DeepChain: Auditable and Privacy-Preserving Deep Learning with Blockchain-based Incentive

Jiasi Weng, Jian Weng, *Member, IEEE,* Jilian Zhang, Ming Li, Yue Zhang, Weiqi Luo

Abstract—Deep learning technology has achieved the high-accuracy of state-of-the-art algorithms in a variety of AI tasks. Its popularity has drawn security researchers' attention to the topic of privacy-preserving deep learning, in which neither training data nor model is expected to be exposed. Recently, federated learning becomes promising for the development of deep learning where multi-parties upload local gradients and a server updates parameters with collected gradients, the privacy issues of which have been discussed widely. In this paper, we explore additional security issues in this case, not merely the privacy. First, we consider that the general assumption of honest-but-curious server is problematic, and the malicious server may break privacy. Second, the malicious server or participants may damage the correctness of training, such as incorrect gradient collecting or parameter updating. Third, we discover that federated learning lacks an effective incentive mechanism for distrustful participants due to privacy and financial considerations. To address the aforementioned issues, we introduce a value-driven incentive mechanism based on Blockchain. Adapted to this incentive setting, we migrate the malicious threats from server and participants, and guarantee the privacy and auditability. Thus, we propose to present DeepChain which gives mistrustful parties incentives to participate in privacy-preserving learning, share gradients and update parameters correctly, and eventually accomplish iterative learning with a win-win result. At last, we give an implementation prototype by integrating deep learning module with a Blockchain development platform (Corda V3.0). We evaluate it in terms of encryption performance and training accuracy, which demonstrates the feasibility of DeepChain.

Index Terms—Deep learning, Privacy-preserving training, Blockchain, Incentive

1 Introduction

Recent advances in deep learning based on artificial neural networks have witnessed unprecedented accuracy in various tasks, e.g., speech recognition [1], image recognition [2], drug discovery [3] and gene analysis for cancer research [4], [5]. In order to achieve even higher accuracy, huge amount of data must be fed to deep learning models, incurring excessively high computational overhead [6], [7]. This problem, however, can be solved by employing distributed deep learning technique that has been investigated extensively in recent years. Unfortunately, privacy issue worsens in the context of distributed deep learning, as compared to conventional standalone deep learning scenario.

Privacy-preserving deep learning thus arises to deal with privacy concerns in deep learning, and various models have been around in the past few years [8], [9], [10], [11], [12], [13], [14], [15], [16]. Among these existing work, *federated learning* is the widely adopted system context. Federated learning, also known as *collaborative learning*, *distributed learning*, is essentially the combination of deep learning and distributed computation, where there is a server, called parameter server, maintaining a deep learning model to train and multiple parties that take part in the distributed training process. First, the training data is partitioned and stored at each of the parties. Then, each party trains a deep learning model (the same one as maintained at the parameter server) on her local data individually, and uploads intermediate gradients to the parameter server. Upon receipt of the gradients from all the parties, the parameter server aggregates those gradients and updates the learning model parameters accordingly, after which each of the parties downloads the updated parameters from the server and continues to train her model on the same local data again with the downloaded parameters. This training process repeats until the training errors are smaller than pre-specified thresholds.

This federated learning framework, however, cannot protect the privacy of the training data, even the training data is divided and stored separately. For example, some researchers show that the intermediate gradients can be used to infer important information about the training data [17], [18]. Shokri *et. al* [11] applied differential privacy technique by adding noises in the gradients to upload, achieving a trade-off between data privacy and training accuracy. Hitaj *et. al* [19] pointed out that Shokri's work failed to protect data privacy and demonstrated that a curious parameter server can learn private data through GAN (Generative Adversarial Network) learning. Orekondy *et. al* [20] exploited the intermediate gradients to launch linkability attack on training data, since the gradients contain sufficient data features.

Phong *et. al* [16] proposed to use homomorphic encryption technique to protect training data privacy from curious parameter server. The drawback of their scheme is that they assumed the collaborative participants are honest but not curious, hence their scheme may fail in scenario where some participants are curious. To prevent curious participants, Bonawitz *et. al* [14] employed a secret sharing and symmetric encryption mechanism to ensure confiden-

• *J. S. Weng, J. Weng, J. L. Zhang, M. Li, Y. Zhang and W. Q. Luo are with the College of Information Science and Technology in Jinan University, and Guangdong/Guangzhou Key Laboratory of Data Security and Privacy Preserving, Guangzhou 510632, China.*
E-mail addresses: wengjiasi@gmail.com (J. S. Weng), cryptjweng@gmail.com (J. Weng), jilian.z.2007@smu.edu.sg(J. L. Zhang), limjnu@gmail.com (M. Li), zyueinfosec@gmail.com (Y. Zhang), lwq@jnu.edu.cn (W. Q. Luo).
Jian Weng is the corresponding author.

tiality of the gradients of participants. They assumed that (1) participants and parameter server cannot collude at all, and (2) the aggregated gradients in plain text reveal nothing about the participants' local data. The second assumption, unfortunately, is no longer valid since membership inference attack on aggregated location data is now available [21].

Despite extensive research is underway on distributed deep learning, there are two serious problems that receive less attention so far. The first one is that existing work generally considered privacy threats from curious parameter server, neglecting the fact that there exist other security threats from dishonest behaviors in gradient collecting and parameter update that may disrupt the collaborative training process. For example, the parameter server may drop gradients of some parties deliberately, or wrongly update model parameters on purpose. Recently, Bagdasaryan *et. al* [22] demonstrated the existence of this problem that dishonest parties can poison the collaborative model by replacing the updating model with its exquisitely designed one. Therefore, it is crucial for distributed deep learning framework to guarantee not only confidentiality of gradients, but auditability of the correctness of gradient collecting and parameter update.

The second problem is that in existing schemes those parties are assumed to have enough local data for training and are willing to cooperate in the first place, which are not always true in real applications. For example, in healthcare applications, companies or research institutes are usually facing the difficulty in collecting enough personal medical data, due to privacy regulations such as HIPAA [23] and people's unwillingness to share. As a consequence, lack of training data will result in poor deep learning models in general [24]. On the other hand, in business applications some companies may be reluctant to participate in collaborative training, because they are very concerned about possible disclosure of their valuable data during distributed training [11]. Obviously, it is vital to ensure data privacy and bring in some incentive mechanism for distributed deep learning, so that more parties can actively involved in collaborating training.

In this paper, we propose DeepChain, a secure and decentralized framework based on Blockchain-based incentive mechanism and cryptographic primitives for privacy-preserving distributed deep learning, which can provide data confidentiality, computation auditability, and incentives for parties to participate in collaborative training. The system models of traditional distributed deep learning and our DeepChain are given in Fig. 1. Specifically, DeepChain can securely aggregate local intermediate gradients from untrusted parties through launching transactions, while local training and parameter update are performed by workers (an entity in DeepChain that will be defined shortly) who are incented to process the transactions. Through transaction processing and incentive mechanism, DeepChain achieves collaborative training. Meanwhile, by using cryptographic techniques we ensure data confidentiality and auditability of the collaborative training process as well. To summarize, in this paper we made the following contributions:

- We propose DeepChain, a collaborative training framework with an incentive mechanism that encourages parties to jointly participate in deep learning model training and share the obtained local gradients.
- DeepChain preserves the privacy of local gradients and guarantees auditability of the training process. By employing incentive mechanism and transactions, participants are pushed to behave honestly, particularly in gradient collecting and parameter update, thus maintaining fairness during collaboration training.
- We implement DeepChain prototype and evaluate its performance in terms of encryption efficiency and training accuracy. We believe that DeepChain can benefit AI and machine learning communities, for example, it can audit collaborative training process and the trained model, which represents the learned knowledge. Making the best use of this learned knowledge by combining transfer learning technique can improve both the learning efficiency and accuracy.

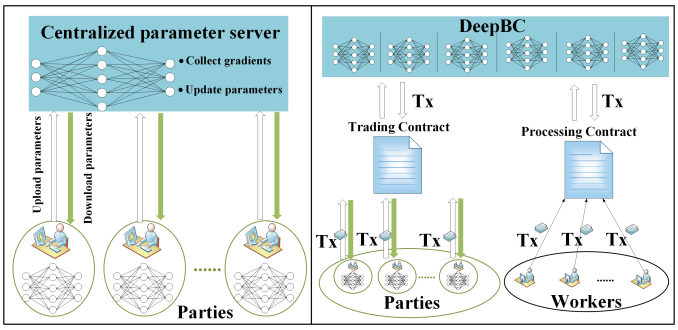


Fig. 1. The left corresponds to traditional distributed deep training framework, while the right is our DeepChain. Here, Trading Contract and Processing Contract are smart contract in DeepChain, together guiding the secure training process, while Tx refers to transaction.

The rest of the paper is organized as follows. In Section 2, we give a brief introduction of Blockchain and deep learning model training. Then, we describe the threat model and security requirements in Section 3. In Section 4, we present our DeepChain, a framework for auditable and privacy-preserving deep learning, and analyze security properties of DeepChain in Section 5. We give implementation details of DeepChain in Section 6, and conduct extensive experiments to evaluate its performance. Finally, we conclude the paper in Section 7.

2 Background

Our work is closely related to Blockchain and deep learning training, and we give background knowledge in this section.

2.1 Blockchain technology

Blockchain was first technology has arisen a surge of interests both in the research community and industry [25]. It becomes an emerging technology as a decentralized, immutable, sharing and time-order ledger. Transactions are stored into blocks containing timestamps and references

(i.e., the hash of a previous block) which are maintained as a chain. In Blockchain, transactions are created by pseudonymous participants and competitively collected to build a new block by an entity called *worker*. The worker who builds a new and valid block can gain amount of rewards so that the chain is continuously lengthened by competitive workers. That presents the incentive mechanism in the Blockchain setting. In addition, pro-developing Blockchain technologies introducing smart contract support Turing-complete programmability, such as Ethereum and Hyperledger. On the other hand, a series of works on transaction privacy are popular by applying cryptographic tools into Blockchain, such as Zerocash [26], Zerocoin [27] and Hawk [28]. Therefore, Blockchain technology's incentive feature and its pro-developing technologies inspire us to solve our scenario issues, such as the absence of incentive function and collaboration fairness.

