A Blockchain-Assisted Key Generation Electric Health Records Sharing Scheme

Qiao Zhang[2], Xiubo Chen[1,2], Haseeb Ahmad[3], Gang Xu[1,4](✉), and Yixian Yang[2]

[1] Guizhou Provincial Key Laboratory of Public Big Data, GuiZhou University, Guizhou 550025, Guiyang, China
gangxu_bupt@163.com

[2] State Key Laboratory of Networking and Switching Technology, Information Security Center, Beijing University of Posts and Telecommunications, Beijing 100876, China

[3] Department of Computer Science, National Textile University, Faisalabad 37610, Pakistan

[4] School of Information Science and Technology, North China University of Technology, Beijing 100144, China

Abstract. Electronic health records (EHRs) contain a large amount of private data of patients. Once these data are compromised during the process of sharing, it may threaten the patients' privacy. In this paper, a novel blockchain-assisted electronic health record sharing scheme is proposed, which utilizes attribute-based encryption (ABE) based on the consortium blockchain to realize the privacy protection in the sharing process of EHRs. Firstly, the master key is negotiated by all consensus nodes of consortium blockchain, no one knows the specific value of the master key. Consensus nodes are acted by medical institutions, they also responsible for generating and managing users' private keys. Secondly, with the help of powerful cloud services, pre-decryption and ciphertext retrieval assignments are outsourced to cloud services for reducing the burden of blockchain. Thirdly, this scheme is also based on searchable encryption, which allows for quick ciphertext lookup from cloud services. Data owners can generate ciphertext indexes for their private data, and data users with retrieval trapdoors can quickly retrieve ciphertexts. Finally, the security and performance of blockchain-assisted electronic health record sharing scheme are analyzed in detail, the results show that our scheme is safe and feasible.

Keywords: Blockchain · Electronic health records · Attribute-based encryption · Data sharing

1 Introduction

With the development of technology in healthcare systems, the scale of medical data has grown rapidly. Among them, EHRs [1] are widely used because they contain all health data of patients, including medical records, medications, and experience reports, etc. In the traditional medical system, EHRs are stored on the internal networks of medical institutions [2] and managed by dedicated internal staff only, resulting in the following problems. On the one hand, the EHRs data between different medical institutions do not interoperate, which triggers the data island effect. Patients need to carry multiple

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022

copies of cases for inter-hospital visits, which brings inconvenience to the cross-hospital diagnosis and treatment [3]. On the other hand, EHRs are used without the patients' consent, which may lead to the leakage of patient privacy. Furthermore, EHRs data are also of great value and easy to be attacked by attackers [4]. To address the above issues and enable secure sharing of EHRs data, a challenging problem is how to achieve fine-grained access control for patients to their EHRs.

ABE is one of the most effective methods to ensure confidentiality while achieving fine-grained access control. ABE was first proposed by Sahai et al. [5], which refers to that the ciphertext can only be decrypted if it meets specific attribute requirements. When in the healthcare system, patients can set adequate attributes for their EHRs data to allow access, such as attending hematologist, nurse-in-charge, etc. Doctors can take their access control structure to match the attributes set by patients. Once the patients' attribute requirements are met, doctors can decrypt and access the data. Blockchain was first proposed by Nakamoto [6] in 2008. It is considered to be a distributed database [7] with decentralization, transparency, and non-tamper ability. There is a special program that can interact with the blockchain called smart contract [8]. It can execute automatically according to a pre-determined program. Since the execution process does not require the involvement of a third party, the results of the execution of smart contracts are trusted. To enable secure sharing of EHRs, the existing solutions [2, 9–13] mainly use ABE for fine-grained access control, where encrypted data are stored on a special server and all access actions are recorded on the blockchain with a smart contract. Wang et al. [2] proposed a C-AB/IB-ES scheme combined with ABE and Identity-based encryption (IBE) to achieve access control. In their scheme, patients first sign their data using Identity-Based Signatures (IBS) to ensure data integrity. All data are stored on a healthcare cloud server and the encryption process is handled by the hospital. Wang et al. [9] believed that the patient can also be the subject of access control. The patient generates and distributes the attribute private key to those who are allowed to access it. And the smart contract records the list of users who are allowed to access it. Huang et al. [10] write every query and write operations of EHRs into the blockchain, which ensures traceability. Naresh et al. [11] utilize the consortium chain to store the hash value of EHRs to ensure integrity. Searchable encryption [14] technology can realize fast retrieval of ciphertext without revealing the plaintext. When there are large-scale encrypted EHRs stored on the server, searchable encryption can be used to accelerate the search. Reference [12, 13] also combines searchable encryption into attribute encryption to improve the speed of ciphertext retrieval. In these schemes, the blockchain is used only as a decentralized database to store access records, not involved in trusted computing. And they all rely on a centralized third-party authority to generate and manage the private keys. In reality, there is no guarantee that a third-party authorized authority will always be credible.

