Refiner: A Reliable Incentive-Driven Federated Learning System Powered by Blockchain

Zhebin Zhang, Dajie Dong, Yuhang Ma, Yilong Ying
Dawei Jiang, Ke Chen, Lidan Shou, Gang Chen
College of Computer Science & Technology, Zhejiang University
{zhebinzhang,dajiedong,myh0032,ylying,jiangdw,chenk,should,cg}@zju.edu.cn

ABSTRACT

Modern mobile applications often produce decentralized data, i.e., a huge amount of privacy-sensitive data distributed over a large number of mobile devices. Techniques for learning models from decentralized data must properly handle two natures of such data, namely privacy and massive engagement. Federated learning (FL) is a promising approach for such a learning task since the technique learns models from data without exposing privacy. However, traditional FL methods assume that the participating mobile devices are honest volunteers. This assumption makes traditional FL methods unsuitable for applications where two kinds of participants are engaged: 1) self-interested participants who, without economical stimulus, are reluctant to contribute their computing resources unconditionally, and 2) malicious participants who send corrupt updates to disrupt the learning process. This paper proposes Refiner, a reliable federated learning system for tackling the challenges introduced by massive engagements of self-interested and malicious participants. Refiner is built upon Ethereum, a public blockchain platform. To engage self-interested participants, we introduce an incentive mechanism which rewards each participant in terms of the amount of its training data and the performance of its local updates. To handle malicious participants, we propose an audit scheme which employs a committee of randomly chosen validators for punishing them with no reward and preclude corrupt updates from the global model. The proposed incentive and audit scheme is implemented with cryptocurrency and smart contract, two primitives offered by Ethereum. This paper demonstrates the main features of Refiner by training a digit classification model on the MNIST dataset.

PVLDB Reference Format:
Zhebin Zhang, Dajie Dong, Yuhang Ma, Yilong Ying
Dawei Jiang, Ke Chen, Lidan Shou, Gang Chen. Refiner: A Reliable Incentive-Driven Federated Learning System Powered by Blockchain.
PVLDB, 14(12): 2659 - 2662, 2021.
doi:10.14778/3476311.3476313

1 INTRODUCTION

There is an increasing interest in developing techniques for learning models from decentralized data distributed across a large number of mobile devices [7]. Models learned from such data can greatly improve intelligent behaviors of mobile applications such as image classification and speech recognition. However, learning from decentralized data poses two challenges: 1) data privacy, i.e., sensitive data stored on mobile devices are disallowed to be uploaded to a centralized location for learning with conventional approaches; 2) massive engagement, i.e., given that the average number of examples in each mobile device is small, participation of a large number of mobile devices is a must for constructing a reasonably sized training dataset.

Federate learning (FL) [7], originally developed by Google, is an emerging technique for learning from decentralized data. To learn a shared global model, instead of storing all data in a centralized server for learning, FL first performs training on local data at each participating mobile device and then aggregates these local updates to obtain a global model. This expose-local-updates strategy instead of expose-raw-data is proven to solve the data privacy challenge reasonably well [3]. The assumption of a collaborative group with a common goal of obtaining high-quality models, however, makes traditional FL methods unsuitable for applications where self-interested and malicious participants are engaged [8]. This is because traditional FL methods assume that participating mobile devices are willing to contribute their computing resources to the learning task as honest volunteers who strictly follow the learning protocol. Therefore, traditional FL methods do not provide mechanisms for handling two kinds of participants: 1) *self-interested participants* who, without proper economical stimulus, have no motivation to contribute their computing resources unconditionally, and 2) *malicious participants* who deviate from the learning protocol and send corrupt local updates to poison the global model.

This paper proposes Refiner, a reliable federated learning system, to tackle the challenges introduced by engaging self-interested and malicious mobile devices. Refiner is built upon Ethereum, a public blockchain platform, and extends traditional FL approaches with a well-designed incentive mechanism and an audit scheme. We assume self-interested participants are rational players whose goal is to maximize profit. To engage self-interested participants, our incentive mechanism rewards each participant with cryptocurrencies based on the amount of data it owns and the performance of its local model updates. The more quality data that a participant contributes, the more rewards that the participant will receive. Although DeepChain [8] also proposes to enhance FL with blockchain-based incentive mechanism, their method only enforces the correct behaviors of the participants, without taking into consideration the quality of the local model updates. In contrast, we reward participants based on both data volume and model quality, which makes our approach more resilient to low-quality model updates.

