Permissioned Blockchain Frame for Secure Federated Learning

Jin Sun, Ying Wu, Shangping Wang, Yixue Fu, and Xiao Chang

***Abstract*—Federated learning is an emerging technology that solves the privacy problem of training data in multi-party machine learning. However, this technology is vulnerable to a series of system security problems. In this letter, we leverage Hyperledger Fabric permissioned blockchain architecture to build a secure and reliable federated learning platform across multiple data owners, where individual local updates are encrypted based on threshold homomorphic encryption and then recorded on a distributed ledger. The security analysis shows that our solution can effectively deal with the existing privacy and security issues in the federated learning system. The numerical results show that the scheme is feasible and efficient.**

***Index Terms*—Federated learning, blockchain, homomorphic encryption, security and privacy.**

I. Introduction

IN RECENT years, the machine learning(ML) has become a key technology artificial intelligence(AI). In order to carry out high-quality ML model training in a multi-party setting, massive amounts of data are collected by the centralized server as the training sample set of the model. However, due to the concerns about data rights and interests, many users don't willing to share their data for model training so that resulting in the "data island" widespread. It is still problematic for the centralized server to collect and process large and scattered data for ML. As a novel distributed ML, federated learning is a promising technology. In federated learning, each distributed client trains the local model on the local database, and then sends the local model gradient to the central coordinator for global model optimization [1]. Compared with traditional ML, in federated learning, the client only needs to send the local model gradient without revealing the local raw data to any other participants.

However, the current federated learning system relies on a central coordinator, so there are still problems of single-point failure and trust. Blockchain is a decentralized, traceable, tamper-proof distributed ledger that can provide data security and data validation for federated learning [2], [3], ensuring data security and consistency of model parameters among untrusted participants [4]. Despite blockchain is an effective solution to replace the attack-prone central coordinator in the federated learning system, the combination of federated learning and blockchain may bring potential security and privacy attacks. First of all, all local updates are recorded as plain texts in blocks. Curious participants or servers can infer sensitive information of a target participant through observation of the local updates submitted by him. Secondly, there is no guarantee that there are no malicious participants alone or in collusion among multi-party entities in a distributed network. Malicious participants may purposefully modify the label of local data and intentionally send adversarial local updates to deteriorate the accuracy of the shared model. The system needs to be able to withstand some security attacks, such as collusion attack, sybil attack, poisoning attack, and inference attack [5].

Manuscript received September 24, 2021; accepted October 14, 2021. Date of publication October 18, 2021; date of current version January 10, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61303223, the Natural Science Foundation of Shaanxi Province under Grant 2021JM-341, the Key R & D projects in Shaanxi Province under Grant 2020GY-006, and the Xi'an Science and technology project under Grant 2020KJRC0092. The associate editor coordinating the review of this letter and approving it for publication was M. H. Cheung. *(Corresponding author: Jin Sun.)*

The authors are with the School of Science, Xi'an University of Technology, Xi'an, Shaanxi 710048, China (e-mail: oksunjin@xaut.edu.cn).

Digital Object Identifier 10.1109/LCOMM.2021.3121297

Existing researches are for developing privacy preserving solution through cryptography algorithm [6], differential privacy [7] or security aggregation. Unfortunately, encrypted local updates cannot be verified and the adoption of differential privacy can affect the utility of the global model, which leads to system security issues. The blockchain-aided federated learning scheme addressing both security and privacy issues has not been reported so far.

In this letter, aiming at solving the security and privacy problems simultaneously, a federated learning system based on Hyperledger Fabric framework was proposed. In particular, the permissioned blockchain records each global model update process, which is possible to verify and track local updates according to the permanent and immutable records. This also facilitates designing an incentive mechanism according to each users' contribution. In addition, homomorphic encryption [8] was used in the scheme to protect users' local model updates. In order to resist the poisoning attack, we add a verification process before the local update aggregation, and provide privacy by adding differential privacy noise to mask the local update during the verification process. Finally, a secure and verifiable aggregation scheme of local updates is achieved through Shamir secret sharing technique [9] instead of the differential privacy technique [7], which balances the trade-off between utility and privacy.

