A Lightweight Mutual Authentication and Key Agreement Scheme for Healthcare Applications with Robustness to Desynchronization Attacks

by

Shamim Akhtar Shihab
B.Sc., American International University-Bangladesh, 2019

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

A Lightweight Mutual Authentication and Key Agreement Scheme for Healthcare Applications with Robustness to Desynchronization Attacks

by

Shamim Akhtar Shihab
B.Sc., American International University-Bangladesh, 2019

Supervisory Committee

Dr. Riham AlTawy, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Issa Traore, Department Member
(Department of Electrical and Computer Engineering)

ABSTRACT

Remote healthcare monitoring system is currently gaining a lot of interest due to their potential to save lives by providing patients with continuous monitoring and quick responses when they are in critical medical condition. With the development of the Internet of Things and wireless body area networks, medical personnel can now use the public channel to get real-time data from the sensors implanted in the patient's body. However, protecting patient confidentiality and privacy of shared data from various threats is a significant challenge due to the openness of wireless communication. This necessitates the implementation of a robust authentication scheme to ensure secure communication between trusted healthcare providers and sensors. To counter these issues, in 2021, Mehedi *et al.* presented a lightweight anonymous user authentication scheme for securely obtaining patient's real-time data. Their protocol is considered practical for deployment on sensor nodes as it only utilizes hash functions and does not require any public key cryptography. In this paper, we demonstrate how their protocol loses synchronization when a message is blocked/jammed and how in some scenarios, the protocol is exposed to the risk of session key disclosure. To overcome these threats, we propose a lightweight mutual authentication scheme to provide data security and privacy in healthcare. The proposed system uses a one-way hash chain technique to ensure forward secrecy and a flag parameter mechanism to make it resistant to desynchronization attacks while achieving user and sensor node anonymity. With the demonstration of both formal and informal analysis, the proposed protocol is ensured to withstand the identified attacks in Mehedi *et al.*'s scheme. The comparative analysis in terms of security and performance with relevant protocols indicates that the proposed protocol ensures higher security with considerably lower computation and communication overheads, making it suitable for practical implementation in a lightweight healthcare environment.

Contents

List of Tables

List of Figures

List of Acronyms

IoT	Internet of Things
IoMT	Internet of Medical Things
WBAN	Wireless Body Area Networks
WMSN	Wireless Medical Sensor Networks
WSN	Wireless Sensor Network
MITM	Man-in-the-Middle Attack

ACKNOWLEDGEMENTS

My deepest gratitude goes to my supervisor Dr. Riham AlTawy, for her constant support, suggestions, and guidance throughout my master's program. Without her consistent instruction and valuable insights, this thesis could not have reached its present form. I also would like to express my gratitude to my parents and sister for their endless support, love, and encouragement.

Chapter 1

Introduction and Motivation

The growing application of IoT and the advancement of wireless networks are expanding the prospect of employing IoT technology in the healthcare industry, especially for remote patient monitoring and precise diagnosis of patients. The Internet of medical things (IoMT) is another branch of IoT where health workers use wireless media to communicate with IoT-enabled medical devices or sensor nodes to get rapid access to medical data. It can significantly increase the efficiency and accessibility of proper medical treatment while also lowering costs and reducing mistakes [1]. The use of IoMT devices is expected to save the healthcare industry up to USD 300 billion [2]. Recognizing the enormous demand for remote health monitoring in the future, investors are attracted to invest in this sector. By 2028, the global market for IoMT is expected to reach USD 187.60 billion [3].

Following the Covid-19 outbreaks, it is clear that keeping track of a large number of patients continuously who are admitted to the hospital is challenging for health workers. In IoMT technology, sensors attached to the patient body gather health-related information and make them accessible to authorized healthcare providers in real-time. This enables their providers to keep a constant eye on them and be aware of any emergencies that may arise, allowing

them to respond effectively.

1.1 Problem Statement and Motivation

In IoMT, the communication between the patients' sensors and the doctor occurs through an open wireless channel, which makes it extremely vulnerable to a variety of threats. Patients' information is extremely sensitive, and any disruption of exchanged messages or malicious activity such as altering messages or deleting the message by an adversary might put their lives at risk. As a result, reliable authentication methods are required to guarantee that only authorized parties can participate in communication and a shared key between authorized parties ensures that the confidentiality of exchanged messages is protected to reap the benefits of IoMT. It is possible to address the security vulnerabilities posed by IoT using the traditional authentication protocols [4]. However, because IoT sensor nodes have limited storage space, battery life, and computing capability, these traditional asymmetric encryption-based solutions are not feasible for IoMT networks [5].

For practical implementations, lightweight authentication protocols are proposed to make the best use of available resources. However, often lightweight implementations do not ensure additional security such as forward secrecy, desynchronization, and protection against session key disclosure simultaneously. It is therefore necessary to develop a secure authentication protocol that can withstand wider forms of attacks while also being applicable for lightweight use in the healthcare environment.

In 2021, Mehedi *et al.* [6] proposed a user authentication and key agreement protocol to access IoT sensor node data securely for patient monitoring. The proposed protocol only uses hash and xor operation which makes it very lightweight. Mehedi *et al.* claims that their protocol can withstand most of the significant attacks and is computationally very efficient

since it employs fewer hash operations than other protocols. Such a lightweight scheme motivates us to investigate how well it protects against various attack vectors and maintains data privacy. This leads us to identify a few weaknesses in Mehedi *et al.*'s scheme, build on it and propose an improved authentication scheme that provides better security.

1.2 Research Objectives

Our research objectives are given below.

- Analyze the security of Mehedi *et al.*'s lightweight anonymous authentication protocol with respect to various attack vectors.
- Propose an improved authentication scheme to address those problems found in the analyzed scheme. Specifically, propose an anonymous mutual authentication scheme that can resist desynchronization attacks in different scenarios and ensures data privacy by maintaining forward secrecy.
- Design the protocol to be computationally feasible and efficient in order to be applicable in a real-world deployment. In that regard, present a comparative analysis in terms of security and performance with other relevant protocols.

1.3 Organization of the Thesis

The remainder of this thesis is organized in the following manner.

Chapter 2 (Literature Review). In this chapter, we discuss the IoT architecture, some of the previous related works, and a description of the cryptographic primitives that we have used in our protocol.

Chapter 3 (Review of Mehedi *et al.* Authentication Scheme). In this chapter, we describe

our threat model and review Mehedi *et al.* user authentication Schemes in detail. After that, we demonstrated the weaknesses that we found in the scheme.

Chapter 4 (Proposed Protocol). In this chapter, we describe our proposed authentication scheme in detail including our system architecture, objectives, and assumptions. Then we discuss the security properties that we achieve through informal and formal analysis.

Chapter 5 (Performance Evaluation). This chapter covers the computation, communication, and storage costs of the proposed protocol and a comparison of security parameters with other relevant schemes.

Chapter 6 (Final Discussion and Conclusion). In this chapter, the final outcomes of this thesis and possible future work are discussed.

Chapter 2

Background

2.1 IoT Architecture

The Internet of Things (IoT) is a network system that connects a large number of heterogeneous devices to communicate and share data via the Internet. As depicted in Figure 2.1, the basic IoT architecture can be divided into three layers [5, 7, 8].

Perception Layer: It is a physical layer that collects data such as location, temperature, speed, humidity, etc. It senses the surrounding physical phenomena by utilizing several sensing technologies, including RFID, WSN, GPS, and so on. Several attacks such as node capture, replay attack, and eavesdropping are common security threats to the perception layer.

Network Layer: The network layer receives data from the perception layer and delivers it to the application layer. It routes data using a variety of technologies, including ZigBee, 3G/4G mobile communication, Wifi, and Bluetooth. Threats identified in this layer include denial-of-service attacks, man-in-the-middle attacks, eavesdropping, etc.

Application Layer: This layer directly interacts with and provides service to the end user.

This layer contains different kinds of IoT applications such as smart homes, remote patient monitoring, smart agriculture, smart grid, etc. Some of the major issues faced by the application layer are data privacy and accessibility, data integrity, and the ability to cope with massive data.

A proper authentication scheme for identifying the authenticity of objects can solve most of the common security problems in the three layers [8].

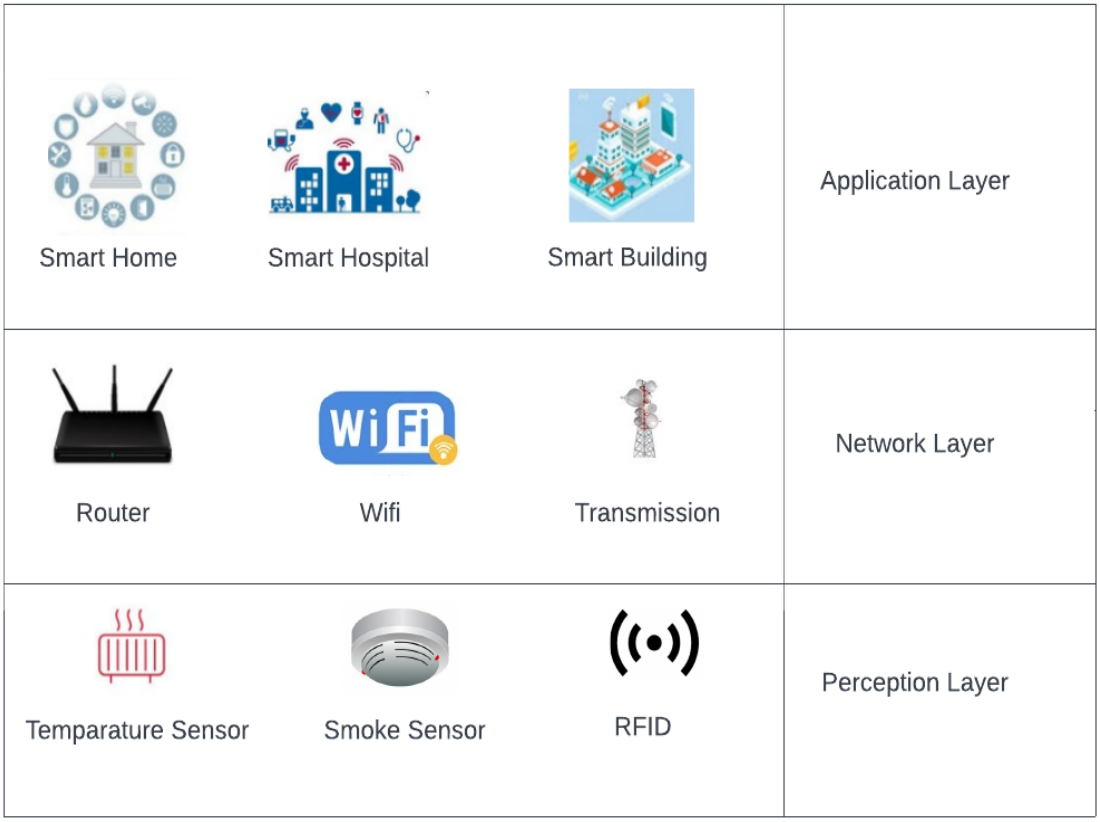

Figure 2.1: IoT architecture

2.2 IoT Authentication

Authentication is the process of validating and assuring the identity of users and devices in a network. A critical part of protecting IoT networks is to give only authorized parties access to the network so that the information system is secure from malicious activity. Therefore, it is essential to have a robust authentication system in place to counter various attacks on IoT networks.

Many different authentication protocols have been introduced thus far, each with a different design and security method [9, 10, 11]. IoT authentication protocols can be categorized into several groups depending on a variety of factors. Figure 2.2 depicts the taxonomy of the IoT authentication scheme [5]. Depending on the authentication factor, IoT authentication can be classified as identity-based authentication or physical/behavioral traits-based authentication. In identity-based authentication, one party can use symmetric or asymmetric cryptography to authenticate their identity to the other party. On the other hand, the entity can demonstrate its authenticity through physical traits (e.g., fingerprints, retina scans) or behavioral attributes (e.g. voice, gesture). Tokens are pieces of data generated by authentication servers that are used to identify different entities [12]. IoT authentication can also be classified into two categories based on the use of tokens in the authentication. In token-based authentication, tokens are used as one-time passwords generated by the authentication server, and entities can prove their authenticity as long as the tokens are valid. However, in non-token-based authentication, no tokens are used for identification; instead, other credentials (e.g. user id, password) are utilized.

The procedure of IoT authentication is classified into three types based on how different entities participate in the authentication process: one-way, two-way, and three-way authen-

tication. Only one entity proves its legitimacy to another entity in one-way authentication before communication between two entities occurs. In two-way authentication, both parties prove their authenticity to each other before engaging in communication. This method of authentication is known as mutual authentication. In three-way authentication, on the other hand, a third entity authenticates two communicating entities via mutual authentication. Finally, the architecture of IoT authentication is divided into two categories: distributed and centralized. In a distributed architecture, two participating parties are authenticated by following a straightforward authentication process. In a centralized architecture, entities rely on a trusted third party for authentication which stores and distributes secret credentials between other parties for authentication.