2.2 Deep learning and distributed deep learning

A typical deep learning model consists of three layers, namely input layer, hidden layer and output layer. A deep learning model can contain multiple hidden layers, where the number of layers is called *depth* of the model. Each hidden layer can have certain number of neurons, and neurons at different layers can learn hierarchical features of the input training data, which represent different levels of abstraction. Each neuron has multiple inputs and a single output. Generally, the output of neuron i at layer $l-1$ connects to the input of each neuron at layer l. For the connection between two neurons, there is a weight assigned to it. For example, $w_{i,j}$ is a weight assigned to the connection between neuron i at layer $l-1$ and neuron j at layer l. Each neuron i also has a bias b_i. These weights and bias are called *model parameters*, which need to be learned during the training.

Back-Propagation (BP) [29] is the most popular learning method for deep learning, which consists of feed forward step and back-propagation step. Specifically, in feed forward step, the outputs at each layer are calculated based on parameters at previous layer and current layer, respectively.

A key component in deep neural network training is called *activation*, which is the output of each neuron. Activation is used to learn non-linear features of inputs via function $Act(\cdot)$. To compute the output value of a neuron i at layer l, $Act(\cdot)$ takes all the n inputs of i from layer $l-1$ as the input. In addition, we assume that weight $w_{j,i}$ is associated with the connection between neurons j at layer $l-1$ and neurons i at layer l, and b_i is the bias of neuron i. Then, the value of neuron i at layer l can be obtained by $Act_i(l) = Act_i(\Sigma_{j=1}^{n}(w_{j,i} * Act_j(l-1)) + b_i)$.

The back-propagation step employs gradient descent method, which gradually reduces the model error E_{total}, i.e., the gap between model output value V_{output} and the target value V_{target}. Assume that there are n output units at the output layer. Then, the gap can be calculated by $E_{total} = \frac{1}{2}\Sigma_{i=1}^{n}(V_{target_i} - V_{output_i})^2$. Once E_{total} is available, weights $w_{j,i}$ can be updated through $w_{j,i} = w_{j,i} - \eta * \frac{\partial E_{total}}{\partial w_{j,i}}$, where η is the learning rate and $\frac{\partial E_{total}}{\partial w_{j,i}}$ is the partial derivative of E_{total} with respect to $w_{j,i}$. This is the main idea of gradient descent method. The learning process repeats until the pre-specified number of iterations to train is reached.

When training a complex and multi-layer deep learning model, the aforementioned training procedure requires high computational overhead. To alleviate this problem, distributed deep learning training has been proposed recently, and some research work [30], [31], [32], [33], [34] and system implementations have been around, such as DistBelief [35], Torch [36], DeepImage [37] and Purine [38]. Generally, there are two approaches for distributed training, namely, model parallelism and data parallelism, where the former partitions a training model among multiple machines and the latter splits up the whole training dataset.

Our work focuses on the data parallelism approach, i.e., we have multiple machines and each machine maintains a copy of the training model while keeps a subset of the whole dataset as model input. These machines share the same parameters of the training model, by uploading/downloading parameters to/from a centralized parameter server. Then, machines upload their local training gradients, based on which the training model is updated by using (Stochastic Gradient Descent). They download updated parameters from the parameter server and continue to train the local model. This process repeats until machines obtain the final trained model.

3 Threats and Security Goals

In this section, we discuss threats to collaborative learning, and security goals that DeepBC can achieve to tackle those threats.

Threat 1: Disclosure of local data and model. Although in distributed deep training each party only uploads her local gradients to the parameter server, still adversaries can infer through those gradients important information about the party's local data by initiating an inference attack or membership attack [18]. On the other hand, based on the gradients adversaries may also launch parameter inferring attack to obtain sensitive information of the model [19].

Security Goal: Confidentiality of local gradients. Assume that participants do not expose their own data and at least t participants are honest (i.e., no more than t participants colluded to disclose parameters). Then each party's local gradients cannot be exposed to anyone else, unless at least t participants collude. In addition, if in any circumstance participants do not disclose the downloaded parameters from the collaborative model, then adversaries could not gain any information about the parameters. To achieve this goal, in DeepChain each participant individually encrypts and then uploads gradients obtained from her local model. All gradients are used to update parameters of the collaborative model encrypted collaboratively by all participants, who then obtain updated parameters via collaborative decryption in each iteration. Here, collaborative decryption means that at least t participants provide their secret shares to decrypt a cipher.

Threat 2: Participants with inappropriate behaviors. Consider a situation that participants may have malicious behaviors during collaborative training. They may choose their inputs at will and thus generate incorrect gradients, aiming to mislead the collaborative training process. As a

consequence, when updating parameters of collaborative model using the uploaded gradients, it is inevitable that we will get erroneous results. On the other hand, in collaborative decryption phase dishonest participants may give a problematic decryption share and they may be selfish, aborting local training process early to save their cost for training. In addition, dishonest participants may delay trading or terminate a contract for her own benefit, which makes the honest ones suffer losses. All these malicious behaviors may fail the collaborative training task.

Security Goal 1: Auditability of gradient collecting and parameter update. In DeepChain, assume that majority of the participants and more than $\frac{2}{3}$ of the workers are honest in gradient collecting and parameter update, respectively. During gradient collecting, participants' transactions contain encrypted gradients and correctness proofs, allowing the third party to audit whether a participant gives a correctly encrypted construction of gradients. For parameter update, on the other hand, workers claim computation results through transactions that will be recorded in DeepChain. These transactions are auditable as well, and computation results are guaranteed to be correct only if $\frac{2}{3}$ workers are honest. After parameters are updated, participants download and collaboratively decrypt the parameters by providing their decryption shares and corresponding proofs for correctness verification. Again, anyone third party can audit whether the decryption shares are correct or not.

Security Goal 2: Fairness guarantee for participants. DeepChain provides fairness for participants through timeout-checking and monetary penalty mechanism. Specifically, for each function with smart contracts DeepChain defines a time point for it. At the time point after function execution, results of the function are verified. If the verification failed, it means that (1) there exist participants not being punctual by the time point, and (2) some participants may incorrectly execute the function. For either of the two cases, DeepChain applies the monetary penalty mechanism, revoking the pre-frozen deposit of dishonest participants and re-allocating it to the honest participants. Therefore, fairness can be achieved, because penalty will never be imposed on honest participants behaved punctually and correctly, and they will be compensated if there exist dishonest participants.

4 THE DEEPCHAIN MODEL

In this section, we present DeepChain, a secure and decentralized framework for privacy-preserving deep learning.

4.1 System overview

Before introducing DeepChain, we give definitions of related concepts and terms used in DeepChain.

• **Party:** In DeepChain, a party is the same entity as defined in traditional distributed deep learning model, who has similar needs but unable to perform the whole training task alone due to resource constraints such as insufficient computational power or limited data.

• **Trading:** When a party got her local gradients, she sends out the gradients through a smart contract called *trading contract* to DeepChain. This process is called *trading*. Those contracts can be downloaded to process by *worker* (an entity in DeepChain that will be defined shortly).

• **Cooperative group:** A cooperative group is a set of parties who have a same deep learning model to train.

• **Local model training:** Each party trains her local model independently, and at the end of a local iteration the party generates a contract to trade by attaching her local gradients to the contract.

• **Collaborative model training:** Parties of a cooperative group train a deep learning model collaboratively. Specifically, after deciding a same deep learning model and parameter initialization, the model is trained in an iterative manner. In each iteration, all parties trades their gradients, and workers download the contracts to process the gradients. The processed gradients are then sent out by workers through smart contract called *processing contract*. The correctly processed gradients are used to update parameters of the collaborative model by the leader who is selected from workers. Parties download the updated parameters of the collaborative model and update their local models accordingly. After that parties begin next iteration of model training.

• **Worker:** Similar to *miners* in BitCoin, workers are incented to process transactions that contain training weights for collaborative model update. Workers compete to work on a block, and the first one finishes the job is a leader. The leader will gain block rewards that can be consumed in the future, for example, she may use rewards to pay for usage fee of trained models in DeepChain.

• **Iteration:** Deep learning model training consists of multiple steps called *iterations*, where at the end of each iteration all the weights of neurons of the model are updated once.

• **Round:** In DeepChain, a *round* refers to the process of the creation of a new block.

• **DeepCoin:** DeepCoin, denoted as $\$Coin$, is a kind of asset on DeepChain. In particular, for each newly generated block DeepChain will generate certain amount of $\$Coin$ as rewards. Participants in DeepChain consist of parties and workers, where the former gain $\$Coin$ for their contributions to local model training, and the latter are rewarded with $\$Coin$ for helping parties update training models. Meanwhile, a well-trained model will cost $\$Coin$ for those who have no capability to train the model by themselves and want to use the model. This setting is reasonable because recent work on model-based pricing for machine learning has found applications in some scenarios [39], [40]. We define a *validity value* for $\$Coin$, which essentially is the time interval of a round. Validity value is related to consensus mechanism in DeepChain, and we will discuss it in detail in 4.2.5.

DeepChain combines together Blockchain techniques and cryptographic primitives to achieve secure, distributed, and privacy-preserving deep learning. Suppose there are N parties P_j, $j = 1, ..., N$, and they agree on some predefined information such as a concrete collaborative model and initial parameters of the collaborative model. The information is attached to a transaction Tx^0_{co} signed by all parties. Assume the address corresponding to transaction Tx^0_{co} is pk_{it_0}, where it_0 is the initial iteration. At the end of

iteration i, the updated model in Tx^i_{co} is attached to a new address pk_{it_i}. All addresses are known to the parties.

Intermediate gradients from party P_j are enveloped in transaction $Tx^i_{P_j}$, and all those transactions are collected by a *trading contract* at round i. Note that intermediate gradients are local weights $C_{P_j}(\triangle \mathbf{W}_{i,j})$, where C is a cipher used by party P_j to encrypt the weights. When all transactions $\{Tx^i_{P_j}\}$ at round i have been collected, trading contract uploads them to DeepChain. After that, workers download those transactions $\{Tx^i_{P_j}\}$ to process via *processing contract*. Specifically, workers update the weights by computing $C(\mathbf{W}_{i+1}) = \frac{1}{N} \cdot C(\mathbf{W}_i) \cdot \prod_{j=1}^{N} C_{P_j}(\triangle \mathbf{W}_{i,j})$, where $C(\mathbf{W}_i)$ is the weight at round i in Tx^i_{co}, and $C(\mathbf{W}_{i+1})$ is the updated weights that will be attached to Tx^{i+1}_{co} for updating the local models in next round $i+1$.