With the rise of the cloud servers, some services related to EHRs are transferred to cloud servers. Cloud servers are generally considered to have the unlimited computing power and can perform assigned tasks as required. The low price, as well as infinite storage space have drawn more people's attention. They [15–18] store the encrypted data on it, or use cloud services to assist in computing. Hua et al. [15] pointed out that the EHRs ought to be outsourced to cloud for storage after encryption, which ensures patients' privacy safety while reducing the strain on local systems. To address the issue of high computational overhead in ABE decryption, Zhang et al. [16] proposed a matching stage before decryption scheme, which improves decryption efficiency for outsourced data storage in cloud computing. IPFS is also one of cloud storage services. It allows data to be shared across multiple organizations while also avoiding single points of failure. References [17, 18] store the encrypted data on IPFS. Despite the tremendous benefits, security and privacy remain cloud servers' most important considerations. On the one hand, cloud servers allow access on the public network, which means that an attacker can also access the data. On the other hand, cloud services are often provided by third parties. It is unknown whether third-party institutions are always credible [19, 20]. Hence, the security of cloud services is significant and worth considering.

Through the analysis of the existing schemes, we remark that the security of the existing EHRs sharing systems is mostly based on a centralized third-party authority. However, a single third-party authority will also have a single point of failure, especially, it's difficult to find an always trusted third-party authority in reality. To address this problem, we propose a blockchain-assisted key generation EHRs sharing scheme. Our specific contributions are listed below.

1. We propose a EHRs sharing scheme without the third-party trusted authorized authority. Compared with the existing EHRs scheme, our scheme removes the third-party trusted authorized authority. Medical institutions play the role of consortium chain nodes to realize the safe sharing of EHRs among them. We use the consortium blockchain nodes to generate and manage the master key and attribute private key. All access operations need to be recorded on the chain.
2. In our scheme, we outsource the attribute pre-decryption and ciphertext retrieval assignments that consume a lot of computing power to the cloud servers, which takes pressure off the blockchain system while ensuring security.
3. We propose a EHRs sharing scheme also based on a searchable encryption. Data users with search trapdoors can retrieve data quickly from cloud servers.
4. The proposed scheme is feasible in practice. We implement our scheme based on the open-source cryptographic library and build a consortium blockchain network on Hyperledger Fabric. Performance analysis shows that our scheme can achieve fine-grained access control and fast decryption of EHRs.

The structure of this paper is as follows: Sect. 2 briefly introduced the basic knowledge. Our scheme architecture, system procedure, and security model are presented in Sect. 3. We introduced our scheme in detail in Sect. 4. After security proof and performance analysis in Sect. 5, we summarize the scheme in the last section.

2 Preliminaries

2.1 Bilinear Map

For two cyclic groups G_a and G_b with prime order p. Suppose that $e : G_a \times G_a \to G_b$ is a general map, we call e as a bilinear map [21] if e satisfies the following three properties.

- Bilinearity: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $g_1, g_2 \in G_a$ and $a, b \in Z_p$.

- Non-degeneracy: There exits $g_1, g_2 \in G_a$ such that $e(g_1, g_2) \neq 1$, where 1 is the unit of G_b.
- Computability: For all $g_1, g_2 \in G_a$, $e(g_1, g_2)$ can be calculated quickly in a polynomial-time.

2.2 Pedersen (*k*, *N*) Secret Share (PSS)

Suppose there are n participants $P_1, P_2, P_3..., P_n$ who are equal to each other. They select a sub-secret S_i respectively, and all sub-secrets add up to the main secret S in the form of an algebraic sum. Denote G_p is a finite field with a large prime order p. The Pedersen (k, n) Secret Share [22] protocol is composed of the next four steps.

1. Main-secret production: Each participant picks a random number $N_i \in Z_p$ as the sub-secret, and adds them to generate the main-secret S by Eq. (1).

$$S = \sum_{i=1}^{n} N_i \tag{1}$$

2. Sub-share production: All participants independently execute the Shamir secret sharing algorithm [23]. Each participant P_j randomly selects a k-1 degree of polynomial $f_j(x)$ and sets $f_j(0) = N_j$. Then, P_j computes n sub-shares $ss_{ji} = f_j(x_i)$ for $i = 1, 2, 3..., n$ and sends ss_{ji} to P_i through a secret channel.
3. Main-share production: After receiving n sub-share ss_{ij} for $i = 1, 2, 3..., n$, P_j calculates its master-share ms_j with Eq. (2).

$$ms_j = \sum_{i=1}^{n} ss_{ij} \tag{2}$$

4. Main-secret recovery: If there are more than k participants $P_i \in P_R$, the main-secret can be recovered with Eq. (3).

$$S = \sum_{P_i \in P_R} ms_i \prod_{P_j, P_i \in P_R, j \neq i} \frac{j}{j-i} (\mathrm{mod} p) \tag{3}$$

2.3 Decision Bilinear Diffie-Hellman (DBDH) Assumption

Let G and G_T be two cyclic groups with prime order p, and e is a bilinear map. Randomly select $x, y, z \in Z_p$ and $T \in G_T$, denote $X = g^x$, $Y = g^y$, $Z = g^z$. It is difficult to determine whether T is equal to $e(g, g)^{xyz}$ in any probabilistic polynomial-time [24].

