Mechanism Design for An Incentive-aware Blockchain-enabled Federated Learning Platform

Kentaroh Toyoda and Allan N. Zhang
*Singapore Institute of Manufacturing Technology, A*STAR, Singapore*
kentaroh.toyoda@ieee.org

Abstract**—Recent technological evolution enables Artificial Intelligence (AI) model training by users' mobile devices, which accelerates decentralized big data analysis. In particular, Federated Learning (FL) is a key enabler to realize decentralized AI model update without user's privacy disclosure. However, since the behaviour of workers, who are assigned a training task, cannot be monitored, the state-of-the-art methods require a special hardware and/or cryptography to force the workers behave honestly, which hinders the realization. Furthermore, although blockchain-enabled FL has been proposed to give workers reward, any rigorous reward policy design has not been discussed. In this paper, to tackle these issues, we present a novel method using mechanism design, which is an economic approach to realize desired objectives under the situation that participants act rationally. The key idea is to introduce repeated competition for FL so that any rational worker follows the protocol and maximize their profits. With mechanism design, we propose a generic full-fledged protocol design for FL on a public blockchain. We also theoretically clarify incentive compatibility based on contest theory which is an auction-based game theory in economics.**

1. Introduction

AI has been adopted not only in computer science but also in wider areas such as engineering, manufacturing, and biomedical. In these areas, there is a huge demand for collaborative model learning where many users or entities collaboratively train a common model by feeding their own local data. Recently, not only computers but also recent mobile devices such as smartphones and tablet devices equip neural engines to enable AI-assisted applications for better user experience. By capturing these technological advance, in 2016, several researchers in Google proposed the concept of FL where users with mobile devices can contribute for the model training process without disclosing data [1]. Its key idea is that a central server chooses K devices each time and distributes a deep learning model to them, and each device updates it with its own local data and reports only a model update to the central server. The central server averages their results and repeats these procedures many times for improving the global model. This way somewhat preserves user data privacy since the data itself are not sent to the central server.

However, FL has several issues for real implementation. Among them, one of the most severe issues is that workers are assumed to be volunteers; there is no incentive mechanism in return to their contribution. Hence, rational workers have no motivation to work correctly and thus they may deviate the designed protocol. Very recently, several platforms have been proposed to solve this issues by using blockchain, e.g. [2], [3]. Weng *et al.* proposed a blockchain-enabled privacy-preserving deep learning platform termed DeepChain [2]. In DeepChain, their own original cryptocurrency called as DeepCoin is given to workers as incentive. Syayan *et al.* also proposed a public blockchain based privacy-preserving machine learning platform Biscotti [3]. Users with their own data update weights and bias of a neural network model by stochastic gradient descent and store them to the blockchain.

However, the state-of-the-art blockchain-enabled FL platforms still have several issues. The first issue is that to make the rational workers behave *correctly*, they require a special hardware such as Intel SGX's Trusted Execution Environment (TEE) and/or heavy computation such as homomorphic encryption, which hinders realization. By considering the fact that the processes of deep learning are mainly executed on mobile devices, it is better not to use any heavy cryptographic computation and special hardware. The second critical issue is that there is no rigorous proposal for reward policy, i.e. how much should be rewarded to workers so that any rational worker acts honestly.

In this paper, we aim to solve the above issues by introducing *mechanism design*, which is an approach to designing economic mechanisms or incentives, toward desired objectives, in strategic settings, where participants act rationally. More specifically, the key idea is to introduce a competitive model update method so that any rational worker follows the protocol and maximizes their profits. Each worker chosen in a certain round selects top model updates submitted by the workers in the previous round and updates its own model with them. Here, workers' reward is decided by vote by the next round workers. The motivation of choosing the models that achieve the best k models is that their model updates will have a more chance to be voted in the next round, meaning that the more reward will be obtained. The workers in the next round still cannot sabotage because their models are competed and voted by workers in the subsequent round. In this way, unlike the existing

978-1-7281-0858-2/19/$31.00 ©2019 IEEE
978-1-7281-0858-2/19/$31.00 ©2019 IEEE

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:14:35 UTC from IEEE Xplore. Restrictions apply.

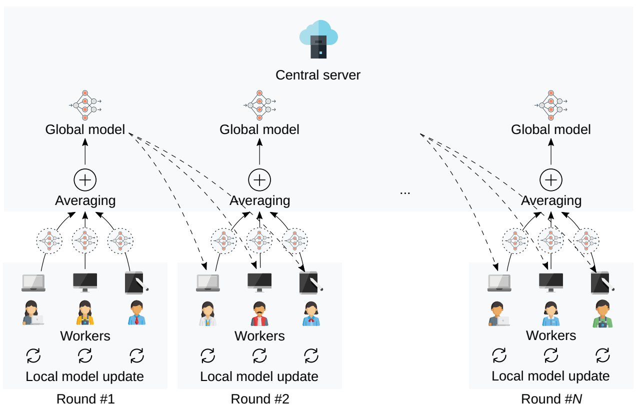
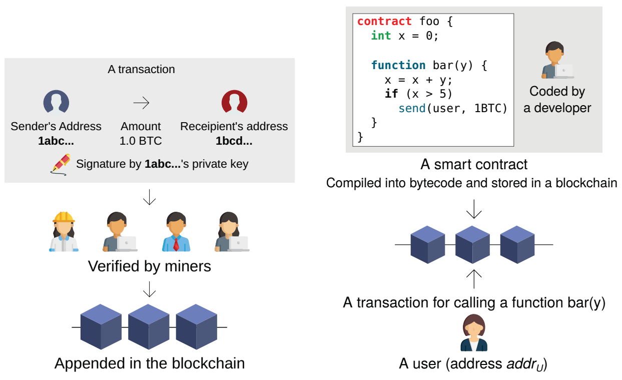


Figure 1: Overview of FL.


