electronics

MDPI

Article

Enhancing Hospital Data Security: A Blockchain-Based Protocol for Secure Information Sharing and Recovery

Jihyeon Ryu and Taeseok Kim *

School of Computer and Information Engineering, Kwangwoon University, Seoul-si 01897, Republic of Korea; jhryu@kw.ac.kr
* Correspondence: tskim@kw.ac.kr

Abstract: Hospitals that store sensitive patient medical records have recently faced issues such as the inability to recover medical data and breaches of patient privacy due to hacker attacks. These attacks on medical data often involve ransomware, which obfuscates the entire hospital's data, making them inaccessible, and can also occur when hospitals share patient information during transfers of care. In this study, we propose a new authentication protocol to prevent and address such issues within hospital systems. The proposed protocol encrypts medical records on a private blockchain, allowing them to be securely shared among institutions, hospitals, and insurance companies, ensuring data recovery even if a ransomware attack paralyzes the server. Additionally, the protocol facilitates the systematic sharing of patient medical records between hospitals or between hospitals and insurance companies by distributing session keys. In this study, we demonstrate that the proposed protocol provides 11 security properties, including forward and backward secrecy, user untraceability, and resistance to replay attacks. We also evaluate the communication and computational costs, proving that the protocol is feasible for practical use.

Keywords: blockchain-based authentication; secure authentication scheme; hospital data security

Academic Editors: Li Pan, Ning Liu and Conghui Zheng

Received: 16 December 2024
Revised: 27 January 2025
Accepted: 30 January 2025
Published: 1 February 2025

Citation: Ryu, J.; Kim, T. Enhancing Hospital Data Security: A Blockchain-Based Protocol for Secure Information Sharing and Recovery. *Electronics* **2025**, *14*, 580. https://doi.org/10.3390/electronics14030580

1. Introduction

Recently, hacker attacks on hospitals have become a significant concern. Hackers may steal patient data or implant malware within hospital systems, rendering the data inaccessible [1]. One of the most malicious attacks is ransomware, where hackers lock down critical patient and medical data on the hospital's servers and demand payment in exchange for restoring access [2]. Even if the hospital pays the ransom, hackers often demand more money or fail to unlock the data. These types of attacks can occur even in well-structured hospital systems if, for example, operating system updates are neglected or malicious email attachments or links are mistakenly opened [3].

In particular, when a hospital needs to transfer a patient's medical information to another hospital due to relocation or deteriorating health, serious security issues can arise: (1) An attacker may impersonate a hospital and send attachments or links containing malware to attack the internal servers of the hospital. (2) There could be privacy breaches where an attacker intercepts and steals patient medical information during the transfer between hospitals. (3) An attacker might disrupt communication between hospitals. To prevent such attacks, a robust security system for secure interaction between hospitals is essential [4].

To address these security challenges, we propose a new protocol system that incorporates blockchain technology. This proposed protocol provides a solution in cases where a

hospital's internal system is compromised and server recovery is difficult by allowing the hospital to retrieve encrypted information from the blockchain. To securely transfer patient data and medical records between hospitals, the protocol records the relevant information on the blockchain and ensures that only verified institutions can interact by generating session keys for encryption. Additionally, when a patient wishes to transfer their treatment from one hospital to another, they often face the inconvenience of resubmitting insurance and identity verification documents. Our protocol addresses this issue by allowing insurance companies to be verified and registered on the blockchain, enabling them to access patient information and record the patient's insurance status securely.

Our proposed protocol is designed to use a private blockchain to facilitate secure information sharing between hospitals, government agencies, and insurance companies. To register on and access this blockchain, verification by the government agency is required. Once verified, the institution is registered on the blockchain along with the government agency's digital signature. The protocol is built using the government agency's elliptic curve cryptography (ECC)-based public–private key system, making it easier for other institutions to send encrypted messages to the government agency. Furthermore, if data issues arise within an institution's internal server, the institution can retrieve and restore its encrypted data from the private blockchain using its symmetric key.

Our main contributions are as follows:

- We propose a new protocol that enhances hospital data security. By utilizing blockchain technology, our protocol provides a preventive measure against ransomware attacks and inaccessible data within institutional servers, such as hospitals. It also offers a structured security protocol for data sharing between hospitals, ensuring that patient information is handled securely and protected against attackers.
- We demonstrate that the proposed protocol not only enables data recovery and secure data sharing through blockchain but also provides 11 security properties, including mutual authentication, forward and backward secrecy, user untraceability, and resistance to replay attacks.
- We evaluate the communication and computational costs of the proposed protocol and confirm that it is practically applicable in institutional settings. Thus, we establish that our proposed protocol is both practical and secure.

The remainder of this paper is structured as follows. In Section 2, we introduce related work. In Section 3, we describe ECC, the blockchain network, the system model, and the threat model. Section 4 provides a detailed explanation of the proposed protocol. Subsequently, in Sections 5 and 6, we evaluate the security and performance, respectively. In Section 7, we explore the implications of these results and discuss the limitations of our protocol. Finally, we conclude the paper in Section 8.

2. Related Work

Recently, there has been a surge of research on user authentication protocols for data sharing between medical sensor devices and servers. Masud et al. [5] proposed a lightweight and anonymity-preserving user authentication scheme for IoT-based healthcare. However, their approach is vulnerable to offline secret attacks and insider attacks. To address these issues, Kim et al. [6] developed a lightweight user authentication framework for medical IoT in 2023. Nevertheless, Zhou et al. [7] pointed out that Kim et al. [6]'s scheme does not guarantee forward and backward secrecy. In 2024, Zhou et al. [7] proposed a new device authentication and key exchange protocol for IoT that includes a Physical Unclonable Function (PUF) to enhance security.

Research on authentication protocols that utilize blockchain for data storage has also been active since 2022. Wang et al. [8] proposed an authentication protocol for wireless med-

ical sensor networks using blockchain. However, Yu and Park [9] identified vulnerabilities in Wang et al. [8]'s protocol, demonstrating that it was susceptible to man-in-the-middle attacks and session key disclosure attacks. In response, Yu and Park [9] proposed a new, enhanced security protocol for blockchain-based medical sensor networks. However, Kang et al. [10] discovered that critical parameters were being shared during smart contracts in Yu and Park [9]'s protocol. To address this, Kang et al. [10] introduced a new protocol in 2024 that improves security by having the gateway perform computations, thus addressing the vulnerabilities in Yu and Park [9]'s design.

Recently, there has been an increase in research utilizing blockchain to exchange patients' medical data [11]. Traditional medical records are stored in centralized databases of various hospitals and are used within the medical field. In contrast, with blockchain, patients can access their medical information and, if deemed appropriate, grant third parties access to their medical records. In 2019, Patel et al. [12] proposed a blockchain storage framework for medical imaging data. However, Patel et al. [12]'s model did not take security considerations into account. In 2019, Zhu et al. [13] introduced a cloud resource-sharing model and developed a data simulation study based on consensus-oriented blockchain technology. They assumed the use of Ethereum code to provide a blockchain environment and employed PoA (Proof of Authority) for block generation. Zhu et al. [13] also did not consider security requirements when registering data within the blockchain.

The recent leaks of medical data, including patients' personal information, occur more frequently when patient data are transferred from one healthcare institution to another rather than during transmission from IoT devices. Additionally, to prevent data loss caused by the increasingly serious issue of ransomware infections in institutions, it is essential to implement a system capable of fully recovering data across the entire hospital's medical system. Therefore, we propose a system for securely transferring patient data between hospitals. Furthermore, by storing hospital information on the blockchain, the proposed system ensures that data can be recovered in case of server failures in the future.

3. Preliminaries

In this section, we describe the foundational knowledge necessary for the proposed scheme, including elliptic curve cryptography (ECC), blockchain technology, the proposed system model, and the threat model. The detailed descriptions are as follows.

