IEEE ICBC'24 1570982025

Incentivized Federated Learning with Local Differential Privacy using Permissioned Blockchains

***Abstract*—Federated Learning (FL) is a collaborative machine learning approach that enables data owner nodes to retain their data locally, preventing its transfer to a central server. It involves sharing only the local model parameters with the server to update a global model, which is then disseminated back to the local nodes. Despite its iterative convergence, FL has several limitations, such as the risk of single-point failure, inadequate incentives for participating nodes, and potential privacy breaches. While Local Differential Privacy (LDP) is often perceived as the *de facto* standard for addressing privacy concerns, the other challenges of FL have not yet been discussed comprehensively, even for Locally Differentially Private Federated Learning (LDP-FL). We propose an integrated approach that uses permissioned blockchains to guard against a single point of failure and a token-based incentivization (TBI) mechanism for encouraging participation in LDP-FL. In our scheme, participating nodes receive tokens upon sharing their model parameters, which can subsequently be used to access updated global models. The number of tokens awarded for parameter sharing is determined by ϵ - the privacy factor of LDP, ensuring that the nodes do not overly obfuscate the data they share. The primary objective is establishing suitable incentives for active participation in model construction through LDP-FL, thereby discouraging opportunistic behavior. We demonstrate the feasibility of our approach by developing the *Blockchain-based TBI-LDP-FL framework* (hereinafter, referred to as BTLF) on Hyperledger Fabric. Extensive results of experimentation establish the efficacy of BTLF.**

***Index Terms*—Federated Learning, Blockchain, Local Differential Privacy, Incentivization, Hyperledger Fabric**

I. Introduction

Machine learning (ML) models have achieved remarkable success in various domains due to the large-scale datasets available for robust training. Federated Learning (FL) [1], which is a form of collaborative learning, has emerged as a flexible and dynamic technique facilitating algorithmic training through independent client sessions, each leveraging its own unique dataset. In this framework, a central server selects a model for training, subsequently communicating it to a set of local nodes, which in turn conduct local model training utilizing their respective datasets. Unlike traditional distributed learning, instead of the entire dataset, only the resulting model parameters are shared with the server in FL. The server utilizes the collected local parameters to update global model weights, which are then disseminated back to the local nodes [2]. The defining characteristic of FL lies in its ability to facilitate multiple entities in collaboratively developing a robust ML model without the inherent necessity of data sharing and its associated high data leakage risks.

FL has been studied extensively in recent years, and the field has developed rapidly [3], [4]. However, the traditional FL frameworks still face several shortcomings that reduce the reliability of the whole system [5]–[8]. First, the FL framework typically relies on a central server to consolidate local training outcomes for updating the global model. However, the server's vulnerability to intentional manipulation, network failures, and external attacks poses a significant threat. A server compromise can lead to the failure of the entire FL system. Secondly, with a multitude of participants in FL, it becomes impractical to assume the absolute honesty of all clients adhering to the FL protocols during local model training. Dishonest clients may introduce false data, significantly impacting the performance of the global model.

Further, despite the basic premise of FL to solely transmit local model parameters without sharing the complete data, exposure of private information due to inference attacks at the server still remains possible under certain conditions [9]. This susceptibility can trigger a range of privacy-related issues. Another critical concern is that standard FL models lack an incentivization structure, compelling participating clients to contribute their computational resources without receiving adequate compensation. Absence of incentives not only hampers honest participation but also limits the engagement of a sufficiently large number of clients, often found necessary for data-intensive tasks. The above challenges necessitate an integrated and robust framework that considers the existing limitations while ensuring the efficiency and correctness of the model training process. In order to address these, we present an innovative approach through the integration of *permissioned blockchains* [7], *Local Differential Privacy* (LDP) [10], and *Token-Based Incentivization* (TBI).

Inclusion of each of these facets in our framework is crucial for various reasons. Using a blockchain network instead of a central aggregator/server is robust against potential server crashes. With blockchain, model aggregation is distributed across multiple nodes, reducing the vulnerability to a single point of failure. It can also effectively mitigate model parameter tampering risks by malicious clients or the aggregated weights by the server as these are written to the ledger and are operated on well formed smart contracts. Supporting LDP in the FL process minimizes the possibility of revealing private data to the server by the clients [11]. The local nodes can perturb their model parameter values pertaining to a privacy budget ϵ before writing to the blockchain ledger. On the other hand, the Token-Based Incentivization (TBI) mechanism awards tokens to the clients when the server/aggregator receives the model parameters from the corresponding nodes. These tokens can subsequently be used by the clients to

accesss the updated global model parameters. The quantum of tokens awarded is a function of the privacy parameter ϵ of LDP used by a client, thus ensuring that the nodes do not over-restrict the data they share. TBI promotes participation of clients in federated model building and at the same time discourages free lunchers.

In summary, this paper makes the following novel contributions.

1) We use Hyperledger Fabric as the underlying technology with well formed chaincodes for sharing of local model parameters by the clients and global model parameters by the server in a Federated Learning setting.
2) We develop chaincodes for implementing Local Differential Privacy while sharing model parameters by the clients, thereby removing the need to trust the server performing aggregation in the FL process.
3) Besides carrying out local model training, the client nodes are also equipped with a token based incentivization mechanism that enables them to acquire new tokens while sharing their model parameters and use them at the time of reading the global parameters, both in accordance with a predefined policy.

To the best of our knowledge, there is no existing work in the literature that combines incentivization with locally differentially private federated learning in a permissioned blockchain setting.

