Computers and Electrical Engineering 99 (2022) 107803

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Slight: A lightweight authentication scheme for smart healthcare services☆

Haleh Amintoosi [a], Mahdi Nikooghadam [a], Mohammad Shojafar [b], Saru Kumari [c,*], Mamoun Alazab [d]

[a] Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran
[b] ICS/6GIC, University of Surrey, Guildford, GU27XH, United Kingdom
[c] Department of Mathematics, Chaudhary Charan Singh University, Meerut, 250004, India
[d] Charles Darwin, Casuarina, Northern Territory, 0810, Australia

ARTICLE INFO

Keywords:
Critical Infrastructure
Internet of Medical Things
Authentication
Key agreement
Lightweight

ABSTRACT

Critical infrastructures (CIs) include the vital resources for the country's economic and health systems and should be kept secure. We face improvements in the Internet of Things which brings benefits and, at the same time, dependency for CIs. Internet of Medical Things (IoMT) is among the CI sectors that gather health-related information from patients via sensors and provide healthcare services accordingly. However, research has highlighted that this large-scale system opens the door to the patients' private data disclosure. Recent work has concentrated on proposing authentication schemes to address this challenge. Motivated by this, in this paper, we introduce a secure and lightweight authentication and key agreement model named Slight. We informally prove Slight's security and robustness against attacks and formally by using the Scyther tool. We analyze Slight's performance to show it causes minimal computational overhead (0.0076 ms) and comparable communication overhead (1632 bits), making it suitable for IoMT.

1. Introduction

With the recent development in the Internet of Things (IoT), we are now witnessing a significant shift towards a world that is more and more connected. The number of devices that will be connected to the Internet in 2025 is expected to be 75 billion. Such connectivity, while bringing potential benefits such as assisted decision-making and efficient data delivery, makes us more the subject of security threats.

The increasing dependency of Industrial Control Systems (ICS) on IoT connections brings greater security risks to the Critical Infrastructures (CIs) such as energy plants, healthcare services, water systems, and transportation. Failure to provide security toward ICS may cause loss of human lives, information disclosure, denial of service, and physical disruption. Hence, Critical Infrastructure Protection (CIP) has now become a hot research topic in industrial environments.

Critical infrastructures in the healthcare sector have gained popularity, as we face the rapid increase in life expectancy in the world, which has resulted in the rise in the number of older people. Based on the United Nations' reports, by the year 2050, the population of the old generation will be about 2 billion, mostly living on their own, suffering from continual diseases that may

☆ This paper is for special section VSI-cei. Reviews were processed by Guest Editor Dr. Imran Razzak and recommended for publication.

* Corresponding author.
E-mail addresses: amintoosi@um.ac.ir (H. Amintoosi), mahdi.nikooghadam@mail.um.ac.ir (M. Nikooghadam), m.shojafar@surrey.ac.uk (M. Shojafar), saryusiirohi@gmail.com (S. Kumari), alazab.m@ieee.org (M. Alazab).

https://doi.org/10.1016/j.compeleceng.2022.107803
Received 7 March 2021; Received in revised form 1 February 2022; Accepted 7 February 2022
Available online 24 February 2022
0045-7906/© 2022 Elsevier Ltd. All rights reserved.

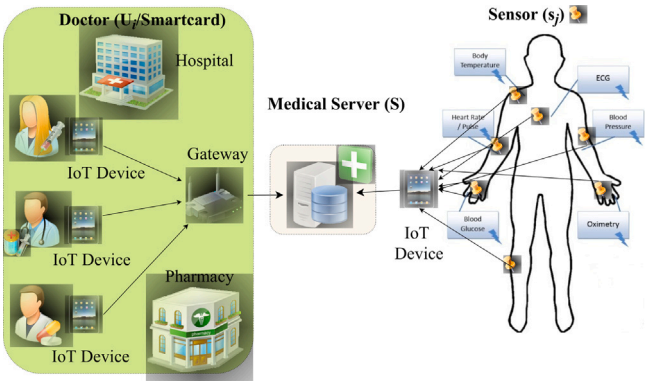

Fig. 1. Slight Architecture.

threaten their lonely life. So, it is vital to provide accurate and efficient health-related services for the old and elderly population. To this end, the Internet of Medical Things (IoMT) [1] was emerged to support patients with health-related services in online mode, e.g., virtual consultations provided by specialists and medical staff. In such systems, the medical information of patients such as heart rate is gathered by smart wearable body sensors, and communicated through the Internet to be stored in medical servers residing in the cloud. Doctors and other medical staff can then access these data to monitor the patient's health condition, diagnose potential disease, and send remote medical advice.

Motivation

Preserving the privacy and security of patients' health-related data is a vital challenge in remote healthcare systems. In IoMT, medical information transfer is normally done via a wireless medium, so an adversary can eavesdrop on the communication channel and acquire health-related information, leading to a breach in the patient's privacy. The attacker may also inject falsified data into the system, resulting in false diagnosis or injury. IoT devices also suffer from hacks. A recent report published by F-Secure[1] shows that 82% of IoT devices have been exposed to cyberattacks in the first half of 2019. Hence, security is considered to have high priority in smart healthcare applications [2], and becomes even more challenging in pandemic times such as the Spread of COVID-19 virus, due to the rapid adoption of IoMT in the health ecosystem [3]. On the other hand, IoMT consists of resource-constrained sensor devices, which cannot perform complex cryptographic operations. This demonstrates the significance of providing a balance between secure communication and resource utilization. Research has been done on giving secure authentication schemes for remote healthcare services, including IoMT.

Contribution

This paper proposes a lightweight authentication and key agreement model for IoMT smart healthcare applications as a case study of CI named *Slight*. Slight has three components, as depicted in Fig. 1. The patient (or sensor) is equipped with wearable sensor devices which gather health-related data. The medical server stores medical data collected by sensors and transmitted via a mobile device or tablet as a gateway. The user who can be a doctor, a nurse, or any other medical staff, obtains authenticated access to the stored data and provides the patient with healthcare advice. We prove Slight's robustness against famous attacks, e.g., replay attacks and impersonation attacks, and its capability to provide security requirements, namely, perfect forward secrecy. We also review the authentication scheme presented by Aghili et al. [4], called LACO, and prove that their scheme is prone to malicious sensor attacks and server impersonation attacks. Moreover, it is unable to provide perfect forward secrecy.

Specifically, our contributions are listed as follows:

- We propose a lightweight authentication scheme for smart healthcare applications in IoMT that we call *Slight*, which has three entities: the medical staff (such as doctors), the medical server, and the patients (or sensors). The proposed scheme enables the medical staff and patients to register to the medical server, authenticate themselves, and communicate with each other through the server in a secure manner. The scheme also enables the doctor to transfer his access privileges to another doctor. Slight also provides password update for the medical staff.
- We proved the security of Slight formally via the Scyther tool [5]. Moreover, we informally justify the scheme's strength in terms of resistance to known attacks and provision of perfect forward secrecy and known-key secrecy.
- We show that Slight achieves the minimum computational complexity (i.e., 0.0076 ms), and acceptable communication overhead, compared to related models. This makes Slight well-suited for IoMT with computationally constrained sensor devices.
- We also perform cryptanalysis of Aghili et al.'s scheme [4] (i.e., LACO), and demonstrate that it is insecure against malicious sensor attacks and server impersonation attacks.

[1] https://blog.f-secure.com/attack-landscape-h1-2019-iot-smb-traffic-abound/.

In the following, the related work is discussed in Section 2. Aghili et al.'s scheme [4] description and security analysis are presented in Section 3 and Section 4, respectively. The details of the proposed model are expressed in Section 5. Section 6 analyzes the security of the proposed scheme. We assess the performance in Section 7. The conclusion comes in Section 9.

2. Related work

We review the related articles that have addressed the authentication challenge in health-related applications. In 2014, Mishra et al. [6] presented a biometric-based authentication protocol for telecare medical information systems. Jiang et al. [7] studied the three-factor scheme of Lu et al. [8] and showed that it is prone to a series of attacks, including tracking attack, offline identity guessing attack, user/server impersonation attack, and offline password guessing attack. Thus, to overcome these drawbacks, they proposed a new three-factor scheme. Zhang et al. [9] also reviewed the method presented by Mishra et al. [6] and showed that it is insecure against various attacks and cannot guarantee perfect forward secrecy. In 2019, Ostad-sharif et al. [10] presented a scheme for TMIS and demonstrated that it could solve the existing security drawback incurring low communication and computational overheads. However, authors in [11] explained that [10] is prone to key compromise password guessing attacks and key compromise impersonation attacks. Khatoon et al. [12] designed an Elliptic Curve Cryptosystem (ECC)-based authentication model for TMIS and expressed that their scheme is secure and provides essential security requirements. In 2018, Aneesh and Deepthi [13] reviewed Li et al.'s proposed scheme for Wireless Body Area Networks (WBAN) [14] and highlighted some of its problems, including insecurity against sensor node impersonation attacks. They also presented a hybrid authentication and key agreement scheme using the physiological signal. Marko et al. [2] also reviewed Li et al.'s scheme [14] and then proposed a new scheme that addresses its drawbacks.

Liu et al. [15] in 2019 designed an authentication scheme for WBAN. Amel et al. [16] proposed an anonymous context-aware authentication scheme for WBANs.

In 2020, Masud et al. [3] proposed a lightweight key establishment scheme for IoMT. Their scheme makes use of Physical Unclonable Functions (PUF) to provide the network devices with the means to verify the doctor's/sensor node's legitimacy before agreeing on a shared session key. Their scheme was shown to incur low computation and communication overhead, compared to the related schemes. Wong et al. [17] also proposed a three-factor authentication method for remote healthcare systems in 5G in 2020. Their scheme is aimed at providing fast authentication which can be applied to multi-server environments. In the same year, Nikooghadam et al. [18] designed a lightweight authentication and key agreement scheme for e-health systems in cloudIoT. In their scheme, three entities: sensor, gateway, and the user communicate with each other in order to exchange health-related data over an insecure channel. Their scheme was shown to obtain minimal computation overhead and resist major security threats.