Proceedings of the VLDB Endowment, Vol. 14, No. 12 ISSN 2150-8097.
doi:10.14778/3476311.3476313

To prevent malicious participants from disrupting the system, our audit scheme, implemented as smart contracts, traces the learning process on the blockchain and punishes participants who deviate from the learning protocol with no rewards. To preclude corrupt updates from the global model, a committee of validators, randomly chosen from participating mobile devices, are employed for checking the quality of local updates. Only qualified local updates are accepted for updating the global model.

The rest of this paper is organized as follows. Section 2 presents an overview of Refiner, describing participants, adversary model and the system architecture. Section 3 presents the plan for demonstrating the major functions of Refiner.

2 SYSTEM OVERVIEW

This section presents an overview of Refiner. The key ideas of Refiner are summarized as two points: 1) an audit scheme which employs a committee of randomly chosen validators for evaluating and aggregating local model updates and 2) an incentive scheme which utilizes smart contracts to reward participating mobile devices according to their contributions. We can prove that as long as a majority of the validators in the committee are honest, Refiner can reliably engage self-interested mobile devices and prevent malicious participants from disrupting the learning task. Due to space limitations, this paper presents the design of Refiner. Algorithm details and analysis will be presented in a separate research paper. We shall refer to a smart contract as a contract and use the two terms interchangeably. Below, Section 2.1 describes different participants in Refiner. Section 2.2 presents the adversary model. Section 2.3 presents Refiner's system architecture and workflow.

2.1 Roles

There are four kinds of participants in Refiner, listed as follows.

Administrator. There is only one administrator in Refiner. The administrator is responsible for setting up an entry point, e.g., a website, for participants to access the system and maintaining a registry smart contract for posting and querying FL tasks. The administrator is not involved in any FL task.

Requester. For each FL task, there is only one requester who is responsible for submitting the task and paying rewards to workers and validators. A requester is not expected to have any data for training, but is required to provide validation datasets which serve as the standards for evaluating the quality of the local model updates produced by workers. Since the quality evaluation has direct influences on the model aggregation process, the validation datasets should be carefully chosen to reflect the requester's demand.

Worker. Workers are mobile devices that collectively learn a global model from their local data. Workers in Refiner are analogous to clients in traditional FL systems. We assume that a massive number of self-interested workers, who seek to maximize their profits, will engage in an FL task.

Validator. Validators are mobile devices responsible for evaluating the quality of local model updates and aggregating qualified local updates into a global model. Validators also receive rewards from the requester. Each mobile device in Refiner can register either as a worker or as a validator. But Refiner disallows a mobile device to register as both a worker and a validator simultaneously.

2.2 Adversary Model

Refiner handles three kinds of malicious participants: cheating requester, malicious workers and compromised validators.

Cheating Requester: Traditional FL methods employ the requester for aggregating local model updates and producing the final global model. Such an approach suffers from the *payment default attack*. After the global model is produced, a cheating requester may leave the system without paying participants. Refiner deals with payment default attack by wrapping the global model retrieval and payment as a single transaction and relying on smart contracts for processing this transaction.

Malicious Workers: Malicious workers are mobile devices that produce fabricated or corrupt local model updates to defraud rewards. Refiner handles malicious workers by randomly choosing a committee of validators to evaluate the performance of the submitted model updates on the validation dataset provided by the requester. Rewards are only paid to workers who produce qualified local model updates.

Compromised Validators: Compromised validators are mobile devices which produce incorrect evaluation results. In Refiner, compromised validators come in two ways: 1) *Sybil attack*, i.e., an attacker fabricates a large number of aliases to join the system as validators; 2) *fake evaluation results attack*, i.e., the compromised validator steals the evaluation results from an honest validator and sends the stolen results to cheat for rewards. Refiner employs a validator committee selection scheme which is similar to the block proposer selection algorithm presented in Algorand [2] to tackle the Sybil attack and a cryptographic commit scheme to prevent fake evaluation results.

2.3 System Architecture

The architecture of Refiner is depicted in Figure 1. Our system is implemented on top of Ethereum, a prevalent public blockchain platform[1]. At system startup, the administrator deploys a *registry contract* on Ethereum. The registry contract serves as the catalog of registered FL tasks, workers, and validators and provides functions such as task status updating and participants registration. Refiner learns a model from data stored on mobile devices in iterative rounds. During learning, participants need to privately exchange data which is unsuitable to be stored on blockchain. Thus we employ a secure data sharing scheme for persisting these data to IPFS [1], a decentralized storage system[2]. We describe the data sharing scheme first. Then, we present each step of a learning round in detail.

