Federated Learning:A Distributed Learning Approach to reach Nash Equilibrium using Lagragian Convergence And Merkle tree based verification for untrusted participation

Pranav Mittal[1], Pranav Verma[1], Prathamesh Chaudhari[1], and Rishija Raj[1]

[1]Indian Institute Of Technology(BHU),Varanasi

December 14, 2021

Abstract

Federated Learning is essentially a distributed machine learning technology or machine learning framework. In cross-silo federated learning (FL), organizations cooperatively train a global model with their local data. The organizations, however, may be heterogeneous in terms of their valuation on the precision of the trained global model and their training cost. Meanwhile, the computational and communication resources of the organizations are non-excludable public goods. To motivate efficient cooperation, in this paper, we propose an incentive mechanism for cross-silo FL that addresses the challenges resulting from the private information of the organizations and the public goods feature.Moreover, based on the proposed mechanism, we propose a distributed algorithm that enables the organizations to maximize the social welfare in a decentralized manner. To prevent cheating, we propose TrustFL, a practical scheme that leverages Trusted Execution Environments (TEEs) to build assurance of participants' training executions with high confidence. Specifically, we use TEE to randomly check a small fraction of all training processes for tunable levels of assurance, while all computations are executed on the co-located faster yet insecure processor (e.g., GPU) for efficiency.

1 Introduction

Federated Learning methodologies enables machine learning models to be trained effectively while ensuring legal compliance by preserving data privacy of the participants. Federated Learning define a machine learning framework under a virtual model, which is designed to solve the problem of different data owners collaborating without exchanging data .The virtual model is the best model for all parties to aggregate data together, and their respective regions serve local targets according to the model. Federated Learning requires that the modeling result should be close to the traditional model, i.e., the data of multiple data owners are gathered in one place for modeling results. Federated learning (FL) is recently proposed to obtain a high-quality collaborative deep neural network (DNN) model without compromising data privacy . Different from traditional centralized training, it distributes training tasks into individual data owners and aggregates their locally-computed model updates as the global model. Such a framework can be widely applied in multifarious application domains, where one user without data wants to buy an accurate model and data owners are incentivized to contribute data yet have privacy concerns or are restricted by privacy regulations like EU GDPR . Take the diagnosis modeling scenario as an example. The medical researchers who need a precise disease predictive model can initiate a learning task with bounty. The clinics/hospitals with a large number of electronic health records can participate in the task, train the model locally, submit model updates, and reap rewards. . FL can be classified into two types : cross-device FL and cross-silo FL. In cross-device FL, an organization (e.g., company, institution) acts as the central server. This organization is the owner of the global model. That is, it initiates the FL and owns the trained global model. The devices are the clients and perform local training. On the other hand,in cross-silo FL, a third party entity acts as the central server and is responsible for the coordination of training.A set of organizations act as the clients to perform local training. They are also the owners of the global model and can make use of the trained global model [1]. In this work, we focus on cross-silo FL. In the cross-silo FL each organization can choose the processing capacity that it allocates for local training.The handling limits from various associations will influence the accuracy of the prepared worldwide model, i.e., the degree that the prepared worldwide model fits the nearby information of the associations. The choice of the handling limit will likewise influence the computational expense of the associations. Besides, the correspondence cost of every association relies upon how regularly it needs to trade model updates with the focal server. On the other hand, the associations might be heterogeneous as far as their valuation on the accuracy (i.e., the degree that an association is worried about the accuracy of the prepared worldwide model) just as the computational and correspondence costs. Subsequently, the determination of the handling limit must be upgraded to guarantee proficient participation, i.e., boost the social government assistance. Notwithstanding, the associations are free elements and may indeed, even be business contenders. Every association might be self interest and may not target upgrading the social government assistance. As an outcome, it is important to plan a motivation system to inspire proficient participation in cross-silo FL. We will apply the modified FedAvg algorithm for better efficiency. Federated averaging (FedAvg) is a communication efficient algorithm for the distributed training with an enormous number of clients. In FedAvg, clients keep their data locally for privacy protection; a central parameter server is used to communicate between clients. This central server distributes the parameters to each client and collects the updated parameters from clients. FedAvg is mostly studied in centralized fashions, which requires massive communication between server and clients in each communication. Moreover, attacking the central server can break the whole system's privacy. We will apply the modified FedAvg with momentum, which is implemented on clients that are connected by an undirected graph.It will be based on stochastic gradient descent. To further reduce the communication cost, we also consider the quantized FedAvg. We prove convergence of the (quantized) FedAvg under trivial assumptions; the convergence rate can be improved when the loss function satisfies the PŁ property. Finally, we numerically verify the efficacy of FedAvg. Modified FedAvg will improve the convergence rate , accuracy and security.