A transaction
Sender's Address
1abc...
Amount
1.0 BTC
Receipient's address
1bcd...
Signature by 1abc...'s private key
Verified by miners
Appended in the blockchain
Coded by
a developer
A smart contract
Compiled into bytecode and stored in a blockchain
A transaction for calling a function bar(y)
A user (address addr_U)

```
contract foo {
  int x = 0;

  function bar(y) {
    x = x + y;
    if (x > 5)
      send(user, 1BTC)
  }
}
```

(a) Generic processes for transferring cryptocurrency.

(b) Smart contract and code execution via a transaction.

Figure 2: Example of transaction and smart contract in a cryptocurrency.

works, our design will naturally make rational workers behave honestly without any heavy cryptography and special hardware. To theoretically prove that our protocol design is incentive compatible, we leverage several novel auction theories such as contest theory with multiple prizes, e.g. [4]–[6]. We clarify the optimal conditions for reward policy in a blockchain-enabled FL platform.

The contribution of this paper is as follows.

1) We design a competitive incentive mechanism design for a blockchain-enabled FL platform under the assumption that any worker may deviate the protocol if they have financial merits to do so.
2) We propose a full-fledged protocol which can be implemented on top of the existing public blockchains, e.g. Ethereum.
3) We provide a rigorous theoretical analysis to clarify incentive compatibility based on contest theory.

The rest of this paper is organized as follows. The preliminaries including brief introduction of FL, cryptocurrency, and blockchain are summarized in Section 2. Section 3 deals with the related work. The proposed method is described in Section 4. Section 5 deals with the theoretical analysis. The conclusions and future work are presented in Section 6.

2. Preliminaries

In this section, the fundamentals of FL and blockchain-related technology are described for a better understanding of the blockchain-enabled FL platform.

2.1. Federated Learning

FL was first proposed in 2016 as a decentralized deep learning platform that allows users to collectively reap the benefits of shared models trained from this rich data, without the need to centrally store it [1]. Figure 1 illustrates the basic procedures of FL. A learning task is solved by a loose federation of participating devices (referred to as workers or clients) which are coordinated by a central server. Each worker computes a model update with its own local data to the current global model maintained by the server and uploads only the updated model. By this way, since no local data is uploaded to the central server, big data stored on decentralized users can be used for large model update while somewhat preserving the privacy of workers.

By considering the fact that neural chips have been embedded in for mobile devices, FL will be getting more popular in the near future. However, there exist many issues such as (i) the existence of malfunctioning nodes that intentionally send adversarial updates to make the aggregated model less accurate and (ii) the lack of motivation for contributing the model update.

Many researchers have tried to tackle the above issues. In particular, cryptocurrency and blockchain have been extensively studied as a solution to them, e.g. [2], [3], [7], [8]. One of the basic motivations of utilizing cryptocurrency and blockchain for FL is to give monetary incentive to participating nodes [9], [10]. Furthermore, smart contract, which is a powerful programmable code execution in blockchain, facilitates the security of FL on the blockchain. To get a better understanding of why cryptocurrency, blockchain, and smart contract are fused to FL, the fundamental of them is explained.

2.2. Blockchain and Its Related Technologies

Cryptocurrency and blockchain first emerged as a digital decentralized currency `Bitcoin` and its ledger [11]. To understand how cryptocurrency is exchanged on top of the blockchain, we briefly explain how Bitcoin works since it is the very basic one. Unlike the conventional bank, no identifier or credential is required for opening an account for receive and send Bitcoin. Instead, in Bitcoin, *address*, which can be computed from a pair of public and private keys, is used. Hence, addresses can be easily created by any computers and even smartphones. Bitcoin (BTC) is

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:14:35 UTC from IEEE Xplore. Restrictions apply.

```
struct Participant {
  bool isRequester;
  bool isWorker;
}
mapping (address => Participant) public
    participants;
function enrollParticipant(
  address _addr,
  bool _isRequester,
  bool, _isWorker) public onlyAdmin {
    participants[_addr].isRequester =
        _isRequester;
    participants[_addr].isWorker = _isWorker
        ;
}
```

Figure 3: The pseudo-code of the user registration function written in `Solidity`.

exchanged in between addresses through a message called *transaction*. Figure 2(a) shows an example of a simplified Bitcoin transaction. In this figure, user who controlled his/her Bitcoin address `1abc...` sends 1 BTC to another user who owns `1bcd...` in the transaction. Here it may be wondering how to prove that the owners of `1abc...` and `1bcd...` actually spend their Bitcoin. The key solution is digital signature: In general, a transaction must be signed with a sender's private key so that its signature can be verified by its pair-wise public key. Any such transaction is broadcast by the participants to the Bitcoin P2P network. Shortly, the transaction is verified and stored in an append-only public ledger, namely *blockchain* by *miners*. Miners compete each other to get incentive including newly minted Bitcoin and transaction fee for each block.

The invention of Bitcoin further yields emerging notion so-called smart contract. Smart contract is a computer program operated on top of the blockchain, which is supported on several blockchains such as Ethereum [12]. Figure 2(b) illustrates a toy example of a smart contract and code execution via a transaction. In this example, a developer designed a smart contract code *foo* that stores integer x and users can update its value as $x = x + y$ and get 1 BTC if x is greater than 5 by calling function *bar*(y) via transaction. Any programmable procedures, i.e. calculation, data storing, and cryptocurrency transfer, can be realized with smart contract. In `Ethereum`, javascript-like programming language `Solidity` is often used for coding smart contracts.