2.3.1 Data Sharing. We employ a double encryption scheme for a participant to share data with target receivers. The encryption scheme requires that each participant in the system has a public-private key pair. Suppose a worker wants to share its model updates w with specific validators, the worker first encrypts w with a symmetric key $\mathcal{K}$ as $\mathsf{enc}(w, \mathcal{K})$. Then the worker stores $\mathsf{enc}(w, \mathcal{K})$ in IPFS with a file handle $H^{w,\mathcal{K}}$. Finally, the worker encrypts $\mathcal{K}$ with

[1] Although Ethereum is adopted in our system implementation, the incentive and audit scheme we proposed can actually be implemented in any blockchain platform that supports cryptocurrency and smart contracts.

[2] Similar to the case of selecting Ethereum as our blockchain platform, our secure data sharing scheme can also be implemented on other distributed storage systems.

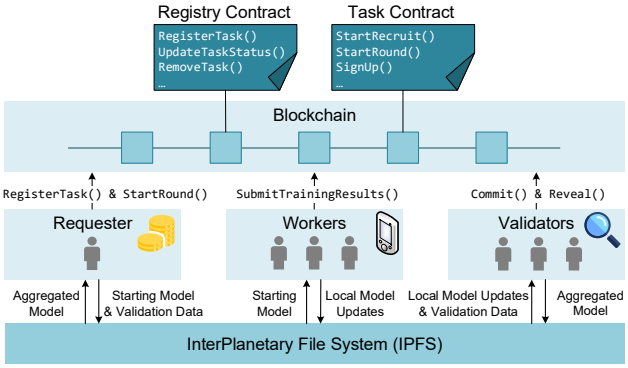

Figure 1: Refiner Architecture

the public key of each validator and shares the encrypted key and the file handle with the target validator through blockchain.

2.3.2 Task Registration. A requester launches a new FL task by deploying a task contract on the blockchain. The task contract specifies: 1) the *task description*, e.g., data format, target performance, the numbers of workers and validators; and 2) the *quantities of rewards* that will be paid to workers and validators in each round. Notably, Refiner allows the requester to choose the global model updating method from two available algorithms: 1) `FedAvg` [7] and 2) `FedProx` [5]. After the task contract is deployed, the requester invokes `RegisterTask(addr)` to register the address of the deployed task `addr` to the registry contract. The status of the newly created task is `PENDING`.

2.3.3 Member Recruitment. Mobile devices register their roles, namely *workers* or *validators*, to Refiner with their Ethereum addresses. After a new FL task T is submitted, the requester changes the task status to `RECRUITING`. Then, registered mobile devices will receive the new task notification via the Ethereum event handling mechanism. After that, interested workers join the task T by invoking the `SignUp()` function of the task contract. Meanwhile, registered validators perform the committee selection algorithm to choose a committee of validators for T. We employ the same committee selection algorithm as the one proposed in Algorand [2]. The algorithm employs a verifiable random function and a cryptographic sortition scheme to choose committee members from registered validators based on their Ethereum account balance. The committee selection algorithm is proven to address the Sybil attack [2]. After a sufficient number of participants are enrolled, the status of the task T is changed to `PENDING`.

2.3.4 Round Initialization. In this step, the requester prepares a starting global model w and a validation dataset D for the workers and the validators committee respectively. The requester securely shares these data with the intended recipients through the data sharing scheme presented in Section 2.3.1. The requester invokes the `StartRound()` function of the task contract with the file handles and encrypted keys. Then the task status is changed to `TRAINING`.

2.3.5 Local Training. Each worker i retrieves the global model w from IPFS with the file handle and decrypts it with the key. Next, the worker updates the global model w on its local dataset. After computing the local model updates w_i, the worker i securely stores w_i in IPFS and shares w_i with the validators committee by invoking `SubmitTrainingResults()` with the file handle and encrypted keys of w_i. After all workers have committed their training results or a timer has expired, the status of the task becomes `EVALUATING`.