The rest of the paper is organized as follows. Section II provides a brief overview of the preliminaries. Section III reviews recent literature on related topics. The proposed framework is described in Section IV. Details of Hyperledger Fabric based implementation of BTLF is presented in Section V. Section VI discusses the results of an extensive set of experiments carried out on an end-to-end working BTLF system, and finally conclude with future directions for research in Section VII.

II. Preliminaries

This section introduces some of the basic concepts underlying the work presented later in the paper.

A. Federated Learning

Federated learning (FL) [2] orchestrates collaborative training of local deep neural network (DNN) models among *N* distributed clients connected to a central server. The process commences with the server randomly initializing the model parameters, denoted as μ_o, which are then distributed to the clients to initialize their respective model copies. Each client independently trains its model using its local dataset for multiple epochs, eventually producing updated model parameters, μ_u. The process of federated learning in each round includes averaging all the μ_u's received from the clients to obtain μ_{fed} (the parameters of the global model). Such an iterative process, known as a federation round, continues through multiple rounds until μ_{fed} either converges or the predetermined number of rounds is completed [11]. It has been shown that μ_{fed} produces almost the same accuracy as that of a local model trained on the complete data [12].

It is well known that collection and analysis of user data at scale is a driving force behind the recent progress in machine learning. As a result, there has been a significant increase in user data volume in recent years. However, data collected from users tend to be private and also sensitive. Further, the same can potentially be linked to other more confidential information. Owing to this, users are always sceptical about divulging their personal data, especially in the face of new and emerging technologies for in-depth data mining and analytics. In order to regulate the activities of corporate firms with respect to consumer data they have access to, various countries have enacted privacy laws such as General Data Protection Regulation (GDPR) [13] and California Consumer Privacy Act (CCPA) [14]. Therefore, privacy preservation when dealing with data becomes an urgent issue that needs to be addressed [15]. Privacy concerns in the context of Federated Learning have been raised as well, especially due to its potential vulnerability to inference attacks [16].

B. Local Differential Privacy

Differential privacy is a mathematical framework for ensuring the privacy of individuals in datasets, first proposed by [17]. It allows data to be analyzed without revealing sensitive information about any particular user in the dataset, thereby providing a strong guarantee of the individual's privacy. According to their definition, the outcome of the mechanism should not be significantly affected by the presence or absence of any particular record in the dataset. Hence, for any two datasets that differ in only one record, if the probability of any outcome occurring is nearly the same, then the mechanism is considered to be differentially private.

In the traditional model of differential privacy, also known as centralized differential privacy (CDP), there exists a trusted aggregator or curator which collects and holds the sensitive data of all the individuals. It is responsible for protecting their privacy by adding a measured amount of noise, i.e. perturbing the data by infusing sufficient noise to the output. Effectively, the goal is to mask the contribution of individual data elements and yet preserve overall analysis accuracy. The aggregator, however, has to first collect original data of the users before releasing the perturbed aggregated information publicly [15].

An issue with CDP is that there is an implicit but complete trust on the data curator, which may not always be the situation in real world. Even the largest and most reputable companies cannot guarantee their customers' privacy and may fall victim to data breaches. To address the problem of trusting a central authority for managing user data, the mechanism of local differential privacy (LDP) was proposed. In LDP, a user's data is perturbed locally before it is sent to the aggregator. The original data is only accessible to the owner, which provides a much stronger privacy guarantee for the user. In the LDP model, a curator holds only a perturbed version of the data and not the original data. Also, all the training and any form of querying is performed on this perturbed dataset only. Thus,

LDP protects against data disclosure even to the untrusted curators and relieves the burden on the trusted data curators to keep data secure [15]. LDP is often defined in terms of what is known as ϵ-Local Differential Privacy. As defined in [18], a randomized mechanism $\mathcal{A}$ is said to satisfy ϵ-local differential privacy if for any two input sets D_1 and D_2 differing on at most one element, and for any set of outcomes $O \subseteq Range(\mathcal{A})$,

$$Pr[\mathcal{A}(D_1) \in O] \leq exp(\epsilon) \times Pr[\mathcal{A}(D_2) \in O]$$

Here $\epsilon > 0$ is the privacy budget; the smaller the value of ϵ, the stricter protection with lower data availability, and vice versa. Federated Learning is a natural application domain for LDP and has been investigated from different aspects and application domains [19], [20], [21], [22], [23].

III. Related Work

Several works [8], [24]–[32] in the recent literature have explored decentralized federated learning by integrating FL with blockchain-based decentralized execution. While majority of these [26], [28], [30] mainly focus on secured decentralized training of the FL model through multiple clients, some [27], [31], [32] have also considered application-specific use cases for decentralized machine learning. A recent research by Hai *et al.* [32] introduce an innovative integration of FL and blockchain technology in developing a medical records recommendation system. Pokhrel and Chai [8] present FL with blockchain for autonomous vehicles along with automobile design challenges. In yet another domain-specific application, Liu *et al.* [33] have discussed a secure FL framework for 5G networks, where both effective learning and security have a significant role. In recent years, there has been some research towards privacy-preserving in federated learning. For example, Bhowmick *et al.* [34] introduced a method that protects against the reconstruction of parameters, thus enhancing privacy in FL. In contrast, Hao *et al.* [19] focused on efficiency and confidentiality in federated deep learning. Although these existing works have explored secured and privacy-preserved decentralized model training through blockchain-based FL, they primarily focused on model privacy rather than data privacy for individual clients participating in the federated training procedure. *Local Differential Privacy* becomes essential in this context to ensure data privacy for the individual clients participating in the model training procedure.