3 System Design

3.1 System Architecture

The architecture of our scheme is shown in Fig. 1, which contains five participants, BC, Patients, IPFS, CSP, Data Users and CSP.

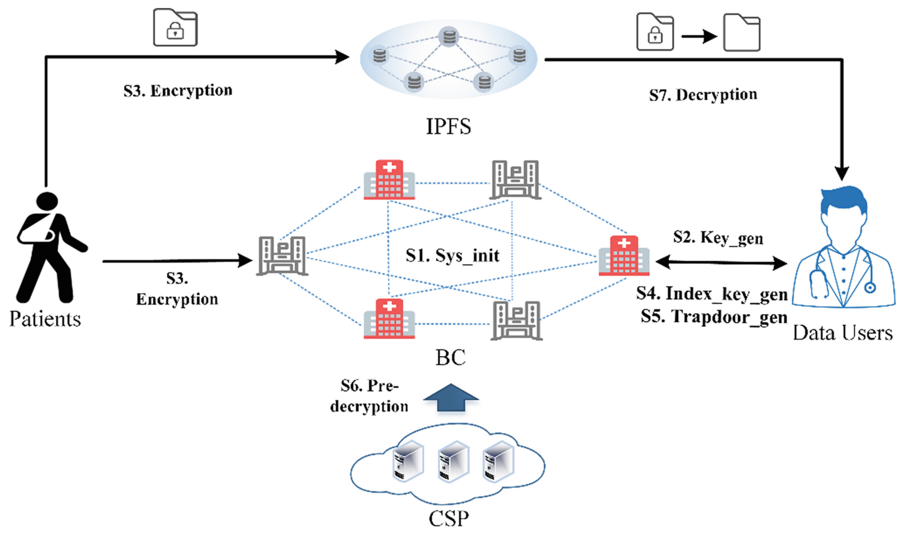

Fig. 1. System architecture.

BC: A consortium blockchain composed of multiple medical institutions including regulatory authorities. BC is responsible for generating and managing the master key and users' private keys. And it is also in charge of assigning outsourced decryption or retrieval assignments, and recording all users' operations on the chain.

Patients: The data owner in the system. After the treatment in the hospital, patients get medical data from the doctor, such as their own cases or medication status. Then they select appropriate keywords and attribute sets according to the data content. After encrypting the data and calculating the index, patients upload the encrypted data to IPFS, index to BC for recording.

DU: Data user in the system. DU can be doctors in other hospitals, staff from regulatory authorities, etc. If attribute set embedded in DU's access structure satisfies the patient's attribute requirement, he/she can obtain the Patient's data. At the same time, DU can also generate search trapdoors to search ciphertexts.

CSP: Cloud service provider. It is responsible for the index-retrieval and pre-decryption tasks assigned by BC. We assume that CSP has high computing power and always keeps online. In addition, we suppose that CSP is honest and curious about security.

IPFS: The IPFS is built by multiple medical institutions. We use IPFS to store EHRs from various healthcare facilities, avoiding the single point of failure associated with centralized storage.

3.2 System Procedure

S1. *System initialization*: All blockchain nodes perform $SysInit(1^{\lambda}) \rightarrow (MK, PK)$ to generate the public key PK and master key MK. Especially, MK is generated through all blockchain nodes by PSS protocol, no one knows the exact master key value.

S2. *Key generation*: In this subsection, all blockchain nodes generate two kinds of keys: outsourced private key SK_{out} and attribute private key SK_{attr}. $KeyGen(\mathbb{A}, PK, MK) \rightarrow (SK_{attr}, SK_{out})$ is performed by BC. It takes the user's access structure $\mathbb{A}$, global public key PK, master key MK as input and outputs the user's outsourced private key SK_{out}, the user's attribute private key SK_{attr}. Finally, BC sends the SK_{attr} and SK_{out} to DU in a secret channel.

S3. *Encryption*: First, patients select appropriate access attributes and keywords according to data content. Then he/she runs AES encryption over the data, and uploads the AES ciphertext to IPFS. Next, he/she executes $Encrypt(K_{aes}, PK, w) \rightarrow CT$ for fine-grained access control, and gets attribute ciphertext CT. Patients also execute $IndexGen(PK, CT, KW) \rightarrow Ix(kw)$ to generate the index $Ix(kw)$. Finally, he/she submits tuple $(CT, Ix(kw))$ to BC for recording and latter search.

S4. *Index key generation*: First, BC executes the algorithm $IndexKeyGen(PK, BF_{\beta}, \mathbb{A}) \rightarrow IK$. Among them, $\mathbb{A}$ is DU's access structure, BF_{β} is a commitment value. This algorithm outputs the query private key IK.