In 2019, Aghili et al. [4] reviewed the scheme presented by Zhang et al. [19] that it is vulnerable against insider attacks, denial of service attacks, and de-synchronization attacks. They then proposed a new authentication and key agreement scheme, called LACO, which has the user group (doctors, nurses, etc.), the medical server, and the patient group (i.e., the sensors) components. They showed the security of their scheme formally, and informally and claimed that it is secure against existing attacks. However, in this article, we prove in Section 4 that LACO is not secure against malicious sensor attacks and server impersonation attacks. A malicious sensor node can communicate directly with the user without the server being notified. The adversary can also impersonate the server and establish a session key with the user directly.

As can be inferred from the above discussion, there still is a strong demand for a secure and lightweight authentication and key agreement scheme that addresses the drawbacks mentioned above and is secure against various attacks. We claim that Slight can satisfy such a requirement.

3. Review of LACO scheme

We review the LACO protocol [4]. Notations are depicted in Table 1.

3.1. Setup phase

In the setup phase, $M_j = h(PID_j||s)$ is computed for the sensor j_{th} by the medical server S, in which, PID_j and s are the sensor's identity and the server's master key, respectively. Finally, M_j is stored in the sensor's memory.

3.2. Registration phase

During this step, the smart card is issued by the medical server S for the user U_i, as follows.

- Step 1. U_i chooses an identity ID_i and transfers it to the S via the message Msg_1.
- Step 2. S checks the uniqueness of ID_i in its database. If so, the medical server computes $X_{1i} = h(IDS_i||ID_i||r_s)$ and $Y_{1i} = h(X_{1i}||s)$ where r_s is a random number, s is its master key and IDS_i is the smartcard identity. Then, S computes $HACO_j$ and $Z_{1j} = h(X_{1i}||Y_{1i}) \oplus HACO_j$ and stores values of X_{1i}, Z_{1j} in its database.
  At the end, $\langle X_{1i}, Y_{1i}, Z_{1j}, h_{Bio}(.)\rangle$ is saved on the smartcard and is given to U_i.
- Step 3. The user inserts ID_i and the password PW_i, extracts his biometric data B_i and computes $A_{1i} = h_{Bio}(B_i) \oplus h(PW_i||ID_i)$ and $B1_i = Y_{1_i} \oplus h(ID_i||PW_i||h_{Bio}(B_i))$. It then sets the flag=0, deletes Y_{1i} from the smartcard and adds $\langle A_{1i}, B_{1i}, flag\rangle$ to it.
  Finally, the smartcard contains:
  $\langle A_{1i}, B_{1i}, flag, X_{1i}, Y_{1i}, Z_{1j}, h_{Bio}(.)\rangle$. Fig. 2 depicts this step.

Fig. 2 depicts LACO's registration phase.

Table 1
Notations of LACO.

Notation	Description
S	The medical server
U_i	The i_{th} user (doctor)
ID_i	i_{th} user's identity
PID_j	j_{th} sensor's identity
IDS_i	Identity of the smartcard delivered to i_{th} user
PW_i	i_{th} user's password
B_i	i_{th} user's biometric parameters
r_x and k_x	The random numbers
T_x	The current time stamp
s	Medical server's master key
SK_u, SK_s, SK_p	The session keys of doctor, server, and sensor
$HACO_j$	The hash of the access control string
$h(.)$	A one-way hash function
$h_{Bio}(.)$	Biohash function
$\oplus$	XOR operation
$\|\|$	Concatenation operation

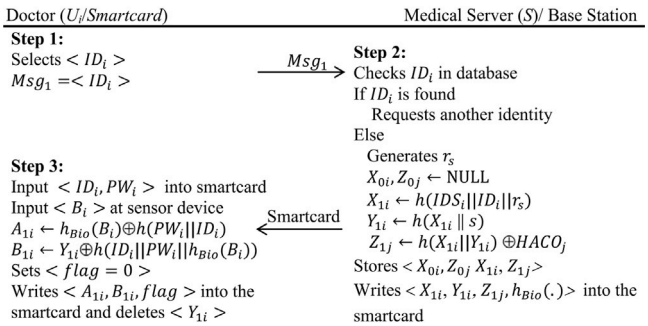

Fig. 2. Registration phase of LACO.

3.3. Login phase

To access the medical server's data, the user inserts his smartcard, enters his ID_i' and PW_i', and also extracts his biometric information B_i' via the sensor device. The smartcard then calculates $A_{ni}' = h_{Bio}(B_i') \oplus h(PW_i'||ID_i')$. It then compares the computed A_{ni}' with A_{ni} stored in the smartcard. If $A_{ni}' \neq A_{ni}$, the smart card is rejected. Otherwise, random numbers K_u and r_i, and a timestamp T_1 are generated. U_i then computes messages $C_1 = K_u \oplus h(X_{ni}||Y_{ni}'||T_1), C_2 = PID_j \oplus h(X_{ni}||Y_{ni}'||Z_{nj}||T_1)$. In this equation, PID_j is the identity of the sensor that the doctor intends to access, and $Y_{ni}' = B_{ni}' \oplus h(ID_i'||PW_i'||h_{Bio}(B_i'))$. Then, the smartcard checks the flag and if zero, it means that the previous session has reached to the end. So, it calculates $C_3 = X_{ni}||Z_{nj}$ and sets the flag = 1. Otherwise, it means that the session has not terminated and the has not done the update. So, it computes $C_3 = h(r_j||X_{ni}||Y_{ni}')||h(r_i||Y_{ni}'||Z_{nj})$ and $C_4 = h(C_1||C_2||C_3||K_u||PID_j||T_1||r_i)$ and sends the message Msg_2 as $\langle C_1, C_2, C_3, C_4, r_1, T_1 \rangle$, to S on an insecure channel.

Fig. 3 shows LACO's login and authentication phase.

3.4. Authentication

In this step, the user (e.g., the doctor) performs a set of steps to prove his authenticity to the medical server. Once being authenticated, he shares a session key with other entities. Steps are as follows:

- Step 1. AS mentioned above at the last step of the login phase, Msg_2 is sent to the server S. Once received, its freshness is checked via comparing whether $|T_2 - T_1| <?\Delta T$. If satisfied, in order to be sure that X_{ni} and Z_{nj} are valid, for each tuple of $\langle X_{ni} = (X_{0i}, X_{1i}), Z_{nj} = (Z_{0j}, Z_{1j}) \rangle$ in its database it computes $Y_{ni}' = h(X_{ni}||s)$ and checks whether
  (i) $C_3 = h(r_j||X_{ni}||Y_{ni}')||h(r_j Y_{ni}'||Z_{nj})$
  (ii) $C_3 \oplus (X_{ni}||Z_{nj}) = 0$.
  if not equal, the session is rejected. Otherwise, S computes
  (i) $K_u' = C_1 \oplus h(X_{ni}||Y_{ni}'||T_1)$
  (ii) $PID_j' = C_2 \oplus h(X_{ni}||Y_{ni}'||Z_{nj}||T_1)$
  (iii) $C_4' = h(C_1||C_2||X_{ni}||Z_{nj}||K_u'||PID_j'||T_1||r_i)$.
  Finally, S compares C_4' with the received C_4 and authenticates U_i if equal. Next, S gets the user's access control string $HACO_j = Z_{nj} \oplus h(X_{ni}||Y_{ni})$ and if valid, it infers that U_i can communicate with PID_j. At last, S computes

Doctor (U_i/*Smartcard*)	Medical Server (S)/Base Station	Patient /Sensor Nodes (P_j)
	Login phase	

U_i inserts smartcard into terminal
Inputs <ID_i', PW_i'> and B_i'
Computes
$A_{ni}' \leftarrow h_{Bio}(B_i') \oplus h(PW_i' \| ID_i')$
If $A_{ni}' \neq A_{ni}$ then <Reject>
Else
Generates K_u, T_1, r_i
$Y_{ni}' \leftarrow B_{ni}' \oplus h(ID_i' \| PW_i' \| h_{Bio}(B_i'))$
$C_1 \leftarrow K_u \oplus h(X_{ni} \| Y_{ni}' \| T_1)$
$C_2 \leftarrow PID_j \oplus h(X_{ni} \| Y_{ni}' \| Z_{nj} \| T_1)$
If $flag = 0$ then
$C_3 \leftarrow X_{ni} \| Z_{nj}$
Sets $< flag = 1 >$
Else
$C_3 \leftarrow h(r_i \| X_{ni} \| Y_{ni}') \| h(r_i \| Y_{ni}' \| Z_{nj})$
$C_4 \leftarrow h(C_1 \| C_2 \| C_3 \| K_u \| PID_j \| T_1 \| r_i)$
$Msg_2 =< C_1, C_2, C_3, C_4, r_i, T_1 >$
$\xrightarrow{Msg_2}$		

	Authentication phase	

	Step 1:
	If $\|T_1 - T_2\| > \Delta T$ then <Reject>
	Else
	For each tuple $< X_{ni}, Z_{nj} >$ in database
	$Y_{ni}' \leftarrow h(X_{ni} \| s)$
	If $C_3 \neq h(r_i \| X_{ni} \| Y_{ni}') \| h(r_i \| Y_{ni}' \| Z_{nj})$ then
	If $C_3 \oplus (X_{ni} \| Z_{nj}) \neq 0$ then <Reject>
	Else
	$K_u' \leftarrow C_1 \oplus h(X_{ni} \| Y_{ni}' \| T_1)$
	$PID_j' \leftarrow C_2 \oplus h(X_{ni} \| Y_{ni}' \| Z_{nj} \| T_1)$
	$C_4' \leftarrow h(C_1 \| C_2 \| C_3 \| K_u' \| PID_j' \| T_1 \| r_i)$
	If $C_4' \neq C_4$ then <Reject>
	Else <U_i is authenticated >
	$HACO_j \leftarrow Z_{nj} \oplus h(X_{ni} \| Y_{ni}')$
	$M_j' \leftarrow h(PID_j' \| s)$,
	$C_5 \leftarrow HACO_j \oplus h(M_j' \| T_2)$
	$C_6 \leftarrow K_u' \oplus HACO_j$
	$C_7 \leftarrow h(HACO_j \| M_j' \| K_u' \| T_2)$
	$Msg_3 =< C_5, C_6, C_7, T_2 >$
	$\xrightarrow{Msg_3}$
		Step 2:
		If $\|T_2 - T_3\| > \Delta T$ then <Reject>
		Else
		$HACO_j' \leftarrow C_5 \oplus h(M_j \| T_2)$
		$K_u' \leftarrow C_6 \oplus HACO_j$
		$C_7' \leftarrow h(HACO_j \| M_j \| K_u' \| T_2)$
		If $C_7' \neq C_7$ then <Reject>
		Else <U_i is authenticated >
		Generates K_p
		$SK_p \leftarrow h(HACO_j \| PID_j \| K_u' \| K_p)$
		$C_8 \leftarrow h(SK_p \| M_j \| T_3)$
		$C_9 \leftarrow K_u' \oplus K_p$
		$Msg_4 =< C_8, C_9, T_3 >$