Our proposed solution will also take care of the various issues while building the model.First, the operation of the incentive mechanism will not rely on the private information (e.g., valuation on precision, costs) of the organizations. Second, the mechanism will also address the fact that the computational and communication resources in cross-silo FL provided by the organizations are public goods.

Under the federated mechanism, each participant has the same identity and status and can establish a shared data strategy. Since the data does not transfer, it does not reveal user privacy or affect data speci[U+FB01]cations.However, FL lacks visibility to participants' local training processes, making it vulnerable to misbehaving participants that do not execute training tasks as intended. The misbehaviours may reduce the model accuracy or impede the model convergence. Especially, in an FL-based marketplace , it may cause severe economic loss, as participants can defraud rewards with no/less training efforts. Regarding these issues, there are some defense solutions . For example, they remove/weaken outliers from model updates of participants, so as to prevent model poisoning attacks, or guarantee the convergence of the global model . Unfortunately, the above defenses only concentrate on eliminating markedly different training results mainly incurred by bad input data. None of them considers whether local training processes are correctly executed, which is of equal importance in the FL system. So, we will design the threat model using the 'commit and prove' method in which the verification of the round will be done after the completion of the training rounds.It will solve these two problems. First, assuring correct execution at participants can be combined with the above solutions ,so as to build a trustworthy FL system resisting against data poisoning attacks as well as some stealthy backdoor attacks. Second, it can demote the misused widespread application of the FL-based marketplace, since it will disable aligned transactions by preventing lazy participants with no/less training efforts. We will talk about this further in a little detail.

To make the process fair for everyone, this paper will also focus to assign penalty to defraud participants or an organization when they have been found guilty. Also, we would like to discuss the possibilities when the model will have different number of training rounds and also when the system will have non-identical datasets.

2 Solutions and contribution

To motivate efficient cooperation, in this paper, we propose a modified FedAvg algorithm for cross-silo FL that addresses the challenges resulting from the private information of the organizations and the public goods feature. The proposed mechanism helps the organizations to answer the following questions: (a) How much processing capacity should each organization allocate for local training? (b) How much should each organization be compensated (by other organizations) for its local training? Moreover, based on the proposed mechanism, we propose a distributed algorithm that enables the organizations to maximize the social welfare in a decentralized manner. We implemented the building blocks of Federated Learning (FL) and trained one from scratch on the MNIST digit data set.

3 System Model

We consider a scenario with N organizations. Let N = 0, . . . , N −1 denote the set of organizations. Each organization has its own local data. Let Sn denote the collected dataset of organization n ∈ N . Let Sn denote the number of data units in set Sn, i.e., Sn = |Sn|. The organizations train a global model using cross-silo FL to learn the collected data. Let denote the weights of the global model. The organizations aim to find the optimal weights of the global model ω ∗ that minimize the expected loss L(ω) over the datasets

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} \left\{ L(\boldsymbol{\omega}) \sum_{n\in\mathcal{N}} \frac{S_n}{\sum_{n'\in\mathcal{N}} S_{n'}} l\left(\boldsymbol{\omega}; \mathcal{S}_n\right) \right\}$$

where l(ω; Sn) is the loss over dataset Sn given ω. On the other hand, the organizations may have different valuation on the precision of the trained global model and have different computational and communication costs. Thus, some organizations may need to pay other organizations to compensate the cost of the latter ones. Otherwise, some organizations may not cooperate.