2.3.6 Model Evaluation and Aggregation. Each validator in the validators committee retrieves the validation dataset and local model updates using the file handles and encrypted keys. The validator first evaluates each local model w_i on the validation dataset D by computing $\mathcal{L}(D; w_i)$ where $\mathcal{L}$ is the loss function. To prevent corrupt local model updates, only qualified local model updates are accepted. The local model update w_i is qualified only if w_i satisfies $\mathcal{L}(D; w_i) \leq \epsilon$ where ϵ is the *model qualification threshold* specified in the task description. After all local model updates are processed, the validator updates the global model by aggregating qualified local model updates: $w \leftarrow \sum_i \frac{n_i}{n} w_i$ where n_i is the number of training examples from worker i and n is the total number of examples from all workers in this learning round. Next, the validator calculates the worker's contributions in terms of *marginal model performance loss*. Specifically, for each worker j, the validator computes an aggregated model $w^j \leftarrow \sum_{i \neq j} \frac{n_i}{n} w_i$ without worker j's updates. Then, the contribution of worker j is determined by $\mathcal{L}_j = \mathcal{L}(D; w) - \mathcal{L}(D; w^j)$. Next, the validator ranks the workers according to their contributions. Finally, each validator i securely shares the aggregated model w with the requester and commits to the task contract a manifest $\mathcal{M}_i$ consisting of the evaluation results, the file handle of the aggregated model and its encrypted key.

To tackle compromised validators who steal manifests from others, we adopt the cryptographic commit scheme [6]. The scheme consists of two phases: 1) *commit* and 2) *reveal*. In the commit phase, each validator i generates a commitment of $\mathcal{M}_i$ as $c_i = \mathcal{H}(\mathcal{M}_i, s_i)$ with a fresh salt s_i and sends c_i to the task contract. After all commitments are submitted or a timer expired, the reveal phase starts and the task status is changed to `REVEALING`. Each validator sends $\mathcal{M}_i$ and s_i to the task contract. After all manifests are submitted or a timer expired, the task status is changed to `REWARDING`.

2.3.7 Reward Distribution. The task contract verifies each manifest by checking $c_i = \mathcal{H}(\mathcal{M}_i, s_i)$. A majority rule is adopted to determine a consensus manifest. Validators who produce manifests that agree with the consensus manifest will be rewarded with an equal amount of Ether. Workers are rewarded according to their contributions as recorded in the consensus manifest.

2.3.8 Task Termination. In the final step, the requester retrieves the global model from IPFS. The whole FL task terminates if any one of the following conditions is satisfied: 1) the task has run for a predefined number of rounds, 2) the performance of the global model has reached a pre-defined target, and 3) the requester explicitly invokes the `StopTraining()` function of the task contract. Once the task terminates, the status of the task is changed to `TERMINATED` and the task is removed from the registry contract.

3 DEMONSTRATION

We demonstrate the main features of Refiner by training a digital classification model on the MNIST dataset[4], i.e., running FL tasks, rewarding participants, and handling malicious behavior. Refiner

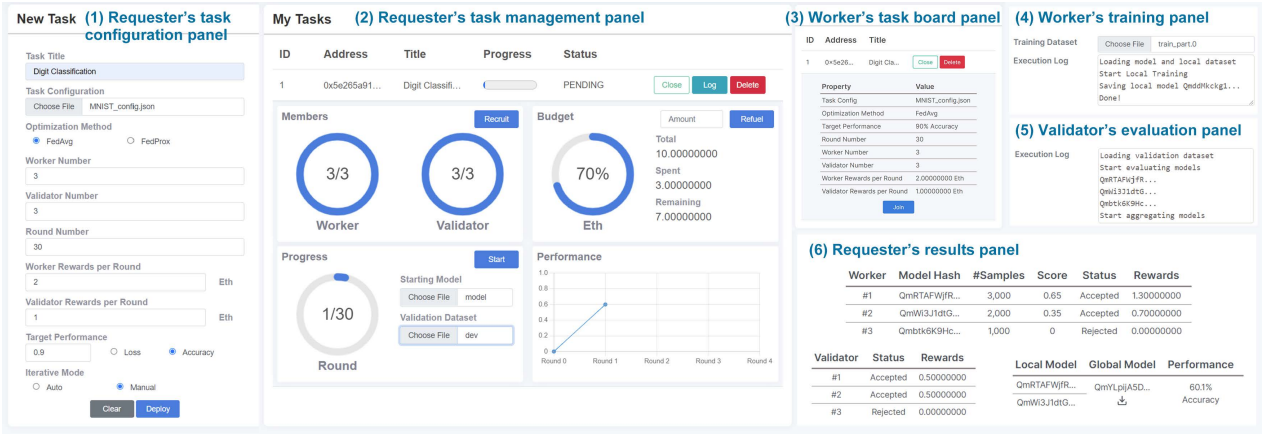

Figure 2: Refiner User Interface

supports two learning modes: *auto* and *manual*. We invite the audience to interact with the system in each step of the training process in the manual learning mode. The demonstration plan is as follows:

