Hindawi
Security and Communication Networks
Volume 2022, Article ID 3670107, 16 pages
https://doi.org/10.1155/2022/3670107

Research Article

A User-Centered Medical Data Sharing Scheme for Privacy-Preserving Machine Learning

Lianhai Wang, Lingyun Meng, Fengkai Liu, Wei Shao, Kunlun Fu, Shujiang Xu, and Shuhui Zhang

Qilu University of Technology (Shandong Academy of Sciences), Shandong Computer Science Center (National Supercomputer Center in Jinan), Shandong Provincial Key Laboratory of Computer Networks, Jinan 250014, China

Correspondence should be addressed to Lianhai Wang; wanglh@sdas.org

Received 25 June 2022; Accepted 24 August 2022; Published 30 September 2022

Academic Editor: Wenbo Shi

With the rapid development and application of artificial intelligence technology, medical data play an increasingly important role in the medical field. However, there are privacy protection and data ownership issues in the process of data sharing, which brings difficulties to machine learning and data mining. On the one hand, for fear that they may risk being held accountable by users or even breaking the law due to these issues, healthcare providers are reluctant to share medical data. On the other hand, users are also reluctant to share medical data due to the possibility of privacy disclosure in the data sharing process. To improve the security and privacy of shared medical data, we propose a user-centered medical data sharing scheme for privacy-preserving machine learning. Our solution combines blockchain and a trusted execution environment to ensure that adversaries cannot steal the ownership and control of user data during sharing. A blockchain-based noninteractive key sharing scheme is proposed that allows only the users and the TEE to decrypt the shared data. At the same time, we design an auditing mechanism to facilitate users to audit the sharing process. The security analysis shows that the scheme ensures the privacy and security of user data during storage and sharing. We have completed simulation experiments to demonstrate the effectiveness and efficiency of our scheme.

1. Introduction

In the era of the digital economy, data have become a new factor of production and an important basic strategic resource. Data support the future development and drive the progress in business or scientific fields. In particular, with the development of IoT technology in health care, the healthcare ecosystem generates many medical data, such as electronic medical records, monitoring data, imaging data, and smart wearable device data. These data contain a huge amount of information [1], which can assist physicians in clinical decision-making and play an important role in drug development, intelligent diagnosis, medical image recognition, and precision medicine [2]. Therefore, how to handle and utilize the growing amount of healthcare data has become an unavoidable problem.

With the rapid development and application of artificial intelligence (AI) technology, scholars have established several medical artificial intelligence models for intelligent analysis and decision-making using of medical data, especially for specialized medical record data. They have achieved fruitful research results [1]. The AI-based medical analysis requires medical data from multiple medical institutions, pharmaceutical companies, or individuals for extensive sample annotation and training [3].

For hospitals, due to the characteristics of medical data, such as privacy and sensitivity, there may be data security and privacy leakage risks in the sharing process. Once the patient's private data are leaked, the hospital will face medical disputes and even legal liability. On the other hand, the issue of ownership and access control of medical data are also controversial. If the hospital shares or uses the patient's medical data without authorization, it may be held accountable by the patient. Therefore, hospitals are always reluctant to share these data.

For patients, sharing data also exposes them to the risk of privacy breaches. Moreover, most patients' medical data are recorded in hospitals or healthcare facilities. Even if these

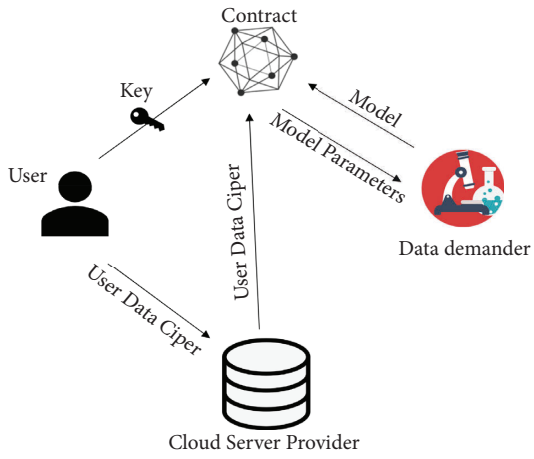

Figure 1: Medical data sharing framework.

data belong to the patients [4], it is difficult for patients to access their medical data. In addition, because of the reproducible nature of data, patients risk losing ownership of their data once they share them. So, there is also a reluctance to share their medical data with patients.

With the development of blockchain technology, its broad application prospects and technical features have attracted the attention of scholars related to various industries. Its distributed storage, peer-to-peer transmission, consensus mechanism, and confidentiality algorithm provide numerous novel solutions for data storage and sharing. Some researchers have used blockchain to implement on-chain data storage and smart contract-based access control to data [5]. Access rights are controlled by techniques such as identity-based [6] and attribute-based access controls [7] to ensure the privacy and security of shared data. However, under existing strategies, the efficiency in performing data sharing is lower due to the blockchain's storage space and computational performance. Moreover, once data are acquired by data demanders, they can access or use the data without any control. Therefore, this approach cannot prevent individuals or organizations from sharing data illegally, and it is even more difficult to ensure illegal analysis and misuse of data.

Some schemes combine blockchain with proxy re-encryption [8] to ensure the security and privacy of medical data during the sharing process. However, proxy re-encryption requires many computational resources, and the system is relatively inefficient. Moreover, these schemes do not consider the issue of control and ownership of data [9].

To address the above issues, we propose a user-centric medical data sharing scheme oriented to privacy-preserving machine learning to achieve efficient medical data sharing that protects data privacy. The model framework is shown in Figure 1. To ensure control and ownership of user data during storage and sharing, we design a new sharing model by combining blockchain with a trusted execution environment (TEE), although the combination of TEE and blockchain will encounter difficulties in data exchange and trust. But we propose a way of combining on-chain and off-chain to solve the problem of data exchange. In addition, we solve the trust problem between the two through signature authentication. We also design a new key sharing scheme for authorization management. In summary, this paper contributes the following:

(i) We propose a blockchain-based system for data sharing and privacy protection. A TEE obtains and deploys machine learning models from the blockchain, where data are decrypted for model training. Then, the training results are verified on the chain and shared with the data demander.

(ii) We have designed a new key sharing scheme to share and manage keys through smart contracts, which allows users to authorize their data off-chain.

(iii) We build a user audit mechanism. The record of the sharing process of users' medical data is stored on the blockchain, in which users can query at any time for auditing the sharing process.

(iv) We implement a prototype of our model and validate its effectiveness. The experiment shows that our scheme can protect the privacy and security of user medical data and ownership without incurring significant additional time overhead compared to existing solutions.

The remaining part of the paper is organized as follows. We begin by introducing some related works in Section 2. Section 3 is concerned with some preliminaries used in this paper. In Section 4, we describe the system model and design goals. In Section 5, the proposed system operational details are presented. In Section 6, we did a security and functional analysis. Program design and evaluation are presented in Section 7. Finally, this paper is concluded in Section 8.

2. Related Work

In this section, we review some research solutions for secure data sharing. To solve the problems of inefficiency and poor scalability of traditional medical data sharing systems, some schemes [10–12] are proposed using blockchain combined with cloud storage to solve the data security problem of data sharing. The data are stored encrypted in the cloud, and then, the storage index and data hash are uploaded to the blockchain for secure data storage and sharing. However, this data sharing method has no access control mechanism, and data security is difficult to guarantee. Meanwhile, when the data are shared, the data owner loses control and ownership of the data, which also poses a threat to the privacy and security of the data owner.

To protect the privacy and security of user data, some solutions propose controlled access [13] to manage healthcare data sharing. The literature [14] uses attribute-based sharing techniques, but when the access policy is modified, attribute revocation and encryption of the data are required again, which increases the computational overhead. Moreover, the tamper-proof nature of blockchain also complicates the modification of access control policies. Gu et al. proposed an efficient and simple attribute-based sharing scheme [15] that reduces computational costs and enables privacy protection. However, this scheme consumes

many computing resources when modifying the access policy. Wang and Song [16] proposed an electronic health record system built using attribute-based controlled access and blockchain technology. However, the whole system is too large, expensive to run, and inefficient to execute. Guo et al. [17] proposed an attribute-based signature scheme combined with blockchain technology that can protect user data privacy. However, the computational performance of the blockchain leads to an inefficient system. Moreover, when a malicious user obtains data through access control, there is a threat to user data privacy.

To ensure secure data sharing, several research proposals have proposed the use of cryptography [18] for data privacy security protection. Renpeng Zou et al. proposed SPChain [19] to enable medical data sharing for users in a privacy-preserving manner by using a proxy re-encryption scheme. However, this solution is difficult to implement, consumes too many resources, and has slow processing speed and poor portability. Chen et al. [20] proposed an anonymous medical data sharing scheme based on a cloud server and proxy re-encryption algorithm to improve the security of private medical data sharing. However, the original data in this scheme will still be accessed, and data privacy may be compromised.