S5. *Trapdoor generation*: DU performs the algorithm $TrapDoor(PK, IK, kw, \beta) \rightarrow Td_{kw}$. Among the inputs, kw is the keyword for search and β is a blinding factor. This algorithm outputs the trapdoor Td_{kw}.

S6. *Pre-decryption*: Pre-decryption is run by CSR. Firstly, DU submits trapdoor Td_{kw} to BC. Then BC commissions CSR to execute the algorithm $Search(Td_{kw}, (CT, Ix(kw))) \rightarrow \perp/CT$. If keywords in Td_{kw} satisfies one of the index-set of $Ix(kw)$, CSR will run $Pre-decrypt(CT, PK, SK_{out}) \rightarrow Q_{pre}$ to gain partial ciphertext Q_{pre} and return it to BC. Finally, BC sends Q_{pre} to DU.

S7. *Local decryption*: After receiving the partial ciphertext Q_{pre}, DU executes algorithm $Decrypt(Q_{pre}, CT, PK, SK_{attr}) \rightarrow K_{aes}$ to obtain AES key K_{aes}. Then, DU downloads the ciphertext from IPFS, decrypts it with K_{aes}, and verifies its integrity.

3.3 Security Model

The security requirement for our blockchain-assisted key generation electric health records sharing scheme is based on Li et al.' scheme [25]. The difference is that we use blockchain to generate and manage all keys, and there are no curious KG-CSP and TypeII-Adversary in [25]. Replayable chosen ciphertext attack (RCCA) security was put forward in [26], which allows modifying the ciphertext, and cannot effectively change the implicit information. The adversary $\mathscr{A}$ can be described as a malicious DU, it will conspire with curious CSR to decrypt the data stored on the cloud servers. The $\mathscr{A}$ is permitted to obtain all users' outsourced private key SK_{out} and trapdoor Td_{kw}. The definition of the game between challenger and adversary $\mathscr{A}$ is as follows.

Setup: Challenger executes the System initialization method in Sect. 3.1 to get PK and MK. Then the challenger sends PK to $\mathscr{A}$ and saves MK as a secret.

Query Phase: Challenger first creates an empty collection C and the adversary $\mathscr{A}$ repeatedly initiate the following queries:

1. KeysGeneration. Upon receiving an access structure $\mathbb{A}$, challenger runs key-generation algorithm to get SK_{attr} and SK_{out}. Challenger stores key tuple (SK_{attr}, SK_{out}) with $\mathbb{A}$ in collection C and sends it to $\mathscr{A}$.
2. TdGeneration. After receiving an access structure $\mathbb{A}$, challenger performs $IndexKeyGen(PK, BF_\beta, \mathbb{A}) \rightarrow IK$ and $TrapDoor(PK, IK, kw, \beta) \rightarrow Td_{kw}$ to generate a trapdoor Td_{kw}. Challenger stores Td_{kw} in collection C and sends Td_{kw} to $\mathscr{A}$.
3. Decrypt. After receiving $\mathbb{A}$ and ciphertext tuple (CT, Q_{pre}), challenger checks whether the access structure exits in the tuple ($\mathbb{A}$, (SK_{attr}, SK_{out}), Td_{kw}) of collection C. If so, it executes algorithm $Decrypt(Q_{pre}, CT, PK, SK_{attr})$ and sends K_{aes} to $\mathscr{A}$. Else, it returns null value.

Challenge: The adversary $\mathscr{A}$ sends two plaintexts M_0,M_1 of equal length and a challenging attribute set w^* to challenger. It should be noted that w^* can't satisfy the access structure $\mathbb{A}$. The challenger chooses $\mu \in \{0, 1\}$, and runs $Encrypt(M_\mu, PK, w) \rightarrow CT$. Then challenger returns the challenge ciphertext CT to the adversary.

Restrictions:

Since w^* can't satisfy the access structure $\mathbb{A}$, the adversary $\mathscr{A}$ can't launch the KeysGeneration algorithm.

The adversary $\mathscr{A}$ can't launch TdGeneration query that the result equals to neither M_0 nor M_1.

Guess: The adversary $\mathscr{A}$ gives a guess $\mu' \in \{0, 1\}$ for μ.

Our scheme can meet RCCA-security. If the adversary's advantage in winning the game is negligible at best in any polynomial-time, such as $\left|P(\mu' = \mu) - \frac{1}{2}\right| < \varepsilon$.

4 System Scheme

Before introducing the detailed definition of the system, we firstly define the Lagrange coefficient $\Delta_{i,S}(x) = \sum_{i=0}^{n-1} S(i) \prod_{j \in S, i \neq j} \frac{x-j}{i-j}$ for $i, j \in S$, n is the length of S.

Our scheme is based on the Outsourced Attribute-based Encryption (OABE) [25] and the tree-based access structure. The user's private key contains a tree-based access structure $\mathbb{A}$, the attribute set w is embedded in ciphertext, and U is the set of all attributes, d is a pre-set threshold value. If $\gamma(w, \mathbb{A}) = 1$, S is an attribute set that satisfies $S \in \{w \cap \mathbb{A}\} \wedge |S| = d$. Based on the above structure, we use blockchain to take the place of authorized authority (AA) to generate and manage the user's private key, and add integrity verification phase in the scheme.