Environment. The demonstration is hosted in cloud. A private test-bed Ethereum blockchain is setup beforehand. We use NFS instead of IPFS as the storage module. Five workers and eight validators are run inside a virtual cloud server.

Task Registry. We first log in to Refiner as a requester and submit a new FL task to the system. The audience will specify configuration items of the task through the *task configuration panel* (Figure 2(1)). Once the task is successfully created, its task contract will be deployed on the blockchain. After that, the audience can view the task in the *task management panel* (Figure 2(2)). The panel shows detailed information of the task, including progress of learning, budgets, members and global model performance.

Member Recruitment. In the second step, the audience will recruit members for the submitted FL task through the *task management panel*. Then, we log in to Refiner as a worker. Figure 2(3) shows the *task board panel* for the worker. The validators committee, on the other hand, is automatically selected. Thus, the committee selection scheme is not demonstrated in a step-by-step manner.

Round Initialization. The audience starts a new learning round with the *task management panel* (Figure 2(2)). To make the demonstration interesting, we first invite the audience to supply some rewards. Then, the audience will feed a starting model and a validation dataset into the *progress subpanel* and launch the round.

Local Training. In this step, the audience views tasks that require training through worker's *training panel* (Figure 2(4)). We prepare a good dataset with correct labels and a bad dataset with intentionally shuffled labels. The audience can choose one for training and then submit the local model updates.

Model Evaluation and Aggregation. We now log in to Refiner as a validator and guide the audience to evaluate the performance of local model updates produced by workers. Figure 2(5) shows the *evaluation panel* for the validator. The validator first evaluates the performance of all workers' local model updates and then aggregates qualified local updates into a global model. The contribution of each worker is calculated and the results are submitted to the blockchain. If the audience chooses the bad dataset in the previous step, the worker will be deemed malicious and its local updates will be rejected.

Reward Distribution. Reward distribution is performed after the consensus manifest is determined. The rewards in the task contract will be transferred to the accounts of the qualified workers and validators. We will show the amount of rewards that they receive (Figure 2(4) and Figure 2(5)). We will also log in as the requester and show the results of the learning round (Figure 2(6)). The audience will see that the rewards paid to workers are proportional to the contributions made by those workers. Malicious workers whose local model updates are rejected and compromised validators whose evaluation results are inconsistent will not be paid.

Task Termination. In the final step, the performance of a learning curve will be presented to the audience (Figure 2(2)). We shall end the demonstration by retrieving the global model and using the model to perform digit classification.

ACKNOWLEDGMENTS

This work was supported by the Key Research and Development Program of Zhejiang Province of China (Number 2021C01009), National Natural Science Foundation of China (Number 62050099), and Fundamental Research Funds for the Central Universities.

REFERENCES

[1] J. Benet. 2014. IPFS - Content Addressed, Versioned, P2P File System. *CoRR* abs/1407.3561 (2014). arXiv:1407.3561
[2] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. 2017. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In *Proc. SOSP*. 51–68.
[3] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang. 2019. Incentive Mechanism for Reliable Federated Learning: A Joint Optimization Approach to Combining Reputation and Contract Theory. *IoT-Journal* 6, 6 (2019), 10700–10714.
[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
[5] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. 2020. Federated Optimization in Heterogeneous Networks. In *Proc. MLSys*, Vol. 2. 429–450.
[6] Y. Lu, Q. Tang, and G. Wang. 2018. On Enabling Machine Learning Tasks atop Public Blockchains: A Crowdsourcing Approach. In *Proc. ICDMW*. 81–88.
[7] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proc. AISTATS*, Vol. 54. 1273–1282.
[8] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo. 2019. DeepChain: Auditable and Privacy-Preserving Deep Learning with Blockchain-based Incentive. *TDSC* (2019), 1–18.