symmetry

Article

An Authentication Protocol for the Medical Internet of Things

Nagwa El-Meniawy [1], Mohamed R. M. Rizk [2], Magdy A. Ahmed [3] and Mohamed Saleh [4,*]

[1] Department of Computer Engineering, Pharos University, Alexandria 21649, Egypt; eng.nagwa.elmeniawy@gmail.com
[2] Department of Electrical Engineering, Alexandria University, Alexandria 21544, Egypt; mrmrizk@ieee.org
[3] Department of Computer Engineering, Alexandria University, Alexandria 21544, Egypt; magdy@alexu.edu.eg
[4] Department of Electrical Engineering, Pharos University, Alexandria 21649, Egypt
* Correspondence: mohamed.saleh@pua.edu.eg

Abstract: The progress in biomedical sensors, Internet of Things technologies, big data, cloud computing, and artificial intelligence is leading the development of e-health medical systems, offering a range of new and innovative services. One such service is remote patient monitoring, where medical professionals are able to collect and examine a patient's medical data remotely. Of course, in these systems, security and privacy are of utmost importance and we need to verify the identities of system users before granting them access to sensitive patient-related data. To this end, several authentication protocols have been recently designed specifically for e-health systems. We survey several of these protocols and report on flaws and shortcomings we discovered. Moreover, we propose an authentication protocol that enables a medical professional and the network of sensors used by a patient to authenticate each other and share a cryptographic key to be used for security in a communication session. The protocol also enables the dynamic assignment of patients to doctors in order to control access to patients' data. We perform a security analysis of the protocol both formally, using the ProVerif protocol analysis tool, and informally, demonstrating its security features. We show that our protocol achieves mutual authentication, secret key establishment, forward secrecy, and anonymity. In terms of performance, the protocol is computationally lightweight, as it relies on symmetric key cryptography. This is demonstrated by comparing the computational cost of our protocol (in terms of execution time) with that of other similar protocols.

Keywords: authentication; body sensor networks; communication system security; cryptographic protocols; Internet of Things

check for updates

Citation: El-Meniawy, N.; Rizk, M.R.M.; Ahmed, M.A.; Saleh, M. An Authentication Protocol for the Medical Internet of Things. *Symmetry* **2022**, *14*, 1483. https://doi.org/10.3390/sym14071483

Academic Editors: Chin-Ling Chen and José Carlos R. Alcantud

Received: 19 June 2022
Accepted: 17 July 2022
Published: 20 July 2022

1. Introduction

An aging population and an increasing number of persons living with multiple chronic conditions are factors currently exerting pressure on healthcare systems worldwide [1]. There is a growing need for healthcare staff and for efficient and effective systems to support them. This need is even more critical during emergency conditions such as those present during the COVID-19 pandemic. The digital transformation in the healthcare sector offers ways to improve the quality of medical services and to support medical staff. It provides a spectrum of solutions commonly referred to as "digital health" or "eHealth" [2]. For instance, assisted living technologies [1] make use of biomedical sensors [3], internet connectivity, and software applications to monitor health conditions of patients while they are at home. Sensors collect real-time medical data that are sent to medical centers to be stored in Electronic Health Records (EHRs) for analysis by doctors, possibly with the help of AI-based decision support systems. Thus, treatment can be decided, emergencies can be avoided and the need for hospitalization reduced, alleviating pressure on medical facilities and staff. The objective is to enable patients to live independently while receiving quality healthcare services at home.

Systems that remotely collect health-related data [4] provide a lot of possibilities for progress in medical research, diagnosis, and treatment. Of course, biomedical sensors are an enabling technology for such systems, e.g., assisted living, telehealth, and m-health [1]. A sensor comprises a sensing unit, a processing unit, and a transceiver, and is usually battery-powered. Sensors are placed in or on a patient's body to form a Wireless Body Area Network (WBAN) [5,6], also called Body Sensor Network (BSN). Each sensor in the network monitors the value of some physiological variable such as blood pressure, heart electrical activity, or body temperature. Using the WBAN's protocol stack, sensors send their readings wirelessly to a special node in the WBAN called the sink. Data are then read from the sink either locally or remotely. In an Internet of Things (IoT) framework, the sink has Internet connectivity and acts as an IoT gateway [7]. Data can thus be relayed, through the cloud, to remote monitoring stations to be examined by medical staff. This Medical Internet of Things (MIoT) is subject to a lot of research in order to provide maximum benefit for both doctors and patients.

Remote monitoring systems of the MIoT open up new possibilities for offering medical services and improving the quality of life for patients. However, when transmitting and storing medical data using a publicly accessible IT infrastructure, security and privacy are major concerns [8,9]. Patients' data need to be protected in transit and while being stored. This protection is defined in terms of security objectives that should be achieved such as data secrecy, data integrity, user authentication, etc. When designing a system, we first need to decide on the security objectives (properties) that we aim to achieve. Then, security mechanisms are implemented in the form of cryptographic operations and protocols in order to achieve these objectives.

One of the main security objectives in the MIoT is entity authentication is the verification of the identity of communicating parties to be able to establish secret keys for communication, enforce access control policies, and manage authorizations. While there has been a lot of research in the design and analysis of authentication protocols for computers [10] and the Internet of Things (IoT) [11], the nature of MIoT networks dictates the design of new protocols. The communication architecture of these networks involves several types of communicating nodes. At the patient's end, biomedical sensors collect data and send them to a gateway node that forwards them to a medical center where they may be monitored online by medical staff or stored for later examination. Sensor nodes are limited in computational and storage capacities. They are also usually battery-powered and therefore face the risk of premature depletion of energy if the algorithms they run consume too much power for the battery's capacity. On the other hand, medical professionals need to examine patients' data locally at the medical center, or even remotely using a variety of devices. All of these factors have lead to an active research track in developing authentication protocols for the MIoT.

Our contribution is to develop a mutual authentication and key establishment protocol for MIoT networks that is lightweight in terms of computation and storage needs. The protocol is executed between a medical professional in possession of a smart card device, a trusted server, and the IoT gateway of a patient's WBAN. We demonstrate the security of our protocol by the use of the ProVerif automated security analysis tool and by informal arguments. We also demonstrate its performance in terms of computational cost in comparison to other similar protocols. The novelty of our approach is that it provides a combination of features that are essential for remote health monitoring applications. First, it allows doctors to monitor a set of sensors, instead of a single one as is the case in most previous works. This is necessary in medical settings where doctors need to know the values of a set of physiological parameters not just a single one. Second, our protocol enables the establishment of an access control policy where doctors may be granted or denied access to one or more patients' data. This policy is dynamic and under the control of medical centers. Finally, our protocol supports both real-time monitoring of patient's data by doctors or the storage of these data for later examination. As another contribution, we point out weaknesses in previously published protocols that, to the best of our knowledge, have not been previously reported. These weaknesses may lead to attacks that compromise security. We demonstrate several of these attacks.

This paper is organized in five sections starting with the introduction. In Section 2, we survey the state-of-the-art in authentication protocols for the MIoT and present some weaknesses that we discovered in these protocols. This is followed by the presentation of our own protocol in Section 3 and its performance and security analysis in Section 4. Finally, the paper is concluded in Section 5.

2. Previous Work

Communication in MIoT can be classified into intra-BAN, inter-BAN, and beyond BAN [12]. Intra-BAN communication takes place between sensor nodes in the WBAN, including the sink. Inter-BAN communication is between sinks in different WBANs, possibly through relay nodes or access points. Finally, beyond-BAN communication occurs with remote parties in the cloud.

Security issues in intra-BAN communications overlap those in sensor networks [13]. There are, however, some characteristics that are particular to intra-BAN networks, based on the fact that they are composed of nodes placed in or on a human body. For instance, in these networks, physiological signals can be used for authentication and key agreement [14]. Similarly, security issues in inter-BAN and beyond-BAN communications overlap those in IoT networks [9]. However, inter-BAN and beyond-BAN networks have special features and requirements since they are used in medical applications. These applications involve three parties: patient, doctor, and medical center (server). They deal with medical data which are of sensitive nature requiring privacy policies. Furthermore, patients need to be dynamically assigned to doctors and access to patients' data must be controlled. Hence, a security mechanism for medical applications must take all these factors into account. Our authentication protocol focuses on beyond-BAN communication. We therefore survey protocols with the same scope and exclude protocols dealing solely with inter-BAN and intra-BAN communication such as the one developed by Das et al. [15].

2.1. Protocols

Authentication protocols for beyond-BAN communication differ in the following aspects:

- Security objectives besides authentication, e.g., key establishment, forward secrecy [16], etc.
- Communicating parties, i.e., the roles involved in a protocol session.
- Support for real-time communication.
- Design logic, cryptographic operations (encryption, hashing, etc.), and execution steps.

In the work of Yeh [17], an authentication and key establishment protocol is designed with three communicating parties: sensor node, gateway (called local processing unit) and server. There is no mention of medical professionals connecting to the server. Therefore, it is implicitly assumed that doctors will read patients' data in the offline mode, i.e., there is no possibility of monitoring a patient in real time. The protocol uses Elliptic Curve Cryptography (ECC) [18] whose computational cost is around an order of magnitude higher than symmetric encryption [19,20]. We will therefore limit our attention to protocols using symmetric cryptography.

