Delta T$, and if it is valid, computes as follows:

   $V_9' = h(V_7 \parallel Sk_{gs} \parallel T_3)$
   $N' = V_7 \oplus h(Sk_{gs} \parallel T_3)$
   $SK_{GWN} = h(N' \oplus M)$
   $V_{10} = h(SK_{GWN} \parallel V_6 \parallel V_8 \parallel T_3 \parallel T_4)$

   At the end of the computation, GWN sends the message $M_4 = \{V_6, V_8, V_{10}, T_3, T_4\}$ to P.
5. P checks the timestamp $|T_4 - T_c| < \Delta T$. If it is in range, P computes $V_8' = h(V_6 \parallel b \parallel T_3)$. It also computes $M' = V_6 \oplus h(b \parallel T_3)$, $SK_p = h(N \oplus M')$ and $V_{10}' = h(SK_p \parallel V_6 \parallel V_8 \parallel T_3 \parallel T_4)$.

4.2.5. Password Change Phase

The practitioner should be able to change his/her password if he/she wants to do so (e.g., for security reasons or because of a lost password). The procedure is as follows:

1. P inputs his/her ID_p and PW_p to SC.
2. SC computes $Mask(PW_p) = h(PW_p \parallel R)$, $a^* = h(Mask(PW_p) \parallel ID_p)$. SC verifies that $a^* = a$: if so, it computes $b = d \oplus h(Mask(PW_p) \parallel a)$, $h(K) = h(Mask(PW_p) \parallel b) \oplus c$. Finally, SC sends the message $\{$*Enter new password*$\}$ to P.
3. P inputs his/her new password PW_p^{new} to SC.
4. SC computes $Mask(PW_p)' = h(R \parallel PW_p^{new})$, $a' = h(Mask(PW_p)' \parallel ID_p)$, $d' = b \oplus h(Mask(PW_p)' \parallel a')$, $c' = h(K) \oplus h(Mask(PW_p)' \parallel b')$. Finally, SC replaces $\{a, c, d\}$ with $\{a', c', d'\}$.

5. Analysis of Target Schemes

5.1. Analysis of Adavoudi-Jolfaei et al.'s Scheme

In this section, we prove that the scheme put forth by Adavoudi-Jolfaei et al. [11] has some security vulnerabilities. The details are as follow.

5.1.1. Loss of Smart Card Information

Attacker $\mathcal{A}$ can easily decrypt the information on the SC in the following two cases. The first case is an insider attack in the registration phase, while the second case is loss of synchronization in the login phase. Insider attack is the stronger of the two: it should be considered when there is no apparent loss of synchronization.

Insider Attack

In the registration phase, Attacker $\mathcal{A}$ extracts the smart card SC when GWN sends information to U. He/she can then read the information stored on the SC $\{Sk_{ug}, SU_{id}, KEM_{ug}, Ts_{ug}, G, h(\cdot)\}$ that is not encrypted.

Loss of Synchronization

1. An attacker $\mathcal{A}$ steals U's smart card SC, which contains sensitive information $\langle Sk^*_{ug}, f^*_u, (SU^*_{id}, KEM^*_{ug}), Tsug, G^*, P_U, Gen(\cdot), Rep(\cdot), h(\cdot)\rangle$.
2. In the loss of synchronization case, $\mathcal{A}$ can thus see the user's login message $M_{A_1} = \{AU_{id}, G'_j, N_x, Ts_{ug}(ifreq), SN_{id}, V_1\}$. $\mathcal{A}$ computes as follows:

$$h(h(U_{id}) \oplus h(U_{psw}) \oplus h(RS_U)) = AU_{id} \oplus SU^*_{id}$$
$$Sk_{ug} = Sk^*_{ug} \oplus h(h(U_{id}) \oplus h(U_{psw}) \oplus h(RS_U))$$
$$KEM_{ug} = KEM^*_{ug} \oplus h(h(U_{id}) \oplus h(U_{psw}) \oplus h(RS_U))$$
$$G = G^* \oplus h(h(U_{id}) \oplus h(U_{psw}) \oplus h(RS_U))$$

5.1.2. User Impersonation Attack

Attacker $\mathcal{A}$ can carry out a user impersonation attack (the victim is assumed to be U). The details are as follows.

1. $\mathcal{A}$ generates random numbers $N_\mathcal{A}$ and computes:

$$N_{x\mathcal{A}} = Sk_{ug} \oplus N_\mathcal{A}$$
$$G'_{j\mathcal{A}} = G_j \oplus N_\mathcal{A}$$
$$AID_\mathcal{A} = h(U_{id} \parallel Sk_{ug} \parallel N_\mathcal{A} \parallel Ts_{ug})$$
$$V_{1\mathcal{A}} = h(AU_{id} \parallel G_{j\mathcal{A}} \parallel Sk_{ug} \parallel N_\mathcal{A} \parallel SN_{id})$$

   $N_{x\mathcal{A}}, G'_{j\mathcal{A}}, AID_\mathcal{A}$ and $V_{1\mathcal{A}}$ from Sk_{ug} and G_j obtained from the stolen smart card attack.
2. $\mathcal{A}$ transmits the login request $M_{\mathcal{A}1} = \{AID_\mathcal{A}, G'_{j\mathcal{A}}, N_{x\mathcal{A}}, Ts_{ug}, SN_{id}, V_{1\mathcal{A}}\}$ to the gateway node GWN.
3. After GWN obtains the login request from $\mathcal{A}$, first, it verifies Ts_{ug} and calculates:

$$N_\mathcal{A} = Sk_{ug} \oplus N_{x\mathcal{A}}$$
$$G_j = G'_{j\mathcal{A}} \oplus N_\mathcal{A}$$
$$h(GWN_{id} \parallel U_{id} \parallel w) = G^\#_j \oplus G_j$$
$$APM_j = APM^\#_j \oplus h(GWN_{id} \parallel U_{id} \parallel w)$$
$$AID_\mathcal{A} = h(U_{id} \parallel Sk_{ug} \parallel N_\mathcal{A} \parallel Ts_{ug})$$
$$V_{1\mathcal{A}} = h(AU_{id} \parallel G'_{j\mathcal{A}} \parallel Sk_{ug} \parallel N_\mathcal{A} \parallel SN_{id})$$

GWN checks if $AID_{\mathcal{A}}$ and V_1 is valid. GWN does not detect the presence of the attacker. Unfortunately, GWN still believes it is in communication with U.

As a result, attacker $\mathcal{A}$ will be verified as GWN by user U. Therefore, the user impersonation attack is successful.

5.1.3. Session Key Attack

Assume that Attacker $\mathcal{A}$ has access to the DB. At this time, Attacker $\mathcal{A}$ can extract the session key SK of user U and sensor node SN as follows.

1. Assume that attacker $\mathcal{A}$ can access the database $DB = \langle Ts_{ug}, (SU_{id}, KEM^{\#}_{ug}), Sk^{\#}_{ug}, Sk^{\#}_{gs}, U^{\#}_{id}, G^{\#}, APM^{\#}\rangle$. He/she will use the data $Sk^{\#}_{ug}$.
2. Attacker $\mathcal{A}$ extracts the message $M_{\mathcal{A}_2} = \{AU_{id}, APM'_j, SK', T, V_2\}$ and calculates:

$$\begin{aligned}
&h\ (GW_{id} \parallel U_{id} \parallel w) = Sk_{ug} \oplus Sk^{\#}_{ug}\\
&APM_j = APM^{\#}_j \oplus h\ (GW_{id} \parallel U_{id} \parallel w)\\
&h\ (Sk_{gs}) = APM'_j \oplus APM_j\\
&SK = h(Sk_{gs}) \oplus SK'
\end{aligned}$$

Thus, attacker $\mathcal{A}$ has successfully seized the session key SK.