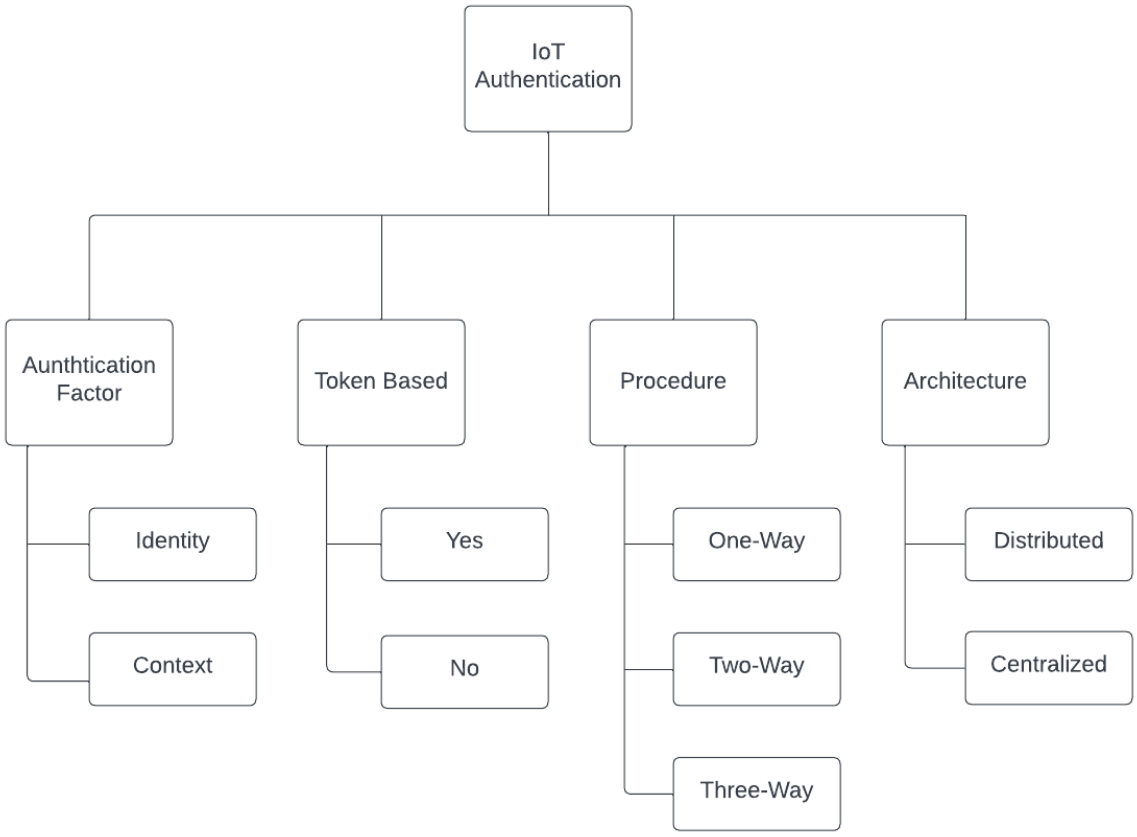

Figure 2.2: Taxonomy of IoT authentication

2.3 Attacks on Authentication Protocols

In what follows, we discuss the attacks that can be used against authentication protocols

- **Desynchronization attack**. In this attack, the adversary tries to disrupt the synchronization of exchanged values between communicating entities by jamming or dropping messages. A desynchronization attack prevents one entity from updating its state while the other entity has already updated its state, resulting in inconsistent states between the two entities.

- **Replay attack**. In this attack, the attacker captures previous communication between two entities and resends the message to a legitimate entity. As a result, when an entity receives a valid message, it considers it to be coming from a legitimate entity and starts communicating.

- **Impersonation attack**. In this attack, the adversary impersonates a legitimate entity and engages in communication with another entity without being detected.

- **Offline guessing attack**. In this attack, the attacker uses intercepted messages obtained via a public channel along with some private credentials to guess the user's credentials.

- **Session key disclosure attack**. In this attack, the attacker obtains some secret credentials by intercepting previously exchanged messages which she uses to generate a valid session key.

- **Man in the middle attack**. In this attack, the adversary acts like a proxy between the communicating entity where it can intercept the messages that are being exchanged and attempt to modify the messages before sending them to the receiving entity.

- **Privileged insider attack**. In this attack, the adversary is an authorized system user (i.e., insider) who may break certain security properties that she is not supposed to be able to do.

- **Anonymity and tracking attack**. In this attack, the adversary can reveal the true identity of a communicating entity and can track its activities by following the messages from the previous sessions.

2.4 Related Work

Various user authentication and key agreement protocols have been introduced in the last few years in response to the challenge of providing security with less processing power.

Turkanovic *et al.* [13] introduced a novel lightweight authentication technique based solely on hash and XOR computations in 2014, which Chang *et al.* [14] investigated and discovered to be vulnerable to user impersonation, sensor node spoofing, and offline password guessing attacks. Li *et al.*, on the other hand, claimed that Chang *et al.*'s scheme is unsuitable for practical implementation, lacks mutual authentication, and is vulnerable to stolen smart card and tracking attacks [15]. They further proposed a three-factor robust and energy-efficient application that eliminates the identified weakness in Chang *et al.*'s protocol.

Kumar *et al.* proposed a two-factor user authentication scheme using symmetric key cryptography to access patient data with the help of Wireless Medical Sensor Networks (WMSN) [16]. Later, He *et al.* pointed out Kumar *et al.*'s protocol could not provide user anonymity and was subject to privileged insider attacks and offline password-guessing attacks [17]. By addressing those issues, He *et al.* proposed an improved authentication scheme for healthcare [17]. However, the protocol developed by He *et al.* was susceptible to offline password guessing and impersonation attacks, as Wu *et al.* has pointed out [18].

Wu *et al.* suggested a new authentication technique and claimed that it could address the problems with He *et al.*'s protocol [18]. However, Wu *et al.*'s technique was later shown by Srinivas *et al.* to be ineffective against denial-of-service attacks and offline identity guessing attacks and is still susceptible to offline password guessing and impersonation attacks [19]. Srinivas *et al.* then proposed an enhanced authentication for WMSN that is computationally more efficient than Wu *et al.* [18] method to address the security flaw of Wu *et al.*'s scheme [19]. Unfortunately, sensor node anonymity is not considered in Srinivas *et al.*'s scheme.

In 2018, Amin *et al.* [20] proposed a new two-factor authentication scheme for WMSN, where three different parties (user, gateway, sensor) are involved in the construction of session keys. However, this scheme has weaknesses like offline guessing and desynchronization attacks [21]. Wu *et al.* suggested a two-factor robust and lightweight mutual authentication protocol [21]. With the formal and informal analysis, they proved that their protocol is secure against common threats such as offline guessing, user tracking, impersonation, and session key disclosure attack. Later, Li *et al.* showed that Wu *et al.* failed to accomplish forward secrecy and had no user-friendliness provisions [22].

Ibrahim *et al.* proposed an anonymously authenticated and key-exchanged protocol between two parties (server and sensor) for Wireless Body Area Networks (WBAN) [23]. The scheme is vulnerable to desynchronization attacks because it requires synchronized parameter updates between sensor nodes and servers during the authentication process. Although Li *et al.* claim that their scheme is secure against desynchronization attacks, it is not forward-secure and is susceptible to an impersonation attack if a sensor node is captured [24].

Sharma *et al.* proposed a Cloud-IoT-based user authentication scheme for remote patient

monitoring where sensor node anonymity is not taken into consideration [25]. They claimed that their scheme is resistant to DoS, reply, parallel session, insider, and man-in-the-middle attacks based on formal and informal analysis. Later, Ryu *et al.* identified a design flaw in the protocol by Sharma*et al.* where they demonstrated that the session keys computed by sensor node and gateway are not identical [10]. They also found out that the scheme is susceptible to password guessing, sensor impersonation attacks, and session key leakage attacks. To address those vulnerabilities, they suggest a three-factor user authentication scheme for smart health care which is designed only with XOR and a hash function.

A mutual authentication scheme based on ECC was proposed by Kalra *et al.* [26] where Kumari *et al.* [27] demonstrated that the scheme is vulnerable to offline password guessing and insider attacks, and it cannot be used to achieve mutual authentication and device anonymity. Even though Kumari *et al.* later proposed a new ECC-based authentication approach to overcome its flaws in [26], both schemes are subject to losing synchronization between IoT and cloud servers if a Dos attack occurs.

In 2019, Xu *et al.* [28], presented a lightweight anonymous mutual authentication and key agreement protocol for WBAN, based on solely on hash and XOR operation. With formal analysis, they proved that their proposed scheme can keep the secrecy of sensor node real identity and master key of hub node. However, Alzahrani *et al.* [29] demonstrated that the technique provided in [28] is vulnerable to reply attacks, key-compromise impersonation attacks, and identity guessing attacks. To counter those issues, Alzahrani *et al.* [29] proposed an improved mutually authenticated and key agreement protocol for WBAN whose security properties are formally validated by BAN logic and ProVerif.

Liu and Chung [30] proposed a user authentication scheme for patient monitoring using a smart card. The system uses bilinear pairing and needs a legitimate authority for user

authentication. Since bilinear pairing requires a large computational overhead, the proposed scheme is not suitable for WSN. Later, Challa *et al.* [31] found out that Liu and Chung [30] scheme is vulnerable to privilege insider attacks, user impersonation attacks, and offline password guessing attacks and it fails to provide user anonymity and mutual authentication. Challa *et al.* [31] present a three-factor user authentication technique using ECC to address these limitations.

Finally, Mehedi *et al.* proposed a lightweight user authentication system to protect against reply attacks, impersonation attacks, and man-in-the-middle attacks. The protocol is one of the most lightweight schemes so far as it uses only hash functions. In that scheme, the temporary IDs of the user and sensor are updated each session to prevent the attacker from tracking the identity of the user and sensor node.

2.5 Hash Functions

A hash function is a function that maps arbitrary strings into strings of fixed length. It takes a message as an input and computes a fingerprint for this message [32]. Let h be a hash function with n-bit output, i.e., h is a deterministic function that takes an arbitrary length input, and outputs a binary string of length n. Formally, $h : \{0,1\}^* \rightarrow \{0,1\}^n$.

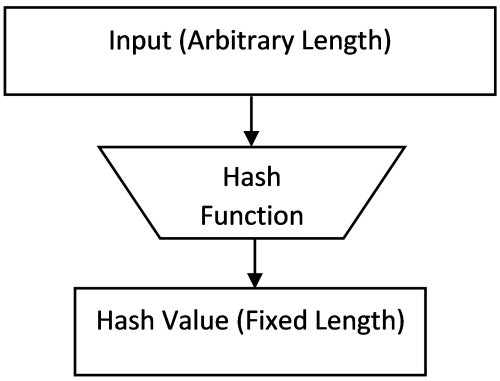

Figure 2.3: Hash function

A secure hash function must satisfy the following requirements:

- **Pre-image resistance.** A hash function h is preimage resistant (one-way) if for essentially all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output, i.e., to find any preimage x such that $h(x) = y$ when given any y for which a corresponding input is not known.

- **Second pre-image resistance.** A hash function h is second-preimage resistant (weak collision resistant) if it is computationally infeasible to find any second input which has the same output as any specified input, i.e., given x, to find a 2nd-preimage $y \neq x$ such that $h(x) = h(y)$.

- **Strong collision resistance.** A hash function h is collision resistant if it is computationally infeasible to find any two distinct inputs x and y which hash to the same output, i.e., $h(x) = h(y)$

Hash Chains. A hash chain is a cryptographic mechanism that is generated when multiple hash operations are applied to a single string of data. Lamport [33] first proposed the hash chain concept, which has been applied to data authentication, one-time passwords, and a micro-payment system [34].

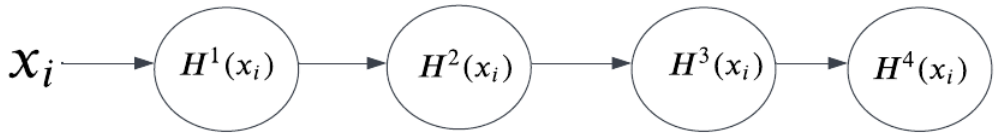

Figure 2.4: Hash chain of length 4.

We assume x_i is a string of data. As depicted in Figure 2.4, if x_i is hashed 4 times, it will create a hash chain where $H^4(x_i) = h(h(h(h(x_i))))$.

If the adversary has access to $H^4(x_i)$, finding $H^3(x_i)$ in a hash chain without knowing x_i is computationally infeasible due to the pre-image property of the hash function.

Chapter 3

Review and Analysis of Mehedi *et al.* 's Scheme

In 2021, Mehedi *et al.* proposed a lightweight and efficient authentication protocol for accessing the data collected by sensor nodes in the healthcare field [6]. Before exchanging information, the user and sensor must first verify each other's legitimacy with the help of a gateway. Following the verification of their authenticity, a session key is established and shared between the user and the sensor node for secure communication. Here, the gateway uses a centralized authentication architecture to assist the user and sensor in mutual authentication. In this chapter first, we are going to review Mehedi *et al.* 's scheme and then discuss its weakness.

The scheme proposed by Mehedi *et al.* can be divided into two phases: registration and mutual authentication. For ease of understanding of the scheme, notations are included in Table 3.1.

Table 3.1: Notations for Mehedi *et al.*'s scheme

Notation	Description
D_{ID},S_{ID}	Doctor/user ID, sensor ID
PW_D	User password
R_{req}	Registration request
R^1_{SG}	Long term random secret of User
R_{SN}, R^2_{SG}	Long-term random secret of Sensor
SK	Session key
S_{TID}	Temporary identity of Sensor
D_{TID}	Temporary identity of User
N_D,N_G, N_S	Nonce generated by user, gateway and sensor
$h(.)$	One-way hash function
$\oplus$	Bit wise XOR operation
$\|\|$	Concatenation operator

3.1 Threat Model

We use Canetti and Krawczyk (CK) [35] security model to evaluate the security of Mehedi *et al.* and our proposed system in an insecure channel. The attacker is assumed to have the following abilities:

1. The adversary has the ability to eavesdrop and intercept the messages exchanged over the communication channel. Besides, adversaries also can drop, reply and modify the transmitted message between the communicating parties (i.e., User, Gateway, and Sensor).

2. An adversary can physically seize the user's device and perform a power analysis attack to get the stored parameters saved in the user's device. The adversary can execute this attack once the user registration process is completed.

3. The communicating entity's long-term secret value can be obtained by the adversary. Additionally, the attacker has access to temporary data that is specific to a certain session.

3.2 Registration

Before beginning the authentication phase, both the user and sensor are needed to be registered with the gateway and share their unique secret credentials with the gateway. This will help the gateway to identify the user and sensor in the mutual authentication phase and use the stored parameters shared during the registration phase to carry on the authentication process. A private secure channel is being used for exchanging messages during this registration phase.

3.2.1 User Registration

- **Step 1.** At first, the user enters her id D_{ID} and password PW_D on his device which initiates a registration request by sending D_{ID} and PW_D to the gateway through a secure channel.