SysInit(1^λ): All blockchain nodes of BC run this algorithm, and λ is a secure parameter. BC chooses two multiplicative cyclic groups of prime order p: G_a with generator g, G_b, and a bilinear map $e : e(G_a, G_a) = G_b$. Each consensus node of BC cooperates to generate a parameter $x \in Z_p$ by the PSS protocol. BC calculates $g_1 = g^x$, randomly chooses g_2, $h \in Z_p$. BC chooses $\{u_i \in U\}_{0<i\leq n} \in Z_p$ to build the attribute universal U with length n. BC also selects secure hash functions:$H_a : \{0, 1\}^* \rightarrow Z_p$,$H_b : G_b \rightarrow \{0,$

$1\}^{\log p}$. Finally, BC sets $PK = \{G_a, G_b, g, g_1, g_2, h, \{u_i\}_{0<i\leq n}, H_a, H_b\}$, $MK = x$, and publishes PK as the global public key, keeps the MK as a secret.

KeyGen$(\mathbb{A}, PK, MK)$: This algorithm is run by blockchain nodes of BC. When receiving the access structure $\mathbb{A}$, BC selects a random value $x_a \in Z_p$ and calculates $x_b = x - x_a \bmod p$. Then BC randomly selects a $d-1$ degree of polynomial $f(x)$ and sets $f(0) = x_a$. For $i \in \mathbb{A}$, BC randomly chooses $\forall r_i \in Z_p$, calculates $d_{i_0} = g_2^{f(i)}(g_1 h_i)^{r_i}$, and $d_{i_1} = g^{r_i}$. Denote DU's outsourced private key $SK_{out} = \{d_{i_0}, d_{i_1}\}_{i\in\mathbb{A}}$. Next, BC computes $d_{\theta_0} = g_2^{x_b}(g_1 h)^{r_\varphi}$ and $d_{\theta_1} = g^{r_\varphi}$ where $r_\varphi \in Z_p$. Denote φ as the default attribute. Let DU's attributed private key $SK_{attr} = \{d_{\theta_0}, d_{\theta_1}\}$. Finally, BC returns SK_{attr}, SK_{out} to DU in a secret channel.

IndexKeyGen$(PK, BF_\beta, \mathbb{A})$: On receiving index-key generation request with DU's access structure $\mathbb{A}$ and a commitment $BF_\beta = g_2^{1/\beta}$ (DU randomly chooses $\beta \in Z_p$ and keeps it as secret). BC searches $(g_1 h)^{r_\varphi}$ associated with $\mathbb{A}$ and calculates $IK = g_2^{x/\beta}(g_1 h)^{r_\varphi}$. Then BC returns the IK to DU.

Encrypt(K_{aes}, PK, w): After treatment, patients get their case plaintext PT from the doctor. Patients use AES encryption PT with random $K_{aes} \in G_2$ to obtain AES ciphertext M', compute $H_a(PT) = h'$, and update M' to IPFS to get IPFS address p'. Patients choose an attribute set w associated with PT. Patients also choose $\eta \in Z_p$ randomly and compute $C_0 = K_{aes}e(g_1, g_2)^\eta$, $C_1 = g^\eta$, $C_i = (g_1 h_i)^\eta$ for $i \in w$, and $C_\varphi = (g_1 h)^\eta$. Denote ciphertext $CT = (\omega \cup \{\varphi\}, C_0, C_1, \{C_i\}_{i\in\omega}, C_\varphi)$.

IndexGen(PK, CT, KW): According to the ciphertext CT, patients select the appropriate keyword set KW. For each keyword $i \in KW$, patients compute $k_i = e(g_1, g_2)^\eta \cdot e(g, H_a(kw_i))^\eta \in G_2$ and let $K_i = H_b(k_i)$. Patients also let $K_1 = C_1 = g^\eta$, $K_2 = C_\theta = (g_1 h)^\eta$ and $Ix(KW) = (K_1, K_2, K_i)$. Patients send $(CT, Ix(KW))$ with h', p' to BC, then BC sends $(CT, Ix(KW))$ to CSR.

TrapDoor(PK, IK, kw, β): To generate trapdoor Td_{kw} with keyword kw. DU computes $T_q(kw) = H_a(kw)IK^\beta$, $D_1 = d_{\theta_1}^\mu$ and let $I = (I_{i_0} = d_{i_0}, I_{i_1} = d_{i_1})$, $Td_{kw} = (T_q(kw), I, D_1)$.

