IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 9 May 2022, accepted 26 May 2022, date of publication 12 July 2022, date of current version 26 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3186682

Electronic Health Records Sharing Model Based on Blockchain With Checkable State PBFT Consensus Algorithm

ZHEN PANG[1], YUAN YAO[1], QIUYAN LI[1], XIAOQIN ZHANG[2], AND JING ZHANG[3]

[1]Xiyuan Hospital, China Academy of Chinese Medical Sciences, Haidian, Beijing 100010, China
[2]Chongqing Communication Design Institute Company Ltd., Network & Information Security Research Institute, Chongqing 401331, China
[3]Tianjin Electronic Information College, Jinnan, Tianjin 300072, China

Corresponding author: Qiuyan Li (pangz999@163.com)

ABSTRACT With the popularity of IoT devices and cloud technology in the medical industry. Sharing EHRs (Electronic Health Records) among medical institutions improves the accuracy of medical diagnosis and promotes the development of public medical. However, it is difficult to share EHRs among hospitals, and patients typically don't know about the usage of their health records. In this paper, we propose a patient-controlled EHRs sharing scheme based on cloud computing collaborating blockchain technology. The medical abstract and the access strategy are stored in the blockchain to avoid being tampered with. To achieve the fine-grained access control, we propose the attribute-based encryption scheme and multi-keyword encryption scheme to encrypt EHRs. Moreover, we proposed a node-state-checkable Practical Byzantine Fault Tolerance consensus algorithm (sc-PBFT) to prevent the Byzantine nodes from sneaking into the consortium blockchain. First, we check the state of the elected master node to avoid the master node having any malicious records. Then, using pre-prepared, prepare, and commit processes to complete the consensus request submitted by the client. At last, the proposed consensus algorithm evaluates the state of the master node according to the completion of the three-stage process to reduce the impact of the malicious node on the whole consortium blockchain. By doing this, the malicious node will be marked and isolated into the isolation area. The experimental results show that the proposed sc-PBFT algorithm has better handling capability and lower consensus latency. Compared with the PBFT algorithm in the case of Byzantine nodes, sc-PBFT not only improves the robustness of the consortium blockchain network but also improves the handling capability.

INDEX TERMS Data sharing, EHR, blockchain, Byzantine node, PBFT, consortium blockchain.

I. INTRODUCTION

The widespread use of medical equipment and sensors integrate with cloud computing and IoT technology provides high-efficiency medical services [1]. In the healthcare systems, the hardware devices such as smartphones, tablet PC, RFID, sensor, the implantable medical device are connected to the internet providing information interaction and service. The electronic health records (EHRs) record the patients' diagnostic and treatment information which contributes to the provision of convenient health record storage services [2]. Doctors can provide more efficient treatment plans by synthetically analyzing the previous diagnosis and the treatment effect for the patients. At the same time, sharing EHRs also allows excellent medical institutions from various regions to do comprehensively, accurately, and quickly measures to improve treatment efficiency and the public medical health level [3], [4]. Besides, the information island phenomena of medical data stored by different hospitals are obvious. Patients always have no acknowledgement of the usage of their EHRs controlled by hospitals and stored in the cloud server. Since the patient's personal information and medical records are stored in EHRs, it is crucially important to protect EHRs from attack during the medical data sharing

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Asif.

process [5]. To safely share medical data between different hospitals, the technology called Attribute-based Encryption (ABE) is proposed to encrypt the EHRs and upload the ciphertexts to the cloud [6]. To express more flexible access policies, the ABE technology evolves into the Ciphertext-Policy Attribute-Based Encryption (CP-ABE) and the Key-Policy Attribute-Based Encryption (KP-ABE) [7], [8]. The CP-ABE is more suitable for data access control. For CP-ABE, the ciphertext is connected to an access control structure in the encryption stage while the users' private key is connected with the attribute sets in the process of building a secret key. Although these methods can encrypt EHRs and provide fine-grained access control, cloud servers are usually semi-trusted and may execute the users' commands, and they may be curious about the users' privacy information. Once the cloud servers are under specific attacks or absence of supervision, the EHRs may be tampered with, altered, or destructed.

As the core technology of bitcoin, blockchain has the characteristics of decentralization, traceability, and programmability and it can protect the stored data from being tampered with and forged [9]. In specific scenarios, the blockchain can also hide the data and can be used to realize safe and trust EHR management [10]–[12]. Due to the EHRs usually contain large-scale, cross-media health data such as CT and other medical image data, and the storage and data sharing efficiency of blockchain are not high. There is an urgent need to combine cloud computing technology and blockchain technology to reinforce complementary advantages.

A. OUR WORK

In this paper, we propose a patient-controllable EHRs sharing scheme based on cloud computing collaborating blockchain technology. To prevent the cloud server from being tampered with and to audit the integrity of data, we store the medical abstract and the access strategy in the blockchain. Using CP-ABE to encrypt the EHRs, and only the user that satisfies the attribute access strategy can access the medical data, which achieves the fine-grained access control. For the encrypted EHRs, the data-sharing model supports multiplekeywords searching which improves search efficiency and accuracy. Secondly, we proposed a node-state-checkable Practical Byzantine Fault Tolerance consensus algorithm (sc-PBFT) to prevent the Byzantine nodes from sneaking into the consortium blockchain. In the proposed consensus algorithm, we check the node state of the elected master node to avoid the elected master node having any malicious records. Then, using pre-prepared, prepare, and commit processes to complete the consensus request submitted by the client. At last, using the proposed consensus algorithm to evaluate the state of the master node according to the completion of the three-stage process. By doing this, the malicious node will be marked and be isolated into the isolation area, which can reduce the impact of the malicious node on the whole consortium blockchain.

B. LITERATURE REVIEW

With the rapid development of IoT technology, more and more people attach importance to the security and privacy of personal information. As a new type of patient information carrier, EHR has become an indispensable tool for modern medical services [13]. It is considered to be an efficient method to improve the quality of medical services and accelerate biological medical discoveries. However, the data system of EHR involves the patients' privacy, and the security level of the current EHR storage methods is low and is prone to data leakage [14]. Moreover, the existing EHR data cannot be delivered between the databases of different institutions, and the information isolated island phenomenon will hinder the precision medicine development. With the rise of cloud computing, the storage methods of EHRs have developed significantly. Zhang *et al.* firstly proposed the cloud-based security requirements of the EHR system and put forward some suggestions to ensure the security of EHRs in the cloud environment [15]. Yang *et al.* used the attributes of encryption technology to protect medical data in the cloud and proposed a new searchable encryption scheme [16]. Xhafa *et al.* proposed an attribute-based cloud computing technology to store the EHR data with privacy consciousness [17]. However, these cloud-based security solutions still have shortcomings and they heavily rely on cloud service providers. The privacy leak of EHRs probably occurs once the cloud providers are under some targeted attacks. In addition, driven by interests, most network devices directly allow access on the public internet, as a result, hackers can steal sensitive medical data through technical methods and conduct illegal transactions to make huge profits [18]. To ensure the security of the sensitive medical data in the cloud environment, Alshehri *et al.* proposed a CP-ABE-based fine-grained flexible access control strategy to encrypt the EHRs and only the users with corresponding attributes can decrypt the ciphertext [19]. Yang *et al.* proposed a healthy data sharing scheme that supports keywords retrieval and timing-start proxy re-encryption. The data users can search the EHRs within the required time and the data owners can control the search and decryption time [20]. Huang *et al.* proposed a kind of recommendation protocol based on Euclid Similarity distance to reduce the patients' privacy leakage [21]. However, the computing ability of the proposed algorithm is low with a huge cost of storage. To recommend suitable conditional attributes to replace the proxy re-encryption scheme and proposed a fine-grained privacy protection EHR sharing scheme based on cloud computing. In actual application scenarios, the attributes of data users are not unchanged, Yuan *et al.* used attribute encryption to encrypt EHRs and uploaded them to cloud servers to facilitate the sharing of data [22]. At the same time, to ensure the security of the ciphertexts, the authors used version number marking and proxy re-encryption to achieve the revocation of the current user attribute. Rao proposed a secure attribute-based signature and encrypted EHR sharing scheme, compared with the previous methods this method supports shorter