While some efforts have been made to employ blockchain in the context of federated learning [7], limited research has focused on incorporating LDP in this domain. Critical design choices necessitate careful evaluation, including determining whether the same nodes perform all three crucial operations: model parameter calculation, LDP-related computation, and distributed ledger updating. Notably, in domains like the Internet of Things (IoT), where end devices may lack adequate computational power, allocating tasks to edge nodes has significant implications for device privacy.

Among the various application-specific domains, Li et al. [35] consider FL for segmenting brain tumors. In contrast, Lu et al. [20] integrate differential privacy with federated learning in mobile edge computing. They emphasize the importance of privacy-preserving approaches in urban settings. Likewise, the focus of the work of Zhao et al. [22] is LDP-based FL for the Internet of Things. In a more general setting, the work of Truex et al. [21] gives a formal privacy guarantee for LDP in federated learning.

Furthermore, existing literature provides limited insights into the development of appropriate incentive schemes for federated learning. Kong et al. [36] propose an incentivization mechanism that does not monetize data like the other approaches. Instead, model performance is used as the reward, i.e., those making more significant contributions can access more accurate models. It has been shown that clients will benefit by sharing as much data as they possess to participate in federated learning under this incentive mechanism. Some blockchain-based FL techniques have been proposed using specific cryptocurrencies as incentives [37]. However, the potential legal implications of such currencies can impact participants' willingness to engage. Xu et al. [18] model incentivization in federated learning with differential privacy in industrial IoT as a Stackelberg game with the aggregating server as the leader and the client nodes as followers. The server tries to maximize the utility of its available, total budget by appropriately rewarding the clients for data sharing. A systematic and comprehensive survey on incentivization in federated learning can be found in [38].

In contrast, as proposed in this paper, the token-based incentive approach circumvents these issues and allows for seamless integration across multiple blockchain networks, directly or through appropriate conversion factors. Aligning the incentive quantum with the LDP privacy factor ϵ will foster fairness and encourage active participation in the federated learning process. However, several research challenges are associated with such an integrated approach. First, the entire process should work autonomously by executing well-developed chaincodes. While the actual local model training works off-chain, model parameter sharing, crediting, and debiting of tokens, as well as global parameter calculation and dissemination, happen on-chain. Further, the incentivization scheme must be fair and depend on the privacy budget level for the individual clients. Thus, LDP and TBI are to be designed in an integrated manner, which should also be amenable to implementation as chaincodes on a permissioned blockchain network. The following section describes how our BTLF framework handles these challenges.

IV. Proposed Approach

In this section, we outline our proposed approach to address the various limitations faced by FL with the integration of permissioned Blockchain, LDP and TBI mechanisms. Fig. 1 shows the high-level flow diagram of the proposed solution.

A. Blockchain-based LDP-FL

To participate in the FL process, a client must complete registration and authorization within the permissioned

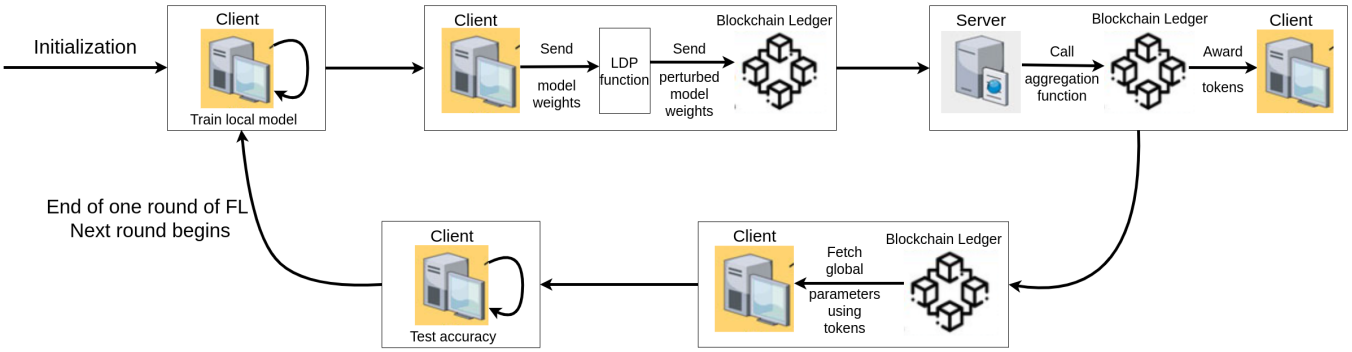


Figure 1: An overview of the BTLF framework

blockchain network. Thereafter, the client follows the pseudo code given in Algorithm 1. Upon initiation, each client within the blockchain network initializes a deep neural network (DNN) model with randomized parameters as shown in Line 1 of Algorithm 1. Subsequently, this model undergoes training using the available local dataset for multiple epochs, resulting in the updating of the model parameters. The updated model parameters are then subjected to perturbation, pertaining to the prescribed privacy factor ϵ of the LDP technique. Following this, the perturbed weights are securely written onto the blockchain ledger.

Algorithm 1 Pseudo code for Client

1: $(m, w) \leftarrow$ init()
2: **while** till convergence or maxRounds reached **do**
3: $w \leftarrow$ trainModel(m, w)
4: $w' \leftarrow$ LDP(w, ϵ) ▷ perturbing the parameters of local model
5: putParameters(w', ϵ)
6: $w \leftarrow$ globalParameters()
7: **end while**

In the traditional LDP approach, each client uniformly uses the same ϵ for perturbing the model parameters. However, the exploration of distinct privacy factors in a multi-privacy framework remains an active area of research. By incorporating multiple privacy factors, our approach facilitates better incentivization and increases fairness among the participating clients.