Some scholars have used cryptography-based schemes to address these issues to protect the privacy and security of medical data during data analysis and after requesters access the data. Kosba et al. proposed a blockchain-based platform for contract development [21]. The platform uses a zero-knowledge proof-based cryptographic protocol to handle private data, rather than storing private transaction data directly on the blockchain, effectively guaranteeing private data security. However, this scheme requires many computing resources and is not scalable. The literature [22] proposes a blockchain privacy protection scheme based on homomorphic encryption, but it requires many computational resources, and its practicality and applicability are greatly limited.

To ensure that data in shared data do not leave the authorization system and thus do not disclose sensitive information, the literature in [23] proposes a combined blockchain and federation learning scheme for sharing privacy-preserving IoT data among distributed multiple parties. Another framework for sharing vehicle data based on blockchain and federated learning for edge computing is proposed in the literature [24] for the internet of vehicles (IoV). However, this scheme suffers from data poisoning that affects the global model and backdoor attacks. Zhou et al. proposed a health insurance storage system [25], which utilizes secret sharing techniques and secure multiparty computing that allows for the sharing of patient data between hospitals and insurance companies. However, this scheme does not guarantee that all servers are fully trusted, and data requesters have to receive responses from multiple nodes before accessing the data. Shamir's secret sharing and secure multiparty computation (MPC) were applied in Shrier et al. [26] to achieve data sharing while satisfying user privacy. Yue et al. [27] proposed a medical data gateway (HDG) to analyze medical data using secure multiparty computing while ensuring user control privileges. The literature [28, 29] combines multiparty secure computation and differential privacy to guarantee the accuracy of the output results without losing data privacy at the user's end. However, in complex computational tasks, the results can significantly differ from the noise-free results, making the results unusable.

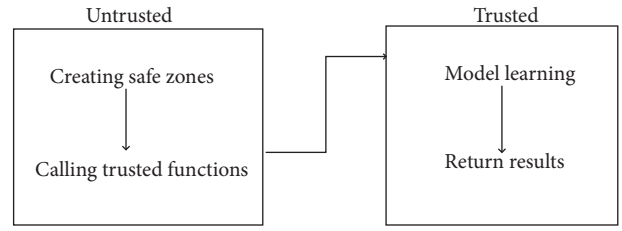

Figure 2: SGX schematic.

3. Preliminaries

3.1. Trusted Execution Environment. The trusted execution environment (TEE) is a secure zone in the computing platform, using a combination of trusted computing and virtualization isolation techniques. The TEE provides a trusted execution environment for "security-sensitive applications" while protecting the confidentiality and integrity of associated data. ARM's TrustZone technology implements hardware isolation mechanisms, mainly for embedded mobile terminal processors, to create separation between the secure and nonsecure worlds. In addition to TrustZone, based on the ARM architecture, Intel has also released a trusted execution environment based on its processor architecture: Intel SGX. SGX is a set of instructions that enhance the security of application code and data, providing them with more robust protection against disclosure or modification. Calling a program in the trusted zone requires defining the eCall interface and declaring the structure and size of the data to be passed. Intel SGX provides good integrity and confidentiality protection for its applications due to its hardware-level implementation. Since its release, it has been sought after by academia and industry and is used in scenarios such as outsourced cloud computing and sensitive data aggregation. Microsoft has proposed a database architecture EnclaveDB [30], based on SGX that runs within a secure zone. The data within the trusted database are implemented so that even when hackers compromise the server operating system; the data are still not accessible to the hackers. In addition to database applications, Kunkel et al. [31] ported machine learning [32] to the SGX secure zone, allowing the machine learning training and prediction process to take place within the secure zone, thereby protecting the privacy of the raw data. The SGX processing is shown in Figure 2.

3.2. Smart Contracts. Ethereum is a common public blockchain open-source platform whose main and most characteristic feature is integrating smart contract function. Ethereum provides a decentralized Ethereum virtual

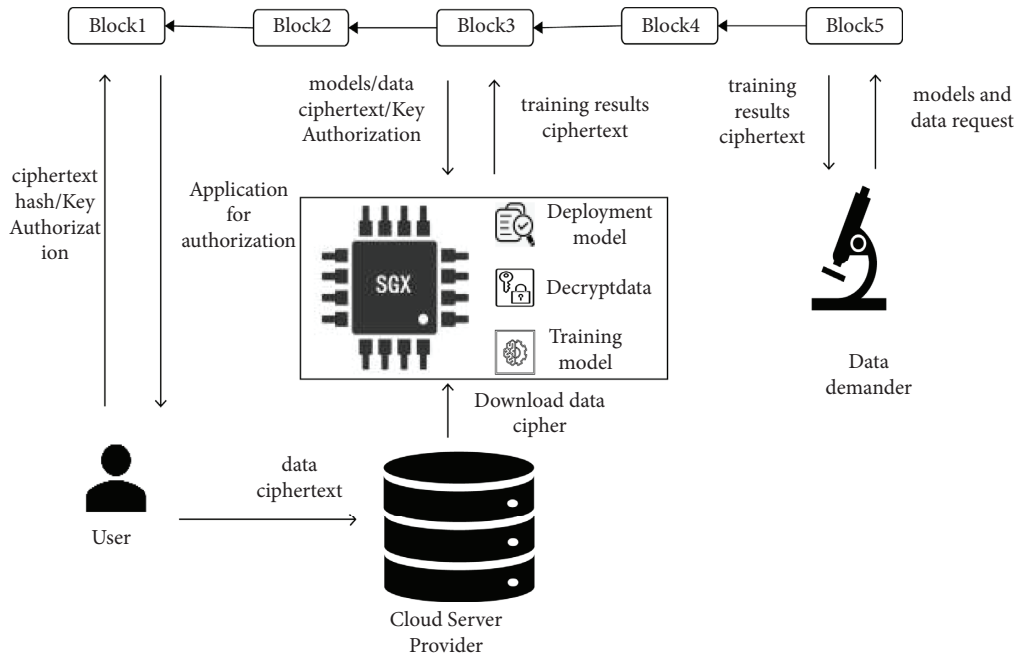

Figure 3: System model.

machine to users who join the Ether Place through its cryptocurrency, Ether (ETH), which provides peer-to-peer smart contract computing to all users. Ethereum is a typical representative of Blockchain 2.0, which increases the scalability and flexibility of the protocol based on Blockchain 1.0. By providing users with various modules, users can flexibly build smart contracts that suit their needs and deploy them into the Ethereum network by consuming Ether. Since the Ethereum virtual machine is Turing-complete, the business provided by Ethereum smart contracts is almost endless. In theory, any computer business can be deployed into Ethereum in the form of smart contracts to realize decentralized business operation and ensure the proper operation of the business as much as possible. Each command executed by the smart contract requires a certain amount of consumption, which uses gas as the unit. Also, different commands require different gases. Each transaction must first set a value called gasLimit, the maximum consumption value and miners have the right to choose which transaction to pack first. Generally, the larger the value of gasLimit, the more attractive miners are to pack. Ethereum is a highly integrated blockchain system in practical use, and users/developers mainly use smart contracts to publish some transactions and functional modules in Ethereum.

4. System Model

4.1. High-Level Overview. The flow of users sharing personal medical data in this solution is shown in Figure 3. The user encrypts and uploads the data to the cloud server for storage. Then, the user uploads the data cryptographic hash and data storage index to the blockchain. The data demander uploads the model and data request to the blockchain and then deploys the model into a TEE via a smart contract. The smart contract sends the data request to the user, who performs the key authorization.

The TEE downloads the data cipher hash and storage index on the blockchain and uses the storage index to download the data cipher from the cloud server. Then, TEE gets the key authorization from the smart contract and decrypts the data cipher to get the original data. The model is then trained using the user's medical data. Finally, the model training results are encrypted using the public key of the data demander and uploaded to the blockchain. The data demander then retrieves the training result ciphertext from the blockchain, makes its private key to decrypt it, and obtains the model training result.

4.2. System Architecture. The architecture of the proposed system is shown in Figure 3. It consists of six entities: user, medical research institute, TEE, blockchain, smart contract, and storage server. More details of the system are shown below.

User. The user stores medical data encrypted in the storage container and stores information such as the returned storage index and ciphertext hash in the blockchain. At the same time, users also authorize access requests to medical research institutions and encrypt the encryption key using TEE's public key, which is then stored on the blockchain. After the shared data are finished, the user can also call the chain code to query the data generated during the sharing process, thus playing a supervisory role.

Data Demander. Medical research institutions make the required models and submit them to the model review smart contract for review, then upload the machine learning models to the blockchain, and finally get the trained models from the blockchain.