This result shows that Adavoudi-Jolfaei et al.'s scheme does not satisfy the requirement of key security.

5.2. Analysis of Sharma and Kalra's Scheme

5.2.1. Design Error in Sharma and Kalra's Scheme

There is a fatal error in Sharma and Kalra's paper. The design of their scheme is wrong. During the authentication phase of their scheme, the session keys computed by SC and GWN are not identical. If the session key is not the same, when authentication is finished and the message is transmitted, there is a problem, because encryption is not properly performed. That is, mutual authentication would be processed incorrectly. We describe this problem in detail.

SK_p is the session key that the practitioner generates. SK_{GWN} is the session key that GWN calculates. When sending and receiving messages later, this session key is encrypted.

$$\begin{aligned}
SK_p &= h(N \oplus M')\\
&= h(N \oplus (V_4 \oplus h(Sk'_{gs} \parallel T_1 \parallel T_2)))\\
&= h(N \oplus h(Sk_{gs} \parallel T_1 \parallel T_2) \oplus M \oplus h(Sk'_{gs} \parallel T_1 \parallel T_2))\\
&= h(N \oplus M \oplus h(Sk_{gs} \parallel T_1 \parallel T_2) \oplus h(Sk'_{gs} \parallel T_1 \parallel T_2))
\end{aligned}$$

$$\begin{aligned}
SK_{GWN} &= h(N' \oplus M)\\
&= h((V_2 \oplus h(b' \parallel T_1)) \oplus M)\\
&= h(N \oplus h(b \parallel T_1) \oplus h(b' \parallel T_1) \oplus M)\\
&= h(N \oplus h(h(ID_p \parallel K) \parallel T_1) \oplus h(h(ID'_p \parallel K) \parallel T_1) \oplus M)\\
&= h(N \oplus M \oplus h(h(ID_p \parallel K) \parallel T_1) \oplus h(h(V_1 \oplus h(h(K) \parallel T_1) \parallel K) \parallel T_1))\\
&= h(N \oplus M \oplus h(h(ID_p \parallel K) \parallel T_1) \oplus h(h(ID_p \oplus h(h(K) \parallel T_1) \oplus h(h(K) \parallel T_1) \parallel K)\\
&= h(N \oplus M \oplus h(h(ID_p \parallel K) \parallel T_1) \oplus h(h(ID_p \parallel K) \parallel T_1))\\
&= h(N \oplus M)
\end{aligned}$$

In this phase, $Sk_{gs} = h(SN_{id} \parallel K)$, but $Sk_{gs} = h(SN_{id})$. Therefore, the session keys computed by SC and GWN are not the same. Therefore, the authentication phase should be changed as follows.

1. If P logs in successfully, SC computes $b = d \oplus h(Mask(PW_p) \parallel a)$, $h(K) = c \oplus h(Mask(PW_p) \parallel b)$, $V_1 = ID_p \oplus h(h(K) \parallel T_1)$. SC selects a random nonce N and calculates $V_2 = N \oplus h(b \parallel T_1)$, $V_3 = h(V_1 \parallel V_2 \parallel N \parallel T_1)$. Finally, SC posts the message $M_1 = \{V_1, V_2, V_3, T_1, SN_{id}\}$ to GWN.
2. GWN checks the timestamp $|T_1 - T_c| < \Delta T$. If it is in range, GWN computes $MSN_{id} = SN_{id} \oplus h(h(K) \parallel T_2)$ and chooses a random nonce M. GWN continues to calculate $V_4 = h(Sk_{gs} \parallel T_1 \parallel T_2) \oplus M$ and $V_5 = h(SN_{id} \parallel V_4 \parallel T_1 \parallel T_2 \parallel M)$. Finally, GWN sends the message $M_2 = \{V_1, V_2, V_3, V_4, V_5, T_1, T_2, MSN_{id}\}$ to the sensor node SN.
3. SN checks the validity of $|T_2 - T_c| < \Delta T$. If it is valid, SN continues the operation $SN'_{id} = MSN_{id} \oplus h(h(K) \parallel T_2)$ and checks that $SN_{id} \overset{?}{=} SN'_{id}$. SN keep calculating as follows:

$$\begin{aligned}
&M' = V_4 \oplus h(Sk_{gs} \parallel T_1 \parallel T_2)\\
&V'_5 = h(SN_{id} \parallel V_4 \parallel T_1 \parallel T_2 \parallel M')\\
&ID'_p = V_1 \oplus h(h(K) \parallel T_1)\\
&b' = h(ID'_p \parallel K)\\
&N' = V_2 \oplus h(b' \parallel T_1)\\
&V'_3 = h(V_1 \parallel V_2 \parallel N' \parallel T_1)\\
&V_6 = M' \oplus h(b' \parallel T_3)\\
&V_7 = N' \oplus h(Sk_{gs} \parallel T_3)\\
&V_8 = h(V_6 \parallel b' \parallel T_3)\\
&V_9 = h(V_7 \parallel Sk_{gs} \parallel T_3)
\end{aligned}$$

   Finally, SN sends the message $M_3 = \{V_6, V_7, V_8, V_9, T_3\}$ to GWN.
4. GWN checks the timestamp $|T_3 - T_c| < \Delta T$ and if it is valid, computes as follows:

$$\begin{aligned}
&V'_9 = h(V_7 \parallel Sk_{gs} \parallel T_3)\\
&N' = V_7 \oplus h(Sk_{gs} \parallel T_3)\\
&SK_{GWN} = h(N' \oplus M)\\
&V_{10} = h(SK_{GWN} \parallel V_6 \parallel V_8 \parallel T_3 \parallel T_4)
\end{aligned}$$

   At the end of the computation, GWN sends the message $M_4 = \{V_6, V_8, V_{10}, T_3, T_4\}$ to P.
5. P checks the timestamp $|T_4 - T_c| < \Delta T$. If it is in range, it computes $V'_8 = h(V_6 \parallel b \parallel T_3)$. It also computes $M' = V_6 \oplus h(b \parallel T_3)$, $SK_p = h(N \oplus M')$ and $V'_{10} = h(SK_p \parallel V_6 \parallel V_8 \parallel T_3 \parallel T_4)$.

In addition to pointing out the correctness problem in Sharma and Kalra's scheme, as discussed in Section 5.2.1, we demonstrate several attack methods that are fatal to the scheme. We describe the methods in detail as follows:

5.2.2. Password Guessing Attack

In the registration phase, if Attacker $\mathcal{A}$ masquerades as GWN, then he/she can easily obtain P's PW_p. PW_p is hashed only once in $Mask(PW_p)$ with R. We assume that R can be extracted from the smart device, and the attacker knows the target user's identity. Moreover, then, Attacker $\mathcal{A}$ knows the practitioner P's ID_p, and he/she can extract the information in SC using reverse engineering or a side-channel attack.

