IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 11, 2021, accepted November 7, 2021. Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2021.3128622

ST-BFL: A Structured Transparency Empowered Cross-Silo Federated Learning on the Blockchain Framework

UMER MAJEED[1], LATIF U. KHAN[1], ABDULLAH YOUSAFZAI[1,2], ZHU HAN[1,3], (Fellow, IEEE), BANG JU PARK[4], AND CHOONG SEON HONG[1], (Senior Member, IEEE)

[1]Department of Computer Science and Engineering, Kyung Hee University, Yongin 17104, South Korea
[2]School of Systems and Technology, University of Management and Technology, Lahore 54770, Pakistan
[3]Electrical and Computer Engineering Department, University of Houston, Houston, TX 77004, USA
[4]Department of Electronic Engineering, Gachon University, Seongnam 13120, South Korea

Corresponding author: Choong Seon Hong (cshong@khu.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant by the Korean Government through the MSIT (Evolvable Deep Learning Model Generation Platform for Edge Computing) under Grant 2019-0-01287, in part by the Korea Institute of Energy Technology Evaluation and Planning (KETEP), and in part by the Ministry of Trade, Industry and Energy (MOTIE) of the Republic of Korea under Grant 20209810400030.

ABSTRACT Federated Learning (FL) relies on on-device training to avoid the migration of devices' data to a centralized server to address privacy leakage. Moreover, FL is feasible for scenarios (e.g., autonomous cars) where an enormous amount of data is generated every day. Transferring only local model updates in the case of FL is highly communication-efficient compared to transferring all data in the case of centralized machine learning (ML). Although FL offers many advantages, it also has some challenges. A malicious aggregation server can infer device information via local model updates. Another downside of FL is the centralized aggregation server that can malfunction due to an attack or physical damage. To address these issues, we propose a novel Structured Transparency empowered cross-silo Federated Learning on the Blockchain (ST-BFL) framework. In ST-BFL, homomorphic encryption, FL-aggregators, FL-verifiers, and smart contract are employed, which satisfy various structured transparency components, such as input privacy, output privacy, output verification, and flow governance. We present the framework architecture, algorithms, and sequence diagram of our ST-BFL framework to show how different entities interact in ST-BFL for the FL process. We also present a simplified class diagram of ST-BFL's smart contract for an FL task. Finally, we perform a simulation to analyze our framework from the perspective of aggregation time, accuracy, and storage size. The qualitative and quantitative evaluation shows that ST-BFL has the same accuracy as traditional FL. However, ST-BFL provides input privacy, output privacy, input verification, output verification, and flow governance at the expense of relatively higher computation and communication costs than traditional FL.

INDEX TERMS Blockchain, Ethereum, federated learning, flow governance, homomorphic encryption, input privacy, input verification, output privacy, output verification, smart contract, structured transparency.

I. INTRODUCTION

The rapid proliferation and advances in communication and information technologies generate data at diverse variety, high velocity and tremendous volume [1]. To fully reap the data for various applications, one can feed the generated data to Machine Learning (ML) algorithms [2], [3]. However, this involves privacy-transparency tradeoffs, such as the privacy dilemma and the transparency dilemma [4]. *Privacy dilemma* refers to whether or not to disclose information for accomplishing a certain goal. Disclosing information may lead to potential harm (such as copy problem) by malicious actors [4]. A *transparency dilemma* occurs when someone has to make a decision without access to complete information about the decision. In other words, a transparency dilemma occurs when someone is forced to make a critical decision under limited knowledge of the predicament under consideration. Providing more access to

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

the data for making a decision may lead to privacy leakage. Therefore, a tradeoff must be made between transparency (information disclosing) and privacy. Private and secure ML schemes aim to solve the privacy dilemma, copy problem, and transparency dilemma for artificial intelligence-based applications. Numerous techniques related to secure and private-AI, have been developed to overcome these privacy-transparency trade-offs. One such technique is Federated Learning.

Federated learning (FL) [5]–[9] is a collaborative learning paradigm where an ML model is learned over geographically dispersed devices without sending devices' raw data to the centralized server for training. The privacy of the raw data remains intact up to a certain level, and devices only send the local learning model updates to the central server, which are aggregated to yield the global model. FL has brought a revolution in the scope and volume of artificial intelligence-based applications and services that can be enabled using ML and deep learning. However, there are still many challenges to enabling privacy-preserving and secure FL such as:

- *Single-point-of-failure:* The typical FL process involves a single central server as an aggregator that can suffer from a single-pint-of-failure issue due to security attack or physical damage.
- *Poisoning attacks:* Malicious devices may send poisoned local model updates to the aggregation server that will lead to high global convergence time or abort the learning process.
- *Inference attacks:* FL itself is not a complete data privacy solution. An adversary may exploit the local model updates of devices for performing inference attacks. Additionally, the aggregation server itself can also infer the devices' information from the local learning model updates.
- *Aggregation attacks:* The aggregation server may be attacked by a malicious adversary to compute the aggregated global model incorrectly. This results in an inaccurate global model. and will prolong the FL global model convergence time.
- *Auditability:* Since the FL process can be subject to poisoning attacks, inference attacks, and other malicious attempts by adversaries. To improve the security of the FL process, an audit of the FL process is necessary. Auditability of the FL process can be achieved by recording all the meta-data related to local model updates, global model aggregation process, as well as accessibility information in a secure log.

Motivated by the above challenges in FL, we design a novel framework, entitled, Structured Transparency empowered cross-silo Federated Learning on the Blockchain (ST-BFL) framework. *Structured transparency* [4], [10] is a framework to solve privacy-transparency tradeoffs for information flows systematically. Structured transparency has five components, namely input privacy, output privacy, input verification, output verification, and flow governance. Structured transparency allows us to make effective use of information by preventing potential misuse. Structured transparency ensures input data autonomy by ensuring input privacy, the computational process' integrity, and transparent-auditable flow governance. In this paper, We analyze and empower FL in the context of structured transparency. In this regard, the ST-BFL framework employs several techniques to satisfy different components of structured transparency. We believe that this study is the first to empower FL using structured transparency.

Our contributions are summarized as follows:

- We present a structured transparency perspective on federated learning by considering the federated learning process as 'information flow' in a structured transparency context.
- We propose the ST-BFL framework that employs homomorphic encryption at the client layer, FL-aggregation at the FL-aggregation layer, FL-verification at the FL-verification layer, and smart contract & IPFS at the distributed decentralized layer to satisfy different components of structured transparency.
- We perform a quantitative and qualitative evaluation of our framework. It is shown that ST-BFL has the same accuracy as traditional FL. However, ST-BFL provides more benefits, such as input privacy, output privacy, output verification, and flow governance.

The rest of the paper is organized as follows: Section II analyzes the literature review of blockchain enabled FL. Section III discuss the preliminaries related to our work. Section IV presents the proposed framework. Section V discuss the simulation setup, quantitative evaluation, and qualitative evaluation of our proposed framework. Section VI concludes our paper.

II. RELATED WORK

Few works [7], [11], [12] considered blockchain for FL to enable robust learning by avoiding the use of a centralized aggregation server. Kim *et al.* [11] proposed BlockFL, which is blockchain-based FL architecture with an on-device aggregation mechanism. The local model updates are conducted by user devices on their respective local data samples. The local model updates are then sent to miners, who then accumulate them to form a new block for the current global iteration. The formed block is propagated to the network for consensus and added to an immutable ledger. The main downside of the design is that the aggregation of local models to obtain a global model is performed on-device (using the FedAvg algorithm [13]) by each participating device. The on-device aggregation is infeasible due to storage, latency, and computing cost for computing the global model from a block of local models with millions or billions of parameters. Mobile devices are heterogeneous in computing, latency, and storage capabilities. Expecting every device to be able to do on-device aggregation is impractical. Another issue is the on-chain storage of local models. The block size does not

allow on-chain storage of models having a model size in megabytes (MBs). Moreover, they assume that no straggling devices are participating in the FL process, which is absurd.

Majeed and Hong [7] proposed a blockchain-based FL scheme called FLchain, which supports learning of multiple global models for different tasks by leveraging channels in a distributed ledger. Each global-model learning task is allocated a separate channel. They introduced the ''Global Model State Trie'', similar to ''Account State Trie'' in Ethereum, to store aggregated models in a Merkle tree. However, the drawback of the proposed scheme is the lack of any validation mechanism for the computed global model by miners and the lack of any incentive mechanism. Moreover, this scheme also has an on-chain storage issue. Li *et al.* [12] proposed BFLC, a blockchain-enabled federated learning framework with committee consensus. This framework uses blockchain for local model exchange and also has on-chain global model storage. The novel committee consensus requires fewer computing resources and is robust to malicious attacks such as poisoning attacks, but the downside of their approach is that each FL task requires an independent dedicated blockchain.

Federated learning is not a fully-privacy preserving scheme. Researchers use homomorphic encryption with federated learning to mitigate membership inference attacks by the central server. A fully homomorphic encryption scheme was first presented by Gentry [14]. A comprehensive survey on homomorphic encryption is presented by Acar *et al.* [15]. Zhang *et al.* [16] proposed BatchCrypt, a framework for communication efficient homomorphic encryption for cross-silo FL. However, BatchCrypt has significant computation overhead for pre-processing and post-processing. BatchCrypt encodes quantized gradients in a batch in the form of a single long integer. This reduces the computation required for homomorphic encryption and decryption, but pre-processing and post-processing overhead at the local devices restrains BatchCrypt usage. Moreover, asymmetric quantization and de-quantization procedures involved may reduce global accuracy [17].

The works in [7], [11], [12] considered blockchain and FL. However, they did not effectively consider privacy preservation and global model verification at the aggregation stage. Meanwhile, the work in [16] studied communication-efficient privacy preservation for FL only. By contrast, our work considers privacy preservation, global model verification at the aggregation stage, removal of single-point of failure of the central aggregation server, off-chain aggregation, off-chain storage, and effective flow governance.

III. PRELIMINARIES

This section gives a brief introduction to the technologies and platforms which helped in designing the framework architecture of ST-BFL or helped in the implementation and evaluation of ST-BFL.

A. ETHEREUM

Ethereum [18] is a popular blockchain platform for deploying decentralized applications. Ethereum is the protocol envisioned and developed by Vitalik Buterin in 2013 and 2015, respectively. Ethereum is widely known as the ''World Computer'' and provides a universal programmable blockchain ecosystem. Ethereum runs decentralized applications (DApps) by executing smart contracts on Ethereum's distributed network. Each participant in the network has a unique private key associated with their unique Ethereum Address (EA) for signing transactions [19]. The private key is stored in the Ethereum Wallet.