4 Our FL Model

We will apply the modified FedAvg with momentum, which is implemented on clients that are connected by an undirected graph.It will be based on stochastic gradient descent. To further reduce the communication cost, we also consider the quantized FedAvg. We prove convergence of the (quantized) FedAvg under trivial assumptions; the convergence rate can be improved when the loss function satisfies the PŁ property. Finally, we numerically verify the efficacy of FedAvg. Modified FedAvg will improve the convergence rate , accuracy and security. The data we will be using is horizontally partitioned, so we will simply be doing component wise parameter averaging which will be weighed based on the proportion of data points contributed by each participating client. Here's the federated averaging equation we are using, it comes one of the pioneering works on federated learning

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \quad \text{where} \quad F_k(w) = \frac{1}{n_k} \sum_{i\in\mathcal{P}_k} f_i(w)$$

On the right hand side, we are estimating the weight parameters for each client based on the loss values recorded across every data point they trained with. On the left, we scaled each of those parameters and sum them all component-wise. Firstly we calculate the proportion of a client's local training data with the overall training data held by all clients. First we obtained the client's batch size and used that to calculate its number of data points. We then obtained the overall global training data size. Finally we calculated the scaling factor as a fraction.Then we scale each of the local model's weights based on the value of their scaling factor calculated and then finally sums all client's scaled weight together. Finally, With 10 clients each running 1 local epoch on top of 100 global communication rounds, the truncated test result has found to have a accuracy of around 96.5 percent which is better than the 94.5 percent accuracy for the SGD model after 100 epochs.

5 Incentive mechanism design problem

Incentive Mechanism Design Problem: To motivate efficient cooperation among the organizations, we aim to design an incentive mechanism. This mechanism should be able to address the public goods feature of the resources of the organizations. We consider an incentive mechanism as follows: - Each organization $n \in \mathcal{N}$ submits a message profile (γ_n, π_n) to the central server. Message γ_n indicates the number of training rounds that organization n expects to have. Message π_n indicates the unit monetary transfer per training round that organization n expects to pay or receive. Let $\gamma = (\gamma_n, n \in \mathcal{N})$ and $\pi = (\pi_n, n \in \mathcal{N})$. - The central server computes and announces the processing capacity vector $\boldsymbol{f}(\gamma)=(f_n(\gamma), n \in \mathcal{N})$ and the monetary transfer vector $m(\gamma, \pi)=(m_n(\gamma, \pi), n \in \mathcal{N})$ We aim to design $f(\gamma)$ and $m(\gamma, \pi)$ in the incentive mechanism. Note that the actual monetary transfer to organization n (i.e., $m_n(\gamma, \pi)$) depends on the choice of $m(\gamma, \pi)$. Thus, it is not necessarily equal to the monetary transfer specified by organization n in message profile (i.e., the number of training rounds γ_n multiplied by the unit monetary transfer π_n). Given the incentive mechanism, the organizations have to decide the message profiles to submit. The strategic interaction among the organizations can be modeled as a non-cooperative game. Let $\left(\gamma^{\mathrm{NE}}, \pi^{\mathrm{NE}}\right)$ denote the NE of the game. That is, given the message profiles of other organizations, organization $n \in \mathcal{N}$ cannot increase its payoff by deviating from $\left(\gamma_n^{\mathrm{NE}}, \pi_n^{\mathrm{NE}}\right)$. We aim to design $f(\gamma)$ and $m(\gamma, \pi)$ in the incentive mechanism such that the NE of the game satisfies the following properties $\mathrm{P1}, \mathrm{P2}$, and $\mathrm{P3}$:

P1 Social efficiency: The processing capacity vector under NE, i.e., $f\left(\gamma^{\mathrm{NE}}\right)$, is the optimal solution of problem .