TEE. A trusted, isolated, and independent execution environment exists independently of an untrusted operating system, providing a secure and confidential space for private data and sensitive computations in an untrusted environment, whose security is typically guaranteed through hardware-related mechanisms. In this scheme, we perform operations such as decrypting the user's data cipher, verifying data integrity, training model, and uploading training model parameters to the blockchain.

Storage Device. It is used in this solution to store user-encrypted medical data.

Smart Contracts. They are used to deploy machine learning algorithms, invoke medical data for sharing, and data transfer. They rely on smart contracts to achieve data scheduling and processing of data generated by the process of sharing data to achieve a regulatory mechanism.

Blockchain. The Ethernet public chain is used here. In this scheme, the blockchain is used to store data hashes, model, data integrity verification results, and model learning results.

4.3. Threat Model. To better describe the working process of the system model, we rely on the following assumptions:

(1) Smart contract records are reliable and readily available. This is because it is difficult for an attacker to tamper with the records posted on the blockchain, which is essentially a distributed ledger that runs all the time.

(2) This solution uses an off-chain storage system, mainly responsible for storing user medical data ciphertext. When the data are stored, a storage index is generated and used to store it in the blockchain.

(3) In this scheme, the computing process in the TEE cannot be accessed by the outside world in any way.

4.4. Design Goals. In this solution, we aim to achieve secure sharing of user medical data, for which we propose the following design objectives:

(1) *Secure Storage and Sharing.* Users' medical data should be securely stored, and no entity should be able to tamper with this information. In addition, it is guaranteed that no entity may view and tamper with the user's medical information during the sharing process.

(2) *Shared Data Can Be Supervised.* Users want to supervise the operations that their personal medical data undergo during the sharing process to prevent illegal use and analysis of personal medical data.

(3) *Efficiency.* A large amount of user medical data need to be stored and shared on time, so it should have high storage and sharing efficiency.

(4) *Data Dedicated Exclusive Use.* The user's medical data will only be used and analyzed for legitimate dedicated use, and any illegal manipulation of the user's data is not feasible.

(5) *User's Data Ownership.* This is done first to prevent malicious accounts from changing the user's account and tampering with the user's data ownership. Second, it ensures that after data sharing, the ownership of the data remains with the user.

(6) *Security of Computing Environment.* It ensures the privacy and security of users' medical data processing and does not disclose any private information when computing and analyzing data.

(7) *Ability to Resist Other Attacks.* To enhance security further, the protocol should provide resilience to other common attacks, such as replay attacks.

5. Our Proposed Protocol

The controlled medical data sharing for machine learning proposed in this solution can be specifically divided into the following phases: system initialization, medical data storage, machine learning model deployment, training model, and data demander to obtain training results. In Table 1, we illustrate some of the notations in the scheme.

5.1. System Initialization. Before the system starts to execute, we complete the initialization work. The specific steps are as follows:

(1) *Basic Initialization.* A cyclic additive group G with generating element g and prime order Q is chosen on an elliptic curve $E(Fp)$ over a finite field Fp and a one-way hash function $H1$: $\{0,1\}^* -> Z_q$. Then, the symmetric key encryption function Encrypt and decryption function Decrypt, Encrypt_*ECC* asymmetric key encryption algorithm and decryption algorithm Decrypt_*ECC*, and RSA signature function Sig_*RSA* and verification function Verify_*RSA* are selected.

(2) *Blockchain Initialization.* We create the file genesis.json containing the configuration parameters to build the Ethereum blockchain. Each node generates a public-private key pair $\{PK, SK\}$. One set of predesignated nodes is responsible for mining. The rest of the network collectively trusts these nodes to validate transactions and create new blocks. In our case, the trusted institutions consisted of medical research institutions, insurance companies, and regulatory agencies. Trusted institutions perform various functions, including adding data to decentralized file systems, uploading corresponding transactions to the blockchain, and validating various transactions received from external users, such as permission requests and permission grants.

(3) *Smart Contract Deployment.* In this scheme, there are four types of contract components: registration contract, data contract, authorization contract, model contract, and audit contract:

(a) *Registration Contract.* All nodes are registered anonymously on the registration contract to prevent users from providing false data or data

Table 1: Table notations.

Notation	Description
SK	Private key
PK	Public key
KE	Symmetric key
UDATA_i	Raw data of user
Model_j	Model of demander
UDT	Data type

demanders from providing illegal models. The registration information includes the public key and the role of the node.

(b) *Data Contract*. The data contract stores a list of data records that indicate the mapping relationship between the data and the user. Each data in the list consist of the user's public key, the data cipher hash, the off-chain original data storage index, and the user's signature. The ability to add, modify, and delete data is also provided in this contract. In addition, the TEE and users can retrieve and download data through this contract.

(c) *Authorized Contract*. The authorization contract assists the user in encrypting the data encryption key and authorizing the key to the data demander.

(d) *Model Contract*. The model contract stores a list of models and a list of model training results. Data demanders can upload models and download model training results through this contract. TEE can also store model training results through model contracts.

(e) *Audit Contract*. The audit list is set up, and the list data include information about the data owner, the data demander, the data integrity verification results, and the models trained on the data. Users can audit the process of sharing personal medical data through an audit contract.

5.2. Medical Data Storage. Due to the limitation of SGX memory, the user's medical data need to be preprocessed before uploading. We sort and label data by system requirements prior to data storage before storing it. In order to store the user medical data information, the structure $\{\mathrm{PK_i}, \mathrm{UDT}, \mathrm{DA_i}, \mathrm{US_i}, \mathrm{HDS_i}, DT_i, CM_i\}$ of the data storage transaction TD_i is designed. In this, TD_i is a public transaction and any node can access the data in TD_i. TD_i contains the user's public key PK_i, timestamp DT_i, hash of data ciphertext $H\ DS_i$, user signature US_i, data storage index DA_i, medical data typology $U\ DT$, and encryption key ciphertext CM_i. The following description illustrates the process of storing user medical data.

5.2.1. Data Preprocessing

(a) The user cuts and divides the data according to the system requirements, and then labels the divided data.

5.2.2. Data Upload

(a) *Encrypt the Raw Data*. User i calls Encrypt function to generate a new symmetric key KEY_i using private key SK_i and random number n_i. Then, we encrypt the medical data $U\mathrm{DATA}_i$ with key KEY_i to generate ciphertext DS_i.

$$DS_i = \mathrm{Encrypt}(\mathrm{KEY}_i, U\mathrm{DATA}_i). \tag{1}$$

(b) *Store ciphertext to Cloud Server*. User i stores medical data ciphertext to the cloud server and then gets the storage index DA_i.

5.2.3. Data on the Chain

(a) *Generate Hash Index*. User i uses hash function to generate hash value $H\ DS_i$ for data ciphertext DS_i.

$$HDS_i = H1(DS_i). \tag{2}$$

(b) *User Signature*. User i uses the signature function Sig_RSA to sign the data ciphertext hash to get the signature US_i.

$$US_i = \mathrm{Sig}_R SA(SK_i, H\ DS_i). \tag{3}$$

(c) *Encrypting the Symmetric Key*. User i invokes the authorized contract to obtain the public key PK_T of the TEE, the public key $PK_C$of the contract. Then, the user generates random numbers r_U and r_C, encrypts the symmetric key, and generates the ciphertext CM_i.

$$CM_i = \mathrm{KEY}_i + r_C PK_C + r_U PK_T. \tag{4}$$

(d) *Publish Stored Data Transactions*. User i invokes the data contract to store TD_i into the blockchain.

$$TD_i = \{PK_i, UDT\ , DA_i, US_i, HDS_i, DT_i, CM_i\}. \tag{5}$$

(e) *User Authorization*. The user invokes the authorization contract to upload the $r_U g$ and $r_C PK_C$.

Algorithm 1 shows the process of storing medical data from user i to the cloud server and blockchain.

5.3. Machine Learning Model Deployment

5.3.1. Model Storage. In order to store the machine learning model on the blockchain, the data structure $\{\mathrm{MID}_j, PK_j, MT_j, \mathrm{model}_j, H\mathrm{model}_j, SM_j, RM_j\}$ of the model storage transaction IM_j is designed. It contains the data demander node identifier MID_j, public key PK_j, timestamp MT_j, model model_j, hash of model $H\mathrm{model}_j$, signature SM_j of the medical institution, and data demand RM_j. The storage process of the user medical data is described as follows:

Input: Raw Data $UDATA_i$; Public Key PK_i; Data Type UDT; Random Number n_i; Private Key SK_i;
Output: Storage Index DA_i; Blockchain Transaction TD_i;
Stage 1: Upload data to the cloud server:
(1) Generate symmetric key KEY_i by PK_i
(2) Encrypted raw data $DS_i = \text{Encrypt}(KEY_i, UDATA_i)$
(3) Send DS_i to the cloud server and get DA_i
Stage 2: Upload TD_i to Blockchain
(4) Generate timestamp DT_i
(5) Generate ciphertext hash $HDS_i = H1(DS_i)$
(6) User Signature $US_i = \text{Sig\_RSA}(SK_i, HDS_i)$
(7) Call the authorization contract to get $\{r_C, PK_C, PK_T\}$
(8) Generate random number r_U
(9) Encryption symmetric key KEY_i: $CM_i = KEY_i + r_C PK_C + r_U PK_T$
(10) $TD_i = \{PK_i, UDT, DA_i, US_i, HDS_i, DT_i, CM_i\}$
(11) Call the data contract to upload data TD_i to the blockchain
(12) Call the Authorized contract to upload data TD_i, $r_U g$ and $r_C PK_C$
(13) return (DA_i, TD_i);

Algorithm 1: Medical data storage.