B. SMART CONTRACT

The concept of smart contract was devised by Szabo in [20], [21]. Smart contract's original idea was to embed contractual clauses using hardware and software so that breaching the contract is effectively detrimental to the breaching party in the agreement [22]. The introduction of smart contracts in the blockchain is considered a breakthrough for the implementation of a broad scope of applications [23]. Smart contracts on Ethereum [24] are usually written in a high-level programming language known as Solidity. Smart contracts are immutable and, once deployed, cannot be changed. The execution of smart contracts on Ethereum consumes gas. Each smart contract is allotted a unique contract account address for deployment on Ethereum. The smart contract is executed on each Ethereum full node and the state variables of the smart contract are stored in account storage trie associated with the contract account on every full node [25].

C. SMART CONTRACT STORAGE

Each smart contract is assigned permanent storage by the Ethereum Virtual Machine (EVM) to maintain its state. Smart contract storage can extend up to an addressable space of 2^{256} slots, while each slot contains 32-bytes. A smart contract can read and write from any location of its persistent storage. The state of a smart contract is determined by contract-level variables, also known as ''state variables'' [26]. The cardinality and structure of state variables remain the same throughout the smart contract life cycle. However, the content of state variables may be updated through transactions [27]. The state variables are stored as key/value pairs mapping 32-byte keys to 32-byte values. The smart contract also uses 256-bit-addressable space called memory for transient storage of intermediate values during computations.

D. SOLIDITY

Solidity is a high-level Turing-complete general-purpose programming language for writing smart contracts. Solidity is an object-oriented programming-language that supports events, modifiers, multiple inheritance, and control structures. Solidity has its compiler to generate the bytecode (a low-level language) that runs on the Ethereum Virtual

Machine (EVM). Solidity is the official and most popular smart-contract language on Ethereum. In Solidity, the type of each variable needs to be explicitly specified, so it is a statically typed language [28].

E. REMIX IDE

Remix [29] is an integrated development environment (IDE), which allows to write, compile, debug, test, deploy and run Solidity smart contracts from the ease of an online web browser. It has a source code editor with an auto-complete option and a file manager. Once a smart contract is compiled in Remix, the bytecode and the application binary interface (ABI) are available to download. A JavaScript VM is provided to interact with the smart contract in the default mode. Five accounts are provided, each with a 3,000,000 gas limit.

F. TRUFFLE

Truffle [30] is the command-line interface (CLI) framework for coding, compiling, building, migrating, deploying, and testing Ethereum decentralized applications (DApps). Though decentralized applications can be developed without Truffle, Truffle accelerates the development process. Truffle allows seamless integration of web technologies with smart contracts to produce complete DApps as an end-product. Compiling of smart contracts on truffle results in bytecode, which is deployed on Ethereum as a smart contract. Truffle can interact with Geth, Parity, and Ganache seamlessly.

G. GANACHE

Ganache [31], [32] is a tool to create a local private ethereum blockchain for development and testing purposes. Ganache GUI has an interface to list available accounts, transactions, deployed smart contracts, and emitted events. From Ganache, a blockchain developer can also access a particular transaction's data as well as the storage of a smart contract. Blockchain developers use Ganache for prototyping in the development environment before deploying to the mainnet or testnets. The read/write transactions can be submitted to Ganache through a JSON RPC API server. As soon as transactions are received, Ganache processes them instantly and updates the corresponding state. Therefore, Ganache is 10 times faster than Ethereum testnets.

H. InterPlanetary FILE SYSTEM

InterPlanetary File System (IPFS) [33] is a open-source peer-to-peer universal content-addressed file storage and sharing system. IPFS platform [34] is a hypermedia protocol, which assists the web in becoming safer, faster, decentralized, and open. IPFS stores the file in an immutable and persistent way. However, versioning is supported. IPFS assigns a global unique cryptographic hash for each file stored on it. The files having exactly the same content have the same digital fingerprint (hash), thus reducing redundancy in the network. This IPFS hash can be used to retrieve the immutable file later. Asymmetric Cryptography can be applied to encrypt/decrypt files for secure sharing. Since blockchains usually have limited storage capacity, IPFS has become popular for high-throughput large data storage for decentralized applications [35]. IPFS uses a distributed hash table (DHT) to track its data. The IPFS hash of a file is stored on Ethereum smart contracts link Ethereum and IPFS, as Ethereum itself does not provide any mechanism to connect to the external world.

I. HOMOMORPHIC ENCRYPTION

Homomorphic Encryption [36] is a way to perform computation on encrypted data such that the results are also encrypted. For an encryption scheme (P, C, K, E, D), where the plain texts are P and C, E is the encryption algorithm, D is the decryption algorithm, and K is the keyspace. We assume that the plain texts belong to group $(P, \lozenge)$, while the ciphertext belong to group $(C, \circ)$. The ecryption maps plain text $P \in (P, \lozenge)$ to ciphertext $C \in (C, \circ)$ such that $E_k : P \rightarrow C$ where $k \in K$ is secret key. If

$$E_k(a) \circ E_k(b) = E_k(a \lozenge b), \quad \forall a, b \in P, \ \forall k \in K. \tag{1}$$

then the encryption scheme E_k is homomorphic scheme [37].

Fully Homomorphic Encryption (FHE) supports the addition of ciphertexts, as well as multiplication with a scalar [37], [38].

$$E_k(s.a) = s.E_k(a), \quad \forall a \in P, \ \forall s \in R_+, \ \forall k \in K. \tag{2}$$

Homomorphic Encryption schemes are classified into two main categories based on how the involved parties encrypt their data. These are Single-Key Homomorphic Encryption (SKHE) and Multi-key Homomorphic Encryption (MKHE). In SKHE, there is a single pair of secret-public keys [39]. The encryption and evaluation can be generally performed using the public key. However, the decryption can only be performed using the secret key. The typical cross-silo FL schemes involving SKHE rely on the honesty of the key-generator who holds the secret key. To address these trust issues, MKHE is the solution. In MKHE, each party encrypts their own plain text using their own different secret key. MKHE allows operations on ciphertext using the joint public context [40]. The resultant ciphertext is then jointly decrypted by the involved parties. In this way, no party can decrypt the other party's individual original ciphertext. However, MKHE schemes are still in infancy. FL requires a homomorphic scheme (SHKE or MKHE) that supports operations such as multiplication with unencrypted floats (scalar) and the addition of encrypted signed floats. As the practical implementation of MKHE is not widespread due to lack of effective and feasible libraries, we have used CKKS in ST-BFL, which is an SKHE and a public key encryption scheme [41].

J. CROSS-SILO FEDERATED LEARNING

Cross-silo FL allows few organizations to collaboratively train a ML model. In this work, we consider a collaboratively

TABLE 1. Summary of notations.

Symbols	Description
FL_Task_f	f_{th} FL task
f	index for FL task
O_F	set of organizations/Silos for FL_Task_f
N	cardinality of O_F
k	index for organization/Silos
t	global epoch
a	index for FL-Aggregators
v	index for FL-verifiers
TP_f	task publisher for FL_Task_f
$FL-Trainers_f$	set of FL trainers for FL_Task_f
$FL-Trainer_{k,f}$	k_{th} FL-Trainer for FL_Task_f
AS_f	Aggregation Scheme for FL_Task_f
MA_f	Model Architecture for FL_Task_f
IMW_f	Initial Model Weights for FL_Task_f
$SKHE_{f,t}^{Pri}$	Private context of SKHE for global iteration t and FL_Task_f
$SKHE_{f,t}^{Pub}$	Public context of SKHE for global iteration t and FL_Task_f
$ELMW_{k,f,t}$	Encrypted Local Model Weights of k_{th} FL-trainer at global iteration t of FL_Task_f
$EGM_{f,t}$	Encrypted Global Model at global iteration t of FL_Task_f
$DGM_{f,t}$	Decrypted Global Model at global iteration t of FL_Task_f
NV_{verify}	Number of verifiers who voted that the FL-aggregator has aggregated correctly
$NV_{select,f,t}$	Number of verifiers assigned at global iteration t of FL_Task_f
$NV_{threshold,f,t}$	Number of verifiers required to vote in support of FL-aggregator at global iteration t of FL_Task_f to confirm epoch

trained NN (deep Dense Neural Network (DNN) or Convolution Neural Network (CNN)) in a federated manner using the supervised learning approach from the distributed labeled datasets available to each organization. Formally, consider a cross-silo FL task FL_Task_f with N organizations, the set of organizations is denoted as $O_F = \{O_1, O_2, \ldots, O_N\}$. Each organization has its own dataset D_k. Where, $n_k = |D_k|$ are the total number of samples in D_k, then $n = \sum_k D_k$ are the total samples for all the datasets $D_k, k \in K$. The NN has C classes for compact Euclidean feature space $\mathcal{X}$ and label space $\mathcal{Y} = [C]$, where $[C] = \{1, 2, \ldots, C\}$. The goal of FL for NN is to minimize the overall loss function [42]

$$\min_{\boldsymbol{\omega}} f(\boldsymbol{\omega}) = \sum_{k \in O} \frac{n_k}{n} F_k(\boldsymbol{\omega}),$$

where

$$F_k(\boldsymbol{\omega}) = \frac{1}{n_k} \sum_{r \in \mathcal{D}_k} f_r(\boldsymbol{\omega}). \tag{3}$$

Here, the local loss of NN is denoted by F_k. $f_r(\boldsymbol{\omega})$ denotes the multiclass cross-entropy loss on a data point $\{\boldsymbol{x}, y\}$ for one-hot-encoded labels and is defined as [43]

$$f_r(\boldsymbol{\omega}) = -\sum_{q=1}^{C} \mathbb{1}_{y=q} \log p_q(\boldsymbol{x}, \boldsymbol{\omega}). \tag{4}$$

Here, if $\boldsymbol{\omega}$ is the weight matrix of the NN, then $p_q(\boldsymbol{x}, \boldsymbol{\omega})$ gives the probability that $x \in X$ belongs to class q.

At global iteration $t + 1$, the local model weights are updated by organization k as (note: we skipped formulation for local iterations here for simplicity)

$$\boldsymbol{\omega}_{t+1}^k \leftarrow \boldsymbol{\omega}_t - \eta g_k, \forall k, \quad \text{where} \quad g_k = \nabla F_k(\boldsymbol{\omega}_t). \tag{5}$$

At global iteration $t + 1$, the global model weights are updated as [44]

$$\boldsymbol{\omega}_{t+1} \leftarrow \sum_{k \in O} \frac{n_k}{n} \boldsymbol{\omega}_{t+1}^k. \tag{6}$$

For $\sigma_k = \frac{n_k}{n}$, (6) becomes

$$\boldsymbol{\omega}_{t+1} \leftarrow \sum_{k \in O} \sigma_k \boldsymbol{\omega}_{t+1}^k, \tag{7}$$

where σ_k is the weight scaling factor for organization k.