Gope and Hwang [20] designed an authentication and key establishment protocol between a user U, a gateway G and a sensor node SN, where user anonymity is targeted. In their terminology, a gateway is actually a "server-class" device that connects users to multiple clusters of sensors. Traffic between sensors in a cluster and an external party passes through a sink node, called the cluster head. The protocol uses only two mathematical operations: secure hashing and XOR. However, the protocol does not account for the case where readings from multiple sensors are aggregated, e.g., by the cluster head. Furthermore, we discovered a serious flaw that lets an intruder, at one session, know the key established in the previous session, as will be detailed in Section 2.2.2.

The authentication protocol by Li et al. [21] is designed for three communication parties: the medical professional (user U), the gateway node G of the patient's WBAN and a single sensor in the WBAN. The protocol targets user and sensor anonymity with respect to the intruder. It uses hashing and symmetric encryption/decryption, where a single key is shared between multiple parties. This use of keys means that the compromise of a single key compromises communication with *all* users or sensor nodes. Furthermore, users are required to register themselves at the patient's WBAN gateway node, i.e., a doctor responsible for N patients will have to register N times which is impractical. We also note that the gateway G stores no information about users, it therefore has no means of differentiating between users for access control purposes.

A study by Sharma and Kalra [22] proposed a mutual authentication and key establishment protocol between a user U who is a medical professional, a patient's WBAN gateway node G, and a single sensor in the WBAN. The protocol aims to achieve forward secrecy and user anonymity with respect to the intruder. The only mathematical operations used are XOR and hashing. The sensor's ID is sent in clear text in the first message from the user U to the gateway G, and therefore sensor anonymity is not achieved. Furthermore, an intruder monitoring a doctor's messages will relate the doctor's identity to a sensor's identity which can be linked to a patient's identity by monitoring the gateway. Moreover, the gateway does not store information related to specific users and therefore is unable to differentiate between users leading to infeasibility of access control. Finally, the gateway stores only the value of a long-term secret key K which is used to create secrets shared with users and sensors. Therefore, if the value of K is compromised, the security of all communications is also compromised.

The paper by Xu et al. [23] presents a protocol between a server, an access point (acting as WBAN gateway), and a single sensor node. There is no mention of the authentication process between a medical professional and the server. The protocol uses only hashing and XOR operations, its goal is to provide mutual authentication and session key establishment between the server and the sensor node while preserving forward secrecy. A single long-term sever key k_{ser} is used for communications with all sensor nodes. Therefore, if k_{ser} is compromised, the whole network is compromised, and no procedure is given to renew k_{ser}. A later work by Park et al. [24] found the protocol vulnerable to node capture attacks where an intruder may capture a sensor node and read plain text values stored in it. The proposed solution is to encrypt values stored in sensors using keys derived from a user's ID and password. However, sensor nodes are not usually equipped with user interfaces for login purposes, and a remote login requires the design of yet another protocol. Moreover, the proposed protocol fails to achieve forward secrecy. This is because it computes session keys K_s as a hash value, and all inputs to the hash function can be easily computed if the server key k_{ser} is known.

Shin and Kwon [25] developed an authentication protocol that is executed between user U, a gateway G and a sensor node SN. A gateway is actually a "server-class" device, and sensors are grouped into clusters, where a cluster may be the WBAN of a patient. The protocol uses only hashing and XOR operations, and three-factor authentication of users. Zhu et al. [26] analyzed the protocol and discovered that it fails to provide forward secrecy and may also be vulnerable to offline password-guessing and desynchronization attacks. Desynchronization occurs when a shared variable, between two or more parties, has a different value at each party. This may be due to incomplete sessions, which we discuss in Section 2.2.5. Zhu et al. also proposed a modified version of the protocol, which we discuss in Sections 2.2.1, 2.2.2 and 2.2.5.

A lightweight mutual authentication and key agreement protocol is presented by Soni and Singh [27]. It is executed between patients and servers in medical centers, so that a patient's data can be securely stored on a server. The protocol uses hashing and XOR. It is, however, vulnerable to offline password-guessing attacks and does not mention authentication of doctors to servers or how to implement access control policies.

The work by Shreya et al. [28] presents a mutual authentication and key agreement protocol between a user (doctor), medical cloud server, and IoT gateway. The gateway collects data from medical sensors to be read by medical staff. The protocol uses symmetric encryption/decryption, hashing, and XOR operations. However, it is vulnerable to offline password guessing. Furthermore, if a single secret session key is known by the intruder, he will be able to obtain all previous and future session keys, as we report in Section 2.2.2.

2.2. Discovered Weaknesses

In the following, we present weaknesses discovered in the surveyed protocols, and we group these weaknesses according to their root cause. We use $h(\cdot)$, $\oplus$, and $\|$, to denote cryptographic hash functions, the XOR operation, and the concatenation operation, respectively. Furthermore, discovered attacks on surveyed protocols which are presented in the current section, in addition to the analysis presented in Section 2.1 are summarized in Table 1.

Table 1. Summary of the analysis of surveyed authentication protocols.

Protocol	**Comments**
Yeh [17]	⇒ If an intruder knew the value of one session nonce, he will be able to compute values of nonces in other sessions.
Gope and Hwang [20]	⇒ Communication with a single sensor per session (no support for data aggregation). ⇒ Attack: value of previous session's key can be known. ⇒ Off-line password attack is possible. ⇒ Desynchronization is possible.
Li et al. [21]	⇒ Single key shared with all users. ⇒ No differentiation between users leads to infeasibility of access control. ⇒ Doctors have to register themselves with each patient. ⇒ Off-line password attack is possible.
Sharma and Kalra [22]	⇒ Sensor's anonymity is not achieved. ⇒ No differentiation between users leads to infeasibility of access control. ⇒ Single key shared with all users.
Xu et al. [23]	⇒ Communication with a single sensor per session (no support for data aggregation). ⇒ Single key shared with all users. ⇒ If an intruder knew the value of one session nonce, he will be able to compute values of nonces in other sessions.
Park et al. [24]	⇒ Sensor nodes are required to provide a local login interface to users, which is impractical. ⇒ Protocol fails to achieve forward secrecy.
Shin and Kwon [25]	⇒ Protocol fails to provide forward secrecy. ⇒ Protocol is vulnerable to desynchronization attacks.
Soni and Singh [27]	⇒ Offline password-guessing attack is possible. ⇒ No mention of authentication for doctors.
Zhu et al. [26]	⇒ Attack against anonymity is possible. ⇒ Denial of service attack against sensors is possible. ⇒ Off-line password attack is possible. ⇒ Desynchronization is possible.
Shreya et al. [28]	⇒ Offline password-guessing attack is possible. ⇒ If a secret session key is known, all previous and future session keys can be known.

2.2.1. The Use of XOR for Secrecy

In various reviewed protocols, when a secret value x needs to be sent, it is XORed with a key K or a hash value $h(m)$, i.e., it is sent as $x \oplus K$ or $x \oplus h(m)$. This is secure as long as multiple secret values $x_1, x_2, \ldots x_i, \ldots$ are XORed with a random or pseudo-random stream of keys such as in the Vernam or RC4 ciphers [18]. If, on the other hand, multiple values are XORed with the same key (or hash value), then secrecy is compromised. Consider, for instance, the case where we need to send two secret values x_1 and x_2, and they are sent as $c_1 = x_1 \oplus K$ and $c_2 = x_2 \oplus K$, respectively. Now an intruder sees c_1 and c_2 and performs the operation $c_1 \oplus c_2 = x_1 \oplus x_2$. If the intruder knows x_1, he can easily compute x_2, and vice versa, without having to know the key. In fact, in the case of more than two plain text messages, if the intruder knows only one of them, he can easily compute all the other ones. A protocol with this weakness is the one by Yeh [17], where the message $M_1 = h(K_i \parallel bs_i) \oplus N_i$ is sent in each session by sensor i. The value N_i, which is a nonce freshly generated each session, is secret. It is, therefore, XORed with the hash value, which is constant an all sessions. Therefore, if the intruder is able to know the value of one nonce, he will be able to compute the values of all other nonces. Another example is the protocol by Xu et al. [23]. At the start of the authentication session between a sensor node and the server, a secret nonce r is sent encrypted by the server key K_{ser}. All nonces from all sensors in all sessions are sent encrypted by the same key K_{ser}. Therefore, if the intruder was able to know the value of a single nonce, she can easily compute all other values.

We discovered that the improper use of XOR for encryption leads to an attack against anonymity in the work of Zhu et al. [26]. The protocol attempts to hide the identity of users using pseudonyms. During a session, the pseudonym is sent in clear text inside messages and is used to identify the user in the current session. At the end of each session, user U_i is assigned a new pseudonym PID_i^1 to be used in the next session. Therefore, an intruder reading two messages sent in different sessions will not be able to decide if these messages have the same sender or not, since the pseudonym is changed in each session. For this scheme to work, PID_i^1 should be sent secretly. Otherwise, the intruder will be able to relate the current pseudonym with the next one, and having a dynamic pseudonym will be useless. In fact, PID_i^1 is sent secretly by XORing it with a secret value privately known by the user. In the following, we demonstrate a multi-session attack, which enables an attacker to relate a value of the pseudonym to the next one. We use subscript i for user U_i and superscripts 1 and 2 to denote two consecutive sessions, e.g., PID_i^1 and PID_i^2 are the pseudonyms secretly sent to user U_i in some session and the next one, respectively. Values that do not change from one session to another are written without superscripts. We focus on the last message to be received by U_i in a session. This message contains two fields: p_i^1 and T^1, where T^1 is a timestamp that is sent in clear text. Knowing that $p_i^1 = h(TID_i \parallel HID_i) \oplus PID_i^1 \oplus HID_i \oplus T^1$, the intruder can execute the computation: $p_i^1 \oplus T^1 = h(TID_i \parallel HID_i) \oplus PID_i^1 \oplus HID_i$. Then, after the next session, the intruder can compute $(p_i^1 \oplus T^1) \oplus (p_i^2 \oplus T^2) = PID_i^1 \oplus PID_i^2$. Therefore, knowing PID_i^1, the intruder will know PID_i^2. In fact, PID_i^1 is sent in clear text during the next session and this way the intruder will be able to know that the sequence of pseudonyms $PID_i^1, PID_i^2, \ldots$ belongs to the same user. This attack is possible even though the intruder does not know the value of HID_i which is secretly assigned to the user during registration to the system.