3.1. Elliptic Curve Cryptography

Elliptic curve cryptography (ECC) [14–17] is a public key cryptosystem that provides strong encryption with fewer key bits. Due to its ability to offer higher security with smaller key sizes compared to other public key systems like RSA, ECC has garnered significant attention in recent research [14,15]. It leverages the complexity of the discrete logarithm problem applied to elliptic curves. While the computational methods differ, the underlying properties remain similar to those of the discrete logarithm problem. First proposed by V. Miller [16] in 1985 and N. Koblitz [17] in 1987, ECC operates based on the following equation.

$$y^2 = x^3 + ax + b \quad mod\ p \qquad a, b \in F_p \tag{1}$$

In this equation, x and y represent the x-coordinate and y-coordinate on the Cartesian plane, respectively. The parameters a and b must satisfy the nonsingularity condition on the elliptic curve, ensuring that the curve does not have any cusps or self-intersections. The finite field F_p refers to the Galois field over a prime number p. By using this equation, it is known that the following security properties are satisfied.

1. Elliptic Curve Computational Diffie–Hellman Problem (ECCDHP): Given nmP, it is impossible to find nP and mP.

2. Elliptic Curve Decisional Diffie–Hellman Problem (ECDDHP): Given nP and mP, it is impossible to find nmP.
3. Elliptic Curve Discrete Logarithm Problem (ECDLP): Given P and nP, it is impossible to find n.

In this context, the expressions nP, mP, and nmP represent the results of performing the point multiplication operation on the point P n times, m times, and nm times, respectively. This means that the point P is multiplied by itself n times, m times, and nm times through the elliptic curve point addition operation.

3.2. Blockchain Network

Blockchain is a distributed database technology where transaction records are maintained and managed not by a central server, but by all participants in a Peer-to-Peer (P2P) network. By storing transaction data across multiple computers connected to the blockchain network, it ensures high security and transparency. Blockchain networks offer significant advantages over traditional centralized server models, such as protection against Distributed Denial of Service (DDoS) attacks and resilience against ransomware or other malicious software infections. If data are compromised on one server, they can be restored from other servers in the network.

When a new block is created in the blockchain, transactions are recorded, and each transaction is hashed to create a Merkle tree, which is then written into the block's header. To ensure that blocks are linked together, the header of the next block contains the hash value of the previous block's header. This process ensures the integrity and continuity of the blockchain. Figure 1 illustrates the structure of the blockchain.

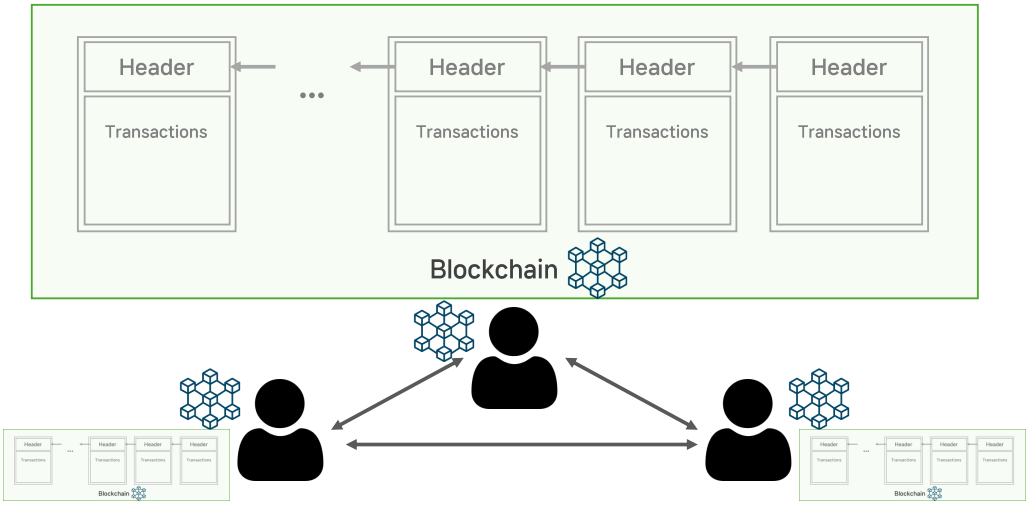

Figure 1. Blockchain structure.

Blockchain networks, due to their high security and transparency, are difficult to forge or tamper with, making them suitable for use in both public institutions and private sectors where data integrity is crucial. Beyond just storing transaction records, blockchain can also facilitate transaction agreements without the need for a trusted third party, a concept known as a smart contract. These characteristics of blockchain are being explored for applications in authentication, payment and remittance, securities trading, and more.

3.3. System Model

Our system is designed in compliance with the General Data Protection Regulation (GDPR) and the Health Insurance Portability and Accountability Act (HIPAA) [18,19].

According to Recitals 5(39) of the GDPR, any processing of personal data should be lawful and fair. Additionally, it should be transparent to natural persons that personal data concerning them are collected, used, consulted, or otherwise processed, and to what extent the personal data are or will be processed. HIPAA establishes detailed privacy rules regarding how Protected Health Information (PHI) can be used and disclosed, security rules specifying essential standards and safeguards to protect PHI, and breach notification rules requiring organizations to inform patients and relevant authorities in the event of a PHI data breach.

In our model, when a patient's personal data are transferred from one hospital to another, the patient is transparently informed of this process in advance. Our system is designed to adhere to the requirements of both GDPR and HIPAA, ensuring the protection of patient data at all times.

The proposed model consists of four nodes: the patient, hospital, government agency, and insurance company. In this model, it is assumed that the hospital, government agency, and insurance company share a private blockchain. Figure 2 illustrates the structure of the system model. The details of each node are as follows.

1. Patient: The patient visits a hospital and presents their identification card. There is no need for the patient to bring insurance documents, as the verification of their insurance status is automated. In cases where the patient initially visits a primary care hospital but their condition worsens, the patient may be referred to a secondary hospital. In such situations, the patient does not need to carry any documents to the secondary hospital.
2. Hospital: The hospital stores the patient's visit records on the blockchain, making them accessible for verification by the insurance company. Additionally, the hospital encrypts and stores the patient's medical records on the blockchain. In the event of issues such as ransomware infection affecting the hospital's internal servers, the hospital can recover the patient's information by accessing the encrypted data shared on the blockchain. Before accessing the blockchain, the hospital registers its ID and password with the government agency. During registration, the hospital's identity information is securely stored on the blockchain, ensuring that the ID and password are not exposed. Using this registered information, the hospital can prove its identity. Furthermore, the hospital can, if needed, create a session key with other hospitals or insurance companies to securely exchange encrypted patient information. This session key exchange is conducted with verification from the government agency.
3. Insurance Company: Similar to hospitals, the insurance company registers its ID and password with the government agency and stores this information on the blockchain. The insurance company can create a session key to securely share information with hospitals and access the blockchain to register or update the patient's insurance information.
4. Government Agency: The government agency assists in the creation of accounts for hospitals and insurance companies and records authorized transactions on the blockchain. The agency facilitates the generation of session keys between hospitals and between hospitals and insurance companies. Additionally, the government agency verifies whether each institution is properly registered on the blockchain.
5. Private Blockchain: The hospital, insurance company, and government agency each share the same blockchain across their servers. Each block is linked to the next by recording the hash value of the previous block in the header of the subsequent block. The transactions within each block include obfuscated account information from hospitals or insurance companies, as well as patient information and encrypted medical records entered by the hospital.

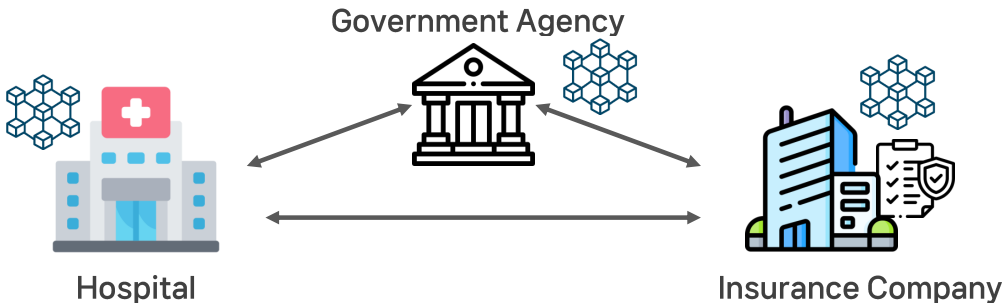

Figure 2. System model.