IV. PROPOSED FRAMEWORK

In this section, we describe our proposed ST-BFL framework that is a structured transparency empowered cross-silo federated learning on the blockchain. ST-BFL utilizes homomorphic encryption, FL-aggregation, FL-verification, and smart contract to empower federated learning with structured transparency.

Fig. 1 shows the layered architecture of ST-BFL that consists of four layers: the client layer, the distributed decentralized layer, the FL-aggregation layer, and the FL-verification layer. Additionally, the framework has a blockchain oracle service for each FL-task, and the ST-BFL market service manager. Each FL_Task_f has its own client layer. The client layer consists of FL trainers (silos) taking part in the FL process for a FL_Task_f, and a FL-task publisher TP_f. The k^{th} FL trainer is denoted by $FL-Trainer_{k,f}$, where $k \in [0, N]$ and N denotes total number of cross-silo trainers in FL_Task_f. The summary of notations is given in Table 1. Each $FL-Trainer_{k,f}$ has the dataset $D_{k,f}$. The distributed decentralized layer consists of the Ethereum blockchain network, and a decentralized file storage system known as IPFS. The IPFS and Ethereum blockchain are formed by a network of nodes distributed over the globe. Moreover, these nodes are not controlled by a single entity, and thus we term this layer as distributed decentralized layer. A smart contract SM_f orchestrating the FL process is deployed on the Ethereum blockchain network for each FL_Task_f by the corresponding FL-task publisher TP_f. The *FL-aggregation layer* consists of servers SA_a, where $a \in [0, A]$, which offers the aggregation service for FL tasks. A is the total number of FL-aggregator servers registered in the cross-silo

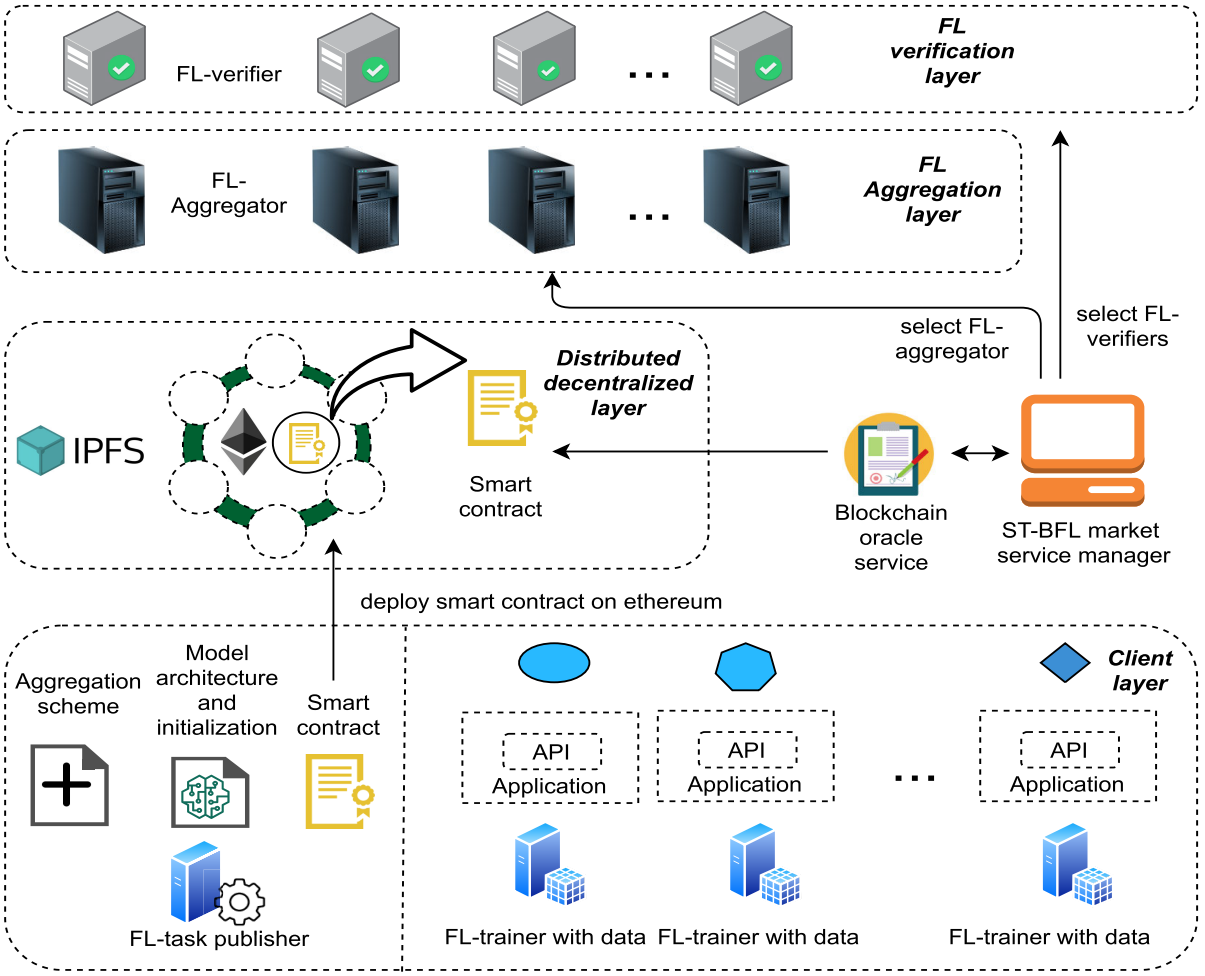


FIGURE 1. Layered architecture of ST-BFL framework.

FL market. The *FL-verification layer* consists of servers SV_v, where $v \in [0, V]$, which offer verification service for FL tasks, i.e., they will verify that the FL-aggregator assigned to an FL-task has performed the aggregation in accordance with the corresponding aggregation scheme. V is the total number of FL-verifiers servers registered in the cross-silo FL market.

The FL-aggregation layer and FL-verification layer are considered as a part of the ST-BFL marketplace. The ST-BFL market service manager selects and manages the FL-aggregators and FL-verifiers in the FL-aggregation layer and FL-verification layer. The FL-task publisher registers the FL-task to the ST-BFL market service manager. The FL-aggregator and FL-verifiers for the aggregation process at each global epoch are selected and assigned by the ST-BFL market service manager for the FL task. A blockchain oracle service [45] for the FL-task is also deployed to interact with the ST-BFL market service manager and the corresponding smart contract deployed on the Ethereum blockchain. The mapping of structured transparency components to our proposed framework is explained in Table 2. The related benefits to federated learning in ST-BFL are also listed. Next, we discuss the challenges related to the implementation of our framework.

- *Transaction data size:* In most of the popular blockchain platforms, there is an upper limit to the transaction's data size supported. Ethereum has a transaction size upper limit of 128KB. Also, storage of the whole model on the ethereum network is expensive, So, we stored the model on IPFS and instead stored the hash on Ethereum smart contract.
- *IPFS public accessibility:* The files on IPFS are publicly accessible provided you have some identifier for the file. To overcome this privacy issue, we can employ public-key cryptography to encrypt the privacy-sensitive content such as HE public contexts, decrypted global models. The encrypted contents of the file can only be decrypted using the corresponding private key generated for the corresponding global iteration. The private key can only be shared with relevant entities. However, for simplicity, we skip this cryptographic process in our work.

TABLE 2. Structured transparency mapping to ST-BFL framework, and related benefits to FL.

Structured Transparency Component	Description	How structured transparency component is implemented in ST-BFL framework ?	Benefit to FL in ST-BFL framework
Input Privacy	• The input to information flow does not leak privacy themselves.	• Input privacy is achieved by homomorphic encryption. • All the local models are encrypted using SKHE. • The task publisher holds the SKHE secret key, but the smart contract does not allow the task publisher to access the local models directly, thus preserving privacy from task publisher too.	• Prevents inference attacks on local models.
Output Privacy	• The output of information flow does not leak privacy.	• Output privacy is achieved as the aggregation process results in a homomorphic encryption global model. • The global model can only be decrypted by the task publisher.	• Prevents unauthorized access to the global model.
Input Verification	• The inputs to the information flow are authenticated and correct.	• Input Verification is not needed in cross-silo FL as we consider FL Trainers to be honest • Input Privacy and Input Verification is difficult to achieve simultaneously under current research.	-
Output Verification	• The output of the information flow is processed and generated according to the policy.	• The output of the ST-BFL framework is generated by the corresponding FL-aggregator for each epoch. • The output verification is achieved through FL-verifiers which vote on whether the FL-aggregator has aggregated the local models correctly.	• Prevents aggregation attacks when computing global model.
Flow Governance	• The Flow Governance deals with the process of how the flow is orchestrated. • who is authorized to change the flow process? • How the flow can be changed.	• Flow Governance is maintained initially by the FL task publisher. • Once the smart contract for FL task is deployed, Flow Governance is maintained by the ST-BFL market and smart contract.	• Remove single-point-of-failure of central server. • Provides auditability. • Prevent unauthorized access to local and global models.

- *Poisoning attacks:* Since the local models are encrypted using homomorphic encryption, it is difficult to verify the local models for poisoning attacks.

To realize ST-BFL, we made the following assumptions: first, since the FL-verifiers are randomly selected from the market based on reputation and capability, they have to verify the FL-aggregator in a limited time. It is assumed that they do not collude with the FL-aggregator. Second, it is assumed that the majority of FL-verifiers are honest, i.e., they perform the task as per specifications of FL information flow. Otherwise, we will get into *recursive oversight problem* [4], which is difficult to resolve. Third, in this framework, we only consider the cross-silo FL. In the cross-silo FL scheme, there is no incentive to send the poisonous local models by the FL-trainers as the trainers do not get any reward. Instead, their incentive is to jointly train the global model with high accuracy for deployment in their business. So we assume that there are no poisoning attacks by the trainers. Fourth, it is assumed that FL-aggregators and FL-verifiers do not leak the privacy of local models of an FL-trainer to FL-task publisher or other FL-trainers. Since for ST-BFL, the FL-task publisher or other FL-trainers can decrypt the local model. This problem can be resolved by using MKHE. Fifth, the cross-silo FL-Trainers and FL-task publisher have reliable and efficient communication connections as they are part of cross-silo federated learning. Sixth, the ST-BFL market service manager is reliable and secure.