There are many advantages to use blockchain and its related technologies for FL platform. The first advantage is that smart contract and cryptocurrency enable us to design the incentive-aware FL for motivating participating users [9], [10]. Cryptocurrency is used as monetary reward and smart contract is used to send cryptocurrency to workers in return to their contribution. The second one is to facilitate the implementation of FL platform by smart contract. It can be imagined that functions required in FL such as task request by requesters and model update by workers are realized with smart contract and blockchain. For example, Figure 3 shows how the user registration is realized in our platform. With this pseudo-code, it can be realized to store the information of participants in the system. Specifically, the platform administrator `Admin` can enroll a participant as task requester and/or worker (`isRequester` and `isWorker`) through the function `enrollParticipant()`.

In the next section, we review the state-of-the-art blockchain-enabled FL platforms.

3. Related Work

So far, many distributed machine learning platforms have been proposed as listed in TABLE 1. Shae and Tsai proposed a concept that leverages the smart contract of blockchain for large medical data processing [7].

Lu *et al.* proposed an approach to crowdsource machine learning tasks to users atop public blockchain [14]. In their proposal, a commitment scheme is introduced to avoid the situation that a malicious worker simply copies and reports other workers' outputs. In addition, an incentivized strategic game is adapted for workers to behave correctly.

Preuveneers *et al.* proposed a permissioned blockchain based FL platform [13]. The motivation to adapt permissioned blockchain is to enable auditing of trained machine learning models. Their platform is implemented on top of `Multichain` [1] and can be used for network anomaly detection with a network trace dataset `CICIDS2017`.

Idé proposed a permissioned blockchain-enabled collaborative platform for anomaly detection [15]. The permissioned blockchain is used for collaborative model learning and each participant node uses it for the calculation of anomaly scores.

Chen *et al.* proposed a privacy-preserving decentralized machine learning platform with differential privacy [16]. In their platform coined `LearningChain`, the workers update their own data with perturbation for preserving privacy while the mining nodes aggregate them and calculate global updates.

Kim *et al.* analyzed an end-to-end latency of blockchain-enabled FL [8]. When mobile devices are assumed to participate the model training processes, the latency due to the communications affects the performance of FL [18]. In [8], several metrics such as latency versus the block generation rate are evaluated on their original blockchain.

Weng *et al.* proposed a blockchain-enabled privacy-preserving deep learning platform termed `DeepChain` [2]. To avoid the users' information leakage, the additive homomorphic encryption of threshold Pailler encryption is introduced. Their original coin `DeepCoin` is given to users as incentive but misbehaviours such as stale reports and incorrect execution are punished.

Syayan *et al.* proposed a public blockchain based privacy-preserving machine learning platform `Biscotti` [3]. Users with their own data update weights and bias of a neural network model by stochastic gradient descent and store them to the blockchain. To preserve privacy of

1. https://www.multichain.com/

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:14:35 UTC from IEEE Xplore. Restrictions apply.

TABLE 1: Summary of existing work related with blockchain for AI. '-' denotes the detail is not specified or not implemented.

Reference	Blockchain Type	Used Blockchain	Overview
[7]	-	-	Concept of utilizing smart contract for decentralized medical data processing
[13]	Private	`Multichain`	Permissioned blockchain-enabled FL
[14]	Public	-	Blockchain-based crowdsourcing platform for machine learning tasks
[15]	Permissioned	`Hyperledger Fabric`	blockchain as a collaborative platform
[16]	Public	Ethereum	Blockchain-based decentralized machine learning platform with differential privacy
[8]	-	Original	Latency analysis of blockchain-enabled FL
[2]	Private	`Corda` V3.0	Blockchain-based deep learning platform that distribute incentive to workers
[3]	Public	Original	Blockchain-based machine learning platform
[17]	Private	Original	Blockchain-based machine learning platform

contributing users, a noise addition technique, commitment scheme, and Shamir's secret sharing scheme are introduced. Furthermore, to eliminate negatively contributing updates, the notion of reject on negative influence is introduced.

Zhu *et al.* proposed a blockchain-enabled decentralized machine learning platform for eliminating malfunctioning nodes that negatively influence the model update [17]. The global machine learning model is repeatedly updated. In each round, one of the participates broadcasts its local model update and other participates sends back feedback based on how much their local models improve.

3.1. Unsolved Issues

Although many methods have been proposed so far, the following two issues hinder the realization of incentive-aware blockchain-enabled machine learning platforms; (i) infeasibility of platform and (ii) lack of rigorous reward policy.

We first discuss the infeasibility. Since requesters cannot monitor how workers behave, the state-of-the-art methods try to force the worker honestly by introducing cryptographic approaches, e.g. [2], [3]. Specifically, hardware-based attestation such as Intel SGX's TEE and/or homomorphic encryption hinder the implementation on top of general blockchains, e.g. Ethereum. Hence, it is better not to use heavy cryptographic computation such as homomorphic encryption and special hardware. Furthermore, the implementation should be on top of well-known public blockchain such as Ethereum otherwise few reward incentive can be distributed to workers. More specifically, some FL functions of many methods such as [3], [8] are associated with the block generation process, which means their own blockchains and cryptocurrencies must be invented. However, from the participants perspective, it may be doubtful if such minor cryptocurrency actually has value.