3.4. Threat Model

The fundamental principles of cybersecurity comprise the CIA triad: confidentiality, integrity, and availability. These three elements are considered the most critical concepts in information security. Confidentiality ensures that sensitive information is protected from unauthorized access attempts, meaning that data should only be accessible to authorized individuals. Integrity guarantees that data must not be altered during transmission and must not be modified by unauthorized parties. Availability ensures that approved parties can easily access the data, meaning that the systems used to store and present information must remain operational.

We consider attacking blockchain models that could potentially compromise these three principles. We propose a new attack model for the use of blockchain by referencing various attack models. This is a more robust model that incorporates elements from the traditional Dolev–Yao attack model. The details are as follows:

1. The attacker can intercept all messages exchanged over public channels between institutions (such as hospitals and insurance companies) and the government agency, as well as messages exchanged directly between institutions.
2. The attacker can register as a legitimate institution, gaining access to the blockchain as an authorized entity.
3. The government agency's private key is fully secure, and any digital signatures made by the government agency are trusted.
4. The attacker may attempt to exploit an outdated session key to predict or derive a new session key.

4. Proposed Scheme

This section provides detailed information about the protocol we propose. Our proposed protocol consists of the following phases: the registration phase for insurance companies and hospitals, the key exchange phase between hospitals, the phase where hospitals encrypt patient medical information and store it on the blockchain, the phase where a hospital transfers patient information to another hospital, and the password change phase for insurance companies and hospitals. We represent the registration step in Figure 3 and the authentication step in Figure 4. The notations used in the protocol are presented in Table 1, and the detailed descriptions are as follows.

H_i		G
Input ID_i, PW_i New $r_i \in \{0,1\}^l$ $AID_i = h((ID_i \oplus PW_i) \parallel r_i)$		
	$\xrightarrow{AID_i}$	
		$HID_i = h(AID_i \parallel x)$ Generate an asymmetric key pair (pk, sk) $\{AID_i, h(AID_i \parallel HID_i), E_{sk}(AID_i)\}$ in Blockchain
	$\xleftarrow{HID_i, pk, P}$	
$H_1 = h(ID_i \parallel PW_i) \oplus r_i$ $H_2 = h(PW_i \parallel ID_i) \oplus HID_i$ $H_3 = h(PW_i \parallel r_i \parallel HID_i)$		

Figure 3. Registration phase.

H_i		G		H_j
Input ID_i, PW_i $r_i = h(ID_i \parallel PW_i) \oplus H_1$ $HID_i = h(PW_i \parallel ID_i) \oplus H_2$ Confirm $H_3 = h(PW_i \parallel r_i \parallel HID_i)$ $AID_i = h((ID_i \oplus PW_i) \parallel r_i)$ Generate α, T_1 $A = \alpha P$ $M_1 = E_{pk}(AID_i \parallel HID_i \parallel T_1 \parallel AID_j \parallel A \parallel 1)$				
	$\xrightarrow{M_1}$	$AID_i \parallel HID_i \parallel T_1 \parallel AID_j \parallel A \parallel 1 = D_{sk}(M_1)$ Confirm $T_1, HID_i = h(AID_i \parallel x)$ Confirm $h(AID_i \parallel HID_i)$ in Blockchain $HID_j = h(AID_j \parallel x)$ Confirm $h(AID_j \parallel HID_j)$ in Blockchain $B = A \oplus HID_j, C = AID_i \oplus AID_j$ $I_1 = h(B \parallel C \parallel HID_j)$ Generate T_2 $M_2 = \{B \parallel T_2 \parallel C \parallel I_1\}$		
			$\xrightarrow{M_2}$	Login and Calculate r_j, AID_j, HID_j Confirm $T_2, I_1 = h(B \parallel C \parallel HID_j)$ Confirm $AID_i = C \oplus AID_j$ in Blockchain Generate β, T_3 $A = B \oplus HID_j$ $D = \beta P$ $SK = \beta A$
			$\xleftarrow{M_3}$	$M_3 = E_{pk}(AID_j \parallel HID_j \parallel T_3 \parallel AID_i \parallel D \parallel 0)$
		$AID_j \parallel HID_j \parallel T_3 \parallel AID_i \parallel D \parallel 0 = D_{sk}(M_3)$ Confirm $T_3, HID_j = h(AID_j \parallel x)$ Confirm $h(AID_j \parallel HID_j)$ in Blockchain $HID_i = h(AID_i \parallel x)$ Confirm $h(AID_i \parallel HID_i)$ in Blockchain Generate T_4 $I_2 = h(HID_i \oplus AID_j \oplus D)$		
	$\xleftarrow{M_4}$	$M_4 = \{D \parallel T_4 \parallel I_2\}$		
Confirm $T_4, I_2 = h(HID_i \oplus AID_j \oplus D)$ $SK = \alpha D$				

Figure 4. Authentication phase.

Table 1. Descriptions of symbols.

Symbol	Description
H_i	i-th hospital or insurance company
G	Government agency
(pk, sk)	G's public-private key pair in ECC
SK	The session key
x	G's secret key
ID_i	H_i's identity
PW_i	H_i's password
P	ECC generator
AID_i	H_i's anonymous identity
HID_i	H_i's hidden identity
$E_{key}(x)$	Encrypt x with key
$D_{key}(x)$	Decrypt x with key
T_i	i-th timestamp
$\oplus$	Bitwise XOR operation
$\parallel$	Concatenation operation

4.1. Registration Phase for Insurance Companies and Hospitals

In this phase, when a hospital is established or a new insurance company is launched, the relevant information of the hospital or company is stored on the blockchain through a government agency. When a hospital is established, the government agency verifies the information sent by the hospital, such as the licenses of medical professionals and business registration certificates. After verifying the newly established hospital, the government agency records the hospital's information on the blockchain along with the government's certification code. Similarly, when a new insurance company is launched, the government agency verifies the company's information and records it on the blockchain along with the government's certification code. The detailed descriptions are as follows.

1. The i-th insurance company or hospital H_i inputs its identity ID_i and password PW_i for registration. Then, it generates a random string r_i of length l. Next, it computes $AID_i = h((ID_i \oplus PW_i) \parallel r_i)$ and sends AID_i to the government agency G through a secure channel.
2. The government agency G verifies the hospital or company information of H_i and computes $HID_i = h(AID_i \parallel x)$ where x is a secret key for G. Using G's public key pair pk and sk, the G registers the following information on the blockchain: $\{AID_i, h(AID_i \parallel HID_i), E_{sk}(AID_i)\}$. Here, $E_{sk}(AID_i)$ represents the digital signature of G using its private key sk. Once the information of H_i has been successfully registered on the blockchain, G sends $\{HID_i, pk, P\}$ to H_i through a secure channel. The value P is used for elliptic curve computations, and although it includes modular arithmetic values, those details are omitted in this study.
3. H_i receives $\{HID_i, pk, P\}$ from G and verifies that the information $\{AID_i, h(AID_i \parallel HID_i), E_{sk}(AID_i)\}$ registered on the blockchain matches its AID_i. At this time, H_i can check whether G's digital signature, $D_{pk}(E_{sk}(AID_i))$, corresponds to its AID_i. After this verification, H_i proceeds to compute the following: $H_1 = h(ID_i \parallel PW_i) \oplus r_i$, $H_2 = h(PW_i \parallel ID_i) \oplus HID_i$, $H_3 = h(PW_i \parallel r_i \parallel HID_i)$. Finally, H_i stores $\{H_1, H_2, H_3, pk, P\}$ on its local device.

4.2. Authentication and Key Exchange Phase Between Hospitals

In this phase, two hospitals, H_i and H_j, exchange a session key to facilitate the transfer of patient information between them. To ensure that both institutions are legitimate and registered on the blockchain, they obtain verification from the government agency, G. After the verification, H_i and H_j exchange a session key that allows them to securely communicate with each other. The detailed descriptions are as follows.