- **Step 2.** After receiving the message in step 1, the gateway stores D_{ID}, PW_D and generates randomly R^1_{SG} for the user specifically. Then it computes $\alpha = \left(D_{\mathrm{ID}} \oplus R^1_{\mathrm{SG}}\right) \oplus \mathrm{PW_D}$ and generates a temporary id of user by computing $D_{\mathrm{TID}} = R^1_{\mathrm{SG}} \oplus D_{\mathrm{ID}}$. It then stores $R^1_{SG}, D_{\mathrm{TID}}$ in its memory for future use and sends α to the user through a private secure channel.

- **Step 3.** Upon receiving α, the user extracts $\mathrm{R^1_{SG}} = (\alpha \oplus \mathrm{PW_D}) \oplus \mathrm{D_{ID}}$. Then a user temporary id is derived by computing $\mathrm{D_{TID}} = \mathrm{R^1_{SG}} \oplus \mathrm{D_{ID}}$. After that, the user computes $\beta = \mathrm{h}\left(\mathrm{PW_D} \| \mathrm{R^1_{SG}}\right) \oplus \mathrm{D_{TID}}$ and stores β, R^1_{SG} and D_{TID} in its memory for

future use.

3.2.2 Sensor Node Registration

- **Step 1.** At first sensor node retrieves its id S_{ID} from its memory and randomly generates R^1_{SN}, a long term secret for sensor. Then, it sends R^1_{SN} and S_{ID} to the gateway through a secure channel.

- **Step 2.** After receiving the message, the gateway stores R^1_{SN} and S_{ID} in its memory and generates a secret value for the sensor R^2_{SG} randomly. Then, it computes $\delta = \left(\mathrm{S_{ID}} \oplus \mathrm{R^2_{SG}}\right) \oplus \mathrm{R^1_{SN}}$ and sensor node temporary id $S_{\mathrm{TID}} = R^2_{SG} \oplus S_{ID}$. The gateway stores R^1_{SG} and S_{TID} in its memory and sends δ to the sensor node through a secure channel.

- **Step 3.** Upon reception, sensor node retrieves R^2_{SG}, S_{TID} by computing $\mathrm{R^2_{SG}} = \left(\delta \oplus \mathrm{R^1_{SN}}\right) \oplus \mathrm{S_{ID}}$ and $S_{TID} = R^2_{SG} \oplus S_{ID}$. Finally, the sensor node stores $R^1_{\mathrm{SN}}, R^2_{SG}, S_{\mathrm{TID}}$ in its memory.

3.3 Mutual Authentication

To receive data from the sensor node and create a secure communication channel between the sensor node and the user, all three participating parties (user, sensor, and gateway) need to be mutually authenticated, and the gateway shares a common session key between the user and sensor node. As illustrated in Fig. 3.1, the detailed step-by-step process of the mutual authentication process is described as follows.

- **Step 1.** The user first inputs his password PW_D onto the device, then the user calculates $Q = h\left(PW_D \| R^1_{SG}\right) \oplus D_{TID}$. The user then compares Q to β, where β

was stored during the registration process. If both values are the same, the phase is continued; otherwise, it is aborted. After that, it generates a nonce N_D^1, and computes $\mathrm{N_D^{1^*}} = \mathrm{N_D^1} \oplus \mathrm{PW_D}$, $\lambda = h\left(R_{SG}^1 \| PW_D\right)$. Then $\left\{\mathrm{N_D^{1^*}}, \mathrm{D_{TID}}, \lambda, \mathrm{S_{TID}}\right\}$ is sent as a message $\mathrm{M_{UA}^1}$ to the gateway through a public channel.

- **Step 2.** After receiving information from the user, the gateway checks the freshness of N_D^1 by computing $N_D^1 = N_D^{1^*} \oplus PW_D$. The process continues if N_D^1 is fresh; else, it discontinues. Given that the nonce includes a timestamp, it can provide a time for its creation. The gateway then checks the received $\mathrm{D_{TID}}$ and S_{TID} with the ones in its database. If it does not match, the procedure will end there; otherwise, it will proceed. The gateway computes $\lambda^* = \mathrm{h}\left(\mathrm{R_{SG}^1} \| PW_D\right)$ and compares it to λ from its database to determine if the user successfully authenticated to continue the process. Afterwards, the gateway generates the nonce N_G^1 and then calculate $\mathrm{G_W^1} = \mathrm{N_G^1} \oplus \mathrm{S_{TID}}$, $\mathrm{G_W^2} = \mathrm{h}\left(\mathrm{R_{SN}^1} \| R_{SG}^2\right)$. The session key SK is then generated by the gateway and encrypted inside the $\mathrm{SK_S} = \left(\mathrm{SK} \oplus \mathrm{R_{SN}^1}\right) \oplus \mathrm{N_G^1}$. Finally, the gateway generates another random value $\mathrm{R_{SG}^3}$ and compute $\mathrm{G_W^3} = \mathrm{R_{SG}^3} \oplus \mathrm{R_{SN}^1}$. Thereafter, it sends $\left\{ G_W^1, G_W^2, D_{TID}, SK_S, G_W^3\right\}$ as a message $\mathrm{M_{UA}^2}$ to sensor node via public channel.

- **Step 3.** Upon receiving $\left\{ G_W^1, G_W^2, D_{TID}, SK_S, G_W^3\right\}$ from the gateway, the sensor node at first extract the nonce $\mathrm{N_G^1} = \mathrm{G_W^1} \oplus \mathrm{S_{TID}}$ and checks whether the nonce is current or not. The sensor node terminates the session if it is not fresh. Then the gateway compute $S_N^1 = h\left(R_{SN}^1 \| R_{SG}^2\right)$. After that, the gateway checks whether computed S_N^1 matches with received G_W^2. If it matches, the sensor node retrieves the session key $\mathrm{SK} = \left(\mathrm{SK_S} \oplus \mathrm{N_G^1}\right) \oplus \mathrm{R_{SN}^1}$. Afterwards, the sensor node generates the

nonce N_S^1 and computes $\mathrm{S_N^2} = \mathrm{N_S^1} \oplus \mathrm{S_{TID}}$, $\mathrm{S_N^3} = \mathrm{h}\left(\mathrm{R_{SG}^2} \left\|\mathrm{R_{SN}^1}\right\| \mathrm{SK}\right)$, $\mathrm{S_N^4} = \mathrm{R_{SG}^2} \oplus \mathrm{R_{SN}^2}$. Thereafter, the sensor node retrieves $\mathrm{R_{SG}^3} = \mathrm{G_W^3} \oplus \mathrm{R_{SN}^1}$ and use $\mathrm{R_{SG}^3}$ parameters to update the sensor node new temporary id $S_{TID}^{\mathrm{new}} = R_{SG}^3 \oplus S_{ID}$. Sensor node then stores $\left\{ R_{SN}^2, R_{SG}^3, S_{\mathrm{TID}}^{\mathrm{new}} \right\}$ parameters in its memory and sends information $\left\{ \mathrm{S_N^2},\ \mathrm{S_N^3},\ \mathrm{S_N^4} \right\}$ as a message $\mathrm{M_{UA}^3}$ to gateway via public channel.

- **Step 4.** Upon receiving the message from the sensor, the gateway retrieves the nonce $\mathrm{N_S^1} = \mathrm{S_N^2} \oplus \mathrm{S_{TID}}$ and only continues the process if the nonce is verified to be fresh. The gateway then computes $\mathrm{G_W^4} = \mathrm{h}\left(\mathrm{R_{SG}^2} \left\|R_{\mathrm{SN}}^1\right\| \mathrm{SK}\right)$. After that, the gateway checks whether computed G_W^4 matches with received S_N^3. If both values are matched, the sensor node and gateway have successfully mutually authenticated. Afterward, the gateway retrieves $R_{SN}^2 = S_N^4 \oplus R_{SG}^2$ and updates the new sensor node temporary id in gateway side by computing $\mathrm{S_{TID}^{new}} = \mathrm{R_{SG}^3} \oplus \mathrm{S_{ID}}$. Then gateway stores $\left\{R_{SN}^2, R_{SG}^3, S_{\mathrm{TID}}^{\mathrm{new}}\right\}$ parameters. Now gateway generates nonce N_G^2 and and computes $\mu = D_{ID} \oplus N_G^2$. A session key is generated and enclosed secretly in $\mathrm{SK_U} = (\mathrm{SK} \oplus \mathrm{PW_D}) \oplus \mathrm{N_G^2}$. Afterwards, gateway calculates $\eta = h\left(D_{ID} \,\|PW_D\|\, SK \| N_G^2\right)$, $\mathrm{G_W^5} = \mathrm{R_{SG}^4} \oplus \mathrm{PW_D}$. Then gateway calculate new temporary id for user , $D_{\mathrm{TID}}^{\mathrm{new}} = R_{SG}^4 \oplus D_{ID}$. Finally, gateway stores $\left\{ R_{SG}^4, D_{\mathrm{TID}}^{\mathrm{new}} \right\}$ and sends $\left\{\mu, \mathrm{SK_U}, \eta, \mathrm{G_w^5}\right\}$ as as a message $\mathrm{M_{UA}^4}$ to user through public channel.

- **Step 5.** After receiving the message the user retrieves the nonce $\mathrm{N_G^2}$ by computing $\mathrm{N_G^2} = \mu \oplus \mathrm{D_{ID}}$. If the nonce is fresh, the user retrieves the session key $\mathrm{SK} = \left(\mathrm{SK_U} \oplus \mathrm{N_G^2}\right) \oplus \mathrm{PW_D}$. Afterwards it computes $\phi = h\left(D_{ID} \,\|PW_D\|\, SK \| N_G^2\right)$ and compares ϕ with η. If both values match, the user and gateway have established proper mutual authentication. Then, the it extracts $\mathrm{R_{SG}^4} = \mathrm{G_W^5} \oplus \mathrm{PW}_D$ and updates

user temporary id $D_{TID}^{new} = R_{SG}^{4} \oplus D_{ID}$. Finally, between the user and the sensor node, a private session key SK is successfully established.

3.4 Security Analysis

In this section, we investigate the protocol developed by Mehedi *et al.* while taking into account the threat model discussed in section 3.1. The protocol has shown some security vulnerabilities in its attempt to design an anonymous and computationally efficient protocol. In this section, we have demonstrated some of the weaknesses that we found by analyzing the scheme.

3.4.1 Long-term Secret Disclosure Attack

Here, we show that if an adversary obtains a session key SK for a specific session, adversary can get sensor node long-term secret parameter R_{SN}^{1} in the following way:

1. Adversary monitors the exchanged messages $M_{UA}^{1} = \{N_{D}^{1*}, D_{TID}, \lambda, S_{TID}\}$ and $M_{UA}^{2} = \{G_{W}^{1}, G_{W}^{2}, D_{TID}, SK_{S}, G_{W}^{3}\}$ shared in public channel and captures G_{W}^{1}, SK_{S} and S_{TID} for further use.

2. After that, adversary then uses G_{W}^{1} and S_{TID} to compute $N_{G}^{1} = G_{W}^{1} \oplus S_{TID}$.

3. Now, adversaryf possesses all the essential information SK, N_{G}^{1} and SK_{S} to compute the long term secret parameter of sensor node $R_{SN}^{1} = SK \oplus N_{G}^{1} \oplus SK_{S}$

3.4.2 Session Key Disclosure Attack and Lack of Forward Secrecy

During the mutual authentication phase, when the sensor node verifies the authenticity of the gateway, it can extract the session key SK from the exchanged message shared between

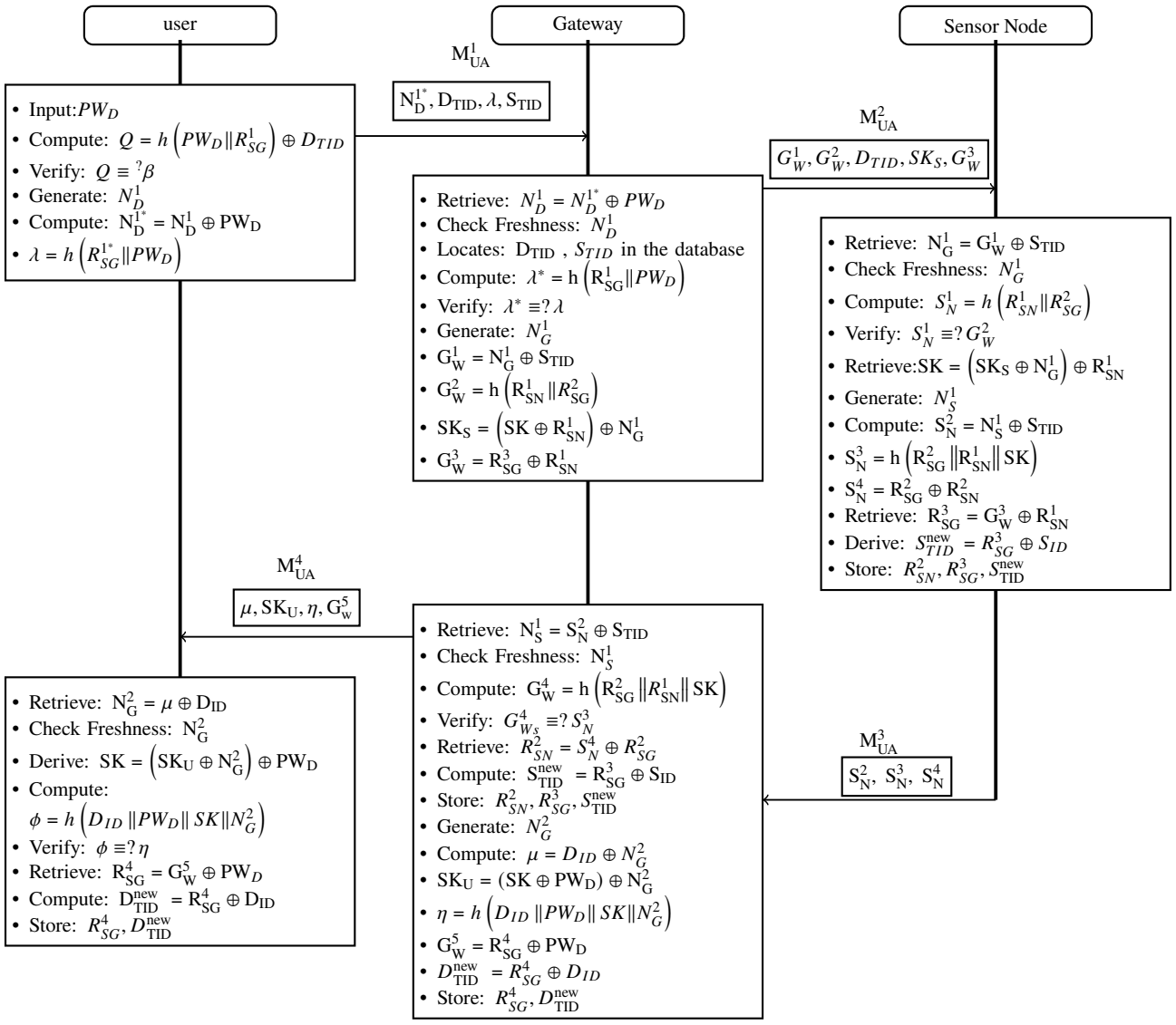

Figure 3.1: Mutual authentication and key agreement phase of Mehedi *et al.*'s Scheme

the gateway and the sensor. We show that if an adversary obtains the sensor node long-term secret parameter R_{SN}^1, eventually adversary can obtain the session key shared between the user, gateway, and sensor as follows :

1. Adversary monitors the session between user and gateway, gateway to the sensor to capture the following exchanged massages $\mathrm{M}_{\mathrm{UA}}^1 = \{N_D^{1*}, D_{TID}, \lambda, S_{TID}\}$ and $\mathrm{M}_{\mathrm{UA}}^2 = \{G_W^1, G_W^2, D_{TID}, SK_S, G_W^3\}$ and capture G_W^1, SK_S and S_{TID} for further use.