ciphertexts and requires fewer double linear pair calculations [23]. In 2008, Satoshi firstly proposed the blockchain, which can be regarded as a distributed database that satisfies the decentralization, tamper resistance, and asymmetric encryption. In addition, smart contracts in the blockchain can also enhance the interoperability of medical data and have strong application potential in the medical field [24]. Xue *et al.* designed a blockchain-based medical data-sharing model, by providing the sensory mechanism this method can solve the problems of data inspection, storage, and synchronization between medical institutions, but this method has some shortcomings in data storage [25]. Dagher *et al.* proposed a framework using the Ethereum platform to transfer the ownership and control of EHRs to data owners. The authors also developed a variety of smart contracts and used proxy re-encryption technology to further protect the privacy of EHRs [26]. Guo *et al.* proposed an attribute-based multi-authority signature scheme. This method only considered privacy protection for the access control mechanism rather than the data itself. From the perspective of data itself, protecting data privacy is simpler and more effective than access control mechanisms [27]. Rahman *et al.* combined distributed key generation technology with the proxy re-encryption technology, designing data sharing security protocol, but this method doesn't support one-to-many data sharing [28]. Analyzing the above existing EHR sharing schemes, although these above-mentioned schemes achieved a certain degree of success in this field of EHR sharing, there are still some areas that can be improved. For example, only using cloud servers for data storage may cause data tampering or leakage, etc. Using the blockchain alone can't avoid the problem of low efficiency when storing large-scale medical data. Therefore, it is necessary to integrate blockchain with cloud storage technology together to realize EHR data sharing. At the same time, how to store data safely and how does the patient controls the data access, and how data users perform ciphertext search efficiently, and how to improve the efficiency of data sharing, are worthy of further investigation. The mainstream consensus algorithm of blockchain including Proof of Work (POW), Proof of Stack (PoS), Delegated Proof of Stake (DPOS), Raft Consensus, and Practical Byzantine Fault Tolerance (PBFT) Algorithm. Different consensus algorithms have different characteristics, and they can be roughly divided into two categories, one is the BFT-based consensus algorithm [29], which can tolerate a certain percentage of Byzantine nodes but it will take a long time to achieve consensus. For example, the PoW algorithm is used as the consensus algorithm for bitcoin, which requires blockchain nodes to calculate randomness hash values to prove their work, and the nodes can achieve consensus only the result achieves at least 51% of the whole network nodes' agreement [30]. Reference [31] proposed a scalable Byzantine Fault Tolerance (SBFT), which can tolerate the number of Byzantine nodes as less than a third of the total number of nodes. This mechanism can reduce the number of blocks on the blockchain communication overhead between the points, so the number of the nodes can be further expanded. Crash Fault Tolerance (CFT) algorithm can tolerate the failures in the consensus, although it can help the network reach consensus quickly, it needs to run in the network with an authentication mechanism [32]. Reference [33] proposed a partially decentralized consensus algorithm with a high communication speed, but it needs to elect a reliable master node. According to the above analyze, it is necessary to propose a consensus algorithm taking the consensus efficiency and consensus security into account.

C. THE ORGANIZATION OF THIS ARTICLE

The structure of this paper is organized as follows. In Section 1, we review the related works about EHR sharing and consensus algorithms. Then, in Section 2, we present the details of our proposed EHR sharing scheme and related algorithms. Section 3 offers the details of the improved PBFT consensus algorithm. In Section 4, we present the comparative analyses of the proposed EHR sharing model and the consensus algorithm. Finally, we make some conclusions and the future work in Section 5.

II. THE PROPOSED EHR SHARING MODEL

In this section, we elaborate on the design goal of the proposed EHR data-sharing model and focus on explaining the logical framework and running process of the model. It should be noted that in this section we focus on the overall architecture of the proposed model, and give the details on the cloud computing collaborated blockchain-based EHR sharing scheme.

A. THE DESIGN GOAL OF THE PROPOSED EHR SHARING MODEL

The design goal of our proposed model can be summarized as follows.

Patient-controllable data sharing. In practice, patients can't fully grasp the usage of their EHRs, this paper focuses on the design of a patient-controllable EHR sharing model in which the patient can decide the access rights of his/her EHR.

Protect the privacy of EHR. Due to EHRs store lots of private information, it is important to guarantee the integrity of EHR. This paper combines cryptography and blockchain technology to protect patients' EHRs by distributing attributes and encrypting. By doing so, we can protect the integrity and auditability of EHRs during the data sharing process.

Fine-grained and efficient access control. By using the improved CP-ABE scheme, we achieve the goal of data fine-grained and efficient access control.

Safe search for keywords. This paper constructs a safely searchable encryption scheme by which data users can take advantage of the multiple keyword algorithm to generate the tokens of the keywords. Only the user's identity satisfies the preset access strategy, the user has access to the historical EHR. If the identity of the sniffer doesn't satisfy the preset access strategy, the sniffer can't get the EHR even if he/she acknowledges the keyword.

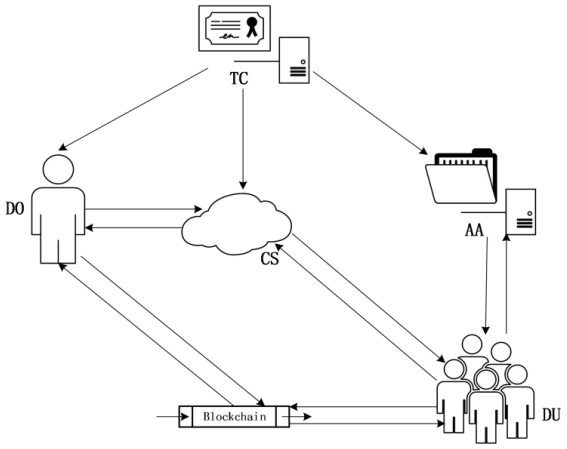

FIGURE 1. The logic framework of the EHR sharing model.

B. THE LOGIC STRUCTURE AND RUNNING PROCESS OF THE PROPOSED MODEL

In this paper, we combine blockchain technology with cloud servers for storing data in both on-chain and off-chain (cloud servers). At the same time, we use the CP-ABE scheme and multiple keywords safe searchable encryption scheme to achieve fine-grained and efficient access control. Figure 1 shows the relations among the main entities of the EHR sharing system. The main entities of the EHR sharing model are shown below.

Trusted Center(TC): ***TC*** is completely trusted, which sets the master public key and secret key of the system, and generates the public and private key pair for DO and CS.

Attribute Agency (AA): ***AA*** can be hospitals, insurance companies, or other medical research institutions. It verifies the identity of DU and generates a private key for a legal DU.

Consortium Blockchain (BC): ***BC*** stores the abstract of the encrypted data and verifies the DU's request.

Cloud Server (CS): ***CS*** is a cloud service provider which stores the encryption data and the access strategy.

Data Owner (DO): ***DO*** is the patient who encrypts the EHR, stores data, and generates the data access strategy.

Data user (DU): ***DU*** is the doctor, or the insurance company manager or other users who want to view patients' EHRs.

The running process of this model is shown as follows:

Step 1: Global Setting. It is executed by ***TC***, including setting global public parameters, public attribute keys, and system master public key and master secret key.

Step 2: Secret key generating. ***TC*** verifies the identities of ***DO*** and ***CS*** and set public and private key pair for them. ***DU*** sends a request to ***AA*** when it wants to join the system. ***AA*** assigns identity *uid* to the legal ***DU*** and attributes *A aid*, *uid*.

Step 3: Data encryption. ***DO*** selects the keywords for the data and set the keyword index. Randomly Select the secret key $k\theta$ to encrypt the data and encrypt $k\theta$.

Step 4: Data storage. As shown in Figure 2, to avoid the data stored in ***CS*** being tampered with, the data is stored in both on-chain and off-chain (cloud server).

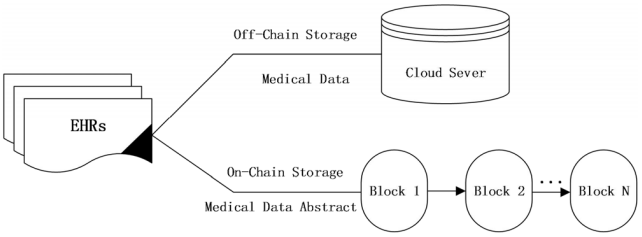

FIGURE 2. The structure of on-chain and off-chain data storage.

Medical data and medical data abstract are clarified in Figure 3. ***DO*** encrypts the original medical data and uploads it to ***CS***, and store the medical data abstract in the blockchain. Merkle tree structure [34] can be used to verify the integrity and authenticity of the data. ***DO*** encrypts EHRs and gets the ciphertext *ct*, then signs *ct* and gets the signature Sig_{DO}. At the same time, ***DO*** calculate the hash of the cipheryext, and send cipheryext, Sig_{DO}, and hash value to ***CS*** to verify the signature of ***DO*** and the hash of *ct*, if the hash of *ct* is the same as the sent hash, it proved that ***CS*** received the right cipheryext and return the data location to ***DO***, otherwise return $\perp$.