1. First, H_i logs in by entering its ID_i and PW_i. Using the stored values $\{H_1, H_2, H_3, pk, P\}$, H_i performs the following calculation: $r_i = h(ID_i \parallel PW_i) \oplus H_1$, $HID_i = h(PW_i \parallel ID_i) \oplus H_2$. H_i verifies that its ID_i and PW_i are correct by checking against $H_3 = h(PW_i \parallel r_i \parallel HID_i)$. Next, H_i generates α for the creation of the session key and a timestamp T_1. After selecting the AID_j of hospital H_j with which it wants to communicate, H_i computes the following: $AID_i = h((ID_i \oplus PW_i) \parallel r_i)$, $A = \alpha P$ and $M_1 = E_{pk}(AID_i \parallel HID_i \parallel T_1 \parallel AID_j \parallel A \parallel 1)$. Then, H_i sends M_1 to the government agency, G, through a public channel.
2. G receives the encrypted document sent by H_i and decrypts it using its private key, sk: $AID_i \parallel HID_i \parallel T_1 \parallel AID_j \parallel A \parallel 1 = D_{sk}(M_1)$. If G receives a final message of 1, it performs the following. G checks whether the timestamp T_1 falls within the valid time. Next, G verifies HID_i using its secret value x and ensures that the AID_i and $HID_i = h(AID_i \parallel x)$ pair exists on the blockchain. Additionally, G computes $HID_j = h(AID_j \parallel x)$ using AID_j and verifies that the AID_j and HID_j pair is also present on the blockchain. Once all verifications are complete, G generates a

new timestamp, T_2, and computes the necessary values to create a message for H_j: $B = A \oplus HID_j$, $C = AID_i \oplus AID_j$, and $I_1 = h(B \parallel C \parallel HID_j)$. Finally, G computes $M_2 = \{B \parallel T_2 \parallel C \parallel I_1\}$ and sends it to H_j through a public channel.

3. Before receiving the message from G, H_j logs into its device in the same mannar as H_1. Upon logging in, H_j retrieves r_j, AID_j, and HID_j and generates β and timestamp T_3. Using the value of M_2 received from G, H_j verifies $I_1 = h(B \parallel C \parallel HID_j)$ and AID_i and then computes the following:$A = B \oplus HID_j$, $D = \beta P$. Afterward, H_j calculates the session key $SK = \beta A$ for communication with H_i and encrypts the message $M_3 = E_{pk}(AID_j \parallel HID_j \parallel T_3 \parallel AID_i \parallel D \parallel 0)$ to be sent to G.
4. G receives the encrypted document sent by H_j and decrypts it using its private key, sk: $AID_j \parallel HID_j \parallel T_3 \parallel AID_i \parallel D \parallel 0 = D_{sk}(M_3)$. If G receives a final message of 0, it performs the following. G checks whether timestamp T_3 falls within the valid time. Next, G verifies HID_j using its secret value x and ensures that the AID_j and $HID_j = h(AID_j \parallel x)$ pair exists on the blockchain. Additionally, G computes $HID_i = h(AID_i \parallel x)$ using AID_i and verifies that the AID_i and HID_i pair is also present on the blockchain. Once all verifications are complete, G generates a new timestamp T_4 and computes the necessary values to create a message for H_i: $I_2 = h(HID_i \oplus AID_j \oplus D)$. Finally, G computes $M_4 = \{D \parallel T_4 \parallel I_2\}$ and sends it to H_i through a public channel.
5. H_i receives M_4 from G and verifies T_4 and $I_2 = h(HID_i \oplus AID_j \oplus D)$. After the verification, H_i computes the session key $SK = \alpha D$ for communication with H_j.

4.3. Phase for Encrypting Patient Medical Information and Storing It on the Blockchain

When a patient visits a hospital, the hospital encrypts the patient's information and records it on the blockchain. The key elements of the patient's information include whether their identity has been verified by the government, whether there is valid insurance coverage for the medical services received, and the expiration date of the insurance. To ensure these details, verification from both the government agency and the insurance company is required. The detailed descriptions are as follows.

1. Before hospitals and insurance companies store detailed patient information, they generate their respective public–private key pairs and a symmetric key. The symmetric key is used to encrypt the patient's medical records and insurance coverage details, which are then stored on the blockchain. The public–private key pair is used by the hospital and insurance company to create digital signatures, allowing them to verify that the values were generated by them.
2. The hospital encrypts the patient's identity information and hospital details using the government agency's public key and sends it to the government agency. After verifying the patient's identity, the government agency updates the patient's visit records on the blockchain along with the government agency's digital signature.
3. The hospital verifies the patient's identity through the blockchain and, after treating the patient, encrypts the patient's medical records using the hospital's private key and updates them on the blockchain. Along with this, the hospital also updates the blockchain with a digital signature to confirm that the information was indeed created by the hospital.
4. The insurance company verifies that the patient's identity information, as certified by the government and registered on the blockchain, matches the insurance coverage details. The insurance company then updates the blockchain with information regarding the patient's insurance coverage. Along with this update, the insurance company also includes a digital signature to confirm that the information was indeed created by the insurance company.

Using the proposed protocol, hospitals can verify the patient's insurance information through the blockchain. This allows them to provide details on the insurance policies the patient holds whenever requested by the patient.

4.4. Phase for Transferring Patient Information from One Hospital to Another

When a patient's condition worsens, requiring transfer to a higher-level hospital, or when the patient relocates and needs treatment at a different hospital, the hospital must send the patient's medical records, including identity and insurance information, to the new hospital. The protocol for this process is as follows.

1. Each hospital generates a session key SK through the protocol described in Section 4.2 to securely encrypt and transfer the patient's information to one another.
2. The hospital sending the patient's information decrypts the patient's medical records stored on the blockchain. It then performs symmetric encryption using the session key SK, which was previously established with the receiving hospital. The sending hospital also attaches a digital signature to the encrypted information, allowing the receiving hospital to verify the authenticity of the data.
3. The receiving hospital decrypts the encrypted patient information using the session key SK. After verifying the patient's information, the hospital updates the blockchain by encrypting the acknowledgment of receiving the patient's information using its private key for symmetric encryption. Along with this update, the hospital also adds its digital signature to the blockchain to confirm the authenticity of the update.

4.5. Password Change Phase for Insurance Companies and Hospitals

In cases where an insurance company or hospital suspects a security threat due to an outdated password, they may wish to change their password. The method for each institution to perform a password change is as follows.

1. Each institution H_i enters its ID_i, old password PW_i^{old} and new password PW_i^{new} to initiate the password change process. After inputting the new password, H_i follows the login procedure. $r_i = h(ID_i \parallel PW_i^{old}) \oplus H_1^{old}$, $HID_i^{old} = h(PW_i^{old} \parallel ID_i) \oplus H_2^{old}$ and $AID_i^{old} = h((ID_i \oplus PW_i^{old}) \parallel r_i)$. H_i confirms $H_3^{old} = h(PW_i^{old} \parallel r_i \parallel HID_i^{old})$. H_i also computes the new $AID_i^{new} = h((ID_i \oplus PW_i^{new}) \parallel r_i)$ using the new password PW_i^{new}. Then, H_i encrypts the message $M_5 = E_{pk}(AID_i^{new} \parallel HID_i^{old} \parallel AID_i^{old} \parallel 11)$ using the G's public key pk and sends it to G via a public channel.
2. G decrypts the received M_5 from H_i as follows: $(AID_i^{new} \parallel HID_i^{old} \parallel AID_i^{old} \parallel 11) = D_{sk}(M_5)$. If the value following the previous AID_i^{old} is 11, G recognizes it as a password change request and proceeds with the following steps. G verifies HID_i^{old} using its secret value x and ensures that the AID_i^{old} and $HID_i^{old} = h(AID_i^{old} \parallel x)$ pair exists on the blockchain. Then, G constructs the new $HID_i^{new} = h(AID_i^{new} \parallel x)$ using its secret value x. Using G's public key pair pk and sk, the G updates the following information on the blockchain: $\{AID_i^{new}, h(AID_i^{new} \parallel HID_i^{new}), E_{sk}(AID_i^{new})\}$. Here, $E_{sk}(AID_i^{new})$ represents the digital signature of G using its private key sk. Finally, G sends $M_6 = AID_i^{new} \oplus HID_i^{new} \oplus HID_i^{old}$ to H_i via a public channel.
3. H_i receives M_6 from G and computes the new $HID_i^{new} = M_6 \oplus HID_i^{old} \oplus AID_i^{new}$. Using the new PW_i^{new} and HID_i^{new}, H_i then calculates the new values of H_1^{new}, H_2^{new}, and H_3^{new} as follows: $H_1^{new} = h(ID_i \parallel PW_i) \oplus r_i$, $H_2^{new} = h(PW_i^{new} \parallel ID_i) \oplus HID_i^{new}$, $H_3^{new} = h(PW_i^{new} \parallel r_i \parallel HID_i^{new})$. Finally, H_i updates H_1^{new}, H_2^{new}, and H_3^{new} on its device.