2.2.2. Missing Secrecy of some Messages

In some protocols, values that should be secret are sent as clear text. For instance, in the protocol by Gope and Hwang [20], a user U is assigned a sequence number TS_{UG} which is updated at the end of the authentication session by a new value TS_{UG}^{new}. Therefore, at the end of session i, U receives TS_{UG}^{new} inside the message $TS = h(K_{UG} \parallel ID_U \parallel N_U) \oplus TS_{UG}^{\text{new}}$, where K_{UG} is a secret key shared between U and G, ID_U is the user's hidden identity and N_U is a secret nonce. This way TS_{UG}^{new} is kept secret from intruders. Furthermore, G generates a session key SK and sends it to U secured inside the message $SK'' = h(K_{UG} \parallel ID_U \parallel N_U) \oplus SK$. If an intruder knew TS_{UG}^{new}, he can easily compute $TS \oplus TS_{UG}^{\text{new}} = h(K_{UG} \parallel ID_U \parallel N_U)$ now the intruder can compute the session key $SK = SK'' \oplus h(K_{UG} \parallel ID_U \parallel N_U)$. In fact, this attack will take place since at the start of session $i + 1$, U will send TS_{UG} in clear text to G (TS_{UG}^{new} of session i is TS_{UG} of session $i + 1$). This way at the start of session $i + 1$ the intruder will able to know the session key of the previous session i.

Furthermore, in the protocol by Zhu et al. [26], a gateway G sends requests to some sensor S_j. In the request, a value NG_j is sent in clear text to the sensor, and if the value is above some threshold, the sensor will not respond to the request. The idea is to save the sensor's resources according to some protocol logic. The problem is that NG_j is not encrypted nor protected against modification. Hence, an intruder can easily read and modify (e.g., increment) its value without being detected, thereby executing a successful denial of service attack.

In the work by Shreya et al. [28], the session key is computed as $SK = h(RN_1) \oplus h(RN_2) \oplus h(CID) \oplus h(SK_{EN})$. The first three hash values can be known by the intruder using publicly sent messages. In Step 7 of the protocol's description, the message $Mg_{10} = CID \parallel Mg_3$ is sent in clear text over the channel from gateway node to user node, where $Mg_3 = h(CID) \parallel h(RN_1) \parallel h(RN_2)$. Obviously, this enables the intruder to know the current values of $h(CID)$, $h(RN_1)$, and $h(RN_2)$. The value $SK_{EN} = ID_N \parallel S_k$ does not change from one session to another. Therefore, if the intruder was able to know the value of a single session key SK, he will be able to compute $h(SK_{EN}) = SK \oplus h(RN_1) \oplus h(RN_2) \oplus h(CID)$. The intruder will therefore be able to compute all previous and future session keys provided that he knows the message Mg_{10} of these sessions.

2.2.3. Ambiguous Definition of Anonymity

Anonymity is defined as the inability of an observer to link an action to the identity of an agent (a communicating party) [29]. An action in our setting is mostly the sending or receiving of a message and the observer is either an honest communicating party or the intruder. Therefore, for instance, sender anonymity is an inability to link the sent message to the identity of an agent. Here, the observer may be the intruder but it may also be the agent receiving the message. Another example is relationship anonymity [30] which involves both the sending and receiving of a message. Therefore, we may know which agent sent a message and which one received it but we are unable to know who is communicating with who. Various published authentication protocols claim to provide anonymity [22], yet they fail to clarify what type of anonymity they mean and with respect to which observer.

2.2.4. Smart Cards Vulnerable to Offline Password Guessing

Humans are authenticated based on what they know (e.g., a password), what they have (e.g., a smart card), or what they are (e.g., a biometric) [18]. We can combine two or three of the aforementioned authentication factors to have two-factor or three-factor authentication, respectively. The purpose is to avoid known weaknesses when relying on passwords only, especially given that humans tend to choose passwords that they can easily remember, which are generally low-entropy and vulnerable to dictionary attacks. Some of the protocols we have reviewed use smart cards for authenticating users to servers in a way that is vulnerable to offline password-guessing attacks. In these attacks, the intruder guesses a password and is able to verify the correctness of their guess offline, i.e., without communicating with the server. They are thus more serious attacks than online password guessing in which the server is alerted with each unsuccessful login attempt and may disable the user account after a predetermined number of failed logins. In the following discussion, we assume that the intruder has access to the user's smart card and biometric information (if used by the protocol) and is trying to guess the user's passwords.

In the protocol by Gope and Hwang [20], the user inputs an ID and password to the smart card which verifies the password and sends a login request to the server only when the password is correct. Hence, an intruder can execute multiple offline password-guessing attempts and immediately know if the guess is correct or not by monitoring whether the smart card sends a message or not. The same argument applies to both the protocol by Zhu et al. [26] and the one by Li et al. [21], where the smart card does not communicate with the server unless the login attempt is a valid one. The main cause in all the previous cases is that the verification of the user's identity is made by the smart card without communicating with the server. An intruder in possession of the smart card is able to access its storage in order to read values, execute code, and read the code's output even under the assumption of tamper-resistant cards [31]. It is therefore necessary not to store clear-text data or code on the smart card that will enable the intruder to decide the correctness of a guessed password.

2.2.5. Renewal of Some Values Each Session

Some cryptographic keys are long-term, i.e., they are used for a period of time that spans multiple sessions, others are ephemeral, i.e., they are established in a session to be used in this session only. Prior to running any authentication sessions, long-term keys need to be preloaded into communicating devices using a secure channel. Thereafter, these keys are used in cryptographic operations to securely establish session keys. Long-term keys are not renewed at each session but every n sessions. Usually, they also need to be renewed using special communication steps. This is particularly important in unreliable networks where message loss may occur. Assume we need to renew a long-term key K_{AB} between nodes A and B and the new key is K'_{AB}. One of the nodes, say A, updates K_{AB} and sends a message m to B, which should update K_{AB} upon receiving m. If m is lost, A and B will be desynchronized since A will store K'_{AB}, while B will still have K_{AB}. Therefore, the renewal process must ensure that, by its end, both agents will agree on the value of the key, either new or old. In the case of session keys, this problem is not a major issue because when the session key could not be established for any reason, A and B can just start a new session as long as they agree on the values of long-term keys.

The protocol by Gope and Hwang [20] renews long-term keys in the communication steps of each session. An example is the key K_{gs} shared between the network's gateway G and a sensor S. This key is used to encrypt the session key SK when it is sent from G to S. The sensor S renews K_{gs}, then sends its last message m_l in the current session to G. After receiving this message, G renews the value of K_{gs} and now the new value will be used in the next session. Therefore, if m_l is lost or blocked by the intruder, G and S will end up having different values of K_{gs}. This situation will make future sessions impossible.

Moreover, Zhu et al. [26] propose a protocol that avoids desynchronization of the values of a variable PID shared between a user U and a gateway G. Their solution to the desynchronization problem is to save a copy of the old value of PID at the gateway. This way, at the next session, if U uses the old value, G will still be able to recognize it. However, in their protocol, the communication between G and a sensor node S is not protected against desynchronization. The value of a secret X_{sj} shared between G and S is updated at G overriding the old value, then G sends a message m' to S. Upon receiving m', S updates its own copy of X_{js}. The message m' contains the value of a counter NG_j that is stored at G and incremented by an amount (value n_j) each time G sends message m'. Another counter N_j is kept at the sensor S, and, upon receiving NG_j, S computes the value $N = (NG_j - N_j)/n_j$. Then, S updates X_{js} N times and sets $N_j = NG_j$. This way, even if the intruder blocked m' multiple times, S will be able to re-synchronize its values with G. As we mentioned in Section 2.2.2, the problem is that NG_j is not encrypted nor protected against modification. Therefore, an intruder can easily change its value and desynchronize G and S.

3. Our Protocol Design

In this section, we present our protocol for mutual authentication between a doctor (or healthcare professional) and the IoT gateway that relays a patient's data to the cloud. Therefore, we focus on beyond-BAN communications. Besides authentication, We aim to achieve the following objectives:

1. Fresh session key establishment.
2. Perfect forward secrecy.
3. Relationship anonymity [30] with respect to the intruder.

We assume the presence of an active intruder I able to perform the following actions [18]:

- Eavesdrop on, and store, all messages communicated over publicly accessible channels.
- Perform computations on messages.
- Register with the network as a legitimate user.

- Block any message from reaching its intended destination.
- Exchange messages with any other node on the network.
- Impersonate other users.