We use the LDP mechanism proposed in [23]. The algorithm, applied to the weights W of a model, produces a perturbed tuple W^* by introducing randomness to each dimension of W. For each weight $w \in W$, assuming $w \in [c-r, c+r]$ where c is the center and r is the radius of w's range, the following LDP mechanism to perturb w has been proposed:

$$w^* = \begin{cases} c + r \cdot \dfrac{e^\epsilon + 1}{e^\epsilon - 1} & \text{with probability } \dfrac{(w-c)(e^\epsilon - 1) + r(e^\epsilon + 1)}{2r(e^\epsilon + 1)}, \\ c - r \cdot \dfrac{e^\epsilon + 1}{e^\epsilon - 1} & \text{with probability } \dfrac{-(w-c)(e^\epsilon - 1) + r(e^\epsilon + 1)}{2r(e^\epsilon + 1)}. \end{cases} \quad (1)$$

From Equation 1 we may observe that, with a given privacy budget ϵ, the larger the range r, the larger the range of perturbed weights w^*, i.e. greater is the noise in the perturbed weights. In our approach, based on prior knowledge, all the clients and the server agree on the same weight range at the beginning when initializing weights, represented by (c, r). In addition, each client chooses a value of ϵ. When sending its local parameters to the server, each client perturbs them using the proposed LDP function given by Equation 1.

For any weight $w \in [c-r, c+r]$, where c is the center and r is the radius of w's range, it has been proven in [23] that the proposed mechanism satisfies ϵ-LDP w.r.t. $[c-r, c+r]$. Both ϵ and r have an impact on the privacy level. The degree of concealment of the individual data in a crowd of data is determined by ϵ, and r determines the size of the crowd.

After writing the perturbed weights in the ledger, clients can access the global model parameters as shown in Line 6 of Algorithm 1. Clients with access to the global model parameters then update their local model parameters and iteratively fine-tune their models using their respective datasets for multiple epochs. This process continues until convergence is achieved or the maximum predefined rounds are reached.

Conversely, the server must also complete registration and authorization within the blockchain network. Algorithm 2 shows the pseudo code of the chaincode function invoked by the server after authorization. In Lines [2-4], the chaincode function accesses the model parameters of a random subset of num clients from the complete set of clients to maintain fairness across the network. To achieve this it takes help of `getClientList` which returns a list of all clients with data for the corresponding round as shown in Line 2. Now, the list is randomly reordered using a seed and the first num clients are selected. Further the clients' selection and parameter aggregation are not done by the server but instead are performed by the chaincode. This ensures fairness even when the server is malicious. The parameters are aggregated as the average of the local model weights obtained from each of the clients as shown in Lines [6-10] of Algorithm 2. Equation 2 describes this calculation for a single round. The aggregated parameters are subsequently updated on the ledger, accessible to all participating clients. This chaincode will be invoked by

the server either till convergence or a predefined number of rounds is reached.

$$\mu_{fed} = \frac{\sum_{i=1}^{i=num} \mu_u[i]}{num} \tag{2}$$

Algorithm 2 Pseudo code of chaincode invoked by server

```
1: function GetRoundData(num int, seed int)
2:     clientList ← getClientList(round)
3:     clientList ← shuffle(clientList, seed)
4:     clientList ← clientList[: num]
5:     mu_u ← getParameters(clientList)
6:     aggregate ← 0
7:     for μ_u[i] in μ_u do
8:         aggregate ← aggregate + μ_u[i]
9:     end for
10:    μ_fed ← aggregate / num
11:    issueTokens(clientList)
12:    putParameters(μ_fed)
13: end function
```

B. Integrating Token-Based Incentivization with LDP-FL

Augmenting the entire setup with a token-based incentivization mechanism enhances the practicality of the system. The clients may access the updated global parameters only if they possess a sufficient number of tokens. This controlled access is facilitated by the permissioned blockchain, wherein read and write permissions can be governed by the availability of tokens. In order to facilitate this, those clients which have been polled by the server will be awarded tokens (refer to Line 11 of Algorithm 2) that they can later use to fetch the updated global parameters from the server, thus refining their local model.

Our proposed incentivization mechanism employs a simple linear function based on the privacy budget ϵ to reward clients who share greater amounts of information. Since keeping ϵ too low results in almost total obfuscation of weights and conversely a very high ϵ implies negligible privacy, hence ϵ may be taken to lie between ϵ_{min} and ϵ_{max}. We award those clients having $\epsilon = \epsilon_{min}$ with 0.5 tokens and those having $\epsilon = \epsilon_{max}$ with 1 token. All other values of ϵ are allocated tokens between these two extremes. The following equation demonstrates this, where T is the number of tokens awarded:

$$T = 0.5 + \frac{\epsilon - \epsilon_{min}}{2(\epsilon_{max} - \epsilon_{min})} \tag{3}$$

Participants with lower values of ϵ receive fewer tokens, while those with higher ϵ values receive a proportionately greater token reward. This encourages clients to share their data more freely in order to reap the benefits of federated learning. The cost of fetching global weights for each client has been kept as 1 token.

When initializing the entire setup we allocate a few tokens to each client, to ensure some form of stability in the initial stages so that the clients have adequate tokens to fetch global parameters. Gradually they will become solely dependent on their privacy budget ϵ to obtain tokens for consumption.

V. Implementation Details

In this section, we give a detailed description of our implementation of the BTLF framework using Hyperledger Fabric (HLF), a modular and extensible open-source system specifically designed for deploying and operating permissioned blockchains [39]. In HLF, a *chaincode* is a collection of smart contracts, representing the business logic governing transactions within the network. It is run on the HLF distributed ledger network and facilitates execution of transactions. An *asset* in HLF refers to any digital or physical entity that has value and is owned or controlled by an organization or an individual. While there are several competing LDP-FL approaches available, the mechanism proposed in [23] has been used in our work due to its practical approaches towards supporting LDP-FL in real-life applications.