5. Security Analysis of the Proposed Scheme

In this section, we verify the security of the proposed protocol. We conducted an informal security analysis to ensure that the protocol satisfies 11 security properties. Additionally, we performed a formal security analysis using the security verification tool ProVerif. The detailed descriptions are as follows.

5.1. Formal Security Analysis

We performed a formal analysis of our proposed protocol using ProVerif 2.05 [20]. ProVerif is a tool for verifying the integrity of authentication protocols and simulating attacks in cryptographic security verification. It provides powerful features to analyze cryptographic protocols in scenarios with unlimited sessions and unbounded message space, making it widely used in protocol research [21–23].

We validated the correctness of our protocol using the code in Table 2. The first query, 'query idi:bitstring; inj-event(endHi(idi)) ==> inj-event(startHi(idi))', represents a correlation between the events 'endHi' and 'startHi', which is used to verify security properties. The result of this query can be one of the following two outcomes:

- Query inj-event(endHi(idi)) ==> inj-event(startHi(idi)) is true.
- Query inj-event(endHi(idi)) ==> inj-event(startHi(idi)) is false.

Table 2. ProVerif code for queries.

```
(*—queries—-*)
query idi:bitstring; inj-event(endHi(idi))  ==> inj-event(startHi(idi)).
query idg:bitstring; inj-event(endG(idg)) ==> inj-event(startG(idg)).
query idj:bitstring; inj-event(endHj(idj)) ==> inj-event(startHj(idj)).
query attacker(SK).

(*—process—*)
process
((!Hi) | (!Go) | (!Hj))
```

The first result, 'Query inj-event(endHi(idi)) ==> inj-event(startHi(idi)) is true', indicates that when the protocol started and Hi input idi, it was authenticated correctly. The second result, 'Query inj-event(endHi(idi)) ==> inj-event(startHi(idi)) is false', indicates that the authentication failed when idi was input. The same interpretation applies to other queries, such as 'query idg:bitstring; inj-event(endG(idg)) ==> inj-event(startG(idg))' and 'query idj:bitstring; inj-event(endHj(idj)) ==> inj-event(startHj(idj))'.

The fourth query, 'query attacker(SK)', checks whether the attacker can compromise SK. This query can also result in one of the following two outcomes:

- Query not attacker(SK[]) is true.
- Query not attacker(SK[]) is false.

The first result, 'Query not attacker(SK[]) is true', means that the attacker was unable to compromise SK, even when multiple SK values existed. This indicates that the protocol successfully defended against the attacker. On the other hand, the second result, 'Query not attacker(SK[]) is false', implies that there is a method by which the attacker could have successfully compromised SK. Our results are summarized in Table 3, and the detailed code is presented in Tables 4–7.

Table 3. ProVerif query results.

```
Verification summary:
Query inj-event(endHi(idi)) ==> inj-event(startHi(idi)) is true.
Query inj-event(endG(idg)) ==> inj-event(startG(idg)) is true.
Query inj-event(endHj(idj)) ==> inj-event(startHj(idj)) is true.
Query not attacker(SK[]) is true.
```

Table 4. ProVerif code for defining values and functions.

```
(*—channels—*)
free privateChannel1:channel [private].
free privateChannel2:channel [private].
free publicChannel1:channel.
free publicChannel2:channel.

(*—constants—-*)
const P:bitstring.
free IDi:bitstring [private].
free AIDi:bitstring.
free AIDj:bitstring.
free PWi:bitstring [private].
free IDj:bitstring [private].
free PWj:bitstring [private].
free x:bitstring [private].
free G:bitstring [private].
free sk:bitstring [private].

(*—shared key—-*)
free SK:bitstring [private].

(*—functions—-*)
fun concat(bitstring, bitstring):bitstring.
fun h(bitstring):bitstring.
fun xor(bitstring, bitstring):bitstring.
fun ECCkeygen(bitstring):bitstring.
fun mul(bitstring, bitstring):bitstring.
fun enc(bitstring, bitstring):bitstring.
fun dec(bitstring, bitstring):bitstring.
equation forall p:bitstring, q:bitstring; xor(xor(p, q), q) = p.
equation forall m:bitstring, n:bitstring; mul(mul(P, m), n) = mul(mul(P, n), m).
equation forall a:bitstring, key:bitstring; dec(enc(a, ECCkeygen(key)), key) = a.

(*—events—*)
event startHi(bitstring).
event endHi(bitstring).
event startG(bitstring).
event endG(bitstring).
event startHj(bitstring).
event endHj(bitstring).
```

Table 5. ProVerif code for Hi.

```
(*—Hi process—-*) let Hi =
new ri:bitstring;
let AIDi = h(concat(xor(IDi, PWi), ri)) in
out(privateChannel1,(AIDi));
in(privateChannel1, (XHIDi:bitstring, Xpk:bitstring, XP:bitstring));
let H1i = xor(h(concat(IDi, PWi)), ri) in
let H2i = xor(h(concat(PWi, IDi)), XHIDi) in
let H3i = h(concat(concat(PWi, ri), XHIDi)) in

event startHi(IDi);
new alpha:bitstring;
new T1:bitstring;
let A = mul(XP, alpha) in
let M1 = enc((AIDi, XHIDi, T1, AIDj, A, 1), Xpk) in
out(publicChannel1,(M1));
in(publicChannel1,(XXD: bitstring, XT4: bitstring, XI2: bitstring));
let XXI2 = h(xor(xor(XHIDi,AIDj),XXD)) in
if (XI2 = XXI2) then
let XSK = mul(XXD, alpha) in
event endHi(IDi).
```

Table 6. ProVerif code for G.

```
(*—-G process—-*)
let Go =
in(privateChannel1,(XAIDi:bitstring));
let HIDi = h(concat(XAIDi, x)) in
let pk = ECCkeygen(sk) in
out(privateChannel1, (HIDi, pk, P));

in(privateChannel2,(XAIDj:bitstring));
let HIDj = h(concat(XAIDj, x)) in
out(privateChannel2, (HIDj, pk, P));

event startG(G);
in(publicChannel1,(XM1:bitstring));
let (XAIDi:bitstring, XXHIDi:bitstring, XT1:bitstring, XAIDj:bitstring, XA:bitstring, 1) = dec(XM1, sk) in
if (XXHIDi = HIDi) then
let B = xor(XA, HIDj) in
let C = xor(XAIDi, XAIDj) in
let I1 = h(concat(concat(B, C), HIDj)) in
new T2:bitstring;
out(publicChannel2, (B, T2, C, I1));
in(publicChannel2,(XM3:bitstring));
let (XAIDj:bitstring, XXHIDj:bitstring, XT3:bitstring, XAIDi:bitstring, XD:bitstring, 0) = dec(XM3, sk) in
if (XXHIDj = HIDj) then
let I2 = h(xor(xor(HIDi, XAIDj), XD)) in
new T4:bitstring;
out(publicChannel1,(XD, T4, I2));
event endG(G).
```

Table 4 defines the variables and functions used in our ProVerif code. The channels ‘privateChannel1’ and ‘privateChannel2’ used during the registration phase are configured as private. We defined the elliptic curve constant ‘P’, as well as the identity ‘IDi’, anonymous identity ‘AIDi’, and password ‘PWi’ of ‘Hi’. Similarly, the identity, anonymous identity, and password for ‘Hj’ are defined as ‘IDj’, ‘AIDj’, and ‘PWj’.