1. Attacker $\mathcal{A}$ extracts a and R from P's smart device SC.
2. $\mathcal{A}$ compares a and $h(h(PW_p \parallel R) \parallel ID_p)$, so that he/she can guess the password in a brute force attack.

Easily guessing a password implies knowing the practitioner's identity and likely, passwords, and having access to his/her smart device, so it is virtually the same as a practitioner. This process can also be used to pretend to be a medical professional and directly engage with the patient's healthcare-related information.

In order to make passwords difficult to guess, the authentication process should involve more robust data encryption.

5.2.3. Stealing the Session Key

We found that the session key SK_{GWN} can be extracted if attacker $\mathcal{A}$ used attacks Section 5.2.2. The details are as follows.

1. $\mathcal{A}$ computes $Mask(PW_p) = h(PW_p \parallel R)$, $b = d \oplus h(Mask(PW_p) \parallel a)$.
2. $\mathcal{A}$ steals the message M_1 and extracts V_2, T_1. Then, he/she calculates $N = V_2 \oplus h(b \parallel T_1)$.
3. $\mathcal{A}$ also steals V_6 and T_3 in M_3 and computes $M = V_6 \oplus h(b \parallel T_3)$.
4. Finally, he/she finds the session key $SK_{GWN} = h(N \oplus M)$.

$\mathcal{A}$ now has the session key to use in future messages. This session key allows attackers to check the messages. This process is a serious breach of confidentiality.

5.2.4. Sensor Impersonation Attack

A sensor impersonation attack is also possible. Since GWN only assesses the validity of the timestamp to check the sensor separately, the attacker can impersonate the sensor by sending just the timestamp. This attack generates meaningless data and wastes time.

6. Proposed Scheme

To address the problems of Sharma and Kalra's scheme and Adavoudi-Jolfaei et al.'s scheme, we propose a three-factor-based authentication scheme. We specifically introduce a new factor that is based on the practitioner's biometrics data. In addition, our scheme contains a procedure to validate both GWN and the sensor. The flow of the entire scheme is shown in the Algorithm 1.

Algorithm 1 Proposed Scheme (Overall Algorithm Flow)

```
1: $Mask(PW_p), ID_p \leftarrow$ RegistrationP($ID_p, PW_p, B_p$)        ▷ registration phase
2: $a, c, d \leftarrow$ RegistrationGWN($Mask(PW_p), ID_p$)
3: $a, c, d, R, P_{bp}, Gen, Rep, h \leftarrow$ RegistrationP2($a, c, d$)        ▷ stores in SC
4: if LoginP($ID_p, PW_p, B'_p$) then        ▷ login phase
5:     $V_1, V_2, V_3, T_1, SN_{id} \leftarrow$ AuthenticationP($ID_p, PW_p, B'_p$)        ▷ authentication phase
6:     $V_1, V_2, V_3, V_4, V_5, T_1, T_2, MSN_{id} \leftarrow$ AuthenticationGWN($V_1, V_2, V_3, T_1, SN_{id}$)
7:     $V_6, V_7, V_{10}, T_3 \leftarrow$ AuthenticationSN($V_1, V_2, V_3, V_4, V_5, T_1, T_2, MSN_{id}$)
8:     $V_6, V_8, V_{11}, T_3, T_4 \leftarrow$ AuthenticationGWN2($V_6, V_7, V_{10}, T_3$)
9:     Result $\leftarrow$ AuthenticationP2($V_6, V_8, V_{11}, T_3, T_4$)
10:    if Result then
11:        return *Authentication successes.*
12:    end if
13: end if
```

6.1. Setup and Registration Phase

6.1.1. Setup Phase

In the setup phase, the gateway node GWN gets its secret key K from the registration center. The center also computes $Sk_{gs} = h(SN_{id} \parallel K)$ and gives it to the sensor node SN. Sk_{gs} and K are stored in GWN and SN.

6.1.2. Registration Phase

The practitioner creates his/her identity and password, and then registers them via the gateway node to receive a smart device. This is summarized in the Algorithm 2. The detailed procedure is as follows:

1. Practitioner P chooses his/her identity ID_p, password PW_p and imprints B_p over a device for biometrics collection, and calculates a biometric information of the practitioner P like Gen

$(B_p) = (R_p, P_{bp})$. He/she generates random number R and computes the masked password $Mask(PW_p) = h(h(PW_p \parallel R) \parallel R_p)$. Finally, he/she sends the registration message $\{Mask(PW_p), ID_p\}$ to the gateway node GWN.

2. After the gateway node GWN obtains the message from the practitioner, it computes as follows:

$$a = h(Mask(PW_p) \parallel ID_p)$$
$$b = h(ID_p \parallel K), c = h(K) \oplus h(Mask(PW_p) \parallel b)$$
$$d = b \oplus h(Mask(PW_p) \parallel a)$$

After calculation, GWN sends the smart device $SC = \{a, c, d\}$ to the practitioner P.

3. P stores $\{a, c, d, R, P_{bp}, Gen, Rep, h\}$ in the smart device SC.

Algorithm 2 Proposed Scheme (Registration Phase)

```
1: procedure REGISTRATIONP($ID_p, PW_p, B_p$)                ▷ P's registration phase
2:     $(R_p, P_{bp}) \leftarrow Gen\ (B_p)$
3:     $Mask(PW_p) \leftarrow h(h(PW_p \parallel R) \parallel R_p)$
4:     return $Mask(PW_p), ID_p$                            ▷ message to GWN
5: end procedure
6: procedure REGISTRATIONP2($a, c, d$)                      ▷ P's registration phase 2
7:     return $a, c, d, R, P_{bp}, Gen, Rep, h$              ▷ stores in SC
8: end procedure
9: procedure REGISTRATIONGWN($Mask(PW_p), ID_p$)            ▷ GWN's registration phase
10:    $a \leftarrow h(Mask(PW_p) \parallel ID_p)$
11:    $b \leftarrow h(ID_p \parallel K)$
12:    $c \leftarrow h(K) \oplus h(Mask(PW_p) \parallel b)$
13:    $d \leftarrow b \oplus h(Mask(PW_p) \parallel a)$
14:    return $a, c, d$                                     ▷ message to P
15: end procedure
```

6.2. Login and Authentication Phase

6.2.1. Login Phase

When the practitioner enters his/her identity and password, the smart device checks whether the practitioner is an authorized party. This phase is summarized in the Algorithm 3. The detailed procedure is as follows:

1. P inputs his/her identity ID_p, password PW_p and biometric information B'_p in his/her smart device.
2. The smart device SC executes the biometric information R_p = Rep (B'_p, P_{bp}). SC computes masked password and the value a as follows:

$$Mask(PW_p) = h(h(PW_p \parallel R) \parallel R_p)$$
$$a' = h(Mask(PW_p) \parallel ID_p)$$

If a and a' are not the same, P fails to login.