The second issue is the lack of rigorous reward policy. When we design the protocol, the following two questions might arise: How many workers should be incentivized?; and how much reward should be distributed to workers based on their contribution? Lu *et al.* proposed a prisoner's dilemma-like game theoretical strategy for a blockchain-enabled crowdsourcing platform [14]. In their platform, if and only if all workers report exactly the same result the reward is distributed equally to the workers otherwise workers lose half of their deposits. Obviously, this approach cannot be applied in FL because each worker has different dataset and thus the results are different by workers. Kim *et al.* proposed that reward is proportionally distributed to workers based on the data size used for training [8]. However, this approach yields another attacking vector that falsely earns reward by feeding a large number of bogus data to train a model. So far, no rigorous reward policy has not been discussed yet.

4. Proposed Blockchain-based FL Platform

To deal with the above issues, we propose a novel incentive aware blockchain-enabled FL platform with mechanism design. The key idea behind our platform is introducing a repeated model update competition design to make any rational worker work hard and follow the protocol[2]. Specifically, each worker chosen in a certain round selects top k model updates submitted by workers in the previous round and updates its own model based on them. The reward to workers in the previous round is decided by the result of vote. The workers in the next round still cannot sabotage because their models are competed and voted by workers in the next round too.

In the following, the system model, the full-fledged procedures, reward policy, and several discussion are dealt with. The notation used in the proposed platform is listed in TABLE 2.

4.1. System Model

Figure 4 illustrates the system model. We assume the existence of (i) the administrator `Admin`, (ii) the set of requesters who commit deep learning tasks $\mathcal{R}$, and (iii) the set of workers $\mathcal{W}$. TABLE 3 lists the role of each participant. The role of `Admin` is to deploy the series of smart contracts on the public blockchain, e.g. `Ethereum`, and to enroll requesters and workers to the platform upon their request. Participants are assumed to know how to access platform, i.e. the location of the smart contracts, through an online

2. This idea may be somewhat similar with the `Bitcoin`'s mining process.

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:14:35 UTC from IEEE Xplore. Restrictions apply.

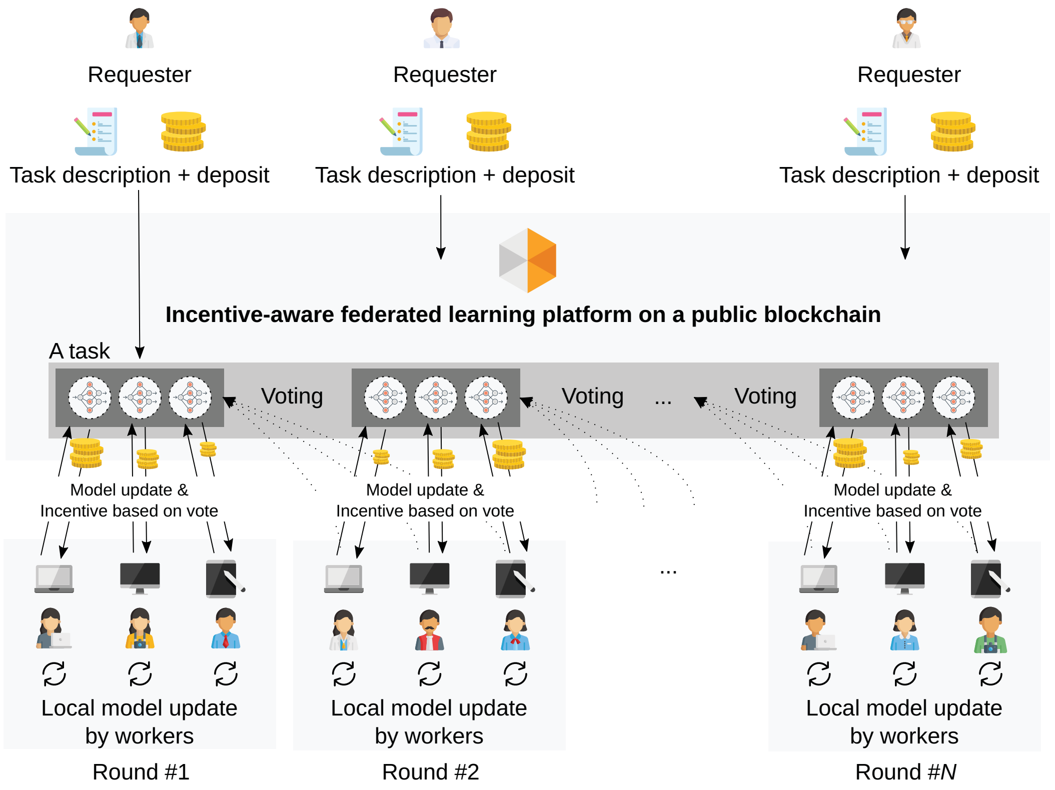


Figure 4: Overview of our FL platform with blockchain.

forum or a website e.g. *Etherscan*[3]. Requesters may neither have data for training nor devices to train deep learning models, while workers have both of data for training and devices for training deep learning models. Any kind of data can be dealt with in this platform, e.g. images, texts, and audio. Let d_i denote worker i's dataset for task t. In the following, since we focus on the specific task t, the subscript t is omitted. We assume that datasets that workers possess for a specific task are independent and identically distributed. This assumption is natural because a requester advertises a model for a specific task, e.g. a deep learning model for identifying cats in pictures, and only the workers who have data specific to this task will join.