Step 4: If $\|T_4 - T_5\| > \Delta T$ then <Reject> Else $K_p' \leftarrow C_9 \oplus K_u$ $SK_u \leftarrow h(HACO_i \| PID_j \| K_u \| K_p')$ $C_{10}' \leftarrow h(SK_u \| K_u \| K_p' \| T_4)$ If $C_{10}' \neq C_{10}$ then <Reject> Else Sets $< flag = 0 >$ Updates $X_{(n+1)i} \leftarrow h(h(r_i \| X_{ni}) \oplus r_i \oplus Y_{ni})$ $Z_{(n+1)j} \leftarrow h(Y_{ni} \| X_{ni}) \oplus HACO_j$ $\xleftarrow{Msg_5}$	**Step 3:** If $\|T_3 - T_4\| > \Delta T$ then <Reject> Else $K_p' \leftarrow C_9 \oplus K_u'$ $SK_s \leftarrow h(HACO_j \| PID_j' \| K_u' \| K_p')$ $C_8' \leftarrow h(SK_s \| M_j' \| T_3)$ If $C_8' \neq C_8$ then <Reject> Else $C_{10} \leftarrow h(SK_s \| K_u' \| K_p' \| T_4)$ Updates $X_{(n+1)i} \leftarrow h(h(r_i \| X_{ni}) \oplus r_i \oplus Y_{ni}')$ $Z_{(n+1)j} \leftarrow h(Y_{ni}' \| X_{ni}) \oplus HACO_j$ $Msg_5 =< C_9, C_{10}, T_4 >$	$\xleftarrow{Msg_4}$

Fig. 3. Login and Authentication phase of LACO.

(i) $M_j' = h(PID_j' \| s)$
(ii) $C_5 = HACO_j \oplus h(M_j' \| T_2)$
(iii) $C_6 = K_u' \oplus HACO_j$
(iv) $C_7 = h(HACO_j \| M_j' \| K_u' \| T_2)$

S then sends Msg_3 as $\langle C_5, C_6, C_7, T_2 \rangle$, to the sensor node P_j.

- Step 2. Upon receiving Msg_3, P_j first checks the timestamp T_2 and if within the valid range, it obtains $HACO_j' = C_5 \oplus h(M_j \| T_2)$. It then extracts $K_u' = C_6 \oplus HACO_j$ and computes $C_7' = h(HACO_j' \| M_j \| K_u' \| T_2)$. If $C_7' = C_7$, P_j authenticates U_i and calculates

the session key $SK_p = h(HACO'_j||PID_j||K'_u||K_p)$ where K_p is the random number. It also calculates $C_8 = h(SK_p||M_j||T_3)$ and $C_9 = K'_u \oplus K_p$. After that, the response Msg_4 as $\langle C_8, C_9, T_3\rangle$ is sent to S.

- Step 3. Once Msg_4 is received, S extracts $K'_p = C_9 \oplus K'_u$ and the session key $SK_s = h(HACO_j||PID'_j||K'_u||K'_p)$. Then, it verifies the message validity by comparing C_8 with $h(SK_s||M'_j||T_3)$. If equal, it accepts the session key, and the session is aborted if otherwise. It then calculates
  (i) $C_{10} = h(SK_s||K'_u||K'_p||T_4)$
  (ii) $X_{(n+1)i} = h(h(r_i||X_{ni}) \oplus r_i \oplus Y'_{ni})$
  (iii) $Z_{(n+1)j} = h(Y'_{ni}||X_{ni}) \oplus HACO_j$
  Finally, the message $Msg_5 = \langle C_9, C_{10}, T_4\rangle$ is forwarded to U_i.
- Step 4. Once the message Msg_5 is received, U_i first checks its freshness of T_4 and if so, he gets $K'_p = C_9 \oplus K_u$ and the session key $SK_u = h(HACO_j||PID_j||K_u||K'_p)$ and computes $C'_{10} = h(SK_u||K_u||K'_p||T_4)$. Then, it checks whether C_{10} received is equal to C'_{10}. If so, the flag is set to 0 and $X_{(n+1)i}$ is updated as $X_{(n+1)i} = h(h(r_i||X_{ni}) \oplus r_i \oplus Y_{ni})$ and $Z_{(n+1)j} = h(Y_{ni}||X_{ni}) \oplus HACO_j$ and rewritten them into the smartcard. If not, the connection ends up.

4. Cryptanalysis of LACO

In this section, we show LACO's weakness in providing perfect forward secrecy and vulnerability to server impersonation attacks.

4.1. Perfect forward secrecy

Perfect forward secrecy provision ensures the safety of the session keys in case of longterms (e.g., IDs or secret keys) are compromised. In the following, we show LACO's weakness in perfect forward secrecy provisioning.

- Step 1. As mentioned above, we assume that the adversary possesses the medical server's secret key s and PID_j (i.e., the identity of j_{th} sensor). So, he can calculate M_j as $M_j = h(PID_j||s)$. Moreover, since $Msg_3 = \langle C_5, C_6, C_7, T_2\rangle$ is exchanged publicly, the attacker has access to its parameters including T_2 and C_5. Having M_j, C_5 and T_2, the adversary is able to calculate $HACO_j$ as $HACO'_j = C_5 \oplus h(M_j||T_2)$.
- Step 2. Similar to the above, C_6 is also available to the adversary since it is included in Msg_3 which is being exchanged on a public channel. Having C_6 and $HACO_j$, the adversary is now able to calculate K'_u, since as mentioned in the second step of authentication, $K'_u = C_6 \oplus HACO_j$.
- Step 3. As shown in step 2 of the authentication, $Msg_4 = \langle C_8, C_9, T_3\rangle$ is sent from the patient to the server on a public channel. So, its parameters including C_9 are available to the adversary. Having C_9 and K'_u from the previous step, the adversary is now able to calculate K_p as $C_9 = K'_u \oplus K_p$. due to the reversibility of $\oplus$.
- Step 4. Having $HACO'_j$ (from step 1), PID_j (from the assumption), K'_u (computed in step 2) and K_p (computed in step 3), the adversary can calculate SK_p as $SK_p = h(HACO'_j||PID_j||K'_u||K_p)$.

4.2. Malicious sensor attack

Assume that a registered patient (i.e., sensor) plans to communicate maliciously with the doctor and establish a session key without the intervention of the server. In the following, we demonstrate that the adversary can bypass the server and calculate the session key with the doctor.

- Step 1. As the adversary is a pre-registered patient (i.e., a sensor device), the server has stored M_j into the memory of the sensor device, as mentioned in the setup phase.
- Step 2. Since $Msg_3 = \langle C_5, C_6, C_7, T_2\rangle$ is exchanged publicly, the adversary can access and falsify it. Assume that the adversary selects his own parameters T_2^*, C_5^* and C_6^* instead of T_2, C_5 and C_6, respectively. As mentioned above, the adversary has a valid M_j. So, he is able to calculate $HACO_j^* = C_5^* \oplus h(M_j||T_2^*)$ and $K_u^* = C_6^* \oplus HACO_j^*$.
- Step 3. Now that the adversary has M_j, $HACO_j^*$, K_u^* and T_2^*, he can calculate $C_7^* = h(HACO_j^*||M_j||K_u^*||T_2^*)$ in a way that $C_7^* = C_7$ and hence, he will be authenticated as a legal server.
- Step 4. Having K_u^* from the previous step and C_9 on a public channel, the adversary can calculate K_p^* as $K_p^* = C_9 \oplus K_u^*$.
- Step 5. Now, the adversary computes
  $SK_p^* = h(HACO_j^*||PID_j^*||K_u^*||K_p^*)$.
- Step 6. Since $Msg_5 = \langle C_9, C_{10}, T_4\rangle$ is transferred publicly, the adversary can access and falsify it. Assume that the adversary selects his own parameters T_4^* and C_9, instead of T_4 and C_9, respectively. As mentioned above in step 4, the adversary can calculate SK_p^*, which is equal to SK_s.
- Step 7. Also, as mentioned in step 2 and 3, he can calculate K_u^* and K_p^*.
- Step 8. Having SK_s from step 6, he can now calculate $C_{10}^* = h(SK_s||K_u^*||K_p^*||T_4^*)$ in a way that $C_{10}^* = C_{10}$. In other words, the adversary (i.e., the malicious patient) establishes a communication with the doctor and agrees on a session key, without the server being notified.

Table 2
Notations of the proposed scheme.

Notation	Description
S	The medical server
U_i	The i_{th} user (doctor)
ID_i	i_{th} user's identity
IDS_j	j_{th} sensor's identity
X_j	j_{th} sensor's secret key
SID	Server's identity
PW_i	i_{th} user's password
d_i and q_p and T_p	Random numbers
T_1, T_2, T_3, T_4, T_5	Current time stamp
s	Medical server's master key
sk_u	Session key of doctor node
sk_u, sk_s, sk_p	Session key of server node
sk_p	Session key of sensor node
$h(.)$	A one-way hash function
$\oplus$	XOR operation
$\|\|$	Concatenation operation

4.3. Server impersonation attack

In the following, we demonstrate that the adversary can impersonate the server for the doctor.