(1) We upload models to the blockchain.

(a) *Generating Models*. The data demander j produces and generates machine learning models and data requirements.

(b) *Obtaining the Model Hash*. The data demander j uses the hash function to calculate the hash value of the model.

$$H\text{model}_j = H1(\text{model}_j). \tag{6}$$

(c) *Data Demander Signature*. Data demander j uses the signature function Sig_*RSA* to sign the data ciphertext hash to get the signature SM_j.

$$SM_j = \text{Sig\_}RSA(SK_j, H\text{model}_j). \tag{7}$$

(d) *Posting Stored Data Transactions*. Data demander j invokes a data contract to store IM_j into the blockchain.

$$IM_j = \{\text{MID}_j, PK_j, MT_j, \text{model}_j, H\text{model}_j, SM_j, RM_j\}. \tag{8}$$

Algorithm 2 shows the process of storing the machine learning model and data requirements to the blockchain by the data demander j.

5.3.2. Model Deployment. The TEE retrieves the blockchain to get the model after it is stored on the blockchain. To ensure the authenticity of the model, the model needs to be validated. For this purpose, the data structure $\{ST_j, \text{HRT}_j, VT_j\}$ for model validation is designed, where ST_j is the signature of TEE, HRT_j is the model integrity verification result, and VT_j is the signature verification result. The process of model deployment is illustrated as described in the following:

(1) Model deployment

(a) *Download the Model*. After the model is stored to the blockchain, TEE retrieves the blockchain through the model contract and gets IM_j.

(b) *Verify Integrity*. TEE first retrieves the model hash $H\text{model}_j$ from IM_j, takes the model hash, and gets the hash $H\text{model}_j'$. TEE compares whether $H\text{model}_j$ and $H\text{model}_j'$ are equal and gets the result HRT_j. Then, it verifies the signature and gets the verification result VT_j.

$$\begin{aligned} H\text{model}_j' &= H1(\text{model}_j), \\ \text{HRT}_j &= \text{if}(H\text{model}_j == H\text{model}_j'), \\ VT_j &= \text{Verify\_}RSA(PK_j, SM_j). \end{aligned} \tag{9}$$

(c) *Upload Integrity Result*. TEE uses the private key SK_T to sign $\{\text{HRT}_j, VT_j\}$ to get the signature ST_j. Then, it invokes the authorization contract to upload the integrity verification result and store $\{ST_j, \text{HRT}_j, VT_j\}$ into the blockchain.

$$ST_j = \text{Sig\_RSA}(SK_T, \{\text{HRT}_j, VT_j\}). \tag{10}$$

(d) *Deploy Model*. The TEE deploys the model after verifying its integrity.

Algorithm 3 shows the process of model deployment to TEE.

5.4. Model Training

5.4.1. Data Verification. After the successful deployment of the model, TEE uses the authorization contract to retrieve the blockchain and obtain the user data $\{PK_i, UDT, DA_i, US_i, HDS_i, DT_i, CM_i\}$ that match the data requirements. Then, we download the data ciphertext DS_i from the cloud server according to the label type of the data required by the model. To ensure the authenticity and

Input: **model**: Data demander node identification MID_j; Public Key PK_j; Private Key SK_j;
Output: Blockchain Transaction IM_j;
(1) Generate models and data requirements
(2) Generate timestamp MT_j
(3) Generate model hash $H\mathrm{model}_j = H1(\mathrm{model}_j)$
(4) Signature $SM_j = \mathrm{Sig\_}RSA(SK_j, H\mathrm{model}_j)$
(5) $IM_j = \{\mathrm{MID}_j, PK_j, MT_j, \mathrm{model}_j, H\mathrm{model}_j, SM_j, RM_j\}$
(6) Call the model contract to upload data IM_j to the blockchain

Algorithm 2: Model storage.

Input: Private Key SK_T;
Output: Verify the result of the hash value HRT_j; The result of verifying the signature VT_j; Signature ST_j;
(1) Call the model contract to download model $\{\mathrm{MID}_j, PK_j, MT_j, \mathrm{model}_j, H\mathrm{model}_j, SM_j, RM_j\}$
(2) Generate model hash $H\mathrm{model}_j\prime = H1(\mathrm{model}_j)$
(3) $\mathrm{HRT}_j = \mathrm{if}(H\mathrm{model}_j == H\mathrm{model}_j\prime)$
(4) Verify signature $VT_j = \mathrm{Verify\_RSA}(PK_j, SM_j)$
(5) Sign off on the validation results $ST_j = \mathrm{Sig\_}RSA(SK_T, \{\mathrm{HRT}_j, VT_j\})$
(6) Deploy model
(7) Call the model contract to upload $\{ST_j, \mathrm{HRT}_j, VT_j\}$ to the blockchain
(8) return $\{ST_j, \mathrm{HRT}_j, VT_j\}$

Algorithm 3: Model deployment.

integrity of the data, the data need to be verified. For this purpose, the data structure $\{ST_i, \mathrm{HRT}_i, VT_i\}$ for data validation is designed, where ST_i is the signature of TEE, HRT_i is the model integrity verification result, and VT_i is the signature verification result. As described below, the process of model deployment is illustrated:

(a) *DownloadTD_i*. TEE invokes the data contract to retrieve the blockchain and download the TD_i that matches the data requirements.

(b) *Download Data CipherDS_i*. We retrieve the cloud server to download the corresponding data cipher DS_i according to the data type required by the model.

(c) TEE first retrieves the data cipher hash HDS_i from TD_i and then hashes the model to get the hash HDS_i'.TEE compares whether HDS_i' is equal to HDS_i and gets the result HRT_i. Then, it verifies the signature and gets the verification result VT_i.

$$\begin{aligned} \mathrm{HDS}_i' &= H1(DS_i), \\ \mathrm{HRT}_i &= \mathrm{if}(\mathrm{HDS}_i == \mathrm{HDS}_i'), \\ VT_i &= \mathrm{Verify\_RSA}(PK_i, US_i). \end{aligned} \tag{11}$$

(d) *Upload Integrity Result*. TEE uses private key SK_T to sign $\{\mathrm{HRT}_i, VT_i\}$ to get signature ST_i, and then invoke authorization contract to upload integrity verification result and store $\{ST_i, \mathrm{HRT}_i, VT_i\}$ into blockchain.

$$ST_i = \mathrm{Sig\_}RSA(SK_T, \{\mathrm{HRT}_i, VT_i\}). \tag{12}$$

Algorithm 4 shows the process of data downloading.

5.4.2. Data Acquisition and Model Training. After obtaining the user data cipher and verifying the data integrity, TEE invokes the authorization contract to request a key. TEE uploads the data integrity verification result and model integrity verification result to the authorization contract. The authorization contract will receive the $r_U g$ and $r_C PK_C$ and send to TEE. TEE receives the $r_U g$ and $r_C PK_C$ to calculate the decryption key. The process is illustrated as described in the following:

(1) Key authorization

(a) *Contract Authorization*. The authorization contract sends $r_C PK_C$ to TEE along with $r_U g$.

(2) Key acquisition

(a) *Retrieve Key*. TEE retrieves the key cipher CM_i from TD_i.