4.2. Procedures

Our platform consists of several procedures; (i) user registration, (ii) task post, (iii) task join, (iv) model update, (v) reward distribution, and (vi) task finish. Although we assume that `Ethereum` is used as blockchain because it is one of the most prevail blockchains that supports smart contract, our system must be implemented on any other public blockchain that supports cryptocurrency and smart contract.

4.2.1. User Registration. All participating users, i.e. requesters and workers, must register themselves to the platform. `Admin` need enroll users upon request. For `Admin` to enroll a participant, every participating user must give its (i) `Ethereum` address (`addr`) that will be used for receiving task and its reward and (ii) whether it is enrolled as requester and/or worker. After the registration is finished, requesters can post deep learning tasks and workers can join the model update tasks for reward.

3. https://etherscan.io/

4.2.2. Task Post. Any participant who enrolled as requester can post a machine learning task through the contract execution. A requester must specify (i) the model description, e.g. the data format, the number of layers, the number of units, loss function, learning rate, and activation functions and (ii) parameters, e.g. the application period for this task, the starting time T, the number of workers in each round K', the total reward in each round r, and (vi) deposit of total reward $D = rN$.

4.2.3. Task Join. After a task is posted, event notification is sent to all registered workers through the `Ethereum`'s event handling function. Then, each worker decides whether or not to join this task. If a worker has decided to join it, the task join function in the smart contract should be called before its start time. Upon the request, the smart contract registers the caller only if it is registered as a worker and aborts the code execution otherwise. In the implementation perspective, workers' `Ethereum` addresses are stored on an array.

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:14:35 UTC from IEEE Xplore. Restrictions apply.

TABLE 2: Notation Table.

ITEM	DEFINITION
$\mathcal{R}$	The set of entire requester
$\mathcal{W}$	The set of entire worker
t	The index of a task
i	The index of a worker
e	The index of a round
k	The number of chosen top model updates by workers in the next round
N	The number of total rounds of task t
K	The number of total workers joined to task t
C	The fraction of workers in each round
K'	The number of chosen workers in each round, i.e. $K' = \max(\lceil K \cdot C \rceil, 1)$
T	The start time of task t
T_{com}	The time for accepting a commitment
T_{rev}	The time for accepting a reveal
$\mathcal{W}_t$	The set of workers joined to task t
r	The total reward distributed to workers in a round
$r_{i,e}$	The reward distributed to worker i in round e
D	The deposit for task t, $D = rN$
d_i	The set of data that worker i of round e possesses
B	Batch size in deep learning
η	Learning rate in deep learning
$\mathcal{H}(\cdot)$	A hash function, e.g. SHA-256
$\mathsf{enc}(\cdot, \mathcal{K})$	Encryption functions with a symmetric key $\mathcal{K}$
$\mathsf{dec}(\cdot, \mathcal{K})$	Decryption functions with a symmetric key $\mathcal{K}$
$\mathcal{L}(\cdot)$	A loss function, e.g. mean square error, in deep learning
$\mathsf{split}(\mathsf{d}, B)$	A function to split the dataset d randomly into B batches

4.2.4. Task Start. After application period has passed, the requester chooses the number of rounds N for model update and the number of workers in a round K' based on the set of workers joined to task t, i.e. $\mathcal{W}_t$[4]. As will be explained later, the requester should not reveal N to workers beforehand but it need disclose it when round N is finished. Hence, we leverage a commitment scheme for N. Specifically, the requester sends $\mathcal{H}(N, s)$ to the smart contract, where s is a one-time pseudo-random value which must be disclosed together with N at the reveal phase. In addition, the requester initializes the deep learning model as w_0, encrypts with a symmetric key $\mathcal{K}_0$, and places the encrypted w_0 in the blockchain[5]. The requester can use any algorithm for initializing the model, e.g. [19].

4.2.5. Model Update. After the model training for a task started, every round, K' workers are randomly chosen from the workers who registered task t by the smart contract[6]. Before the workers start their own model update, some security-related procedures are required. To allow the chosen workers to securely retrieve the previous models and submit their model updates, in round e, the requester generates a symmetric $\mathcal{K}_e$ that is used for securely storing workers' model updates and sends it together with the previous round key $\mathcal{K}_{e-1}$ to the workers in round e. The two keys are encrypted with each worker's public key one by one and are sent to the workers. This procedure is required because the contents of public blockchain is visible to anyone. Hence, workers in round e use $\mathcal{K}_e$ to encrypt their own model updates to avoid the situation that non-registered participants know the updated models with free.

4. Note that if the number of joining workers is 0 or few, such task is automatically terminated and the deposit is paid back to the requester.