A. Chaincode Implementation

Within the HLF network, each client and server functions as a node in the HLF infrastructure. If a client or server's organization is not yet enrolled in the network upon logging in, the enrollment process is initiated. After the organization enrollment is completed, the client or server can proceed to enroll and register as a member or participant within that specific organization. The client and server backends are developed using Node.js, with support for HLF, and the chaincodes are written in Golang.

The chaincode asset definition is as follows:

- **Layer struct:** It stores the weights and biases for a particular layer in the neural network model of the client.
- **NeuralNetworkModel struct:** This stores an array of Layer structs, each representing a layer of the neural network model.
- **ClientData struct:**
  - **ClientID:** This field stores the Common Name (CN), which is a field in X.509 certificates used for client authentication. It serves as an identifier for a client and is also utilized as a key to store the data of a client. Typically, the CN of all clients is different across organizations.
  - **Data:** This is a NeuralNetworkModel struct. It stores the current state of the model for the particular client.
  - **Tokens:** This field stores the current available tokens with this specific client.
  - **Round:** This field stores the last round in which model parameters were pushed.
  - **RoundSeen:** This is a list of all the rounds in which the client fetched the global model parameters from the server. This list is necessary to ensure that if clients request global parameters for a round that has already been requested, tokens are not deducted.
  - **Epsilon:** This field stores the value of the privacy budget ϵ set by the client.

There are several chaincode functions, with certain ones executable by either the client, server, or both. The client has the ability to invoke the `PutData` and `GetResult` chaincode functions. The client invokes `PutData` to write local model parameters to ledger and `GetResult` to get the global parameters from the ledger. On the other hand, the `GetRoundData` chaincode function is exclusive to the server, serving the purpose of parameter aggregation and it uses `SelectSubSet` to select random num clients.

B. Client and Server Workflow

After registration, each client is assigned a total of N random samples from each of the ten digit classes, resulting in $10N$ samples for each individual client. Here N can be varied as per requirement. Upon initialization, each client's neural network model is equipped with randomly generated weights and biases. The ledger is also initialized for the client by calling the `InitLedger()` function of chaincode.

In each round, the client chooses a random value of privacy budget. It then trains the local model using the current model parameters and the local training data, which consists of $10N$ samples. The updated parameters ($params$) are then perturbed by calling the LDP function along with the privacy budget of the client. Now, the disguised weights are written to the ledger by invoking the `PutData()` function of the chaincode. This transaction is endorsed by the endorsers in accordance with the endorsement policy.

Given that the system operates asynchronously, issues such as node failure or network disturbances may cause the client to fail to update local parameters, hence its round number may fall behind. To address this, the round R_i of the client is updated as $\min((R_i)_{\text{client}} + 1, (R_i)_{\text{server}} + 1)$, where $(R_i)_{\text{client}}$ and $(R_i)_{\text{server}}$ denote the latest rounds in which the client and server wrote their parameters into the ledger respectively.

Subsequently, the server invokes the `GetRoundData(`num`,`$seed$`)` chaincode function. This function returns a random subset of num clients' local parameters for the round. The clients who have been polled, are rewarded with tokens according to the token-based incentivization function (Equation 3). Then all the local model parameters are aggregated. The aggregated parameters or the global parameters ($params$) are written to the ledger. The Pseudo code given in Algorithm 2 summarizes this step.

Once the global parameters have been updated, the clients may request these parameters from the ledger. This requires the client to have a sufficient number of tokens. If so, then the global parameters are retrieved using the `GetResult(`$serverCN$`,`$round$`)` chaincode function, which fetches the parameters for round $round$. Upon retrieval, the available tokens decrease by the cost of reading from the ledger (currently kept as 1 token). If the client requests global parameters for a round it has previously requested, the cost to read that round's global parameters will be zero. The local parameters are then updated with the global model parameters.

Consequently, if the client had adequate tokens, the local model will be the updated global model itself. Otherwise it will only be the locally trained model. At the end of the round, each client tests their model on their training dataset to obtain the accuracy. The process then repeats, starting a new round.

VI. Experimental Results

In this section, we present the results of our experiments with the implementation mentioned in the last section. The dataset and experimental setup is first described followed by the results.

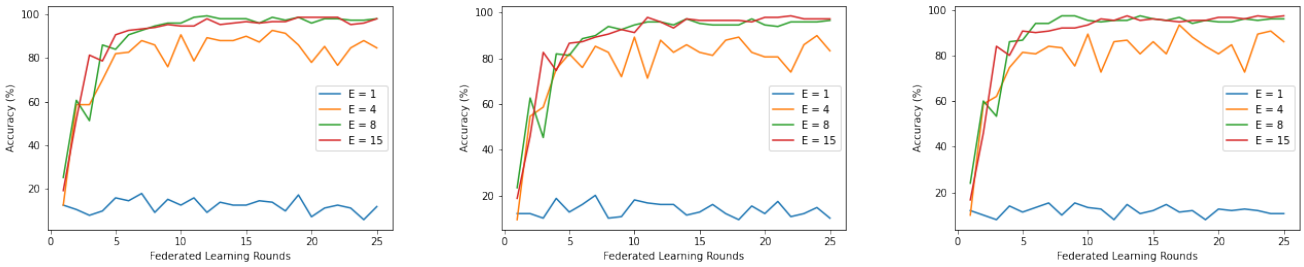
A. Model Architecture and Dataset Preparation

While the proposed architecture for BTLF is ML model and dataset agnostic, for our experiments, we have used a Convolutional Neural Network (CNN) model for image classification. The classification task considered is to identify the handwritten digits in the MNIST dataset [40], where each image is a gray scale 28×28 pixel representation of a digit (0-9). The first layer in the model is a convolutional layer with 16 filters and a 3×3 convolutional kernel. Rectified Linear Unit (ReLU) activation is applied. Next, a 2×2 max pooling layer is employed to down-sample spatial dimensions, before passing to the third layer. This is again a convolutional layer with 32 filters, a 3×3 kernel and ReLU activation. Another 2×2 max pooling operation is applied before flattening the output from the convolutional layers into a 1D array. Now a dense layer is added with 128 units and ReLU activation. Finally, the output layer is applied comprising 10 neurons, representing the 10 possible classes (digits 0-9). Softmax activation is employed to obtain the class probability distribution.