4.2 Components of DeepChain

DeepChain consists of five building blocks that collectively achieve distributed and privacy-preserving deep learning, namely, DeepChain bootstrapping, incentive mechanism, asset statement, cooperative training and consensus protocol.

4.2.1 DeepChain bootstrapping

DeepChain bootstrapping consists of two steps, i.e., DeepCoin distribution and genesis block generation. Assume that all parties and workers have registered (i.e., having a valid account) in DeepChain, where each one uses an address pk that corresponds to a DeepCoin unit for launching a transaction.

In the first step, DeepCoin distribution realizes DeepCoin allocation among parties and workers, and initially each party or worker is allocated with the same amount of DeepCoins. Then in the second step, a genesis block is generated at round 0, which contains initial transactions recording ownership statements for each DeepCoin.

After the genesis block is created, a random seed $seed_0$ is also publicly known, which is randomly chosen by initially registered users through a routine for distributed random number generation. When DeepChain keeps running, at round i, $seed_{i-1}$ is used for generating $seed_i$. It is worth mentioning that these random seeds are crucial for DeepChain, because they guarantee randomness when selecting a leader to create a new block at each round. The idea of introducing random seeds is motivated by Algorand's cryptographic sortition [41], [42], and details will be given in Section 4.2.5.

4.2.2 Incentive mechanism

An incentive can act as a driving force for participants to actively and honestly take part in a collaborative training task, and the goal of incentive mechanism is to produce and distribute value, so that participant gets rewards or penalties based on her contribution. The introduction of incentive mechanism is crucial for collaborative deep learning, due to the following reasons. First, for those parties who want a deep learning model but have insufficient data to train the model on their own, incentive can motivate them to join the collaborative training with their local data. Second, with reward and penalty, incentive mechanism ensures that (1) parties are honest in local model training and gradient trading, and (2) workers are honest in processing parties' transactions.

For ease of understanding the incentive mechanism, we give an example consisting of two parties. These two parties contribute their data to collaborative training via launching transactions. Suppose the data possessed by the two parties is not equal in quantity. Each party can launch transactions and pay transaction fee based on the amount of data she owned. Generally, the large amount of data a party has, the less fee she will pay. The two parties agree on the total amount of fees for collaboratively training the model. The worker who successfully creates a new block when processing transactions can be the leader and earn the rewards. Note that transaction issuing and processing are verifiable, meaning that if some party poses an invalid transaction, the party would be punished. On the other hand, if a leader incorrectly processes a transaction, she will be punished accordingly. When collaborative training finished, parties themselves can benefit from the trained model that can bring revenue for them through charged services to those users who want to use the trained model.

To give a formal description of the incentive mechanism, we first introduce two properties, i.e., *compatibility* and *liveness* of the incentive mechanism for participants. Then, we further explain that parties and workers have incentive to behave honestly. Assume that we guarantee data privacy and security of the consensus protocol (explained in Section 4.2.5). We use v_c and v_i to denote the value of the trained collaboratively model and the trained individual model i, respectively, and we assume that v_c is greater than v_i.

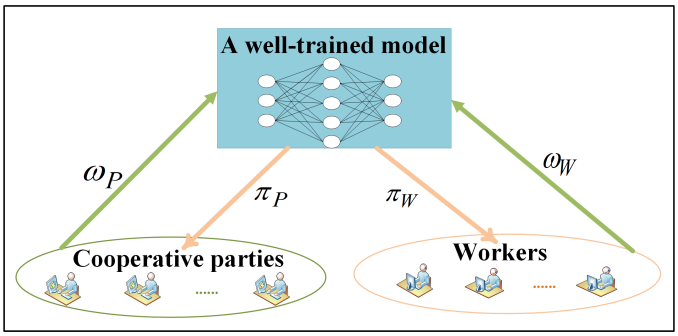


Fig. 2. The incentive mechanism of DeepChain, where ω_P and ω_W represent the contributions of a party and a worker for maintaining v_c, respectively, and π_P and π_W represent their payoffs, respectively.

First, we say the incentive mechanism exhibits compatibility if each participant can obtain the best result according to their contributions. Meanwhile, a participant has liveness only if she is willing to update her local training model with value v_i by continuously launching transactions and each worker also has incentive to update the parameters of the collaborative training model with value v_c. Below we describe importance of these two properties with respect to participant's true contribution and the corresponding payoff. Let ω_P and ω_W be the contributions of a party and a worker to the final trained model, respectively, and π_P and π_W be their corresponding payoffs, respectively. At first, we assume that participant's contribution originates from her

correct behaviors with an high probability, and later we will explain that this assumption is reasonable.

Liveness: both the party and the worker have the same common interest to obtain a trained collaborative model. Because if a party costs v_i during the whole training process, then she would gain v_c in the end, which is attractive for her because v_c is greater than v_i. On the other hand, a worker will process transactions for collaboratively constructing the training model in order to earn rewards with probability, with which she could pay for the deep learning services in DeepChain. Note that the probability a worker obtains reward depends on the quantity of rewards she has already earned. The larger the quantity, the higher probability she can get reward. As a result, both the party and the worker are incented to build the collaborative training model.

Compatibility: the more a party contributes ω_P, the more she will gain π_P. This holds for a worker too. During the collaborative training process, both party and worker incentively contribute best towards a training model $Max(\omega_P) \bigwedge Max(\omega_W)$, where the maximum total payoff is $Max(\pi_P) + Max(\pi_W)$. If any participant did not perform well, i.e., $(\omega_P = 0) \bigvee (\omega_W = 0)$, then there is no reward, i.e., $(\pi_P = 0) \bigwedge (\pi_W = 0)$. Here, $\bigwedge$ means '*and*' and $\bigvee$ means '*or*'. So we have

Payoff=

$$\begin{cases} Max(\pi_P) + Max(\pi_W) & \textbf{If} \quad Max(\omega_P) \bigwedge Max(\omega_W) \\ (\pi_P = 0) \bigwedge (\pi_W = 0) & \textbf{If} \quad (\omega_P = 0) \bigvee (\omega_W = 0) \end{cases}$$

Next, we explain the assumption that participant's contribution originates from her correct behaviors with a high probability. We show that each party and worker are value-driven to behave correctly in each round so that they could obtain the highest payoff [43]. The value is highest for a participant when she behaves correctly. We formalize it as $Value(1) = \pi_P - \omega_P(1)$ for a party, and similarly $Value(1) = \pi_W - \omega_W(1)$ is for a worker. The numeric value 1 means 100 percent. It then is replaced with $Pr_c(\cdot)$, which is used to measure the behavior correctness with probability of a participant. Thus, $\omega_P(1)$ means a party behaves correctly with probability 100 percent, and $Value(1)$ is the according value in this case, which is the expected case. Then, we say that ω_P has probability $Pr_c(P)$ to be provided correctly, and $Value(Pr_c(P))$ is related to $Pr_c(P)$ due to the party's actual behavior being verified. Assume that a method verifying a party's malicious behavior is correct with probability $Pr_v(P)$. Then the probability that a dishonest party would be caught is $Pr_v(P) * (1 - Pr_c(P))$. Once the dishonest party is caught, she is punished by forfeiting her deposit and the loss is denoted as f_P.

Thus, the final value according to the party's correct behavior can be represented as

$$\begin{aligned} &Value(Pr_c(P)) = \\ &\pi_P * (1 - Pr_{vc}(P)) - f_P * Pr_{vc}(P) - \omega_P * Pr_c(P) \end{aligned} \tag{1}$$

where $Pr_{vc}(P) = Pr_v(P) * (1 - Pr_c(P))$. The above value reaches maximum only when the party behaves honestly, i.e., $Pr_c(P) = 1$. Then $Value(1) = \pi_P - \omega_P(1)$ holds. This indicates the importance of the incentive mechanism. Specifically, the values of $Pr_v(P)$, π_P, and f_P can be determined through the following theorems.

TABLE 1
Summary of notations

Notations	Implications
pk_P^{psu}	a pseudo-generated public key of party P
sk_P	a secret key of the party P
q	a randomly selected big prime
G_1	cyclic multiplicative cyclic groups of prime order q
G_2	cyclic multiplicative cyclic groups of prime order q
g	a generator of group G_1
Z_q^*	{1,2, ..., q-1}
e	a bilinear map e: $G_1 \times G_1 \rightarrow G_2$
H_1	a collision-resistant hash function mapping any string into an element in Z_q^*
H_2	a collision-resistant hash function mapping any string into an element in G_1
$C()$	a cipher generated by Paillier.Encrypt algorithm
$Enc()$	the encryption by individual parties
$model_{co}$	collaborative deep learning model (collaborative model for short) to train

Theorem 1. If $f_P/\pi_P > (1 - Pr_{vc}(P))/Pr_{vc}(P)$, where $Pr_{vc}(P) = Pr_v(P) * (1 - \theta)$, then a party is honest at least with probability θ.

Proof. We need to prove that for any $Pr_c{}'(P) < \theta$, $Value(Pr_c{}'(P))$ is smaller than $Value(\theta)$. Without the loss of generality, we prove that for any $Pr_c{}'(P) < \theta$, we have $Value(Pr_c{}'(P)) < 0$. In other words, we have $Value(Pr_c{}'(P)) = \pi_P * (1 - Pr_{vc}{}'(P)) - f_P * Pr_{vc}{}'(P) - \omega_P(Pr_c{}'(P)) < 0$. When we set $f_P/\pi_P > 1/Pr_{vc}{}'(P) - 1$, then we have $\pi_P * (1 - Pr_{vc}{}'(P)) - f_P * Pr_{vc}{}'(P) < 0$. Thus, $Value(Pr_c{}'(P)) < 0$ holds.

For a worker, analysis of the incentive mechanism is similar to the above analysis for a party, expect that the worker's payoff is obtained with probability. We denote this probability by Pr_{leader}, then we could determine the relationship between the four values Pr_{leader}, $Pr_v(W)$, π_W, and f_W by the following theorem, so as to encourage a worker to be honest.