Such an intruder is capable of executing many types of attacks [10], such as reflection attacks, replay attacks, etc. This is a common model where it is assumed that the intruder has total control over the network.

3.1. Network Architecture

The network environment where the protocol runs is depicted in Figure 1. This setting is used, for instance, in applications where doctors need real-time access to data collected from patients. Data are examined by doctors for purposes of diagnosis or follow-ups. We identify the following principals that take part in protocol execution:

- The doctor (or healthcare professional) identified by U, for user.
- The doctor's smart card device, given identification D.
- The server identified by S.
- The IoT gateway (identity G). It relays data between server S and the sensors of the patient's WBAN.
- The patient, identified by A.

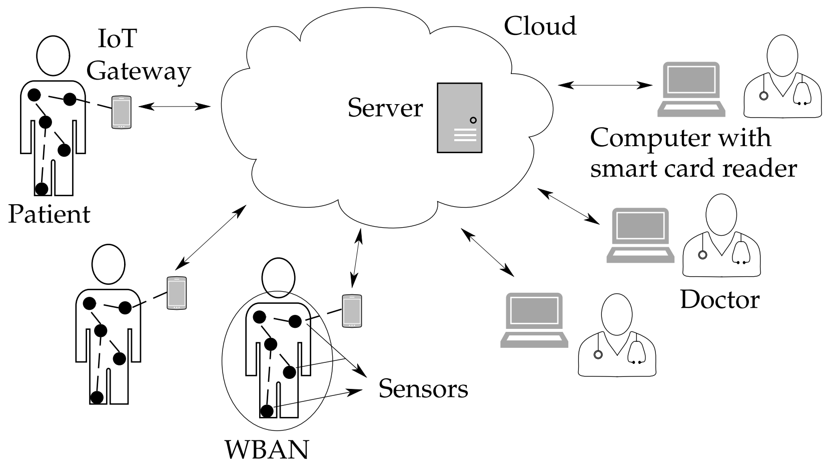

Figure 1. A network topology for remote health monitoring.

A smart card device (identified by D) is used at the doctor's side to enable doctors to log in from any machine as long as it is equipped with a smart card reader. Furthermore, using a smart card, we are able to implement three-factor authentication: possession of the smart card, biometric recognition, and knowledge of a secret password. Moreover, by executing the protocol with the WBAN's gateway, instead of a single sensor, we enable the monitoring of several sensors in a single session since the gateway may aggregate traffic from multiple sensors. This is beneficial in medical settings where a doctor needs to check values of multiple physiological parameters. In addition to the previous advantages, our protocol gives the server S the ability to dynamically assign patients to doctors. Therefore, S may grant a doctor the right to read a patient's data collected by the gateway from sensor nodes, or it may revoke this right, as will be explained in Sections 3.2 and 3.3. Finally, the protocol uses symmetric cryptographic algorithms [18] as they are more efficient than asymmetric ones in terms of computational cost, storage, and energy [19]. Algorithms with a small footprint are preferable in our situation since gateways and smart card devices are usually resource-limited devices. The use of symmetric cryptography implies that, initially, values of secret keys need to be shared offline or via a secure channel. In the following sections, we present the initialization process, followed by the description of the authentication steps. Then, we demonstrate how to renew long-term secret keys and passwords. We will make use of the symbols listed in Table 2.

Table 2. Table of used symbols.

Symbol	Meaning
U	User (doctor)
D	Identity of U's smart card device
G	IoT gateway
S	Server's identity
A	Patient's identity
I_U	ID of user U
P_U	Password of user U
N_S, N_D, N_G	Nonces generated by S, D, and G, respectively.
$h(\cdot)$	Hash function
$H(\cdot)$	Bio-hash function
$e(\cdot,\cdot), d(\cdot,\cdot)$	Encryption and decryption algorithms using key and message.
$e_K(\cdot), d_K(\cdot)$	Encryption and decryption functions using key K implemented as $e(K,\cdot)$ and $d(K,\cdot)$, respectively.
$\{m\}_K$	Sometimes this notation is used for $e(K,m)$
$m_1 \parallel m_2$	Concatenation of messages m_1 and m_2
$\parallel_{i=1}^{L} m_i$	Shorthand for $m_1 \parallel m_2 \parallel \ldots \parallel m_L$
$\langle m_1, m_2, \ldots, m_L \rangle$	Tuple of messages, i.e., $(m_1 \parallel \ldots \parallel m_L)$
$\parallel_{i=1}^{L} \langle m_i, n_i \rangle$	Shorthand for $\langle m_1, n_1 \rangle \parallel \ldots \parallel \langle m_L, n_L \rangle$

3.2. Initialization and Registration

During initialization, both the doctor and the patient are required to register with the server S. To initialize a doctor's smart card device D, the doctor registers at the server S as user U by choosing an identification I_U, and a secret password P_U. At the same time, the doctor's biometric template data B_U, such as a fingerprint or facial image, is captured. The server now computes the value of a key $K_U = h(I_U \parallel h(P_U \parallel H(B_U)))$. In this computation, $h(\cdot)$ is a cryptographic hash function, e.g., MD5, and $H(\cdot)$ is a biohash function [32,33]. A biohash is a one-way function used for Biometric Template Protection (BTP) [34], i.e., given the function's output $H(B_U)$, it is unfeasible to obtain the biometric template B_U. Moreover, the biohash function has two properties that make its use advantageous. First, it uses a random value as a parameter. The advantage is that if the intruder knew the value $H(B_U)$, another function $H'(\cdot)$ could be constructed with a different value of the random parameter, and we will obtain a different output for the same biometric template, i.e., $H(B_U) \neq H'(B_U)$. Second, the output $H(B_U)$ is invariant with respect to rotation, translation or scaling of the captured biometric image [33].

The server, then, generates a secret key K_{DS} and the following code and data are loaded on D:

- Code: A cryptographic hash algorithm $h(\cdot)$, a bio-hash algorithm $H(\cdot)$, an encryption algorithm $e(\cdot,\cdot)$, and a decryption algorithm $d(\cdot,\cdot)$.
- Data: A device identifier D, and the encrypted value $\{K_{DS}\}_{K_U}$.

As a user record, server securely stores $\langle D, I_U, K_{DS} \rangle$. At the patient's side, patient A registers at the server S, when being provided with the necessary sensors needed for remote monitoring. Sensors are attached to A's body, and they form a WBAN where one sensor node will act as IoT gateway. The gateway is loaded with an identity G and a secret key K_{GS} that is shared with S. The patient's record at the server securely stores the tuple $\langle G, I_A, K_{GS} \rangle$, thereby associating the identity G of the gateway to that of the patient (I_A).

The initialization phase is finalized by the server S assigning patients to doctors. When a doctor U finishes registration, S loads the value $\{\ \|_{i=1}^{L} \langle G_i, N_{S_i} \rangle \}_{K_{DS}}$ into the doctor's smart card D. Each tuple $\langle G_i, N_{S_i} \rangle$ represents an assignment to a patient (gateway G_i), so that the doctor is now authorized to monitor the patient's data. Each nonce N_{S_i}, generated by S, is a unique identifier for this assignment. It also serves as indicator that the assignment is still current, and as defense against replay attacks, as will be shown in the description of authentication steps. All tuples are also saved in the doctor's record on the server S, so that the server keeps record of which patient is assigned to which doctor. We note that all data exchanged during registration with the server are assumed to be communicated via secure channels. By the end of the registration phase, we have the following configuration:

- The server's user record stores $\langle D, I_U, K_{DS} \|_{i=1}^{L} \langle G_i, N_{S_i} \rangle \rangle$
- The server's patient record stores $\langle G, I_A, K_{GS} \rangle$
- The doctor's smart card device D stores $\langle h(\cdot), H(\cdot), e(\cdot,\cdot), d(\cdot,\cdot), D, \{K_{DS}\}_{K_U}, \{\ \|_{i=1}^{L} \langle G_i, N_{S_i} \rangle \}_{K_{DS}} \rangle$
- The patient's IoT gateway G stores $\langle G, K_{GS} \rangle$

3.3. Authentication Steps

When the initialization phase has been completed, authentication can be executed. This starts with a doctor U inserting their smart card device D in a reader and entering their ID I'_U and password P'_U. Furthermore, their biometric B'_U is captured. The smart card computes the value of the key $K'_U = h(I'_U \parallel h(P'_U \parallel H(B'_U)))$ and is able to obtain the value of K_{DS} if $K'_U = K_U$. Consequently, the list of gateway ID's $\|_{i=1}^{L} \langle G_i, N_{S_i} \rangle$ is obtained by decrypting the corresponding cipher text, and the doctor is prompted to choose one of their patients to monitor. Then, a sequence of communication steps is executed between the smart card device D, the server S and the IoT gateway G on the patient's body. These steps are shown in Figure 2. Interaction between protocol parties is also illustrated in Figure 3.

In step 1, the message $m_1 = D \parallel \left\{ D \parallel G \parallel N_D \parallel N_S \right\}_{K_{DS}}$ is sent from D to S. When the doctor chooses to monitor G, the value N_S is read from the smart card and sent in m_1. Furthermore, D generates a fresh nonce N_D. Encryption by K_{DS} ensures secrecy and the inclusion of N_S ensures freshness to S. When S receives m_1, it reads the first field (D) to know the message's origin. Now the server retrieves the data stored for D, which includes I_U and K_{DS}. The server uses K_{DS} to decrypt $\left\{ D \parallel G \parallel N_D \parallel N_S \right\}_{K_{DS}}$ and verifies that the first field in the decrypted message is D. Now the server checks the set of stored values $\|_{i=1}^{L} \langle G_i, N_{S_i} \rangle$ to verify that the doctor is authorized to monitor the gateway G and also to check N_S for freshness. If all checks are successful, S executes step 2.