Algorithm 3 Proposed Scheme (Login Phase)

```
1: procedure LOGINP($ID_p, PW_p, B'_p$)                        ▷ P's login phase
2:     $R_p \leftarrow Rep\ (B'_p, P_{bp})$
3:     $Mask(PW_p) \leftarrow h(h(PW_p \parallel R) \parallel R_p)$
4:     $a' \leftarrow h(Mask(PW_p) \parallel ID_p)$
5:     if $a = a'$ then
6:         return $True$                                       ▷ login successes
7:     else
8:         return $False$                                      ▷ login fails
9:     end if
10: end procedure
```

6.2.2. Authentication Phase

The practitioner's login information and the sensor node execute a mutual authentication process. This phase is summarized in Algorithms 4–6. The procedure is as follows:

1. If P logs in successfully, SC selects a random nonce N and computes as follows:

$$b = d \oplus h(Mask(PW_p) \parallel a)$$
$$h(K) = c \oplus h(Mask(PW_p) \parallel b)$$
$$V_1 = ID_p \oplus h(h(K) \parallel T_1)$$
$$V_2 = N \oplus h(b \parallel T_1)$$
$$V_3 = h(V_1 \parallel V_2 \parallel N \parallel T_1)$$

Finally, SC sends the message $M_1 = \{V_1, V_2, V_3, T_1, SN_{id}\}$ to GWN.

2. GWN checks the timestamp $|T_1 - T_c| < \Delta T$. If it is in range, GWN computes as follows:

$$ID' = V_1 \oplus h(h(K) \parallel T_1)$$
$$V'_3 = h(V'_1 \parallel V_2 \parallel (V_2 \oplus h(h(ID' \parallel K) \parallel T_1)) \parallel T_1)$$

GWN also checks $V_3 = V'_3$. If it is valid, GWN chooses a random nonce M and computes as follows:

$$MSN_{id} = SN_{id} \oplus h(h(K) \parallel T_2)$$
$$V_4 = h(Sk_{gs} \parallel T_1 \parallel T_2) \oplus M$$
$$V_5 = h(SN_{id} \parallel V_4 \parallel T_1 \parallel T_2 \parallel M)$$

Finally, GWN sends the message $M_2 = \{V_1, V_2, V_3, V_4, V_5, T_1, T_2, MSN_{id}\}$ to the sensor node SN.

3. SN checks the validity of $|T_2 - T_c| < \Delta T$. If it is valid, SN continues the operation $SN'_{id} = MSN_{id} \oplus h(h(K) \parallel T_2)$ and checks $SN'_{id} \overset{?}{=} SN_{id}$. SN computes as follows:

$$M' = V_4 \oplus h(Sk_{gs} \parallel T_1 \parallel T_2)$$
$$V'_5 = h(SN_{id} \parallel V_4 \parallel T_1 \parallel T_2 \parallel M')$$

It also checks the validity of $V'_5 \overset{?}{=} V_5$. If it is valid, SN computes:

$$ID' = V_1 \oplus h(h(K) \parallel T_1)$$
$$b' = h(ID' \parallel K), N' = V_2 \oplus h(b' \parallel T_1)$$
$$V_6 = M' \oplus h(b' \parallel T_3), V_7 = N' \oplus h(Sk_{gs} \parallel T_3)$$
$$V_8 = h(V_6 \parallel b' \parallel T_3)$$
$$V_9 = h(V_7 \parallel Sk_{gs} \parallel T_3)$$
$$V_{10} = h(V_8 \parallel V_9 \parallel T_3)$$

Finally, SN sends the message $M_3 = \{V_6, V_7, V_{10}, T_3\}$ to GWN.

4. GWN checks the timestamp $|T_3 - T_c| < \Delta T$ and if it is valid, GWN computes:

$V_8' = h(V_6 \parallel b \parallel T_3)$
$V_9' = h(V_7 \parallel Sk_{gs} \parallel T_3)$
$V_{10}' = h(V_8' \parallel V_9' \parallel T_3)$

and checks $V_{10} \overset{?}{=} V_{10}'$. If it is also valid, GWN computes:

$N' = V_7 \oplus h(Sk_{gs} \parallel T_3)$
$SK_{GWN} = h(N' \oplus M)$
$V_{11} = h(SK_{GWN} \parallel V_6 \parallel V_8 \parallel T_3 \parallel T_4)$

At the end of the computation, GWN sends the message $M_4 = \{V_6, V_8, V_{11}, T_3, T_4\}$ to P.

5. P checks the timestamp $|T_4 - T_c| < \Delta T$. If it is in range, P computes $V_8' = h(V_6 \parallel b \parallel T_3)$ and checks $V_8 \overset{?}{=} V_8'$. P also computes $M' = V_6 \oplus h(b \parallel T_3)$, $SK_p = h(N \oplus M')$ and checks $V_{11}' = h(SK_p \parallel V_6 \parallel V_8 \parallel T_3 \parallel T_4) \overset{?}{=} V_{11}$.

Algorithm 4 Proposed Scheme (P's Authentication Phase)

```
1: procedure AUTHENTICATIONP($ID_p, PW_p, B_p'$)                ▷ P's authentication phase
2:     $R_p \leftarrow Rep\ (B_p', P_{bp})$
3:     $Mask(PW_p) \leftarrow h(h(PW_p \parallel R) \parallel R_p)$
4:     $a' \leftarrow h(Mask(PW_p) \parallel ID_p)$
5:     $b \leftarrow d \oplus h(Mask(PW_p) \parallel a)$
6:     $h(K) \leftarrow c \oplus h(Mask(PW_p) \parallel b)$
7:     $V_1 \leftarrow ID_p \oplus h(h(K) \parallel T_1)$
8:     $V_2 \leftarrow N \oplus h(b \parallel T_1)$
9:     $V_3 \leftarrow h(V_1 \parallel V_2 \parallel N \parallel T_1)$
10:    return $V_1, V_2, V_3, T_1, SN_{id}$                      ▷ message to GWN
11: end procedure
12: procedure AUTHENTICATIONP2($V_6, V_8, V_{11}, T_3, T_4$)    ▷ P's authentication phase 2
13:    if $|T_4 - T_c| >= \Delta T$ then
14:        return $False$                                       ▷ authentication fails
15:    end if
16:    $V_8' \leftarrow h(V_6 \parallel b \parallel T_3)$
17:    if $V_8 != V_8'$ then
18:        return $False$                                       ▷ authentication fails
19:    end if
20:    $M' \leftarrow V_6 \oplus h(b \parallel T_3)$
21:    $SK_p \leftarrow h(N \oplus M')$
22:    $V_{11}' \leftarrow h(SK_p \parallel V_6 \parallel V_8 \parallel T_3 \parallel T_4)$
23:    if $V_{11}' != V_{11}$ then
24:        return $False$                                       ▷ authentication fails
25:    end if
26:    return $True$                                            ▷ authentication successes
27: end procedure
```