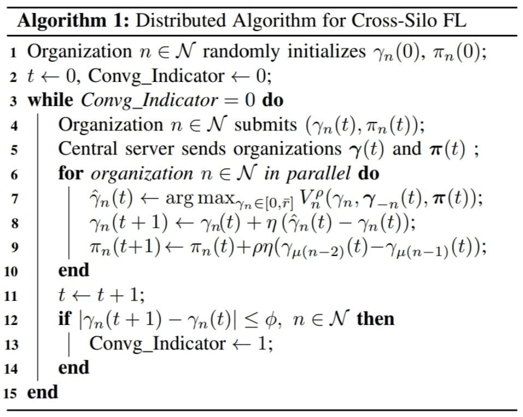
P2 Individual rationality: Each organization is no worse off through participating in cross-silo FL, i.e., $V_n\left(f_n\left(\gamma^{\mathrm{NE}}\right), r\left(\boldsymbol{f}\left(\gamma^{\mathrm{NE}}\right)\right), m_n\left(\gamma^{\mathrm{NE}}, \pi^{\mathrm{NE}}\right)\right) \geq 0$ for $n \in \mathcal{N}$

P3 Budget balance: The summation of the monetary transfer of all organizations is zero, i.e., $\sum_{n \in \mathcal{N}} m_n\left(\gamma^{\mathrm{NE}}, \pi^{\mathrm{NE}}\right)=0$. In other words, the monetary transfer can operate among the organizations without any third party investment.

6 Distributed Algorithm Design

In this section, we propose a distributed algorithm that enables the organizations to achieve the NE of Game. The proposed algorithm can address the challenges regarding the nonconvexity of problem and the unknown private information of the organizations. The main idea is to first reformulate the social welfare maximization problem. Since the saddle point of the Lagrangian of the reformulated problem is the NE of Game, we can then design a distributed algorithm that converges to the saddle point of the Lagrangian and hence the NE of Game.

Distributed Algorithm: Our proposed algorithm is inspired by the distributed accelerated augmented Lagrangian method [29], which is a distributed algorithm for achieving the saddle point of the Lagrangian of a constrained problem. We have modified the algorithm in [29] to adapt to the cross-silo FL. scenario, Specifically, we replace the notations r and λ in $\mathcal{L}(r, \lambda)$ with notations γ and π, respectively. In the proposed algorithm, the organizations aim to find the saddle point of $\mathcal{L}(\gamma, \pi)=\sum_{n \in \mathcal{N}} \mathcal{L}_n\left(\gamma_n, \pi\right)=\sum_{n \in \mathcal{N}} V_n\left(f_*\left(\gamma_n\right), \gamma_n, m_n\left(\eta_n 1, \pi\right)\right)$ The obtained saddle point is the NE of Game 1 .



```
Algorithm 1: Distributed Algorithm for Cross-Silo FL
1  Organization $n \in \mathcal{N}$ randomly initializes $\gamma_n(0)$, $\pi_n(0)$;
2  $t \leftarrow 0$, Convg_Indicator $\leftarrow 0$;
3  while Convg_Indicator $= 0$ do
4      Organization $n \in \mathcal{N}$ submits $(\gamma_n(t), \pi_n(t))$;
5      Central server sends organizations $\boldsymbol{\gamma}(t)$ and $\boldsymbol{\pi}(t)$ ;
6      for organization $n \in \mathcal{N}$ in parallel do
7          $\hat{\gamma}_n(t) \leftarrow \arg\max_{\gamma_n \in [0,\bar{r}]} V_n^{\rho}(\gamma_n, \boldsymbol{\gamma}_{-n}(t), \boldsymbol{\pi}(t))$;
8          $\gamma_n(t+1) \leftarrow \gamma_n(t) + \eta\left(\hat{\gamma}_n(t) - \gamma_n(t)\right)$;
9          $\pi_n(t+1) \leftarrow \pi_n(t) + \rho\eta(\gamma_{\mu(n-2)}(t) - \gamma_{\mu(n-1)}(t))$;
10     end
11     $t \leftarrow t+1$;
12     if $|\gamma_n(t+1) - \gamma_n(t)| \leq \phi,\ n \in \mathcal{N}$ then
13         Convg_Indicator $\leftarrow 1$;
14     end
15 end
```