The Adam optimizer was chosen as the optimization algorithm, allowing adaptive learning rate modifications during training. Categorical cross entropy was used as the loss function since it is suitable for multi-class classification tasks with one-hot encoded labels. Additionally, the model's performance was evaluated using the accuracy metric. To train the model, we use images from the MNIST dataset. It is comprised of labeled grayscale images of handwritten digits, serving as the input data for our training process. Each image is represented as a 28×28-pixel grid, and the corresponding output signifies the label of the handwritten digit.

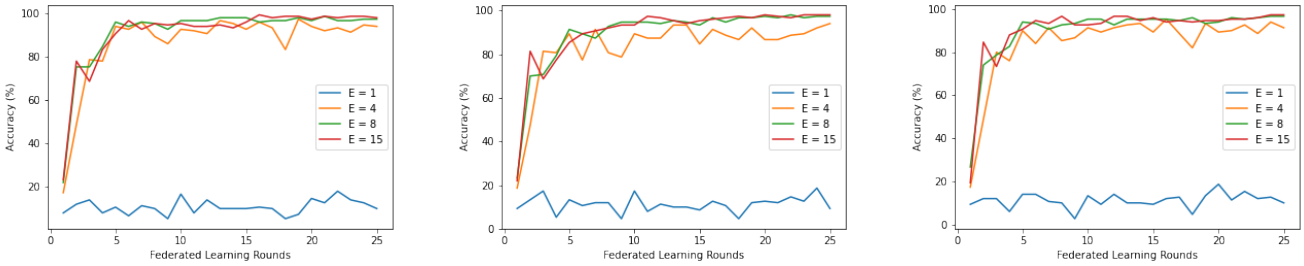
B. Detailed Results

Since there are several design parameters that can affect the accuracy of BTLF, we carry out a number of experiments. In each, some of the parameters are kept fixed while varying the rest. In Figure 2, we make each client run local training on its dataset with the current set of weights for five epochs. The updated weights are written to the ledger once these epochs are completed. There are 75 samples for each digit used in training. We show the impact of variation in the number of FL rounds on accuracy for different values of the privacy parameter ϵ. Figures 2 (a)-(c) depict the results for three representative client nodes (Client 1 - Client 3). From the figures, it is observed that while ϵ = 1 leads to poor training, as the value of ϵ is increased, accuracy goes up significantly with more number of federated learning rounds. The same



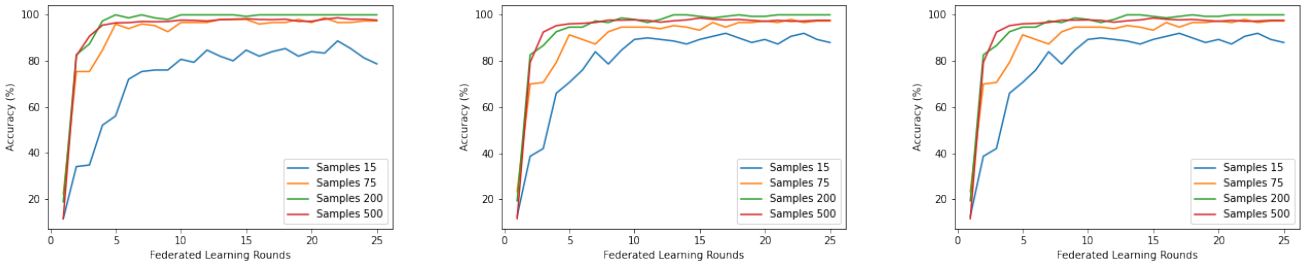
(a) Client 1 (b) Client 2 (c) Client 3

Figure 2: Model accuracy progression over Federated Learning rounds. Each subplot shows the individual accuracy trajectory of a client model trained with **5 epochs** and 75 samples per digit across different ϵ values



(a) Client 1 (b) Client 2 (c) Client 3

Figure 3: Model accuracy progression over Federated Learning rounds. Each subplot shows the individual accuracy trajectory of a client model trained with **10 epochs** and 75 samples per digit across different ϵ values



(a) Client 1 (b) Client 2 (c) Client 3

Figure 4: Variation of accuracy with training dataset size for individual clients, each trained with 10 epochs and a fixed privacy budget (ϵ=8)