2. After that, adversary can use the captured message G_W^1, S_{TID} to compute $N_G^1 = G_W^1 \oplus S_{TID}$

3. Now adversary has knowledge of all the needed parameters R_{SN}^1 N_G^1 and SK_S to compute the session key $SK = R_{SN}^1 \oplus N_G^1 \oplus SK_S$

An adversary with access to R_{SN}^1 or can obtain R_{SN}^1 from leakage of a session key illustrated in section 3.4.1, can use the above method to obtain any session keys from a previous or future session and extract valuable information with that session key in this protocol. As a result, Mehedi *et al.*'s scheme does not provide forward secrecy.

3.4.3 Desynchronization Attack

To achieve user and sensor node anonymity, Mehedi *et al.* update its temporary user id and sensor node id after each successful authentication. However, the protocol only considers the best-case scenario, in which all messages are properly received by the intended entities. We show in what follows that it is prone to desynchronization due to its reliance on successful message delivery. Particularly, when one entity changes temporary identity while the other entity is unable to update and keeps the prior temporary identity due to a message being blocked or jammed, the temporary identity between the entities loses its synchronized value. The following are examples of such an attack in different scenarios.

- **Attack scenario 1.** The sensor node changes its temporary sensor node id S_{TID} to a new temporary sensor node id S_{TID}^{new} in step 3 of the mutual authentication phase after successful mutual authentication between the gateway and the sensor node. The sensor node then stores the new temporary id S_{TID}^{new} in its memory before sending the message $M_{UA}^{3}=\{ S_{N}^{2}, S_{N}^{3}, S_{N}^{4}\}$ to the gateway. Upon receiving the message, the gateway retrieves the new temporary sensor id S_{TID}^{new} and stores it on the gateway side. If the adversary blocks message M_{UA}^{3}, the gateway side will be unaware of the new temporary sensor node id update and will continue to communicate using the old temporary id S_{TID}, whilst the sensor node will use the new temporary id S_{TID}^{new}. This is how the dynamic temporary identity synchronization mechanism fails between the gateway and sensor node. Because when the new session starts, in step 2, the nonce N_G^1 is encrypted within $G_W^1 = N_G^1 \oplus S_{TID}$ with the old temporary id S_{TID} on the gateway side, the sensor node could not extract the correct nonce N_G^1 with their new temporary id S_{TID}^{new} after receiving message M_{UA}^{2} from the gateway. On the sensor node side, this nonce N_G^1 is also used to calculate the correct session key SK in the sensor node side. As a result authentication between the sensor node and gateway would fail afterward due to the sensor's temporary IDs being different between the sensor node and gateway.

- **Attack scenario 2.** In step 4 of the Mehedi *et al.* scheme's authentication phase, the gateway changes the temporary user id D_{TID} to a new temporary user id D_{TID}^{new} and the temporary sensor id S_{TID} to a new temporary sensor id S_{TID}^{new}. The new temporary ids D_{TID}^{new} and S_{TID}^{new} are then stored in gateway, and the message $M_{UA}^{4} = \{\mu, SK_U, \eta, G_w^5\}$ is sent to the user. After receiving the message and confirming their credibility, the user and gateway are both mutually authenticated. The user updates its temporary id D_{TID} to new temporary id D_{TID}^{new}. If a malicious attacker blocks the message M_{UA}^{4} the temporary

id of the user remains unchanged on the user side while on the gateway side temporary id of the user has been already updated. In this way, the attacker has disrupted the dynamic temporary identity synchronization mechanism between the gateway and the user by interrupting communication. As a result, while initiating a new session, the user can only utilize the previous temporary identities of the user D_{TID} and the sensor node S_{TID}, and the user can no longer authenticate with the gateway using old temporary ids.

Chapter 4

The Proposed Scheme

In this chapter, we present a lightweight mutual authentication and key agreement scheme to address the weaknesses we have identified in Mehedi *et al.*'s protocol.

4.1 System Objectives

In our proposed protocol, we want to achieve the following goals.

1. **Mutual authentication and key agreement.** Mutual authentication is essential between legitimate users and sensors in order to access the life-critical information provided by sensors. In addition, the user and sensors should be able to agree on a shared session key that protects the security of the communicated information.

2. **Forward secrecy.** Forward secrecy ensures that even if an attacker obtains a long-term secret key, she is not be able to derive any session keys from prior sessions. It assures that the attacker will not be able to reveal any critical information about the patient that was exchanged in a previous communication.

3. **Resistant against desynchronization.** For forward secrecy and pseudonym identity,

some parameters are constantly being updated among the entities. As a result, if an adversary drops or jams messages, variables in one entity may be updated while variables in the other might not, resulting in the desynchronization of values between them. The objective is for the protocol to have a mechanism for detecting desynchronization and initiating measures to bring all parties back into synchronization.

4. **Anonymity.** The intruder is always looking for loopholes and monitoring the transmitted message. In this case, the anonymity of the user and sensor's real identity ensures that the adversary cannot track the behavior of the user or sensor from the exchanged message and target specific sensor/doctor with attacks.

5. **Efficiency.** The computation, storage, and battery capacity of IoT nodes are limited. As a result, in order to be compatible with resource-constrained IoT devices, an authentication protocol that can guarantee confidentiality, forward secrecy, and resilience to adversary attacks needs to be lightweight and efficient for practical application.

4.2 Architecture

Our scheme, as depicted in Fig. 4.1, comprises three main entities: a user, a gateway, and a sensor. A user can see real-time data from a sensor node placed in the patient's body using a mobile application that is installed on their mobile device. We assume that the app is secure and that the device is not infected. The sensor collects and transmits the data to the gateway using wireless protocols such as Bluetooth, Wi-Fi, and ZigBee. Because sensor nodes have limited communication and processing capability, the gateway serves as an intermediary node to relay information between the user and the sensor. Both user and sensor need to prove their legitimacy to gateway and vice versa before establishing a secure communication

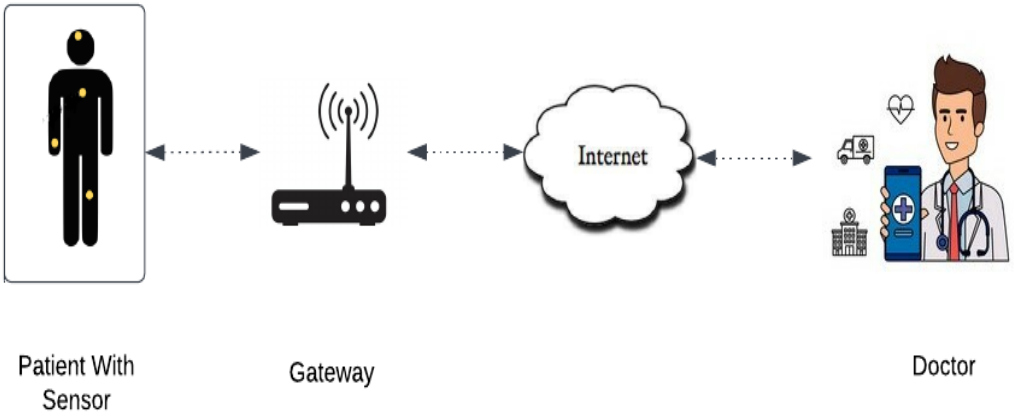

Figure 4.1: System architecture

channel between the user and sensor node. In this regard, initially, the gateway generates a session key after the user is able to prove its legitimacy. Afterward, the gateway helps both sensor and user to establish a common session key if both gateway-sensor and user-gateway can mutually authenticate each other. The common session key is then used to authenticate and encrypt the exchanged messages between the sensor and the user. For the protocol description, we consider a single doctor who gets data from a single sensor node using a specific gateway. The protocol can be generalized for a multiple doctors, sensors, and gateways scenario.

4.3 Assumptions

The following assumptions are taken into account when designing our proposed protocol.

1. The computational power, storage, and battery of the user device and sensor node are all limited. The gateway, on the other hand, does not have any computational or storage limitations and is powered by the main power source.

2. We assume the user knows the initial temporary id of the sensor node that he wants to connect with.

3. The gateway node is regarded as a trustworthy entity with tamper resistance hardware and it must be physically secured.

4. In this scheme, there are two channels, i) a public channel that is used for exchanging messages between the sensor and user, and ii) a secure private channel, for the registration phase where the adversary has no influence on the transmitted message. Such a secure channel may be established throgh physical proximity where the IoT devices are brought very close to the gateway while exchanging messages. The communication channel between IoT and gateway is regarded as secure because there are no attackers nearby and it can only occur within very close range (e.g., 1-2 meters). The adversary, on the other hand, controls the public channel that is used during the post-registration phase. An adversary can drop and modify a communication carried via the public channel. IoT devices and gateways can communicate in close proximity to form a private secure channel.

4.4 Proposed Scheme

In this section, we explain our proposed scheme in detail. The proposed scheme is divided into three phases. 1) user registration 2) sensor node registration and 3) mutual authentication and key agreement. All the notations used in the proposed system are listed in Table 4.1.

Table 4.1: Notations for the proposed scheme

Notation	Description
ID_{ui}, ID_{si}	Doctor/User identity, sensor node identity
PW_D	Password of the user
R_1,R_2, R_3	Random numbers
N_{gs},N_{ss},	Sequence number in gateway side, sequence number in sensor side
SK	Session key
K_{gu}	Secret of user shared between user and gateway
K_{gs}	Secret of sensor shared between gateway and sensor
S_{TID}	Temporary identity of sensor
S_{TIDold}	Old temporary identity of sensor
TID_{ui}	Temporary identity of user
TID_{inew}, TID_{iold},	Temporary identity of user on gateway side
SK	Session key
T_1,T_2, T_3, T_4	Timestamp
$h(.)$	One-way hash function
$\oplus$	Bit wise XOR operation
$\|\|$	Concatenation operator

4.4.1 User Registration

The user must first register with the gateway in order to get real-time data captured by the sensor node. The steps of user registration are described in detail below.

- At first the user enters the user identity ID_{ui}, password PW_D into her mobile device and transmits ID_{ui} through a private channel to the gateway.
- After receiving ID_{ui}, the gateway at first generates three random numbers TID_{ui}, K_{gu}, b_i and sets $TID_{ui} = TID_{inew}$, $TID_{iold} = null$. Then, it computes

$B_i = h(ID_{ui}||b_i)$ and stores the value $\{ID_{ui}, TID_{inew}, TID_{iold}, K_{gu}, B_i\}$ in its database. Thereafter, it sends $\{TID_{ui}, B_i, K_{gu}\}$ to the user via secure channel.

- Upon receiving the message , the user generate a random number r_i , computes $A_i = h(PW_D||r_i), E_i = h(ID_{ui}||PW_D) \oplus r_i, K_i = A_i \oplus K_{gu}, V_0 = h(A_i||ID_{ui}||r_i)$ and sets flag = 0. Then, it stores$\{V_0, TID_{ui}, E_i, B_i, K_i\}$ in its device.

4.4.2 Sensor Node Registration

Prior to transmitting real-time information, the sensor node must first register with the gateway. The steps of sensor node registration are described in detail below.

- Initially, the sensor node sends its real identity ID_{si} to the gateway via a private channel.
- Upon reception, the gateway generate a random number K_{gs} and set a sequence number $N_{GS} = N_{SS} = 0$. After that it computes temporary identity for the sensor $S_{TID} = K_{gs} \oplus ID_{si}$ and sets $S_{TIDold} = S_{TID}$ followed by storing $\{K_{gs}, N_{gs}, S_{TID}, S_{TIDold}, ID_{si}\}$ in its memory. Then, the gateway sends the parameters $\{K_{gs}, N_{ss}, S_{TID}\}$ to the sensor via a secure channel.
- After receiving $\{K_{gs}, N_{ss}, S_{TID}\}$, the sensor stores them in its memory for further use.

4.4.3 Authentication and Key Agreement

A user and a sensor node must first authenticate one another and create a shared session key in order to communicate. As illustrated in Fig. 4.3, the detailed step-by-step process of login and the mutual authentication process is described as follows.