Now, we discuss the sequence of the proposed ST-BFL framework as shown in Fig. 2. The sequence steps for ST-BFL are below:

- Step 1: A FL-Task publisher TP_f deploy the new smart contract on Ethereum for FL_Task_f. The agreed-upon aggregation scheme AS_f, model architecture MA_f, unencrypted initial global weights IMW_f are uploaded to IPFS and logged into the smart contract by the Cross-silo-FL-task publisher. This step is summarized in procedure *Initialize_FL_Task*() of Algorithm 1.
- Step 2: For each global epoch t of TP_f, the Task publisher TP_f generates a new SKHE private context $SKHE_{f,t}^{Pri}$ and send the associated public context to the FL trainers $FL - Trainers_f$.Both SKHE public context

Algorithm 1 FL-Task Publisher TP_f

procedure INITIALIZE_FL_TASK(k)
  Deploy Smart Contract SM_f
  Upload Aggregation Scheme $AS_f \rightarrow$ IPFS
  uploadAggregationSchemeTx() $\rightarrow SM_f$
  Upload Model Architecture $MA_f \rightarrow$IPFS
  uploadModelArchitectureTx() $\rightarrow SM_f$
  Upload Initial Model Weights $IMW_f \rightarrow$ IPFS
  uploadInitialModelWeightsTx() $\rightarrow SM_f$
  Register SM_f to ST-BFL market through ST-BFL market manager service
  The ST-BFL market manager service will deploy a corresponding blockchain oracle for the FL-task
  Make Payment to ST-BFL market
end procedure
procedure TP_INITIALIZE_FL_TASK_EPOCH(f, t)
  Generate SKHE private and public context ($SKHE^{Pri}_{f,t}$, $SKHE^{Pub}_{f,t}$)
  Upload SKHE public context $SKHE^{Pub}_{f,t} \rightarrow$ IPFS
  uploadSKHEpublicContextTx(f,t) $\rightarrow SM_f$
  Send SKHE public context $SKHE^{Pub}_{f,t} \rightarrow FL\text{-}Trainers_k$
end procedure
procedure DECRYPT_GLOBAL_MODEL(f, t)
  downloadEncryptedGlobalModelTx(f,t) $\rightarrow SM_f$
  Download encrypted global model $EGM_{f,t} \leftarrow$ IPFS
  Decrypt $EGM_{f,t}$ to decrypted global model $DGM_{f,t}$
  Upload Decrypted Global Model $DGM_{f,t} \rightarrow$ IPFS
  uploadDecryptedGlobalModelTx(f.t) $\rightarrow SM_f$
end procedure

Algorithm 2 $FL - Trainer_{k,f}$

1: **procedure** FL_TRAINER_TASK(k, f, t)
2: **if** t == 1 **then**
3: *downloadInitialModelWeightsTx()* $\rightarrow SM_f$
4: **Download** Initial Model Weights $IMW_f \leftarrow$ IPFS
5: **Train** Local model $LM_{k,f,t}$ on dataset $D_{k,f}$ using IMW_f
6: **else**
7: *downloadDecryptedGlobalModelTx(f,t-1)* $\rightarrow SM_f$
8: **Download** Decrypted Global Mode $DGM_{f,t-1} \leftarrow$ IPFS
9: **Train** Local model $LM_{k,f,t}$ on dataset $D_{k,f}$ using $DGM_{f,t-1}$
10: **end if**
11: **Encrypt** Local model $LM_{k,f,t}$ by $SKHE^{Pub}_k$
12: **Upload** Encrypted Local Model Weights $ELMW_{k,f,t} \rightarrow$ IPFS
13: *uploadLocalModelWeightsTx(t, ipfsHash($ELMW_{k,f,t}$))* $\rightarrow SM_f$
14: **end procedure**

Algorithm 3 : Smart Contract SM_f

1: **procedure** SM_FL_TASK(f, t)
2: **Set** votes $= 0$
3: **Determine** $NV_{select,f,t}$ using blockchain oracle
4: **Determine** $NV_{threshold,f,t}$ using blockchain oracle
5: **Select** FL-aggregator $SA_{f,t}$ using blockchain oracle
6: **Select** FL-verifiers $SV_{j,f,t} \in SV_{f,t}$, $|SV_{f,t}| = NV_{select,f,t}$ using blockchain oracle
7: **Collect** votes
8: **if** *votes* $>= NV_{threshold,f,t}$ **then**
9: **Confirm** $EGM_{f,t} = EGM_{f,t,a}$
10: **else if** *votes* $< NV_{threshold,f,t}$ **then**
11: **Goto** line 2 and repeat
12: **end if**
13: **end procedure**

$SKHE^{Pub}_{f,t}$ and SKHE private context $SKHE^{Pri}_{f,t}$ are uploaded to the IPFS and logged to the smart contract. However, the private context can only be accessed by the task publisher through the smart contract. The access to the public context of SKHE is provided to different entities as the FL flow progress as per policy by the smart contract. This step is summarized in procedure *TP_Initialize_FL_Task_Epoch*(k, t) of Algorithm 1.

- Step 3: For each global epoch t, each FL trainer $FL - Trainer_f$ trains the local model $LM_{k,f,t}$ using their local datasets $D_{k,f}$. If this is the first global epoch, then the local models are trained using Initial Model Weights IMW_f, otherwise, the decrypted global model of previous global iteration $DGM_{k,f,t-1}$ is used. The local models are encrypted using the public context of SKHE scheme $SKHE^{Pub}_{f,t}$ for the global epoch, and then uploaded to IPFS, afterward IPFS-hash and other meta-data are added to the transaction. The transaction is submitted to the smart contract on the blockchain ethereum) network. This step is summarized in procedure *FL_trainer_Task*(t) of Algorithm 2.
- Step 4: Meanwhile an FL-aggregator $SA_{f,t}$ and some FL-verifiers $SV_{f,t}$ are randomly selected from the ST-BFL market based on reputation and capability by the ST-BFL market service manager. The meta-data of the selected FL-aggregator $SA_{f,t}$ and FL-verifiers $SV_{f,t}$ is sent to the SM_f through the corresponding blockchain oracle. The number of selected FL-verifiers depends upon the payment to market by Task publisher TP_f. This number is defined by a threshold T_{select}). The detailed procedure for selecting FL-aggregator and FL-verifiers is out of the scope of this work.
- Step 5: After waiting for some pre-set time, The selected FL-aggregator $SA_{f,t}$ and FL-verifiers $SV_{j,f,t} \in SV_{f,t}$ downloads the aggregation scheme AS_f, model architecture MA_f, the public context of SKHE scheme $SKHE^{Pub}_{f,t}$, and all corresponding encrypted local models $ELMW_{k,f,t}$ for the current global epoch t. The encrypted local models are aggregated by FL-aggregator and

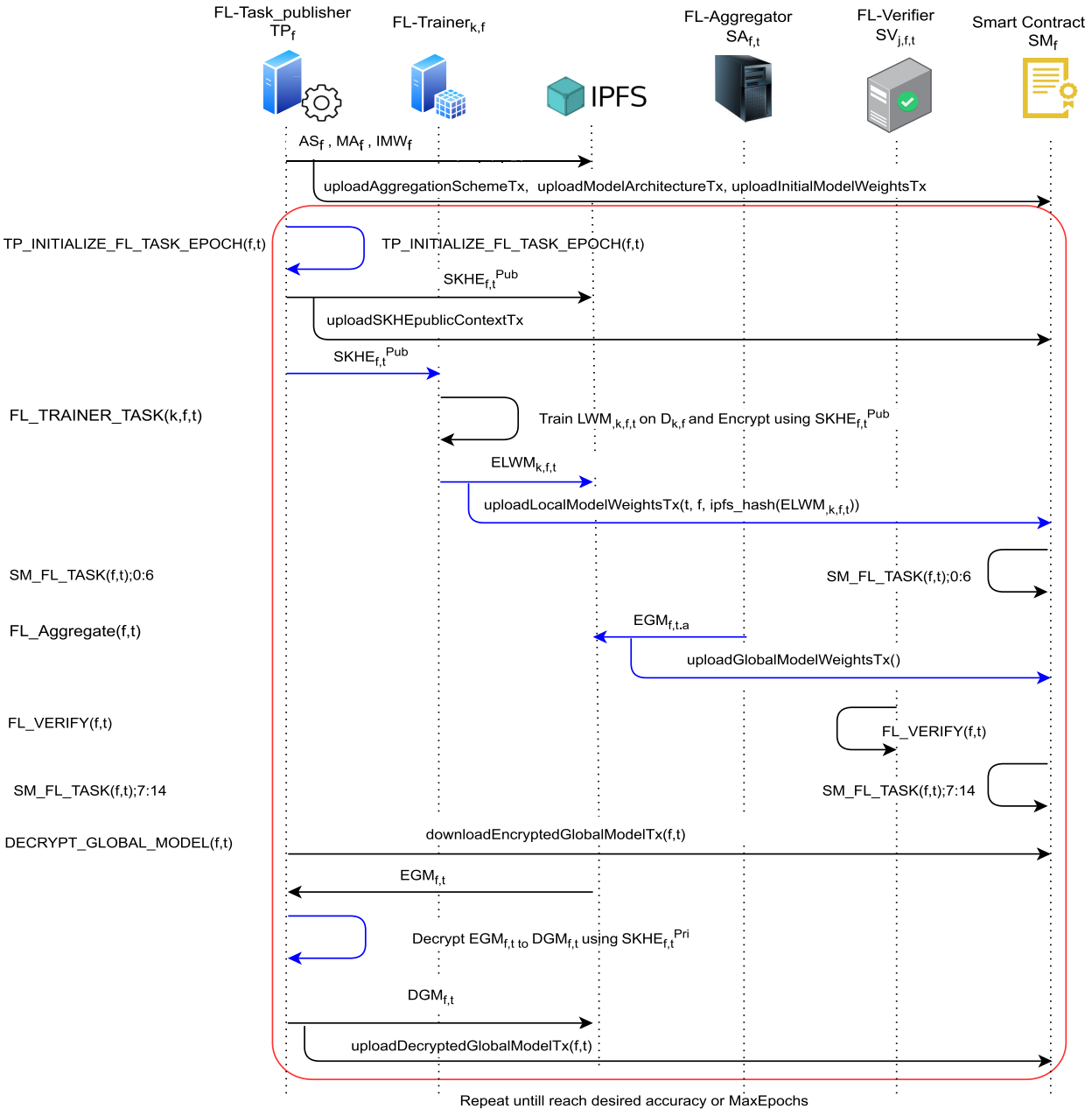


FIGURE 2. Sequence diagram of interaction between entities in ST-BFL framework.

FL-verifiers using the public context of homomorphic scheme $SKHE_{f,t}^{Pub}$. The FL-aggregator uploads the aggregated global model to the IPFS and submits a transaction. The FL-verifiers also aggregate the model and verifies that the model submitted by FL-aggregator is correct. Since the SKHE, and aggregation scheme is deterministic, both FL-aggregator and FL-verifiers should be able to generate the same aggregated global model. If $NV_{verify} >= NV_{select} * 2/3$ FL-verifiers vote that the FL-aggregator has aggregated according to the corresponding aggregation scheme, then the FL-aggregator model is termed as valid. Otherwise, step 4 is again performed by the SM_f and blockchain oracle. This ensures that the aggregation is performed as per the FL aggregation scheme AS_f.