Figure 1: algorithm 1

The proposed algorithm is given in Algorithm 1. where the organizations update message profiles for multiple iterations until the algorithm converges, Let $t \in Z_{+}$denote the iteration index. Each organization n first randomly initializes its message profile $\left(\gamma_n(0), \pi_n(0)\right)$. While the convergence indicator ConvIndicator $=0$, each organization n submits message profile $\left(\gamma_n(t), \pi_n(t)\right)$ in iteration t. Then, the central server sends the submitted message profiles to the organizations. Steps $7-9$ correspond to how each organization n updates its message profile. In Step 7. organization n first computes [U+20B9]$_n(t)$ by deriving the value of $\gamma_n \in[0, \vec{r}]$ that maximizes

$$
\begin{aligned}
V_n^\rho\left(\gamma_n, \gamma_{-n}, \pi\right)= & V_n\left(f_n\left(\gamma_n\right), \gamma_n, m_n\left(\gamma_n \mathbf{1}, \pi\right)\right) \\
& -\rho \sum_{n \in N}\left(\gamma_\mu(n-2)-\gamma_{\mu(n-1)}\right)^2
\end{aligned}
$$

where ρ is a penalty coefficient. The second term can be regarded as a term for penalizing the different number of training rounds submitted by the organizations $\left(i \cdot e_n \gamma_{\mu(n-2)} \neq \gamma_{p(n-1)}\right.$ for $\left.n \in \mathcal{N}\right\rangle$ In Steps 8 and 9 , organization n computes the updated message profile $\left(\gamma_n(t+1), \pi_n(t+1)\right)$ by considering a step size $\eta \in(0,1)$. A larger η implies a more aggressive update. An intuition behind Step 9 is as follows. Suppose the number of training rounds submitted by organization $\mu(n-2)$ is much larger than that submitted by organization $\mu(n-1)$ (i.e., the difference $\gamma_\mu(n-2)(t)-7 \mu(n-1)(t)$ is large). Then, $\pi_n(t+1)$ is large based on Step 9. According to (11), the lame value of $\pi_n(t+1)$ will lend to a small monetary transfer to organization $\mu(n-2)$ and a large monetary transfer to organization $\mu(n-1)$. Hence, in the next iteration, organizations $\mu(n-2)$ and $\mu(n-1)$ will reduce and increase the number of training rounds that they submit, respectively. The algorithm terminates when the absolute difference between 7un$(t+1)$ and $\gamma_n(t)$ for all $n \in \mathcal{N}$ is smaller than a predefined threshold ϕ.

7 Results

Monetary transfer of the 10 organisations we trained are as follows: [0.03598416473487054, 0.038631880465888724, 0.035673221615228456, 0.03468094833334501, 0.033872066376512244, -0.019310889512137996, -0.007881981282104988, -0.03647497250979781, -0.057253940050642105, -0.05792049817116207]

We can see that the total monetary transfer is 0 i.e. no organisation is worse than other organization.

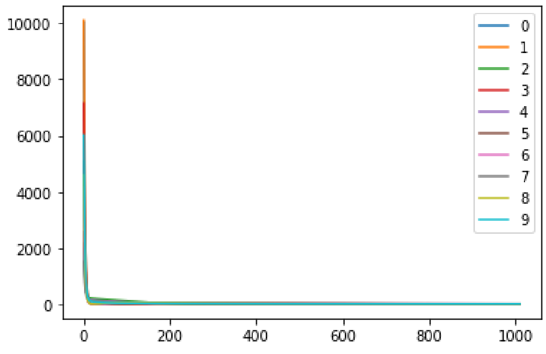


Figure 2: rounds(Homogenous Profile (Utility is same))