II. Background

A. Hyperledger Fabric

Fabric is a distributed operating system for permissioned blockchain and one of the Hyperledger projects hosted by the Linux Foundation [10]. Fabric ledger is a series of records of ordered and tamper-proof transactions. The structure is composed of a blockchain storing the immutable and ordered records and a status database to record the current data state. Blockchain services are the core part of Fabric, including consensus mechanism management, deployment of Fabric ledgers, storage of ledgers, and communication between peers. The key components of Fabric are as follows:

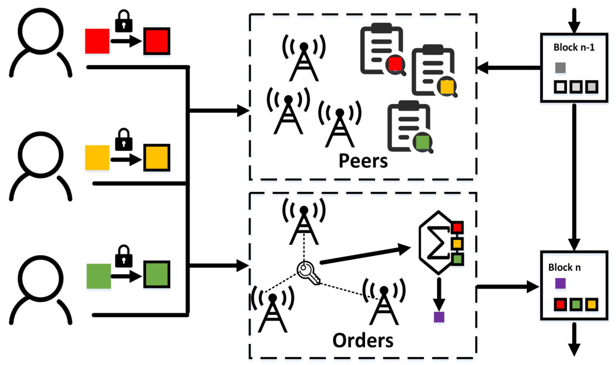

Fig. 1. System architecture.

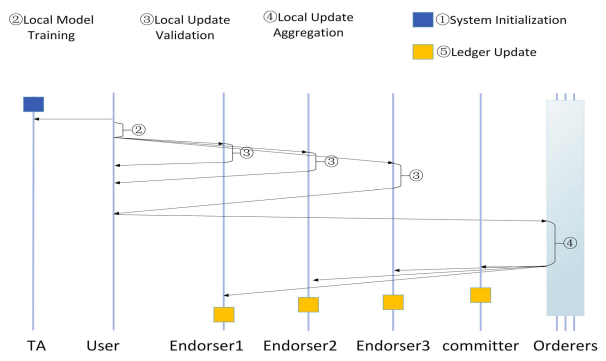

Fig. 2. Transaction flow in the proposed scheme.

- Membership Service Provider (MSP): It manages the identity of all nodes in the network, which contains several rules to determine whether the identity is valid or not.
- Channel: Each channel is a private subnetwork allowing data isolation and confidentiality. Each channel runs independently and can only be shared between users within the channel.
- Peer: Peer is the basic unit that performs operations and stores data in Fabric. The endorsing peer is responsible for verifying the transaction proposal and returning the endorsement response to the transaction proposer. The order node orders the endorsed transactions according to the consensus protocol. The committing peer verifies the validity of the ordered transactions and updates the block in the ledger.
- Chaincode: System chaincode includes system configuration, endorsement, and verification policies; User chaincodes are used to implement business logic and interact with the ledger based on specific requirements.

B. Homomorphic Cryptosystem

A crypotosystem property, which allows arbitrary computation of the corresponding plaintext by performing operations on the ciphertext without decrypting it, is known as full homomorphic. Howerever, the current full homomorphoc encryption algorithm still has the practical efficiency problem. Therefore, a partial homomorphic cryptosystem known as Paillier is used in our scheme. Paillier cryptosystem is a public key cryptography with additive homomorphism, that is, given the ciphertext $E(m_1)$ and $E(m_2)$, the ciphertext of $m_1 + m_2$ can be obtained by the calculation:

$$E(m_1 + m_2) = E(m_1) \cdot E(m_2).$$

III. Proposed Scheme

The proposed scheme consists of three entity roles: users, nodes, Trusted Authority (TA). A user can be an individual or an organization who owns the data collected from IoT devices and aims at collaboratively training the shared model as shown in Fig.1. Nodes are the entities to maintain the blockchain with computing and storage capabilities. TA are considered fully trusted and will never collude with any entity or disclose any private information. The transaction flow in the Fabric network between participants is shown in Fig.2.

A. System Initialization

1) Parameters Generation: **Step1**: TA chooses the security parameters of the system, digital signature scheme (RSA, ECDSA, etc.), and initializes the public parameters of the system.

Step2: Fabric network initialization. TA initializes a channel for the model to be trained with nodes and an endorsement policy. An endorsement policy specifies a set of nodes to endorse a transaction and the specific endorsement chaincode. TA initializes parameters of the shared model which is stored in the genesis block.