- Step 1. Assume that the sensor's memory is not tamper-proof (as nothing related to this matter has been mentioned in LACO). Hence, the adversary can access the sensor's memory and obtain M_j and PID_j.
- The rest of steps are similar to steps 2 to 8 of the malicious sensor attack, as mentioned above. So, we do not repeat them to save space.

5. Slight: The proposed scheme

We propose a Slight, lightweight, and secure authentication protocol for smart health systems, to address the security vulnerabilities of related schemes including LACO [4]. Fig. 1 shows the architecture of the proposed framework. As explained before in Section 1, patients (also known as sensors, equipped with wearable IoT sensors, first register to the medical server S and then submit their health-related private data stored in S. Doctors or other medical staff (known as users), who intend to access the patient's data, also register to the medical server. Once registered to the server, users and sensors mutually authenticate each other via the medical server. Upon authentication is performed, parties agree on a session key that will be used for further secure communication between them. Doctors are also able to transfer their data ownership to another if required.

Slight includes registration phase, login and authentication phase, ownership transfer phase, and password update phase. The notations have been demonstrated in Table 2.

5.1. Doctor registration phase

The doctor U_i registers to the medical server and requests a smartcard for further login. i demonstrates the number of registered doctors. It is set to 0 at first and incremented once a new doctor registers to the server. Steps are shown in Fig. 4 and are as below:

- Step 1. The user U_i selects a userID ID_i, a password pw_i and a random number a_i, calculates $p_i = h(ID_i||pw_i||a_i)$ and transmits $\langle p_i, ID_i\rangle$ securely to the medical server.
- Step 2. Upon receiving the registration request, the server first searches the database to see whether another user with the same ID_i has previously registered or not. If so, the server asks for a new identity. Otherwise, it selects a random number b_i and computes $bID_i = h(b_i||ID_i)$, $M_i = h(b_i||ID_i||bID_i)$, $N_i = h(SID||S||b_i)$, $A_i = N_i \oplus p_i$ and $B_i = h(M_i||N_i)$. At the end, the server stores $\langle b_i, A_i, B_i\rangle$ on the smartcard and $\langle A_i, M_i, B_i, ID_i\rangle$ to the database (which resides in the tamper-proof memory of the server). It then increments i by one, showing that one new doctor has been registered to the server. The server then securely sends the smartcard to the doctor.
- Step 3. The user adds the random number a_i to the smart card. At the end, the smartcard has the values $\langle b_i, A_i, B_i, a_i\rangle$ stored on it.

5.2. Sensor registration phase

The patient (sensor P_j) registers to the medical server. j shows the number of registered sensors. It is set to 0 at first, and incremented once a new sensor (patient) registers to the server, as shown in Fig. 5 and expressed below.

Registration Phase

Doctor(U_i/Smartcard)		**Medical Server(S)/Base Station**
Selects random number a_i		
Chooses ID_i, pw_i		
$p_i = h(ID_i \| pw_i \| a_i)$		
	$\xrightarrow[\text{(Secure Channel)}]{\langle p_i, ID_i \rangle}$	
		Searched ID_i in database
		If ID_i exists, requests a new identity
		else
		Selects random number b_i
		$bID_i = h(b_i \| ID_i)$
		$M_i = h(b_i \| ID_i \| bID_i)$
		$N_i = h(SID \| S \| b_i)$
		$A_i = N_i \oplus p_i$
		$B_i = h(M_i \| N_i)$
		stores (b_i, A_i, B_i) on smartcard
		Stores (A_i, M_i, B_i, ID_i) to database
		$i = i + 1$
	$\xleftarrow[\text{(Secure Channel)}]{\text{smartcard}}$	
Adds a_i to smartcard		
Smartcard $[b_i, A_i, B_i, a_i]$		

Fig. 4. Registration phase of the proposed scheme (User to the server).

Registration Phase

Sensor		**Medical Server(S)/Base Station**
Selects a random number f_j		
$G_j = h(IDS_j \| X_j \| f_j)$		
	$\xrightarrow[\text{(Secure Channel)}]{\langle G_j, f_j, IDS_j \rangle}$	
		$w_j = h(IDS_j \| s \| f_j)$
		Stores G_j, IDS_j, f_j
		$j = j + 1$
	$\xleftarrow[\text{(Secure Channel)}]{\langle w_j \rangle}$	
Stores w_j, f_j in its tamper-proof memory		

Fig. 5. Registration phase of the proposed scheme (Sensor to the server).

- Step 1. At first, the sensor P_j selects a random number f_j and then calculates $G_j = h(IDS_j || X_j || f_j)$ where IDS_j and X_j are the identity and the secret key of the sensor, respectively. It then securely sends $\langle G_j, f_j, IDS_j \rangle$ to the medical server.
- Step 2. S calculates $w_j = h(IDS_j || s || f_j)$ (s: the server's master key). It then stores $\langle G_j, IDS_j, f_j \rangle$ to its database. Note that we assume that the server's memory, where the database is stored, is tamper-proof. j is also incremented by one to show the registration of one new sensor to the server. At the end, the server securely transmits w_j to the sensor.
- Step 3. The sensor stores $\langle w_j, f_j \rangle$ in its tamper-proof memory.

Login and Authentication Phase

Doctor (U_i/**Smartcard**)	**Medical Server(S)**	**Sensor** s_j
Inserts smart card		
Inputs ID_i^*, pw_i^*		
$p_i^* = h(ID_i^* \| pw_i^* \| a_i)$		
$N_i^* = A_i \oplus p_i^*$		
$bID_i^* = h(b_i \| ID_i^*)$		
$M_i^* = h(b_i \| ID_i^* \| bID_i^*)$		
$B_i^* = h(M_i^* \| N_i^* \| p_i^*)$		
$B_i = ?B_i^*$		
Selects random number d_i,	Selects time stamp T_2	
Selects time stamp T_1	$\|T_2 - T_1\| <?\Delta T$	
$M1_i = h(M_i \| N_i \| T_1) \oplus d_i$	$N_i = h(SID \| s \| b_i)$	
$M2_i = h(M_i \| N_i \| d_i)$	$d_i = h(M_i \| N_i \| T_1) \oplus M1_i$	
$\xrightarrow{\langle M1_i, T_1, b_i, M2_i \rangle}$	$M2_i^* = h(M_i \| N_i \| d_i)$	Selects time stamp T_3
	$M2_i^* = ?M2_i$	$\|T_3 - T_2\| <?\Delta T$
	Selects random number q_p	$q_p = z_j \oplus (d_i \| IDS_j \| f_j)$
	$z_j = h(d_i \| IDS_j \| f_j) \oplus q_p$	Selects random number T_p
	$w_j = h(IDS_j \| s \| f_j)$	$G_j^* = h(IDS_j \| X_j \| f_j)$
	$Y_j = h(w_j \| G_j \| q_p)$	$Y_j^* = h(W_j \| G_j^* \| q_p)$
	$\xrightarrow{\langle d_i, z_j, Y_j, T_2 \rangle}$	$Y_j^* = ?Y_j$
		$sk_p = h(q_p \| IDS_j \| T_P)$
		$Q_j = h(G_j \| f_j \| w_j) \oplus T_p$
		$Auth_j = h(sk_p \| w_j \| T_3)$
		$\xleftarrow{\langle Q_j, T_3, Auth_j \rangle}$
	Select $T_4, \|T_4 - T_3\| <?\Delta T$	
	$p_i = N_i \oplus A_i$	
Selects time stamp T_5	$T_p = h(G_j \| f_j \| w_j) \oplus Q_j$	
$\|T_5 - T_4\| <?\Delta T$	$sk_s = h(q_p \| IDS_j \| T_p)$	
$M3_i^* = h(u_i \| v_i \| p_i \| T_4)$	$Auth_s = h(sk_s \| w_j \| T_3)$	
$M3_i^* = M3_i$	$Auth_s = ?Auth_j$	
$T_p = p_i \oplus u_i$	$u_i = p_i \oplus T_p, v_i = p_i \oplus q_p$	
$q_p = p_i \oplus v_i$	$M3_i = h(u_i \| v_i \| p_i \| T_4)$	
$sk_u = h(q_p \| IDS_j \| T_p)$	$\xleftarrow{\langle T_4, u_i, v_i, M3_i \rangle}$	

Fig. 6. Authentication phase of the proposed scheme.

5.3. Authentication phase

To access the medical server's data, the user U_i inserts his smartcard. Steps to authenticate the user, as demonstrated in Fig. 6 are as below:

- Step 1. The user U_i inserts his identity and password ID_i^*, pw_i^*, respectively. Then, the smartcard computes $p_i^* = h(ID_i^* || pw_i^* || a_i)$, $N_i^* = A_i \oplus p_i^*$, $bID_i^* = h(b_i || ID_i^*)$, $M_i^* = h(b_i || ID_i^* || bID_i^*)$ and $B_i^* = h(M_i^* || N_i^* || p_i^*)$. It then checks whether the computed B_i^* equals to B_i stored on the smartcard to make sure that the user is the owner of the smart card and the card has not been stolen. Then, the smartcard selects the random number d_i and time stamp T_1 and calculates $M1_i = h(M_i || N_i || T_1) \oplus d_i$ and $M2_i = h(M_i || N_i || d_i)$. The smartcard then sends $\langle M1_i, T_1, b_i, M2_i \rangle$ to the medical server.
- Step 2. On message receipt, the server chooses the time stamp T_2 and verifies $|T_2 - T_1| <?\Delta T$. If so, the server calculates $N_i = h(SID || s || b_i)$, $d_i = h(M_i || N_i || T_1) \oplus M1_i$ and $M2_i^* = h(M_i || N_i || d_i)$ and checks whether $M2_i^* = ?M2_i$. If so, the user

is authenticated. Next, S selects random number q_p and calculates $z_j = h(d_i||IDS_j||f_j) \oplus q_p$, $w_j = h(IDS_j||s||f_j)$, and $Y_j = h(w_j||G_j||q_p)$. At the end, the server sends $\langle d_i, z_j, Y_j, T_2 \rangle$ to the sensor.