5. We assume that any data is stored on the blockchain but database can be also used instead.

6. To choose workers randomly, we can use a public randomness source outside of the blockchain, e.g. `Oraclize`[7].

Algorithm 1: Model Update by Worker i in Round e.

```
Input  : Model updates committed by workers of
         round $e-1$, $\{w_{i,e-1}\}_{i \in K'}$ (when $e \geq 2$)
         or $w_0$ (when $e = 1$)
         Dataset $\mathsf{d}_i = \{\mathbf{x}_j, y_j\}$
Output: $w_i$
/* 1. Choose the top $k$ previous
   model updates with its own data
   (Except for the first round):  */
if $e \geq 2$ then
    for $m \in K'$ do
        $l_m = \frac{1}{|\mathsf{d}_i|} \sum_{j \in |\mathsf{d}_i|} \mathcal{L}(\mathbf{x}_j, y_j; w_{m,e-1})$
    end
    $\mathcal{M}_{i,e} \leftarrow$ Choose $k$ models' indices of which $l_m$
     is in $k$ lowest values.
end
/* 2. Average the top $k$ previous
   model updates                  */
if $e == 1$ then
    $w'_{i,e} \leftarrow w_0$
else
    $w'_{i,e} \leftarrow \frac{1}{k} \sum_{m \in \mathcal{M}_{i,e}} w_{m,e-1}$
end
/* 3. Model update with own data  */
$\mathcal{B} \leftarrow \mathsf{split}(\mathsf{d}_i, B)$
$w_{i,e} \leftarrow w'_{i,e}$
for each local epochs $E$ do
    for each batch $b$ in $\mathcal{B}$ do
        $w_{i,e} \leftarrow w_{i,e} - \eta \nabla \mathcal{L}(w_{i,e}, b)$
    end
end
return $w_{i,e}$
```

After encrypted $\mathcal{K}_{e-1}$ and $\mathcal{K}_e$ are received, each worker in round e decrypts them with their own private keys, retrieves the models in the previous round from the blockchain [8], and update models as shown in Alg. 1. Updated models are submitted to the blockchain through the implemented smart contract. However, to avoid other workers from seeing the contribution of workers in the same round, commitment scheme is adapted [14]. Specifically, we divide the model update procedure into two phases; (i) commit phase and (ii) reveal phase. In commit phase, worker i in round e calculates the commitment of the model update $w_{i,e}$ as $\texttt{commit}(w_{i,e}) = \mathcal{H}(w_{i,e}, s_{i,e})$. Then, the commitment is

8. For the workers in the first round, only the initial model w_0 is retrieved.

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:14:35 UTC from IEEE Xplore. Restrictions apply.

TABLE 3: Role of each participant.

Participant	Role	Has data?	Has devices for model training?
Administrator	Deploy the smart contracts on the public blockchain and enroll requesters and workers to the smart contract upon their request	-	-
Requester	Commit ML tasks to get trained models	Not necessarily	Not necessarily
Worker	Train the models of tasks committed by requesters for reward	Yes	Yes

Algorithm 2: Requester's procedures of reveal check.

```
Input  : The set of commitments and reveals:
         ${commit(w_{i,e}), s_{i,e}, (w_{i,e})}$
Output: The list of workers' indices that passed
         commitment: $\mathcal{I}_e = \{i\}$
/* Check the reveals one by one. */
$\mathcal{I}_e \leftarrow \{\}$
for $i \in K'$ do
    if $\mathcal{H}(w_{i,e}, s_{i,e}) == \mathtt{commit}(w_{i,e})$ then
        $\mathcal{I}_e$.append($i$)
    end
end
return $\mathcal{I}_e$
```

encrypted with $\mathcal{K}_e$ as $\mathtt{enc}(\mathtt{commit}(w_{i,e}), \mathcal{K}_e)$ with a fresh salt $s_{i,e}$ and is submitted. Then, after T_{com} passed, reveal phase starts and each worker should reveal the used salt to the smart contract by deadline T_{rev}. After T_{rev} elapsed, the requester fetches the commitments and reveals and checks their consistencies as shown in Alg. 2.

4.2.6. Reward Distribution. As shown in Alg. 1, in the commit phase of model update, each worker in round e votes top k previous models (exactly one vote for one model). Based on the aggregated votes, the smart contract calculates the vote counts of workers in round $e-1$. Based on the result of vote counts, reward is distributed to workers as $r_1 \geq r_2 \geq \cdots \geq r_{K'} \geq 0$. Hence, the most voted worker receives r_1 and the first runner-up receives r_2, and so on. Since the total reward in each round is fixed to r,

$$\sum_{j \in [1, K']} r_j = r. \tag{1}$$

Due to the space limitations, the optimal structure of r_j will be discussed in the extended paper.

4.2.7. Task Finish. The model update and reward distribution are repeated N times. The model updates by workers in the final round N cannot be voted because there is no workers in the next round. Hence, in the last round, the reward is equally distributed to the workers. However, this yields a problem that reduces the motivation for the workers to work correctly; If the workers knew that they were chosen in the final round, they could receive rewards by just sending one of the previous model updates. If this happened, the motivation for honest calculation of the workers in the previous rounds would still get lowered since their model updates might not be correctly voted by the workers in the last round. Then, the same thing would happen in between the previous workers and their previous workers as well. Hence, to make the model work effectively, the last workers must not know if they are in the final round. To avoid this, the requester need reveal N after N rounds have been finished. This is why the requester should use the commitment scheme for N. In order not to be guessed by the workers from the rest of deposit, the requester should give larger deposit D than actual total reward $N \cdot r$. The requester does not lose the rest of deposit since it is returned after the all tasks are finished.

4.3. Discussion

Our platform has several big advantages against the existing ones such as the feasibility of implementation. Our design only requires a few symmetric key encryption and decryption, which are very lightweight compared with the homomorphic encryption. Furthermore, none of the special hardware is required for implementation. This means that any public blockchain supporting smart contract can be used as the basis of our FL platform.

4.3.1. Effect of Time Drift. Our platform has several deadlines for procedures; (i) task join (workers), (ii) commit and reveal phases of model update (workers), (iii) report of consistency check of commit and reveal (a requester). Since the clocks of miners are used as the standard times, in order not to miss each deadline, each participant should synchronize their clocks with the standard time offered by NTP (Network Time Protocol) servers.