Step 5: Data access. ***DU*** uses the searchable encryption algorithm to calculate the search token base on the public key and keyword set. ***DU*** sends a search request to the master node. After receiving the search request, the node will run the matching algorithm to verify whether the keyword index matches the data token. If it is successfully matched, it is shown that the data file isn't deleted and the blockchain will return the corresponding data abstract to ***DU***, otherwise return $\perp$. When executing this algorithm, ***DU*** firstly calculates the content secret key $k\theta$ and decrypt the data location, then send a request to ***CS***. If the attribute set of ***DU*** satisfies the access strategy defined by ***DO***, ***CS*** will output the corresponding cipheryext, otherwise return $\perp$.

Step 6: Decryption. ***DU*** firstly calculates the hash value of the cipheryext and compare it with the hash stored in the blockchain. If the two hash values are the same, ***DU*** will decrypt the secret key $k\theta$,and the cipheryext.

C. MODEL CONSTRUCTION AND ALGORITHM DESIGN

Although CP-ABE scheme can realize fine-grained data access control and encryption, the cloud servers are usually semi-trusted, and they will execute the user's commands. In the absence of oversight or under specific attacks, the data stored in cloud servers may be tampered with, lost or disclosed. To overcome the shortcoming of CP-ABE, we collaborate cloud and blockchain architecture to improve CP-ABE scheme. The details of model construction and the related algorithms can be described as below.

Firstly, ***TC*** performs the global setting to generate the public key and the main secret key. The input of this algorithm is secure parameters and attribute set. The output of this algorithm is a public parameter *GP*, the secret key of the public attribute PAK_x, the public key *PK* and the main secret key *MK*. ***TC*** selects two bilinear group *G* and G_T with a

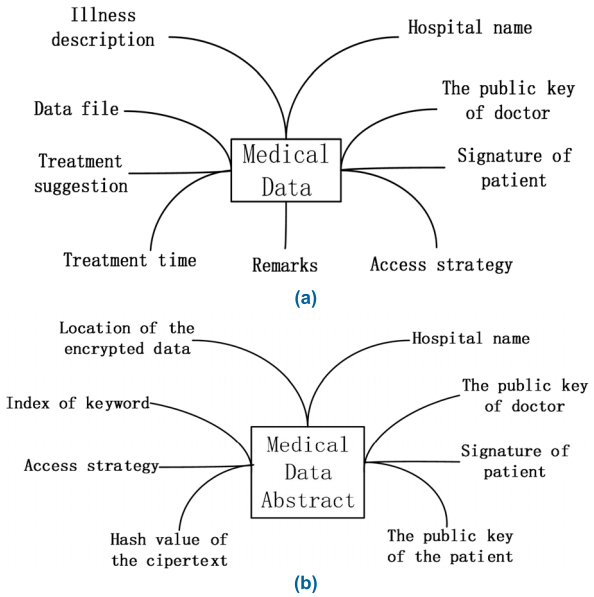

FIGURE 3. (a) The components of medical data. (b) The components of medical data abstract.

prime number p order, g and g_0 are the generators of G. Define e: $G \times G \rightarrow G_T$ as a bilinear mapping and two hash functions as: $H : \{0,1\}^* \rightarrow G$, $H_1 : \{0,1\}^* \rightarrow Z_p^*$, where Z_p^* is the multiplicative group of residual rings. ***TC*** verifies the identity of ***AA*** and assigns the legal global identity *aid* and defines the number of ***AA*** as l. ***TC*** outputs the $GP = \{G, G_T, H, H_1, e, p, g, g_0, k\}$. Define the attribute set as A, randomly select a element $v_x \in Z_p$ and calculate the public attribute secret key $PAK_x = g^{v_x}$ for each attribute element x. ***TC*** selects elements $a, \alpha, \beta \in Z_p^*$ to calculate the public key and the main secret key as below.

$$PK = \{GP, g^a, g^\alpha, e(g,g)^\beta, PAK_x\} \tag{1}$$

$$MK = \{\alpha, g^\beta, \{v_x | x \in A\}\} \tag{2}$$

Secondly, ***TC*** generates secret keys for ***DO*** and ***CS***. The input of this algorithm is a random number, and the output is the public and secret key pair of ***DO*** and ***CS***. ***TC*** selects two big prime numbers p and q, and calculates $n = p.q$. Define the Euler function of n as $\varphi(n)$. ***TC*** select a random number T, where T and $\varphi(n)$ are co-prime and $T \in [1, \varphi(n)]$. Calculate the congruence as $S.T \equiv 1 \bmod (\varphi(n))$, and the secret key of DO as $SK_{DO} = S$ and the public key as $PK_{DO} = \{T, n\}$. ***TC*** generates the public and secret key pair for ***CS*** according to the above steps, the secret key and the public key of CS are calculated as $SK_{CS} = S'$ and $PK_{CS} = \{T', n'\}$.

Thirdly, generate the secret key of ***DU***. The input and output of this algorithm are random numbers, and the attribute set of ***DU*** and the secret key of ***DU***. ***DU*** submits the application to ***AA***, and ***AA*** verifies the legality of the identity of ***DU*** and assigns the identity *uid* and the attribute set $A_{aid,uid}$, then generates the secret key. ***AA*** selects $t \in Z_p^*$ and calculates $D = g^\beta.g^{at}, D' = g', D'' = g^{a\alpha}$. For $\forall x \in A_{aid,uid}$, calculate $D_x = h(x)^{\frac{1}{vx}}$. The secret key of ***DU*** is set as $SK_{DU} = (D, D', D'', D_x)$.

Fourthly, encrypting and store data. Before encrypt data, ***DO*** should perform the encryption algorithm and generate the index. The input and output of this algorithm are data set F and keyword index I_W. Assuming the shared data set of ***DO*** is $F = (F_1, F_2, \ldots, F_d)$, where d is the number of F. ***DO*** calculates each data F_θ, $1 \leqslant \theta \leqslant d$, and selects a random number τ_θ to calculate $I_\theta = g^{\tau_\theta}$. Select the keyword set $W = \{w_1, w_2, \ldots w_m\}$ where m is the number of keywords in W. Set the index for each keyword w_j as $\{I_{w_j}\} = \{\bar{I}, \{I_{\theta,j}, I_\theta\}\}$, and the index for the keyword set as$I_W = \{I_{w_j}\}, 1 \leqslant j \leqslant m$. In the data encryption algorithm, the input and output are the data file F, the content secret key k_θ, the public key PK, the access strategy (M, ρ) and the cipheryext of data file, cipheryext of the content secret key.***DO*** randomly selects the secret key k_θ from the secret key space. Use the symmetric encryption algorithm to get the cipheryext $C_\theta = Enc(F_\theta)$ of F_θ. Input PK,k_θ and access strategy (M, ρ) to get the cipheryext of k_θ.

$$C' = k_\theta.e(g.g)^{\beta s} \tag{3}$$

For (M, ρ), $M \in \mathrm{R}^{l \times n}$, ρ maps each row of M to a different attribute. DO selects s as a secret key and selects a random number $y_1, y_2, \ldots, y_n \in Z_p$, and defines $v = (s, y_2, \ldots, y_n)$. Calculate $\lambda_i = M_i.v$, $1 \leqslant i \leqslant l$, where M_i is the *ith* row of M, and randomly select $r_i \in Z_p$ to calculate $C'' = g^s$, $C_i = (g^{v\rho(i)})^{r_i}$, $C' = g^{a\lambda_i}.H(\rho(i))^{r_i}$ and $CT' = \{(M, \rho), C', C'', C_i, C_i'\}$. Then the cipheryext of ct can be calculated as below.

$$CT = (I_W, C_\theta, CT') \tag{4}$$

After encrypting, the cipheryext will be uploaded to ***CS***. The input and output of the cipheryext uploading algorithm are the cipheryext ct and the hash value of ct, the signature of ***DO***. In this algorithm, ***DO*** calculates the hash and signature of ct as $H(CT)$ and $Sig_{DO}(CT) = H(CT)^S \bmod n$, then ***DO*** sends CT, $H(CT)$ and Sig_{DO} to ***CS***. ***DO*** will do the authentication based on the equation $Sig_{DO}(CT) = H(CT)$ and compare the calculated has with the sent hash, if the two hash is the same,***CS*** will return 1 and the data location to ***DO***, otherwise, return $\perp$. It is the blockchain transaction process after uploading the cipheryext to ***CS***. The input and output of blockchain transaction algorithms are the data abstract block and its signature, and the transaction sheet. ***DO*** encrypts the data location using the content secret key k_θ to get the cipheryext $CT_{Location}$. ***DO*** stores $H(CT)$, I_W, (M, ρ),$CT_{Location}$, PK_{DO}, hospital name, doctor identities information into the abstract block, and calculates the hash as $H(trans)$. ***DO*** records timestamp, transaction ID, and signs the transaction as $Sig_{DO}(trans) = H(trans)^S \bmod n$. ***DO*** submits the transaction, signature and verification request to the blockchain node. After receiving the transaction sheet, the blockchain node receives the transaction sheet and extracts the public key and verifies the signature of the transaction sheet. If $Sig_{DO}(trans)^S = H(trans)$, comparing the hash value of the transaction sheet, if the two hash is the same, the data

is normal and performs the consensus operation, otherwise, the date can't be store in the block.