Theorem 2. If $f_W/\pi_W * Pr_{leader} > (1 - Pr_{vc}(W))/Pr_{vc}(W)$, where $Pr_{vc}(W) = Pr_v(W) * (1 - \epsilon)$, then a worker will be honest at least with probability ϵ.

Proof. The proof is similar to the proof of **Theorem** 1, so we omit it.

4.2.3 Asset statement

For ease of presentation, we list related cryptographic notations used in this section in Table 1. A party needs to state her asset, which enables her to find cooperators and accomplish her deep learning task. Asset statement does not reveal the content of asset, since it is simply some description of the asset, e.g., what kind of deep learning tasks the asset can be used for. Specifically, party P states an asset by sending an asset transaction, which is introduced later.

We recall the formation of a transaction. Note that a transaction is launched by a pseudo public key address pk_P^{psu} generated by P according to her wish in the following form.

$$pk_P^{psu} \in \{g_1^{sk_P}, g_2^{sk_P}, ..., g_n^{sk_P}\}$$

Here, n is an integer. P selects a secret key $sk_P \in Z_q^*$ and generates n public keys $g_i^{sk_P} \in G_1$, $i \in [1, n]$. q and g are

pre-specified parameters, and g_i equals to g^{r_i}, where r_i is a random element in Z_q^*. Suppose that party P_1 sends a transaction with her address $pk_{P_1}^{psu}$ to state her asset $data\_P_1$ as follows

$$Tran_{P_1} = pk_{P_1}^{psu} \rightarrow \Big\{ \Big(pk_{data\_P_1} = g^{H_1(data\_P_1)}, \sigma_{j\_P_1} = (H_2(j) \cdot g^{H_1(data_{j\_P_1})})^{H_1(data\_P_1)} \Big), \text{"Keywords"} \Big\}.$$

In this transaction, the first part in the braces consists of $pk_{data\_P_1}$ and $\sigma_{j\_P_1}$, which is the statement proof that party P_1 indeed possesses asset $H_1(data\_P_1)$ without leaking the content of $data\_P_1$. In particular, $\sigma_{j\_P_1}$ contains l components, where $data\_P_1$ is divided into l blocks represented by $data_{j\_P_1}$, $j \in [1, l]$. The second part "Keywords" is the description of the asset $data\_P_1$. In our implementation, "Keywords" is in `JSON` form that includes four fields, i.e., data size, data format, data topic and data description. With this transaction $Tran_{P_1}$, P_1 can fulfill her asset statement. We assume that the first stated asset is authentic, which is reasonable in Blockchain.

4.2.4 Collaborative training

Based on stated assets, parties who have similar deep learning task can constitute a collaborative group, and the collaborative training process consists of the following four steps.

• **Collaborative group establishment.** According to similar "Keywords", parties can establish a collaborative group. It is worth noting that parties may get more detailed information about "Keywords" through off-line interactions and this is not the focus of our paper. Before forming a collaborative group, parties can audit cooperator's asset to ensure authenticity of the asset ownership. The auditing process can be done by using the method in [44], and we omit the details for brevity.

Suppose there are N parties P_1, P_2, ..., P_N that constitute a group with pseudonymity, i.e., pseudo public keys $pk_{P_1}^{psu}, pk_{P_2}^{psu}, ..., pk_{P_N}^{psu}$ and their corresponding secret keys $sk_{P_1}, sk_{P_2}, ..., sk_{P_N}$ are privately kept, respectively. Since different party launches transactions using her own pseudo public key $pk_{P_i}^{psu}$, transactions signed by the according secret key sk_{P_i} can be verified to ensure that those transactions are from the same cooperative party P_i.

• **Collaborative information commitment.** After the collaborative group is formed, parties agree on the information which is for securely training a deep learning model. In this step, we assume that a trusted component (e.g., a trusted hardware like Intel SGX [45]) only takes part in the setup phase in Threshold Paillier algorithm [46], and it does not involve in other process. If there does not exist such a trusted component, we can accomplish the setup phase by using a distributed method such as the one in [47]. Parties agree on the following information.

(1) Number of cooperative parties, N.

(2) Index of the current round, r.

(3) Parameters of Threshold Paillier algorithm.

We have the following equation

$$PK_{model} = (n_{model}, g_{model}, \theta = as, V = (v, \{v_i\}_{i \in [1,...,N]}))$$

where modulus n_{model} is the product of two selected safe primes, and $g_{model} \in Z_{n_{model}^2}^*$, $a, s, \theta, v, v_i \in Z_{n_{model}}^*$. And $SK_{model} = s$ is randomly divided into N parts, where $s = f(s_1 + ... + s_N)$ and f is a function of secret sharing protocol. Each party owns a proportion of secure key s_i, v and $\{v_i\}$, $i \in [1, ..., N]$ are public verification information, where v_i corresponds to s_i. A threshold $t \in \{\frac{N}{2}, ..., N\}$ is set as such that more than t parties together can decrypt a cipher.

Note that training gradients to be encrypted are vectors with multiple elements, i.e., $\triangle \mathbf{W_{i,j}} = (w_{i,j}^1, ..., w_{i,j}^l)$ where the length of $\triangle \mathbf{W_{i,j}}$ is l, i is the index of current training iteration, and $j = 1, ..., N$. Due to the problem of cipher expansion, we encrypt a vector into one cipher instead of multiple ciphers with respect to multiple elements. Suppose that each value $w_{i,j}^1, ..., w_{i,j}^l$ is no larger than integer d, $d > 0$. We choose a l-length super increasing sequence $\vec{\alpha} = (\alpha_1 = 1, ..., \alpha_l)$ that simultaneously meets conditions (1) $\sum_{l=1}^{i-1} \alpha_l \cdot N \cdot d < \alpha_i, i = 2, ..., l$, and (2) $\sum_{i=1}^{l} \alpha_i \cdot N \cdot d < n_{model}$. We then compute $(g_{model}^1, ..., g_{model}^l) = (g_{model}^{\alpha_1}, ..., g_{model}^{\alpha_l})$.

(4) A collaborative model $model_{co}$ to be trained.

For a collaborative model $model_{co}$, parties agree on the training neural network, the training algorithms, and configurations of the network such as number of network layers, number of neurons per layer, size of mini-batch and number of iterations. Beside those information, they also reach a consensus on initial weights $\mathbf{W}_0$ of $model_{co}$. Note that weights $\mathbf{W}_i$ would be updated to $\mathbf{W}_{i+1}$ after the i-th iteration of training. They protect $\mathbf{W}_0$ by applying **Paillier.Encrypt** algorithm, i.e., $C(\mathbf{W}_0) = g_{model}^{\mathbf{W}_0} \cdot (k_0)^{n_{model}}$, where k_0 is randomly selected from $Z_{n_{model}}^*$. Note that we compute $g_{model}^{\mathbf{W}_0}$ with the help of the chosen super increasing sequence, i.e., $g_{model}^{\mathbf{W}_0} = g_{model}^{\alpha_1 \cdot w_0^1 + ... + \alpha_l \cdot w_0^l}$, so that we generate a cipher for weight vector $\mathbf{W}_0 = (w_0^1, ..., w_0^l)$.

(5) A commitment on $SK_{model} = s$, with respect to PK_{model}.

Commitment $commit_{SK_{model}}$ is obtained by combining parties' commitments on their secret shares s_i. Recall that r is the index number of the current round. We have

$$commit_{SK_{model}} = (Enc(s_1||r||Sign(s_1||r)), ..., Enc(s_N||r||Sign(s_N||r)))$$

here, $||$ denotes concatenation.

(6) The initial weights $\mathbf{W}_{0,j}$ of local model of party j.

Each party provides her local model's initial weights that are encrypted by **Paillier.Encrypt** algorithm, i.e., $C(\mathbf{W}_{0,j}) = g_{model}^{\mathbf{W}_{0,j}} \cdot (k_j)^{n_{model}}$, where $k_j \in Z_{n_{model}}^*$, $j \in \{1, ..., N\}$.

(7) A amount of deposits $d(\$Coin)$.

Each cooperative party is required to commit some amount of deposits for secure computation. During collaborative training, if a party misbehaves on purpose, her deposit $d(\$Coin)$ would be forfeited and compensated for other honest parties. Otherwise, those deposits would be refunded after the training process finished.

All the above collaborative information are recorded in a transaction $Tran_{co}$ that is uploaded to DeepChain. Specifically, $Tran_{co}$ is in the following form and is attached to a commonly coordinated address pk_{co}^{psu}.

$$Tran_{co} = pk_{co}^{psu} \rightarrow \Big\{ N, r, PK_{model}, d, \vec{\alpha}, model_{co}, commit_{SK_{model}}, C(\mathbf{W}_{0,j}), d(\$Coin) \Big\}.$$

In addition, two roles called *trader* and *manager* are defined for parties in a collaborative group, which will be explained shortly. Next we introduce how collaborative training is securely accomplished through the remaining two steps, namely, *Gradient collecting via Trading Contract* and *Parameter updating via Processing Contract*.

First of all, parties iteratively trade their gradients through *Trading Contracts* that are executed by a manager selected from cooperative parties. The trading gradients are honestly encrypted by each trader and meanwhile the correct proofs of encryption are attached that indicate two security requirements, i.e., confidentiality and auditability. Herein, we say gradient transactions are generated. In terms of confidentiality, if a trader does not disclose her gradients, then no one can gain information about the gradients. In addition, traders (at most t parties) need to cooperatively decrypt the updated parameters. Similar to [28], we assume that the manager does not disclose what she knows. In terms of auditability, there exist the proofs of correct encryption which can be auditable. When cooperatively decrypting, each trader presents her own decryption proof. Those proofs are generated non-interactively, and publicly auditable by any party on DeepChain.

Through timeout-checking and monetary penalty mechanism, behaviors of the traders and the manager are forced to be authentic and fair. Even if the manager colludes with traders, the outcome of *Trading Contract* cannot be modified [28]. In addition to *Trading Contract*, *Processing Contract* is responsible for parameter updating. Workers process transactions by adding up gradients, and send computation results to *Processing Contract*. *Processing Contract* verifies correct computation results and updates model parameters for the group. These two contracts are iteratively invoked, so as to accomplish the whole training process. Details of the two steps are given below.