Step 3: In each iteration process, TA needs to generate and distribute the keys:

1) TA randomly selects two strong primes p and q, computs $n = pq$ and are satisfied with $p = 2p' + 1$, $q = 2q' + 1$, and $\gcd(n, \varphi(n)) = 1$.

2) Let $m = p'q'$, and $\beta \in_R Z_n^*$ be a randomly element. Then TA randomly chooses $(a, b) \in_R Z_n^* \times Z_n^*$ and set $\theta = L(g^{\beta m}) = a\beta m \bmod n$. Let secret key $SK = \beta m$ and PK consists of g, n, and θ.

3) Let SK is shared with Shamir scheme: let $a_0 = \beta m$, TA randomly chooses t values $(a_1, a_2, .., a_t)$ in $\{0, \ldots, n \times m - 1\}$ and constructs a polynomial:

$$f(x) = \sum_{i=0}^{t} a_i x^i.$$

Then he computs the share $s_i = f(i) \bmod n^m$ and distributes s_i to the i^{th} odering node.

2) Participants Registration: When a new participant (user or node) joins the network, he must register with the TA. The participant generates a unique identity identifier m_{ID} through their own built-in algorithm and registers with TA using m_{ID} and system public parameters. With the identity certificate, users can join the channel for specific model training, inquiry ledger status, and submit local update transactions.

B. Local Model Training

During the local training, users train the latest global model locally and encrypt local updates. In order to prevent malicious users from launching poisoning attacks, users need to send local updates data to endorsing nodes to verify the accuracy of local updates data and filter out malicious updates before model aggregation. In order to ensure the privacy of local updates data in the verification process, users use differential privacy (DP) to mask their local updates by adding noise. The specific process is as follows:

Step 1: After the user U_j obtaining the $i-1^{th}$ global model ω_i from the latest block B_i, he trains the global model on the local private dataset $D_{i,j} \subseteq D_j$ and calculates the loss function:

$$L(D_{i,j}, \omega_{i,j}) = \frac{1}{D_{i,j}} \sum_{(x_j,y_j)\in D_{i,j}} l(y_j, f(x_j, \omega_{i,j})).$$

The user operates iteratively and calculates local model updates $\nabla\omega_{i,j} = \nabla L(D_{i,j}, \omega_{i,j})$.

Step 2: The user U_j puts forward a noising request to TA, and TA returns $(\xi, sign_{TA}(E_{PK}(\xi)))$ to U_j, where ξ is from a normal distribution of (ε,δ)-differentially private [11]. U_j adds noise to $\nabla\omega_{i,j}$ and obtain the masked SGD update values: $\nabla\tilde{\omega}_{i,j} = \nabla\omega_{i,j} + \xi$. U_j randomly selects $r_j \in Z_n^*$ and computes the ciphertext of $\nabla\omega_{i,j}$ and $\nabla\tilde{\omega}_{i,j}$ by using the Paillier encryption algorithm:

$$E_{PK}(\nabla\omega_{i,j}, r_j) = c_{i,j} = g^{\nabla\omega_{i,j}} {r_j}^n \bmod n^2,$$
$$E_{PK}(\nabla\omega_{i,j}+\xi, r_j) = \tilde{c}_{i,j} = g^{\nabla\omega_{i,j}+\xi} {r_j}^n \bmod n^2.$$

Step 3: U_j initiates a local update data proposal $Tranprsl_{i,j}$:

$$Tranprsl_{i,j} = (\tilde{c}_{i,j}||c_{i,j}||\Delta\tilde{\omega}_{i,j}||E_{PK}(\xi)|| sign_{TA}(E_{PK}(\xi))||Cert_{U_j}||tp||\delta_j^P).$$

where δ_j^P is a signature on the proposal with his private key, which is used to guarantee the authenticity and integrity of the transaction, and tp is the transaction timestamp. U_j submits $Tranprsl_{i,j}$ to the endorsing nodes for endorsement through invoking the Fabric SDK.