- Step 3. To check the freshness of the received message, the sensor chooses time stamp T_3 and checks $|T_3 - T_2| <?\Delta T$. If it holds, the sensor calculates $q_p = z_j \oplus (d_i||IDS_j||f_j)$. It then selects random number T_p and calculates $G_j^* = h(IDS_j||X_j||f_j)$ and $Y_j^* = h(W_j||G_j^*||q_p)$. The sensor then checks whether Y_j^* equals Y_j received from the server. If so, it proves that the message is from the server and hence, the server is authenticated. It then calculates sk_p as $sk_p = h(q_p||IDS_j||T_P)$.
  Also, $Q_j = h(G_j||f_j||w_j) \oplus T_p$ and $Auth_j = h(sk_p||w_j||T_3)$. At the end, the sensor sends $\langle Q_j, T_3, Auth_j \rangle$ to the server.
- Step 4. To check whether the message is new or not, the server selects time stamp T_4 and verifies $|T_4 - T_3| <?\Delta T$. It then authenticates the message by calculating $p_i = N_i \oplus A_i$, $T_p = h(G_j||f_j||w_j) \oplus Q_j$, $sk_s = h(q_p||IDS_j||T_p)$, and $Auth_s = h(sk_s||w_j||T_3)$ and comparing $Auth_s$ with $Auth_j$ received from the sensor. If they are equal, it is proved that the message has been send by the sensor. The server then calculates $u_i = p_i \oplus T_p$, $v_i = p_i \oplus q_p$ and $M3_i = h(u_i||v_i||p_i||T_4)$ and sends $\langle T_4, u_i, v_i, M3_i \rangle$ to the user.
- Step 5. The user, upon receiving the message from the server, first checks its freshness through selecting T_5 and checking whether $|T_5 - T_4| <?\Delta T$. If so, it calculates $M3_i^* = h(u_i||v_i||p_i||T_4)$ and checks if it is the same as $M3_i$ received from the server. If so, the message is proved to be sent by the server. The user then calculates $T_p = p_i \oplus u_i$ and $q_p = p_i \oplus v_i$ and then, calculates the session key sk_u as $sk_u = h(q_p||IDS_j||T_p)$.

5.4. Ownership transfer phase

In this phase, a doctor who has access to the data of a specific patient (i.e., j_{th} sensor), passes his access rights to another doctor. At the end of the phase, access permissions of the first doctor U_1 to j_{th} sensor is revoked and transferred to the second doctor U_2, as shown in Fig. 7.

- 1. The second doctor U_2 inserts his smart card and enters his ID_2^* and pw_2^*. Next, in order to authenticate U_2 as the smartcard owner, the smartcard computes $p_2^* = h(ID_2^*||pw_2^*||a_2)$, $N_2^* = A_2 \oplus p_2^*$, $bID_2^* = h(b_2||ID_2^*)$, $M_2^* = h(b_2||ID_2^*||bID_2^*)$, and $B_2^* = h(M_2^*||N_2^*||p_2^*)$, and checks whether $B_2 = B_2^*$. If not, the smartcard is rejected. Otherwise, the smartcard selects the random number r_2, and computes $AID_1 = h(ID_1||r_2) \oplus ID_2$ and $K_1 = h(ID_2||r_2)$, where ID_1 is the identity of the first doctor (U_1). At the end, U_2 sends K_1, AID_1, and r_2, as well as the ownership transfer request in the form of $\langle K_1, AID_1, r_2, OT_{Request} \rangle$ message to U_1 through the server.
- 2. Upon receiving the message, U_1 inserts his smartcard and enters ID_1^* and pw_1^*. To authenticate U_2 as the owner of the card, the following calculations are done: $p_1^* = h(ID_1^*||pw_1^*||a_1)$, $N_1^* = A_1 \oplus p_1^*$, $bID_1^* = h(b_1||ID_1^*)$, $M_1^* = h(b_1||ID_1^*||bID_1^*)$, $B_1^* = h(M_1^*||N_2^*||p_1^*)$. The smartcard then checks whether $B_1 = B_1^*$. If so, it is verified that the smartcard belongs to U_1. Then, the smartcard computes $ID_2^* = AID_1 \oplus h(ID_1||r_2)$ and $K_1^* = h(ID_2^*||r_2)$, and checks whether $K_1^* =?K_1$ to ensure that the message has been sent from the second doctor U_2. He then selects the random number r_1 and calculates $BID_1 = h(SID||r_1) \oplus ID_1$, $k_2 = h(ID_1||r_1)$, and $CID_1 = h(SID||r_1) \oplus ID_2$, where SID is the server's identity. He also computes $k_3 = h(ID_2||r_2)$ and $k_4 = h(k_2||k_3)$, and sends $\langle k_4, r_1, r_2, BID_1, CID_1 \rangle$ to the medical server.
- 3. Once the message is received at the server, it first retrieves the doctors' identities ID_1 and ID_2 as $ID_1 = h(SID||r_1) \oplus BID_1$, $k_2 = h(ID_1||r_1)$ and $ID_2 = h(SID||r_2) \oplus CID_1$. Next, in order to authenticate the message, it computes $k_3 = h(ID_2||r_2)$ and $k_4^* = h(k_2||k_3)$ and checks whether $k_4^* = k_4$. If so, the message has been verified. Now, the server updates the Access Control List (ACL) to revoke the access permissions of U_1 to j_{th} sensor and transfer it to U_2. U_1 will no longer have access to the data.

5.5. Password update phase

This section describes how the user's password is updated. The following are the steps needed to do so.

- Step 1. The user inserts his smart card, and enters ID_i^* and pw_i^*. To verify the smartcard ownership of the user, the smart card calculates the following:
  $p_i^* = h(ID_i^*||pw_i^*||a_i)$
  $N_i^* = A_i \oplus p_i^*$
  $bID_i^* = h(b_i||ID_i^*)$
  $M_i^* = h(b_i||ID_i^*||bID_i^*)$
  $B_i^* = h(M_i^*||N_i^*||p_i^*)$
  The ownership is verified by comparing B_i^* with B_i stored on the smartcard.
- Step 2. The user enters pw_i^{**} as the new password. Then, the smartcard calculates:
  $p_i^{**} = h(ID_i^*||pw_i^{**}||a_i)$
  $N_i^{**} = A_i \oplus p_i^{**}$
  $bID_i^* = h(b_i||ID_i^*)$
  $M_i^* = h(b_i||ID_i^*||bID_i^*)$
  $B_i^{**} = h(M_i^*||N_i^{**}||p_i^{**})$
  In the end, B_i is replaced by B_i^{**} in the smartcard.

Ownership transfer phase of the proposed scheme

Doctor(U_1)		**Server(S)**	**Doctor**(U_2)
			inserts smart card
			inputs ID_2^*, pw_2^*
			$p_2^* = h(ID_2^* \| pw_2^* \| a_2)$
			$N_2^* = A_2 \oplus p_2^*$
			$bID_2^* = h(b_2 \| ID_2^*)$
			$M_2^* = h(b_2 \| ID_2^* \| bID_2^*)$
			$B_2^* = h(M_2^* \| N_2^* \| p_2^*)$
			$B_2 = ?B_2^*$
			selects random num r_2
			$AID_1 = h(ID_1 \| r_2) \oplus$
			ID_2
			$K_1 = h(ID_2 \| r_2)$
	$\xleftarrow{\langle K_1, AID_1, r_2, OT_{Request} \rangle}$		
			inserts smart card
inputs ID_1^*, pw_1^*			
$p_1^* = h(ID_1^* \| pw_1^* \| a_1)$			
$N_1^* = A_1 \oplus p_1^*$			
$bID_1^* = h(b_1 \| ID_1^*)$			
$M_1^* = h(b_1 \| ID_1^* \| bID_1^*)$			
$B_1^* = h(M_1^* \| N_2^* \| p_1^*)$			
$B_1 = ?B_1^*$			
$ID_2^* = AID_1 \oplus h(ID_1 \| r_2)$			
$K_1^* = h(ID_2^* \| r_2), K_1^* = ?K_1$			
selects random number r_1			
$BID_1 = h(SID \| r_1) \oplus ID_1$			
$k_2 = h(ID_1 \| r_1)$			
$CID_1 = h(SID \| r_1) \oplus ID_2$			
$k_3 = h(ID_2 \| r_2)$			
$k_4 = h(k_2 \| k_3)$	$\xrightarrow{\langle k_4, r_1, r_2, BID_1, CID_1 \rangle}$		
		$ID_1 = h(SID \| r_1) \oplus BID_1$	
		$k_2 = h(ID_1 \| r_1)$	
		$ID_2 = h(SID \| r_2) \oplus CID_1$	
		$k_3 = h(ID_2 \| r_2)$	
		$k_4^* = h(k_2 \| k_3), k_4^* = ?k_4$	
		Updates ACL for U_1, U_2	

Fig. 7. Ownership transfer phase of the proposed scheme.

6. Security analysis of the proposed scheme

In this section, we prove the security of the proposed scheme in terms of resistance against various attacks. We also formally demonstrate its success in satisfying the security properties using the Scyther tool.

6.1. Informal security analysis

Perfect Forward Secrecy

If the adversary knows the longterms of a scheme (e.g., IDs or secret keys) but cannot obtain the session key sk_p, the scheme is considered to provide perfect forward secrecy. In Slight, $sk_p = h(q_p || IDS_j || T_P)$, includes random numbers T_P and q_p. So, even if the attacker knows the server's secret key or the sensors' passwords, he cannot retrieve sk_p.

Replay Attack

In Slight, the server is able to identify whether the adversary replays an old message $\langle M1_i, T_1, b_i, M2_i \rangle$ by verifying $|T_2 - T_1| \leq \Delta T$ and terminating the session if not so. The adversary may replace T_1 with current time T_1^{**} and transmit $\langle M1_i, T_1^{**}, b_i, M2_i \rangle$ to the server. In this case, the server computes $d_i = h(M_i || N_i || T_1^{**}) \oplus M1_i$ and $M2_i^{**} = h(M_i || N_i || d_i)$ and verifies $M2_i^{**} = M2_i$, and if not equal, it infers that the timestamp has been manipulated. The same story is for T_3 in $\langle Q_j, T_3, Auth_j \rangle$ in step 4 of the authentication phase, where the attacker may replace T_3 with T_3^{**} and sends $\langle Q_j, T_3^{**}, Auth_j \rangle$. In this case, the server first checks $|T_4 - T_3| \leq \Delta T$.