- **Step 1.** At first, user enters id ID_{ui} and password PW_D in the device which computes, $r_i^* = E_i \oplus h\left(ID_{ui} \| PW_D\right)$, $A_i^* = h\left(PW_D \| r_i^*\right)$, and $V_0^* \equiv h\left(A_i^* \| ID_{ui} \| r_i^*\right)$. Then it compares V_0^* with the stored value V_0. If they do not match, the login request is rejected. Otherwise, it computes $K_{gu} = A_i^* \oplus K_i$ and checks whether $Flag = 0$ or not. If $Flag = 0$, it updates $K_{gu}^* = h\left(K_{gu}^*\right)$ and updates $Flag = 1$. After that it generates a random number R_1 and timestamp T_1, computes $F_1 = \mathrm{R}_1 \oplus S_{TID} \oplus h\left(TID_{ui} \| B_i \| K_{gu}\right)$ and $V_1 = h(ID_{ui} \| R_1 \| B_i \| TID_{ui} \| K_{gu} \| T_1)$. At the end, the user sends $\{ F_1, V_1, TID_{ui}, T_1\}$ information as a message M^1 to the gateway via public channel.

- **Step 2.** After receiving message M^1 from a user, the gateway at first checks the validity of the timestamp $\left|T_1^* - T_1\right| < \Delta T$. If the difference between the current timestamp T_1^* with the received timestamp T_1 exceeds the predefined threshold value ΔT, the gateway terminates the session. Otherwise, the gateway looks for TID_{ui} in its database and extracts the corresponding values. If they are found, the gateway checks TID_{ui} matches with TID_{inew} or TID_{iold}

  a. If $TID_{ui} = TID_{inew}$, then it indicates that the temporary id of user is updated at both parties (user and gateway) in the previous session. Accordingly, the gateway updates $K_{gu}^* = h\left(K_{gu}^*\right)$ as it has been already updated on the user side in this session. Then, it computes $R_1^* = \mathrm{F}_1 \oplus S_{TID} \oplus h\left(TID_{ui} \| B_i \| K_{gu}^*\right)$ and $V_1^* = h(ID_{ui} \| R_1^* \| B_i \| TID_{ui} \| K_{gu} \| T_1)$. If the computed V_1^* does not match the received value V_1, the gateway terminates the session. Otherwise, the gateway generates a new pseudonym id TID_{inew}^* for user , and set $TID_{iold} = TID_{ui}$, $TID_{inew} = TID_{inew}^*$.

  b. If $TID_{ui} = TID_{iold}$, then it indicates that the temporary id TID_{ui} of the

user is not updated on the user side whereas in the gateway side it is already updated to TID_{inew}. In this scenario, the gateway computes $R_1^* = \mathrm{F}_1 \oplus S_{TIDold} \oplus h(TID_{iold} \| B_i \| K_{gu})$ and $V_1^* = h(ID_{ui} \| R_1^* \| B_i \| TID_{iold} \| K_{gu} \| T_1)$ by using the old temporary id of the user TID_{iold} and the old temporary id of sensor S_{TIDold}. Then it checks whether the computed V_1^* matches the received V_1. If it matches, then the gateway generates a new pseudonym id TID_{inew}^* and updates $TID_{inew} = TID_{inew}^*$.

Then, the gateway generates a session key SK, a random number R_2 and a timestamp T_2 and stores R_2 in G_1 by computing $G_1 = R_2 \oplus K_{gs}$. Further, the gateway computes $F_2 = SK \oplus R_2 \oplus h\left(K_{gs} \| ID_{si} \| N_{gs}\right)$ and $V_2 = h\left(R_2 \| SK \| ID_{si} \| K_{gs} \| N_{gs} \| T_2\right)$. Subsequently it updates K_{gs} by computing $K_{gs}^* = h\left(K_{gs} \| ID_{si}\right)$ and updates $N_{gs} = N_{gs} + 1$. After that, it updates the sensor node temporary id by computing $S_{\mathrm{TID}}^{\mathrm{new}\,*} = ID_{si} \oplus K_{gs}$. Finally, it sends $\left\{ G_1, F_2, V_2, N_{gs}, T_2 \right\}$ as a message M^2 to the sensor node via the public channel.

- **Step 3.** Upon reception, the sensor node checks the freshness of the timestamp T_2 and computes $K_{gs}^* = h^{N_{gs}-N_{ss}-1}\left(K_{gs} \| ID_{si}\right)$. Then it retrieves $R_2^* = G_1 \oplus K_{gs}^*$ and retrieves the session key by computing $SK^* = F_2 \oplus R_2^* \oplus h\left(K_{gs} \| ID_{si} \| \left(N_{gs} - 1\right)\right)$. After that, the sensor node computes $V_2^* = h\left(R_2^* \| SK \| ID_{si} \| K_{gs} \| \left(N_{gs} - 1\right) \| T_2\right)$ and checks whether V_2^* matches with received V_2. The sensor node terminates the session if it does not match. Otherwise, the sensor node update $K_{gs} = h\left(K_{gs} \| ID_{si}\right)$ and make $N_{ss} = N_{gs}$. Then, it generates a random R_3 and timestamp T_3 and computes $F_3 = R_3 \oplus h\left(K_{gs} \| ID_{si} \| N_{ss}\right)$ and $V_3 = h\left(SK \| ID_{si} \| K_{gs} \| N_{ss} \| R_3 \| T_3\right)$. It then updates its temporary id by $S_{\mathrm{TID}}^{\mathrm{new}\,*} = ID_{si} \oplus K_{gs}$. The sensor node then finally sends

$\{ T_3, F_3, V_3\}$ as a message M^3 to the gateway via public channel.

- **Step 4.** The gateway at first checks the validity of the timestamp T_3 after receiving the message from the sensor node and computes $R_3^* = F_3 \oplus h\left(K_{gs}\|ID_{si}\|N_{gs}\right)$ and $V_3^* = h\left(SK\|ID_{si}\|K_{gs}\|N_{gs}\|R_3^*\|T_3\right)$. Then, the gateway compares V_3^* with the received value V_3. A similar value indicates the accomplishment of the mutual authentication of the sensor node at the gateway, otherwise, the session is terminated. After that, the gateway computes $F_4 = SK \oplus TID_{inew} \oplus h\left(R_1\|TID_{ui}\|K_{gu}\|B_i\right)$ and $V_4 = h\left(SK\|TID_{inew}\|ID_{ui}\|R_1\right)$. Then, gateway encloses TID_{inew}^* secretly within $P_1 = TID_{inew} \oplus B_i \oplus R_1$ and sends $\{ P_1, F_4, V_4, T_4\}$ as the message M^4 to the user via the public channel.
- **Step 5.** After receiving message M^4, the user checks the freshness of the timestamp T_4 and retrieves $TID_{inew}^* = P_1 \oplus B_i \oplus R_1$. After that, the user derives $SK^* = F_4 \oplus TID_{inew} \oplus h\left(R_1\|TID_{ui}\|B_i\|K_{gu}\right)$ and computes $V_4^* = h\left(SK\|TID_{inew}\|ID_{ui}\|R_1\right)$. Then, it checks whether V_4^* is matched with the received V_4. If they match, then it updates the user temporary id $TID_{ui} = TID_{inew}$ and sets $Flag = 0$.

4.5 Security Analysis

In this section, we examine the security of our protocol through formal and informal analysis and demonstrate its resistance to desynchronization attacks, reply attacks, and privileged insider attacks, as well as its ability to achieve forward secrecy and identity anonymity.

4.5.1 Informal Security Analysis

The following is a list of prevented attacks of our protocol.

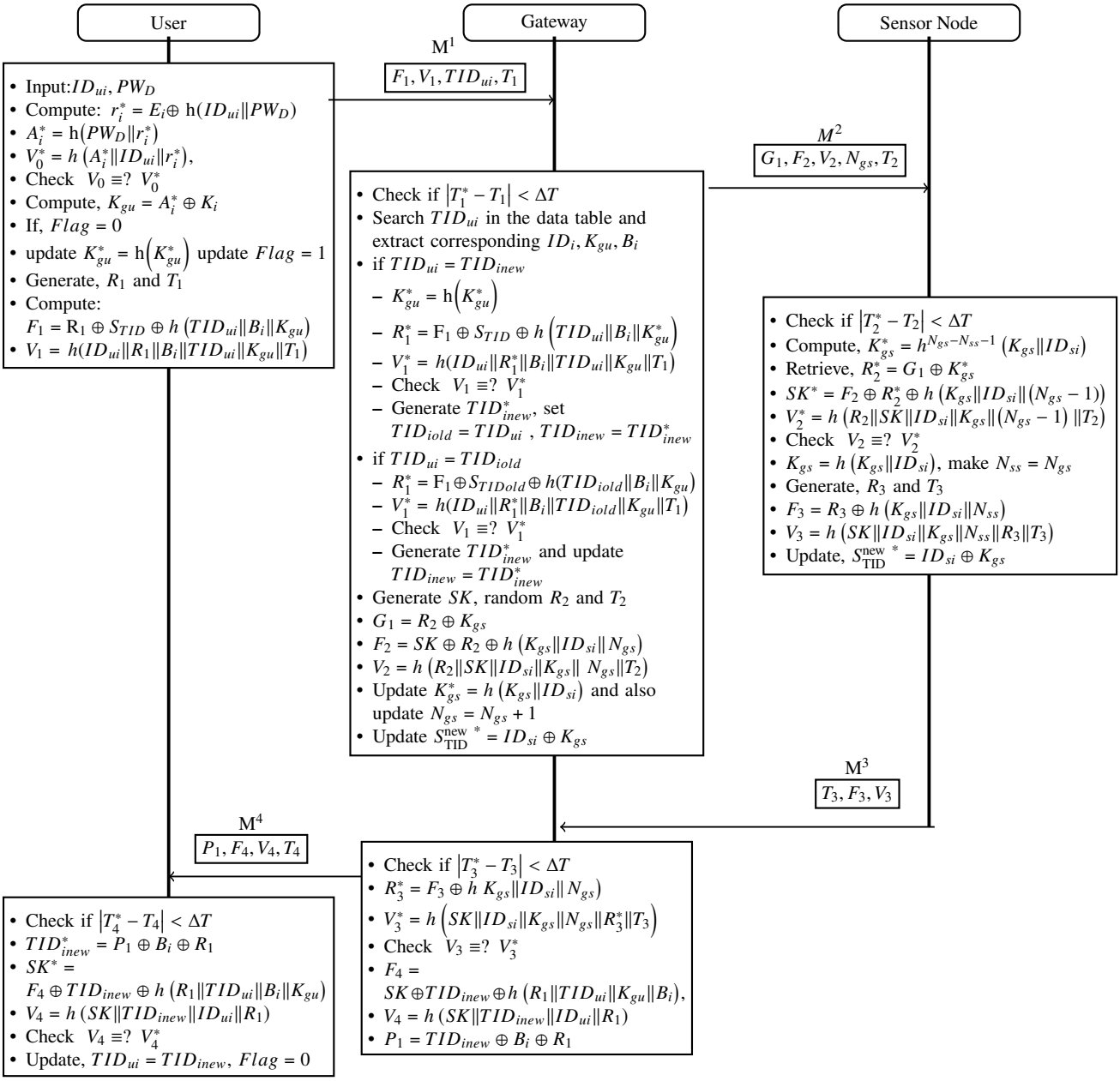

Figure 4.2: Mutual authentication phase of the proposed scheme

- **Desynchronization attack.** In order to achieve sensor and user anonymity and forward secrecy, we use the temporary ids and hash chain technique, respectively, in our proposed protocol. We employ the $Flag$ parameter and two temporary variables N_{gs}, N_{ss} that record the number of updates in the hash chain value to ensure the synchronization of temporary ids and the hash chain values between parties. The different scenarios of desynchronization attacks where an adversary blocks certain messages from reaching their destination are illustrated in Fig. 4.3. In what follows, we discuss how our proposed synchronization mechanism works to resist those scenarios of desynchronization attacks.

  - **Scenario 1.** At the beginning, when $Flag = 0$, the temporary id of the user is the same in the gateway and the user side. The user then updates the K_{gu}, sets $Flag = 1$, and sends M^1 to the gateway. If the message M^1 is blocked by the adversary, K_{gu} on the gateway side does not get updated, although it has already been changed on the user side. Nevertheless, the user and gateway will still be in synchronization despite this attack because the user will not update K_{gu} in its side in the following session since $Flag = 1$ is already set. The $TID_{ui} = TID_{inew}$ equation will be maintained on the gateway side in that scenario and will update the K_{gu} hash chain value to provide coherence between the user and gateway sides for the K_{gu} value. In this way, despite the message M^1 drop, the user and gateway will continue to be in a synchronized state.

  - **Scenario 2.** If the adversary blocks M^2, the temporary user id between the gateway side and the user will not be the same because the temporary user id will be updated to a new temporary user id on the gateway side in step 2 but in the user side it will not be updated because the message is blocked. In this

case when the new session starts, as $Flag = 1$ on the user side, the user will not update K_{gu} and use its temporary user id which is not updated on the user side because of the message block. In this regard, the gateway checks whether $TID_{ui} = TID_{iold}$ and if it holds, the gateway will not update K_{gu} as it is not updated on the user side to have the same K_{gu} value in both sides. Here, the gateway use stored old temporary user id TID_{iold} to authenticate the user on the gateway side. Moreover, if the adversary blocks the message M^2, the hash chain value K_{gs} and temporary sensor node id will not be matched between the gateway side and sensor node side. However, the proposed scheme is adaptive to tackle this kind of desynchronization of values. The serial number N_{gs} records the updates of K_{gs} in the gateway side and N_{ss} records the updates of K_{gs} in the sensor node side. These two values then can be used to find $N_{gs} - N_{ss} - 1$ times the hash operation needed to synchronize K_{gs} value on the sensor node side to authenticate the message received from the gateway. In this process, K_{gs} updated value will be the same in both the sensor node and gateway. Since the sensor node temporary id update is directly dependent on K_{gs} value, synchronization of K_{gs} value will also be used to have the same updated temporary sensor node id ($S_{\text{TID}}^{\text{new}\ *}$) between the gateway and sensor node side.