In step 2, S sends the message $m_2 = \{G \parallel D \parallel N_D\}_{K_{GS}}$ to G, where the values G, D, and N_D are those previously received in m_1. When G receives m_2, it decrypts it, and verifies that the first field is its own identity. Then it proceeds to step 3 where it generates a fresh nonce N_G and sends the message $m_3 = G \parallel \{G \parallel D \parallel N_G \parallel N_D\}_{K_{GS}}$ to S. When S receives message m_3, it reads the first field of the message (G) to choose the right key K_{GS}. Then, it uses K_{GS} to decrypt $\{G \parallel D \parallel N_G \parallel N_D\}_{K_{GS}}$ and verifies that the first field in the decrypted message is G and that N_D is the one previously sent in m_2. It then proceeds to step 4.

$$
\begin{array}{lll}
\text{step 1.} & D \rightarrow S: & m_1 = D \parallel \left\{ D \parallel G \parallel N_D \parallel N_S \right\}_{K_{DS}} \\
\text{step 2.} & S \rightarrow G: & m_2 = \left\{ G \parallel D \parallel N_D \right\}_{K_{GS}} \\
\text{step 3.} & G \rightarrow S: & m_3 = G \parallel \left\{ G \parallel D \parallel N_G \parallel N_D \right\}_{K_{GS}} \\
\text{step 4.} & S \rightarrow D: & m_4 = \left\{ D \parallel G \parallel N_D \parallel N_G \parallel K_{DG} \|_{i=1}^{L} \langle G_i, N_{S_i} \rangle \right\}_{K_{DS}} \\
\text{step 5.} & S \rightarrow G: & m_5 = \left\{ G \parallel D \parallel N_G \parallel N_D \parallel K_{DG} \right\}_{K_{GS}}
\end{array}
$$

Figure 2. Authentication protocol.

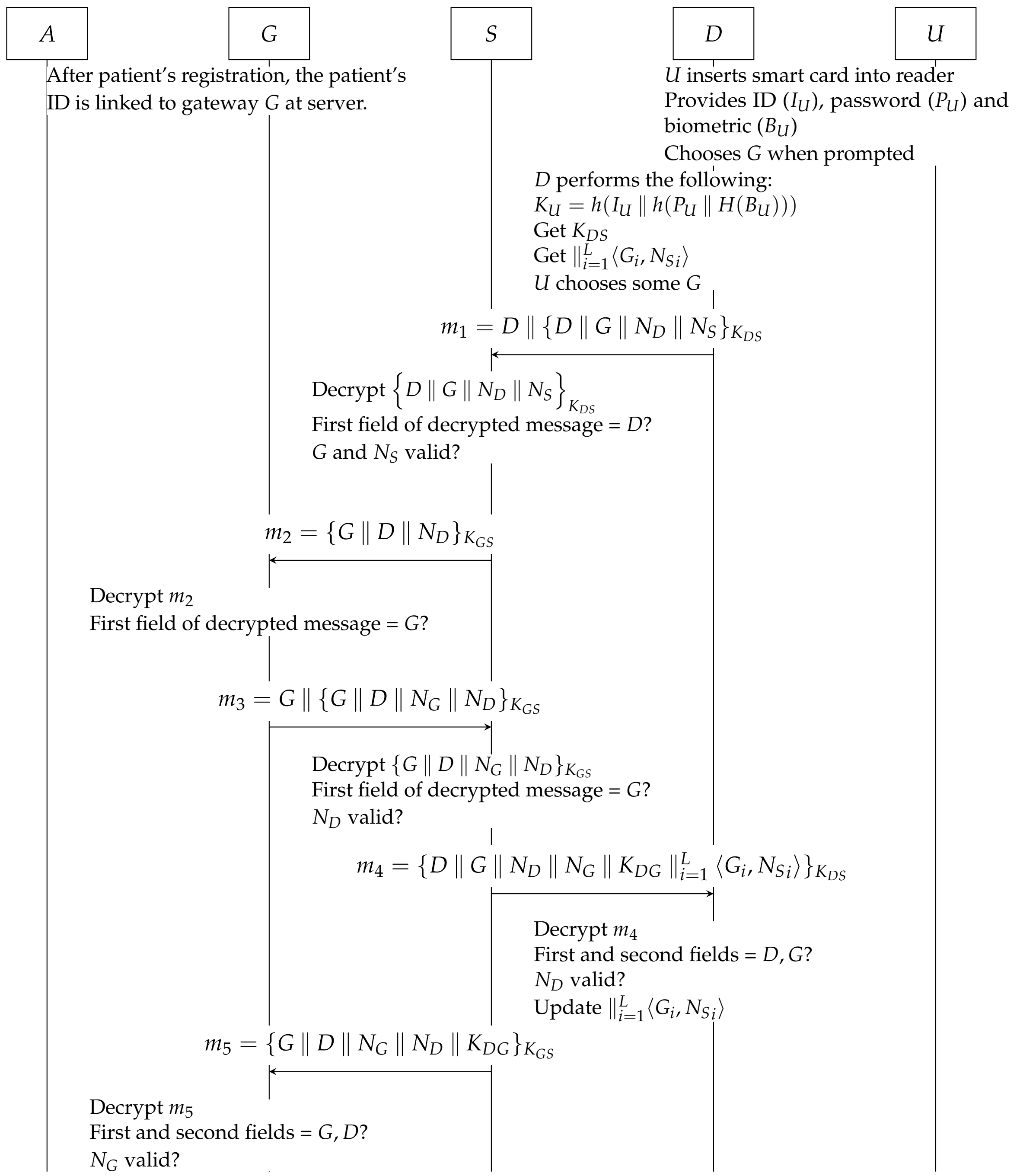
A
G
S
D
U
After patient's registration, the patient's ID is linked to gateway G at server.
U inserts smart card into reader
Provides ID (I_U), password (P_U) and biometric (B_U)
Chooses G when prompted
D performs the following:
$K_U = h(I_U \parallel h(P_U \parallel H(B_U)))$
Get K_{DS}
Get $\parallel_{i=1}^{L}\langle G_i, N_{Si}\rangle$
U chooses some G
$m_1 = D \parallel \{D \parallel G \parallel N_D \parallel N_S\}_{K_{DS}}$
Decrypt $\left\{D \parallel G \parallel N_D \parallel N_S\right\}_{K_{DS}}$
First field of decrypted message = D?
G and N_S valid?
$m_2 = \{G \parallel D \parallel N_D\}_{K_{GS}}$
Decrypt m_2
First field of decrypted message = G?
$m_3 = G \parallel \{G \parallel D \parallel N_G \parallel N_D\}_{K_{GS}}$
Decrypt $\{G \parallel D \parallel N_G \parallel N_D\}_{K_{GS}}$
First field of decrypted message = G?
N_D valid?
$m_4 = \{D \parallel G \parallel N_D \parallel N_G \parallel K_{DG} \parallel_{i=1}^{L} \langle G_i, N_{Si}\rangle\}_{K_{DS}}$
Decrypt m_4
First and second fields = D, G?
N_D valid?
Update $\parallel_{i=1}^{L}\langle G_i, N_{Si}\rangle$
$m_5 = \{G \parallel D \parallel N_G \parallel N_D \parallel K_{DG}\}_{K_{GS}}$
Decrypt m_5
First and second fields = G, D?
N_G valid?

Figure 3. Steps of authentication protocol.

In step 4, S generates a session key K_{DG} and sends the message $m_4 = \{D \parallel G \parallel N_D \parallel N_G \parallel K_{DG} \parallel_{i=1}^{L} \langle G_i, N_{Si}\rangle\}_{K_{DS}}$ to D. The message is decrypted by D, and the values D, G, and N_D are compared with the corresponding ones sent in m_1. The nonce N_D ensures freshness to D. If the values are equal, K_{DG} is accepted as a fresh session key and the values in the tuples $\langle G_i, N_{Si}\rangle$ are encrypted by K_{DS} so that the cipher text $\left\{ \parallel_{i=1}^{L} \langle G_i, N_{Si}\rangle\right\}_{K_{DS}}$ overwrites the stored one. This way the server renews all the doctor's assignments to patients.

In step 5, S sends the message $m_5 = \{G \parallel D \parallel N_G \parallel N_D \parallel K_{DG}\}_{K_{GS}}$. The message is decrypted by G, and the values G, D, N_G, and N_D are compared with the corresponding ones in messages m_2 and m_3. If the values are equal, K_{DG} is accepted as a fresh session key. By the end of this step, K_{DG} is established as a secret session key shared between D and G and session communication can proceed.

It is worth noting that if the data requested by the doctor is already stored at the server, then only steps 1 and 4 are executed to achieve mutual authentication between doctor and server. This case is beneficial when a doctor needs to review a patient's historical data that have been previously collected in past sessions. The differentiation between a request for current real-time data and a request for historical data is left as an application implementation issue since it will not affect the abstract data structure of messages.