Algorithm 5 Proposed Scheme (GWN's Authentication Phase)

```
 1: procedure AUTHENTICATIONGWN($V_1$, $V_2$, $V_3$, $T_1$, $SN_{id}$)     ▷ GWN's authentication phase
 2:     if $|T_1 - T_c| >= \Delta T$ then
 3:         return $False$                                               ▷ authentication fails
 4:     end if
 5:     $ID^{'} \leftarrow V_1 \oplus h(h(K) \parallel T_1)$
 6:     $V_3^{'} \leftarrow h(V_1^{'} \parallel V_2 \parallel (V_2 \oplus h(h(ID^{'} \parallel K) \parallel T_1)) \parallel T_1)$
 7:     if $V_3 != V_3^{'}$ then
 8:         return $False$                                               ▷ authentication fails
 9:     end if
10:     $MSN_{id} \leftarrow SN_{id} \oplus h(h(K) \parallel T_2)$
11:     $V_4 \leftarrow h(Sk_{gs} \parallel T_1 \parallel T_2) \oplus M$
12:     $V_5 \leftarrow h(SN_{id} \parallel V_4 \parallel T_1 \parallel T_2 \parallel M)$
13:     return $V_1$, $V_2$, $V_3$, $V_4$, $V_5$, $T_1$, $T_2$, $MSN_{id}$    ▷ message to SN
14: end procedure
15: procedure AUTHENTICATIONGWN2($V_6$, $V_7$, $V_{10}$, $T_3$)          ▷ GWN's authentication phase 2
16:     if $|T_3 - T_c| >= \Delta T$ then
17:         return $False$                                               ▷ authentication fails
18:     end if
19:     $V_8^{'} \leftarrow h(V_6 \parallel b \parallel T_3)$
20:     $V_9^{'} \leftarrow h(V_7 \parallel Sk_{gs} \parallel T_3)$
21:     $V_{10}^{'} \leftarrow h(V_8^{'} \parallel V_9^{'} \parallel T_3)$
22:     if $V_{10} != V_{10}^{'}$ then
23:         return $False$                                               ▷ authentication fails
24:     end if
25:     $N^{'} \leftarrow V_7 \oplus h(Sk_{gs} \parallel T_3)$
26:     $SK_{GWN} \leftarrow h(N^{'} \oplus M)$
27:     $V_{11} \leftarrow h(SK_{GWN} \parallel V_6 \parallel V_8 \parallel T_3 \parallel T_4)$
28:     return $V_6$, $V_8$, $V_{11}$, $T_3$, $T_4$                       ▷ message to P
29: end procedure
```

Algorithm 6 Proposed Scheme (SN's Authentication Phase)

```
 1: procedure AUTHENTICATIONSN($V_1$, $V_2$, $V_3$, $V_4$, $V_5$, $T_1$, $T_2$, $MSN_{id}$)   ▷ SN's authentication phase
 2:     if $|T_2 - T_c| >= \Delta T$ then
 3:         return $False$                                               ▷ authentication fails
 4:     end if
 5:     $SN_{id}^{'} \leftarrow MSN_{id} \oplus h(h(K) \parallel T_2)$
 6:     if $SN_{id}^{'} != SN_{id}$ then
 7:         return $False$                                               ▷ authentication fails
 8:     end if
 9:     $M^{'} \leftarrow V_4 \oplus h(Sk_{gs} \parallel T_1 \parallel T_2)$
10:     $V_5^{'} \leftarrow h(SN_{id} \parallel V_4 \parallel T_1 \parallel T_2 \parallel M^{'})$
11:     if $V_5^{'} != V_5$ then
12:         return $False$                                               ▷ authentication fails
13:     end if
14:     $ID^{'} \leftarrow V_1 \oplus h(h(K) \parallel T_1)$
15:     $b^{'} \leftarrow h(ID^{'} \parallel K), N^{'} = V_2 \oplus h(b^{'} \parallel T_1)$
16:     $V_6 \leftarrow M^{'} \oplus h(b^{'} \parallel T_3)$
17:     $V_7 \leftarrow N^{'} \oplus h(Sk_{gs} \parallel T_3)$
18:     $V_8 \leftarrow h(V_6 \parallel b^{'} \parallel T_3)$
19:     $V_9 \leftarrow h(V_7 \parallel Sk_{gs} \parallel T_3)$
20:     $V_{10} \leftarrow h(V_8 \parallel V_9 \parallel T_3)$
21:     return $V_6$, $V_7$, $V_{10}$, $T_3$                               ▷ message to GWN
22: end procedure
```

6.3. Password Change Phase

The practitioner can change his/her password. The procedure is as follows:

1. P inputs his/her identity ID_p, password PW_p and biometric information B'_p in his/her smart device.
2. The smart device SC executes $Rep\ (B'_p, P_{bp}) = R_p$. SC computes $Mask(PW_p) = h(h(PW_p \parallel R) \parallel R_p)$, $a^* = h(Mask(PW_p) \parallel ID_p)$. SC checks $a^* \overset{?}{=} a$. If so, it computes $b = d \oplus h(Mask(PW_p) \parallel a)$, $h(K) = h(Mask(PW_p) \parallel b) \oplus c$. Finally, SC sends the message $\{Enter\ new\ password\}$ to P.
3. P inputs his/her new password PW_p^{new} to the smart device SC.
4. SC computes $Mask(PW_p)^{'} = h(R \parallel PW_p^{new})$, $a^{'} = h(Mask(PW_p)^{'} \parallel ID_p)$, $d^{'} = b \oplus h(Mask(PW_p)^{'} \parallel a^{'})$, $c^{'} = h(K) \oplus h(Mask(PW_p)^{'} \parallel b^{'})$. Finally, SC replaces $\{a, c, d\}$ with $\{a^{'}, c^{'}, d^{'}\}$.

7. Security Analysis

In this section, we analyze the security of the proposed scheme in two ways: formal security analysis and informal security analysis. We use a formal protocol verification tool called ProVerif in Section 7.1, to demonstrate the security of our scheme. We also provide a theoretical security analysis of the protocol in Section 7.2. Through this verification, we have demonstrated how safe our scheme can be in reality.

7.1. Formal Security Analysis

We use ProVerif to analyze the security and correctness of our scheme; ProVerif is widely used to verify security protocols [7,30,31]. ProVerif is a software tool that formally verifies the security of cryptographic protocols. We define basic cryptographic primitives such as hash function, encryption, digital signature and bit-commitment.

We used three channels: a registration channel (*cha*), a practitioner–gateway node channel (*chb*) and a gateway node–sensor node channel (*chc*). The variables, constants, secret key, functions and events are defined in Table A1.

The "Registration" and "Login and Authentication" phases for practitioners are shown in Table A2. The "Registration" and "Authentication" phases for gateway nodes are shown in Table A3. The "Authentication" phase for sensor nodes is shown in Table A4. Tables 2 and 3 show a query and the corresponding query results.

When we run the query in Table 2, we obtain the following result:

1. RESULT inj-event(EVENTA) ==> inj-event(EVENTB) is true.
2. RESULT inj-event(EVENTA) ==> inj-event(EVENTB) is false.
3. RESULT not attacker(QUERY) is true.
4. RESULT not attacker(QUERY) is false.

"RESULT inj-event (EVENTA) == > inj-event (EVENTB) is true." means that EVENTA to EVENTB has been authenticated. On the other hand, "RESULT inj-event (EVENTA) == > inj-event (EVENTB) is false." means that the authentication from EVENTA to EVENTB was not successful. "RESULT not attacker (QUERY) is true." means that an attacker cannot get a free name QUERY, and "RESULT not attacker (QUERY) is false." means that an attacker can trace a QUERY [32].

The results of the query in Table 2 are shown in Table 3. In that case, the authentication process is correctly performed and the attacker cannot obtain IDp.