– **Scenario 3.** If the adversary block the messages M^3 and M^4, it will not create desynchronization of hash chain values K_{gu} and K_{gs} between participated parties. The gateway and user have the same updated K_{gu} and the gateway and sensor side K_{gs} value has also been updated and equivalent to one another. Due to the fact that the temporary sensor id has already been updated in both sensor nodes and gateways in steps 2 and step 3, the new temporary sensor id will

be the same after blocking message M^3 and M^4. However, in this scenario, the temporary user id has already been updated on the gateway side, whereas the user is unable to have the updated temporary user id due to the blocking of messages M^3 and M^4. As a result, when the subsequent session starts, the user side will have $Flag = 1$ and in the gateway the equation $TID_{ui} = TID_{iold}$ holds. So, K_{gu} will be the same and will not get updated at both the user and gateway sides. Hence, the user can prove its authenticity to the gateway with its old temporary user id.

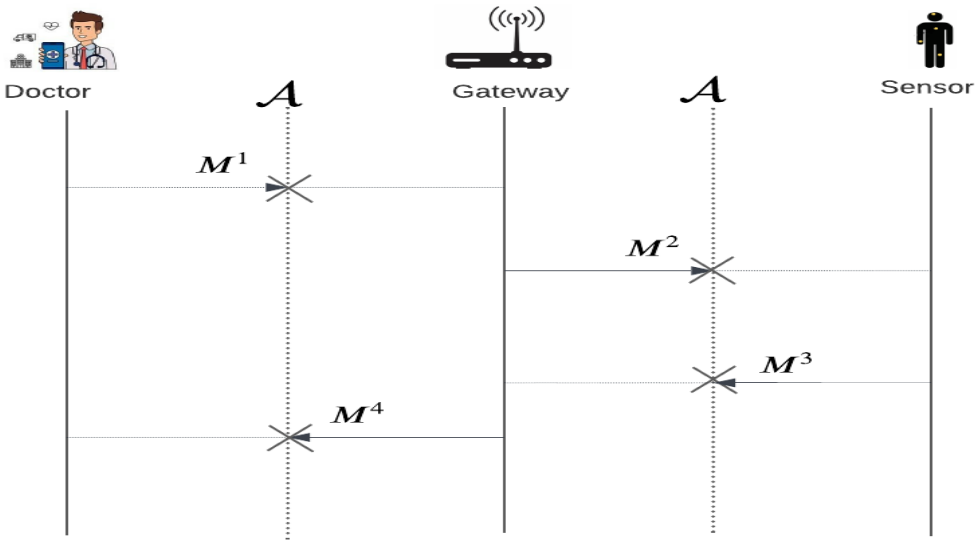

Figure 4.3: Desynchronization attacks on the proposed protocol

- **Replay attacks.** A replay attack can be carried out by intercepting legitimate messages and purposefully delaying or repeating them while pretending to be the originator of the message. We assume the adversary captures the message M^1 and replayed it to the gateway. In our scheme, we use timestamps to prevent replay attacks. After receiving the message from the user, the gateway checks the freshness of the message by $\left|T_1^* - T_1\right| < \Delta T$. It can identify messages from an old session if the transmission

delay exceeds the allowed limit ΔT and consequently, terminates the session. Similarly, the timestamp is included in all other messages M^2, M^3, and M^4 so that it can be secured from reply attacks.

- **Impersonation attack.** An adversary can impersonate a user by providing a valid authentication message $\{ F_1, V_1, PID_{ui}, T_1\}$ to the gateway. However, the adversary can not compute a valid message because the adversary has no knowledge of the user's real identity ID_{ui}, random R_1, and shared secret K_{gu}. An adversary can not find these credentials by intercepting the previous message or extracting secret credentials from the user's mobile device. Similarly, to impersonate the sensor node, an adversary needs to provide a valid authentication message containing$\{ T_3, F_3, V_3\}$. For computing this message, the adversary needs to know the random R_3, the shared secret K_{gs}, and the sensor node real identity ID_{si}. As the adversary could not obtain these values by intercepting previous messages as they are randomized, the adversary can not impersonate the sensor node.

- **Forward secrecy attack.** Forward secrecy implies that if the long-term secret is compromised, the session keys from past sessions can not be retrieved. We assume the long-term secret of the user shared with gateway B_i is compromised. With this knowledge, the attacker still cannot recover past session keys on the user side. On the user side, the session key can be computed by $SK = F_4 \oplus TID_{inew} \oplus h\left(R_1 \| TID_{ui} \| B_i \| K_{gu}\right)$. To compute the previous session key, an adversary needs to have knowledge of K_{gu}^{old} of the previous session and R_1 which is randomly generated in each session. As K_{gu} is updated by $K_{gu} = h\left(K_{gu}^{old}\right)$ in each session where $h()$ is a one-way function, the adversary can not efficiently retrieve K_{gu}^{old} from the current K_{gu}. Similarly, on the sensor node side, the session key can be obtained by computing

$SK = F_2 \oplus R_2 \oplus h\left(K_{gs} \| ID_{si} \| \left(N_{gs} - 1\right)\right)$. To obtain the session key of the previous session, the adversary needs to have knowledge about randomly generated R_2 and K_{gs}^{old} of the previous session. As K_{gs} is also updated by $K_{gs} = h\left(K_{gs}^{old}\right)$ each session, it is infeasible for adversary to obtain K_{gs}^{old} from the current K_{gs}. As a result, in both of the cases, the adversary is not able to obtain the previous session key because of the use of the hash chained value K_{gu} and K_{gs}.

- **Anonymity attack.** To protect the real identities of the user and sensor, temporary user id TID_{ui} and temporary sensor node id S_{TID} are used. The adversary also could not retrieve the user's real identity and track it with the knowledge of TID_{ui} as TID_{ui} is generated randomly and updated randomly each session. Similarly, the temporary sensor node is updated after completing authentication by computing $S_{\text{TID}}^{\text{new}\,*} = ID_{si} \oplus K_{gs}$. Since K_{gs} is a hash value that gets updated each session, it is infeasible for adversaries to track the sensor's actions. Consequently, the proposed scheme protects the real identity of the user ID_{ui} and the sensor ID_{si}.

- **Stolen device attack.** In this attack, the adversary is successful if she is able to get the secret credentials of the user, i.e., the password PW_D and real identity ID_{ui}. We assume the adversary has access to a user's device and can extract the stored values $\{V_0, TID_{ui}, E_i, B_i, K_i\}$. In our protocol, the password PW_D is masked by $E_i = h(ID_{ui}||PW_D) \oplus r_i$ during the registration phase. The adversary can obtain E_i but cannot get PW_D as r_i random numbers which are unknown to the adversary and not stored during the registration. Additionally, because of the one-wayness of the hash function, the adversary can not recover user password PW_D. Also, the user id ID_{ui} is not stored in plaintext during the registration phase. In our protocol, the user id ID_{ui} is masked within $E_i = h(ID_{ui}||PW_D) \oplus r_i$, $B_i = h(ID_{ui}||b_i)$ and

$V_0 = h(A_i||ID_{ui}||r_i)$. As the adversary has no knowledge about random r_i, b_i and password PW_D, the adversary is not able to compute the user id ID_{ui} even if she uses the extracted secret values from the user device.

4.5.2 Formal Security Analysis

We validate the security attributes of our proposed scheme using ProVerif. ProVerif is a popular automated tool for verifying security properties of cryptographic protocols [21] [24] [10]. It supports a wide range of symmetric and asymmetric cryptography protocols, including hashing, digital signatures, elliptic curve problems, and Diffie-Hellman key agreements [36].

Using ProVerif, we give evidence that that our protocol achieves mutual authentication between the participants and the secrecy of the established session key. Users, gateways, and sensor nodes are the three participants in ProVerif's parallel execution process. The code for modeling our protocol is given in Appendix A.

As shown in Figure 4.4, we define eight events, and each event indicates either the beginning or completion of authentication between two parties. For instance "event UiGWNbegi(participant)" indicates the beginning of the authentication process between the user and the gateway, whereas "event UiGWNathenticated(participant)" denotes the completion of the authentication process. Followed by a query that triggers these two events such as "query x_11: participant; inj-event(UiGWNauthenticated (x_11)) ==> inj-event(UiGWNbegin(x_11))". The verification results is shown in Figure 4.5 where "Query inj-event(UiGWNauthenticated((x_11)) ==> inj-event(UiGWNbegin (x_11)) is true." which indicates that authentication between user and the gateway is achieved successfully. Similarly, the rest of the 3 query results show the authentication between gateway-user,

```
(*--Authentication queries--*)
event UiGWNbegin(participant).
event UiGWNauthenticated(participant).
event GWNUibegin(participant).
event GWNUiauthenticated(participant).
event GWNSNbegin(participant).
event GWNSNauthenticated(participant).
event SNGWNbegin(participant).
event SNGWNauthenticated(participant).
query x_11: participant; inj-event(UiGWNauthenticated(x_11)) ==> inj-
event(UiGWNbegin(x_11)).
query x_22: participant; inj-event(GWNUiauthenticated(x_22)) ==> inj-
event(GWNUibegin(x_22)).
query x_33: participant; inj-event(GWNSNauthenticated(x_33)) ==> inj-
event(GWNSNbegin(x_33)).
query x_44: participant; inj-event(SNGWNauthenticated(x_44)) ==> inj-
event(SNGWNbegin(x_44)).
(*--Session Key Queries--*)
free uSK: bitstring [private].
free gSK : bitstring [private].
free sSk : bitstring [private].
query attacker(uSK);
      attacker(gSK);
      attacker(sSk).
```

Figure 4.4: Events and queries

gateway-sensor, and sensor-gateway is done successfully which proves our protocol achieves mutual authentication.

We further check the secrecy of the session key across the three parties where “uSk” is the session key at the user side, and “gSK” and “sSK” are the session keys at the gateway and sensor node sides, respectively. The phrase “query attacker(uSK)” indicates inquiring about the confidentiality of the session key on the user side, as seen in Figure 4.4. On the other hand, “ Query not attacker(uSK[]) is true.” indicates that the session key has not been compromised on the user side. As we can see in Figure 4.5, the confidentiality of the session key is preserved in the proposed protocol because it has not been disclosed at any of parties’ sides.

```
--------------------------------------------------------------
Verification summary:

Query inj-event(UiGWNauthenticated(x_11)) ==> inj-event(UiGWNbegin(x_11)) is true.

Query inj-event(GWNUiauthenticated(x_22)) ==> inj-event(GWNUibegin(x_22)) is true.

Query inj-event(GWNSNauthenticated(x_33)) ==> inj-event(GWNSNbegin(x_33)) is true.

Query inj-event(SNGWNauthenticated(x_44)) ==> inj-event(SNGWNbegin(x_44)) is true.

Query not attacker(uSK[]) is true.

Query not attacker(gSK[]) is true.

Query not attacker(sSk[]) is true.

--------------------------------------------------------------
```

Figure 4.5: ProVerif results

Chapter 5

Performance Evaluation

Due to IoT's resource-constrained nature, designing lightweight authentication that not only uses limited resources efficiently but also delivers enough security is one of the challenges. In this chapter, we discuss the costs associated with the computation, communication, and storage of our proposed schemes. In the end, we investigate how well our proposed scheme withstands potential threats by comparing it with other relevant schemes. We consider the authentication phase for comparing computation and communication overhead, as user and sensor node registration is relatively simple and not frequently executed. Table 5.1 summarizes the findings of our proposed scheme's performance.

5.1 Computational Cost

The computing overhead consists of all the cryptographic operations required on the user, gateway, and sensor sides to accomplish mutual authentication and key agreement.

The execution time of performing hash operation (T_h) is $0.0121ms$ and symmetric encryption/decryption is (T_{ed}) $0.0428ms$ on Raspberry Pi 4 Model B/8GB with a 1.5 GHz 64-bit Quad-core ARM Cortex-A72 processor where SHA-256 over 256 bytes of data for the

Table 5.1: Computation, communication and storage costs of the proposed scheme

Description	Value
User computation	8 hashes
Gateway computation	10 hashes
Sensor computation	5 hashes
Communication user-gateway	104 bytes
Communication gateway-sensor	104 bytes
User storage	80 bytes
Gateway storage	$N_i \times 80 + N_s \times 80$ bytes
Sensor storage	48 bytes

hash function and AES-CBC mode with a key size of 128 bits for symmetric encryption are considered [37]. We do not take into account the cost of XOR operations because its execution time is negligible. Table 5.2 shows the breakdown of the computational cost in terms of cryptography operations at the user, gateway, and sensor node sides for our proposed scheme with other authentication protocols. For comparison, we chose the five authentication protocols as they are lightweight and rely on only symmetric key cryptography [21], [17], [6], [20] and [19]. Additionally, those five protocols comprise three parties involved in the mutual authentication process similar to our proposed scheme. Figure 5.1 shows the proposed scheme's total execution time in comparison to the same related protocols. Except for Mehdi *et al.*'s protocol [6], our proposed protocol has the lowest total execution time. Although the protocol proposed by Mehedi *et al.* [6] is computationally more efficient than ours, it cannot provide all of the security features that our proposed scheme can provide. Specifically, as we show in Chapter 3, the scheme is vulnerable to desynchronization attacks and session key disclosure attacks.