(b) *Decrypt Key*. TEE uses $r_C PK_C$ and $r_U g$ to decrypt the encryption key.

$$\begin{aligned} \mathrm{KEY}_i &= CM_i - r_C PK_C - r_U PK_T \\ &= CM_i - r_C PK_C - r_U SK_T g \\ &= CM_i - r_C PK_C - r_U g SK_T. \end{aligned} \tag{13}$$

(3) Data decryption

(a) *Decrypt the Data Cipher*. TEE calls the decrypt function Decrypt, decrypts the data cipher $\mathrm{DS_i}$ using KEY_i, and gets the data $U\mathrm{DATA}_i$.

$$U\mathrm{DATA}_i = \mathrm{Decrypt}(\mathrm{KEY}_i, DS_i). \tag{14}$$

(4) Model training

Input: Private Key $\mathbf{SK_T}$;
Output: Verify the result of the hash value HRT_i; The result of verifying the signature VT_i; Signature ST_i;
(1) Call the data contract to download data $\{PK_i, U\ DT\ , DA_i, US_i, H\ DS_i, DT_i, CM_i\}$
(2) Download data ciphertext from cloud storage server DS_i
(3) Generate model hash $\mathrm{HDS}_i\prime = H1(DS_i)$
(4) $\mathrm{HRT}_i = \mathrm{if}(\mathrm{HDS}_i == \mathrm{HDS}_i\prime)$
(5) Verify signature $VT_i = \mathrm{Verify\_}RSA(PK_i, US_i)$
(6) Sign off on the validation results $ST_i = \mathrm{Sig\_RSA}(SK_T, \{\mathrm{HRT}_i, VT_i\})$
(7) Call the data contract to upload $\{ST_i, \mathrm{HRT}_i, VT_i\}$ to the blockchain
(8) return $\{ST_i, \mathrm{HRT}_i, VT_i\}$

Algorithm 4: Data verification.

Input: Private Key SK_T; Ciphertext of the key CM_i; Ciphertext of the data $\mathrm{DS_i}$;
Output: Encrypted keys KEY_i; raw data $U\mathrm{DATA}_i$; Training results of the model MRT_j;
(1) Call the Authorization contract to download $\{r_C PK_C, r_U g\}$
(2) Decrypted keys: $\mathrm{KEY}_i = CM_i - r_C PK_C - r_U PK_T$
$= CM_i - r_C PK_C - r_U SK_T g$
$= CM_i - r_C PK_C - r_U g SK_T$
(3) Decrypt data: $U\mathrm{DATA}_i = \mathrm{Decrypt}(\mathrm{KEY}_i, DS_i)$
(4) Training model with data
(5) return $\{\mathrm{KEY}_i, U\mathrm{DATA}_i, \mathrm{MRT}_j\}$

Algorithm 5: Data acquisition and model training.

(a) TEE uses data from multiple users to train the model on the demander of the data and obtains the training result MRT_j after the model is trained.

Algorithm 5 shows the process of decrypting the data and training the model.

5.5. Training Result Acquisition. After the model training is completed, the training results need to be encrypted and stored in the blockchain in order to protect the privacy and security of the model training results. The data demander gets the model ciphertext from the chain and then decrypts it to get the model training results. The process is illustrated as described in the following.

5.5.1. Training Result Storage

(a) *Encrypt Training Results*. TEE uses the public key of the data demander to encrypt the training results and get the ciphertext.

$$\mathrm{CMRT}_j = \mathrm{Encrypt\_}ECC\left(PK_j, \mathrm{MRT}_j\right). \quad (15)$$

(b) *Hash*. TEE takes a hash of the training result ciphertext.

$$\mathrm{HCMRT}_j = H1\left(\mathrm{CMRT}_j\right). \quad (16)$$

(c) *Generate Signature*. To ensure the authenticity of the training results, TEE signs the cryptographic hash of the training results.

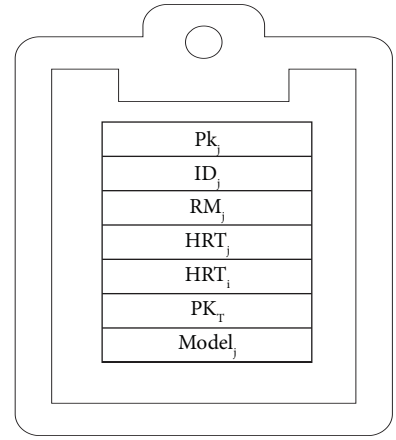

Figure 4: Structure of audit information.

$$\mathrm{SMRT}_j = \mathrm{Sig\_RSA}\left(SK_T, \mathrm{HCMRT}_j\right). \quad (17)$$

(d) *Upload to Blockchain*. TEE invokes the model contract to upload $\left\{\mathrm{CMRT}_j, \mathrm{HCMRT}_j, \mathrm{SMRT}_j\right\}$ to the blockchain.

Algorithm 6 demonstrates this process.

5.5.2. User Audit Information Storage. To ensure that users audit the sharing process of personal medical data at any time and avoid the unauthorized use of data, we design the user audit structure $\left\{\mathrm{MID}_j, RM_j, \mathrm{HRT}_j, \mathrm{HRT}_i, PK_T, PK_j, \mathrm{model}_j\right\}$ to store the audit information and then invoke

Input: Public Key PK_j; Training result of the model MRT_j; Private Key SK_T;
Output: Ciphertext of training result CMRT_j; Hash HCMRT_j; Signature SMRT_j;
(1) Encrypt training result $\mathrm{CMRT}_j = \mathrm{Encrypt\_}ECC(PK_j, \mathrm{MRT}_j)$
(2) Generate hash value $\mathrm{HCMRT}_j = H1(\mathrm{CMRT}_j)$
(3) Generate signature $\mathrm{SMRT}_j = \mathrm{Sig\_RSA}(SK_T, \mathrm{HCMRT}_j)$
(4) Call the model contract to upload $\{\mathrm{CMRT}_j, \mathrm{HCMRT}_j, \mathrm{SMRT}_j\}$ to the blockchain
(5) return $\{\mathrm{CMRT}_j, \mathrm{HCMRT}_j, \mathrm{SMRT}_j\}$

Algorithm 6: Training result storage.

Input: Hash HCMRT_j; Signature SMRT_j; ciphertext CMRT_j; Private Key SK_j;
Output: Training result of the model MRT_j;
(1) Call the Authorization contract to download $\{\mathrm{CMRT}_j, \mathrm{HCMRT}_j, \mathrm{SMRT}_j\}$
(2) Generate hash value $\mathrm{HCMRT}_j' = H1(\mathrm{CMRT}_j)$
(3) Result = If $(\mathrm{HCMRT}_j == \mathrm{HCMRT}_j')$
(4) $\mathrm{Verify\_RSA}(PK_T, \mathrm{SMRT}_j)$
(5) Decrypt result ciphertext $\mathrm{MRT}_j = \mathrm{Decrypt\_}ECC(SK_j, \mathrm{CMRT}_j)$
(6) return MRT_j

Algorithm 7: Decrypt result ciphertext.

the audit contract to store it into the blockchain for user audit. Here, MID_j is the ID of the data demander, RM_j is the data demand, HRT_i is the model integrity verification result, HRT_i is the data verification result, $\mathrm{PK_T}$ is the public key of TEE, PK_j is the public key of the data demander, and model_j is the model of the data demander. The audit information structure is shown in Figure 4.

5.5.3. Training Rezsult Acquisition

(a) *Download the Training Result Ciphertext*. The data demander invokes the model contract to obtain $\{\mathrm{CMRT}_j, \mathrm{HCMRT}_j, \mathrm{SMRT}_j\}$.

(b) *Verify Ciphertext*. We calculate the hash of the training result ciphertext, compare it with the hash downloaded from the chain, and then verify the signature. If the verification passes, the decryption process is carried out, and if the verification does not pass, the feedback is sent to the blockchain.

$$\begin{aligned}\mathrm{HCMRT}_j' &= H1\left(\mathrm{CMRT}_j\right),\\ \mathrm{Result} &= \mathrm{If}\left(\mathrm{HCMRT}_j == \mathrm{HCMRT}_j'\right),\\ &\quad \cdot \mathrm{Verify\_RSA}\left(PK_T, \mathrm{SMRT}_j\right).\end{aligned} \tag{18}$$

(c) *Decrypt Ciphertext*. The data demander decrypts the data using the private key after successful verification.

$$\mathrm{MRT}_j = \mathrm{Decrypt\_}ECC\left(SK_j, \mathrm{CMRT}_j\right). \tag{19}$$

Algorithm 7 demonstrates this process.

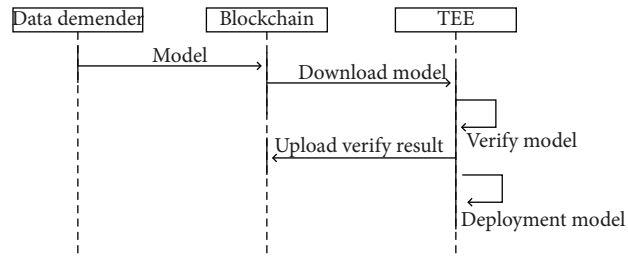

Figure 5: The logical flow of machine learning model deployment.

5.6. Interaction Process Description. The interaction process of the solution consists of ten steps as follows. First, steps 1–4 describe the process of deploying the machine learning model. The logical flowchart is shown in Figure 5. Then, steps 5–8 describe the process of user data storage, sharing, and key authorization. The logic flowchart is shown in Figure 6. The final steps 9-10 describe the process of using user data to train the machine learning model and the process of transmitting the model learning results to the data demander. The logical flowchart is shown in Figure 7:

Step 1: the data demander uploads the machine learning model and data demand to the block company.