Search$(Td_{kw}, (CT, Ix(kw)))$: This algorithm is run by CSR, it runs Eq. (5) to get k_{kw}.

$$\begin{aligned} k_{kw} &= \frac{e(K_1, T_q(kw))}{e(D_1, K_2)} \\ &= \frac{e(g^\eta, H_a(kw)g_2^x(g_1 h)^{r_\varphi\mu})}{e(g^{r_\theta\mu}, (g_1 h)^\eta)} \\ &= \frac{e(g^x, g_2)\cdot e(g, H_a(kw))^\eta \cdot e(g^\eta, (g_1 h)^{r_\varphi\mu})}{e(g^{r_\varphi\mu}, (g_1 h)^\eta)} \\ &= e(g, g_2)^{\eta x}\cdot e(g, H_a(kw))^\eta \end{aligned} \tag{4}$$

Then computes $H_b(k_{kw})$ and compares it with each of K_i in index Ix. If equals, CSR then tries to run pre-decrypt algorithm. Else CSR returns null value.

Pre $-$ *decrypt*(CT, PK, SK_{out}): BC sends CT, PK, SK_{out} to CSR. If attributes in ciphertext match with $\mathbb{A}$ that corresponding to $SK_{out}(SK_{out}$ can be found in $Ix(KW))$, CSR performs Eq. (5) to gain the pre-decrypted ciphertext Q_{pre}, then CSR returns it to BC. Otherwise, CSR returns none. BC takes Q_{pre} with h', p', and returns them to DU.

Decrypt$(Q_{pre}, CT, PK, SK_{attr})$: After receiving Q_{pre} from CSR, DU first performs Eq. (6) to get K_{aes}. Then he/she downloads AES ciphertext M' with p', runs AES decryption, and gains plaintext PH'. DU compares $H_a(PH')$ with h'. If equals, indicating that PH' has not been tampered with, otherwise PH' will be discarded.

$$\begin{aligned}
Q_{pre} &= \frac{\prod_{i \in S} e(C_1, I_{i_0})^{\Delta_{i,S}(0)}}{\prod_{i \in S} e(I_{i_1}, C_i)^{\Delta_{i,S}(0)}} \\
&= \frac{\prod_{i \in S} e(g^{\eta}, g_2^{f(i)}(g_1 h_i)^{r_i})^{\Delta_{i,S}(0)}}{\prod_{i \in S} e(g^{r_i}, (g_1 h_i)^{\eta})^{\Delta_{i,S}(0)}} \\
&= \frac{\prod_{i \in S} e(g^{\eta}, g_2)^{f(i)\Delta_{i,S}(0)} \cdot \prod_{i \in S} e(g^{\eta}, (g_1 h_i)^{r_i})^{\Delta_{i,S}(0)}}{\prod_{i \in S} e(g^{r_i}, (g_1 h_i)^{\eta})^{\Delta_{i,S}(0)}} \\
&= e(g, g_2)^{\eta x_a}
\end{aligned} \tag{5}$$

$$\begin{aligned}
K_{aes} &= \frac{C_0 \cdot e(d_{\theta_1}, C_{\theta})}{Q_{pre} \cdot e(C_1, d_{\theta_0})} \\
&= \frac{K_{aes} e(g_1, g_2)^{\eta} \cdot e(g^{r_{\varphi}}, (g_1 h)^{\eta})}{e(g, g_2)^{s x_a} \cdot e(g^s, g_2^{x_b}(g_1 h)^{r_{\varphi}})} \\
&= \frac{K_{aes} e(g^x, g_2)^{\eta} \cdot e(g^{r_{\varphi}}, (g_1 h)^{\eta})}{e(g, g_2)^{\eta x_a} \cdot e(g^{\eta}, g_2^{x_b}) \cdot e(g^{\eta}, (g_1 h)^{r_{\varphi}})} \\
&= \frac{K_{aes} e(g^x, g_2)^{\eta} \cdot e(g^{r_{\varphi}}, (g_1 h)^{\eta})}{e(g, g_2)^{\eta(x_a + x_b)} \cdot e(g^{\eta}, (g_1 h)^{r_{\varphi}})}
\end{aligned} \tag{6}$$

5 Security and Performance Analysis

5.1 Security Proof

Compare with the scheme [25], our scheme utilizes distributed consortium blockchain rather than a centralized trusted authority to generate, manage master key and secret keys. We assume the consortium blockchain nodes always keep honest. Although there are some nodes that may betray, due to the PSS protocol, as long as no more than n nodes defected at the same time, the security of master key MK can still be guaranteed. The security of PSS protocol has been proved in [22]. We also use the integrity verification algorithm to check whether the data is tampered with.

Suppose that CSR is curious about security, with the help of available resources, CSR can recovery $e(g, g_2)^{\eta x_a}$. But it still can't restore $e(g_1, g_2)^{\eta}$ without getting x_b from SK_{attr}. Under certain circumstances, the curious CSR and malicious user may collude to decrypt others' data. However, due to the random division of MK, this collusion attack could be defended. Specifically, MK is randomly divided into different x_a and x_b for each user, and $x_a + x_b = x \bmod p$. x_a is used for generating SK_{out}, and x_b for SK_{attr}. If and only if SK_{out} matches with SK_{attr}, the ciphertext can be fully decrypted. Therefore, although CSR can get all users' SK_{out}, it cannot decrypt others' ciphertexts without corresponding SK_{attr}.