Table 5.2: Comparison of computation cost

Scheme	**User**	**Gateway**	**Sensor**	**Total**
[21]	$10T_h$	$14T_h$	$5T_h$	$29T_h$
[17]	$3T_{ed}$+$4T_h$	$5T_{ed}$+$2T_h$	$2T_{ed}$+$1T_h$	$10T_{ed}$+$7T_h$
[6]	$3T_h$	$3T_h$	$2T_h$	$8T_h$
[20]	$12T_h$	$16T_h$	$6T_h$	$34T_h$
[19]	$3T_{ed}$+$8T_h$	$2T_{ed}$+$4T_h$	$2T_{ed}$+$4T_h$	$7T_{ed}$+$16T_h$
Proposed	$8T_h$	$10T_h$	$5T_h$	$23T_h$

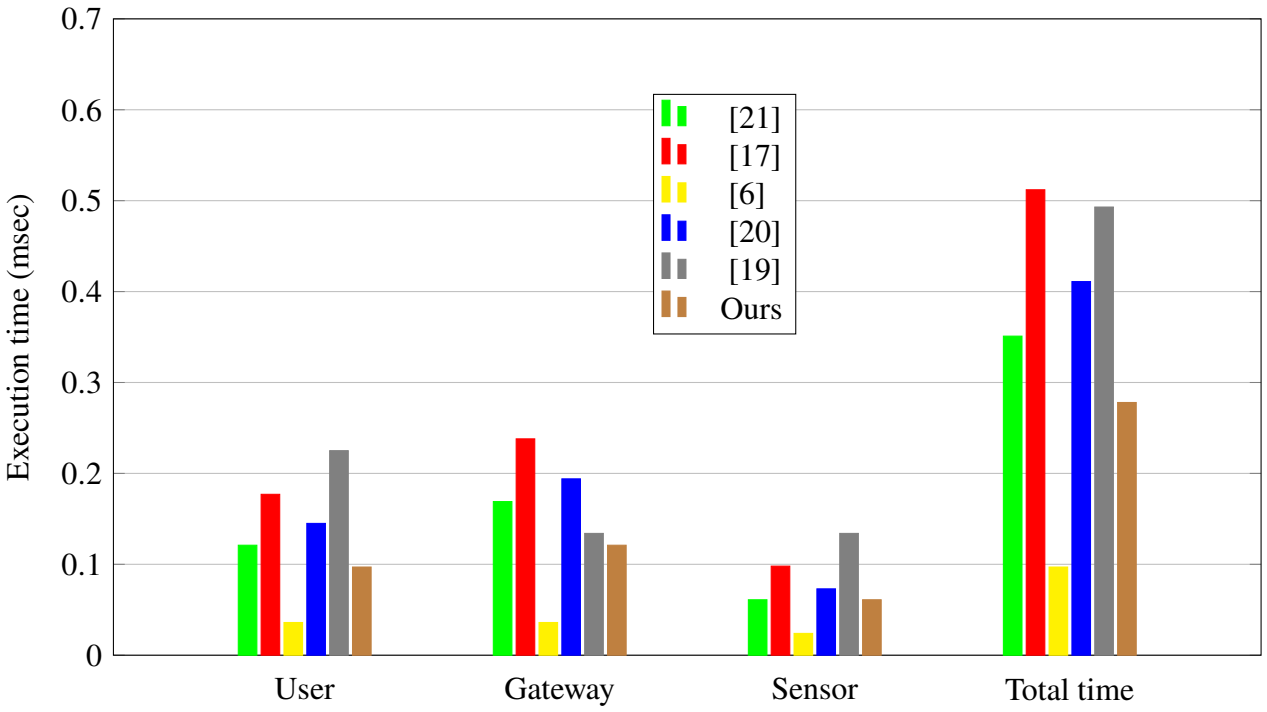

Figure 5.1: Comparison of the total execution time

5.2 Communication Cost

The communication overhead is the total size of messages that need to be exchanged in order to complete the authentication process. To calculate the communication cost of

the proposed and other related schemes, the size of identities, random numbers, sequence numbers, and nonce are considered 128 bits, whereas the timestamp is fixed at 32 bits. Moreover, SHA-256 with a truncated output size 128 bits and AES-CBC encryption with a key size of 128 bits are used.

In the mutual authentication phase, user sends M^1={ F_1, V_1, TID_{ui}, T_1} to gateway which requires (128 + 128 + 128 + 32) = 416 bits. For sending M^2=$\{ G_1, F_2, V_2, N_{gs}, T_2\}$ from gateway to sensor, it requires (128 + 128 + 128 + 128 + 32) = 544 bits. Similarly, to send M^3 = { T_3, F_3, V_3} from sensor to gateway and M^4 = { P_1, F_4, V_4, T_4} from gateway to user, it requires 288 bits and 416 bits respectively. Therefore, the total communication cost of our protocol is 1664 bits or 208 bytes. As depicted in Figure 5.2, our proposed protocol has less communication cost when compared to [21] [20] [6] [38]. Although [19] has a lower communication cost than ours, it does not provide the sensor node anonymity that our proposed scheme offers. Sensor node anonymity is of significant importance as it preserves the privacy of patients' health conditions.

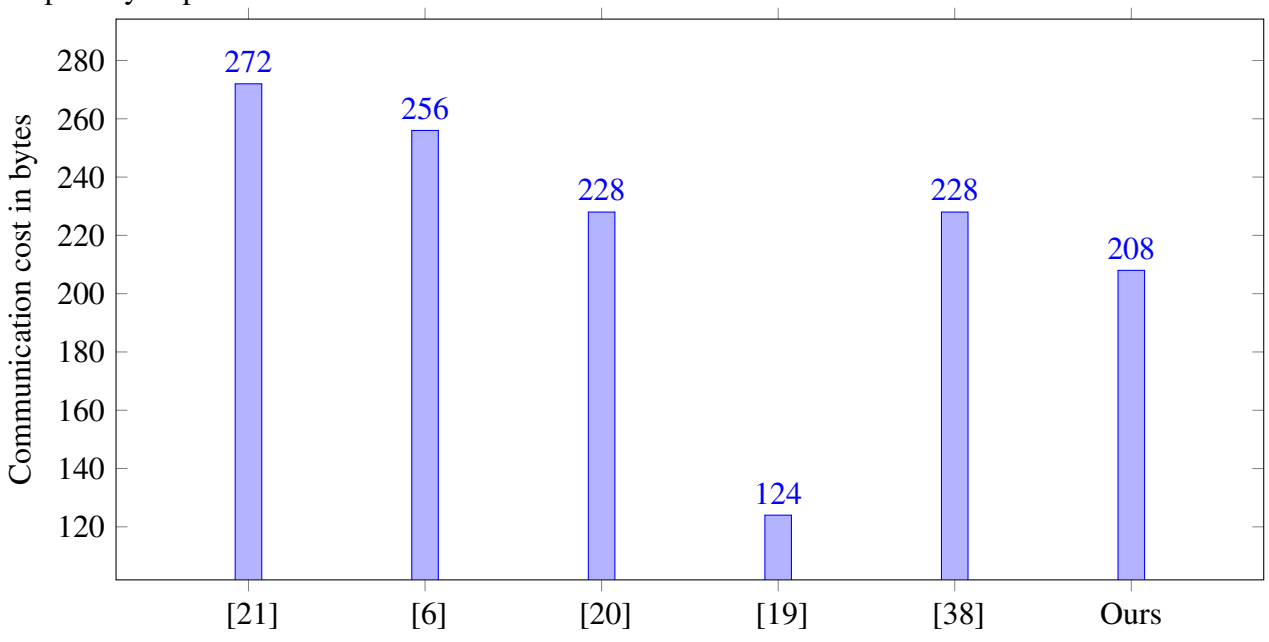

Figure 5.2: Comparison of communication cost

5.3 Storage Cost

During the registration phase, the user, sensor, and gateway need to store secret credentials so that they can use those secret credentials to prove their legitimacy or authenticate other participating parties during the mutual authentication phase. Storage cost refers to the memory size required to store those secret parameters. For the user side of our proposed scheme, there is a requirement to store $\{V_0, TID_{ui}, E_i, B_i, K_i\}$ which requires the storage cost (128 + 128 + 128 + 128 + 128) = 640 bits. Similarly, the sensor node needs to store $\{K_{gs}, N_{gs}, S_{TID}\}$ in its memory which requires the storage cost (128 + 128 + 128) = 384 bits. In this case, the gateway side needs to keep both users and sensor node secret parameters for authentication purposes. As a result, the gateway also needs to store $\{ID_{ui}, TID_{inew}, TID_{iold}, K_{gu}, B_i, K_{gs}, N_{gs}, S_{TID}, S_{TIDold}, ID_{si}\}$, which requires total storage cost (128 + 128 + 128 + 128 + 128 + 128 + 128 + 128 + 128 + 128) = 1280 bits. It can be estimated that each gateway needs to store $N_i \times 80 + N_s \times 80$ bytes, where N_i is a number of registered users, N_s is the number of the registered sensor node.

5.4 Security Features Comparison

In Table 5.3, we compare our proposed protocol's security properties with other related protocols. The notation ✓ indicates that the scheme is secure or that a security feature is provided, whereas the notation × indicates that the scheme is not secure against that attack or that the security feature is not available. As we can see from Table 5.3, our proposed scheme provides greater protection against potential threats.

Table 5.3: Comparison of security parameters with other relevant schemes

Security Property	[21]	[17]	[6]	[20]	[38]	[19]	Ours
Mutual authentication	✓	×	✓	✓	✓	✓	✓
Forward secrecy	×	×	×	×	×	✓	✓
Resilience to impersonation attack	✓	✓	✓	✓	✓	✓	✓
Resilience to reply attack	✓	✓	✓	×	✓	✓	✓
Resilience to desynchronization attack	×	×	×	×	×	✓	✓
User anonymity	✓	✓	✓	✓	✓	✓	✓
Sensor node anonymity	×	×	✓	×	×	×	✓
Resilience to stolen device attack	✓	✓	✓	✓	✓	✓	✓

Chapter 6

Final Discussion and Conclusion

In this thesis, we first analyze Mehedi *et al.* 's lightweight user authentication scheme for healthcare applications and we conclude that although the scheme is computationally efficient, we show that it is vulnerable to desynchronization attacks while achieving identity anonymity. Additionally, session key disclosure attacks have been shown to be possible. To counter these threats, we propose a lightweight authentication protocol that is resistant to desynchronization attacks and upholds forward secrecy and anonymity.

In our proposed scheme, we only use hash and XOR operation for efficiency to achieve mutual authentication and session key agreement. Broadly accepted ProVerif simulation tools are used to formally analyze the security of our protocol. In addition, informal analysis demonstrates the resilience of our system against several security threats. Finally, we compare the performance of our proposed scheme with other relevant schemes and the results show that it outperformed most of the existing protocols while achieving higher security. Therefore, the proposed protocol can be employed to securely access patients' crucial data remotely for the healthcare environment. Additionally, this authentication protocol can also be applied in a number of scenarios where lightweight authentication

is required, such as IoT-based smart homes. For future work, we aim to implement our proposed protocol in an actual IoT environment and test how it performs to tackle different kinds of attacks in a practical scenario.