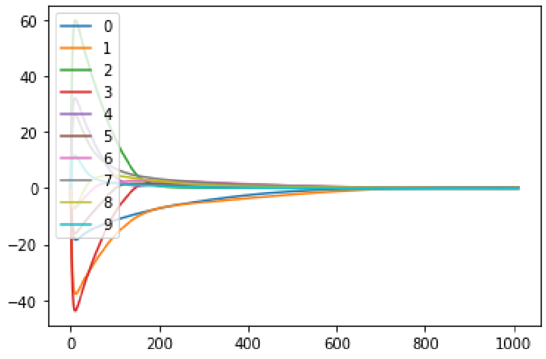


Figure 3: cost(Homogenous Profile (Utility is same))

Iterations	convergence
1	17033.33
100	3.26
200	1.78
300	1.019
400	0.552
515	0.0115

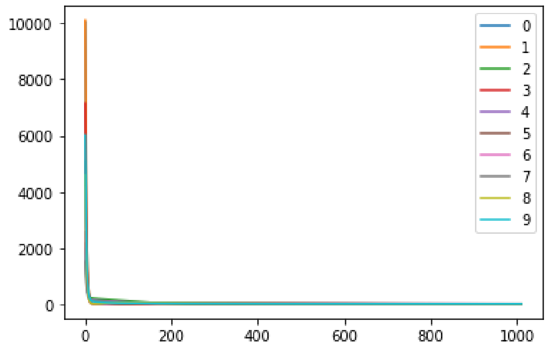


Figure 4: rounds(Non Homogenous Profile (Utility is 5 and 10 for 5 organizations))

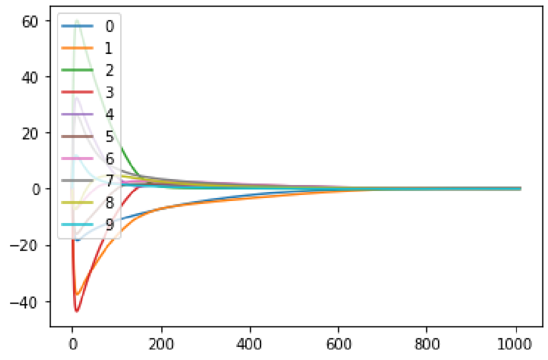


Figure 5: cost(Non Homogenous Profile (Utility is 5 and 10 for 5 organizations))

Iterations	convergence
1	15980.309
100	3.241
200	1.151
300	0.460
400	0.386
500	0.316
600	0.255
700	0.089
800	0.394
900	0.378
1010	$8.658e-05$

8 Enabling execution of federated learning at untrusted participants

We propose TrustFL, a practical scheme that leverages Trusted Execution Environments (TEEs) to build assurance of participants' training executions with high confidence. Specifically, we use TEE to randomly check a small fraction of all training processes for tunable levels of assurance, while all computations are executed on the co-located faster yet insecure processor (e.g., GPU) for efficiency. To prevent various cheating behaviors like only processing TEE-requested computations or uploading old results, we devise a commitment-based method with specific data selection. We design a model in trusted execution environment to prevent model poisoning attacks from untrusted participants which can reduce the efficiency of global model [2].

Working of the Model:

We use a "commit and prove" method in which all training rounds are completed before sampling. We then denote the programs running outside the enclave and inside the enclave as Prog o and Prog i respectively. Prog o stores the hash digests of all model parameters theta at the end of each training session, and sends a commit message to the TEE when all training rounds are completed. Prog i requests to check one random training round. Prog i checks whether these parameters are consistent with the parameters of Prog o. If program is successfully executed, it means participants have honestly performed tasks else it will return an error.