4.3.2. Complexity of Key Distribution. When a requester distributes $\mathcal{K}_{e-1}$ and $\mathcal{K}_e$ to K' workers, the workers public keys are used for the encryption of the two keys. The order of calculation and communications of this procedure is $\mathcal{O}(K')$. This might be a problem when K' is large. If it matters, efficient group key distribution schemes, e.g. [20], might help.

4.3.3. Number of rounds N. To motivate workers to perform honestly we introduced a notion of competition and incentive; However, as the more rounds are finished, the difference of submitted models will be smaller, which degrades the motivation for workers to honestly update models. To avoid this, an appropriate value should be set to N by the requester. We will tackle how to set the optimal N in the future.

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:14:35 UTC from IEEE Xplore. Restrictions apply.

4.3.4. Collusion and Sybil Attack. There is a possibility that an adversary may try to enroll a number of workers in order to get more chance to collude among workers and to poison the model updates. Our platform is somewhat robust against them by design because (i) the process of enrollment requires the approval by a system administrator and (ii) workers are chosen with publicly verifiable randomness, meaning that neither requesters nor workers can control the rounds that workers take charge of. Another mitigation for reducing the number of Sybil participants is to impose enrollment fee for every participant. Since our platform is on top of the blockchain, such modification is easily implemented.

5. Theoretical Analysis

We analyze incentive compatibility based on game theoretical approaches [4]–[6].

For this, we first mathematically model the entire system. The value of the j-th reward is denoted as r_j, where $r_1 \geq r_2 \geq \cdots \geq r_{K'} \geq 0$. Hence, the most voted worker receives r_1 and the first runner-up receives r_2, and so on. Since the total reward in each round is fixed to r, $\sum_{j \in [1,K']} r_j = r$ holds. We assume that the data size and the processing cost for one data are numerically represented as x and c, respectively. Our model can be seen as imperfect contest, meaning that every worker does not know competitors' c. This assumption seems natural in our FL platform. Furthermore, the quality of each data x for model update is identical. We also assume that since the cost of model update, $\mathcal{C}(\cdot)$, is linearly increasing with data size, the worker's cost for model update can be represented as the multiplication of data amount x and its unit cost c, i.e.,

$$\mathcal{C}(c,x) = cx. \tag{2}$$

c differs by workers and the abler worker has the lower c. This happens because the workers' environments are different, e.g. abler workers may possess more energy-efficient processors for model update. More specifically, the cost c is uniformly distributed in $[0,1]$, i.e.,

$$F(c) = c. \tag{3}$$

A worker's payoff $\pi(c,x)$ can be denoted as follows.

$$\pi(c,x) = \begin{cases} u(r_j) - \mathcal{C}(c,x), & \text{if a worker is ranked } j\text{−th} \\ -\mathcal{C}(c,x), & \text{otherwise} \end{cases} \tag{4}$$

where $u(r_j)$ is a von Neumann-Morgenstern utility function, which can represent the worker's risk for the reward. Since rewards are not necessarily guaranteed against contribution, we assume that workers may be risk-averse and thus $u(\cdot)$ is monotone increasing and concave. Let $G(x)$ for $x \geq 0$ denote the cumulative density function that a worker contributes less than or equal to x. $G(x)$ is continuous and strictly increasing in the interval $(0, x_{\max})$, where $x_{\max}$ is the biggest contribution by the most skillful worker. By using Eq. 4, the worker i takes the best strategy to maximize its expected utility as

$$E[\pi(c,x)] = U(x) - cx, \tag{5}$$

where $U(x)$ denotes the expected utility of a worker with its effort x. When worker i is ranked j-th place, $j-1$ workers exerted greater efforts than i and $K'-j$ workers exerted less effort than i. Hence, $U(x)$ is represented as

$$U(x) = \sum_{j=1}^{K'} \binom{K'-1}{j-1} G(x)^{K'-j} \left(1 - G(x)\right)^{j-1} u(r_j). \tag{6}$$

In the following, we analyze the equilibrium only for a round in a specific task as its result will be true for subsequent rounds.

We first show the condition to make every worker exert effort.

Lemma 1. *In order for a requester to make all workers exert their efforts, the following condition must be satisfied;*

$$u(r_{K'}) \geq E[\pi(\bar{c}, x(\bar{c}))]. \tag{7}$$

Otherwise, only workers with $c < \bar{c}$ will participate and others will choose no effort $x(c) = 0$.

Proof. Since our design can be seen as *all-pay contest*, every worker must exert a certain amount of effort to be rewarded before submission. This means that workers are assured to gain some profit if the smallest reward, i.e. $r_{K'}$, is larger than its cost for effort.

The less able workers may choose one of the following strategies: (i) Not participating or (ii) randomly picking a previous worker's model and submitting it. In the latter case, such workers cannot take cost to pick the best previous model because this procedure also takes some cost. Hence, the probability of getting any reward by submitting a randomly picked previous model may be low. ■

Lemma 2. *The abler worker is, the more contribution it offers, which means the ablest worker exerts the most effort.*

Proof. We prove this by showing that $x(c)$ is non-increasing for c where $(0, \bar{c})$. For this, we show the first derivative $dx(c)/dc$ is decreasing. A worker with $c < \bar{c}$ chooses the optimal effort x so that its expected payoff's first order condition with respect to x and the second order condition are satisfied. Specifically,

$$\frac{dE[\pi(c,x)]}{dx} = 0 \Leftrightarrow \frac{dU(c,x)}{dx} = \frac{\partial}{\partial x}\mathcal{C}(c,x) = c, \tag{8}$$

and

$$\begin{gathered} \frac{d^2E[\pi(c,x)]}{dx^2} < 0 \\ \frac{d^2U(c,x)}{dx^2} - \frac{\partial^2}{\partial x^2}\mathcal{C}(c,x) < 0 \Leftrightarrow \frac{d^2U(c,x)}{dx^2} < 0, \end{gathered} \tag{9}$$