Table 7. ProVerif code for Hj.

```
(*—Hj process—-*)
let Hj =
new rj:bitstring;
let AIDj = h(concat(xor(IDj, PWj), rj)) in
out(privateChannel2,(AIDj));
in(privateChannel2, (XHIDj:bitstring, XXpk:bitstring, XXP:bitstring));
let H1j = xor(h(concat(IDj, PWj)), rj) in
let H2j = xor(h(concat(PWj, IDj)), XHIDj) in
let H3j = h(concat(concat(PWj, rj), XHIDj)) in

event startHj(IDj);
in(publicChannel2, (XB:bitstring, XT2:bitstring, XC:bitstring, XI1:bitstring));
let XXI1 = h(concat(concat(XB, XC), XHIDj)) in
if (XXI1 = XI1) then
new beta:bitstring;
new T3:bitstring;
let XXA = xor(XB, XHIDj) in
let D = mul(XXP, beta) in
let SK = mul(XXA, beta) in
let M3 = enc((AIDj, XHIDj, T3, AIDi, D, 0), XXpk) in
out(publicChannel2, (M3));
event endHj(IDj).
```

The variable 'x' represents the government's secret key, while 'sk' is the government's private key. Although 'G' is not directly used, it is defined as the government's virtual identity. The session key 'SK' is defined as a private variable.

Additionally, the following functions are defined: the concatenate function, the hash function, the xor operation, the key generation function for ECC (elliptic curve cryptography), ECC multiplication, and encryption/decryption functions.

The start and end points of the login and authentication phases for 'Hi' are defined as 'startHi' and 'endHi', respectively. Similarly, the start and end points for the login and authentication phases of 'G' and 'Hj' are also defined.

Tables 5–7 describe the registration, login, and authentication phases performed by 'Hi', 'G', and 'Hj'. These phases are programmed according to the structure of the proposed protocol described in Section 4.

5.2. Informal Security Analysis

In this section, we verify the security of the proposed protocol. The proposed protocol provides six security properties: mutual authentication, session key agreement, a password verification process, user-friendly password change, forward and backward secrecy, and user untraceability. Additionally, it resists five types of attacks: replay attacks, user impersonation attacks, on/offline password guessing attacks, insider attacks, and man-in-the-middle attacks. The detailed descriptions are as follows.

5.2.1. Provide Mutual Authentication

The proposed protocol provides mutual authentication between two institutions when they exchange a session key by leveraging the government agency G and the blockchain. When H_i and H_j seek to mutually authenticate, the government agency G plays a crucial role. When G receives an encrypted message from H_i over a public channel, it verifies H_i's identity information against the records on the blockchain using its private key x. After successful verification, G forwards the message to H_j. Similarly, when G receives

an encrypted message from H_j, it verifies H_j's identity information against the blockchain records using its private key x and then forwards the message to H_i.

5.2.2. Provide Session Key Agreement

In the proposed protocol, institutions use symmetric key encryption to securely transmit patient information. For this, they must exchange the session key required for the symmetric encryption. This session key exchange process is conducted alongside mutual authentication. The detailed procedure for this process is described in Section 4.2.

5.2.3. Provide a Password Verification Process

In the proposed protocol, institution H_i performs login using an identity ID_i and password PW_i. The process of verifying whether the ID_i and PW_i are correct occurs twice: first, by checking $H_3 = h(PW_i \parallel r_i \parallel HID_i)$ to ensure the correct input, and second, by having the government agency G verify that the information matches the records AID_i, $h(AID_i \parallel HID_i)$ on the blockchain, thereby completing the password verification process.

5.2.4. Provide User-Friendly Password Change

In the proposed protocol, when a company or hospital H_i requests a password change, the process involves communication with the government agency through a public channel. Once the password is changed, the government agency must update the blockchain with the new AID_i and HID_i information. However, for the institution requesting the password change, such as a company or hospital, the process is simplified. After completing the legitimate login procedure, the institution can easily update the password by entering the new password and storing the updated H_1, H_2, and H_3.

5.2.5. Provide Forward and Backward Secrecy

The proposed protocol uses elliptic curve operations to generate session keys. This method of session key generation ensures that the next session key cannot be derived from the previous session key, and similarly, even if the next session key is known, the previous session key cannot be deduced. Therefore, the proposed protocol provides both forward and backward secrecy.

5.2.6. Ensure User Untraceability

In the proposed protocol, some messages are transmitted in plaintext, while others are encrypted. Information sent to the government agency over a public channel is encrypted using the government agency's public key. Encrypted messages are untraceable, meaning that the sender and the recipient cannot be identified. When the government agency forwards messages to hospitals or insurance companies, it sends them in plaintext. However, these plaintext messages consist of information that changes constantly, ensuring that they remain untraceable.

5.2.7. Resist Replay Attack

In the proposed protocol, timestamps are used during the session key exchange process. Each time a message is sent, the timestamp is verified before performing further computations. This mechanism ensures that replay attacks are not possible, as any attempt to reuse an old message with an outdated timestamp would be detected and rejected.

5.2.8. Resist User Impersonation Attack

In the proposed protocol, secret information such as HID_i is encrypted and transmitted during each exchange. The government agency then compares this encrypted HID_i with the information registered on the blockchain using its secret key x. Because of

this verification process, it is impossible for a malicious attacker to impersonate a legitimate hospital or insurance company. Therefore, the proposed protocol is resistant to user impersonation attacks.

5.2.9. Resist On/Off-Line Password Guessing Attack

In the proposed protocol, the passwords of institutions such as hospitals or insurance companies are stored by applying a hash function to H_1, H_2, and H_3 along with the ID_i and r_i information. Assume that under a very strong assumption, that an attacker manages to infiltrate the institution's internal systems and obtain this information. Even with access to these values, the attacker cannot perform a successful password guessing attack without additional information. Moreover, the values transmitted over the public channel do not reveal any useful information. Even if the attacker intercepts values like AID_i or HID_i, these values are already hashed with other data, making it impossible to deduce the password through guessing.

5.2.10. Resist Insider Attack

Assume that a scenario where a malicious attacker infiltrates an institution, such as a hospital or insurance company. Even if the attacker gains access to the computer storing H_1, H_2, H_3, pk, and P information, they still cannot obtain the ID_i or password PW_i information. Moreover, the attacker cannot generate valid AID_i or HID_i information using H_1 and H_2 alone, as these values depend on additional secure information. Therefore, the proposed protocol is resistant to insider attacks.

5.2.11. Resist Man-in-the-Middle Attack

When institutions such as hospitals or insurance companies are registered, the government agency stores their information on the blockchain, and the institutions verify that their information is correctly recorded. Even if an attacker attempts to intervene in this process, they cannot modify the blockchain data, making such an attack impossible. Additionally, during the session key exchange over a public channel, even if an attacker intercepts the values and attempts to perform calculations, they cannot pass the verification steps, as secret values are checked during the process. Therefore, the proposed protocol is resistant to man-in-the-middle attacks, even when conducted over a public channel.

6. Performance Analysis of the Proposed Scheme

In this section, we evaluate the performance of the proposed study. We assessed the performance of the proposed protocol in terms of computational cost and communication cost. The performance of the proposed protocol was measured based on the performance of previous studies by Kang et al. [10] and Kim et al. [23]. Their experimental environments are in Table 8.

Table 8. Experimental environment.