3.4. Renewal of Passwords and Keys

A doctor U may wish to change their password P_U. In this case, U inserts the smart card device D into a reader and supplies the old values of the ID, password and biometric: I_U, P_U, and B_U, respectively. The smart card computes the value of the key $K_U = h(I_U \parallel h(P_U \parallel H(B_U)))$ and decrypts $\{K_{DS}\}_{K_U}$ to obtain K_{DS}. Now the user is prompted to enter the new password P'_U and the smart card computes the value of a new key $K'_U = h(I_U \parallel h(P'_U \parallel H(B_U)))$, which is used to encrypt K_{DS}. The encrypted value $\{K_{DS}\}_{K'_U}$ is stored on the smart card instead of the old one.

Furthermore, the long-term keys K_{DS} and K_{GS} are used during some time period: an *epoch*. At the end of an epoch, these two keys are renewed by the server. The new ones (K'_{DS} and K'_{GS}), which are generated by the server, will be used in the newly starting epoch. At the doctor's side (smart card device D), m_4, sent from S to D is modified to include K'_{DS}:

$$m'_4 = \left\{D \parallel G \parallel N_D \parallel N_G \parallel K_{DG} \parallel K'_{DS} \parallel_{i=1}^{L} \langle G_i, N_{Si}\rangle\right\}_{K_{DS}}$$

Upon receiving m'_4, D updates its storage to replace K_{DS} by K'_{DS}. Furthermore, to confirm receiving the new key, D replies with another message:

$$m_6 = \{D \parallel N_D\}_{K'_{DS}}$$

When S receives m_6, it decrypts it using K'_{DS} and checks the value of the nonce N_D for equality with the one received in message m_1. If the two values are equal, S replaces the stored value K_{DS} with K'_{DS}. At the patient's side, m_5, sent from S to the IoT gateway G, is modified to include K'_{GS}: $m'_5 = \left\{G \parallel D \parallel N_G \parallel N_D \parallel K_{DG} \parallel K'_{GS}\right\}_{K_{GS}}$. Upon receiving m'_5, G updates its storage to replace K_{GS} by K'_{GS}. Furthermore, to confirm receiving the new key, G replies with another message:

$$m_7 = \{G \parallel N_G\}_{K'_{GS}}$$

When S receives m_7, it decrypts it using K'_{GS} and checks the value of the nonce N_G for equality with the one received in message m_3. If the two values are equal, S replaces the stored value K_{GS} with K'_{GS}.

3.5. Revocation of Smart Card Device

Sometimes, there is a need to revoke the rights of a smart card device D. For instance, this may be the case if a doctor lost the smart card or if the card was stolen. In this case, at the request of the doctor, a new smart card device D' is issued according to the steps detailed in Section 3.2. The new card D' will be loaded with a new key K'_{DS}, and the old card's identity D and key K_{DS} will be added to a revocation list on the server to indicate their invalidity.

4. Analysis of Protocol

In this section, we analyze the performance of our proposed authentication protocol. We start by enumerating its features in comparison to the surveyed protocols, then we compare its computational cost to other protocols of similar objectives. We also demonstrate its suitability for smart card devices, with respect to storage requirements. Then, we provide an informal argument about its security, followed by a demonstration of the attacks it resists. Finally, we present a formal security analysis using the protocol verification tool ProVerif [35].

4.1. Features and Comparison with Previous Works

Our protocol has the following functional features:

F1: Mutual authentication between patient's IoT gateway and server.
F2: Mutual authentication between doctor and server.
F3: Support for both real-time monitoring of patient's data and offline reading of data from server.
F4: Ability to monitor multiple sensors in a single session.
F5: Support for dynamic assignment of patient's to doctors.
F6: Support for the implementation of an access control policy at the server.

Table 3 compares the surveyed protocols with respect to each one of these features. The symbol '_' in a table cell means that the feature (table column) is not applicable for the protocol (table row). Table 3 deals with functional features only, security features are summarized in Table 1.

Table 3. Protocols and features.

Protocol	**F1**	**F2**	**F3**	**F4**	**F5**	**F6**
Yeh [17]	yes	no	no	no	_	_
Gope and Hwang [20]	yes	yes	no	no	_	_
Li et al. [21]	yes	yes	no	no	no	no
Sharma and Kalra [22]	yes	yes	no	no	no	no
Xu et al. [23]	yes	no	no	no	_	_
Park et al. [24]	yes	no	no	no	_	_
Shin and Kwon [25]	yes	yes	no	no	_	_
Soni and Singh [27]	yes	no	no	yes	_	_
Zhu et al. [26]	yes	yes	no	no	_	_
Shreya et al. [28]	yes	yes	yes	yes	possible	possible
Our protocol	yes	yes	yes	yes	yes	yes

4.2. Performance Analysis

The computational cost of a security protocol is estimated as the time it takes to execute the mathematical operations of a single protocol run [36]. Since we focus on protocols using symmetric cryptography, these operations are symmetric encryption/decryption and secure hashing. The execution time of a hash function is almost equal to that of symmetric encryption or decryption [19], we denote this time by T_f. In Table 4, we compare the computational cost of our protocol with other ones, all during authentication steps. For each protocol, we count the number of operations performed by the protocol's entities (user, server, etc.) and multiply this number by T_f.

In Table 4, the symbol '_' in a cell means that this particular protocol does not have the entity indicated by the column. For instance, the protocol by Xu et al. does not have a user role. We also note that we use our notation to describe various protocol roles. For instance, in the protocol by Gope and Hwang, a gateway in their notation is equivalent to our server role.

Table 4. The computational cost of our protocol compared to surveyed protocols.

Scheme	User	Server	GW	Sensor
Xu et al. [23]	_	7 T_f	0	5 T_f
Gope & Hwang [20]	7 T_f	9 T_f	_	3 T_f
Li et al. [21]	8 T_f	13 T_f	_	7 T_f
Shin and Kwon. [25]	13 T_f	15 T_f	_	6 T_f
Park et al. [24]	_	9 T_f	0	11 T_f
Zhu et al. [26]	11 T_f	10 T_f	_	6 T_f
Our protocol	6 T_f	5 T_f	3 T_f	_

In terms of storage requirements, we focus on the resource-constrained devices involved in the protocol. Namely, these are the IoT gateway G and the smart card device D. We assume the size of a single data block (key, cipher text, nonce, ...) to be 256 bits (32 bytes). This estimate is based on sizes of data blocks in well-known algorithms. For instance, the Advanced Encryption Standard (AES) has a maximum key length of 256 bits and a cipher block size of 128 bits. The patient's IoT gateway G stores the pair $\langle G, K_{GS} \rangle$, whose size is 64 bytes, which can be easily accommodated in a device of this class. On the other hand, data stored on the doctor's smart card device are: $\langle D, \{K_{DS}\}_{K_U}, \{ \|_{i=1}^{L} \langle G_i, N_{Si} \rangle \}_{K_{DS}} \rangle$. Their estimated total size is $64 + 64 \times L$ bytes. Smart card devices are equipped with EEPROM storage whose size typically fall in the range between 8 KBytes and 64 Kbytes [37,38]. Thus, the space is enough to accommodate values of L (number of patients assigned to doctor) in the three digit range. Of course, this is more than needed for all practical purposes.

4.3. Informal Security Analysis

In this section, we argue that our protocol achieves its desired security objectives, where each objective is discussed in a separate section.

4.3.1. Authentication

Message m_1 can be constructed only by an agent that knows K_{DS}. To obtain K_{DS}, the agent must possess the smart card and must know K_U. The key K_U is not stored but must be constructed from the user's credentials (ID, password, and biometric). Therefore, m_1 can only be sent after achieving three-factor authentication. Even if an intruder has possession of the smart card and the user's biometric, he will have to guess the password in order to construct K_U and hence obtain K_{DS}. To test their guess, the intruder needs to communicate with the server since there is no way of knowing the correctness of their guess offline. By limiting the number of failed logins, we can greatly reduce the intruder's probability of guessing the correct password.

Replay attacks are prevented using nonces. For instance, the server S knows that message m_1 is current from the nonce N_S. Furthermore, mutual authentication is achieved by the use of nonces as challenges. Therefore, by message m_1, server S is assured of the identity of D (and consequently U), since only an agent that knows K_{DS} (and consequently K_U) could have sent m_1. Similarly, by message m_4, D is assured of the identity of S, since only an agent that knows K_{DS} could have sent m_4 containing the same nonce N_D that was sent inside m_1 in the current session. A similar argument applies to the mutual authentication between S and G.

4.3.2. Session Key

The session key K_{DG} is guaranteed to be fresh by the use of the two nonces N_D and N_G. It is also secret by its inclusion in encrypted messages. The session key is randomly generated by the server each session and its secrecy is dependent on the secrecy of the long-term keys K_{DS} and K_{GS}.

4.3.3. Forward Secrecy

Assume the long-term keys K_{DS} and K_{GS} were known by the intruder. In this case, the session keys of the current epoch will be revealed to the intruder. However, in any previous epochs the session keys will still keep their secrecy.

4.3.4. Relationship Anonymity

The intruder will not be able to know which doctor U is communicating with which patient A. This is because no direct communication exists between them; each communicates with the server. Furthermore, by monitoring messages only, the intruder is not able to relate messages to users (doctors) or patients since this information is not sent inside messages. In fact, the only identities sent are those of the smart card device D and the gateway G, and it is inside the server that the link is made between D and a doctor's identity, and between G and a patient's identity. However, the intruder may be able to deduce this link (D to doctor or G to patient), for instance, by surveying a doctor while using a smart card D and reading messages sent by D. Nonetheless, this does not constitute a successful attack against relationship anonymity. In particular, no message contains both D and G in clear text.