C. Local Update Validation

After the endorsing node receives the local update proposal, the following operations will be performed:

Step1: *Verification of information*. EP checks the validation of the signature and whether the submitter is authorized to submit the transaction on the channel. Moreover, it needs to confirm the consistency between $c_{i,j}$ and $\tilde{c}_{i,j}$ through the homomorphic property of the encryption system:

$$\tilde{c}_{i,j} = c_{i,j} \cdot E_{PK}(\xi).$$

The masked update data $\nabla\tilde{\omega}_{i,j}$ is equal to the unmasked update data in $c_{i,j}$ if and only if the equation holds.

Step2: *Verification of local updates*. The endorsing node get the i^{th} global model ω_i from the latest i^{th} block B_i on the blockchain, and update ω_i with $\nabla\tilde{\omega}_{i,j}$.The accuracy of the updated model on a small testset is the indicator to judge whether $\nabla\tilde{\omega}_{i,j}$ is qualified. If the accuracy is lower than the preset threshold, then the transaction will be discarded. Otherwise, it executes the chaincode function to outcome the endorsement result. Each endorsing node signs with its private key to generate a signature δ_e. The transaction response $\{Endorse_{i,j}^k||\delta_e\}$ is returned back to U_j.

Step3: *Collection of endorsements*. After receiving the transaction response returned by the endorsing node, the user U_j checks the validity of the signature and the consistency of the endorsements received. Ensuring that the endorsement policy is fulfilled, the user submits the local update transaction $Tx_{update\_i,j}$ to the order node:

$$Tx_{update\_i,j} = (Add_{U_j}||c_{i,j}||\{Endorse_{i,j}^k\}_{k=1}^K|| \times Sign_{sk_{U_j}}(c_{i,j})).$$

D. Local Update Aggregation

On the transactions received from the channels, the order node validates the local update transactions for aggregation into the global model. The detailed steps are as follows:

Step 1: *Verification of information*. After collecting enough local update transactions from users, the order node verifies the validity of the user's identity and confirms the validity of the transaction endorsements according to the predefined endorsement policies. Afterward, order nodes calculate ciphertext of aggregated updates by using the homomorphic property of encryption:

$$c_i = \prod_{j=1}^{n} c_{i,j} = g^{\sum \nabla\omega_{i,j}} (\prod r_j) \bmod n^2 (1 \le j \le n)$$

Step 2: *Aggregated updates Decryption*. Each order node $A_k (1 \le k \le l)$ decrypts c_i using its partial key s_k according to the following steps: $d_k = {c_i}^{2l!s_i} \bmod n^2$. If the leader node receives less than t partial decryptions, it cannot obtain the aggregation update operations. If not less than t, he obtains the aggregated updates $\nabla\omega_i$ according to the following steps:

$$\nabla\omega_i = \frac{1}{n}\sum_{j=1}^{n} \nabla\omega_{i,j} = L(\prod_{k\in\Gamma} d_k^{2\psi_{(0,k)}^{\Gamma}}) \times \frac{1}{4n(l!)^2\theta} \bmod n,$$

where $\psi_{(0,k)}^{\Gamma} = l! \times \prod_{k'\in\Gamma\setminus\{k\}} \frac{k'}{k'-k} \in Z$ and $L(x) = \frac{x-1}{n}$.

E. Ledger Update

Step 1: *New block generation*. The order node updates the global model: $\omega_{i+1} = \omega_i + \lambda\nabla\omega_i$ Where the symbol λ represent as the learning rate, and ω_{i+1} is the global model parameters updated in the $(i+1)^{th}$ iteration. The order node adopts the PBFT consensus protocol ordered collected local update transactions chronologically for a pending block B_{i+1}.

Step 2: *Leger Update*. After creating a new block, the order node broadcasts it to the commit nodes in the channel for committing the transactions. committing nodes make sure that the validity of transactions(such as signature and endorsement policies) within the new block. Upon confirmation, the block is appended to the channel's blockchain and an update notification is sent to the users at the same time. The user can obtain the updated global model from the latest block for the next round of training, and then the iteration continues until the accuracy or error of the model is within the specified range.