It then computes $Auth_s = h(sk_s||w_j||T_3^{**})$ and compares $Auth_s$ with $Auth_j$ received from the sensor, which will not be equal if the timestamp has been altered.

Insider Attack

As we discussed in the registration phase in Section 5.1, in the registration request message, $\langle p_i, ID_i \rangle$ is transmitted to the server, instead of the password itself. So, the insider cannot obtain pw_i from this message, as $p_i = h(ID_i||pw_i||a_i)$ has been sent instead, from which, pw_i cannot be acquired since it contains the random number a_i.

Known-session-specific Temporary Information Attack

Robustness to this attack means it should not be possible for the attacker to retrieve the session key sk_p by obtaining the session random numbers d_i, q_p, and T_p. In the registration phase, we make the assumption that the server's memory is tamper-proof. So, the attacker cannot access the identity of the j_{th} sensor (i.e., IDS_j) which has been stored in the server's database. So, even by obtaining random numbers q_p and T_p, the adversary cannot calculate $sk_p = h(q_p||IDS_j||T_P)$.

Known-key Secrecy

As $sk_p = h(q_p||IDS_j||T_P)$ includes session-specific random numbers q_p and T_P which are different with the ones in previous sessions., if an old session key is revealed, session keys that are for other sessions cannot be obtained, implying that the scheme supports known-key secrecy.

Sensor Impersonation Attack

To impersonate the sensor, the attacker attempts to send a semi-valid $\langle Q_j, T_3, Auth_j \rangle$ message to the server. The server in turn, computes $sk_s = h(q_p||IDS_j||T_p)$, where q_p, IDS_j, and T_P are in its own possession (have not been obtained from the message). Next, the server calculates $Auth_s = h(sk_s||w_j||T_3)$ and compares it with $Auth_j$ received via the message. So, if the message has been created by the attacker, the comparison will fail, showing that the message has not been sent by the valid sensor. In other words, the proposed scheme prevents sensor impersonation attacks.

User Impersonation Attack

To impersonate the user, the attacker forwards a semi-valid $\langle M1_i, T_1, b_i, M2_i \rangle$ message to the server. The server calculates $N_i = h(SID||s||b_i)$, $d_i = h(M_i||N_i||T_1) \oplus M1_i$ and $M2_i^* = h(M_i||N_i||d_i)$, where all include parameters that are in the server's possession and have not been extracted from the message. The server then checks whether $M2_i^* = M2_i$. So, if the message has not been created by the user, the comparison fails, meaning that the proposed scheme avoids user impersonation attacks.

Server Impersonation Attack

To impersonate the server, the attacker transmits a semi-valid message $\langle d_i, z_j, Y_j, T_2 \rangle$ to the sensor. Since the sensor computes Y_j^* based on his parameters (and not the parameters included in the received message) and compares it with the received Y_j, it can easily determine whether the message has been sent from the server or not, since in the latter case, the comparison will fail. The attacker may also desire to impersonate the server via sending a semi-valid $\langle T_4, u_i, v_i, M3_i \rangle$ message to the user. In this case, the user computes $M3_i^* = h(u_i||v_i||p_i||T_4)$ and compares it with $M3_i$ included in the message to assure that the message has been sent by the valid server. Although u_i, v_i and T_4 have been obtained from the semi-valid message, p_i is only in the possession of the user. So, $M3_i^*$ will not be equal to $M3_i$ if the message has been created by the attacker.

Denial of Service Attack

In the proposed protocol, freshness of the transmitted messages is guaranteed by using timestamps throughout the authentication phase. Random numbers are also used to prevent the adversary from submitting repetitive messages. Hence, we claim that the proposed scheme is robust against denial of service attacks.

Password Guessing Attack

If the attacker cannot acquire the user's password by knowing the publicly exchanged messages, the scheme is considered secure against the password guessing attack. In Slight, parameters p_i and ID_i are exchanged securely in the registration phase, and hence, the adversary cannot acquire them. Even if he can access them, the user's password pw_i is not revealed from $p_i = h(ID_i||pw_i||a_i)$, due to the random number a_i, which is not in the hand of the adversary.

Stolen Verifier Attack

The scheme is said to be resistant to this attack If the adversary cannot retrieve the session key by obtaining the verification parameters stored into the memory. The memory is considered to be tamper-proof. Even if not so, the adversary cannot obtain the session key as it contains the random number q_i, unknown to the attacker.

Man-in-the-middle Attack

The scheme resists man-in-the-middle attacks since it ensures mutual authentication between parties before establishing the session key. So, even if the adversary eavesdrops on the channel and obtains the exchanged parameters, he cannot manage to impersonate himself as one side of communication for the other side.

Key Compromise Impersonation (KCI) attack

To resist the Key Compromise Impersonation (KCI) attack, the attacker should not be able to impersonate himself as a legal party and exchange a session key with the other party, even if he obtains the server's secret key. As discussed above, the session key $sk_p = h(q_p||IDS_j||T_P)$ includes random numbers T_P and q_p. So, even if the attacker knows the server's secret key, he cannot retrieve sk_p.

Claim				Status	Comments
Slight	Ui	Slight,Ui1	Secret idi	Ok	No attacks within bounds.
		Slight,Ui2	Secret H1(XOR(H1(idi,pwi,ai),vi),idsj)	Ok	No attacks within bounds.
		Slight,Ui3	Alive	Ok	No attacks within bounds.
		Slight,Ui4	Nisynch	Ok	No attacks within bounds.
		Slight,Ui5	Niagree	Ok	No attacks within bounds.
		Slight,Ui6	Weakagree	Ok	No attacks within bounds.
	S	Slight,S1	Secret sid	Ok	No attacks within bounds.
		Slight,S2	Secret H1(XOR(H1(idi,pwi,ai),vi),idsj)	Ok	No attacks within bounds.
		Slight,S3	Alive	Ok	No attacks within bounds.
		Slight,S4	Nisynch	Ok	No attacks within bounds.
		Slight,S5	Niagree	Ok	No attacks within bounds.
		Slight,S6	Weakagree	Ok	No attacks within bounds.
	Sj	Slight,Sj1	Secret idsj	Ok	No attacks within bounds.
		Slight,Sj2	Secret H1(XOR(H1(di,idsj,fj),XOR(H1(di,idsj,fj),XO...	Ok	No attacks within bounds.
		Slight,Sj3	Alive	Ok	No attacks within bounds.
		Slight,Sj4	Nisynch	Ok	No attacks within bounds.
		Slight,Sj5	Niagree	Ok	No attacks within bounds.
		Slight,Sj6	Weakagree	Ok	No attacks within bounds.

Done.

Fig. 8. Formal Security Analysis using the Scyther Tool.

```
usertype TimeStamp;
hashfunction H1;
secret XOR: Function;
secret idi,pwi,ai,bi,sid,s,idsj,fj,Xj,di,qp,Tp,Qj,vi;
macro pi=H1(idi,pwi,ai);
macro bidi=H1(bi,idi);
macro Mi=H1(bi,idi,bidi);
macro Ni=H1(sid,s,bi);
macro Ai=XOR(Ni,pi);
macro Bi=H1(Mi,Ni,pi);
macro Gj=H1(idsj,Xj,fj);
macro wj=H1(idsj,s,fj);
protocol Slight(Ui,S,Sj)
{
role Ui
{
var M3i,vi,ui;
macro pii=H1(idi,pwi,ai);
macro Nii=XOR(Ai,pi);
macro bidi=H1(bi,idi);
macro Mi=H1(bi,idi,bidi);
macro Bii=H1(Mi,Nii,pii);
match(Bi,Bii);
macro M1i=XOR(H1(Mi,Nii),di);
macro M2i=H1(Mi,Ni,di);
send_1(Ui,S,(M1i,bi,M2i));
recv_4(S,Ui,(ui,vi,M3i));
macro M3ii=H1(ui,vi,pii);
match(M3i,M3ii);
macro Tp=XOR(pi,ui);
macro qp=XOR(pi,vi);
macro sku=H1(qp,idsj);
claim(Ui, Secret,idi);
claim(Ui, Secret, sku);
claim (Ui, Alive ) ;
claim (Ui, Nisynch );
claim(Ui,Niagree);
claim(Ui,Weakagree);
};
role S
{
var Authj,Qj;
recv_1(Ui,S,(M1i,bi,M2i));
macro Ni=H1(sid,s,bi);
macro dii=XOR(H1(Mi,Nii),M1i);
macro M2ii=H1(Mi,Ni,dii);
match(M2ii,M2i);
macro zj=XOR(H1(di,idsj,fj),qp);
macro wj=H1(idsj,s,fj);
macro Yj=H1(wj,Gj,qp);
send_2(S,Sj,(di,zj,Yj));
recv_3(Sj,S,(Qj,Authj));
macro pi=XOR(Ni,Ai);
macro Tp=XOR(H1(Gj,fj,wj),Qj);
macro sks=H1(qp,idsj);
macro Auths=H1(sks,wj);
match(Auths,Authj);
macro ui=XOR(pi,Tp);
macro vi=XOR(pi,qp);
macro M3i=H1(ui,vi,pi);
send_4(S,Ui,(ui,vi,M3i));
claim(S, Secret,sid);
claim(S, Secret, sks);
claim (S, Alive ) ;
claim (S, Nisynch );
claim(S,Niagree);
claim(S,Weakagree);
};
role Sj
{
recv_2(S,Sj,(di,zj,Yj));
macro qpp=XOR(H1(di,idsj,fj),zj);
macro Gj=H1(idsj,Xj,fj);
macro Yjj=H1(wj,Gj,qpp);
match(Yjj,Yj);
macro skp=H1(qpp,idsj);
macro Qj=XOR(H1(Gj,fj,wj),Tp);
macro Authj=H1(skp,wj);
send_3(Sj,S,(Qj,Authj));
claim(Sj, Secret,idsj);
claim(Sj, Secret, skp);
claim (Sj, Alive ) ;
claim (Sj, Nisynch );
claim(Sj,Niagree);
claim(Sj,Weakagree);
};
}
```

Fig. 9. Scyther Code of the proposed scheme.