Designing binary tree to store hash values of model parameters:

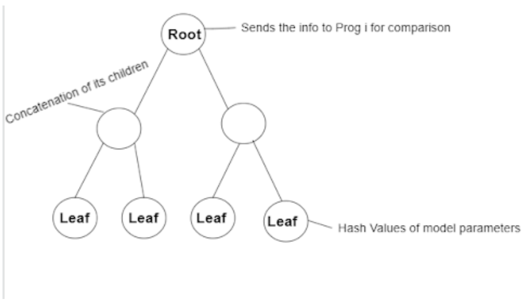
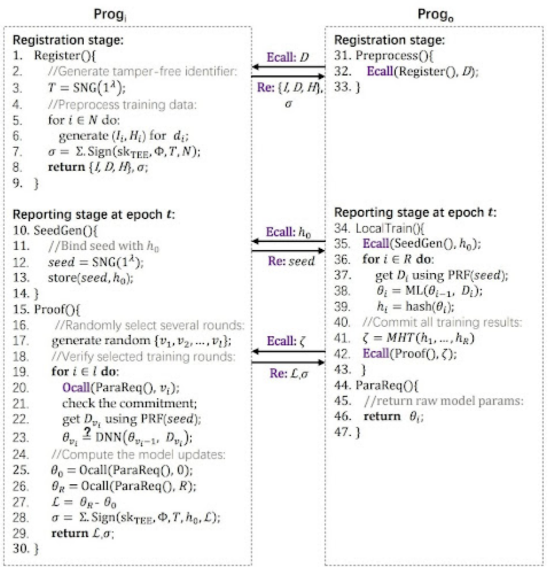

Root
Sends the info to Prog i for comparison
Concatenation of its children
Leaf
Leaf
Leaf
Leaf
Hash Values of model parameters

Figure 6: Merkle tree, a complete binary tree, where leaf nodes are hash values of model parameters in the order

To gain constant performance inside the SGX regardless of the number of training rounds, we adopt the Binary hash tree based commitment method (denoted as BHTbased method). Specifically, the Progo builds a Merkle tree, a complete binary tree, where leaf nodes are hash values of model parameters in the order. The internal nodes are the hash values of the concatenation of their two children. The tree root is submitted to Progi as the commitment. Later, Progi generates several random numbers to identify the training rounds to be verified. Then Progi requests the corresponding input parameters, the hash value of the output parameters, and auxiliary information to check the correctness of the commitment.

Compared to the original SH-based method, the storage overhead of MHT-based method in the SGX reduces to O(1) from O(n) which would never induce SGX paging overhead. Yet, it requires more hash operations with the complexity of O(log n) when checking the commitment.



```
Prog_i

Registration stage:
1.  Register(){
2.      //Generate tamper-free identifier:
3.      $T = \mathrm{SNG}(1^\lambda)$;
4.      //Preprocess training data:
5.      for $i \in N$ do:
6.          generate $(I_i, H_i)$ for $d_i$;
7.      $\sigma = \Sigma.\mathrm{Sign}(sk_{\mathrm{TEE}}, \Phi, T, N)$;
8.      return $\{I, D, H\}, \sigma$;
9.  }

Reporting stage at epoch $t$:
10. SeedGen(){
11.     //Bind seed with $h_0$
12.     $seed = \mathrm{SNG}(1^\lambda)$;
13.     store($seed, h_0$);
14. }
15. Proof(){
16.     //Randomly select several rounds:
17.     generate random $\{v_1, v_2, \ldots, v_l\}$;
18.     //Verify selected training rounds:
19.     for $i \in l$ do:
20.         Ocall(ParaReq(), $v_i$);
21.         check the commitment;
22.         get $D_{v_i}$ using PRF($seed$);
23.         $\theta_{v_i} \overset{?}{=} \mathrm{DNN}(\theta_{v_i - 1}, D_{v_i})$;
24.     //Compute the model updates:
25.     $\theta_0 = \mathrm{Ocall(ParaReq()}, 0)$;
26.     $\theta_R = \mathrm{Ocall(ParaReq()}, R)$;
27.     $\mathcal{L} = \theta_R - \theta_0$
28.     $\sigma = \Sigma.\mathrm{Sign}(sk_{\mathrm{TEE}}, \Phi, T, h_0, \mathcal{L})$;
29.     return $\mathcal{L}, \sigma$;
30. }

Prog_o

Registration stage:
31. Preprocess(){
32.     Ecall(Register(), $D$);
33. }

Reporting stage at epoch $t$:
34. LocalTrain(){
35.     Ecall(SeedGen(), $h_0$);
36.     for $i \in R$ do:
37.         get $D_i$ using PRF($seed$);
38.         $\theta_i = \mathrm{ML}(\theta_{i-1}, D_i)$;
39.         $h_i = \mathrm{hash}(\theta_i)$;
40.     //Commit all training results:
41.     $\zeta = MHT(h_1, \ldots, h_R)$
42.     Ecall(Proof(), $\zeta$);
43. }
44. ParaReq(){
45.     //return raw model params:
46.     return $\theta_i$;
47. }
```