4.3.5. Access Control

A doctor is allowed access only to designated patients. When a doctor U asks for access to patient A_i's gateway (G_i), the server checks U's record to verify that U has the right to monitor G_i. This implies that the doctor has authorization to monitor the data of all sensors communicating with G_i. If a more detailed approach is preferred, a doctor may be assigned a role according to specialization. Then, this role determines what data (sensors) the doctor is authorized to monitor. For instance, a cardiologist may be allowed to monitor a different set of sensors than an internist. To implement this role-based access control, we need to modify messages m_2 and m_3 to replace the identity D with the pair $\langle D, R\rangle$, where R is the doctor's role. Furthermore, in this case, the gateway G, will need to implement and access control list specifying which role has authorization to monitor which sensors.

4.4. Resistance to Common Attacks

Our protocol resists impersonation attacks by replaying messages due to the use of fresh nonces. It also provides protection in case of smart card loss due to the use of three-factor authentication. Moreover, an intruder cannot mount an offline password-guessing attacks since all of the doctor's credentials are checked online by the server.

4.5. Formal Security Analysis

The ProVerif tool [35] was used to verify our protocol and the following security properties were proved:

- Secrecy of session key K_{DG}.
- Mutual authentication between S and D and between S and G. We use the "injective agreement" definition of authentication [39]. This definition states that A is authenticated to B, by a certain run of an authentication protocol, if for each run of the protocol by A (as initiator) there corresponds a unique run of the protocol by B as responder. Moreover, by the end of these runs, both A and B agree on a set of values. These values, in our case, are nonces, which guarantees the recentness of protocol runs and hence prevents replay attacks.

  The code for our protocol in ProVerif is shown below:

```
free c1, c2: channel.

type key.
type nonce.
type host.
```

```
(* Shared key encryption *)

fun encrypt(bitstring,key): bitstring.
reduc forall x: bitstring, k: key; decrypt(encrypt(x,k),k) = x.

free Kdg: key [private].

event DStart(nonce,nonce).
event DEnd(nonce,nonce).
event GStart(nonce,nonce).
event GEnd(nonce).
event SStartD(nonce,nonce).
event SEndD(nonce,nonce).
event SStartG(nonce).
event SEndG(nonce,nonce).

(* Queries *)

query attacker(Kdg).

query x1,x2:nonce; inj-event(SEndD(x1,x2)) ==> inj-event (DStart(x1,x2)).
query x1,x2:nonce; inj-event(DEnd(x1,x2)) ==> inj-event (SStartD(x1,x2)).
query x1,x2:nonce; inj-event(SEndG(x1,x2)) ==> inj-event (GStart(x1,x2)).
query x1:nonce; inj-event(GEnd(x1)) ==> inj-event (SStartG(x1)).

free D, G, S: host.

table keys(host,key).

(* Role of the doctor (D)*)

let processD =
  get keys(D, Kds) in
  in(c1, (minit: bitstring));
  let (=D, Ns_d: nonce) = decrypt(minit, Kds) in
new Nd: nonce;
  event DStart(Nd,Ns_d);
  out(c1, (D, encrypt((D, G, Nd, Ns_d), Kds)));
  in(c1, (m4: bitstring));
  let (=D, =G, =Nd, Ng_d: nonce, kdg_d:key) = decrypt(m4, Kds) in
  event DEnd(Nd,Ns_d).

(* Role of the gateway *)

let processG =
  get keys(G, Kgs) in
  in(c2, m2: bitstring);
  let (=G, =D, Nd_g: nonce) = decrypt(m2, Kgs) in
  new Ng: nonce;
  event GStart(Ng,Nd_g);
out(c2, (G,encrypt((G, D, Ng, Nd_g), Kgs)));
in(c2, m5: bitstring);
  let (=G, =D, =Ng, =Nd_g, Kdg_g:key) = decrypt(m5, Kgs) in
  event GEnd(Nd_g).

(* Server *)

let processS =
  new Ns: nonce;
  new kds:key;
  insert keys(D, kds);
  new Kgs:key;
  insert keys(G, Kgs);
  out(c1, encrypt((D, Ns), kds));
  in(c1, (idd_s: host, m1: bitstring));
  get keys(=idd_s, Kds_s) in
  let (=idd_s, idg_s: host, Nd_s: nonce, =Ns) = decrypt(m1, Kds_s) in
  event SStartD(Nd_s,Ns);
  get keys(=idg_s, Kgs_s) in
  event SStartG(Nd_s);
out(c2, encrypt((idg_s, idd_s, Nd_s), Kgs_s));
  in(c2, (idg_s: host, m3: bitstring));
  get keys(=idg_s, Kgs_s) in
  let (=idg_s, =idd_s, Ng_s: nonce, =Nd_s) = decrypt(m3, Kgs_s) in
  out(c1, encrypt((idd_s, idg_s, Nd_s, Ng_s, Kdg), Kds_s));
  event SEndG(Ng_s,Nd_s);
  out(c2, encrypt((idg_s, idd_s, Ng_s, Nd_s, Kdg), Kgs_s));
  event SEndD(Nd_s,Ns).

(* Start process *)

process
( (!processD) | (!processG) |(!processS))

=============== Verification Result ======================
Verification summary:

(* Secrecy of session key *)
Query not attacker(Kdg[]) is true.

(* Mutual authentication as injective agreeement *)
Query inj-event(SEndD(x1,x2)) ==> inj-event(DStart(x1,x2)) is true.

Query inj-event(DEnd(x1,x2)) ==> inj-event(SStartD(x1,x2)) is true.

Query inj-event(SEndG(x1,x2)) ==> inj-event(GStart(x1,x2)) is true.

Query inj-event(GEnd(x1)) ==> inj-event(SStartG(x1)) is true.
```

Verification results demonstrate that our protocol satisfies its security objectives.

4.6. Implementation Issues and Limitations

The first step toward the implementation of our protocol is the investigation of its integration with current implementations of eHealth systems. To facilitate this integration, it is beneficial to adopt an Authentication as a Service (AaaS) framework [40]. Then, we need to investigate the storage requirements in terms of code and data for all protocol entities. This represents no problem at the server where there is no lack of resources. However, the gateway and smart card device are usually resource-limited devices and storage requirements need to be carefully computed. This was performed in Section 4.2, where we analyzed the protocol's performance. Finally, our protocol uses lightweight symmetric cryptography that is easily implemented on resource-limited devices since it does not have extensive computational requirements and thus save energy and resources.

With respect to limitations, our protocol uses a single server which may constitute a bottleneck in communication affecting the system's scalability. It also constitutes a single point of failure. Therefore, as a future research direction, the possibility of having multiple servers should be investigated. Of course, this necessitates synchronizing data between servers, with all the details involved in such a process. Moreover, currently, the patient's

IoT gateway is not selective in sending sensors' data to the server. As future work, the gateway should be able to send data only from a subset of sensors in the WBAN. This subset may be determined, for instance, by the specialty of the doctor currently reading the data. This issue falls within the design of an access control policy for doctors.

5. Conclusions and Future Work

We presented an authentication protocol for remote health monitoring within the MIoT. The protocol is lightweight, secure, and avoids the security weaknesses found in similar previously published protocols. It provides a method to grant authorizations to doctors to monitor specific patients. The communication with the sensors at the patient's side is performed through the IoT gateway which enables the doctor to monitor multiple sensors with a single connection. This is possible because the gateway acts as a sink in the WBAN and is able to aggregate traffic from groups of sensors. Both the security and performance analyses confirm the suitability of our protocol for use in MIoT.

As future work, we may consider having multiple servers instead of a single one. This will increase the system's capacity to handle more traffic, and help improve reliability by avoiding interruptions in case a single server fails. Moreover, access control policies implemented on the server may be investigated. In this regard, there are two main issues: policy design and policy enforcement. A policy is designed to achieve some security objective and is based on some model of system objects (components, files, etc.), system subjects (users, processes, etc.), and actions (read, delete, etc.). It is interesting to see how to design such policies for standard Electronic Health Records (EHRs) [41]. Moreover, implementations of these policies need to be tested in order to make sure that all policy rules are enforced. This is of extreme importance in the medical field where data are sensitive and patient privacy is protected by law.