Step 2: the data demander invokes the contract to transfer the machine learning model to the TEE.

Step 3: the TEE verifies the integrity of the machine learning model.

Step 4: the TEE deploymodel.

Step 5: the user encrypts the medical data and stores it in the cloud server. Then, the information such as data cryptographic hash, ciphertext of the key, and storage index are uploaded to the blockchain. Then, the user invokes the authorization contract to upload the key.

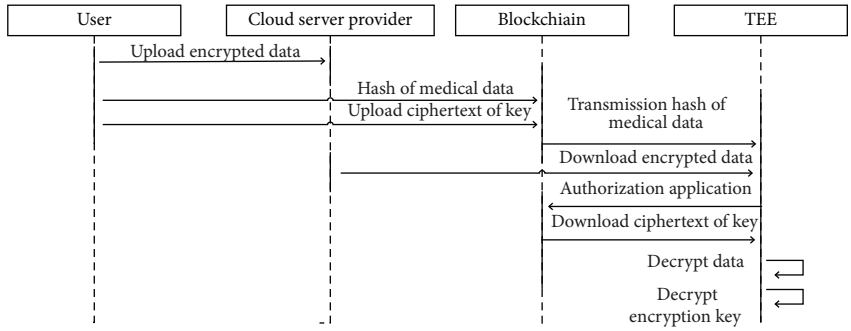

Figure 6: The logical flow of user medical data sharing and key transfer.

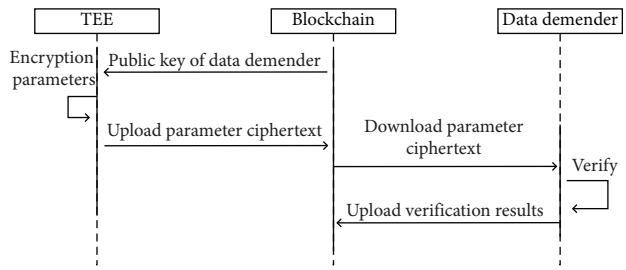

Figure 7: The logical flow of model training and return of the training results.

Step 6: the TEE downloads the user medical data and verifies the integrity of the data.

Step 7: the TEE calls the authorization contract to obtain the key.

Step 8: the TEE decrypts the key ciphertext and then encrypts the data ciphertext to obtain the user data.

Step 9: the TEE uses the user's data to train the model and then encrypts the training results.

Step 10: the TEE calls the contract to upload the encrypted training results to the blockchain.

Step 11: the data demander downloads the training result ciphertext, decrypts it, and obtains the training result.

6. Security and Functional Analysis

In this section, security analysis and functional analysis of the proposed scheme are performed in order to verify that the previously mentioned design objectives are met.

6.1. Security Analysis

6.1.1. Secure Storage and Sharing. In most cases, the storage server is trusted, but sometimes the storage server gets curious about the data and looks at the user's personal data. Therefore, in this scheme, the user data are stored in the storage server in an encrypted state, and the user data cannot be viewed without the key. At the same time, the hash value of data ciphertext is stored in the blockchain to verify the integrity of data, and this mechanism effectively ensures data storage security. During the sharing process, user data are in an encrypted state. Only TEE can use the private key decryption to obtain the encryption key. TEE ensures that the internal computation is hidden, and the internal data cannot be accessed from outside. Therefore, this solution can also ensure the security of data sharing.

6.1.2. Shared Data Can Be Regulated. In this solution, first, users can view their personal medical data at any time. Second, information such as the results of data integrity verification, the identity of the medical research institution, and the models to be trained generated during the sharing process are uploaded to the blockchain. All these information can be viewed by users at any time as a way to regulate personal data and thus prevent the illegal use of personal medical data.

6.1.3. Efficiency. First, in this scenario, we use DES symmetric key to encrypt the user's medical data and upload the data ciphertext to the storage device. The hash of the data ciphertext is then uploaded to the blockchain for storage. This reduces the storage burden of the blockchain and improves the storage efficiency of the system. Second, to achieve efficient machine learning model training on the blockchain, we introduce a combination of on-chain and off-chain approaches. On-chain contracts perform low-complexity operations such as data provisioning and integrity verification, while off-chain TEE performs high-complexity calculations such as data encryption and decryption hash calculation, signing, signature verification, and model training. In this way, the computational burden of the blockchain is significantly reduced, and the efficiency of the whole system is improved.

6.1.4. Dedicated to Medical Data. In this solution, medical data are only used to train machine learning models of medical research institutions. No entity can access the user's medical data and let alone perform other operations on the user's medical data, to ensure the exclusive use and non-misuse of the user's medical data.

6.1.5. User Data Ownership. First, the user's medical data are not really shared but used for training models, and the medical research organization does not really have access to

Table 2: The comparison of functionality and security with current solutions.

Security properties	Zou et al. [19]	Miao et al. [33]	Chen et al. [20]	This article
Safe storage and sharing	√	√	√	√
Blockchain based	√	√	√	√
User auditing	×	√	×	√
Environment security	×	×	×	√
Protecting user ownership	×	×	×	√
Minimized data usage	×	×	×	√

the user's data. Second, the user's data are trained in the TEE, and no entity can be aware of the computational process inside, and much less access the data in the TEE. In this way, the ownership of user data can be protected.

6.1.6. Computing Environment Security. This scheme uses Intel SGX for data decryption and model training. SGX aims to safeguard the confidentiality and integrity of users' critical code and data from malware by making hardware security mandatory and not to rely on firmware and software's security status. SGX's trusted computing base (TCB) contains only hardware, avoiding the pitfalls of software-based TCBs that have their own software security vulnerabilities and threats. In addition, SGX guarantees a TEE at runtime so that malicious code cannot access and tamper with the protected content of other programs at runtime, further enhancing the system's security. SGX's robust, trusted, flexible security features and hardware scalable performance guarantees provide a secure computing environment for this solution.

6.1.7. Resilience against Other Attacks

(1) Replay Attack. The scheme is effective against replay attacks because all transactions in the system contain timestamps and digital signatures. Moreover, since all transactions in the blockchain are transparent, any user can extract the time when the transaction was generated. This way, if a malicious user tries to duplicate a transaction request using a transaction written on the blockchain, then during the validation phase of the transaction, the relevant validation node will detect the time discrepancy and discard the transaction.

(2) Impersonation Attack. In this scenario, TEE provides proof for the issued data, for example, the integrity verification result of the data, the integrity verification result of the model, the public key of the TEE, and the completed model of the training. This is to prevent the illegal elements from impersonating TEE to cheat the user's data encryption key. On the other hand, the user's data will also contain the user's signature to ensure the authenticity of the user's data. This prevents illegals from using malicious data to influence the model's training.

(3) Tampering Attacks. In the process of medical data sharing, there may be cases where users tamper with blockchain information or transaction information, such as changing the owner of published medical data to their own account, thus tampering with the ownership of medical data. The solution is based on Ethernet deployment, and the authenticated nodes generate the blocks. Here, the authenticated nodes need to complete a mandatory authentication process to get the right to generate new blocks. Therefore, blocks are packed by trusted certified nodes, and malicious nodes cannot learn the private keys of trusted certified nodes, so they cannot forge the identity of certified nodes to pack blocks and thus cannot modify block information to forge signatures. Since it is difficult for malicious nodes to tamper with the data on the blockchain, and we store the cryptographic hash of medical data and share records on the chain, this ensures the accuracy and authenticity of the records.

6.2. Function Analysis. In Table 2, we compare our scheme with existing schemes. As can be seen from the table, all these schemes are based on blockchain for data sharing. Among them, Zou et al. [19], Miao et al. [33], and Chen et al. [20] designed privacy protection for the medical data sharing process. However, our solution meets the practical needs of dedicated data dedication, secure data handling, data regulation, and data ownership.

7. Program Design and Evaluation

In this section, we analyze the effectiveness of the proposed scheme through experiments. We have conducted a simulation experiment, which is divided into four parts. First of all, we build the Ethernet blockchain on the Ubuntu 20.0 virtual machine and write an intelligent contract using solidity. Then, we build Intel SGX in Intel (R) Core (TM) i7-9750H CPU @ 2.60 GHz, 16gb RAM, Microsoft Windows 10 operating system, implement trusted execution environment (TEE), and redesign encryption and decryption algorithm, hash algorithm, and signature algorithm in SGX. We realize the functions of data decryption, machine learning, hash generation, signature, and learning result encryption in the security zone. In order to compare the impact of SGX on the efficiency of the whole shared system, we also implement the above functions in a non-SGX environment and compare the computing time overhead in the two environments. Finally, by adjusting different difficulties, we test the appropriate block time and test the throughput of the system.