Theorem 1: Our scheme is RCCA security if DBDH problem can't be solved in any polynomial-time.

The proof of security is not our main content, for a more detailed proof to outsourced attribute-based encryption, refer to Li et al. in [25]. Next, we give the security analysis to demonstrate the security of our system.

5.2 Security Analysis

Confidentiality: Our scheme can meet the confidentiality requirements. Firstly, the patients' EHRs are encrypted by AES with a random AES key, and the ciphertext is uploaded to IPFS for sharing. AES is a popular symmetric encryption algorithm whose security depends on the security of the AES key. However, the AES key is encrypted by attribute encryption. DU can obtain the AES key only if the attribute requirements set by the patients are met, then DU download the ciphertext from IPFS and decrypt it. The AES key is random for every ciphertext, even though DU decrypts a ciphertext, they cannot decrypt other ciphertext of this patient.

Integrity: Comparing with [25], we have added an integrity verification step. Before encrypting EHRs, patients use the hash function like SHA-1 to calculate the message digest. After that, the message digest will be uploaded to the blockchain for recording. After DU downloads the decrypted data, the digest is calculated in the same way. Only if it is the same as the digest stored on the blockchain, which shows that the patients' data has not been tampered with.

Non-repudiation: In our scheme, blockchain is the central authority. All encryption, decryption, and access operations are recorded on blockchain. Because the data on the blockchain can't be tampered, it is always possible to determine who accessed what data and when by querying the blockchain.

5.3 Performance Analysis

Our scheme is based on the java pairing-based cryptography library (JPBC) of version 1.2.1. For the consortium blockchain, we use version 1.4.4 of Hyperledger Fabric to build. And version 1.4.4 of fabric-java-sdk is used to execute the contract. More specifically, our system is deployed on a 64-bit Ubuntu 20.04 system with 3.4 GHz i7-6700 CPU and 8G RAM. Consortium blockchain architecture is configured for 2 organizations with total of 4 nodes, which can represent 4 medical institutions in 2 sectors. In terms of security, the elliptic curve in our scheme uses the default parameters provided by JPBC, which are based on the curve $y^2 = x^3 + x$.

To simplify the experimental, we write the intermediate parameters of the PSS protocol on the blockchain. The master key and public key are generated later and can be queried directly from the chain. Figure 2 illustrates the user's private key generation performance of our system. Note that this also includes the fabric-java-sdk initialization time, which is about 300ms.

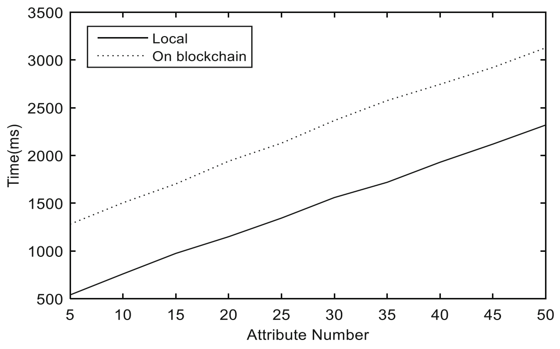

Fig. 2. Key generation time consumed.

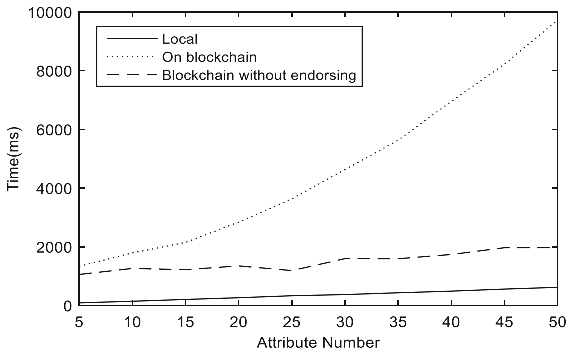

Fig. 3. Pre-decryption time consumed

In Fig. 3, the solid line indicates the time consumption only for pre-decryption at the local machine. Due to the constraints, note that we are only performing the pre-decryption computation with smart contract locally, rather than assigning it to CSR. The solid line can also represent the time consumed for pre-decryption on cloud servers. The dotted line here indicates that the smart contract performs the pre-decryption operation and all the nodes are involved in the endorsement. In contrast, the dashed line indicates that only one node executes the contract. We can conclude that endorsement will have some impact on the performance of key generation. However, there are few people who set 50 attributes on EHRs data. With attribute values limited to 15 or less, endorsements have a lesser impact on the system's performance.

After simulation experiments, it is proved that our system is feasible in practice.

6 Conclusion

We propose a novel blockchain-assisted key generation electric health records sharing scheme. Compared to scheme [25], we remove the Decryption CSP, and replace Key Generation CSP, Storage CSP with BC and IPFS. While remove the TypeII-Adversary type attackers, our scheme is more suitable for sharing data among multiple organizations. To do this, we first encrypt the data with symmetric encryption, and then encrypt the symmetric encryption key with ABE. We also consider the possibility of tampering with the data stored on the cloud servers, and add the integrity verification phase. Hence, our scheme is safer and more practical than scheme [25].