Fifth, the data access. When ***DU*** wants to access data, he/she should generate the search token, and then send the search request to the blockchain node. The blockchain node will perform the matching algorithm to generate the keyword set. The input and output of the matching algorithm are the search token and keyword set, respectively. To generate the search token, for the keyword set $W' = (w'_1, w'_2, \ldots, w'_{m'})$, where m' is the number of keywords. ***DU*** selects $\pi \in Z_p^*$ to calculate $t_1 = g^{a\pi} . \prod_{j=1}^{m'} g^{\pi H_1(w'_j)}$ and $t_2 = g^{a\pi}, t_3 = D''^{\pi}$. ***DU*** generates the search token as $T_{W'} = (t_1, t_2, t_3)$ and sends it to the blockchain node. After receiving the search request, the blockchain node verifies whether there exists an index matching a keyword index of $T_{W'}$, that is to say whether $I_{w_j} \in I_W$. As long as it search successfully, the blockchain will return the transaction sheet, otherwise, return $\perp$.

Sixth, decryption. ***DU*** receives the transaction sheet from the blockchain. If the attribute of ***DU*** isn't in the revocation list ALR_x and the attribute satisfies the access strategy embedded in the access strategy, ***DU*** decrypts $CT_{Location}$ based on k_θ and download the data from ***CS***. After receiving the cipheryext, ***DU*** calculates the hash and check the hash and check that the hash is consistent with the hash stored in the blockchain, if consistent, do the decryption. According to the linear secret sharing scheme, there is a constant $\{\omega_i \in Z_p\}_{i\in I}$ satisfies $\sum_{i\in I} \omega_i \lambda_i = s$. The input and output of the decryption algorithm are CT, SK_{DU}, $CT_{Location}$ and data file F_θ. DU firstly calculate $A = \frac{A_2}{A_1} = e(g,g)^{ats}$, where $A_1 = \prod_{i\in I} e(C_i, D_{\rho(i)})^{\omega_i}, A_2 = \prod_{i\in I} e(C'_i, D')^{\omega_i}$, then calculates the secret key k_θ of content $k_\theta = \frac{C'.A}{e(D,C'')}$. So far, ***DU*** gets the data file stored location by decrypting $CT_{Location}$, and searching the data file to get the cipheryext CT. Then, ***DU*** calculates the hash and compares it with that stored in the blockchain, if the two hash is the same, do the decryption.

III. THE IMPROVED PBFT CONSENSUS ALGORITHM

A. PRACTICAL BYZANTINE FAULT TOLERANCE ALGORITHM

Byzantine Generals Problem is proposed to be a fictitious model to explain the consistency problem. In the consortium blockchain system the Leader node is selected from the information center. Using the membership review of the consortium chain, the system can ensure the majority of the newly added nodes are reliable. However, it is not sure that there is no Byzantine node in these newly added nodes. Once the Byzantine node is elected as the master node, it may discard or change the privacy data. Practical Byzantine Fault Tolerance (PBFT) is a practical method that can tolerate the Byzantine nodes with the number is not more than one-third of a total number of the nodes. PBFT reduces the complexity of BFT from exponential level to polynomial level. In the PBFT consensus model, the nodes can be classified into the master node and backup node. The master node is elected by turns and the number of it can be calculated as follows.

$$s = v \bmod |R| \tag{5}$$

If the master node is reliable, it will collaborate with the backup nodes to perform the pre-prepare, prepare and commit processes. The backup nodes will perform the view change protocol to reelect the new master node when the master node performs a round of consensus or the Byzantine nodes emerge. Whereas the Raft consensus can only tolerate node failure, PBFT can only tolerate the Byzantine nodes with the number is not more than one-third of a total number of the nodes, but the master node is polling selected and the algorithm lacks the judgment of the node state. Once the Byzantine node is selected as the master node, it may result in the client proposal fails to reach consensus until other backup nodes perform the view change protocol [35]. The computational of this process is very high. Secondly, although PBFT can accommodate some Byzantine nodes without impacting other nodes to be consensus, it won't deal with these malicious nodes or punish them. These Byzantine nodes will take malicious measures without paying any price. If these Byzantine nodes are left untreated and are polled as the master nodes, this will reduce the consensus efficiency. According to the above analysis, this paper proposes a node state-checkable PBFT (sc-PBFT) algorithm which will be illustrated in the following subsections.

B. THE NODE STATE DETECTION AND PUNISHMENT

Several improved PBFT algorithms have been proposed, reference [36] proposed an alternated set voting based Practical Byzantine Fault algorithm which can optimize the consensus process and reduces the communication times among the nodes, but the Byzantine nodes are still brought into the backup node-set and are taken as the campaigners in the next round election. The scaled Byzantine nodes will result in the master nodes switching frequently. The authors in [37] brought verifiable random function (VRF) before the three PBFT consensus stages to ensure the randomness of the election of the master node, but VRF will result in the system performance reduced with the scale of the nodes increasing. In this paper, we propose a node state checkable PBFT algorithm that can detect the node state to avoid the Byzantine node serving as the master node and to reduce the invalid communications. Moreover, the proposed consensus algorithm can put the malicious nodes into the isolated area and trace their source and eliminate it.

Different from the nodes anonymously joining into the public blockchain, there is a strict access mechanism for the newly added nodes in the consortium blockchain. From the respects to credibility, security, and performance, the participated consensus nodes in the consortium blockchain are supplied by the information center. In this paper, we take the state attributes of the newly added consensus nodes and assign a state information sheet for each participated

TABLE 1. Node state information sheet.

Node Number	Node State
Node 0	normal
Node 1	unstable
... ...	... ...
Node *n*	malicious

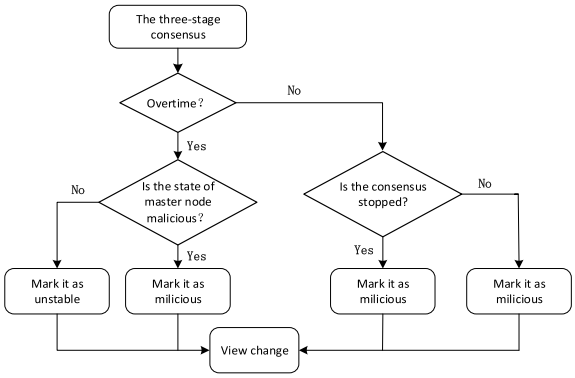

FIGURE 4. The process of node state marking.

consensus node. The node state information sheet is shown Table 1.

There may be many user nodes in the consortium blockchain. While the nodes participating in consensus are from the information centers or enterprises and the number of that is small. Due to the high cost of communication and performance, the number of the nodes that participating in the consensus is not more than 100, and maintain a node state information sheet for each node will not create an additional computational burden. The first time that the system runs, the state information sheet of every consensus node won't record the state of any nodes. Before performing the three-stage consensus processes, the system will check the state of the master node in the node state information sheet, if the master node is not in the sheet, the system will verify the signature of this node, if the signature is right, this node will be added into the node state information sheet. After performing the consensus process, completing or stopping a round of consensus, the participated backup nodes will mark the master node of the current view. The node marking process is shown Figure 4.