• **Gradient collecting via Trading Contract.** As shown in Algorithm 1, *Trading Contract* invokes six functions, i.e., line 1, 4, 7, 10, 13 and 16 of Algorithm 1, for training $model_{co}$. At the end of each of the functions, we declare a time point T_{t_i} to check time-out events, and these six time points satisfy $T_{t_i} < T_{t_{i+1}}$, $i = 1, 2, ..., 5$. To set the time points uses the rule of Greenwich Mean Time, which is supported by programming. The time interval between T_{t_1} and T_{t_6} can be determined according to the time interval between two consecutive training iterations, e.g., for iteration i and $i+1$, we have $|T_{t_6} - T_{t_1}| \leq |T_{i+1} - T_i|$.

By the end of a time point T_{t_i}, function $checkTimeout$ check whether parties finish the according events or not by T_{t_i}. If some party is caught, the monetary penalty mechanism will be performed to forfeit deposit of the party, and the failed step is re-executed. During collaborative training, the six time points are updated accordingly with iterations, e.g., $T'_{t_1} = T_{t_1} + |T_{i+1} - T_i|$.

Algorithm 1 works as follows. As shown in line 1, at the i-th iteration each party P_j, $j \in \{1, ..., N\}$ sends an gradient transaction $Tran^i_{P_j}$ to $receiveGradientTX()$. A publicly auditable proof $Proof_{PK_{i,j}}$ is also attached to the transaction to guarantee encryption correctness. We have

$$Tran^i_{P_j} = \{pk^{psu}_{P_j} : (C(\triangle \mathbf{W}_{i,j}), Proof_{PK_{i,j}}) \rightarrow pk^{psu}_{co}\}$$
$$Proof_{PK_{i,j}} = fsprove_1(\Sigma_{PK}; C(\triangle \mathbf{W}_{i,j}); \triangle \mathbf{W}_{i,j}, k_j; pk^{psu}_{P_j})$$

Algorithm 1: Trading($Tran^i_{P_1}$,...,$Tran^i_{P_N}$)

1 **receiveGradientTX**()
2 checkTimeout(T_{t1})
3 updateTime() // $T'_{t_1} = T_{t_1} + |T_{i+1} - T_i|$
4 **verifyGradientTX**()
5 checkTimeout(T_{t2})
6 updateTime() // $T'_{t2} = T_{t2} + |T_{i+1} - T_i|$
7 **uploadGradientTX**()#attaching to the address pk^{psu}_{co}
8 checkTimeout(T_{t3})
9 updateTime() // $T'_{t3} = T_{t3} + |T_{i+1} - T_i|$
10 **downloadUpdatedParam**()#from the address pk^{psu}_{co}
11 checkTimeout(T_{t4})
12 updateTime() // $T'_{t4} = T_{t4} + |T_{i+1} - T_i|$
13 **decryptUpdatedParam**()
14 checkTimeout(T_{t5})
15 updateTime() // $T'_{t5} = T_{t5} + |T_{i+1} - T_i|$
16 **return**()
17 checkTimeout(T_{t6})
18 updateTime() // $T'_{t6} = T_{t6} + |T_{i+1} - T_i|$

Then in line 4, function $verifyGradientTX()$ verifies correctness of the encrypted gradients via function $fsver_1(\Sigma_{PK}; C(\triangle \mathbf{W}_{i,j}); Proof_{PK_{i,j}}; pk^{psu}_{P_j})$. Specifically, it verifies whether $C(\triangle \mathbf{W}_{i,j})$ is indeed the encryption of $\triangle \mathbf{W}_{i,j}$ with random number k_j. Here, $pk^{psu}_{P_j}$ can be regarded as the identity information attached to the proof, avoiding *replay attack* by a malicious party. In line 7, function $uploadGradientTX()$ uploads the transactions that have been verified successfully. When model parameter update finished, $downloadUpdatedParam()$ retrieves the latest parameters, as can be seen in line 10. Recall that *Processing Contract* computes gradients $\sum_{j=1}^{N} \triangle \mathbf{W}_{i,j}$ for model $model_{co}$.

Suppose that the latest iteration is i, the cipher of the latest parameters is $C(\mathbf{W}_i)$ and we denote it as C_i for brevity. Then $decryptUpdatedParam()$ collects parties' decryption shares on C_i for collaborative decryption, which generates $C_{i,j}$, $j \in 1, ..., N$. Meanwhile, the corresponding proofs for correct shares $Proof_{CD_{i,j}}$ are also provided, as follows.

$$C_{i,j} = C_i^{2\Delta s_j}$$
$$Proof_{CD_{i,j}} = fsprove_2(\Sigma_{CD}; (C_i, C_{i,j}, v, v_j); \Delta s_j; pk^{psu}_{P_j})$$

The proof $Proof_{CD_{i,j}}$ is used to verify validity of the decryption shares, i.e., $\Delta s_j = log_{C_i^4}(C_{i,j}^2) = log_v(v_j)$, through function $fsver_2(\Sigma_{CD}; (C_i, C_{i,j}, v, v_j); Proof_{CD_{i,j}}; pk^{psu}_{P_j})$. If majority of the parties are honest, i.e., $|H| >= N/2$, then C_i can be correctly recovered to plaintext by

$$((\Pi_{j \in H} C_{i,j}^{2\mu_j} - 1)/n_{model})(4\Delta^2\theta)^{-1} \mod n_{model}$$

where μ_j is the Lagrange interpolation coefficient with respect to P_j, and the plaintext is pushed to parties by function $return()$ in line 16.

• **Parameter updating via Processing Contract.** Algorithm 2 summarizes the process of *Processing Contract*, which contains three functions, as shown in line 1, 4, and 7. Suppose that at the i-th iteration of collaborative training, local

Algorithm 2: Processing()

1 **updateTX**()
2 checkTimeout(T_{t7})
3 updateTime() // $T'_{t7} = T_{t7} + T_r$
4 **verifyTX**()
5 checkTimeout(T_{t8})
6 updateTime() // $T'_{t8} = T_{t8} + T_r$
7 **appendTX**()
8 checkTimeout(T_{t9})
9 updateTime() // $T'_{t9} = T_{t9} + T_r$

gradients $C(\triangle \mathbf{W}_{i,j})$, $j \in \{1, ..., N\}$, have been uploaded, then workers competitively execute update operations by

$$C(\mathbf{W}_i) = C(\mathbf{W}_{i-1}) \cdot \frac{1}{N} \cdot (C(- \triangle \mathbf{W}_{i,1}) \cdot C(- \triangle \mathbf{W}_{i,2}) \cdot \\ ... \cdot C(- \triangle \mathbf{W}_{i,N})).$$

Once update operation finished, workers then send the updated results through transactions to function $updateTX()$ in *Processing Contract*, as shown in line 1.

At the meantime, a leader is randomly chosen from the workers by using the consensus protocol of DeepChain (we will discuss it in Section 4.2.5). Note that at this moment we defer the reward to the leader until her computational work is verified by using function $verifyTX()$ as shown in line 4, that employs majority voting policy. In other words, the leader's computational result $C(\mathbf{W}_i)$ will be compared against those of the other workers, and her result is admitted only if the majority of the workers produce the same result. Otherwise, the leader would be punished according the monetary penalty mechanism and she gains no reward. In such case, we repeat the procedure to chosen a new leader from the remaining workers. The more often a worker is punished, the lower probability she will be chosen as a leader. Once we get a legitimate leader, her block with correctly updated result is appended to DeepChain through $appendTX()$, as shown in line 7.

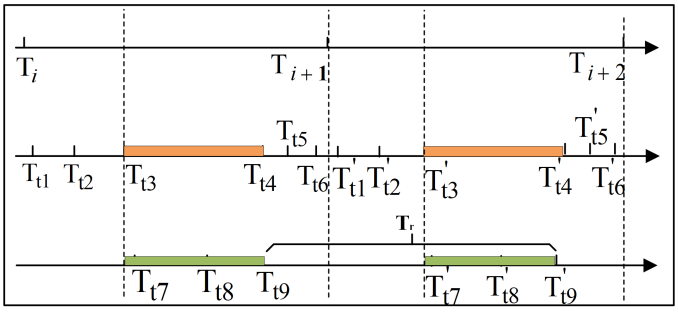


Fig. 3. Configuration of time points. From top to bottom: (1) the timeline of collaborative training, (2) the timeline of trading (in Trading Contract), (3) the timeline of block creation (in Processing Contract).

In *Processing Contract*, time points T_{t7}, T_{t8} and T_{t9} will be updated to $T'_{t7} = T_{t7} + T_r, T'_{t8} = T_{t8} + T_r$ and $T'_{t9} = T_{t9} + T_r$, respectively, where T_r is the time needed to create a new block between consecutive rounds in DeepChain. Figure 3 gives an example of time point configuration scheme to illustrate relationship of time points of the trading and processing contracts. Suppose that at the i-th iteration, the time points are set as such that $T_{t_1} < T_{t_2} < T_{t_3} \leq T_{t_7} < T_{t_8} < T_{t_9} \leq T_{t_4} < T_{t_5} < T_{t_6}$. At the meantime, the relationship between the three time intervals is $T_r \leq |T_{t_6} - T_{t_1}| \leq |T_{i+1} - T_i|$.

Algorithm 3: $F^*_{GradientCollecting}$

1 Receive (*input*, $sid, T_t, pk^{psu}_{P_j}, C(\triangle \mathbf{W}), Proof_{PK_j}, d(\$Coin)$) from $pk^{psu}_{P_{j\in\{1,...,N\}}}$. Assert time $T_t < T_{t_1}$. Receive (*input*, $sid, T_t, pk^{psu}_{P_{j\in\mathbb{C}}}, C(\triangle \mathbf{W}), Proof_{PK_j}, H'$, $h' \times d(\$Coin)$) from $\mathcal{S}$. Assert time $T_t < T_{t_1}$.
2 Compute $fsver_1(C(\triangle \mathbf{W}), Proof_{PK_j})$ for $pk^{psu}_{P_{j\in\{1,...,N\}}}$, and record $\{1, ..., N\} \setminus \mathbb{C}'$.
3 Send(*return*, $d(\$Coin)$) to $pk^{psu}_{P_{j\in\{1,...,N\}\setminus\mathbb{C}'}}$ after T_{t_1}.
4 If $\mathcal{S}$ returns (continue, H''), then send (*output*, Yes or No) to $pk^{psu}_{P_{j\in\{1,...,N\}}}$, and send (*payback*, $(h - h'')d(\$Coin)$) to $\mathcal{S}$, and send (*extrapay*, $d(\$Coin)$) to $pk^{psu}_{P_{j\in H''}}$, else if $\mathcal{S}$ returns (abort), send (*penalty*, $d(\$Coin)$) to $pk^{psu}_{P_{j\in\{1,...,N\}}}$.