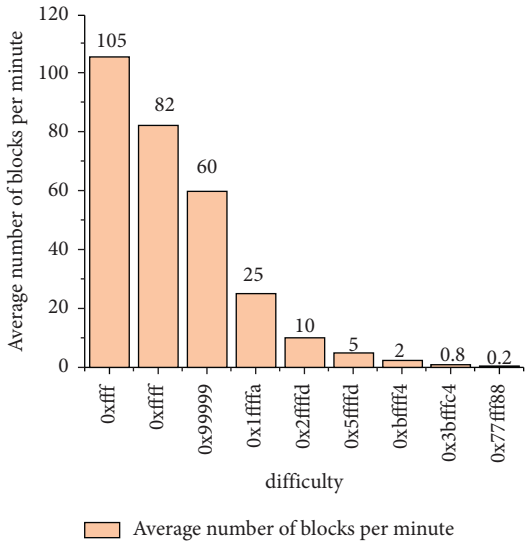

Figure 8: The average number of blocks per minute.

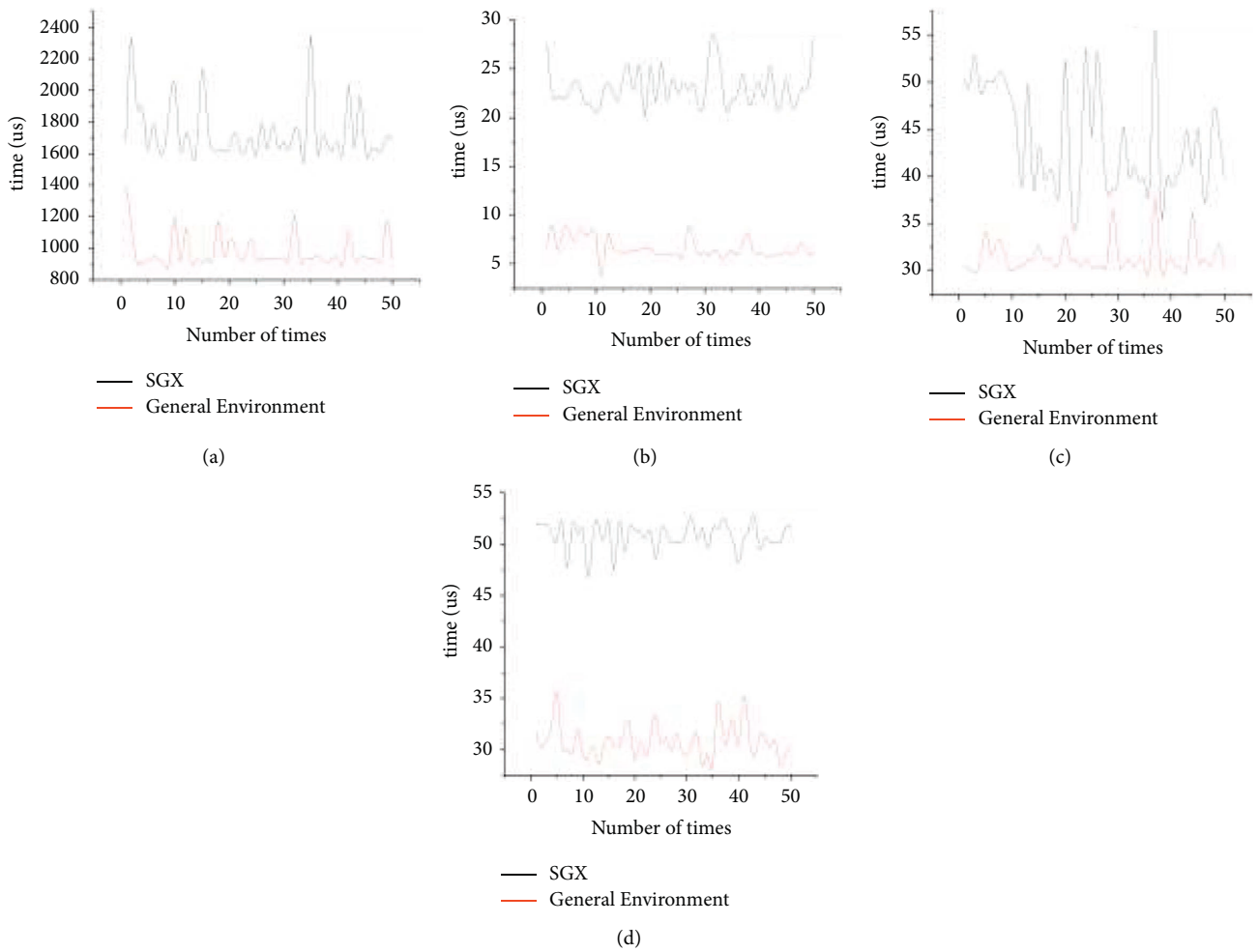

Figure 9: Comparison between SGX and general environment. (a) ECC key pair generation time comparison, (b) comparison of the time overhead of calculating hash values, (c) comparison of the time overhead of generating signatures, and (d) comparison of the time overhead of the whole system.

Table 3: System time overhead for different file sizes.

Data size (kB)	1	2	3	4	5	6
Minimum of NOSGX (us)	58216	170421	276373	460588	544606	705014
Average of NOSGX (us)	67868	179441	307853	524873	600304	725899
Maximum of NOSGX (us)	78341	194061	357842	625825	672885	795646
Minimum of SGX (us)	230948	416418	474531	858997	1028615	1238720
Average of SGX (us)	278031	441992	507291	887812	1181386	1366159
Maximum of SGX (us)	315733	478269	548383	932949	1235315	1456512

7.1. Ethereum Blockchain Building. The blockchain built in this system is based on the Geth client, version 1.9.25-stable-e7872729, which is based on Ether and uses POW as the consensus algorithm. There are 4 mining nodes built in the federated chain network, and the 4 nodes take turns to start mining. We modify the difficulty value in the genesis file to test the block generation time overhead at different difficulty levels. As shown in Figure 8, the vertical coordinate represents the difficulty and the horizontal coordinate represents the number of blocks generated per minute, and we set the scheme out of blocks to 60 blocks per minute.

7.2. Building a Trusted Execution Environment. Based on the excellent performance of Intel SGX, we use SGX as the trusted execution environment. The design of the Enclave of the security zone needs to consider both function and implementation. The analysis of the scheme shows that Enclave needs to have the following functions: (1) generating asymmetric key pairs inside Enclave; (2) exporting public keys outside Enclave; (3) decrypting key ciphertexts and data; (4) training models using data; (5) encrypting training results; and (6) signing, verifying signatures, and computing hash values.

Since SGX's internal functions are not convenient, we rebuilt the algorithm in Enclave for the above functions. We rebuilt ECC asymmetric encryption and symmetric encryption and decryption for higher security and smaller key size. We use the RSA signature for signature and verification signature. For the use of the hash function, we choose the SHA-256 algorithm.

7.3. SGX Performance Evaluation. To test the impact on the overall sharing system efficiency after using SGX, we designed SGX-based data sharing and non-SGX data sharing for comparison. This is shown in Figure 9. We use 3 kB data to compare the ECC key pair generation time comparison, data hash generation time comparison, signature time comparison, and the total time comparison of the whole sharing system between the SGX environment and the non-SGX environment. Figure 9(a) shows the key pair generation time comparison, which takes a little more time in the SGX environment than in the normal environment. However, this time overhead is not significant, and on average, the SGX environment takes 757.307 us more time overhead. We also tested the time overhead of generating data hashes, signing, and verifying signatures in both environments. The additional time overhead for generating hashes, signatures, and verifying signatures in the SGX environment is

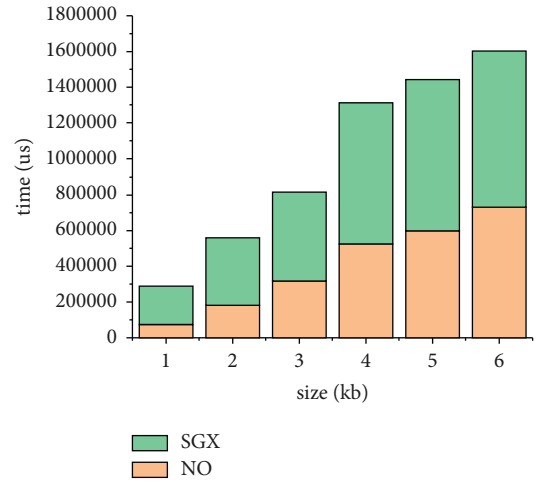

Figure 10: Time comparison for different data sizes.

16.566 us, 12.464 us, and 6.916 us, respectively. There is a small additional time overhead for the above calculations when using SGX, but it is still a microsecond overhead. It does not have a significant impact on the overall system. Figure 9(d) shows the time overhead of the whole system in the SGX environment compared to the non-SGX environment. Overall, the SGX environment still sacrifices some efficiency, but in terms of average time, the SGX environment has an additional 19.9443 ms overhead.