If a round of consensus following a three-stage process, the backup nodes will mark the master node as ''normal''. During the consensus process, if the master node doesn't respond until the consensus process over time and doesn't complete. Since the system can't judge whether the master node accidentally disconnected or doesn't send the consensus proposal deliberately, other backup nodes will mark the master node as ''unstable'' and perform the view-change protocol to elect a new master node to complete the consensus process. If the node with the state ''unstable'' is elected as the master node for two consecutive times and the consensus processes are over time, then the node will be marked as ''malicious''. However, if the ''unstable'' node completes the consensus before the next election of the master node, the ''unstable'' node can still recover to be a normal node. In one round of consensus, if the backup node finds malicious measures such as forging the signature, tampering with the proposal information, the view-change protocol should be launched immediately, and if the view-change is passed by the whole backup nodes, this master node will be marked as ''malicious''.

Assuming that n nodes participating in the consensus process, after n rounds of consensuses all the nodes have been elected as the master nodes and every node participating in the consensus process maintains a complete state information sheet. If the nodes in the consortium blockchain are reliable, then most of the nodes should be normal and few nodes are unstable, this demonstrates that the network status of the consortium is healthy. In reality, there may be some Byzantine nodes in the consortium blockchain which has a strict identity access mechanism, and the number of these Byzantine nodes is difficult to exceed one-third of the number of the consensus participating nodes. Therefore, these Byzantine nodes are presented as the backup nodes and the generated malicious results will be discarded during the consensus process. If these Byzantine nodes are presented as the master nodes in the consensus system and can't tamper users' data, they may bring a large performance or timeout impacts. Therefore, after the Byzantine node is elected as the master node once, it will be marked as ''malicious'' in other reliable nodes' state information sheet. When the Byzantine node is elected as the master node again, and we can eliminate it by the corresponding state ''malicious''. By doing this, we can avoid the unnecessary consensus error-correcting time. The paper also establishes an isolation area for the ''unstable'' and ''malicious'' nodes. If the node used to be marked as ''unstable'', then the isolation area will record the number of ''unstable'' states, if a node has a large number of ''unstable'' states, it will be treated as an aged node and eliminated. If a node has a ''malicious'' mark, it will be put into the isolation area immediately. For m rounds of consensus, a system consensus will be launched and the system administrator can trace or deal with the Byzantine nodes according to the node information recorded in the isolation area.

C. THE CONSENSUS PROCESS OF sc-PBFT

In the proposed sc-PBFT algorithm, the adversary mode can be abstracted as $n = 3f + 1$, where n is the whole number of the nodes that participating in the consensus, f is the number of Byzantine nodes and $f \leqslant \left\lfloor \frac{n-1}{3} \right\rfloor$. Readers can refer to Section IV-A for the above equation relation. Since the traditional PBFT algorithm is based on the state machine principle replication, the nodes in the consensus network can be called replica nodes [29]. The replica nodes can be divided as primary master nodes and backup nodes. The master node can be got based on the replica node ID in a polling pattern. The master node and the backup node run in the view, and each master node corresponds to a view. If the master node

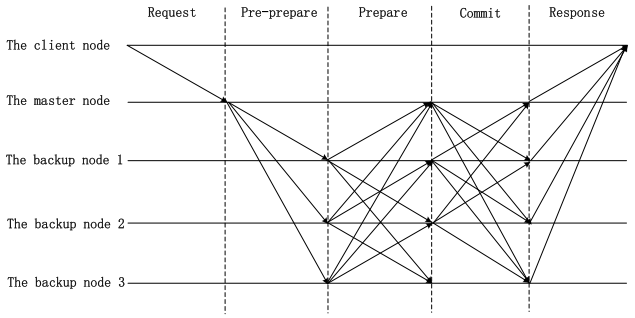

FIGURE 5. The process of three-stage consensus.

is changed we should perform the view-change protocol. Figure 5 illustrates the consensus process of the proposed sc-PBFT.

Although the sc-PBFT algorithm continues to use the three-stage protocol in the consensus process, in the three-stage process, it adds the processes of checking the master node state, marking the master node state, and isolating the Byzantine nodes. Any time-out or malicious measures of the master node will be marked by the backup nodes. As shown in Figure 5, a successful consensus process includes five steps which are shown below.

Step 1. Request: When the patient DO wants to upload the EHR or the data user DU wants to access transaction, they will send request $< Request, m, t, c >$ to the master node *primary* by the client, where the m is the content of the proposal, c is the identity of the client, t is the timestamp of the proposal. In this step, the user will submit the consensus proposal information to the consortium blockchain. This information will be signed by the client and then participate in the core process of consensus.

Step 2. Pre-prepare: The master node will verify the identity of the sender, number m, and generate the corresponding abstract. Pack and send them to the backup node. Define the content of the information as $\ll Pre-Prepare, v, n, d >, m, p >$, where v records the number of the view running the consensus, n is the number of m set by the master node, d is the hash abstract of the proposal and p is the node number of the master node.

Step 3.*Prepare*: When all the backup nodes receive the information sent by the master node, we can select one backup node b_1 to illustrate. b_1 check the state of the master node with the node number p in the state information sheet. If the master node state is malicious, then the backup node b_1 will launch a view-change protocol request to update the master node, if the master node state is normal or unstable, then the backup node will broadcast the information $< Prepare, v, n, d, i >$ to other nodes except b_1, where i is the number of b_1.

Step 4. Commit: Other nodes receive the prepared information from b_1 in Step 3. Similarly, b_1 will receive the prepare information from other backup nodes. b_1 will firstly check the rightness of the sent information, and whether the view number of other nodes' Prepare information is the same as the view number of the master node's Pre-prepare. If there are at last $2f$ Prepare messages from other backup nodes are in accordance with the Pre-prepare message of the master node, the backup nodes of b_1 will broadcast the message $< Commit, v, n, D(m), i >$ to other nodes, where $D(m)$ is the verifiable signature of proposal m.

Similarly, if the Prepare message is broadcasted by the backup node b_1 is normal, it will also receive the Commit messages returned by other nodes. When the number of the Commit message is at last $2f + 1$, it means that the whole network achieves the consensus. However, before the consensus, if the master node finds any malicious measures, it will result in the view-change protocol launched by the backup nodes and mark the state of the master node as malicious, and then put this master node into the isolation area waiting to be processed.

Step 5. Reply: This step means that the consensus process is over, the following is that all the nodes will return the message $< Reply, v, t, c, i, r >$ to the client, where r represents the execution result of the node. The network can only tolerate f Byzantine nodes, if the client receives $2f + 1$ Reply messages it means that the returned result has been achieved the consensus authorization.

IV. MODEL AND PERFORMANCE ANALYSES

In this section, we will analyze the validity and security of the proposed EHR sharing model. By comparing with other existing EHR sharing models, the advantages of the proposed EHR sharing model will be illustrated. Moreover, the performance of the proposed consensus algorithm will also be evaluated.

A. THE FAULT-TOLERANT ABILITY OF sc-PBFT

Although the consortium blockchain has a strict access mechanism, it can't prevent the adversary from disguising the Byzantine node to be a normal node and sneaking it into the consortium blockchain. Assuming that f Byzantine nodes tolerating the malicious measures and make sure that these Byzantine nodes won't make any impact on the consensus result. We need to set the suitable number of nodes n to make the f malicious nodes not be the majority. Before determining n, we would like to clarify that the malicious measures, one is the consensus participating node which doesn't return the executed result after receiving the consensus request. The other is the consensus participating node returns a wrong consensus result to deceive other nodes.

Under this premise, when a client makes a consensus request, it should receive n messages if it is a normal node. Due to the existence of the Byzantine node, the client receives $n - f$ response messages. To ensure the response message becomes the majority, it should hold $n-f > f$, that is to say the whole number of nodes n should be larger than $2f$. On the other hand, even if there are f nodes which don't response to the client, and there may be other reliable nodes that lose their connections, that is to say, there may be an error-response Byzantine node in the $n - f$ response messages. To ensure the system security we need to consider the situation that the

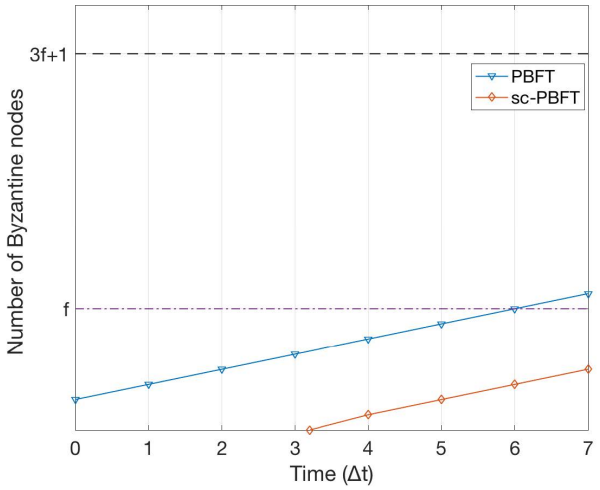

FIGURE 6. The fault-tolerant ability of sc-PBFT and PBFT.

no-response nodes are all the reliable nodes losing connections, and there are f error messages from Byzantine nodes. To ensure the messages from the reliable nodes are majorities, the $n-f$ messages should be divided into $n-f-f$ reliable messages and f wrong messages, and it can be concluded that $n-f-f > f, n > 3f$. Therefore, tolerate f Byzantine nodes the total number of the nodes n should hold $n \geqslant 3f+1$.