Appendix A

In what follows, we provide the modeling code of our protocol using Pro-Verif environment.

```
(*--Normal scenrio is consider, no desynchronization scenrio is
                                        considered--*)
(*--The two public channel--*)
free ch1: channel.
free ch2: channel.

(*--The basic type--*)
type participant.
type nonce.
type key.
type timestamp.
type integer.

(*--Hash operation--*)
fun Hash(bitstring): bitstring.
(*--XOR operation--*)
fun XOR(bitstring, bitstring): bitstring.
equation forall x: bitstring, y: bitstring;
XOR(XOR(x, y), y) = x.
(*--Concat operation--*)
fun Con(bitstring, bitstring): bitstring.
(*--Check timestamp Fresh operation--*)
fun checktimestampfresh(timestamp, bool): bool
reduc forall T: timestamp;
checktimestampfresh(T, true) = true
otherwise forall T: timestamp;
checktimestampfresh(T, false) = false.
(*--Shared key encryption--*)
fun encrypt(bitstring,key): bitstring.
reduc forall x: bitstring, y: key;
decrypt(encrypt(x,y),y) = x.

(*--Type convertion--*)
fun keytobit(key): bitstring    [data,typeConverter].
```

```
fun nontobit(nonce): bitstring [data,typeConverter].
fun bittokey(bitstring): key   [data,typeConverter].
fun timetobit(timestamp): bitstring [data,typeConverter].
fun inttobit(integer): bitstring [data,typeConverter].
fun addk(integer,integer): integer.
reduc forall x: integer, k: integer;
mink(addk(x,k),k) = x.

(*--The basic variables--*)
free user,GWN,SN: participant.
free TIDui: bitstring.
free STID : bitstring.
free Bi: bitstring[private].
free Kgu: bitstring[private].
free Kgs: bitstring[private].
free Ngs: integer.
free Nss: integer.

(*--Authentication queries--*)
event UiGWNbegin(participant).
event UiGWNauthenticated(participant).
event GWNUibegin(participant).
event GWNUiauthenticated(participant).
event GWNSNbegin(participant).
event GWNSNauthenticated(participant).
event SNGWNbegin(participant).
event SNGWNauthenticated(participant).
query x_11: participant; inj-event(UiGWNauthenticated(x_11)) ==> inj-
                                    event(UiGWNbegin(x_11)).
query x_22: participant; inj-event(GWNUiauthenticated(x_22)) ==> inj-
                                    event(GWNUibegin(x_22)).
query x_33: participant; inj-event(GWNSNauthenticated(x_33)) ==> inj-
                                    event(GWNSNbegin(x_33)).
query x_44: participant; inj-event(SNGWNauthenticated(x_44)) ==> inj-
                                    event(SNGWNbegin(x_44)).
(*--Session Key Queries--*)
free uSK: bitstring [private].
free gSK : bitstring [private].
free sSk : bitstring [private].
query attacker(uSK);
      attacker(gSK);
      attacker(sSk).

(*--Role of  User--*)
let processUser(IDui: bitstring, PWd: bitstring, Ei: bitstring,  STID,
                                    Ki: bitstring, V0: bitstring) =
    let ri = XOR(Ei, Hash(Con(IDui,PWd))) in
    let Ai = Hash(Con(PWd,ri)) in
      let V0' = Hash(Con(Con(Ai,IDui),ri))  in
        if V0'= V0 then
        event GWNUibegin(GWN);
      let Kgu = XOR(Ai,Ki) in
```

```
      let Kgu = Hash(Kgu)  in
        new R1: nonce;
        new T1: timestamp;
        let F1 = XOR(XOR(nontobit(R1),STID),Hash(Con(Con(TIDui,Bi),Kgu))
                                    ) in
        let V1 = Hash(Con(Con(Con(Con(Con(IDui,nontobit(R1)),Bi),Kgu),
                                    TIDui),timetobit(T1))) in
          out(ch1, (F1, TIDui, V1, T1, true));

          in(ch1, (P1:bitstring, F4:bitstring, V4:bitstring, T4:
                                              timestamp, isTrue: bool));
          if checktimestampfresh(T4, isTrue) then
        let TIDinew = XOR(XOR(P1,Bi),nontobit(R1)) in
      let uSK = XOR(XOR(F4,TIDinew),Hash(Con(Con(Con(TIDui,Bi),Kgu),
                                    nontobit(R1)))) in
        let V4' = Hash(Con(Con(Con(uSK,TIDinew),IDui),nontobit(R1))) in
        if V4' =V4 then
      let TIDui = TIDinew in
          event UiGWNauthenticated(user).

(*--Role of  GWN--*)

let processGWN(IDui: bitstring, Ngs:integer, IDsi: bitstring) =
        in(ch1, ( F1: bitstring, V1: bitstring, TIDui: bitstring, T1:
                                            timestamp, isTrue: bool));
        if checktimestampfresh(T1, isTrue) then
    let Kgu = Hash(Kgu) in
        let R1 = XOR(Con(F1,STID),Hash(Con(Con(TIDui,Bi),Kgu))) in
        event UiGWNbegin(user);
       let V1' = Hash(Con(Con(Con(Con(Con(IDui,R1),Bi),Kgu),TIDui),
                                          timetobit(T1))) in
        if V1'=V1 then
      new TIDinew: bitstring;
        new R2: nonce;
        new T2: timestamp;
      new gSK: bitstring;
      new k: integer;
        event SNGWNbegin(SN);
      let G1 = XOR (nontobit(R2),Kgs) in
      let F2  = XOR(XOR(gSK,nontobit(R2)),Hash(Con(Con(Kgs,IDsi),
                                    inttobit(Ngs)))) in
        let V2 =  Hash(Con(Con(Con(Con(Con(gSK,nontobit(R2)),IDsi),Kgs),
                                    inttobit(Ngs)),timetobit(T2))) in
      let Kgs=Hash(Con(Kgs,IDsi)) in
        let Ngs=addk(Ngs,k) in
      let STIDnew = XOR (IDsi,Kgs) in
        out(ch2, (G1, F2, V2, Ngs, T2, true));

        in(ch2, (F3: bitstring, V3: bitstring, T3: timestamp, isTrue:
                                          bool));
         if checktimestampfresh(T3, isTrue) then
      let R3 = XOR(F3,Hash(Con(Con(Kgs,IDsi),inttobit(Ngs)))) in
```

```
      let V3'= Hash(Con(Con(Con(Con(Con(IDsi,gSK),Kgs),R3),inttobit(Ngs)
                                    ),timetobit(T3))) in
        if V3' = V3 then
        event GWNSNauthenticated(GWN);
        new T4: timestamp;
      let F4 = XOR(XOR(gSK,TIDinew),Hash(Con(Con(Con(TIDui,Bi),Kgu),R1))
                                    ) in
        let V4 = Hash(Con(Con(Con(gSK,TIDinew),IDui),R1)) in
      let P1 =  XOR(XOR(TIDinew,Bi),R1) in
        out(ch1, (P1,F4,V4,T4,true));
        event GWNUiauthenticated(GWN).

(*--Role of sensor --*)
let processSN(IDsi:bitstring) =
        in(ch2,  (G1:bitstring, F2: bitstring, V2:bitstring, Ngs:integer
                                              , T2:timestamp, isTrue:bool)
                                              );
        if checktimestampfresh(T2, isTrue) then
        event GWNSNbegin(GWN);
      let R2 = XOR(G1,Kgs) in
      new k: integer;
      let sSK = XOR(XOR(F2,R2),Hash(Con(Con(Kgs,IDsi),inttobit(mink(Ngs,
                                    k))))) in
        let V2'=  Hash(Con(Con(Con(Con(Con(sSK,R2),IDsi),Kgs),inttobit(
                                    mink(Ngs,k))),timetobit(T2))) in
        if V2'= V2 then
      let Kgs=Hash(Con(Kgs,IDsi)) in
        let Nss=Ngs in
    new T3: timestamp;
    new R3: nonce;
    let F3= XOR(nontobit(R3),Hash(Con(Con(Kgs,IDsi),inttobit(Nss)))) in
    let V3= Hash(Con(Con(Con(Con(Con(IDsi,sSK),Kgs),nontobit(R3)),
                                    inttobit(Nss)),timetobit(T3))) in
    let STIDnew = XOR (IDsi,Kgs) in
        out(ch2, (F3, V3, T3, true));
        event SNGWNauthenticated(SN).

(*--Start process--*)
process
  new IDui: bitstring;
  new PWd: bitstring;
  new Ei: bitstring;
  new IDsi: bitstring;
  new Ki: bitstring;
  new Ngs: integer;
  let ri = XOR(Ei, Hash(Con(IDui,PWd))) in
  let Ai = Hash(Con(PWd,ri))  in
  let V0 = Hash(Con(Con(Ai,IDui),ri)) in
  (

          (!processUser(IDui, PWd, Ei,  STID,  Ki, V0)) |
```

```
        (!processGWN(IDui,Ngs,IDsi)) |

        (!processSN(IDsi))
)
```

References

[1] S. B. Baker, W. Xiang, and I. Atkinson, "Internet of things for smart healthcare: Technologies, challenges, and opportunities," *Ieee Access*, vol. 5, pp. 26 521–26 544, 2017.

[2] N. Jordan, "Staggering stats on healthcare iot innovation",," *Data Driven Investor, 2020* , vol. 8, 7.

[3] F. B. Insights, "Iomt market worth usd 187.60 billion by 2028, with 29.5% cagr: Market projection by technology, major key players, growth factors, revenue, cagr, regional analysis, industry forecast to 2028," Oct 2021. [Online]. Available: https://www.globenewswire.com/news-release/2021/10/28/2322459/0/en/IoMT-Market-worth-USD-187-60-Billion-by-2028-.html

[4] I. Bhardwaj, A. Kumar, and M. Bansal, "A review on lightweight cryptography algorithms for data security and authentication in iots," in *2017 4th International Conference on Signal Processing, Computing and Control (ISPCC)*. IEEE, 2017, pp. 504–509.

[5] M. El-Hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni, "A survey of internet of things (iot) authentication schemes," *Sensors*, vol. 19, no. 5, p. 1141, 2019.

[6] M. Masud, G. S. Gaba, K. Choudhary, M. S. Hossain, M. F. Alhamid, and G. Muhammad, "Lightweight and anonymity-preserving user authentication scheme for iot-based healthcare," *IEEE Internet of Things Journal*, 2021.

[7] I. Mashal, O. Alsaryrah, T.-Y. Chung, C.-Z. Yang, W.-H. Kuo, and D. P. Agrawal, "Choices for interaction with things on internet and underlying issues," *Ad Hoc Networks*, vol. 28, pp. 68–90, 2015.

[8] M. Saadeh, A. Sleit, M. Qatawneh, and W. Almobaideen, "Authentication techniques for the internet of things: A survey," in *2016 cybersecurity and cyberforensics conference (CCC)*. IEEE, 2016, pp. 28–34.

[9] S. Emerson, Y.-K. Choi, D.-Y. Hwang, K.-S. Kim, and K.-H. Kim, "An oauth based authentication mechanism for iot networks," in *2015 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2015, pp. 1072–1074.

[10] J. Ryu, D. Kang, H. Lee, H. Kim, and D. Won, "A secure and lightweight three-factor-based authentication scheme for smart healthcare systems," *Sensors*, vol. 20, no. 24, p. 7136, 2020.

[11] T. Kothmayr, C. Schmitt, W. Hu, M. Brünig, and G. Carle, "A dtls based end-to-end security architecture for the internet of things with two-way authentication," in *37th Annual IEEE Conference on Local Computer Networks-Workshops*. IEEE, 2012, pp. 956–963.

[12] T. Nandy, M. Y. I. B. Idris, R. M. Noor, L. M. Kiah, L. S. Lun, N. B. A. Juma'at, I. Ahmedy, N. A. Ghani, and S. Bhattacharyya, "Review on security of internet of things authentication mechanism," *IEEE Access*, vol. 7, pp. 151 054–151 089, 2019.

[13] M. Turkanović, B. Brumen, and M. Hölbl, "A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the internet of things notion," *Ad Hoc Networks*, vol. 20, pp. 96–112, 2014.

[14] C.-C. Chang and H.-D. Le, "A provably secure, efficient, and flexible authentication scheme for ad hoc wireless sensor networks," *IEEE Transactions on wireless communications*, vol. 15, no. 1, pp. 357–366, 2015.

[15] X. Li, J. Peng, J. Niu, F. Wu, J. Liao, and K.-K. R. Choo, "A robust and energy efficient authentication protocol for industrial internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1606–1615, 2017.

[16] P. Kumar, S.-G. Lee, and H.-J. Lee, "E-sap: Efficient-strong authentication protocol for healthcare applications using wireless medical sensor networks," *Sensors*, vol. 12, no. 2, pp. 1625–1647, 2012.

[17] D. He, N. Kumar, J. Chen, C.-C. Lee, N. Chilamkurti, and S.-S. Yeo, "Robust anonymous authentication protocol for health-care applications using wireless medical sensor networks," *Multimedia Systems*, vol. 21, no. 1, pp. 49–60, 2015.

[18] F. Wu, L. Xu, S. Kumari, and X. Li, "An improved and anonymous two-factor authentication protocol for health-care applications with wireless medical sensor networks," *Multimedia Systems*, vol. 23, no. 2, pp. 195–205, 2017.

[19] J. Srinivas, D. Mishra, and S. Mukhopadhyay, "A mutual authentication framework for wireless medical sensor networks," *J. Med. Syst.*, vol. 41, no. 5, May 2017.

[20] R. Amin, S. H. Islam, G. Biswas, M. K. Khan, and N. Kumar, "A robust and anonymous patient monitoring system using wireless medical sensor networks," *Future Generation Computer Systems*, vol. 80, pp. 483–495, 2018.

[21] F. Wu, X. Li, A. K. Sangaiah, L. Xu, S. Kumari, L. Wu, and J. Shen, "A lightweight and robust two-factor authentication scheme for personalized healthcare systems using wireless medical sensor networks," *Future Generation Computer Systems*, vol. 82, pp. 727–737, 2018.

[22] W. Li, B. Li, Y. Zhao, P. Wang, and F. Wei, "Cryptanalysis and security enhancement of three authentication schemes in wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[23] M. H. Ibrahim, S. Kumari, A. K. Das, M. Wazid, and V. Odelu, "Secure anonymous mutual authentication for star two-tier wireless body area networks," *Computer methods and programs in biomedicine*, vol. 135, pp. 37–50, 2016.

[24] Z. Xu, C. Xu, W. Liang, J. Xu, and H. Chen, "A lightweight mutual authentication and key agreement scheme for medical internet of things," *IEEE Access*, vol. 7, pp. 53 922–53 931, 2019.

[25] G. Sharma and S. Kalra, "A lightweight user authentication scheme for cloud-iot based healthcare services," *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, vol. 43, no. 1, pp. 619–636, 2019.

[26] S. Kalra and S. K. Sood, "Secure authentication scheme for iot and cloud servers," *Pervasive and Mobile Computing*, vol. 24, pp. 210–223, 2015.

[27] S. Kumari, M. Karuppiah, A. K. Das, X. Li, F. Wu, and N. Kumar, "A secure authentication scheme based on elliptic curve cryptography for iot and cloud servers," *The Journal of Supercomputing*, vol. 74, no. 12, pp. 6428–6453, 2018.

[28] Z. Xu, C. Xu, H. Chen, and F. Yang, "A lightweight anonymous mutual authentication and key agreement scheme for WBAN," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 14, May 2019. [Online]. Available: https://doi.org/10.1002/cpe.5295

[29] B. A. Alzahrani, A. Irshad, A. Albeshri, and K. Alsubhi, "A provably secure and lightweight patient-healthcare authentication protocol in wireless body area networks," *Wireless Personal Communications*, vol. 117, no. 1, pp. 47–69, 2021.

[30] C.-H. Liu and Y.-F. Chung, "Secure user authentication scheme for wireless healthcare sensor networks," *Computers & Electrical Engineering*, vol. 59, pp. 250–261, 2017.

[31] S. Challa, A. K. Das, V. Odelu, N. Kumar, S. Kumari, M. K. Khan, and A. V. Vasilakos, "An efficient ecc-based provably secure three-factor user authentication and key agreement protocol for wireless healthcare sensor networks," *Computers & Electrical Engineering*, vol. 69, pp. 534–554, 2018.

[32] R. AlTawy, "Cryptanalysis of some aes-based cryptographic primitives," Ph.D. dissertation, Concordia University, March 2016, unpublished. [Online]. Available: https://spectrum.library.concordia.ca/id/eprint/981311/

[33] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, no. 11, pp. 770–772, 1981.

[34] H. Zhang, X. Li, and R. Ren, "A novel self-renewal hash chain and its implementation," in *2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, vol. 2. IEEE, 2008, pp. 144–149.

[35] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2001, pp. 453–474.

[36] B. Blanchet, B. Smyth, V. Cheval, and M. Sylvestre, "Proverif 2.00: automatic cryptographic protocol verifier, user manual and tutorial," 2018.

[37] M. Nakkar, R. AlTawy, and A. Youssef, "Gase: A lightweight group authentication scheme with key agreement for edge computing applications," *IEEE Internet of Things Journal*, 2022.

[38] A. K. Das, "A secure and robust temporal credential-based three-factor user authentication scheme for wireless sensor networks," *Peer Peer Netw. Appl.*, vol. 9, no. 1, pp. 223–244, Jan. 2016.