F. Incentive Mechanism

Fabric blockchain implements permission management by embedding access control mechanisms in smart contracts. We introduces a local token named "$FLcoin$" to incentivize users to participate in the training of the shared model. $FLcoin$ is a digital currency issued by a TA, it paid in fiat currency for access to channels or rewards for users. The user earns $FLcoin$ by participating in model updates and monetize $FLcoin$ in fiat currency. The third parties who are interested in the models can make a payment to TA or users through any secure channels of disbursement. Once obtained the coins, the third party sends the access request along with the asset ID and the duration of access (in days). For example, in this case, a third party pays one $FLcoin$ for one day (24 hours) of access to the blockchain model to a user who participates in model updates during the access time. Upon receipt of an access request, TA initiates access TX and the users who submit local updates during the access time receive rewards.

IV. Security Evaluation

A. Correctness Proof

By using the Shanir Secret Sharing scheme, we construct a $t-1$ degree polynomial to share the key between order nodes. Let's consider the worst case, where only $t+1$ order nodes OP_k works, and $k \in \Gamma = \{k_1, \ldots, k_{t+1}\}$.

First, the $t+1$ order nodes decrypt c_i with its partial key pair $s_k = f(k)$ to get d_k $(1 \leq k \leq t+1)$: $d_k = {c_i}^{s_k} \bmod n^2$.

Using Lagrange interpolation formula, we can get:

$$l!f(0) = l!m\beta = \sum_{j\in\Gamma} \psi^{\Gamma}_{(0,j)} f(j) \bmod\ nm.$$

Thus

$$L(\prod_{j\in\Gamma} d_j^{2\psi^{\Gamma}_{(0,j)}} \bmod n^2) = L(c_i^{4(l!)^2 m\beta} \bmod n^2).$$

We calculate $c_i = \prod_{m=1}^{n} c_{i,m} = g^{\sum_{m=1}^{n} \nabla\omega_{i,m}} (\prod_{m=1}^{n} r_m)^n \bmod n^2$ as a substitute:

$$c_i^{4(l!)^2 m\beta} = g^{\sum_{m=1}^{n} \nabla\omega_{i,m} 4(l!)^2 m\beta} (\prod_{m=1}^{n} r_m)^{n4(l!)^2 m\beta} \bmod n^2.$$

So

$$\nabla\omega_i = \frac{1}{n}\sum_{j=1}^{n} \nabla\omega_{i,j} = L(\prod_{k\in\Gamma} d_k^{2\psi^{\Gamma}_{(0,k)}}) \times \frac{1}{4n(l!)^2\theta} \bmod n.$$

The aggregated local update can be restored with not less than t correct decryption shares.

B. Security Proof

Theorem 2: Under the assumption of decisional composite residuosity assumption(DCRS), our scheme can hold the privacy of local updates information.

Proof: Since the Paillier cryptosystem is *IND-CPA* secure [8], any information about the user's local update is not available to an honest but curious participant based on the user's local update ciphertext recorded on the blockchain.

Assuming the existence of a malicious adversary $\mathcal{A}$ with a non-negligible advantage to break the privacy of local updates, in the following security game, we describe an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the semantic security of the original Paillier cryptosystem. In the find stage, after obtaining the public key, $\mathcal{B}$ selects two messages M_0 and M_1 which are sent to an encryption oracle. The encryption oracle randomly selects a bit b to return a ciphertext c of M_b. In the guessing stage, $\mathcal{B}$ attempts to guess which message is encrypted. The interaction between $\mathcal{A}$ and $\mathcal{A}$ is as follows:

Init: $\mathcal{A}$ selects to corrupt t order nodes, we assume that the first t order nodes are corrupted. $\mathcal{A}$ can learn all the secret information of the corrupted order nodes.

Setup: $\mathcal{B}$ first gets the public key $PK = (n, g)$ of $\Sigma_{paillier}$ and chooses $(a_1, b_1, \theta) \in Z_n^* \times Z_n^* \times Z_n^*$ in random, then he sets $g_1 = g^{a_1} \times b_1^n \bmod n^2$.The last equation will be sent to the opponent $\mathcal{A}$. At the same time, $\mathcal{B}$ randomly picks $s_1, s_2, .., s_t$ from $\{0, .., [n^2/4]\}$ and sends $(n, g_1, s_1, s_2, .., s_t)$ to $\mathcal{A}$.