According to the above analyze, if the number of Byzantine nodes holds $f \leqslant \left\lfloor \frac{n-1}{3} \right\rfloor$, these Byzantine nodes can't disturb the consensus results of the system. However, as time goes on, we assume that a Byzantine node comes out once every Δt, if these Byzantine nodes are not dealt with in time, they will reduce the system performance and result in $f > \left\lfloor \frac{n-1}{3} \right\rfloor$, and further threaten the security of the consortium blockchain. Our paper proposes the node state checking and isolation punishment mechanism which can report the node states after m rounds of consensuses to eliminate the Byzantine nodes in the network. Figure 6 shows the fault-tolerant ability of PBFT and sc-PBFT.

B. VALIDITY AND SECURITY OF THE PROPOSED EHR SHARING MODEL

Different from the model proposed in [38], our proposed scheme introduces blockchain technology to carry out sharing EHRs. The rightness of keyword searching and file decryption have been verified in [38].

The cloud server is partially trusted and can execute the user's request. In our proposed model, the cloud server stores the cipheryexts, it can't decrypt the cipheryexts without a decryption key. Therefore, data privacy can be guaranteed.

The access strategy is defined by ***DO***, the content key, and the file location and the file cipheryext will be decrypted when the attributes of ***DO*** satisfy the access strategy. Therefore, ***DU*** can't achieve the ***DO***'s EHR if its identity is not eligible. At the same time, our proposed scheme record the hash value of EHR and the access strategy of ***DO*** in the blockchain, ***DU*** can verify the data integrity and audit the data. Once the cloud server deletes or tamper the data illegally, After getting the data, ***DU*** will calculate the data hash and compare it with the hash stored in the blockchain. ***DU*** will easily find the difference between the two hashes and send the error report according to the hash collision. The proposed consensus algorithm can ensure the service of data storage and query implement successfully when ***DO*** stores data in the blockchain and ***DU*** sends a query request to the blockchain.

DO signs for the cipheryext and transaction based on the public key T. When sending the signature to ***CS*** and verifying the blockchain node, we adopt the RSA [39] algorithm to ensure the validity of the following equations.

$$Sig_{DO}(CT)^T = H(CT) \tag{6}$$

$$Sig_{DO}(trans)^T = H(trans) \tag{7}$$

DO uses the secret key to sign when sending the encrypted EHR to ***CS***. The identity of ***DO*** is secret because the public and secret key pair is fully trusted, which is assigned and stored by ***TC***. When ***DU*** accesses the data it will ensure the security of its identity by using the unique identity and attribute set. In the multi-keyword searching scheme, the sniffer can't guess the keywords and can't generate the corresponding token because the keyword index is generated by a random number. Therefore, the keyword searching scheme is secure.

C. MODEL CONTRASTIVE ANALYSIS

In this subsection, we compare our proposed EHR sharing scheme with other existing schemes in a contrastive way. As shown in Table 2, we summarize the advantages and disadvantages of our proposed EHR sharing model and the existing EHR sharing models. On the other hand, to illustrate the superiority of our proposed scheme, we analyze the problems of the existing sharing model and conclude the counter methods which are shown in Table 3. From Table 2 we can see that our proposed EHR sharing model has some advantages with respect to privacy protection and fine-grained access control. However, we can also see that our proposed scheme can't support smart contracts and in future work, we will use the smart contract to make the transaction process automatically.

D. THE TPS ANALYSIS OF sc-PBFT ALGORITHM

To evaluate the performance of the proposed sc-PBFT algorithm, we will test the effectiveness of sc-PBFT under the conditions of existing and absenting Byzantine nodes. We use 4 PCs to simulate the consortium blockchain network and the configuration of these PCs is shown in Table 4.

Comparing our proposed consensus algorithm with other existing algorithms such as PoW, Raft, PBFT under the conditions of existing Byzantine node and absenting Byzantine node. We test the handling capacity and latency. We would like to clarify that the test won't run on a specific blockchain platform, but is deployed on the above 4 PCs. The blockchain network is completed using Python 3.6. We use a lightweight open-source test framework wrk which is based

TABLE 2. Comparison between different models.

Function	Ref.[21]	Ref.[26]	Ref.[30]	Ref.[32]	Ref.[34]	Our scheme
Blockchain-based	×	√	√	√	√	√
Fine-grained access	√	√	√	×	√	√
Privacy protection	√	√	√	√	√	√
Searchable encryption	×	×	×	√	×	√
Data tamper-proofing	×	√	√	×	√	√
Smart contract	×	√	×	×	×	×

TABLE 3. The problems of the existing sharing model and the counter.

Problem Type	The facing problem	*The counter methods of our scheme*
Degree of patient's participation	The patient can't grasp the usage of his/her EHR.	Our proposed scheme uses CP-ABE to design a patient controllable EHR sharing scheme. The patient can set an access control strategy to determine who can access the EHR.
Privacy protection	EHR stores much private information, which may be under attack during data sharing. The identity of patients and users should be protected when they share or access EHRs.	This paper uses the encryption algorithm to encrypt the EHR and to ensure the security of the data. Using a signature algorithm and record the hash in the blockchain network, by doing like this, we can guarantee integrity and auditability. By encrypting and assigning the attribute to protect the privacy of the patient and DU.
Data interoperability	The data mobility between different hospitals is poor. When a patient sees a doctor, the medical records from other hospitals can't be shared. The integrity of EHR can't be protected.	This paper proposes a cloud-chain collaborated data storage method. Using CP-ABE to encrypt the data. The patient can set an access strategy to determine who can access the data. It means that our proposed scheme can implement data sharing among the hospitals. The user can use the searchable encryption scheme to search the cipheryext.
Medical dispute	When there is a conflict between the doctor and the patient, the authenticity of the EHRs can't be guaranteed.	Using blockchain to store the hash of EHR is data tamper-proofing and can guarantee the data authenticity.

on HTTP protocol and can use the asynchronous event-driven framework to bring large concurrent requests. Due to the client's needs to submit the consortium blockchain user information and data, we use lua script to generate POST requests. We use Transactions Per Seconds (TPS) as the handling capacity which can be defined as below.

$$TPS = \frac{number\ of\ transactions}{seconds} \tag{8}$$

In the experiments, we test Pow, Raft, PBFT, and sc-PBFT consensus algorithms 30 times. From Figure 7 we can see that

TABLE 4. The node configuration in consortium blockchain.

Configuration	Operation System	IP Address
Intel Core i7-7700, 8G	CentOS6.5	192.168.92.102
Intel Core i7-7700, 8G	CentOS6.5	192.168.92.103
Intel Core i7-10700, 16G	CentOS7	192.168.92.104
Intel Core i7-10700,16G	CentOS7	192.168.92.105

the partially decentralized Raft algorithm has the highest *TPS*, this is because the master node dispenses clients' requests, and the follower node only verifies the information content of the Leader node and doesn't judge whether the Leader node is Byzantine node among the Follower nodes. Our sc-PBFT and the PBFT algorithms are ranked the second and third, this is because in the two algorithms, the backup nodes not only verify the information content of the Primary node, but also judge whether the Primary node is the Byzantine node among the backup nodes, and the *TPS* is sacrificed to guarantee the security of the consortium blockchain network. We can also see that the Pow algorithm has the worst *TPS* because this algorithm is completely decentralized and generates new blocks is very difficult. By adding the number of Byzantine nodes we evaluate the *TPS* of our proposed sc-PBFT algorithm.