Algorithm 4: $F^*_{CollaborativeDecryption}$

1 Receive (*input*, $sid, T_t, pk^{psu}_{P_j}, C, C_j, Proof_{CD_j}, d(\$Coin)$) from $pk^{psu}_{P_{j\in\{1,...,N\}}}$. Assert time $T_t < T_{t_5}$. Receive (input, $sid, T_t, pk^{psu}_{P_j} \in \mathbb{C}, C, C_j, Proof_{CD_j}, H'$, $h' * d(\$Coin)$) from $\mathcal{S}$. Assert time $T_t < T_{t_5}$.
2 Compute $fsver_2(C, C_j, Proof_{CD_j})$ for $pk^{psu}_{P_{j\in\{1,...,N\}}}$ and record $\{1, ..., N\} \setminus \mathbb{C}'$.
3 Send(*return*, $d(\$Coin)$) to $pk^{psu}_{P_{j\in\{1,...,N\}\setminus\mathbb{C}'}}$ after T_{t_5};
4 If $\mathcal{S}$ returns (continue, H''), then send (*output*, Yes or No) to $pk^{psu}_{P_{j\in\{1,...,N\}}}$, and send (*payback*, $(h - h'')d(\$Coin)$) to S, and send (*extrapay*, $d(\$Coin)$) to $pk^{psu}_{P_{j\in H''}}$, else if S returns (abort), send (*penalty*, $d(\$Coin)$) to $pk^{psu}_{P_{j\in\{1,...,N\}}}$.

In addition to the above configuration scheme for time points, we employ secure monetary penalty mechanism to guarantee fairness in gradient collecting and collaborative decryption. Specifically, enlightened by the penalty mechanism proposed by Bentov *et al* [48] and Kumaresan *et. al* [49], we design our secure monetary penalty mechanism based on *Trading Contract*, presented in Algorithm 3 and Algorithm 4.

In particular, in *Gradient collecting* (Algorithm 3) fairness is guaranteed due to (1) honest collaborative parties must launch gradient transactions that must be correctly verified before the pre-specified time point, and (2) dishonest parties who launch incorrect transactions or **transactions which are overtime**, will be penalized, with the honest ones being compensated for. In line 1, *Trading Contract* waits to receive a *input* message from $pk^{psu}_{P_j}$ for all $j \in \{1, ..., N\}$ before time T_{t_1}. By defining $\mathbb{C} \subseteq \{1, ..., N\}$ as adversarial parties $\mathcal{S}$ in *input* step, the contract also waits a *input* message from $\mathcal{S}$. Here, sid is session identifier and $d(\$Coin)$ is deposits. H' means the set of the remaining honest parties

and $|H'| = h'$. In line 2, the contract verifies the ciphertext for all $pk_{P_j}^{psu} \in H'$, and records correct parties as $\{1, ..., N\} \backslash \mathbb{C}'$, where $\mathbb{C}'$ means corrupted parties in this step. In line 3, the contract sends *return* messages to $pk_{P_j}^{psu}$ for $j \in \{1, ..., N\} \setminus \mathbb{C}'$. In line 4, waiting a return message from $\mathcal{S}$, if the return message is continue, the contract outputs normally to all $pk_{P_j}^{psu}$ ($j \in \{1, ..., N\}$), sends *payback* message to $\mathcal{S}$, and sends *extrapay* to $pk_{P_j}^{psu}$ in $H^{''}$, where $H^{''}$ means $H' \setminus \mathbb{C}'$, $|H^{''}| = h^{''}$, otherwise, the contract sends *penalty* to $pk_{P_j}^{psu}$ ($j \in \{1, ..., N\}$).

Similarly, fairness is also achieved in *Collaborative decryption* (Algorithm 4), since (1) a party who gives a correct decryption share no later than the pre-defined time point receives no penalty, and (2) If an adversary successfully decrypts the cipher but a legitimate party fails to do so, then the party should be compensated for.

4.2.5 Consensus protocol

Consensus protocol is essential in DeepChain, since it enables all participants to make a consensus upon some event in a decentralized environment. In this section, we present blockwise-BA protocol of DeepChain, based on the work of Algorand [41], [42]. blockwise-BA protocol includes three main steps — (1) A leader who creates a new block is randomly selected by using cryptographic sortition, (2) A committee, consisting of participants whose transactions are included in the new block, verifies and agrees on the new block by executing a Byzantine agreement protocol [50], and (3) Each verifier in the committee tells neighbors the new block by using a gossip protocol [51], [52], so that the new block is known to all participants in DeepChain.

Our consensus protocol meets three properties, *safety*, *correctness*, and *liveness*. In particular, safety means that all honest parties agree on a same transaction history in DeepChain, whereas correctness requires that any transaction agreed by a honest party comes from a honest party. Liveness says that parties and workers are willing to continuously perform activities in DeepChain, hence keeping DeepChain alive. Based on these three properties, we assume that message transmission is synchronous and at most $\frac{2}{3} * \$Coin$ are possessed by honest parties. In this setting, all parties agree on a chain with the largest amount of assets. We give details of the three steps of our consensus protocol below. Suppose block $block_i$ is created at round r_i.

• **Leader selection.** At round r_i, a leader $leader_i$ is randomly chosen from workers who collect transactions into block $block_i$. To choose a leader, we invoke the sortition function of Algorand [41], which includes two functions *leader selection* and *leader verification*, as follows

$$Sortition(sk, seed_i, \tau = 1, role = worker, w, W) \rightarrow \langle hash, \pi, j \rangle$$
$$VerifySort(pk, hash, \pi, seed_i, \tau, role = worker, w, W) \rightarrow j$$

here, sk and pk are owned by worker, and $seed_i$ is a random seed selected based on $seed_{i-1}$, i.e., $seed_i = H(seed_{i-1}||r_i)$, where H is an hash function. $\tau = 1$ means that only one leader is selected from workers $role = worker$. w represents the amount of $\$Coins$ that the participant possesses. It is worth mentioning that w is crucial, because it is used to control the probability that worker can gain reward according to the amount of rewards she has already earned (see section 4.2.2).

Our definition of w is different from that of Alogrand, in that in our Leader Selection w only contains $\$Coins$ that have available validity value, while those without validity value are not considered. In this way, we can eliminate the phenomenon of wealth accumulation, in which a rich participant may become richer because she has a higher probability than her peers to be chosen as the leader. Parameter W is the total amount of $\$Coins$ in DeepChain. Through the two functions, we can randomly select a leader and all participants can also verify whether the selected leader $leader_i$ is legitimate.

• **Committee agreement.** After leader verification, the selected block $block_i$ is sent to the committee. Each participant in the committee verifies the transactions processed by $leader_i$, i.e., to verify whether weight update operations are correct or not. If the committee admits that $block_i$ is right based on a majority voting policy, then participants sign $block_i$ on behalf of the committee; otherwise, $block_i$ is rejected. Note that $block_i$ is valid only if more than $\frac{2}{3}$ of the committee members signed and agreed on it. If $block_i$ is valid, then $leader_i$ gains $\$Coins$ from block reward and transaction coins of $block_i$; otherwise, $block_i$ is discarded and a new empty block is created to replace $block_i$ in DeepChain. This process repeats until the committee agrees on $block_i$.

• **Neighbor gossip.** Suppose $block_i$ has been agreed on by the committee, then participants in the committee are responsible for telling their neighbors $block_i$, by using the popular gossip protocol [51], [52]. Therefore, after this step all participants arrive at a consensus in DeepChain.

5 Security Analysis

In this section, we revisit our security goals of DeepChain presented in section 3 and give security analysis for them.

• **Confidentiality guarantee for gradients.** To achieve this goal, DeepChain employs Threshold Paillier algorithm that provide additive homomorphic property. We assume there exists a trusted setup in Threshold Paillier algorithm [45], and it does not involve in other process. and the secret key cannot leak without collaboration of at least t participants. We also assume that at least t participants are honest. Without loss of generality, both local gradients and model parameters W are encrypted with the Threshold Paillier algorithm, i.e., $C(W) = g_{model}^{W}(k)^{n_{model}}$. Based on the following lemma that is derived from the work [46]'s **Theorem 1**, we can guarantee confidentiality of local gradients and model parameters.

Lemma 1. With the Decisional Composite Residuosity Assumption (DCRA) [53] and the random oracle model $\mathcal{S}$, Threshold Paillier algorithm is t-robust semantically secure against active non-adaptive adversaries $\mathcal{A}$ with polynomial time power to attack, if the following are satisfied.

$$\begin{aligned}&\mathbf{Pr}\big[(w_0, w_1) \leftarrow \mathcal{A}(1^\lambda, F^t(\cdot));\\ &b \leftarrow \{0, 1\};\\ &C \leftarrow \mathcal{S}(1^\lambda, w_b) :\\ &\mathcal{A}(C, 1^\lambda, F^t(\cdot)) = b\big] \leq negl(1^\lambda) + \tfrac{1}{2}\end{aligned}$$

It indicates that the probability for adversaries to distinguish w_0 or w_1 is negligible, in λ which is the system security parameter. Here, $F^t(\cdot)$ means that $\mathcal{A}$ has at most t corrupted parties and learns their information including public parameters, secret shares of the corrupted parties, public verification keys, all decryption shares and validity of those shares. In addition, t-robust means that a Threshold Paillier ciphertext can be correctly decrypted, even in the case that $\mathcal{A}$ can have up to t corrupted parties. Semantic security is a general security proof methodology to measure the security of an encryption algorithm and in our context it measures confidentiality of the encrypted information by using the Threshold Paillier algorithm.