- Step 6: Once the valid encrypted global model $EGM_{f,t}$ for global epoch t is available, it is decrypted by the Cross-silo-FL-task publisher to decrypted global model $DGM_{f,t}$. The accuracy is measured on the test dataset and logged into the smart contract. The decrypted global model is then used for the training of the next global epoch by FL-trainers.

Algorithm 4 FL-Aggregator $SA_{f,t}$

procedure FL_AGGREGATION(f,t)
  downloadAggregationSchemeTx(f) $\rightarrow SM_f$
  Download Aggregation Scheme $AS_f \leftarrow$ IPFS
  downloadModelArchitectureTx(f) $\rightarrow SM_f$
  Download Model Architecture $MA_f \leftarrow$ IPFS
  downloadSKHEpublicContextTx(f,t) $\rightarrow SM_f$
  Download SKHE public context $SKHE_{f,t}^{Pub} \leftarrow$ IPFS
  downloadLocalModelsTx(f,t) $\rightarrow SM_f$
  Download Local Models $ELMW_{k,f,t}, \forall k \leftarrow$ IPFS
  Aggregate to Encrypted Global Model Weights $EGM_{f,t,a}$ using AS_f, MA_f, and $SKHE_{f,t}^{Pub}$
  Upload Global Model Weights $EGM_{f,t,a} \rightarrow$ IPFS
  uploadGlobalModelWeightsTx() to SM_f
end procedure

Algorithm 5 FL-Verifier $SV_{j,f,t}$

procedure FL_VERIFICATION(f,t)
  downloadAggregationSchemeTx(f) $\rightarrow SM_f$
  Download Aggregation Scheme $AS_f \leftarrow$ IPFS
  downloadModelArchitectureTx(f) $\rightarrow SM_f$
  Download Model Architecture $MA_f \leftarrow$ IPFS
  downloadSKHEpublicContextTx(f,t) $\rightarrow SM_f$
  Download SKHE public context $SKHE_{f,t}^{Pub} \leftarrow$ IPFS
  downloadLocalModelsTx(f,t) $\rightarrow SM_f$
  Download Local Models $ELMW_{k,f,t}, \forall k \leftarrow$ IPFS
  Aggregate to Encrypted Global Model-verifier $EGMV_{j,k,t}$ using AS_f, MA_f, and $SKHE_{f,t}^{Pub}$
  if $ipfsHash(EGMV_{j,k,t}) == ipfsHash(EGM_{f,t,a})$ **then**
    $votes = +1 \rightarrow SM_f$
  end if
end procedure

The process then continues from steps 2-6 until some threshold in the context of accuracy or number of epochs is reached. It is necessary to mention that since the local models in ST-BFL are encrypted using the same private key by all FL-trainers, the smart contract is designed in such a way that it denies access to IPFS-hash of the encrypted local model to FL-trainers other than its generator. Only assigned FL-aggregator, FL-verifiers, and the respective FL-trainer (owner/generator) can access the IPFS-hash of the local model from the smart contract. Even TP_f cannot access the IPFS-hash of encrypted local models from the smart contract to preserve privacy.

A simplified UML diagram for the smart contract SM_f is shown in Fig. 6 in Appendix. The transactions to smart contract behave differently based on the address of the caller and so provide different data to different callers in return, thus this property of smart contract preserves the output privacy of the flow according to access rules set in the smart contract. Moreover, the events emitted by the smart contract also do not leak sensitive information.

V. EVALUATION

A. SIMULATION SETUP

We perform simulations on Ubuntu 20.04 with Intel i3@ 3.90 GHz and 16 GB RAM.

1) PYTORCH

We used PyTorch [46] for training and aggregation of state_dicts [47] of local models to global models for each global epoch.

2) INPUT PRIVACY AND HE

Ideally, for fully secure input privacy, we should have used the multi-key homomorphic encryption scheme for better privacy. However, we could not find any feasible library for a multi-key homomorphic encryption scheme for FL as the multi-key homomorphic encryption schemes and corresponding libraries are still in infancy and are at the research phase. So we use a single-key homomorphic encryption tool called TenSEAL. TenSEAL is developed by OpenMined. TenSEAL [48] is an SKHE library that allows encrypting a vector of signed floats to a single encrypted vector ciphertext using CKKS [49] under the hood.

3) REMIX IDE

We code, debug, compiled our smart contract for TP_f on REMIX IDE.

4) BLOCKCHAIN, GANACHE, AND TRUFFLE

We set up a local Ethereum blockchain using Ganache. A solidity smart contract is deployed using Truffle by the FL-task publisher TP_f.

5) IPFS

A local IPFS client is set up using the "ipfs init" command, while the IPFS client is run using the "ipfs daemon" command.

6) DATASET

We use the MNIST and fashion-MNIST datasets for our simulation. Both datasets consist of 60000 training images and 10000 test images each. For the MNIST dataset, we sorted the dataset by image labels, then we assigned the shard of 20000 images to each of the FL-trainer with N = 3. The fashion-MNIST dataset is assigned to FL-trainers with N = 3 according to Table 3.

7) MODEL ARCHITECTURE

The DNN architecture for MNIST is shown in Table 4. The model consists of three fully connected layers. The first and second fully connected layers have relu activation with a dropout of $p = 0.2$. The third connected layer has LogSoftmax activation.

The Relu [50] activation for a scalar x is given by [51]

$$\phi(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0. \end{cases} \tag{8}$$

TABLE 3. Distribution of Fashion-MNIST for N = 3.

Numeric-Label	FL-trainer-1	FL-trainer-2	FL-trainer-3
0	5600	200	200
1	5600	200	200
2	5600	200	200
3	200	5600	200
4	200	5600	200
5	200	5600	200
6	200	200	5600
7	200	200	5600
8	200	200	5600
9	2000	2000	2000

TABLE 4. Dense-NN architecture for MNIST.

Sr	Name	Code	Value	state_dict
1	Input	-	(784,)	-
2	fc1	nn.Linear	(784,200)	fc1.weight fc1.bias
3	Relu	F.relu	-	-
4	Dropout	nn.Dropout	p=0.2	-
5	fc2	nn.Linear	(200,200)	fc2.weight fc2.bias
6	Relu	F.relu	-	-
7	Dropout	nn.Dropout	p=0.2	-
8	fc3	nn.Linear	(200,10)	fc3.weight fc3.bias
9	LogSoftmax	F.log_softmax	dim=1	-

TABLE 5. CNN architecture for Fashion-MNIST.

Sr	Name	Code	Value	activation	state_dict
1	Input	-	(1,28,28)	-	
2	conv1	nn.Conv2d	(1, 6, 5, padding=2)	F.relu	conv1.weight conv1.bias
3	Maxpool	F.MaxPool2d	(2, 2)	-	
4	conv2	nn.Conv2d	(6, 16, 5)	F.relu	conv2.weight conv2.bias
5	Maxpool	F.MaxPool2d	(2, 2)	-	
6	flatten	torch.flatten	(400, 1)	-	
7	fc1	nn.Linear	(400,120)	F.relu	fc1.weight fc1.bias
8	fc2	nn.Linear	(120,84)	F.relu	fc2.weight fc2.bias
9	fc3	nn.Linear	(84,10)	F.relu	fc3.weight fc3.bias
10	LogSoftmax	F.log_softmax	dim=1	-	

The LogSoftmax for scalar $x_i \in x$ is given by [52]

$$\text{LogSoftmax}(x_i) = \log\left(\frac{\exp(x_i)}{\sum_j \exp(x_j)}\right), \quad x_i, x_j \in x. \quad (9)$$

The CNN architecture for fashion-MNIST is shown in Table 5. The architecture is based on Lenet-5 [53].

B. AGGREGATION APPROACHES

For cross-silo FL, the weight-scaling factor in (7) as per aggregation policy for the FL task can be determined based on the number of samples in the local datasets. For unencrypted and encrypted aggregation, this leads to a total of four approaches for comparing the aggregation time.

- *encrypted-lazy approach*: The weight-scaling is performed at the aggregator on encrypted weights. This requires more computation time at the aggregator. This approach is optimal when the weight-scaling factor is not known before homomorphically encrypting the local models at FL-trainers. This approach preserves input privacy.
- *unencrypted-lazy approach*: The weight-scaling is performed at the aggregator on unencrypted weights. This requires more computation time at the aggregator. For this approach, no HE is performed at FL-trainers, so input privacy is not preserved.
- *encrypted-aggressive approach*: The weight-scaling is performed by the FL trainers on un-encrypted weights, afterward homomorphic encryption is performed by FL-trainers. This approach requires less computation time at the aggregator and also preserves privacy.
- *unencrypted-aggressive approach*: The weight scaling is performed by the FL-trainers on un-encrypted weights. For this approach, no HE is performed at FL-trainers. This approach does not preserve input privacy.

C. QUANTITATIVE EVALUATION

1) MODEL SIZE

The storage requirement for FL aggregation depends upon the number of silos and model architecture. The state_dicts of models are stored using the pickle package of python. Fig. 3 (a) and Fig. 3 (b) shows the model size for state_dict of a simple(un-encrypted local model), encrypted (local model), and aggregated encrypted (global) model for NN architecture given in Table 4 for MNIST classification and CNN architecture given in Table 5 for Fashion-MNIST classification respectively. The simple(un-encrypted) state_dict is converted from tensors to list form using tolist() function to perform encryption using TenSEAL. So, the model size is significantly increased per local or per global model to preserve the input privacy of the FL process through homomorphic encryption.