We set the time-out period as 10 seconds. For the mentioned decentralized algorithms Raft, PBFT and sc-PBFT. The four-node take a turn as the master node, and the Byzantine node will take a turn as the master node. While the nodes in the completely decentralized algorithm Pow have equal status and there is no master node. From Figure 8, we can see that when the Byzantine node becomes the master node, Raft algorithm doesn't have any defense abilities and the Byzantine nodes will act as the master node and won't be found. At this moment, the delivered messages in the whole network are all from the Byzantine nodes. While in the PBFT algorithm, when the Byzantine node takes turn as the master node, other backup nodes will identify it as the Byzantine node, which means that the nodes are busy checking the master node and can't deliver the user's messages. Same as PBFT algorithm, the sc-PBFT algorithm also reflects the same property in the first round of consensus. However, after first finding the Byzantine node, when meeting the Byzantine node again the sc-PBFT algorithm will deny it to take a turn as the master node. This makes the sc-PBFT generate 21.7% more blocks than PBFT algorithm in the same time period. Since Pow algorithm is completely decentralized consensus algorithm, and it can run normally if there are not more 51 percent of the nodes launching an attack.

E. THE LATENCY ANALYSIS OF sc-PBFT ALGORITHM

We test the latency of our proposed algorithm and the other three consensus algorithms under the conditions of existing

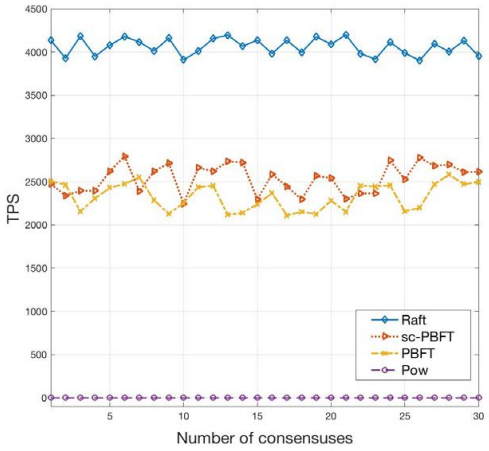

FIGURE 7. TPS test with no Byzantine node.

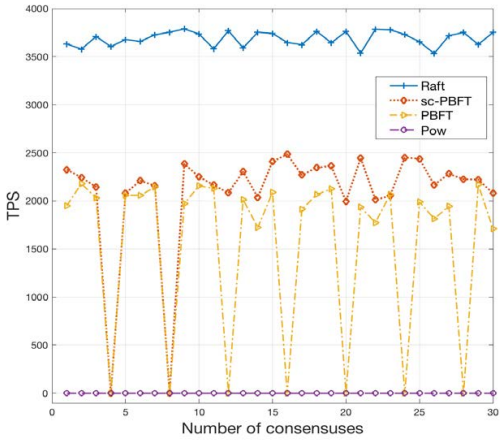

FIGURE 8. TPS test with one Byzantine node.

and absenting Byzantine nodes. We set the time-out period as 10 seconds and the nodes will be declined if there is no consensus response in 10 seconds. The latency equation is defined as below.

$$Latency = Time_{success} - Time_{request} \tag{9}$$

where $Time_{success}$ is the time of consensus complete successfully and $Time_{request}$ is the time of sending request. In this experiment, we calculate the latency 50 times under the conditions of existing and absenting Byzantine nodes. The average latency for the 50 testings is shown Figure 9. From Figure 9 we can see that the latency of Pow algorithm is the highest which is about 25s, but the number of Byzantine nodes is not more than 51% of the total number of the nodes, the Byzantine nodes will have little impact on the network. This is because Pow is a completely distributed consensus algorithm and every node competes to achieve the block, which takes plenty of time. The partially decentralized Raft algorithm has the shortest latency of about 0.4s under the condition of absenting Byzantine node. However, the premise is that no Byzantine node is elected as the master node, once the Byzantine node is elected as the master node the whole network will fall into a disordered state, and the consensus content and the latency will be grasped by

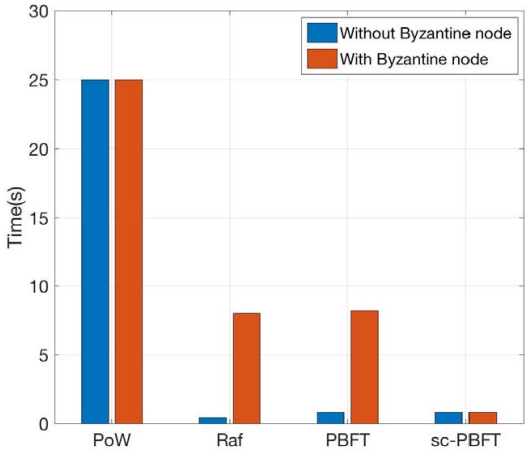

FIGURE 9. Consensus latency with and without Byzantine node.

the Byzantine node until time out. The speed of PBFT and sc-PBFT to achieve consensus is fast, and the latency of the two algorithms doesn't exceed 0.8s. Comparing with Raft algorithm, PBFT and sc-PBFT are more secure. If the number of Byzantine nodes doesn't exceed one-third of the total number of nodes, the tampered information will be found and modified by the backup nodes. However, the Byzantine nodes will make the consensus timeout and impact the user's communication. In the sc-PBFT algorithm, the first time that the Byzantine node served as the master node will be under the same influence, but the backup nodes won't support the Byzantine node to be the master node, which will avoid the Byzantine node disturbing the latency.

V. CONCLUSION

EHR sharing scheme is very important which can ensure the patients achieve comprehensive and accurate treatment. EHR sharing scheme can enforce the information exchange among different hospitals and promote the development of the medical area. Due to the EHR contains a lot of information, we should protect data privacy during the data sharing process.

The proposed scheme stores data in both on-chain and off-chain, which can ensure the authenticity and integrity of the shared data. This paper uses the multi-keywords searchable encryption scheme which can improve the efficiency and the accuracy of the cipheryext. In the consensus process, tolerate and capture the Byzantine node. We propose an improved PBFT consensus algorithm that adds the node state checking and malicious node isolating mechanism. By doing this, the proposed algorithm can isolate the Byzantine node to be the master node and reduce their impact on the consortium blockchain. The experimental results show that our proposed EHR sharing scheme has some advantages comparing other existing schemes. The consensus algorithm can reduce the disturbance of the Byzantine node and improve the production of the consensus result. In addition, there is still additional work to improve the proposed consensus algorithm. Although the proposed algorithm can capture and isolate the Byzantine nodes, it needs to restart the consortium blockchain network when deleting the consensus nodes in the information center. In the future, we will optimize the consortium blockchain network to improve the ability to dynamically add or delete the consensus node. We will also attempt to use the zero-knowledge proof and homomorphic encryption, etc., to encrypt the identity and EHR of the patient.