	Kang et al. [10]	**Kim et al. [23]**
CPU	Intel Core i7-8700 3.20 GHz	Intel Core i7-8565U 1.80 GHz
RAM	48 GB	16 GB
OS	Win10	Win10
Software	Python Cryptography Library	Java Development Kit 17

Let T_h represent the time for a hash function operation, T_E for ECC encryption, T_D for ECC decryption, and T_M for ECC multiplication. In the registration phase, H_i performs four hash operations, while G performs one hash operation and one ECC encryption operation. The details for the registration phase are summarized in Table 9. In the authentication phase,

H_i performs a total of five hash operations, two ECC multiplication operations, and one ECC encryption operation. G performs ten hash operations and two ECC decryption operations. H_j, excluding the login procedure, performs one hash operation, two ECC multiplication operations, and one ECC encryption operation. The detailed breakdown for the authentication phase is provided in Table 10. Finally, in the password change phase, H_i performs eight hash operations and one ECC encryption operation, while G performs three hash operations, one ECC encryption, and one ECC decryption operation. The specifics of the password change phase are detailed in Table 11. The results obtained by experimenting with these values in the experimental environment described in Table 8 are shown in Figure 5.

Table 9. Computational cost in the registration phase.

H_i	G	Total Costs
$4T_h$	$1T_h + 1T_E$	$5T_h + 1T_E$

Table 10. Computational cost in the authentication phase.

H_i	G	H_j	Total Costs
$5T_h + 2T_M + 1T_E$	$10T_h + 2T_D$	$1T_h + 2T_M + 1T_E$	$16T_h + 4T_M + 2T_E + 2T_D$

Table 11. Computational cost in the password change phase.

H_i	G	Total Costs
$8T_h + 1T_E$	$3T_h + 1T_E + 1T_D$	$11T_h + 2T_E + 1T_D$

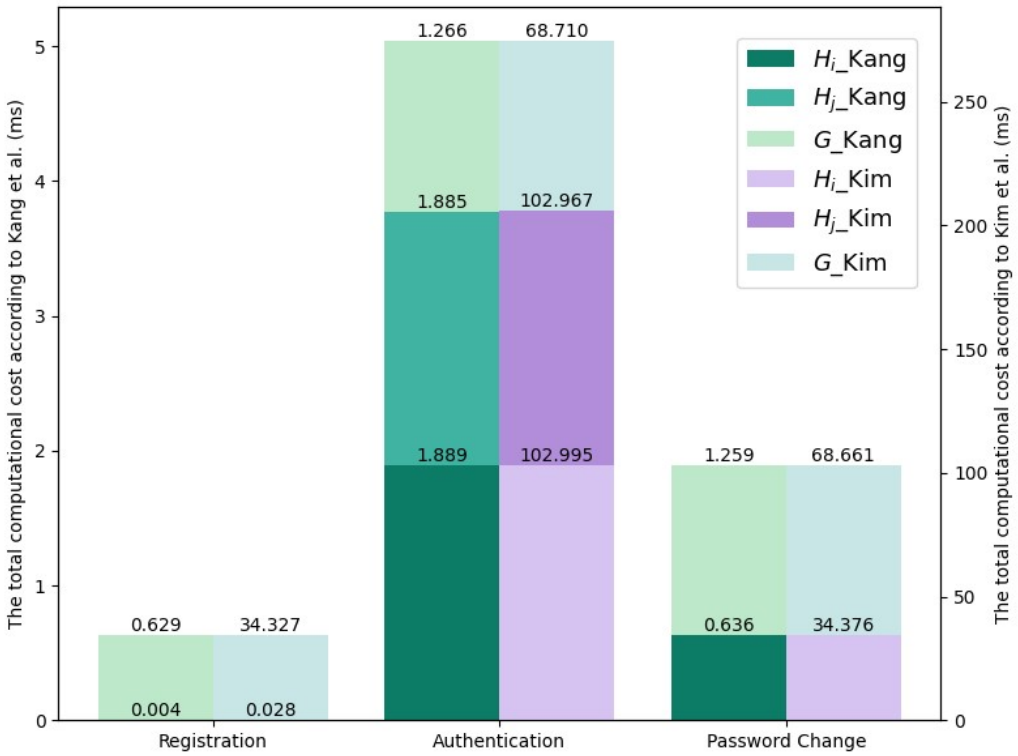

Figure 5. Computational cost.

We also evaluate the communication cost of the proposed protocol. We assume that the hash function is based on SHA-256, resulting in 256-bit outputs. The values for the elliptic curve cryptography parameters, such as pk and P, are set to 320 bits. Timestamps

are 32 bits, and encrypted values are assumed to be in 256-bit segments. This information is summarized in Table 12 and Figure 6, and the detailed breakdown is as follows.

Using these values, the communication cost for the registration phase can be calculated as follows: AID_i is 256 bits, so the cost for H_i is 256 bits. For G, the costs for HID_i, pk, and P are 256, 320, and 320 bits, respectively, resulting in a total communication cost of 896 bits for G. Therefore, the total communication cost for the registration phase is 1152 bits.

In the authentication phase, the communication cost of the message M_1 sent by H_i is calculated as follows: AID_i, HID_i, T_1, AID_j, A, and 1 are combined and encrypted, which sums to 256 + 256 + 32 + 256 + 320 + 1 bits. Rounded up to 256-bit segments, this results in 1280 bits. For M_2, which is sent by G in plaintext, the values B, T_2, C, and T_1 total 320 + 32 + 256 + 256 = 864 bits. For M_4, also sent by G in plaintext, the values D, T_3, and I_2 total 320 + 32 + 256 = 608 bits, bringing the total communication cost for messages sent by G to 1472 bits. The message M_3 sent by H_j includes AID_j, HID_j, T_3, AID_i, D, and 0, which are combined and encrypted, resulting in 256 + 256 + 32 + 256 + 320 + 1 bits. Rounded up to 256-bit segments, this results in 1280 bits. Therefore, the total communication cost for the authentication phase is 4032 bits.

Finally, for the password change phase, the communication cost for the message M_5 sent by H_i is calculated by combining AID_i^{new}, HID_i^{old}, AID_i^{old}, and 11, which results in 256 + 256 + 256 + 2 bits. Rounded up to 256-bit segments, this results in 1024 bits. The cost for M_6, sent by G, which includes all hashed values combined with an exclusive OR operation, is 256 bits. Thus, the total communication cost for the password change phase is 1280 bits.

Table 12. Communication cost of the proposed scheme.

	H_i	G	H_j	Total Costs
Registration phase	256 bits	896 bits	-	1152 bits
Authentication phase	1280 bits	1472 bits	1280 bits	4032 bits
Password change phase	1024 bits	256 bits	-	1280 bits

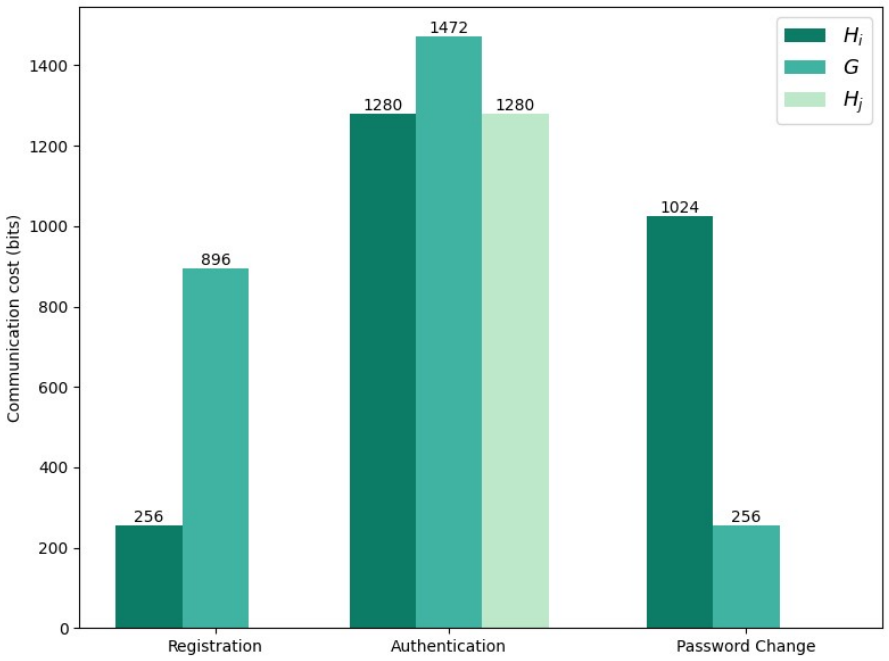

Figure 6. Communication cost.