Table 2. Query.

```
(*—-queries—-*)
query attacker(IDp).
query id:bitstring; inj-event(endP(id)) ==> inj-event(beginP(id)).
query id:bitstring; inj-event(endGWN(id)) ==> inj-event(beginGWN(id)).
query id:bitstring; inj-event(endS(id)) ==> inj-event(beginS(id)).
process
((!P) | (!GWN) | (!S))
```

Table 3. Query Results.

```
RESULT inj-event(endS(id)) ==> inj-event(beginS(id)) is true.
RESULT inj-event(endGWN(id_21256)) ==> inj-event(beginGWN(id_21256)) is true.
RESULT inj-event(endP(id_41657)) ==> inj-event(beginP(id_41657)) is true.
RESULT not attacker(IDp[]) is true.
```

7.2. *Informal Security Analysis*

We present a theoretical analysis of our scheme. We then briefly explain the results of the analysis.

7.2.1. Privileged Insider Attack

In the registration step, the practitioner sends his/her plaintext ID and $Mask(PW_p) = h(h(PW_p \parallel R) \parallel R_p)$ to the gateway node. Since the ID is used in conjunction with secret information K without being exposed to the outside, there is no way for an insider to know the practitioner's personal information. Therefore, it is secure against a privileged insider attack.

7.2.2. Outsider Attack

The SC only contains the information $\{a, c, d, R, P_{bp}, Gen, Rep, h\}$. We cannot infer practitioner P in this case.

7.2.3. Offline ID Guessing Attack

The practitioner's identity is not moved directly in the login and authentication phase after registration. When the sensor checks the practitioner's ID, the information V_1, K, and T_1 is required, all of which cannot be obtained through SC.

7.2.4. Online ID Guessing Attack

The practitioner's identity can only be exposed as V_1, K, and T_1, as mentioned in the offline ID guessing attack. At this time, V_1 and T_1 can be seized through a message sent to the GWN by the practitioner in the authentication phase, but the secret GWN key K cannot be found.

7.2.5. Session Key Disclosure Attack

As shown in Section 5.2.1, the session key consists of $h(N \oplus M)$. N is a value generated by SC and $N = V_2 \oplus h(b \parallel T_1)$, and M is a value generated by GWN and expressed as $M = V_4 \oplus h(X_{GWN} \parallel T_1 \parallel T_2)$. Because the attacker is not the practitioner or GWN, he/she cannot know the session key because X_{GWN} including b and $Mask(PW)$ including K cannot be known.

7.2.6. Practitioner Impersonation Attack

GWN verifies the practitioner in equation $V_3' = h(V_1' \parallel V_2 \parallel V_2 \oplus h(h(ID' \parallel K) \parallel T_1) \parallel T_1) \overset{?}{=} V_3 = h(ID_p \oplus h(h(K) \parallel T_1) \parallel N \oplus h(b \parallel T_1) \parallel N \parallel T_1)$. If an attacker pretends to be an authorized practitioner, then he/she needs to know $b = d \oplus h(Mask(PW) \parallel a)$, $h(K) = c \oplus h(Mask(PW) \parallel b)$.

However, the attacker cannot obtain $Mask(PW)$. Therefore, the attacker cannot impersonate the practitioner P.

7.2.7. Sensor Impersonation Attack

GWN verifies a sensor with the Equation $V_{10} = h(V_8 \parallel V_9 \parallel T_3)$ and checks $V_{10} \overset{?}{=} V_{10}'$ that $V_8 = h(V_6 \parallel b \parallel T_3)$, $V_9 = h(V_7 \parallel Sk_{gs} \parallel T_3)$. This means that a sensor can only prove that it is a sensor if it knows Sk_{gs} and b.

8. Performance Analysis of the Proposed Scheme

The four symbols necessary for comparison are as follows. T_{Rep} is the time to check for a match when recognizing the user's (or practitioner's) biometric B_p. T_h is the time it takes to hash. T_m is the time of the multiplicative operation used in ECC. The time taken for symmetric encryption or decryption is denoted by T_s.These are listed in Table 4. We compared our scheme with the following three schemes of Chen et al. [12], Renuka et al. [13] and Li et al. [14]. The four symbols necessary for comparison are as follows, and depicted as a graph in Figure 2.

Table 4. Notations of Time Symbol.

Symbol	Meaning	Time (ms)
T_{Rep}	time of *Rep*	7.3529 [33]
T_h	time of hash operation	0.0004 [34]
T_m	time of multiplication in ECC	7.3529 [34]
T_s	time of symmetric encryption or decryption	0.1303 [34]

Table 5 summarizes the total time cost for each scheme ([12–14]). The Y-axis shown in Figure 2 is the total time cost in microseconds (ms). This is calculated based on the times shown in Table 4. Table 6 shows the computer hardware and software used to calculate the algorithm's runtime. Li et al.'s scheme [14] uses elliptic curve cryptography, however it has large time costs because of its slow encryption and decryption.

Table 5. Comparison of Computation Costs.

	Chen et al. [12]	**Renuka et al. [13]**	**Li et al. [14]**	**Ours**
User(Practitioner) P	$9T_h + 1T_{Rep} + 1T_s$	$5T_h + 1T_{Rep} + 2T_s$	$8T_h + 1T_{Rep} + 2T_m$	$13T_h + 1T_{Rep}$
GWN	$3T_h + 2T_s$	$2T_h + 2T_s$	$7T_h + 1T_m$	$14T_h$
Sensor node S_j	$4T_h + T_s$	$3T_h + 2T_s$	$4T_h + 2T_m$	$12T_h$
Total time cost (ms)	$16T_h + 4T_s + 1T_{Rep}$ =7.8805	$10T_h + 6T_s + 1T_{Rep}$ =8.1387	$19T_h + 5T_m + 1T_{Rep}$ =44.125	$39T_h + 1T_{Rep}$ =7.3685

The methods used in Chen et al.'s [12] and Renuka et al.'s [13] scheme have lower costs than Li et al.'s [14]. They used symmetric encryption and decryption, but had a significant difference in cost compared to ours.

Table 6. Hardware and Software Condition.

	Specification
CPU	Intel (R) Core(TM) 2T6570 2.1GHz
Memory	4G
OS	Win7 32-bit
Software	Visual C++ 2008

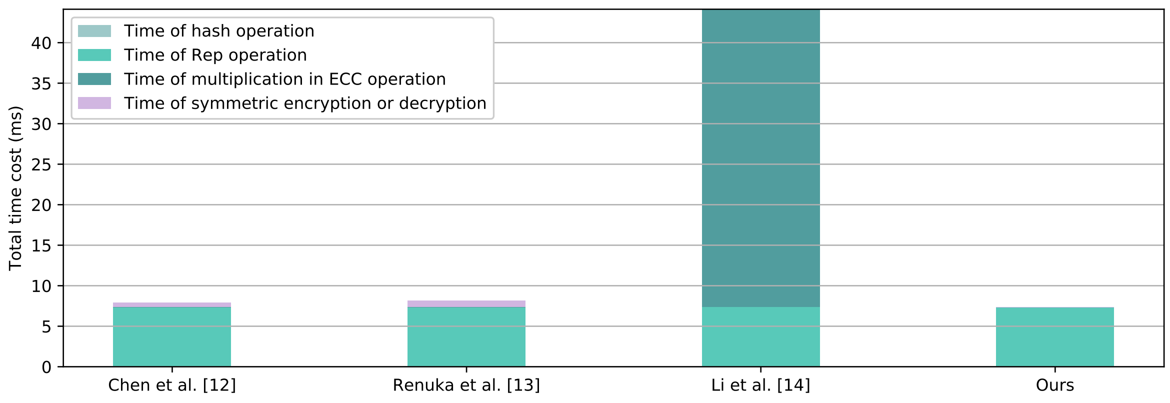

Figure 2. Execution Time of Schemes.