REFERENCES

[1] W. Lian, T. Xue, Y. Lu, M. Wang, and W. Deng, "Research on hierarchical data fusion of intelligent medical monitoring," *IEEE Access*, vol. 8, pp. 38355–38367, 2020.
[2] B. Shickel, P. J. Tighe, A. Bihorac, and P. Rashidi, "Deep EHR: A survey of recent advances in deep learning techniques for electronic health record (EHR) analysis," *IEEE J. Biomed. Health Inform.*, vol. 22, no. 5, pp. 1589–1604, Sep. 2018.
[3] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Blockchain for secure EHRs sharing of mobile cloud based E-health systems," *IEEE Access*, vol. 7, pp. 66792–66806, 2019.
[4] S. M. Shah and R. A. Khan, "Secondary use of electronic health record: Opportunities and challenges," *IEEE Access*, vol. 8, pp. 136947–136965, 2020.
[5] X. Yang, T. Li, W. Xi, A. Chen, and C. Wang, "A blockchain-assisted verifiable outsourced attribute-based signcryption scheme for EHRs sharing in the cloud," *IEEE Access*, vol. 8, pp. 170713–170731, 2020.
[6] H. Xiong and J. Sun, "Comments on 'verifiable and exculpable outsourced attribute-based encryption for access control in cloud computing,'" *IEEE Trans. Depend. Secure Comput.*, vol. 14, no. 4, pp. 461–462, Jul. 2017.
[7] H. Ma, R. Zhang, and W. Yuan, "Comments on 'control cloud data access privilege anonymity with fully anonymous attribute-based encryption,'" *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 866–867, Apr. 2016.
[8] A. Ge, J. Zhang, R. Zhang, C. Ma, and Z. Zhang, "Security analysis of a privacy-preserving decentralized key-policy attribute-based encryption scheme," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 11, pp. 2319–2321, Nov. 2013.
[9] Y. Qu, S. R. Pokhrel, S. Garg, L. Gao, and Y. Xiang, "A blockchained federated learning framework for cognitive computing in industry 4.0 networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2964–2973, Apr. 2021.
[10] Y. Zhuang, L. R. Sheets, Y.-W. Chen, Z.-Y. Shae, J. J. P. Tsai, and C.-R. Shyu, "A patient-centric health information exchange framework using blockchain technology," *IEEE J. Biomed. Health Informat.*, vol. 24, no. 8, pp. 2169–2176, Aug. 2020.
[11] F. Tang, S. Ma, Y. Xiang, and C. Lin, "An efficient authentication scheme for blockchain-based electronic health records," *IEEE Access*, vol. 7, pp. 41678–41689, 2019.
[12] V. Jaiman and V. Urovi, "A consent model for blockchain-based health data sharing platforms," *IEEE Access*, vol. 8, pp. 143734–143745, 2020.
[13] W. Y. Bin Saleem, H. Ali, and N. AlSalloom, "A framework for securing EHR management in the era of Internet of Things," in *Proc. 3rd Int. Conf. Comput. Appl. Inf. Secur. (ICCAIS)*, Mar. 2020, pp. 1–5.
[14] C. Soguero-Ruiz, K. Hindberg, J. L. Rojo-Alvarez, S. O. Skrovseth, F. Godtliebsen, K. Mortensen, A. Revhaug, R.-O. Lindsetmo, K. M. Augestad, and R. Jenssen, "Support vector feature selection for early detection of anastomosis leakage from bag-of-words in electronic health records," *IEEE J. Biomed. Health Informat.*, vol. 20, no. 5, pp. 1404–1415, Sep. 2016.
[15] W. Zhang, Y. Lin, J. Wu, and T. Zhou, "Inference attack-resistant E-healthcare cloud system with fine-grained access control," *IEEE Trans. Services Comput.*, vol. 14, no. 1, pp. 167–178, Jan. 2021.
[16] Y. Yang and M. Ma, "Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for E-health clouds," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 746–759, 2015.
[17] F. Xhafa, G. Mastorakis, C. X. Mavromoustakis, and C. Dobre, "Guest editorial: Special issue on algorithms and computational models for sustainable computing in cloud and data centers," *IEEE Trans. Sustain. Comput.*, vol. 2, no. 2, pp. 47–48, Apr. 2017.
[18] J. Shen, T. Zhou, D. He, Y. Zhang, X. Sun, and Y. Xiang, "Block design-based key agreement for group data sharing in cloud computing," *IEEE Trans. Depend. Sec. Comput.*, vol. 16, no. 6, pp. 996–1010, Nov./Dec. 2019.

[19] S. Alshehri, S. P. Radziszowski, and R. K. Raj, "Secure access for healthcare data in the cloud using ciphertext-policy attribute-based encryption," in *Proc. IEEE 28th Int. Conf. Data Eng. Workshops*, Apr. 2012, pp. 143–146.

[20] X. Li, F. Lv, F. Xiang, Z. Sun, and Z. Sun, "Research on key technologies of logistics information traceability model based on consortium chain," *IEEE Access*, vol. 8, pp. 69754–69762, 2020.

[21] C. Huang, R. Lu, H. Zhu, J. Shao, and X. Lin, "FSSR: Fine-grained EHRs sharing via similarity-based recommendation in cloud-assisted eHealthcare system," in *Proc. 11th ACM Asia Conf. Comput. Commun. Secur.*, May 2016, pp. 95–106.

[22] Y. Yuan and F.-Y. Wang, "Blockchain and cryptocurrencies: Model, techniques, and applications," *IEEE Trans. Syst. Man, Cybern., Syst.*, vol. 48, no. 9, pp. 1421–1428, Sep. 2018.

[23] Y. S. Rao, "A secure and efficient ciphertext-policy attribute-based signcryption for personal health records sharing in cloud computing," *Future Gener. Comput. Syst.*, vol. 67, pp. 133–151, Feb. 2017.

[24] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: http://bitcoin.org/bitcoin.pdf

[25] T. Xue, Q. Fu, C. Wang, and X. Wang, "A medical data sharing model via blockchain," *Acta Autom. Sinica*, vol. 43, no. 9, pp. 1555–1562, 2017.

[26] G. G. Dagher, F. Iqbal, M. Arafati, and B. C. M. Fung, "Fusion: Privacy-preserving distributed protocol for high-dimensional data mashup," in *Proc. IEEE 21st Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2015, pp. 760–769.

[27] B. Shen, J. Guo, and Y. Yang, "MedChain: Efficient healthcare data sharing via blockchain," *Appl. Sci.*, vol. 9, no. 6, p. 1207, Mar. 2019.

[28] M. S. Rahman, A. Al Omar, M. Z. A. Bhuiyan, A. Basu, S. Kiyomoto, and G. Wang, "Accountable cross-border data sharing using blockchain under relaxed trust assumption," *IEEE Trans. Eng. Manag.*, vol. 67, no. 4, pp. 1476–1486, Nov. 2020.

[29] N. Chondros, K. Kokordelis, and M. Roussopoulos, "On the practicality of practical Byzantine fault tolerance," in *Proc. Int. Midd. Conf.*, vol. 7662. New York, NY, USA: Springer, 2012, pp. 436–455.

[30] Y. Wang, C. Tang, F. Lin, Z. Zheng, and Z. Chen, "Pool strategies selection in PoW-based blockchain networks: Game-theoretic analysis," *IEEE Access*, vol. 7, pp. 8427–8436, 2019.

[31] G. G. Gueta, I. Abraham, S. Grossman, D. Malkhi, B. Pinkas, M. Reiter, D.-A. Seredinschi, O. Tamir, and A. Tomescu, "SBFT: A scalable and decentralized trust infrastructure," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2019, pp. 568–580.

[32] I. Alqassem, I. Rahwan, and D. Svetinovic, "The anti-social system properties: Bitcoin network data analysis," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 1, pp. 21–31, Jan. 2020.

[33] J. Wang, "Research progress of consensus algorithm like Paxos," *Comput. Res. Development*, vol. 56, no. 4, pp. 692–707, 2017.

[34] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX Annu. Tech. Conf.* Berkeley, CA, USA: USENIX Association, 2014, pp. 305–320.

[35] Y. Yuan, X. C. Ni, S. Zeng, and F. Y. Wang, "Blockchain consensus algorithms: The state of the art and future trends," *Acta Automatica Sinica*, vol. 44, no. 11, pp. 2011–2022, 2008.

[36] B. Yuan, H. Jin, D. Zou, L. T. Yang, and S. Yu, "A practical Byzantine-based approach for faulty switch tolerance in software-defined networks," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 2, pp. 825–839, Jun. 2018.

[37] N. O. Ahmed and B. Bhargava, "From byzantine fault-tolerance to fault-avoidance: An architectural transformation to attack and failure resiliency," *IEEE Trans. Cloud Comput.*, vol. 8, no. 3, pp. 847–860, Sep. 2020.

[38] J. Sun, L. Ren, S. Wang, and X. Yao, "Multi-keyword searchable and data verifiable attribute-based encryption scheme for cloud storage," *IEEE Access*, vol. 7, pp. 66655–66667, 2019.

[39] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

ZHEN PANG received the master's degree from the North China University of Technology. For many years, he has been engaged in the construction of medical information, the construction of medical information security protection capacity, and the security protection mechanism of medical big data. His research interests include information engineering and information security.

YUAN YAO received the master's degree from the North China University of Technology. His research interests include medical information construction and medical information security construction.

QIUYAN LI received the master's degree from the National Administration of Traditional Chinese Medicine. She is good at treating endocrine system diseases with integrated traditional Chinese and Western medicine, including but not limited to diabetes, menopause syndrome, depression, and insomnia. Long term focus on the research of internal medicine diseases in the treatment of Chinese and Western medicine and the modernization of inspection, auscultation and olfaction, interrogation, and palpation of the pulses of traditional Chinese medicine.

XIAOQIN ZHANG received the Ph.D. degree from the College of Computer Science, Chongqing University, in 2012. Since 2013, she studied at cloud security, big data analysis, and threat intelligence. She is currently a Professor Senior Engineer at Chongqing Communication Design Institute Company Ltd.

JING ZHANG received the master's degree from the Hebei University of Technology, in 2012. She is currently a Lecturer at the Tianjin Electronic Information College. Her research interests include network engineering, information security, and information and image processing.