Author Contributions: Conceptualization, M.S.; methodology, M.S. and N.E.-M.; software, N.E.-M. and M.S.; validation, M.S., M.A.A. and M.R.M.R.; formal analysis, N.E.-M. and M.S.; investigation, N.E.-M.; resources, M.R.M.R. and M.A.A.; writing—original draft preparation, N.E.-M.; writing—review and editing, M.S., M.R.M.R., and M.A.A.; supervision, M.R.M.R., M.A.A. and M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

References

1. *Empowering the Health Workforce: Strategies to Make the Most of the Digital Revolution*; Technical Report; Organisation for Economic Co-Operation and Development (OECD): Paris, France, 2020. Available online: https://www.oecd.org/publications/empowering-the-health-workforce-to-make-the-most-of-the-digital-revolution-37ff0eaa-en.htm (accessed on 6 July 2022).
2. Hallberg, D.; Salimi, N. Qualitative and Quantitative Analysis of Definitions of e-Health and m-Health. *Healthc. Inform. Res.* **2020**, *26*, 119–128. [CrossRef] [PubMed]
3. Wan, H.; Zhuang, L.; Pan, Y.; Gao, F.; Tu, J.; Zhang, B.; Wang, P. Biomedical sensors. In *Biomedical Information Technology*; Feng, D., Ed.; Academic Press: Cambridge, MA, USA, 2020; Chapter 2, pp. 51–79.
4. Angelov, G.; Nikolakov, D.; Ruskova, I.; Gieva, E.; Spasova, M. Healthcare Sensing and Monitoring. In *Enhanced Living Environments: Algorithms, Architectures, Platforms, and Systems (LNCS 11369)*; Springer: Berlin/Heidelberg, Germany, 2019.
5. Gandhi, V.; Singh, J. An automated review of body sensor networks research patterns and trends. *J. Ind. Inf. Integr.* **2020**, *18*, 100132. [CrossRef]
6. Liu, Q.; Mkongwa, K.G.; Zhang, C. Performance issues in wireless body area networks for the healthcare application: A survey and future prospects. *SN Appl. Sci.* **2021**, *3*, 155. [CrossRef]
7. Aboubakar, M.; Kellil, M.; Roux, P. A review of IoT network management: Current status and perspectives. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 4163–4176. [CrossRef]
8. Awotunde, J.; Jimoh, R.; Folorunso, S.; Adeniyi, E.; Abiodun, K.; Banjo, O. Privacy and Security Concerns in IoT-Based Healthcare Systems. In *Privacy and Security Concerns in IoT-Based Healthcare Systems*; Siarry, P., Jabbar, M., Aluvalu, R., Abraham, A., Madureira, A., Eds.; Springer: Cham, Switzerland, 2021; pp. 105–134.
9. Keyvan Mousavi, S.; Ghaffari, A.; Besharat, S.; Afshari, H. Security of internet of things based on cryptographic algorithms: A survey. *Wirel. Netw.* **2021**, *27*, 1515–1555. [CrossRef]
10. Boyd, C.; Mathuria, A. *Protocols for Authentication and Key Establishment*; Springer: Berlin/Heidelberg, Germany, 2003; p. 342.

11. El-Hajj, M.; Fadlallah, A.; Chamoun, M.; Serhrouchni, A. A survey of internet of things (IoT) authentication schemes. *Sensors* **2019**, *19*, 1141. [CrossRef]
12. Punj, R.; Kumar, R. Technological aspects of WBANs for health monitoring. *Wirel. Netw.* **2019**, *25*, 1125–1157. [CrossRef]
13. Radhappa, H.; Pan, L.; Zheng, X.J.; Wen, S. Practical overview of security issues in wireless sensor network applications. *Int. J. Comput. Appl.* **2018**, *40*, 202–213. [CrossRef]
14. Reshan, M.A.; Liu, H.; Hu, C.; Yu, J. MBPSKA: Multi-Biometric and Physiological Signal-Based Key Agreement for Body Area Networks. *IEEE Access* **2019**, *7*, 78484–78502. [CrossRef]
15. Das, A.K.; Wazid, M.; Kumar, N.; Khan, M.K.; Choo, K.K.R.; Park, Y.H. Design of Secure and Lightweight Authentication Protocol for Wearable Devices Environment. *IEEE J. Biomed. Health Inform.* **2018**, *22*, 1310–1322. [CrossRef]
16. Boyd, C.; Gellert, K. A Modern View on Forward Security. *Comput. J.* **2021**, *64*, 639–652. [CrossRef]
17. Yeh, K.H. A Secure IoT-Based Healthcare System with Body Sensor Networks. *IEEE Access* **2016**, *4*, 10288–10299. [CrossRef]
18. Van Oorschot, P.C. *Computer Security and the Internet: Tools and Jewels from Malware to Bitcoin*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2021.
19. Rifà-Pous, H.; Herrera-Joancomartí, J. Computational and Energy Costs of Cryptographic Algorithms on Handheld Devices. *Future Internet* **2011**, *3*, 31–48. [CrossRef]
20. Gope, P.; Hwang, T. A Realistic Lightweight Anonymous Authentication Protocol for Securing Real-Time Application Data Access in Wireless Sensor Networks. *IEEE Trans. Ind. Electron.* **2016**, *63*, 7124–7132. [CrossRef]
21. Li, X.; Niu, J.; Kumari, S.; Liao, J.; Liang, W.; Khan, M.K. A new authentication protocol for healthcare applications using wireless medical sensor networks with user anonymity. *Secur. Commun. Netw.* **2016**, *9*, 2643–2655. [CrossRef]
22. Sharma, G.; Kalra, S. A Lightweight User Authentication Scheme for Cloud-IoT Based Healthcare Services. *Iran. J. Sci. Technol. Trans. Electr. Eng.* **2019**, *43*, 619–636. [CrossRef]
23. Xu, Z.; Xu, C.; Liang, W.; Xu, J.; Chen, H. A lightweight mutual authentication and key agreement scheme for medical internet of things. *IEEE Access* **2019**, *7*, 53922–53931. [CrossRef]
24. Park, K.; Noh, S.; Lee, H.; Das, A.K.; Kim, M.; Park, Y.; Wazid, M. LAKS-NVT: Provably Secure and Lightweight Authentication and Key Agreement Scheme without Verification Table in Medical Internet of Things. *IEEE Access* **2020**, *8*, 119387–119404. [CrossRef]
25. Shin, S.; Kwon, T. A Lightweight Three-Factor Authentication and Key Agreement Scheme in Wireless Sensor Networks for Smart Homes. *Sensors* **2019**, *19*, 2012. [CrossRef]
26. Zhu, L.; Xiang, H.; Zhang, K. A Light and Anonymous Three-Factor Authentication Protocol for Wireless Sensor Networks. *Symmetry* **2022**, *14*, 46. [CrossRef]
27. Soni, M.; Singh, D.K. LAKA: Lightweight Authentication and Key Agreement Protocol for Internet of Things Based Wireless Body Area Network. *Wirel. Pers. Commun.* **2021**. [CrossRef]
28. Shreya, S.; Chatterjee, K.; Singh, A. A smart secure healthcare monitoring system with Internet of Medical Things. *Comput. Electr. Eng.* **2022**, *101*, 107969. [CrossRef]
29. Tiplea, F.L.; Vamanu, L.; Vârlan, C. Reasoning about minimal anonymity in security protocols. *Future Gener. Comput. Syst.* **2013**, *29*, 828–842. [CrossRef]
30. Pfitzmann, A.; Kohntopp, M. Anonymity, Unobservability, and Pseudonymity—A Proposal for Terminology. In *Lecture Notes in Computer Science (LNCS 2009)*; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2009.
31. Wang, D.; Wang, P. Offline dictionary attack on password authentication schemes using smart cards. In *Information Security*; Desmedt, Y., Ed.; Springer International Publishing: Berlin/Heidelberg, Germany, 2015; Volume 7807, pp. 221–237.
32. Goh, A.; Ngo, D.C. Computation of Cryptographic Keys from Face Biometrics. In Proceedings of the 7th IFIP-TC6 TC11 International Conference, CMS 2003, Torino, Italy, 2–3 October 2003; Lecture Notes in Computer Science (LNCS 2828); Springer: Berlin/Heidelberg, Germany, 2003; Volume 2828.
33. Jin, A.T.B.; Ling, D.N.C.; Goh, A. Biohashing: Two factor authentication featuring fingerprint data and tokenised random number. *Pattern Recognit.* **2004**, *37*, 2245–2255. [CrossRef]
34. Sandhya, M.; Prasad, M. Biometric template protection: A systematic literature review of approaches and modalities. In *Biometric Security and Privacy. Signal Processing for Security Technologies*; Jiang, R., Al-maadeed, S., Bouridane, A., Crookes, P., Beghdadi, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2017; Chapter 14.
35. ProVerif. Available online: https://bblanche.gitlabpages.inria.fr/proverif/ (accessed on 15 July 2022).
36. Das, A.K.; Zeadally, S.; He, D. Taxonomy and analysis of security protocols for Internet of Things. *Future Gener. Comput. Syst.* **2018**, *89*, 110–125. [CrossRef]
37. Mayes, K. An introduction to smart cards. In *Smart Cards, Tokens, Security and Applications*, 2nd ed.; Mayes, K., Markantonakis, K., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; pp. 1–29.
38. Hajny, J.; Malina, L.; Martinasek, Z.; Tethal, O. Performance evaluation of primitives for privacy-enhancing cryptography on current smart-cards and smart-phones. In *Data Privacy Management and Autonomous Spontaneous Security, Lecture Notes in Computer Science*; Garcia-Alfaro, J., Lioudakis, G., Cuppens-Boulahia, N., Foley, S., Fitzgerald, W., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; Volume LNCS 8247, pp. 17–33.
39. Cremers, C.J.; Mauw, S.; de Vink, E.P. Injective synchronisation: An extension of the authentication hierarchy. *Theor. Comput. Sci.* **2006**, *367*, 139–161. [CrossRef]

40. Shah, Y.; Choyi, V.; Schmidt, A.U.; Subramanian, L. Multi-factor authentication as a service. In Proceedings of the 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud 2015, San Francisco, CA, USA, 30 March–3 April 2015; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2015; pp. 144–150.
41. de Mello, B.H.; Rigo, S.J.; da Costa, C.A.; da Rosa Righi, R.; Donida, B.; Bez, M.R.; Schunke, L.C. Semantic interoperability in health records standards: A systematic literature review. *Health Technol.* **2022**, *12*, 255–272. [CrossRef]