2) AGGREGATION TIME PER GLOBAL EPOCH

The aggregation time requirement for cross-silo FL aggregation depends upon the number of silos and the aggregation scheme. For N = 3, the average aggregation time per global epoch for MNISTs' NN is shown in Fig. 3 (c). Similarly, for N = 3, the aggregation time per global epoch for Fashion-MNIST's CNN is shown in Fig. 3 (d). The aggregation time for different approaches for different number of FL-trainers $N = [3, 10]$ is shown in Fig. 3 (e) and Fig. 3 (f) for MNIST's NN and Fashion-MNIST's CNN respectively. It seems the aggregation time for the encrypted-lazy approach increases steeply w.r.t N. This is because encrypted multiplication with scalar is computationally expensive in TenSEAL. Comparatively, the encrypted-aggressive approach is less steep as only

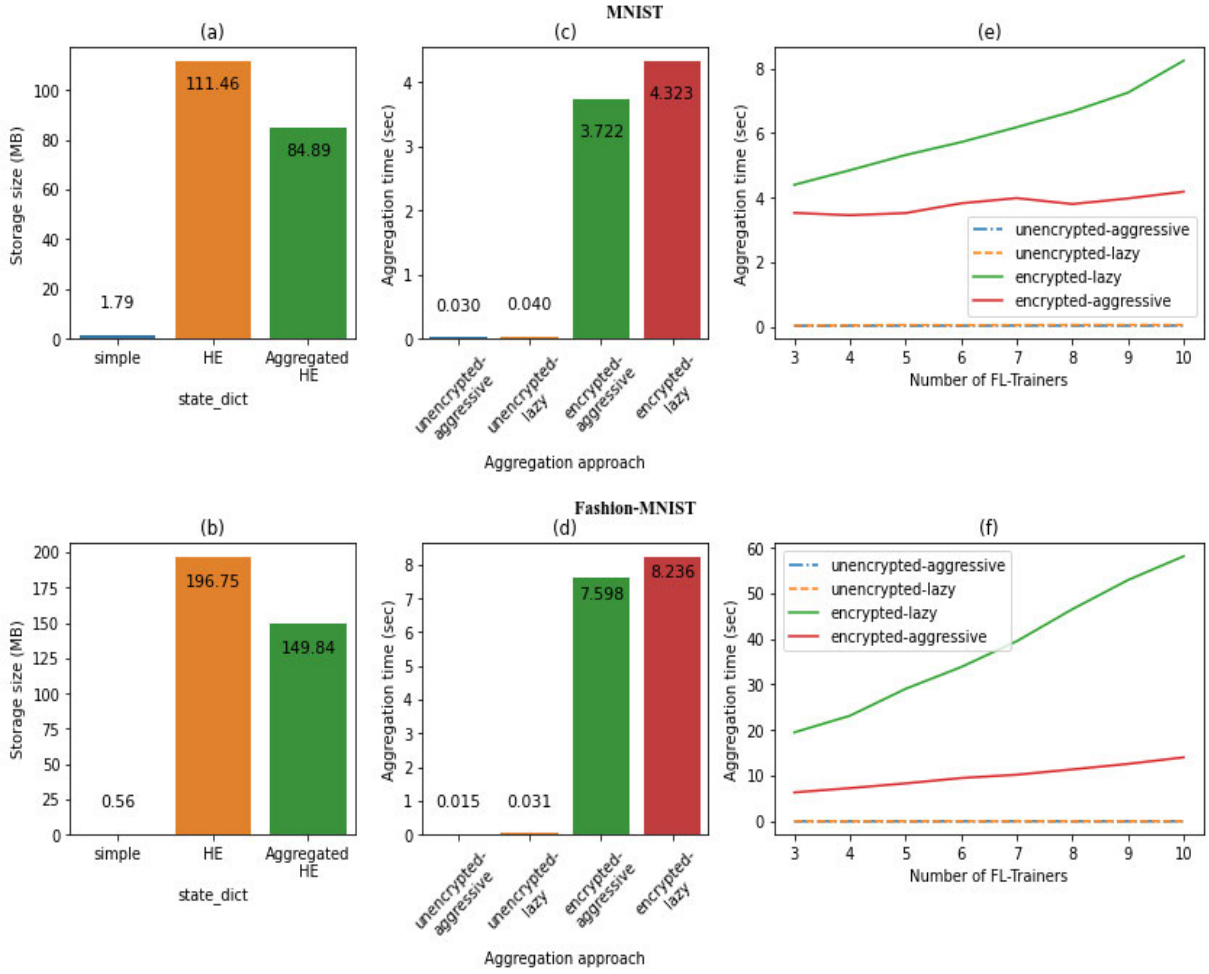


FIGURE 3. (a) Model size in MB for un-encrypted(simple), encrypted (local model), and aggregated encrypted (global model) state_dicts for MNIST dataset, N=3. (b) Model size in MB for un-encrypted(simple), encrypted (local model), and aggregated encrypted (global model) state_dicts for Fashion-MNIST dataset, N = 3. (c) Aggregation time in sec per epoch for different aggregation approaches for MNIST's NN model, N = 3. (d) Aggregation time in sec per epoch for different aggregation approaches for Fashion-MNIST's CNN model, N = 3. (e) Aggregation time in sec per epoch for different aggregation approaches for MNIST's NN model, N = [3,10]. (f) Aggregation time in sec per epoch for different aggregation approaches for Fashion-MNIST's CNN model, N = [3,10].

addition is required in the aggregation process. So for large N, an encrypted-aggressive approach is recommended, provided the weight scaling factor is pre-determined. It is necessary to mention that the aggregation scheme for Fig. 3(e) has been tweaked to accommodate more FL-trainers (silos), thus there is a difference of aggregation time at N = 3 with Fig. 3(d).

3) ACCURACY

Table 6 and Table 7 show the accuracy of global models aggregated through different approaches for MNIST's NN and Fashion-MNIST's CNN models respectively. The accuracy of global models aggregated through unencrypted-lazy, encrypted-lazy, unencrypted-aggressive, and encrypted-aggressive approaches have the same accuracy. This shows that incorporating input privacy in the FL process does not affect accuracy.

4) OUTPUT VERIFICATION

Our framework ST-BFL provides output verification, which means that the aggregation at the FL-aggregator is verified by FL-verifiers. This prevents foul play and malicious activities, thus preventing the accuracy to fall as well as maintain the integrity of the FL process. Fig. 4 and Fig. 5 shows that our framework prevents the aggregation attack at FL-aggregator.

D. QUALITATIVE EVALUATION

1) AUDIT TRAIL

Our FL through Ethereum and IPFS scheme offers a complete trail of the FL process from initialization to convergence. The proposed framework can ensure decentralized accountability for collectively learned ML models by exploiting the non-repudiation property of blockchain, thus increasing the reliability of the FL process. This ensures that the false-aggregation attacks on the FL process can be caught

TABLE 6. Accuracy for MNIST, N = 3.

Epochs	local1	local2	local3	unencrypted lazy	encrypted lazy	unencrypted aggressive	encrypted aggressive
0	-	-	-	0.08	-	-	-
1	0.38	0.39	0.19	0.22	0.22	0.22	0.22
2	0.38	0.39	0.32	0.50	0.50	0.50	0.50
3	0.38	0.39	0.37	0.45	0.45	0.45	0.45
4	0.38	0.39	0.40	0.60	0.60	0.60	0.60
5	0.38	0.39	0.40	0.70	0.70	0.70	0.70
6	0.38	0.39	0.41	0.74	0.74	0.74	0.74
7	0.74	0.74	0.74	0.74	0.74	0.74	0.74
8	0.74	0.74	0.74	0.74	0.74	0.74	0.74

TABLE 7. Accuracy for Fashion-MNIST, N = 3.

Epochs	local1	local2	local3	unencrypted lazy	encrypted lazy	unencrypted aggressive	encrypted aggressive
0	-	-	-	0.04	-	-	-
1	0.79	0.79	0.77	0.71	0.71	0.71	0.71
2	0.79	0.78	0.77	0.86	0.86	0.86	0.86
3	0.79	0.80	0.78	0.87	0.87	0.87	0.87
4	0.78	0.78	0.77	0.86	0.86	0.86	0.86
5	0.78	0.78	0.76	0.87	0.87	0.87	0.87
6	0.78	0.81	0.78	0.87	0.87	0.87	0.87
7	0.79	0.80	0.78	0.87	0.87	0.87	0.87
8	0.79	0.80	0.78	0.87	0.87	0.87	0.87

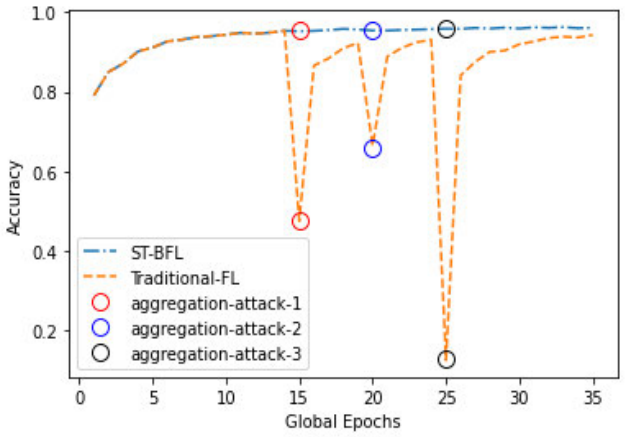


FIGURE 4. Accuracy for ST-BFL and traditional-FL under aggregation attack for MNIST (N = 10).

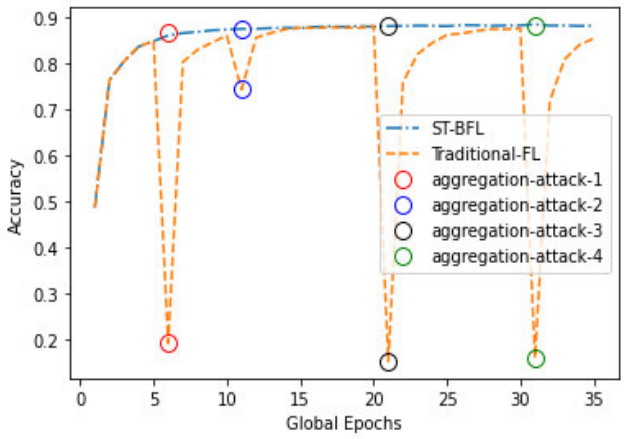


FIGURE 5. Accuracy for ST-BFL and traditional-FL under aggregation attack for Fashion-MNIST (N = 10).

and mitigated by validating the aggregation process by FL-verifiers. It is worthy to mention that given a public context for TenSEAL homomorphic scheme, an aggregation policy, and the input state_dict to aggregation scheme; the resultant state_dict is deterministic that is it has the same byte sequence and so IPFS hash. This makes the output verification of our proposed FL scheme as simple as comparing the IPFS hash of resultant aggregated state_dicts of FL-aggregator and FL-verifiers, while the output privacy remains intact through homomorphic encryption.

Moreover, the audit trail also helps in devising the incentive in terms of coins(monetary) and reputation.

2) STORAGE

Our designed scheme requires very little on-chain storage so the cost of operating is significantly reduced.