In short, our scheme can meet the demand for secure sharing of EHRs among multiple healthcare institutions. In addition, it is also easily scalable, just add nodes on the blockchain.

Funding Statement. This work was supported by the Open Fund of Advanced Cryptography and System Security Key Laboratory of Sichuan Province (Grant No. SKLACSS-202101), the Fundamental Research Funds for Beijing Municipal Commission of Education, Beijing Urban Governance Research Base of North China University of Technology, the Natural Science Foundation of Inner Mongolia (2021MS06006), Baotou Kundulun District Science and technology plan project (YF2020013), and Inner Mongolia discipline inspection and supervision big data laboratory open project fund (IMDBD2020020).

Conflicts of Interest. The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Häyrinen, K., Saranto, K., Nykänen, P.: Definition, structure, content, use and impacts of electronic health records: a review of the research literature. Int. J. Med. Inform. **77**(5), 291–304 (2008)
2. Wang, H., Song, Y.: Secure cloud-based EHR system using attribute-based cryptosystem and blockchain. J. Med. Syst. **42**(8), 1–9 (2018)
3. Uddin, M., et al.: Hyperledger fabric blockchain: Secure and efficient solution for electronic health records. Comput. Mater. Continua **68**(2), 2377–2397 (2021)
4. Peng, X., Zhang, J., Zhang, S., Wan, W., Chen, H., Xia, J.: A secure signcryption scheme for electronic health records sharing in blockchain. Comput. Syst. Sci. Eng. **37**(2), 265–281 (2021)
5. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27
6. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Decentralized Bus. Rev. 21260 (2008)
7. Chen, H., Xu, G., Chen, Y., Chen, X., Yang, Y., et al.: Cipherchain: a secure and efficient ciphertext blockchain via mpeck. J. Quant. Comput. **2**(1), 57 (2020)
8. Wang, S., Yuan, Y., Wang, X., Li, J., Qin, R., Wang, F.Y.: An overview of smart contract: architecture, applications, and future trends. In: 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 108–113. IEEE (2018)
9. Wang, S., Zhang, D., Zhang, Y.: Blockchain-based personal health records sharing scheme with data integrity verifiable. IEEE Access **7**, 102887–102901 (2019)
10. Huang, H., Sun, X., Xiao, F., Zhu, P., Wang, W.: Blockchain-based eHealth system for auditable EHRs manipulation in cloud environments. J. Parallel Distrib. Comput. **148**, 46–57 (2021)

11. Naresh, V.S., Reddi, S., Allavarpu, V.D.: Blockchain-based patient centric health care communication system. Int. J. Commun. Syst. **34**(7), e4749 (2021)
12. Zhang, J., Xu, G., Chen, X., Ahmad, H.: Towards privacy-preserving cloud storage: a blockchain approach. Comput. Mater. Continua **69**(3), 2903–2916 (2021)
13. Ali, M., Xu, C., Hussain, A.: Authorized attribute-based encryption multi-keywords search with policy updating. J. New Media **2**(1), 31–43 (2020)
14. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_30
15. Hua, J., Shi, G., Zhu, H., Wang, F., Liu, X., Li, H.: CAMPS: efficient and privacy-preserving medical primary diagnosis over outsourced cloud. Inf. Sci. **527**, 560–575 (2020)
16. Zhang, Y., Chen, X., Li, J., Wong, D.S., Li, H., You, I.: Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing. Inf. Sci. **379**, 42–61 (2017)
17. Kumar, R., Tripathi, R.: Towards design and implementation of security and privacy framework for internet of medical things (IoMT) by leveraging blockchain and IPFS technology. J. Supercomput. 1–40 (2021)
18. Gao, H., Ma, Z., Luo, S., Xu, Y., Wu, Z.: BSSPD: a blockchain-based security sharing scheme for personal data with fine-grained access control. Wirel. Commun. Mob. Comput. (2021)
19. Han, K., Li, Q., Deng, Z.: Security and efficiency data sharing scheme for cloud storage. Chaos Solitons Fract. **86**, 107–116 (2016)
20. Hwang, Y.W., Lee, I.Y.: A study on data sharing system using ACP-ABE-SE in a cloud environment. Int. J. Web Grid Serv. **17**(3), 201–220 (2021)
21. Beimel, A.: Secure schemes for secret sharing and key distribution (1996)
22. Pedersen, T.P.: A threshold cryptosystem without a trusted party. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 522–526. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-46416-6_47
23. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
24. Zhang, J., Zhang, Z., Chen, Y.: PRE: stronger security notions and efficient construction with non-interactive opening. Theoret. Comput. Sci. **542**, 1–16 (2014)
25. Li, J., Lin, X., Zhang, Y., Han, J.: KSF-OABE: outsourced attribute-based encryption with keyword search function for cloud storage. IEEE Trans. Serv. Comput. **10**(5), 715–725 (2016)
26. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 565–582. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_33