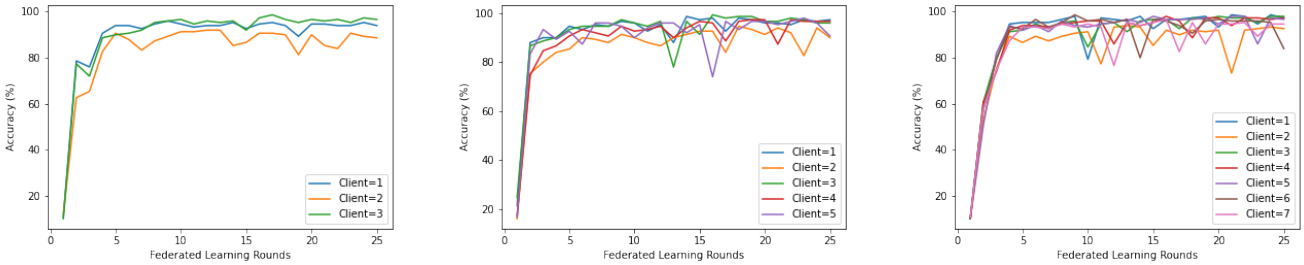
experiments were repeated by setting the number of epochs to 10 and the corresponding results are shown in Figure 3. While a trend similar to that in Figure 2 is seen, it is observed that for the same number of federated learning rounds and the value of ϵ, a higher accuracy is obtained for each client when the number of epochs used for local training is more.

We next study the impact of sample size on accuracy. From the previous two figures, it is observed that an epoch size of 10 gives satisfactory performance, and an ϵ value of 8 leads to convergence in a reasonable number of rounds. Hence, we keep these values fixed in this experiment. Figure 4 shows the variation of accuracy for different dataset sizes in terms of number of samples. The accuracy value has been plotted against the number of FL rounds. From Figures 4 (a)-(c), it is observed that the sample size has a significant impact on the accuracy upto a certain extent. Beyond a sample size of 75, there is not much impact. Also, for lower sample sizes, accuracy reaches a steady value for higher number of rounds. For all the three clients depicted in the figures, the trend is similar, implying that these are steady and reliable observation.

In the experiments so far, we used the same value of ϵ for all the clients. We next vary its value for different clients. In Figure 5(a), the ϵ values are 8, 10 and 12 for Clients 1, 2 and 3, respectively. In Figure 5(b), the values are 5, 10 and

(a) Three clients with ϵ as 8, 10, 12 (b) Three clients with ϵ as 5, 10, 15 (c) Three clients with ϵ as 1, 10, 20

Figure 5: Variation of accuracy with number of FL rounds with different clients using different privacy budgets.



(a) Three clients varying ϵ across rounds (b) Five clients varying ϵ across rounds (c) Seven clients varying ϵ across rounds

Figure 6: End-to-end working of the complete BTLF system with random values of privacy budget (ϵ) used by each client in every round.

15, while for Figure 5(c), the values are 1, 10 and 20. The intent is to study the impact of the absolute as well as relative variation in the values of ϵ across clients. From the plots, it is seen that the value of ϵ as a strong impact on accuracy. As is seen from the figures, when one of the clients (Client 1) has ϵ=1, the accuracy drops significantly. Since lower ϵ implies higher data obfuscation and hence, higher privacy, a value of 1 implies Client 1 completely perturbs its parameters while sharing. This in turn affects the global parameters to such an extent that the entire federated learning process fails.

In the final set of experiments, we not only let the clients have different privacy settings, they are also allowed to dynamically change the value of ϵ across rounds. The effect of token based incentivization is also depicted in these results. From the previous set of experiments, it was observed that ϵ values between 5 to 15 give satisfactory performance. Hence, each client chooses a random value between 5 and 15 in each round while perturbing its data. Further, considering the results in the earlier figures, we keep the number of training epochs fixed at 10 and sample size to 75. With these settings, the results are shown in Figure 6. Note that, for these experiments, we show the results for not only three clients (Figure 6 (a)) but also for five (Figure 6 (b)) and seven clients (Figure 6 (c)). The reason is to show the end-to-end working of the complete BTLF system. It is observed from the figures that, as the clients acquire tokens and later use those for reading the updated weights, the accuracy increases. However, if they obfuscate the data too much by choosing higher ϵ values, accuracy suffers. This is reflected in the dip in their accuracy values in some of the rounds.

VII. Conclusion

In this paper, we have proposed a novel approach for Locally Differentially Private Federated Learning in a permissioned blockchain setting coupled with token-based incentivization. This framework enables the construction of large-scale DNN models using multiple nodes with distinct datasets on a trustless blockchain network. Due to transmission of perturbed parameters of the local models instead of raw data or clean local model parameters, privacy of participants' data is ensured.

We have also shown how the central server typical of FL setup can be replaced by smart contracts, thereby preventing single-point failures and mitigating risks associated with potentially malicious servers or clients. By utilizing a chaincode to randomly sample a subset of nodes, it is further guaranteed that transactions are endorsed by all endorsers, adhering to the endorsement policy and there is no biased selection of clients by a potentially malicious server. Furthermore, we highlighted the asynchronous nature of the system and explained how updating the client's round number in relation to the server's round number and the client's previous round number helps reduce the challenges posed by this non-synchronicity. Future work in this field would involve use of other LDP models and possibly more sophisticated incentivization schemes.

REFERENCES

[1] P. Kairouz *et al.*, "Advances and open problems in federated learning," *CoRR*, vol. abs/1912.04977, 2019. [Online]. Available: http://arxiv.org/abs/1912.04977

[2] B. McMahan *et al.*, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[3] K. Bonawitz *et al.*, "Towards federated learning at scale: System design," *Proceedings of machine learning and systems*, vol. 1, pp. 374–388, 2019.

[4] A. Hard *et al.*, "Federated learning for mobile keyboard prediction," *ArXiv*, vol. abs/1811.03604, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:53207681

[5] E. Bagdasaryan *et al.*, "How to backdoor federated learning," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 2938–2948. [Online]. Available: https://proceedings.mlr.press/v108/bagdasaryan20a.html

[6] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.