7. Discussion

The proposed protocol currently considers the case where two hospitals participate in a single blockchain. Even if the number of participating institutions, such as hospitals or insurance companies, increases, the verification of each transaction will be automatically processed within the blockchain. However, as the number of participants and medical records grows, leading to an accumulation of medical data in the blockchain, the storage burden on blockchain participants may increase. To address this issue, potential solutions include implementing a policy to automatically remove outdated data (e.g., data older than 10 years) or utilizing off-chain storage for individual data points.

The proposed protocol was validated using ProVerif to ensure its integrity, and performance analysis confirmed that the computational and communication costs are reasonable. Therefore, for the proposed protocol to be integrated into existing hospital management systems, additional policies should be established to (1) utilize blockchain participants as relevant institutions and (2) encrypt patients' medical information before storing it on the blockchain.

8. Conclusions

In this study, we proposed a blockchain-based protocol for securely sharing and storing patient data among healthcare institutions such as hospitals and insurance companies. The proposed protocol, based on a private blockchain, includes a recovery system that ensures data can be restored even if an institution's server encounters issues. It also provides 11 security properties, including forward and backward secrecy, user untraceability, and resistance to replay attacks. Finally, we evaluated the performance of the proposed protocol and demonstrated its practical feasibility.

In future work, we plan to develop a protocol that directly integrates IoT devices used in healthcare institutions with blockchain systems. We will consider building an automated system that encrypts and stores patient medical data on the blockchain, enabling patients to have direct access by assigning them a blockchain interface. Additionally, as this study focuses on technical aspects, we plan to address guidelines or frameworks for integrating the blockchain protocol into existing hospital IT systems in future research.

Author Contributions: Conceptualization, J.R. and T.K.; Methodology, J.R.; Software, J.R.; Validation, J.R.; Writing—original draft, J.R.; Writing—review & editing, J.R. and T.K.; Supervision, T.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2023-00239728). The present Research has been conducted by the Research Grant of Kwangwoon University in 2022.

Data Availability Statement: Data are contained within the article.

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

References

1. Zhang, P.; Han, W.; Liu, Q. Research on Predicting the Mental Health of College Students with Prediction Models based on Big Data Technology. *IEIE Trans. Smart Process. Comput.* **2024**, *13*, 393–401. [CrossRef]
2. Yu, P. Design of a Blockchain based Data Security Guarantee System for Logistics Systems. *IEIE Trans. Smart Process. Comput.* **2024**, *13*, 402–413. [CrossRef]
3. Cha, H.; Kim, I.K.; Kim, T. Using a random forest to predict quantized reuse distance in an SSD write buffer. *Computing* **2024**, *106*, 3967–3986. [CrossRef]
4. Ryu, J.; Kang, D.; Lee, H.; Kim, H.; Won, D. A secure and lightweight three-factor-based authentication scheme for smart healthcare systems. *Sensors* **2024**, *20*, 7136. [CrossRef]

5. Masud, M.; Gaba, G.S.; Choudhary, K.; Hossain, M.S.; Alhamid, M.F.; Muhammad, G. Lightweight and anonymity-preserving user authentication scheme for IoT-based healthcare. *IEEE Internet Things J.* **2021**, *9*, 2649–2656. [CrossRef]
6. Kim, K.; Ryu, J.; Lee, Y.; Won, D. An improved lightweight user authentication scheme for the internet of medical things. *Sensors* **2023**, *23*, 1122. [CrossRef] [PubMed]
7. Zhou, X.; Wang, S.; Wen, K.; Hu, B.; Tan, X.; Xie, Q. Security-Enhanced Lightweight and Anonymity-Preserving User Authentication Scheme for IoT-Based Healthcare. *IEEE Internet Things J.* **2024**, *11*, 9599–9609. [CrossRef]
8. Wang, W.; Chen, Q.; Srivastava, Y.G.; Gadekallu, T.R.; Alsolami, F.; Su, C. Blockchain and PUF-based lightweight authentication protocol for wireless medical sensor networks. *IEEE Internet Things J.* **2022**, *9*, 8883–8891. [CrossRef]
9. Yu, S.; Park, Y. A robust authentication protocol for wireless medical sensor networks using blockchain and physically unclonable functions. *IEEE Internet Things J.* **2022**, *9*, 20214–20228. [CrossRef]
10. Kang, T.; Woo, N.; Ryu, J. Enhanced Lightweight Medical Sensor Networks Authentication Scheme Based on Blockchain. *IEEE Access* **2024**, *12*, 35612–35629. [CrossRef]
11. Ratta, P.; Kaur, A.; Sharma, S.; Shabaz, M.; Dhiman, G. Application of blockchain and internet of things in healthcare and medical sector: Applications, challenges, and future perspectives. *J. Food Qual.* **2021**, *2021*, 7608296. [CrossRef]
12. Patel, V. A framework for secure and decentralized sharing of medical imaging data via blockchain consensus. *Health Inform. J.* **2019**, *25*, 1398–1411. [CrossRef] [PubMed]
13. Zhu, X.; Shi, J.; Lu, C. Cloud health resource sharing based on consensus-oriented blockchain technology: Case study on a breast tumor diagnosis service. *J. Med. Internet Res.* **2019**, *21*, e13767. [CrossRef] [PubMed]
14. Abdaoui, A.; Erbad, A.; Al-Ali, A.K.; Mohamed, A.; Guizani, M. Fuzzy elliptic curve cryptography for authentication in Internet of Things. *IEEE Internet Things J.* **2021**, *9*, 9987–9998. [CrossRef]
15. Yang, W.; Hou, C.; Wang, Y.; Zhang, Z.; Wang, X.; Cao, Y. SAKMS: A Secure Authentication and Key Management Scheme for IETF 6TiSCH Industrial Wireless Networks Based on Improved Elliptic-Curve Cryptography. *IEEE Trans. Netw. Sci. Eng.* **2024**, *11*, 3174–3188. [CrossRef]
16. Miller, V.S. Use of elliptic curves in cryptography. In *Advances in Cryptology, CRYPTO'85*; Springer: Berlin/Heidelberg, Germany, 1986.
17. Koblitz, N. Elliptic curve cryptosystems. *Math. Comput.* **2024**, *48*, 203–209. [CrossRef]
18. Principles of Data Processing. Intersoft Consulting. Available online: https://gdpr-info.eu/recitals/no-39/ (accessed on 30 January 2025).
19. Health Insurance Portability and Accountability Act of 1996 (HIPAA). Public Health Law. Available online: https://www.cdc.gov/phlp/php/resources/health-insurance-portability-and-accountability-act-of-1996-hipaa.html (accessed on 30 January 2025).
20. Blanchet, B.; Smyth, B.; Cheval, V.; Sylvestre, M. *ProVerif 2.00: Automatic Cryptographic Protocol Verifier*; User Manual and Tutorial. Available online: https://bblanche.gitlabpages.inria.fr/proverif/manual.pdf (accessed on 30 January 2025).
21. Zhang, J.; Yang, L.; Cao, W.; Wang, Q. Formal analysis of 5G EAP-TLS authentication protocol using proverif. *IEEE Access* **2020**, *8*, 23674–23688. [CrossRef]
22. Edris, E.K.K.; Aiash, M.; Loo, J. Formal verification of authentication and service authorization protocols in 5G-enabled device-to-device communications using ProVerif. *Electronics* **2021**, *10*, 1608. [CrossRef]
23. Kim, K.; Ryu, J.; Lee, H.; Lee, Y.; Won, D. Distributed and Federated Authentication Schemes Based on Updatable Smart Contracts. *Electronics* **2023**, *12*, 1217. [CrossRef]