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:14:35 UTC from IEEE Xplore. Restrictions apply.

are satisfied. By differentiating Eq. 8 with c, we get

$$\begin{aligned}\frac{d^2U(c,x)}{dxdc}\cdot\frac{dx(c)}{dc} &= \frac{\partial^2\mathcal{C}(c,x)}{\partial x\partial c}\\ \frac{dx(c)}{dc} &= \frac{\partial^2\mathcal{C}(c,x)}{\partial x\partial c}\Bigg/\frac{d^2U(c,x)}{dxdc}.\end{aligned} \tag{10}$$

Since $\mathcal{C}(c,x) = cx$, $\partial^2\mathcal{C}(c,x)/\partial x\partial c = 1$. Hence, from Eq. 9, $dx(c)/dc < 0$. ■

6. Conclusions and Future Work

We have introduced mechanism design and contest theory to design a fair incentive-aware blockchain-enabled FL platform. The contribution of this work is that the proposed design naturally overcomes the existing challenge that requesters cannot monitor the workers' behaviors in the blockchain-enabled FL by introducing competition mechanism to FL. Based on this, we have proposed a full-fledged protocol for the incentive-aware blockchain-enabled FL platform. We have proven that the more skillful workers are to contribute more effort for model training tasks under our assumption. The theoretical proof is positive for realizing a fair incentive-aware blockchain-enabled FL platform.

In the future, we will tackle the remaining issues. Specifically, we need to answer the following key questions.

1) How should a requester set the optimal round N?;
2) For workers to maximize their payoff, how much effort should be exerted?;
3) For a requester to obtain the best trained model to its suggesting reward, how should the reward be split?;
4) How much gas cost is required for executing the implemented Ethereum code?

Acknowledgments

This work is supported by the A*STAR Industrial Internet of Things Research Program, under the RIE2020 IAF-PP Grant A1788a0023, and Model Factory@SIMTech.

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," *arXiv preprint arXiv:1602.05629*, 2016.

[2] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and Privacy-preserving Deep Learning with Blockchain-based Incentive," *Cryptology ePrint Archive, Report 2018/679*, 2018.

[3] M. Shayan, C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Biscotti: A Ledger for Private and Secure Peer-to-Peer Machine Learning," *arXiv preprint arXiv:1811.09904*, 2018.

[4] A. Glazer and R. Hassin, "Optimal Contests," *Economic Inquiry*, vol. 26, no. 1, pp. 133–143, 1988. DOI: 10.1111/j.1465-7295.1988.tb01674.x.

[5] B. Moldovanu and A. Sela, "The Optimal Allocation of Prizes in Contests," *American Economic Review*, vol. 91, no. 3, pp. 542–558, 2001.

[6] N. Archak and A. Sundararajan, "Optimal Design of Crowdsourcing Contests," *Proc. of International Conference on Information Systems (ICIS)*, pp. 1–16, 2009.

[7] Z. Shae and J. Tsai, "Transform Blockchain into Distributed Parallel Computing Architecture for Precision Medicine," in *Proc. of IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2018, pp. 1290–1299.

[8] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "On-device Federated Learning via Blockchain and Its Latency Analysis," *arXiv preprint arXiv:1808.03949*, 2018.

[9] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated Machine Learning: Concept and Applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, p. 12, 2019.

[10] K. Salah, M. H. Rehman, N. Nizamuddin, and A. Al-Fuqaha, "Blockchain for AI: Review and Open Research Challenges," *IEEE Access*, vol. 7, pp. 10 127–10 149, 2019.

[11] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," pp. 1–9, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[12] G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," *Ethereum Project Yellow Paper*, pp. 1–32, 2014.

[13] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor, "Chained Anomaly Detection Models for Federated Learning: An Intrusion Detection Case Study," *Applied Sciences*, vol. 8, no. 12, p. 2663, 2018.

[14] Y. Lu, Q. Tang, and G. Wang, "On Enabling Machine Learning Tasks atop Public Blockchains: A Crowdsourcing Approach," in *Proc. of IEEE International Conference on Data Mining Workshops (ICDMW)*, 2018, pp. 81–88.

[15] T. Ide, "Collaborative Anomaly Detection on Blockchain from Noisy Sensor Data," in *Proc. of IEEE International Conference on Data Mining Workshops (ICDMW)*, 2018.

[16] X. Chen, J. Ji, C. Luo, W. Liao, and P. Li, "When Machine Learning Meets Blockchain: A Decentralized, Privacy-preserving and Secure Design," in *Proc. of IEEE International Conference on Big Data*, 2018, pp. 1178–1187.

[17] X. Zhu, H. Li, and Y. Yu, "Blockchain-Based Privacy Preserving Deep Learning," in *Proc. of International Conference on Information Security and Cryptology*, Springer, 2018, pp. 370–383.

[18] T. Nishio and R. Yonetani, "Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge," in *Proc. of IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *Proc. of IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[20] A. Penrig, D. Song, and J. D. Tygar, "ELK, a New Protocol for Efficient Large-Group Key Distribution," in *Proc. of IEEE Symposium on Security and Privacy (SP)*, 2001, pp. 247–262.

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:14:35 UTC from IEEE Xplore. Restrictions apply.