9. Conclusions

Recently, several lightweight two-factor-based authentication protocols, such as Sharma and Kalra's protocol [6] and Adavoudi-Jolfaei et al.'s protocol [11], were proposed for IoT applications. Those protocols have some benefits in efficiency because they are designed with light and straightforward operations, such as hash function and XOR without complicated cryptographic operations. However, we found that those protocols have several security weaknesses.

To address these problems, we use three factors: the practitioner's identity, a password, and biometric information. We propose a lightweight three-factor user authentication scheme to fix several issues in Sharma and Kalra's protocol [6] and Adavoudi-Jolfaei et al.'s protocol [11]. We provide the security verification of the proposed scheme using ProVerif. Furthermore, we show that our proposed method outperforms other proposed symmetric-based or elliptic curve-based methods.

We are motivated to fix the problems of existing protocols and proposed a more efficient and secure authentication scheme. Our scheme is designed only with the hash function and XOR, providing fast and secure authentication. The proposed protocol can be used for various IoT applications such as medical devices.

Our proposed scheme can be implemented with simple operations, but introduces 11 new parameters. Furthermore, while our scheme is lighter than the symmetric-based or elliptic curve-based methods, it is not significantly lighter than the only used XOR and hash method. Therefore, in future work, we aim to develop a simpler authentication scheme with fewer variables. We also aim to develop a lightweight 3-factor authentication scheme with fewer xor operations and hash functions. Our efforts will contribute to creating a faster and more secure authentication scheme.

Author Contributions: Conceptualization, J.R. and D.K.; methodology, J.R. and H.L.; software, H.L.; validation, J.R.; formal analysis, J.R.; writing—original draft preparation, J.R.; writing—review and editing, J.R. and H.K.; supervision, D.W.; project administration, J.R.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This work was supported by an Institute of Information & Communications Technology Planning Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2020-0-00258, Development of On-chain-based Electronic Contract Application Platform Using Zero-Knowledge Proof).

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

Appendix A. ProVerif Code

Table A1. Define Values and Functions.

```
(*—-channels—-*)
free cha:channel [private].
free chb:channel.
free chc:channel.
(*—-constants—-*)
free Ru:bitstring [private].
free IDp:bitstring [private].
free IDg:bitstring [private].
free IDs:bitstring.
free PWu:bitstring [private].
(*—-secret key—-*)
free K:bitstring [private].
(*—-functions—-*)
fun concat(bitstring, bitstring) : bitstring.
fun xor(bitstring, bitstring) : bitstring.
fun h(bitstring) : bitstring.
equation forall a:bitstring, b:bitstring; xor(xor(a, b), b) = a.
(*—-events—-*)
event beginP(bitstring).
event endP(bitstring).
event beginGWN(bitstring).
event endGWN(bitstring).
event beginS(bitstring).
event endS(bitstring).
```

Table A2. Practitioner Scheme.

```
(*—-P process—-*)
let P =
new R:bitstring;
let MPW = h(concat(h(concat(PWp, R)), Rp)) in
out(cha,(IDp, MPW));
in(cha,(Xa:bitstring, Xc:bitstring, Xd:bitstring));
event beginP(IDp);
new T1:bitstring;
new N:bitstring;
let ppb = xor(Xd, h(concat(MPW, Xa))) in
let hK = xor(Xc, h(concat(MPW, ppb))) in
let V1 = xor(IDp, h(concat(hK, T1))) in
let V2 = xor(N, h(concat(ppb, T1))) in
let V3 = h(concat(concat(V1, V2), concat(N, T1))) in
out(chb, (V1, V2, V3, T1, N));
in(chb, (XXV6:bitstring, XXV8:bitstring, XV11:bitstring, XXT3:bitstring, XT4:bitstring));
let pV8 = h(concat(concat(XXV6, ppb), XXT3)) in
let pM = xor(XXV6, h(concat(ppb, XXT3))) in
let SKp = h(xor(N, pM)) in
let pV11 = h(concat(concat(SKp, XXV6), concat(concat(XXV8, XXT3),XT4))) in
event endP(IDp).
```

Table A3. Gateway Node Scheme.

```
(*—–GWN process—–*)
let GWN =
in(cha,(XIDp:bitstring, XMPW:bitstring));
let a = h(concat(XMPW, XIDp)) in
let b = h(concat(XIDp, K)) in
let c = xor(h(K), h(concat(XMPW, b))) in
let d = xor(b, h(concat(XMPW, a))) in
out(cha, (a, c, d));
in(chb, (XV1:bitstring, XV2:bitstring, XV3:bitstring, XT1:bitstring, XN:bitstring));
event beginGWN(IDg);
new T2:bitstring;
new M:bitstring;
let pIDp = xor(XV1, h(concat(h(K), XT1))) in
let pV3 = h(concat(concat(XV1, XV2), concat(XN, XT1))) in
if pV3 = XV3 then
let MID = xor(IDs, h(concat(h(K), T2))) in
let X = h(concat(IDs, K)) in
let V4 = xor(M, h(concat(concat(X, XT1), T2))) in
let V5 = h(concat(concat(IDs, V4), concat(concat(XT1, T2), M))) in
out(chc, (XV1, XV2, XV3, V4, V5, XT1, T2, MID));
in(chc, (XV6:bitstring, XV7:bitstring, XV10:bitstring, XT3:bitstring));
new T4:bitstring;
new M:bitstring;
let pV8 = h(concat(XV6, concat(b, XT3))) in
let pV9 = h(concat(XV7, concat(X, XT3))) in
let pV10 = h(concat(pV8, concat(pV9, XT3))) in
if pV10 = XV10 then
let pN = xor(XV7, h(concat(X, XT3))) in
let SKg = h(xor(pN, M)) in
let V11 = h(concat(SKg, concat(concat(XV6, pV8), concat(XT3, T4)))) in
out(chb, (XV6, pV8, V11, XT3, T4));
event endGWN(IDg).
```

Table A4. Sensor Scheme.

```
(*—–S process—–*)
let S =
in(chc, (XXV1:bitstring, XXV2:bitstring, XXV3:bitstring, XV4:bitstring, XV5:bitstring,
XXT1:bitstring, XT2:bitstring, XMID:bitstring));
event beginS(IDs);
new T3:bitstring;
let XX = h(concat(IDs, K)) in
let pM = xor(XV4, h(concat(concat(XX, XXT1), XT2))) in
let pV5 = h(concat(concat(IDs, XV4), concat(concat(XXT1, XT2), pM))) in
if pV5 = XV5 then
let pb = h(concat(IDp, K)) in
let ppN = xor(XXV2, h(concat(pb, XXT1))) in
let V6 = xor(pM, h(concat(pb, T3))) in
let V7 = xor(ppN, h(concat(XX, T3))) in
let V8 = h(concat(concat(V6, pb), T3)) in
let V9 = h(concat(concat(V7, XX), T3)) in
let V10 = h(concat(concat(V8, V9), T3)) in
out(chc, (V6, V7, V10, T3));
event endS(IDs).
```