To further test the impact of SGX on the system, we performed test simulations using 1 to 6 kB of data, as shown in Table 3. Since the excess time loss of SGX is due to the data going in and out of Enclave, the time increase with SGX is lower than the time increase without SGX as the data size increases, as shown in Figure 10.

7.4. Blockchain Throughput. In this section, we separately calculate the throughput of various transactions. Since we are using the Ethernet blockchain, the block gasLimit of Ethernet determines the number of transactions that can be packed in a block. The current block gasLimit of the Ethernet blockchain we built is 12,000,000 gas. After testing, we get the gasLimit for user-submitted personal medical data transactions to be 62,060 gas. Since we set the block-out speed to 60 minutes, the data processing per second (TPS) for personal user data is 193. Similarly, testing the gasLimit for data demander-submitted transactions is to be 82,355 and the TPS for deployment model transactions is to be 145. The gasLimit of integrity verification transaction is 38444

gas, and the TPS is 312. The gasLimit of SGX uploading machine learning model training result ciphertext and its hash value transaction is 79,928 gas, and the TPS is 150.

8. Conclusions

This paper proposes a user-centric medical data sharing scheme for privacy-preserving machine learning, which implements data encryption storage, blockchain-based data resource distribution, data authorization, and machine learning model training. We also design an auditing mechanism to assist users in auditing the data sharing process. Compared with existing schemes, our proposed scheme ensures the privacy and security of users' data and safeguards the ownership of users' data and achieves the dedicated use and nonmisuse of data. Finally, the functionality of this solution is implemented through simulation experiments, and the experimental results prove the feasibility and effectiveness of the solution. Analyzing the impact of the TEE on the overall system performance demonstrates that the privacy and security of data and the user's data ownership are guaranteed without significant performance degradation. In future work, we intend to reduce the communication overhead of users and increase the throughput of the system.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Shandong Provincial Key Research and Development Program (2021CXGC010107 and 2020CXGC010107) and the National Natural Science Foundation of China (62102209).

References

[1] T. J. Hirschauer, H. Adeli, and J. A. Buford, "Computer-aided diagnosis of Parkinson's disease using enhanced probabilistic neural network," *Plenum Press*, vol. 101, 2015.

[2] H. Jin, Y. Luo, P. Li, and J. Mathew, "A review of secure and privacy-preserving medical data sharing," *IEEE Access*, vol. 7, no. 99, Article ID 61656, 2019.

[3] D. W. Bates, S. Saria, L. Ohno-Machado, A. Shah, and G. Escobar, "Big data in health care: using analytics to identify and manage high-risk and high-cost patients," *Health Affairs*, vol. 33, no. 7, pp. 1123–1131, 2014.

[4] L. Campanile, M. Iacono, F. Marulli, and M. Mastroianni, "Designing a GDPR compliant blockchain-based IoV distributed information tracking system," *Information Processing & Management*, vol. 58, no. 3, Article ID 102511, 2021.

[5] M. Pradnya Patil, "Sangeetha, vidhyacharan bhaskar. Blockchain for IoT access control, security and privacy: a review," *Wireless Personal Communications*, vol. 35, 2020.

[6] P. C. K. Hung and Y. Zheng, "Privacy access control model for aggregated e-health services," in *Proceedings of the EDOC Conference Workshop*, Eleventh International IEEE, Annapolis, MD, USA, October 2007.

[7] Pavithran Deepa and N. Al-Karaki Jamal, "Shaalan khaled. Edge-based blockchain architecture for event-driven IoT using hierarchical identity based encryption," *Information Processing & Management*, vol. 58, no. 3, 2021.

[8] P. Shabisha, A. Braeken, A. Touhafi, and K. Steenhaut, "Elliptic curve qu-vanstone based signcryption schemes with proxy Re-encryption for secure cloud data storage," vol. 21, 2019.

[9] L. Campanile, M. Iacono, F. Marulli, and M. Mastroianni, "Designing a GDPR compliant blockchain-based IoV distributed information tracking system," *Information Processing & Management*, vol. 58, no. 3, Article ID 102511, 2021.

[10] A. Eb, A. Fc, and B. Cg, "BlockHealth: blockchain-based secure and peer-to-peer health information sharing with data protection and right to be forgotten," *ICT Express*, vol. 34, 2021.

[11] W. Shen, T. Hu, C. Zhang, and S. Ma, "Secure sharing of big digital twin data for smart manufacturing based on blockchain," *Journal of Manufacturing Systems*, vol. 61, pp. 338–350, 2021.

[12] Y. Chen, Z. Lu, H. Xiong, and W. Xu, "Privacy-preserving data aggregation protocol for fog computing-assisted vehicle-to-infrastructure scenario," *Security and Communication Networks*, vol. 2018, pp. 1–14, 2018.

[13] L. Yang, W. Zou, and J. Wang, "EdgeShare: a blockchain-based edge data-sharing framework for industrial Internet of things," vol. 43, 2021.

[14] F. Chen, J. Huang, C. Wang et al., "Data access control based on blockchain in medical cyber physical systems," *Security and Communication Networks*, vol. 34, pp. 1–14, 2021.

[15] K. Gu, W. Jia, G. Wang, and S. Wen, "Efficient and secure attribute-based signature for monotone predicates," *Acta Informatica*, vol. 54, no. 5, pp. 521–541, 2017.

[16] H.. Wang and Y.. Song, "Secure cloud-based EHR system using attribute-based cryptosystem and blockchain," *Journal of Medical Systems*, vol. 42, no. 8, pp. 152–164, 2018.

[17] R.. Guo, H.. Shi, Q.. Zhao, and D. Zheng, "Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems," *IEEE Access*, vol. 6, pp. 11676–11686, 2018.

[18] J. S. Lee, C. J. Chew, and J. Y. Liu, "Medical blockchain: data sharing and privacy preserving of EHR based on smart contract,"vol. 74.

[19] R. Zou, X. Lv, and J. Zhao, "SPChain: blockchain-based medical data sharing and privacy-preserving eHealth system," *Information Processing & Management*, vol. 58, no. 4, Article ID 102604, 2021.

[20] Z. Chen, W. Xu, B. Wang, and H. Yu, "A blockchain-based preserving and sharing system for medical data privacy," *Future Generation Computer Systems*, vol. 124, pp. 338–350, 2021.

[21] A. Kosba, A. Miller, and E. Shi, "Hawk: the blockchain model of cryptography and privacy-preserving smart contracts," in *Proceedings of the 2016 IEEE Symposium on Security and Privacy*, pp. 839–858, IEEE Press, Los Alamitos, CA, USA, 2016.

[22] P. Golle, K. Leytonbrown, and I. Mironov, "Incentives for sharing in peer-to-peer networks," *Lecture Notes in Computer Science*, vol. 49, pp. 75–87, 2001.

[23] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2020.

[24] X. Huang, Y. Lu, K. Zhang, S. Maharjan, and Y. Zhang, "Blockchain empowered asynchronous federated learning for secure data sharing in Internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4298–4311, 2020.

[25] L. Zhou, L. Wang, and Y. Sun, "MIStore: a blockchain-based medical insurance storage system," *Journal of Medical Systems*, vol. 42, no. 8, pp. 149–165, 2018.

[26] A. A. Shrier, A. Chang, and N. Diakun-Thibault, *Blockchain and Health IT: Algorithms*, Privacy, and Data, 2016.

[27] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, "Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control," *Journal of Medical Systems*, vol. 40, no. 10, p. 218, 2016.

[28] S. Truex and N. Baracaldo, "A hybrid approach to privacy-preserving federated learning," https://arxiv.org/abs/1812.03224.

[29] H. Li Hao, G. Xu, S. Liu, and H. Yang, "Towards efficient and privacy-preserving federated deep learning," in *Proceedings of the ICC 2019 - 2019 IEEE International Conference on Communications*, pp. 1–6, ICC), Shanghai, China, May 2019.

[30] C. Priebe, K. Vaswani, and M. Costa, "EnclaveDB: A Secure Database Using SGX," in *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP)*, IEEE, San Francisco, CA, USA, May 2018.

[31] R. Kunkel, D. L. Quoc, and F. Gregor, "TensorSCONE: a secure TensorFlow framework using intel SGX," vol. 42, 2019.

[32] M. Abadi, P. Barham, and J. Chen, "TensorFlow: a system for large-scale machine learning," *USENIX Association*, vol. 12, 2016.

[33] Q. Miao, H. Lin, J. Hu, and X. Wang, "An intelligent and privacy-enhanced data sharing strategy for blockchain-empowered Internet of Things," *Digital Communications and Networks*, vol. 88, 2022.