• **Auditability of gradient collecting and parameter update.** This goal ensures that any party can audit the correctness of encrypted gradients and decryption shares in gradient collecting and parameter updating. Recall that we introduce *non-interactive zero-knowledge proof* for these two processes, according to the universally verifiable CDN (UVCDN) protocol [54]. Specifically, four algorithms are presented in section 4.2.4, such as $fsprove_1$, $fsver_1$, $fsprove_2$, $fsver_2$.

Under the framework of UVCDN protocol, public auditability can be guaranteed if there exists a simulator that can simulate correctness proofs of the honest parties and extract witnesses of the corrupted parties. Next, we use *Lemma* 2 and *Lemma* 3 to show that the correctness proof of encrypted gradients is auditable. Similarly, auditability of the correctness proof of decryption shares can be guaranteed under the UVCDN framework and we omit it for brevity.

Lemma 2. Given $X = C(x)$, $x = \triangle\mathbf{W}$, and $c \in \mathbb{C}$ where $\mathbb{C}$ is a finite set called the challenge space, then we have
$\{d \in_R Z_{n_{model}}; e \in_R Z^*_{n_{model}}; a := g^d_{model} e^{n_{model}} X^{-c}: (a; c; d, e)\}$
$\approx$
$\{a_1 \in_R Z_{n_{model}}; b_1 \in_R Z^*_{n_{model}}; a := g^{a_1}_{model} b_1^{n_{model}}; t := (a_1 + cx)/n_{model}; d := a_1 + cx; e := b_1 k_j^c g^t_{model}: (a; c; d, e)\}$

where symbol $\approx$ means that the two distributions are statistically indistinguishable.

Lemma 3. Let $X = C(x) = g^x_{model} r^{n_{model}}$, where $x = \triangle\mathbf{W}$ and $r = k$. Given $(a; s)$ that is generated by the announcement $\Sigma_{PK}.ann$ and a challenge c with respect to the announcement, there exists an extractor $\mathcal{E}$ that can extract the witness of an adversary $\mathcal{A}$, if $\mathcal{A}$ can present two conversations for $(a; s)$, that is,

$$|1 - \mathbf{Pr}[\mathcal{A}(X; x, r; a; s; c) \rightarrow (d, e; d', e'); \mathcal{E}(X; a; d, e; d', e') \rightarrow (x', r') = (x, r)]| \leq negl(1^\lambda)$$

The above formula says that the probability for the extractor $\mathcal{E}$ failing to extract the witness of an adversary is negligible, in system security parameter λ.

• **Fairness guarantee for collaborative training.** Recall that we employ two security mechanisms in Blockchain to enhance fairness during collaborative training, namely the trusted time clock mechanism and secure monetary penalty mechanism. With the trusted time clock mechanism, operations in a contract are forced to finish before respective time point, as can be seen through function $checkTimeout()$ in Algorithm 1 and 2. On the other hand, we define two secure monetary penalty functions for gradient collecting and collaborative decryption, respectively.

To explain these two mechanisms, we introduce the concept of *secure computation with coins (SCC security) in a multi-party setting*, proposed and proved in [48], [49] in a hybrid model as follows.

Lemma 4. Given an input z, security parameter λ, a distinguisher Z, an ideal process IDEAL, an ideal adversary S in IDEAL, an ideal function f, and a protocol π that interacts with ideal function g in a model with adversary A, then we have
$\{\texttt{IDEAL}_{f,S,Z}(\lambda, z)\}_\lambda \in \mathbb{N}, z \in 0, 1^*$
$\equiv_c$
$\{\texttt{HYBRID}_{g,\pi,A,Z}(\lambda, z)\}_\lambda \in \mathbb{N}, z \in 0, 1^*$

where $\equiv_c$ means that the distributions are computationally indistinguishable.

Lemma 5. Let π be a protocol and f a multiparty function. We say that π securely computes f with penalties if π SCC-realizes the functionality f^*.

According to ***Lemma*** 5, we require that a protocol π SSC-realizes F as $F^*_{GradientCollecting}$ and $F^*_{CollaborativeDecryption}$, meaning that they achieves secure gradient collecting and collaborative decryption with penalties, respectively. With these two functionalities and the trusted time clock mechanism, we can achieve fairness for gradient collecting and collaborative decryption, as shown in Algorithm 3 and Algorithm 4.

6 Implementation and Evaluation

In this section, we implement our DeepChain prototype. First, we build a Blockchain to simulate DeepChain. Blockchain nodes are regarded as parties and workers, and they participate in trading and interact with two pre-defined smart contracts, i.e., *Trading Contract* and *Processing Contract*. Generated transactions are serialized in the Blockchain.

We use Corda V3.0 [55] to simulate DeepChain for its adaptability and simplification. Specifically, Corda project is created by R3CEV and has been widely used in banks and financial institutes. It is a decentralized ledger that has some features of Bitcoin and Ethereum [56], such as data sharing based on need-to-know basis and deconflicting transactions with pluggable notaries. A Corda network contains multiple notaries, and our consensus protocol introduced in section 4.2.5 can be executed on them. We build nodes and divide them into parties and workers. Specifically, we set up two CorDapps which agree on the Blockchain. The nodes of one CorDapp serve as parties, and the nodes of another one play the role of workers. According to the application program interface (API) of Corda, we implement our business logic by integrating three main components, such as *State*, *Contract*, *Flow*. In particular, a instance of State is used to represent a fact of a kind of data, and it is immutable once a instance of State is known by all nodes at a specific time point. Contract is used to instantiate some rules on transactions. A transaction is considered to be contractually valid if it follows every rule of the contact. A instance of

TABLE 2
Training configuration

Parameter	value
No. of iterations	1500
No. of epochs	1
Learning rate	0.5
Minimal batch size	64

Flow can define a sequence of steps for ledger updates, e.g., how to launch a transaction from a node to another node.

We build the deep learning environment with Python version 3.6.4, numpy version 1.14.0, and tensorflow version 1.7.0. We select the popular MINIST dataset [57] which contains $55,000$ training samples, $5,000$ verification samples and $10,000$ test samples. Then, we split randomly this dataset into 10 subsets, each one of which has the size of $5,500$ (i.e., $55,000/10$). The number of subset is according to the maximum number of party we consider. Then, we conduct multiple training experiments with 4, 5, 6, 7, 8, 9, 10 parties (E-4, E-5, E-6, E-7, E-8, E-9, E-10 as the abbreviation), respectively. In each experiment, each party possesses one subset of dataset. Our training model derives from Convolution Neural Network (CNN) with structure: `Input` $\rightarrow$ `Conv` $\rightarrow$ `Maxpool` $\rightarrow$ `Fully Connected` $\rightarrow$ `Output`. The weights and bias parameters in Conv layer, Fully Connected layer and Output layer are $w_1 = (10, 1, 3, 3)$ and $b_1 = (10, 1)$, $w_2 = (1960, 128)$ and $b_2 = (1, 128)$, $w_3 = (128, 10)$ and $b_3 = (1, 10)$, respectively. We summarize other training parameters in Table 2. Note that No. of iterations for each experiments are distinct and larger for the experiments with more parties.

Threshold Paillier algorithm is implemented in JAVA within 160-line codes. We set the number of bits of modulus n_{model} to 1024 bits, which is for a security level of 80 bits. It is worth noting that before executing the encryption algorithm, the weight matrices are assembled into a vector, so that only one cipher is generated for a party.

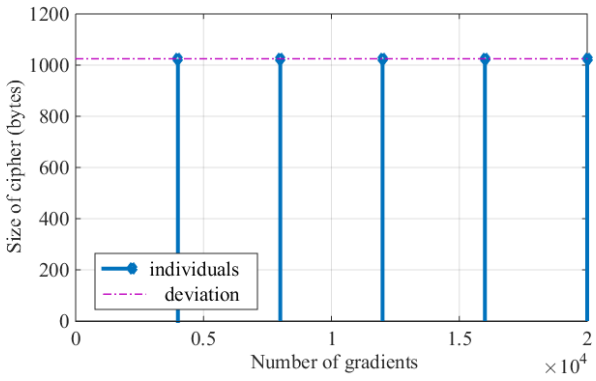


Fig. 4. Impact of No. of gradients on cipher size.

In a word, we implement the above building blocks into three modules, i.e., CordaDeepChain, TrainAlgorithm, and CryptoSystem. We evaluate the feasibility of model training on DeepChain in a multi-party setting by using 4 metrics such as ciphertext size, throughput, training accuracy and total cost of time. Particularly, we evaluate DeepChain on a desktop computer with 3.3GHz Intel(R) Xeon(R) CPU and 16 GB memory. Then, we record final results on average for each metrics with 10-time evaluation. As can be seen in Figure 4, the size of cipher remains constant when we encrypt different amounts of gradients. On the other hand, as the number of gradients increases, the throughput decrease steadily, as shown in Figure 5.

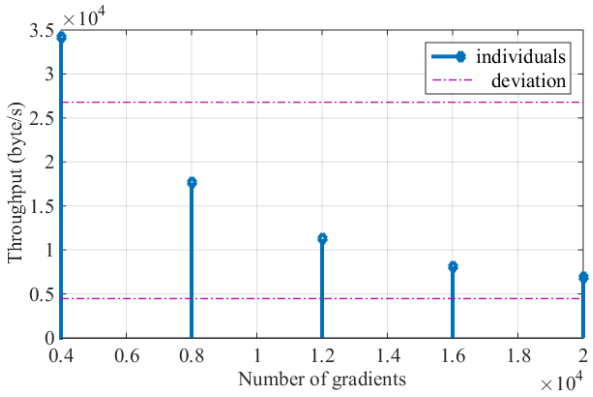


Fig. 5. Impact of No. of gradients on throughput.