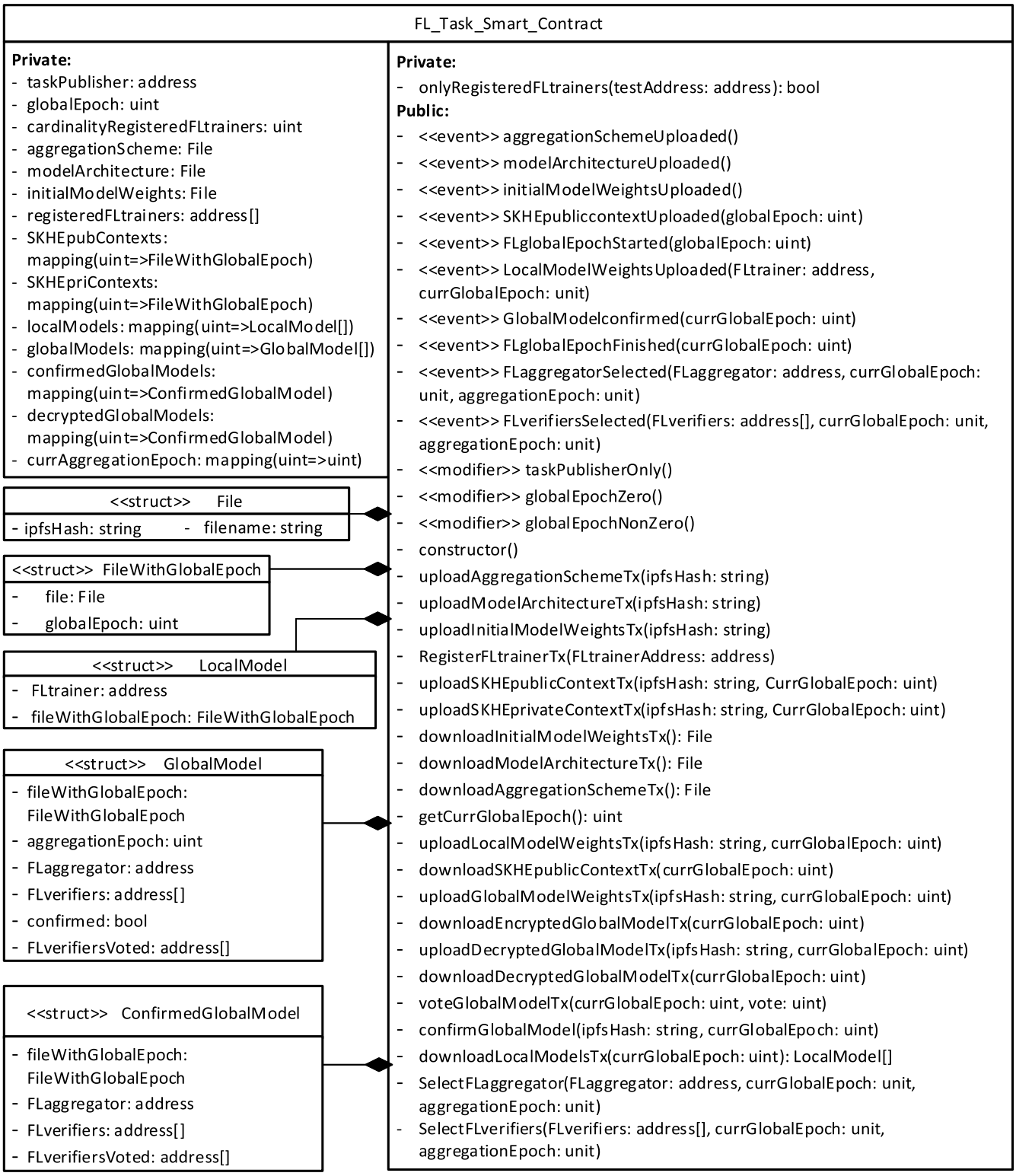


FIGURE 6. Simplified UML diagram for smart contract SM_f.

3) SINGLE-POINT-OF-FAILURE

Our designed scheme removes the single-point-of-failure at the aggregation step. It also enforces the FL-aggregators for honest aggregation.

4) INPUT PRIVACY

The local models are encrypted by homomorphic encryption. Thus input privacy remains intact not only from outsiders, FL-aggregators, FL-verifiers but

also from other trainers participating in the same FL task.

5) OUTPUT PRIVACY

The global models can be decrypted by FL-trainers only. Thus output privacy remains intact from outsiders, FL-aggregators as well as FL-verifiers.

6) INPUT VERIFICATION

Since our framework is concerned with cross-silo FL, there is no incentive for poisoning attack for the FL trainers as they put their resources and time to jointly train the high-accurate global model. However, if necessary in ST-BFL, the encrypted local models can be evaluated on a deterministic shuffled test dataset by the FL-aggregators and FL-verifiers to agree upon encrypted-predicted labels for the encrypted local model. This encrypted-predicted labels and actual (unencrypted) target labels can be sent to TP_f for decryption using $SKHE^{Pri}_{f,t}$. The TP_f calculates the accuracy and afterward log against the respective local model. Though this method relies on the honesty of TP_f.

7) OUTPUT VERIFICATION

The aggregation process by FL-aggregator is validated by FL verifiers, thus output verification is done.

8) FLOW GOVERNANCE

Initially, the FL-task publisher initiates FL flow. Afterward, The smart contract orchestrates the FL process.

VI. CONCLUSION

FL is a distributed ML scheme where training data remain on-device. Local models in FL are still subject to inference attacks. The centralized server is also vulnerable to aggregation attacks and is a single point of failure. To mitigate these issues, we proposed ST-BFL, a structured transparency empowered cross-silo federated learning on the blockchain framework. We describe how different components of structured transparency such as input privacy, output privacy, input verification, output verification, and flow governance are satisfied in our proposed approach with help of homomorphic encryption, FL-aggregators, and FL-verifiers, and smart contract. We performed a quantitative and qualitative evaluation of our framework. It is shown that ST-BFL has the same accuracy as traditional FL. Lastly, we analyze our framework from the perspective of aggregation time, accuracy, and storage size. In the future, we intend to evaluate our proposed framework on multi-key homomorphic (MKHE) schemes as soon as the feasible libraries are available. Moreover, we will work on input verification while maintaining input privacy in the ST-BFL. We will also try to extend our work to be applicable in cross-device FL settings, for this we will integrate the reward mechanism still maintaining the input privacy for the better flow governance of the FL process.

APPENDIX A
SMART CONTRACT UML DIAGRAM

See Fig.6.

REFERENCES

[1] U. Majeed, L. U. Khan, I. Yaqoob, S. M. A. Kazmi, K. Salah, and C. S. Hong, "Blockchain for IoT-based smart cities: Recent advances, requirements, and future challenges," *J. Netw. Comput. Appl.*, vol. 181, May 2021, Art. no. 103007.

[2] K. Bochie, M. S. Gilbert, L. Gantert, M. S. M. Barbosa, D. S. V. Medeiros, and M. E. M. Campista, "A survey on deep learning for challenged networks: Applications and trends," *J. Netw. Comput. Appl.*, vol. 194, Nov. 2021, Art. no. 103213.

[3] L. U. Khan, W. Saad, D. Niyato, Z. Han, and C. S. Hong, "Digital-twin-enabled 6G: Vision, architectural trends, and future directions," 2021, *arXiv:2102.12169*.

[4] A. Trask, E. Bluemke, B. Garfinkel, C. Ghezzou Cuervas-Mons, and A. Dafoe, "Beyond privacy trade-offs with structured transparency," 2020, *arXiv:2012.08347*.

[5] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Stat.*, Fort Lauderdale, FL, USA, Apr. 2017, pp. 1273–1282.

[6] J. Koneáný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," 2016, *arXiv:1610.02527*.

[7] U. Majeed and C. S. Hong, "FLchain: Federated learning via MEC-enabled blockchain network," in *Proc. 20th Asia–Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Matsue, Japan, Sep. 2019, pp. 1–4.

[8] L. U. Khan, Z. Han, D. Niyato, and C. S. Hong, "Socially-aware-clustering-enabled federated learning for edge networks," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 3, pp. 2641–2658, Sep. 2021.

[9] L. U. Khan, Y. K. Tun, M. Alsenwi, M. Imran, Z. Han, and C. S. Hong, "A dispersed federated learning framework for 6G-enabled autonomous driving cars," 2021, *arXiv:2105.09641*.

[10] A. Hall, M. Jay, T. Cebere, and B. Cebere, "Syft 0.5: A platform for universally deployable structured transparency," 2021, *arXiv:2104.12385*.

[11] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Commun. Lett.*, vol. 24, no. 6, pp. 1279–1283, Jun. 2020.

[12] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Netw.*, vol. 35, no. 1, pp. 234–241, Jan. 2021.

[13] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," 2016, *arXiv:1602.05629*.

[14] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Symp. theory Comput.*, New York, NY, USA, 2009, pp. 169–178.

[15] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," 2017, *arXiv:1704.03578*.

[16] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning," in *Proc. Annu. Tech. Conf.*, Jul. 2020, pp. 493–506.

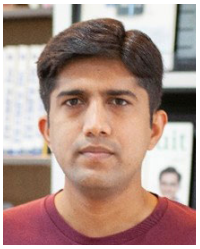
[17] Z. Chai, Y. Chen, A. Anwar, L. Zhao, Y. Cheng, and H. Rangwala, "FedAT: A high-performance and communication-efficient federated learning system with asynchronous tiers," 2020, *arXiv:2010.05958*.

[18] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.

[19] B. Farahani, F. Firouzi, and M. Luecking, "The convergence of IoT and distributed ledger technologies (DLT): Opportunities, challenges, and solutions," *J. Netw. Comput. Appl.*, vol. 177, Mar. 2021, Art. no. 102936.

[20] N. Szabo. (1994). *Smart Contracts*. Accessed: Jul. 30, 2021. [Online]. Available: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html