Phase1: $\mathcal{A}$ chooses a message M and sends it to B. he computes $c = g_1^M x^n \bmod n^2$ which is a valid encryption of M. The decryption shares of the corrupted order nodes can be computed: $d_i = c^{2\Delta s_i} \bmod n^2$, $i = 1, \ldots, t$. Other decryption shares can be obtained by interpolation:

$$d_i = (1 + 2M\theta n)^{\psi^S_{(i,0)}} \times \prod_{j\in S\setminus\{0\}} c^{2s_i\psi^S_{(i,j)}} \bmod n^2$$

Challenge: $\mathcal{A}$ selects and sends two messages M_0 and M_1 to $\mathcal{B}$. $\mathcal{B}$ sends M_0 or M_1 to the encryption Oracle as an input of the find phase. The encryption oracle selects a random bit $v \in \{0, 1\}$ to return a ciphertext c of M_v. $\mathcal{B}$ calculates $c' = c^{a_1} \bmod n^2$ which will be sent to $\mathcal{A}$.

Phase2: Repeated Phase1 in which $\mathcal{A}$ queries $\mathcal{B}$ for the decryption shares of the ciphertext of the selected message.

Guess: $\mathcal{A}$ submits a guess $v' \in \{0, 1\}$ and $\mathcal{B}$ sends v' as the answer in the guess phase. Therefore, $\mathcal{B}$ attacking the semantic secure of Paillier cryptosystem has the same advantages as A winning the game:

$$Adv_B = \frac{1}{2}(\Pr[v = v']) - \frac{1}{2} = \varepsilon.$$

According to literature [8], the definition of semantic security of Paillier cryptosystem under DCRS does not exist. So there is no opponent $\mathcal{A}$ can win the game with a nonnegligible probability ε.

C. Security Analysis

1) *Anti-collusion attack*. When the collusion has a t smaller number of collusion nodes, the private key combined cannot be interpolated by polynomials, and the collusion will not be able to recover any locally updated information.
2) *Anti-sybil attack*. Hyperledger Fabric allows specific users to join specific channels. Based on the Fabric MSP, malicious attackers cannot create multiple identities to increase their impact on model updates.
3) *Anti-poisoning attack*. Each local update data is sent to endorsing nodes for validation and evaluation, and

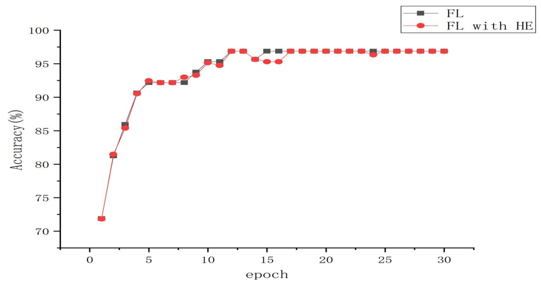

Fig. 3. The effect of homomorphic encryption of model accuracys.

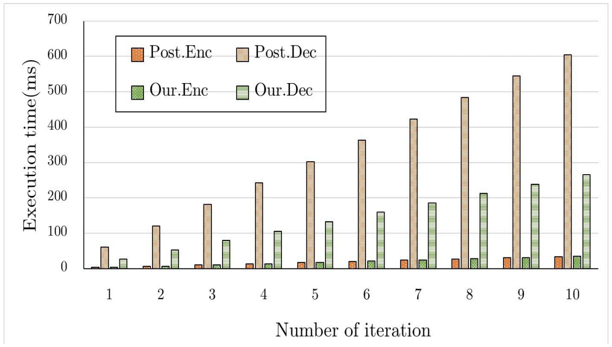

Fig. 4. Time cost comparison of cryptographic algorithm with poster [13].

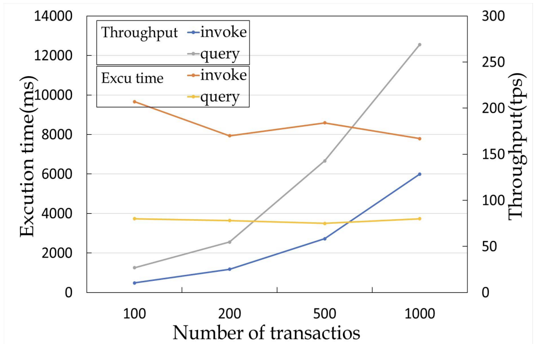

Fig. 5. Performance of invoke and query from/to fabric ledgers under different numbers of transactions submitted).

unqualified local update data are rejected for global updates.