Figure 7: The algorithm for assuring local training execution with TEE. All training rounds are conducted by the Progo at the host participant. Several random-selected rounds are recomputed and checked by the Progi .

To guarantee the freshness of a single training round and final results, we use dynamic-yet-deterministic data selection in each training round. Prog o and Prog i uses same training function. Prog i generates an index and HMAC for each training data which would be restored outside the enclave before training. When conducting local training, Prog o requests a seed from Prog i and each training data at each round (e.g., i) is selected as follows: Iij = PRF(seed × i + j) mod As for cybil based cheating, we eliminate it by enforcing attackers to consume computing resources proportionally to the bounties. We easily use a unique random nonce T as the tamper-free identifier for the participants. This will ensure data of training rounds are different.

9 Assigning Penalty to an Organization

Developing a mechanism to assign a penalty to an organization when it has been found guilty. This will ensure equality for all and will be advantageous to all the organizations that would fairly participate in the competition. This will also be a lesson for those organizations who participated unfairly/used unfair methods in the competition.

Types of Penalty:

- For Sybil Based Cheating.
- One Time Penalty.
- For Unfresh training results.
- For defraud and data manipulation.

Thoughts to work on some assumptions:

- Organisations can choose the number of training rounds that they participate in according to their valuation on their precision and their computational and communication costs.
- We can explore the non-independent and identically distributed(non i.i.d) data of the organizations.

10 Conclusion

In this work, we proposed an incentive mechanism using the Modified FedAvg Algorithm for cross silo FL that addresses the public goods feature. The proposed mechanism can achieve social efficiency, individual rationality, and budget balance. We further proposed a distributed algorithm that enables the organizations to achieve social efficiency without knowing the private information of each other. The simulation results with MNIST dataset show that the proposed algorithm can achieve faster convergence than the conventional Lagrangian method. We also propose TrustFL, a practical scheme to enhance the FL system with assured training execution at participants. While running the entire training computation inside the TEE leads to a severe drop in performance, we use GPU to execute all rounds of training for efficiency and leverage the TEE to ensure whether outside training is correctly executed with tunable levels of assurance.We also use the merkle binary tree to store the hash values of model parameters which reduces it's time complexity. We conduct both theoretical analysis and extensive experiments. They demonstrate that TrustFL can achieve comparable performance with that on GPU, which is about one/two orders of magnitude faster than that fully on SGX. We also Developed a mechanism to assign a penalty to an organization when it has been found guilty.

Acknowledgements

We thank Dr. Hari Prabhat Gupta , SM-IEEE Assistant Professor of Dept. of Computer Science and Engineering, Indian Institute of Technology (BHU), Varanasi, INDIA for providing valuable input and guidance.

References

[1] Ming Tang and Vincent WS Wong. An incentive mechanism for cross-silo federated learning: A public goods perspective. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2021.

[2] Xiaoli Zhang, Fengting Li, Zeyu Zhang, Qi Li, Cong Wang, and Jianping Wu. Enabling execution assurance of federated learning at untrusted participants. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 1877–1886. IEEE, 2020.