References

1. Gregg, M. Trends in Remote Patient Monitoring 2019. Spyglass Consulting Group. Available online: http://www.spyglass-consulting.com/wp_RPM_2019.html (accessed on 12 December 2020).
2. Hu, Y.H.; Tompkins, W.J.; Urrusti, J.L.; Afonso, V.X. Applications of artificial neural networks for ECG signal detection and classification. *J. Electrocardiol.* **1993**, *26*, 66–73. [PubMed]
3. Yeh, Y.C.; Wang, W.J. QRS complexes detection for ECG signal: The Difference Operation Method. *Comput. Methods Programs Biomed.* **2008**, *91*, 245–254. [CrossRef] [PubMed]
4. Van Ess, D.W. ECG Signal Detection Device. US Patent 7,092,750, 15 August 2006.
5. Chung, W.Y.; Lee, Y.D.; Jung, S.J. A wireless sensor network compatible wearable u-healthcare monitoring system using integrated ECG, accelerometer and SpO_2. In Proceedings of the 2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Vancouver, BC, Canada, 20–25 August 2008; pp. 1529–1532.
6. 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]
7. Ryu, J.; Lee, H.; Kim, H.; Won, D. Secure and efficient three-factor protocol for wireless sensor networks. *Sensors* **2018**, *18*, 4481. [CrossRef] [PubMed]
8. Rathore, H.; Al-Ali, A.; Mohamed, A.; Du, X.; Guizani, M. DTW based authentication for wireless medical device security. In Proceedings of the 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC), Limassol, Cyprus, 25–29 June 2018; pp. 476–481.
9. 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**, 1–22. [CrossRef]
10. Choi, Y.; Lee, D.; Kim, J.; Jung, J.; Nam, J.; Won, D. Security enhanced user authentication protocol for wireless sensor networks using elliptic curves cryptography. *Sensors* **2014**, *14*, 10081–10106. [CrossRef] [PubMed]
11. Adavoudi-Jolfaei, A.; Maede, A.T.; Aghili, S.F. Lightweight and anonymous three-factor authentication and access control scheme for real-time applications in wireless sensor networks. *Peer-to-Peer Netw. Appl.* **2019**, *12*, 43–59. [CrossRef]
12. Chen, Y.; Ge, Y.; Wang, Y.; Zeng, Z. An improved three-factor user authentication and key agreement scheme for wireless medical sensor networks. *IEEE Access* **2019**, *7*, 85440–85451. [CrossRef]
13. Renuka, K.; Kumar, S.; Kumari, S.; Chen, C.M. Cryptanalysis and improvement of a privacy-preserving three-factor authentication protocol for wireless sensor networks. *Sensors* **2019**, *19*, 4625. [CrossRef]
14. Li, X.; Niu, J.; Bhuiyan, M.Z.A.; Wu, F.; Karuppiah, M.; Kumari, S. A robust ECC-based provable secure authentication protocol with privacy preserving for industrial internet of things. *IEEE Trans. Ind. Inform.* **2017**, *14*, 3599–3609. [CrossRef]
15. Hu, F.; Jiang, M.; Wagner, M.; Dong, D.C. Privacy-preserving telecardiology sensor networks: Toward a low-cost portable wireless hardware/software codesign. *IEEE Trans. Inf. Technol. Biomed.* **2007**, *11*, 619–627. [CrossRef] [PubMed]
16. Malasri, K.; Wang, L. Design and implementation of a securewireless mote-based medical sensor network. *Sensors* **2009**, *9*, 6273–6297. [CrossRef] [PubMed]
17. Kumar, P.; Lee, S.G.; Lee, H.J. E-SAP: Efficient-strong authentication protocol for healthcare applications using wireless medical sensor networks. *Sensors* **2012**, *12*, 1625–1647. [CrossRef] [PubMed]
18. Khan, M.K.; Kumari, S. An improved user authentication protocol for healthcare services via wireless medical sensor networks. *Int. J. Distrib. Sens. Netw.* **2014**, *10*, 347169. [CrossRef]
19. 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]
20. Wu, F.; Xu, L.; Kumari, S.; Li, X. An improved and anonymous two-factor authentication protocol for health-care applications with wireless medical sensor networks. *Multimed. Syst.* **2017**, *23*, 195–205. [CrossRef]
21. Hossain, M.S.; Muhammad, G. Cloud-assisted speech and face recognition framework for health monitoring. *Mob. Netw. Appl.* **2015**, *20*, 391–399. [CrossRef]

22. Wazid, M.; Das, A.K.; Shetty, S.; Rodrigues, J.J.P.C.; Park, Y. LDAKM-EIoT: Lightweight device authentication and key management mechanism for edge-based IoT deployment. *Sensors* **2019**, *19*, 5539. [CrossRef]
23. Tanveer, M.; Zahid, A.H.; Ahmad, M.; Baz, A.; Alhakami, H. LAKE-IoD: Lightweight Authenticated Key Exchange Protocol for the Internet of Drone Environment. *IEEE Access* **2020**, *8*, 155645–155659. [CrossRef]
24. Gope, P.; Tzonelih, H. A realistic lightweight anonymous authentication protocol for securing real-time application data access in wireless sensor networks. *IEEE Trans. Ind. Electron.* **2016**, *63*, 7124–7132. [CrossRef]
25. Katz, J.; Menezes, A.J.; Van Oorschot, P.C.; Vanstone, S.A. *Handbook of Applied Cryptography*; CRC Press: Boca Raton, FL, USA, 1996.
26. Dodis, Y.; Katz, J.; Reyzin, L.; Smith, A. Robust fuzzy extractors and authenticated key agreement from close secrets. In *Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 232–250.
27. Dodis, Y.; Reyzin, L.; Smith, A. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 523–540.
28. Dolev, D.; Yao, A. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [CrossRef]
29. Moon, J.; Lee, D.; Lee, Y.; Won, D. Improving biometric-based authentication schemes with smart card revocation/reissue for wireless sensor networks. *Sensors* **2017**, *17*, 940. [CrossRef] [PubMed]
30. Lee, H.; Lee, D.; Moon, J.; Jung, J.; Kang, D.; Kim, H.; Won, D. An improved anonymous authentication scheme for roaming in ubiquitous networks. *PLoS ONE* **2018**, *13*, e0193366. [CrossRef] [PubMed]
31. Wu, F.; Li, X.; Sangaiah, A.K.; Xu, L.; Kumari, S.; Wu, L.; Shen, J. A lightweight and robust two-factor authentication scheme for personalized healthcare systems using wireless medical sensor networks. *Future Gener. Comput. Syst.* **2018**, *82*, 727–737. [CrossRef]
32. Blanchet, B.; Smyth, B.; Cheval, V.; Sylvestre, M. ProVerif 2.00: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial. 2018; pp. 5–16. Available online: https://prosecco.gforge.inria.fr/personal/bblanche/proverif/manual.pdf (accessed on 12 December 2020).
33. Das, A.K. A secure and robust temporal credential-based three-factor user authentication scheme for wireless sensor networks. *Peer-to-Peer Netw. Appl.* **2016**, *9*, 223–244. [CrossRef]
34. Xu, L.; Wu, F. Cryptanalysis and improvement of a user authentication scheme preserving uniqueness and anonymity for connected health care. *J. Med. Syst.* **2015**, *39*, 10. [CrossRef]