4) *Anti-inference attack*. Local updates are broadcast in ciphertext to the blockchain network via the cryptographic system and recorded in the ledger.

V. Performance Evaluation

We realized our scheme in our laptop with a 2.10GHz CPU and 16GB configuration based on Ubuntu 20.04.1 LTS. We use PyTorch as the underlying machine learning training library. The original logistic regression algorithm are used for federated learning on breast cancer dataset [12], and then we analyze the difference of the experimental results. The impact of homomorphism encryption on the accuracy of the model is almost negligible as can be seen from Fig.3. In addition, we compare the encryption and decryption time between our scheme and the Poster scheme [13], both of them use a homomorphic encryption algorithm. The comparison results are shown in Fig.4. It can be seen that our cryptographic algorithm is relatively efficient since Poster uses the proxy re-encryption algorithm to ensure the privacy of local updates instead of the Shamir secret sharing algorithm.

A blockchain system is employed based on Fabric v2.0.0 platform to evaluate overall system performance. We analyze the execution cost and throughput(number of successful transactions per second) by changing the transactions up to 1000. The experimental results on distinct numbers of transactions have been demonstrated in Fig.5. The above results indicate the feasibility and effectiveness of our method.

VI. Conclusion

In this work, a novel federated learning architecture is proposed using permissioned blockchain and threshold homomorphic encryption technologies for secure exchange and aggregation of data. No participant can obtain the users' local data information from the records on the blockchain. We address both security and privacy issues in the blockchain-based federated learning system with meticulous design, which is not achieved in the existing schemes. The simulation results show that the scheme is feasible and efficient.

References

[1] R. Lu, W. Zhang, Q. Li, X. Zhong, and A. V. Vasilakos, "Auction based clustered federated learning in mobile edge computing system," 2021, *arXiv:2103.07150*. [Online]. Available: https://arxiv.org/abs/2103.07150

[2] J. Sun, X. Yao, S. Wang, and Y. Wu, "Non-repudiation storage and access control scheme of insurance data based on blockchain in IPFS," *IEEE Access*, vol. 8, pp. 155145–155155, 2020.

[3] S. Wang, D. Zhang, and Y. Zhang, "Blockchain-based personal health records sharing scheme with data integrity verifiable," *IEEE Access*, vol. 7, pp. 102887–102901, 2019.

[4] B. Bera, S. Saha, A. K. Das, and A. V. Vasilakos, "Designing blockchain-based access control protocol in IoT-enabled smart-grid system," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5744–5761, Apr. 2021.

[5] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Computer Security—ESORICS 2020* (Lecture Notes in Computer Science), vol. 12308, L. Chen, N. Li, K. Liang, and S. Schneider, Eds. Cham, Switzerland: Springer, 2020, pp. 480–501.

[6] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Depend. Sec. Comput.*, vol. 18, no. 5, pp. 2438–2455, Oct. 2021.

[7] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4177–4186, Jun. 2020.

[8] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 1592, J. Stern, Ed. Berlin, Germany: Springer, 1999, pp. 223–238.

[9] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[10] X. Xu, G. Sun, L. Luo, H. Cao, H. Yu, and A. V. Vasilakos, "Latency performance modeling and analysis for hyperledger fabric blockchain network," *Inf. Process. Manage.*, vol. 58, no. 1, Jan. 2021, Art. no. 102436.

[11] K. Wei *et al.*, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3454–3469, Apr. 2020.

[12] M. Lichman, "UCI machine learning repository," School Inf. Comput. Sci., Univ. California, Irvine, Tech. Rep., 2007. [Online]. Available: https://archive.ics.uci.edu/ml/citation_policy.html

[13] S. Awan, F. Li, B. Luo, and M. Liu, "Poster: A reliable and accountable privacy-preserving federated learning framework using the blockchain," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 2561–2563.