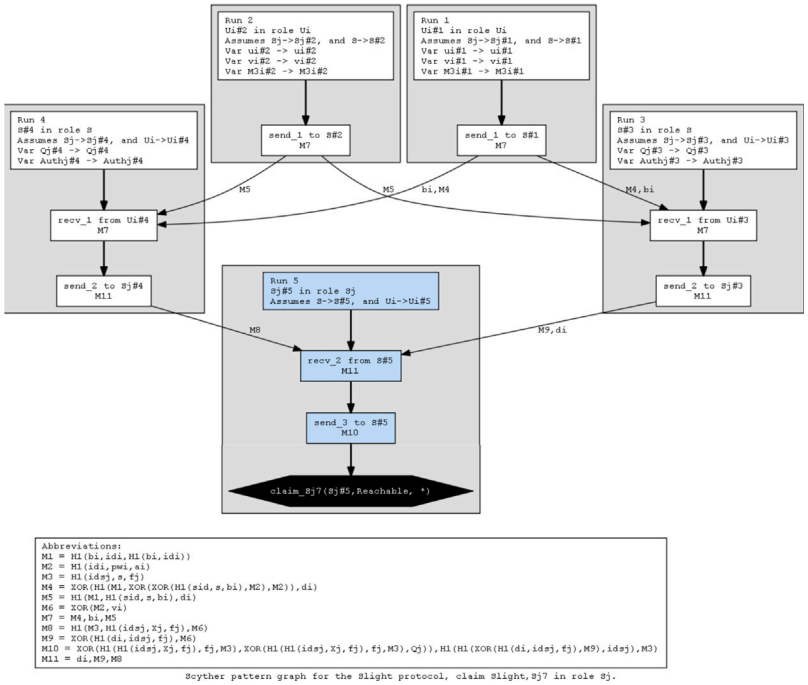

Fig. 10. Pattern of the proposed scheme in Scyther.

6.2. Formal security analysis via Scyther

In this section, we use Scyther [5], which is a formal automated tool, to analyze and verify the security properties of Slight. Each security property can be represented by a claim. For instance, the *Alive* claim ensures the execution of events by the communication party R. *Nisynch* guarantees the send/receipt of all transmitted messages by the sender/receiver.

$(R; secret; rt)$ is used to express R's claim regarding secrecy of rt to the attacker. Finally, *weakagree* ensures safety against the impersonation attack. Scyther analysis result which has been depicted in Fig. 8, proves that our scheme satisfies the above security properties. The Scyther code has been mentioned in Fig. 9. We also give the pattern of our scheme via the Scyther simulator in Fig. 10.

7. Performance analysis

In this section, we illustrate the results of performance analysis of the scheme in comparison with Mishra et al. [6], Lu et al. [8], Khatoon et al. [12], Zhang et al. [19], and Aghili et al. [4]. We also compare the computation and communication time with the schemes mentioned above.

7.1. Security comparison

Table 3 depicts the security comparison results, where — stands for $Nothing$, implying that the related security attribute has not been discussed in the referenced paper. In this table, Slight is robust to all known attacks and supplies perfect forward secrecy and known-key secrecy.

7.2. Computation and communication overhead comparison

Table 4 describes the notations for computational cost calculations. We considered the execution timings expressed in [20]. So, the approximate execution timings of $T_{hf}, T_{mu}, T_{en/d}$ and T_{bh} are 0.0004 ms, 7.3529 ms, 0.1303 ms, and 0.01 ms, respectively. In Slight, only nineteen hash function operations are needed. Hence, our scheme incurs $19T_{hf}$ computation overhead. In Table 5 and Fig. 11, we compare the computation time of Slight with Ostad-Sharif et al. [10,21], Arshad and Nikooghadam [22], Zahid et al. [23], Chen et al. [24], Mishra et al. [6], Lu et al. [8], Khatoon et al. [12], Zhang et al. [19], Wong et al. [17], and Aghili et al. [4]. Slight outperforms others by incurring the total computation cost of 0.0076 ms, while it is 29.41 ms for Ostad-Sharif et al. [21], 29.71 ms for [10], 29.43 ms for Arshad and Nikooghadam [22], 0.2613 ms for Zahid et al. [23], 0.522 ms for Chen et al. [24], 0.4173 ms for Mishra et al. [6], 29.4172 ms for Lu et al. [8], 37.0299 ms for Khatoon et al. [12], 0.0476 ms for Zhang

Table 3
Comparison of security features; SA := security attributes.

SA	[6]	[8]	[12]	[19]	[4]	Slight
SA_1	×	×	✓	✓	✓	✓
SA_2	–	–	–	×	✓	✓
SA_3	×	✓	✓	✓	✓	✓
SA_4	✓	✓	–	×	×	✓
SA_5	–	–	–	–	–	✓
SA_6	–	–	–	–	✓	✓
SA_7	–	×	✓	✓	✓	✓
SA_8	–	×	✓	✓	×	✓
SA_9	–	–	–	–	–	✓
SA_{10}	×	✓	×	✓	✓	✓
SA_{11}	×	×	×	×	✓	✓
SA_{12}	✓	✓	✓	×	×	✓

SA_1: Robustness to password guessing attack
SA_2: Robustness to DoS attack
SA_3: Robustness to replay attack
SA_4: Robustness to insider attack
SA_5: Robustness to known session-specific temporary information attack
SA_6: Robustness to sensor impersonation attack
SA_7: Robustness to user impersonation attack
SA_8: Robustness to server impersonation attack
SA_9: Robustness of known-key secrecy
SA_{10}: Support of perfect forward secrecy
SA_{11}: Support of ownership transfer
SA_{12}: Support of password update
-: Nothing
×: No
✓: Yes

Table 4
Notations used for computation cost comparison.

Symbol	Description
T_{hf}	Time of hash function
$T_{en/d}$	Time of symmetric encryption/decryption
T_{mu}	Time of scalar multiplication of elliptic curve
T_{ad}	Time of point addition of elliptic curve
T_{bh}	Time of a biohashing

et al. [19], 0.3965 ms for Wong et al. [17], and 0.021 ms for Aghili et al. [4]. Moreover, as shown in Table 3, Slight better resists the security threats.

Although Aghili et al.'s scheme [4] is lightweight, it should be noted that it takes a longer time compared to ours as it applies a hash function on the biometric parameter. It also has used more hash functions than our scheme. So, the total computation cost of the proposed scheme is less than Aghili et al.'s scheme [4]. Schemes such as Ostad-Sharif et al. [10,21], Arshad and Nikooghadam [22], Lu et al. [8], and Khatoon et al. [12] have used ECC as the encryption technique, and hence, they are not lightweight. So, they incur higher computation costs compared to ours. Zahid et al. [23] and Chen et al. [24] have also used encryption but incurred less computation cost than the scheme mentioned above, as they used symmetric encryption instead of ECC. However, their computation cost is still higher than the proposed scheme.

We also evaluate Slight in terms of communication overhead (length of exchanged messages in bits) during the login and authentication phase. Based on the results provided in [20], the communication cost is 160 bits for sending identity, 32 bits for the time stamp, 128 bits for encryption/decryption, 320 bits for elliptic curve point multiplication, 32 bits for the realm, and 32 and 160 bits for random number and output hash function, respectively. The results of communication overhead comparison that have been expressed in Table 6 demonstrate that Slight incurs acceptable communication overhead, compared to the related schemes.

To summarize, Slight provides the highest security level and incurs the minimum computational cost and comparable communication cost compared to the related schemes. This is of great importance for e-health services in the IoMT environment, where the protection of patients' private data against various attacks is critical. Simultaneously, the protection mechanism should be lightweight to comply with the resource constraint of IoT devices.

8. Appliance of slight for future IoMT

CI encourages authorities to consider smart realizable behavior of the user's applications, e.g., IoMT. IoMT has been increasingly adopted in smart healthcare systems due to its benefits in reducing unnecessary hospital costs, removing the burden by linking patients to their doctors and enabling the transfer of medical data over a secure communication network. This feature is of utmost importance for elderly people who need continual health monitoring. Moreover, IoMT is increasingly prevalent and is a proper

Table 5
Computation cost comparison.

Scheme	Total computation	Time (ms)
Ostad-Sharif et al. [21]	$4T_{mu} + 14T_{hf} + 2T_{en/d}$	29.41
Ostad-Sharif et al. [10]	$4T_{mu} + 19T_{hf} + 2T_{en/d} + 4T_{ad}$	29.71
Arshad and Nikooghadam [22]	$4T_{mu} + 23T_{hf} + 2T_{ad}$	29.43
Zahid et al. [23]	$15T_{hf} + 2T_{en/d}$	0.2613
Chen et al. [24]	$16T_{hf} + 4T_{en/d}$	0.522
Mishra et al. [6]	$16T_{hf} + 3T_{en/d} + 2T_{bh}$	0.4173
Lu et al. [8]	$4T_{mu} + 14T_{hf}$	29.4172
Khatoon et al. [12]	$5T_{mu} + 12T_{hf} + 2T_{en/d}$	37.0299
Zhang et al.[19]	$19T_{hf} + 4T_{bh}$	0.0476
Wong et al. [17]	$14T_{h} + 3T_{en/d}$	0.3965
Aghili et al. [4]	$28T_{hf} + 1T_{bh}$	0.021
Slight	$19T_{hf}$	**0.0076**

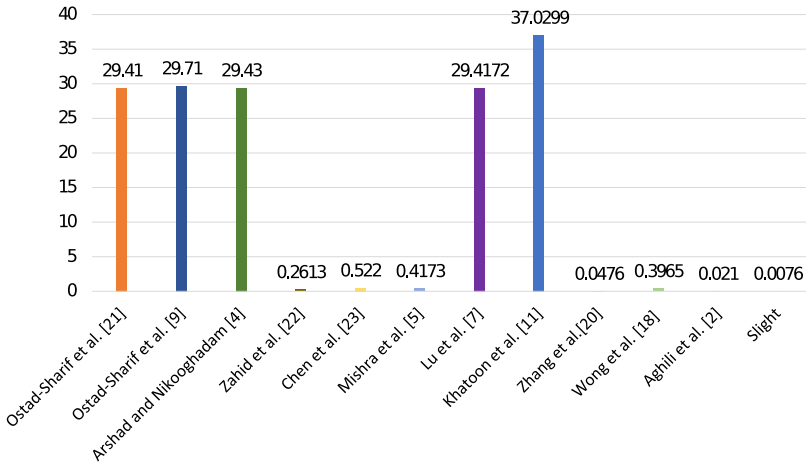

Fig. 11. The execution time comparison among protocols in (ms).