[21] N. Szabo. (1997). *The Idea of Smart Contracts*. Accessed: Jul. 30, 2021. [Online]. Available: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_idea.html
[22] H. Hasan and K. Salah, "Proof of delivery of digital assets using blockchain and smart contracts," *IEEE Access*, vol. 6, pp. 65439–65448, 2018.
[23] T. Hewa, M. Ylianttila, and M. Liyanage, "Survey on blockchain based smart contracts: Applications, opportunities and challenges," *J. Netw. Comput. Appl.*, vol. 177, Mar. 2021, Art. no. 102857.
[24] V. Buterin. (2014). *A Next-Generation Smart Contract and Decentralized Application Platform*. Accessed: Jul. 30, 2021. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper
[25] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," *Future Gener. Comput. Syst.*, vol. 105, pp. 475–491, Dec. 2020.
[26] J. Ellul, "Towards configurable and efficient runtime verification of blockchain based smart contracts at the virtual machine level," in *Leveraging Applications of Formal Methods, Verification and Validation: Applications*, T. Margaria and B. Steffen, Eds. Cham, Switzerland: Springer, 2020, pp. 131–145.
[27] X. Chang, J. Zhu, and S. Zhao, "Dynamic array double-access attack in Ethereum," in *Proc. IEEE 6th Int. Conf. Comput. Commun. (ICCC)*, Chengdu, China, Dec. 2020, pp. 1065–1071.
[28] (2021). *Solidity Documentation*. Accessed: Jul. 30, 2021. [Online]. Available: https://solidity.readthedocs.io
[29] Remix. *Remix-Ethereum IDE*. Accessed: Jul. 30, 2021. [Online]. Available: https://remix.ethereum.org/
[30] T. Suite. *Truffle*. Accessed: Jul. 30, 2021. [Online]. Available: https://truffleframework.com/truffle/
[31] T. Suite. *Ganache*. Accessed: Jul. 30, 2021. [Online]. Available: https://truffleframework.com/ganache/
[32] W.-M. Lee, "Testing smart contracts using Ganache," in *Beginning Ethereum Smart Contracts Programming*. Berkeley, CA, USA: Apress, 2019, pp. 147–167.
[33] J. Benet, "IPFS—Content addressed, versioned, P2P file system," 2014, *arXiv:1407.3561v1*.
[34] *IPFS Powers the Distributed Web: A Peer-to-Peer Hypermedia Protocol Designed to Make the Faster, Safer, and More Open*. Accessed: Jul. 30, 2021. [Online]. Available: https://ipfs.io
[35] N. Zahed Benisi, M. Aminian, and B. Javadi, "Blockchain-based decentralized storage networks: A survey," *J. Netw. Comput. Appl.*, vol. 162, Jul. 2020, Art. no. 102656.
[36] N. Kaaniche, M. Laurent, and S. Belguith, "Privacy enhancing technologies for solving the privacy-personalization paradox: Taxonomy and survey," *J. Netw. Comput. Appl.*, vol. 171, Dec. 2020, Art. no. 102807.
[37] X. Yi, R. Paulet, and E. Bertino, *Homomorphic Encryption Applications*. Cham, Switzerland: Springer, 2014.
[38] A. C.-F. Chan, "Symmetric-key homomorphic encryption for encrypted data processing," in *Proc. IEEE Int. Conf. Commun.*, Dresden, Germany, Jun. 2009, pp. 1–5.
[39] L. Chen, M. Lim, and Z. Fan, "A public key compression scheme for fully homomorphic encryption based on quadratic parameters with correction," *IEEE Access*, vol. 5, pp. 17692–17700, 2017.
[40] K. S. Chong, C. N. Yap, and Z. H. Tew, "Multi-key homomorphic encryption create new multiple logic gates and arithmetic circuit," in *Proc. 8th Int. Symp. Digit. Forensics Secur. (ISDFS)*, Beirut, Lebanon, Jun. 2020, pp. 1–4.
[41] H. Chen, I. Chillotti, and Y. Song, "Multi-key homomorphic encryption from TFHE," in *Proc. 25th Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Kobe, Japan, 2019, pp. 446–472.
[42] U. Majeed, S. S. Hassan, and C. S. Hong, "Cross-silo model-based secure federated transfer learning for flow-based traffic classification," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jeju Island, South Korea, Jan. 2021, pp. 588–593.
[43] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*.
[44] U. Majeed, L. U. Khan, and C. S. Hong, "Cross-silo horizontal federated learning for flow-based time-related-features oriented traffic classification," in *Proc. 21st Asia–Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Daegu, South Korea, Sep. 2020, pp. 389–392.
[45] H. Al-Breiki, M. H. U. Rehman, K. Salah, and D. Svetinovic, "Trustworthy blockchain oracles: Review, comparison, and open research challenges," *IEEE Access*, vol. 8, pp. 85675–85685, 2020.
[46] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, and E. Yang, "PyTorch: An imperative style, high-performance deep learning library," 2019, *arXiv:1912.01703*.
[47] J. Papa, *PyTorch Pocket Reference*. Newton, MA, USA: O'Reilly Media, 2021.
[48] A. Benaissa, B. Retiat, B. Cebere, and A. Eddine Belfedhal, "TenSEAL: A library for encrypted tensor operations using homomorphic encryption," 2021, *arXiv:2104.03152*.
[49] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proc. 23rd Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Hong Kong, 2017, pp. 409–437.
[50] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Int. Conf. Mach. Learn.*, Madison, WI, USA, 2010, pp. 807–814.
[51] W. Alam, S. D. Ali, H. Tayara, and K. Chong, "A CNN-based RNA N6-methyladenosine site predictor for multiple species using heterogeneous features representation," *IEEE Access*, vol. 8, pp. 138203–138209, 2020.
[52] P. Zhao, X. Chen, V. Chung, and H. Li, "DeLFIQE—A low-complexity deep learning-based light field image quality evaluator," *IEEE Trans. Instrum. Meas.*, vol. 70, 2021, Art. no. 5014811.
[53] Q. Liu and C. Huang, "A fault diagnosis method based on transfer convolutional neural networks," *IEEE Access*, vol. 7, pp. 171423–171430, 2019.



UMER MAJEED received the B.S. degree in electrical engineering from the National University of Science and Technology (NUST), Pakistan, in 2015. He is currently pursuing the Ph.D. degree in computer engineering with Kyung Hee University (KHU), South Korea. He is working as a Researcher at the Intelligent Networking Laboratory under a project jointly funded by the prestigious Brain Korea 21st Century Plus and the Ministry of Science and ICT, South Korea. His research interests include blockchain, mobile edge computing, the Internet of Things, machine learning, and wireless networks.

LATIF U. KHAN received the M.S. degree (Hons.) in electrical engineering from the University of Engineering and Technology (UET), Peshawar, Pakistan, in 2017, and the Ph.D. degree in computer engineering from Kyung Hee University (KHU), South Korea, in 2021. Before joining KHU, he has worked as a Faculty Member and a Research Associate at UET. He has published his works in highly reputable conferences and journals. His research interests include federated learning, analytical techniques of optimization, and game theory to edge computing and end-to-end network slicing. He has received the Best Paper Award in 15th IEEE International Conference on Advanced Communications Technology, Pyeongchang, South Korea, in 2013. He is the recipient of the KHU Best Thesis Award 2021.

ABDULLAH YOUSAFZAI received the B.C.S. degree (Hons.) from Hazara University Mansehra, Pakistan, in 2009, the M.S. degree in computer science from the COMSATS Institute of Information Technology, Abbottabad, in 2013, and the Ph.D. degree from the University of Malaya, in 2017. He is currently working as an Assistant Professor at the University of Management and Technology, Lahore, Pakistan. In the past, he worked as a Postdoctoral Research Fellow under the prestigious Brain Korea 21st Century Plus and the National Research Foundation of Korea at the Department of Computer Science and Engineering, Kyung Hee University, Republic of Korea. Besides, he worked as an Assistant Professor with the Department of Computer Science and Engineering, HITEC University, Taxila, Pakistan. Before that, he worked as a Brightspark's Research Assistant at the C4MCCR University of Malaysia and a Backend Web Developer in Pakistan. His work mainly focuses on distributed computing environments comprising cloud computing systems, edge computing, mobile cloud computing, blockchain systems, and the Internet of Things.

ZHU HAN (Fellow, IEEE) received the B.S. degree in electronic engineering from Tsinghua University, in 1997, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, in 1999 and 2003, respectively. From 2000 to 2002, he was a Research and Development Engineer of JDSU, Germantown, MD, USA. From 2003 to 2006, he was a Research Associate at the University of Maryland. From 2006 to 2008, he was an Assistant Professor at Boise State University, ID, USA. He is currently a John and Rebecca Moores Professor with the Electrical and Computer Engineering Department as well as the Computer Science Department, University of Houston, TX, USA. His research interests include wireless resource allocation and management, wireless communications and networking, game theory, big data analysis, security, and smart grid. He has been an AAAS Fellow, since 2019, and an ACM Distinguished Member, since 2019. Since 2017, he has been a 1% Highly Cited Researcher according to Web of Science. He received the NSF Career Award, in 2010, the Fred W. Ellersick Prize of the IEEE Communication Society, in 2011, the EURASIP Best Paper Award for the Journal on Advances in Signal Processing, in 2015, the IEEE Leonard G. Abraham Prize in the field of communications systems (the Best Paper Award in IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS), in 2016, and several best paper awards in IEEE conferences. He is also the Winner of the 2021 IEEE Kiyo Tomiyasu Award, for outstanding early to mid-career contributions to technologies holding the promise of innovative applications, with the following citation: "for contributions to game theory and distributed management of autonomous communication networks." He was an IEEE Communications Society Distinguished Lecturer, from 2015 to 2018.

BANG JU PARK received the B.S. and M.S. degrees in electronic engineering from Kyung Hee University, South Korea, in 1986 and 1988, respectively, and the Ph.D. degree in radio engineering from the Korea Institute of Science and Technology, Kyung Hee University, in 2006. From 1988 to 2012, he was a Science Reporter with JoongAng Daily, South Korea. From 2008 to 2012, he was a Visiting Professor with the College of Bio-Nano Engineering, Gachon University, South Korea. In 2013, he moved to Gachon University, where he is currently a Full Professor with the Department of Electronic Engineering and the Director of the Institute of Bio and Mechatronics. His research interests include future internet and radio manipulation for biomedical instrumental applications.

CHOONG SEON HONG (Senior Member, IEEE) received the B.S. and M.S. degrees in electronic engineering from Kyung Hee University, Seoul, South Korea, in 1983 and 1985, respectively, and the Ph.D. degree from Keio University, Tokyo, Japan, in 1997. In 1988, he joined KT, Gyeonggi-do, South Korea, where he was involved in broadband networks as a member of the Technical Staff. Since 1993, he has been with Keio University. He was with the Telecommunications Network Laboratory, KT, as a Senior Member of Technical Staff and the Director of the Networking Research Team, until 1999. Since 1999, he has been a Professor with the Department of Computer Science and Engineering, Kyung Hee University. His research interests include future internet, intelligent edge computing, network management, and network security. He is a member of the Association for Computing Machinery (ACM), the Institute of Electronics, Information and Communication Engineers (IEICE), the Information Processing Society of Japan (IPSJ), the Korean Institute of Information Scientists and Engineers (KIISE), the Korean Institute of Communications and Information Sciences (KICS), the Korean Information Processing Society (KIPS), and the Open Standards and ICT Association (OSIA). He has served as the General Chair, the TPC Chair/Member, or an Organizing Committee Member for international conferences, such as the Network Operations and Management Symposium (NOMS), the International Symposium on Integrated Network Management (IM), Asia–Pacific Network Operations and Management Symposium (APNOMS), End-to-End Monitoring Techniques and Services (E2EMON), IEEE Consumer Communications and Networking Conference (CCNC), Assurance in Distributed Systems and Networks (ADSN), the International Conference on Parallel Processing (ICPP), Data Integration and Mining (DIM), World Conference on Information Security Applications (WISA), Broadband Convergence Network (BcN), Telecommunication Information Networking Architecture (TINA), the International Symposium on Applications and the Internet (SAINT), and the International Conference on Information Networking (ICOIN). He was an Associate Editor of the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT and the IEEE JOURNAL OF COMMUNICATIONS AND NETWORKS and an Associate Editor for the *International Journal of Network Management* and an Associate Technical Editor of *IEEE Communications Magazine*. He serves as an Associate Editor for the *International Journal of Network Management* and *Future Internet* journal.