[7] Z. Wang and Q. Hu, "Blockchain-based federated learning: A comprehensive survey," *CoRR*, vol. abs/2110.02182, 2021. [Online]. Available: https://arxiv.org/abs/2110.02182

[8] S. R. Pokhrel and J. Choi, "Federated learning with blockchain for autonomous vehicles: Analysis and design challenges," *IEEE Transactions on Communications*, vol. 68, no. 8, pp. 4734–4746, Aug 2020.

[9] K. Wei *et al.*, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020, publisher Copyright: © 2005-2012 IEEE.

[10] G. Cormode *et al.*, "Privacy at scale: Local differential privacy in practice," in *Proceedings of the 2018 International Conference on Management of Data*, ser. SIGMOD '18. New York, NY, USA: ACM, 2018, pp. 1655–1658.

[11] M. A. P. Chamikara *et al.*, "Local differential privacy for federated learning," in *Computer Security – ESORICS 2022*, V. Atluri, R. Di Pietro, C. D. Jensen, and W. Meng, Eds. Cham: Springer International Publishing, 2022, pp. 195–216.

[12] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, jan 2019. [Online]. Available: https://doi.org/10.1145/3298981

[13] General data protection regulation (GDPR). [Online]. Available: https://gdpr-info.eu/

[14] The california consumer privacy act (CCPA). [Online]. Available: https://oag.ca.gov/privacy/ccpa

[15] M. Yang *et al.*, "Local differential privacy and its applications: A comprehensive survey," *arXiv preprint arXiv:2008.03686*, 2020.

[16] Y. Gu, Y. Bai, and S. Xu, "CS-MIA: Membership inference attack based on prediction confidence series in federated learning," *Journal of Information Security and Applications*, vol. 67, p. 103201, 2022.

[17] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography*, S. Halevi and T. Rabin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 265–284.

[18] Y. Xu *et al.*, "Incentive mechanism for differentially private federated learning in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 10, pp. 6927–6939, 2022.

[19] M. Hao *et al.*, "Towards efficient and privacy-preserving federated deep learning," in *2019 IEEE International Conference on Communications (ICC)*, 2019.

[20] Y. Lu *et al.*, "Differentially private asynchronous federated learning for mobile edge computing in urban informatics," *IEEE Transactions on Industrial Informatics*, 2020.

[21] S. Truex *et al.*, "Ldp-fed: Federated learning with local differential privacy," in *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking (EdgeSys 2020)*, 2020, pp. 61–66.

[22] Y. Zhao, J. Zhao, M. Yang, T. Wang, N. Wang, L. Lyu, D. Niyato, and K.-Y. Lam, "Local differential privacy-based federated learning for internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 8836–8853, 2020.

[23] L. Sun, J. Qian, and X. Chen, "LDP-FL: Practical private aggregation in federated learning with local differential privacy," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Z.-H. Zhou, Ed. International Joint Conferences on Artificial Intelligence Organization, 8 2021, pp. 1571–1578, main Track.

[24] W. Issa *et al.*, "Blockchain-based federated learning for securing internet of things: A comprehensive survey," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–43, 2023.

[25] Y. Li *et al.*, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Network*, vol. 35, no. 1, pp. 234–241, 2020.

[26] T. Nguyen *et al.*, "Blockchain-based secure client selection in federated learning," in *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2022, pp. 1–9.

[27] L. Ouyang *et al.*, "Artificial identification: a novel privacy framework for federated learning based on blockchain," *IEEE Transactions on Computational Social Systems*, 2023.

[28] H. B. Desai, M. S. Ozdayi, and M. Kantarcioglu, "BlockFLA: Accountable federated learning via hybrid blockchain architecture," in *Proceedings of the eleventh ACM conference on data and application security and privacy*, 2021, pp. 101–112.

[29] J. A. Khan, W. Wang, and K. Ozbay, "FLOATING: Federated learning for optimized automated trajectory information storing on blockchain," in *2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2023, pp. 1–4.

[30] S. Chakraborty and S. Chakraborty, "Proof of federated training: accountable cross-network model training and inference," in *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2022, pp. 1–9.

[31] C. Meese *et al.*, "BFRT: Blockchained federated learning for real-time traffic flow prediction," in *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 2022, pp. 317–326.

[32] T. Hai *et al.*, "BVFLEMR: An integrated federated learning and blockchain technology for cloud-based medical records recommendation system," *Journal of Cloud Computing*, vol. 11, p. 22, 2022.

[33] Y. Liu *et al.*, "A secure federated learning framework for 5G networks," *IEEE Wireless Communications*, vol. 27, no. 4, pp. 24–31, 2020.

[34] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, "Protection against reconstruction and its applications in private federated learning," *arXiv preprint arXiv:1812.00984*, 2018.

[35] W. Li *et al.*, "Privacy-preserving federated brain tumour segmentation," in *Machine Learning in Medical Imaging*, H.-I. Suk, M. Liu, P. Yan, and C. Lian, Eds. Cham: Springer International Publishing, 2019, pp. 133–141.

[36] S. Kong, Y. Li, and H. Zhou, "Incentivizing federated learning," *arXiv preprint arXiv:cs.cv*, 2022.

[37] Y. Liu *et al.*, *FedCoin: A Peer-to-Peer Payment System for Federated Learning*. Cham: Springer International Publishing, 2020, pp. 125–138. [Online]. Available: https://doi.org/10.1007/978-3-030-63076-8_9

[38] L. Witt *et al.*, "Decentral and incentivized federated learning frameworks: A systematic literature review," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3642–3663, 2023.

[39] E. Androulaki *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, ser. EuroSys '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3190508.3190538

[40] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.