Table 6
The communication cost comparison in (bits).

Protocol	First Msg	Second Msg	Third Msg	Fourth Msg	Total
[21]	640	608	–	–	1024
[10]	672	640	–	–	1312
[22]	704	800	160	–	1664
[23]	512	512	160	–	1184
[24]	128	160	512	–	800
[6]	320	352	160	–	832
[8]	512	512	192	–	1216
[12]	480	480	–	–	960
[19]	480	320	160	160	1120
[17]	320	184	320	120	944
[4]	704	512	352	352	1920
Slight	**384**	**384**	**352**	**512**	**1632**

candidate to prevent, detect, and monitor emerging pandemic diseases such as COVID-19. In fact, IoMT supplies remote monitoring of large numbers of patients by collecting their biometric measurements via wearable sensors and transmitting them to medical servers without exposing the medical staff to infection, hence controlling the spread of infectious diseases. Hence, it can be inferred that the future of the healthcare industry will face a great shift to IoMT systems and applications. However, security will still be a major challenge in IoMT, and if not resolved, it may lead to the privacy breach in patients' medical data. This challenge becomes even more important as we consider the inherent storage and processing constraints of wearable IoT devices.

We propose Slight to address the security challenges in future IoMT. Via our proposed scheme, patients are able to receive remote medical advice from doctors and specialists without being forced to attend the hospitals physically. Besides, they are confident that their personal health data is kept secure and will not be revealed to un-authorized entities. Slight enables authenticated and secure communication between different parties in IoMT. Moreover, Slight is lightweight in regard to incurring minimum computation and acceptable communication costs, making it appropriate for IoMT with a large number of resource-constraint sensor devices. We also suggest that Slight can be applied to 6G technology [25].

In future, we aim to extend the proposed scheme and implement it on hardware microcontrollers such as AT91SAM3X8E or Cortex-M series microcontrollers.

9. Conclusion

With recent advancements on the Internet of Medical Things, preserving the privacy of patients is a major concern that needs to be addressed. This paper first reviewed Aghili et al.'s authentication method proposed for IoMT and proved its vulnerability against insider attacks and server impersonation attacks. We further proposed a lightweight authentication and key agreement scheme for smart healthcare systems, Slight. We discussed the security of the Slight against various attacks and proved it using the Scyther tool. We also compared the scheme's computational overhead with other techniques and demonstrated that Slight achieved minimum computational cost. We also show that Slight incurs comparable communication overhead. To conclude, the proposed scheme successfully satisfies smart healthcare applications' energy consumption requirements and security consideration.

CRediT authorship contribution statement

Haleh Amintoosi: Data curation, Writing – original draft. **Mahdi Nikooghadam:** Conceptualization, Methodology, Simulation. **Mohammad Shojafar:** Supervision, Reviewing and editing. **Saru Kumari:** Investigation, Supervision. **Mamoun Alazab:** Validation, Reviewing and editing.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.compeleceng.2022.107803.

References

[1] Sodhro AH, Pirbhulal S, Sangaiah AK. Convergence of iot and product lifecycle management in medical health care. Future Gener Comput Syst 2018;86:380–91.
[2] Kompara Marko, Islam SK Hafizul, Hölbl Marko. A robust and efficient mutual authentication and key agreement scheme with untraceability for WBANs. Comput Netw 2019;148:196–213.
[3] Masud Mehedi, Gaba Gurjot Singh, Alqahtani Salman, Muhammad Ghulam, Gupta BB, Kumar Pardeep, et al. A lightweight and robust secure key establishment protocol for internet of medical things in COVID-19 patients care. IEEE Internet Things J 2020.
[4] Aghili SF, Mala H, Shojafar M, Peris-Lopez P. LACO: Lightweight three-factor authentication, access control and ownership transfer scheme for e-health systems in IoT. Fut Gen Comput Syst 2019;96:410–24.
[5] Cremers CJF. Scyther: Semantics and verification of security protocols. Eindhoven, Netherlands: Eindhoven university of Technology; 2006.
[6] Mishra D, Mukhopadhyay S, Chaturvedi A, Kumari S, Khan MK. Cryptanalysis and improvement of Yen et al.'s biometric-based authentication scheme for telecare medicine information systems. J Med Syst 2014;38(6).
[7] Jiang Q, Chen Z, Li B, Shen J, Yang L, Ma J. Security analysis and improvement of bio-hashing based three-factor authentication scheme for telecare medical information systems. J Ambient Intell Humaniz Comput 2018;9(4):1061–73.
[8] Lu Y, Li L, Peng H, Yang Y. An enhanced biometric-based authentication scheme for telecare medicine information systems using elliptic curve cryptosystem. J Med Syst 2015;39(3):1–8.
[9] Zhang L, L, Zhu S, Tang S. Privacy protection for telecare medicine information systems using a chaotic map-based three-factor authenticated key agreement scheme. IEEE J Biomed Health Inf 2017;21(2):465–75.
[10] Ostad-Sharif A, Abbasinezhad-Mood D, Nikooghadam M. A robust and efficient ECC-based mutual authentication and session key generation scheme for healthcare applications. J Med Syst 2019;43(1):1–22.
[11] Kumari S, Chaudhary P, Chen C, Khan MK. Questioning key compromise attack on Ostad-Sharif, et al 's authentication and session key generation scheme for healthcare applications. IEEE Access 2019;7:39717–20.
[12] Khatoon S, Rahman SMM, Alrubaian M, Alamri A. Privacy-preserved, provable secure, mutually authenticated key agreement protocol for healthcare in a smart city environment. IEEE Access 2019;7:47962–71.
[13] Koya Aneesh M, Deepthi PP. Anonymous hybrid mutual authentication and key agreement scheme for wireless body area network. Comput Netw 2018;140:138–51.
[14] Li Xiong, Ibrahim Maged Hamada, Kumari Saru, Sangaiah Arun Kumar, Gupta Vidushi, Choo Kim-Kwang Raymond. Anonymous mutual authentication and key agreement scheme for wearable sensors in wireless body area networks. Comput Netw 2017;129(2):429–43.
[15] Liu Xin, Zhang Ruisheng, Zhao Mingqi. A robust authentication scheme with dynamic password for wireless body area networks. Comput Netw 2019;161:220–34.
[16] Arfaoui A, Kribeche A, Senouci S. Context-aware anonymous authentication protocols in the internet of things dedicated to e-health applications. Comput Netw 2019;159:23–36.
[17] Wong AM, Hsu C, Le T, Hesie M, Lin TW. Three-factor fast authentication scheme with time bound and user anonymity for multi-server E-health systems in 5G-based wireless sensor networks. Sensors 2020;20(9):2511.
[18] Nikooghadam M, Amintoosi H, Bagheri N. Lightweight authentication for remote healthcare systems in cloud-IoT. In: 2020 10th International conference on computer and knowledge engineering. 2020, p. 636–43.
[19] Zhang L, Zhang Y, Tang S, Luo H. Privacy protection for e-health systems by means of dynamic authentication and three-factor key agreement. IEEE Trans Ind Electron 2018;65(3):2795–805.
[20] 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).
[21] Ostad-Sharif A, Abbasinezhad-Mood D, Nikooghadam M. An enhanced anonymous and unlinkable user authentication and key agreement protocol for TMIS by utilization of ECC. Int J Commun Syst 2019;32(5):e3913.
[22] Arshad H, Nikooghadam M. Three-factor anonymous authentication and key agreement scheme for telecare medicine information systems. J Med Syst 2014;38(136).

[23] Mehmood Z, Ghani A, Chen G, Alghamdi AS. Authentication and secure key management in E-health services: A robust and efficient protocol using biometrics. IEEE Access 2019;7:113385–97.
[24] 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;7:85440–51.
[25] Stergiou Christos L, Psannis Konstantinos E, Gupta Brij B. IoT-based big data secure management in the fog over a 6G wireless network. IEEE Internet Things J 2020;8(7):5164–71.

Haleh Amintoosi is an Assistant Professor at the Department of Computer Engineering, Ferdowsi University of Mashhad, Iran. She is also a visiting senior lecturer at the School of Computer Science and Engineering, the University of New South Wales, Australia. Haleh received her Ph.D. in Computer Science from UNSW, Australia in 2014. Her research focuses on Security, Cryptography and Computer Networks.

Mahdi Nikooghadam received the M.Sc. degree in Computer Engineering from Ferdowsi University of Mashhad, Iran, in 2021. He is now a Ph.D. student in Computer Engineering at Amirkabir University of Technology, Tehran, Iran. His research interests include information security, cryptography, and security protocols.

Mohammad Shojafar is an Associate professor in the Network Security and an Intel Innovator working in the 6G Innovation Centre (6GIC) at the University of Surrey, UK. In 2016, he received his Ph.D. in ICT from the Sapienza University of Rome, Italy. He is an Associate Editor in IEEE Transactions on Consumer Electronics, Cluster Computing and IET Communications.

Saru Kumari got the Ph.D. degree in mathematics from Chaudhary Charan Singh University, Meerut, India, in 2012, and she is currently the assistant professor at the Department of Mathematics. Her current research interests include applied cryptography and information security. She is on the Editorial Board of more than 12 journals of international repute including seven SCI journals.

Mamoun Alazab received the Ph.D. degree in computer science from Federation University, Australia. He is currently an Associate Professor with the College of Engineering, IT and Environment, Charles Darwin University, Australia. His research is multidisciplinary and focuses on cyber security and digital forensics of computer systems with a focus on cybercrime detection and prevention.