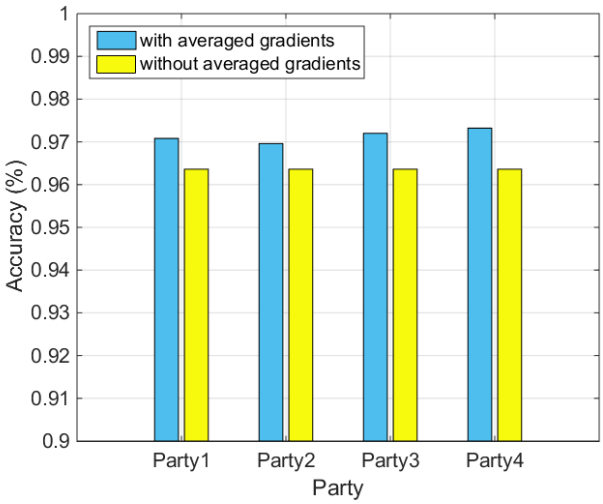


Fig. 6. Training accuracy for the case of four parties.

In terms of training accuracy, we demonstrate that the more party participate in, the higher training accuracy is obtained. We separately create 4, 5, 6, 7, 8, 9, 10 parties

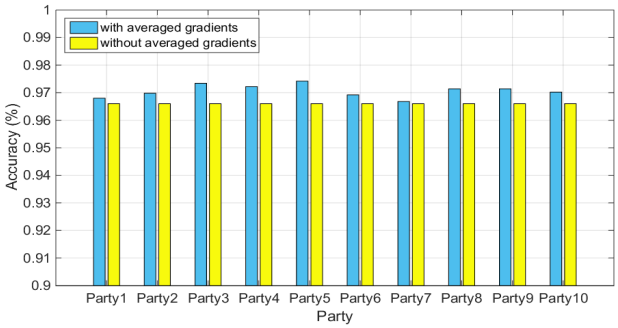


Fig. 7. Training accuracy for the case of ten parties.

TABLE 3
Interpretations of time variants

Variant	interpretation
$t_{15\_iteration}$	time of 15 iterations
$t_{encrypt}$	time of encrypting gradients
$t_{uploadByParty}$	time of uploading gradients
$t_{downloadByWorker}$	time of downloading gradients from all parties
$t_{average}$	time of averaging all gradients
$t_{uploadByWorker}$	time of uploading updated parameters
$t_{downloadByParty}$	time of downloading parameters
$t_{decrypt}$	time of decrypting parameters

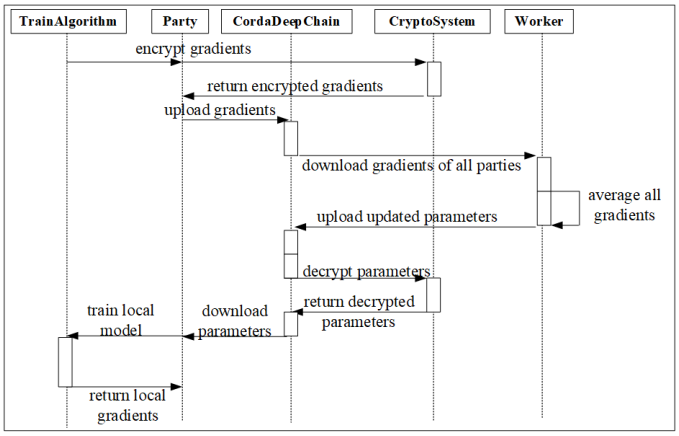


Fig. 8. Interaction in the entire collaborative training process.

participating in the collaborative training and trading. Each party trains the local model with her training dataset that contains 5,500 samples in each separating experiment. Obviously, the more party there are, the larger the size of the total dataset is (i.e., the size of the total dataset is 5,500*4 for E-4, and 5,500*10 for E-10). By sharing gradients on DeepChain, each individual party obtains updated parameters contributed by the gradients from other parties. Specifically, the updated parameters are calculated by averaging gradients of all parties. We also create an external party, denoted as baseline party, who only trains the local model on her dataset (with 5,500 samples) without the averaged gradients from other parties. With the limitation of space, we give out the results of training accuracy in E-4 and E-10 shown in Fig 6 and 7, respectively. We can see that for both cases the training accuracy in collaborative training are higher than the accuracy obtained by the baseline party. In Fig 6, the accuracy of the baseline party with yellow bar is 96.36% while Party 1, Party 2, Party 3, and Party 4 with blue bar have accuracy of 97.08%, 96.96%, 97.20% and 97.32%, respectively. In Fig 7, the accuracy of the baseline party is 96.60% and meanwhile the accuracy from Party 1 to Party 10 are 96.80%, 96.98%, 97.34%, 97.22%, 97.42%, 96.92%, 96.68%, 97.14%, 97.14%, 97.02%, respectively.

At last, we measure the corresponding total time costs for E-4, E-5, E-6, E-7, E-8, E-9 and E-10. Note that we use the same training model on MNIST dataset in each experiment. This total time is the time cost for parties participating in the entire collaborative training process on DeepChain. We depict the process in Fig 8 by using an interaction diagram when implemented. It is worth noting

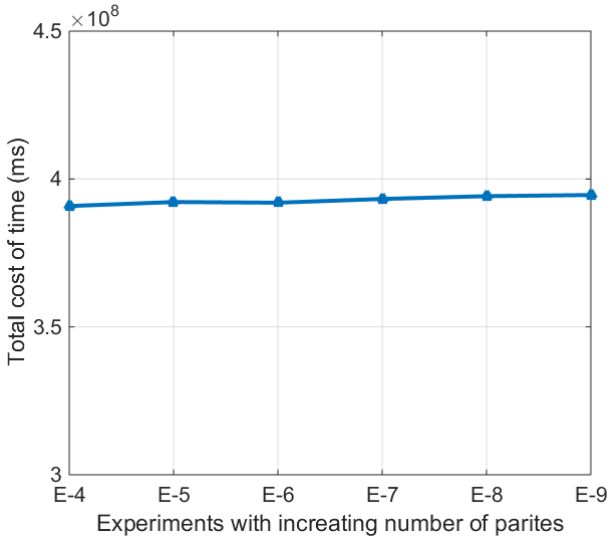


Fig. 9. Total cost of time in different experiments with increasing number of parites.

that, for efficiency, parties only share gradients with 100 times instead of No. of iterations in each experiment. Thus, we let the frequency of sharing gradient be $\#share$, and the number of iteration be $\#iteration$. Then, the total time cost can be measured by $Time \approx \frac{\#iteration}{15} \times (t_{15\_iteration}) + \#share \times (t_{encrypt} + t_{uploadByParty} + (t_{downloadByWorker} + t_{average} + t_{uploadByWorker}) + t_{downloadByParty} + t_{decrypt})$. Herein, the interpretations for each time variant are concluded in Table 3. It is worth noting that the time cost also is determined by the size of the training model. With the increasing size of the training model, both the number of iteration and the time of cryptographic operations become longer. In our experiments, using the training model we mentioned above, the total cost of time grows unobviously with the number of party increasing as can been seen in Fig 9. Specifically, the total time costs for E-4, E-5, E-6, E-7, E-8, E-9, E-10 are $390723965ms$, $390854250ms$, $392225357ms$, $391992180ms$, $393250120ms$, $394207507ms$ and $394593160ms$ at estimate, respectively. The reasons for the unobvious growth is that the increasing number of party makes the time $t_{downloadByWorker}$, $t_{average}$ and $t_{downloadByParty}$ turn slightly longer, but the increased amount of time is unbovious for the total cost of time. The large proportion of the total time is $t_{encrypt}$ and $t_{decrypt}$. In addition, $t_{encrypt}$ and $t_{decrypt}$ are relative to the size of the training model instead of the number of parties.

7 Conclusion and Future Work

In this paper we present DeepChain, a robust and fair decentralized platform based on Blockchain for secure collaborative deep training. Specifically, we introduce an incentive mechanism and achieve three security goals, namely confidentiality, auditability, and fairness.

We formalize the incentive mechanism based on Blockchain in terms of compatibility and liveness. Furthermore, in the context of the incentive mechanism, we demonstrate participants are incentive to behave correctly

with high probability. For confidentiality of local gradients, we employ Threshold Paillier algorithm to protect data. By applying a skillful component into the encryption algorithm, the goal that only one cipher is generated for a party is achieved. In addition to confidentiality, we provide auditability and fairness, by addressing the issue that malicious participants may disrupt the collaborative training process. We integrate the tool of non-interactive zero-knowledge proof to provide auditability of the collaborative training process, and timeout-checking and monetary penalty mechanism of Blockchain to push participants to behave fairly. We finally implement a prototype of DeepChain and evaluate it to illustrate the feasibility from 4 aspects including ciphertext size, throughput, training accuracy and training time.

Next, we desire to discuss the significance of DeepChain in a long-term way. DeepChain stores data which includes not only iterative training parameters but also trained models. On the one hand, it is obvious that trained models create financial values when the model-based pricing market is promising. This brings participants who possess trained models with long-term benefits, since their models can provide service to those who have AI tasks by the way of payment. On the other hand, all training processes and model parameters of each iteration are recorded, which could advance the development of transfer learning. Thus, in the second aspect, we take the first-step consideration that DeepChain can extend the potential value of models to transfer learning. An intuition is that trained models can be applied to train a new model for a different but related AI task. In this case, the security problem should be re-defined, which will be discussed in the future work.

ACKNOWLEDGEMENT

This work was supported by National Key R&D Plan of China (Grant No. 2017YFB0802203 and 2018YFB1003701), National Natural Science Foundation of China (Grant Nos. U1736203, 61732021, 61472165 and 61373158), Guangdong Provincial Engineering Technology Research Center on Network Security Detection and Defence (Grant No. 2014B090904067), Guangdong Provincial Special Funds for Applied Technology Research and Development and Transformation of Important Scientific and Technological Achieve (Grant No. 2016B010124009), the Zhuhai Top Discipline–Information Security, Guangzhou Key Laboratory of Data Security and Privacy Preserving, Guangdong Key Laboratory of Data Security and Privacy Preserving, National Joint Engineering Research Center of Network Security Detection and Protection Technology.

REFERENCES

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," IEEE Signal processing magazine, vol. 29, no. 6, pp. 82–97, 2012.

[2] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "Pcanet: A simple deep learning baseline for image classification?" IEEE Transactions on Image Processing, vol. 24, no. 12, pp. 5017–5032, 2015.

[3] E. Gawehn, J. A. Hiss, and G. Schneider, "Deep learning in drug discovery," Molecular informatics, vol. 35, no. 1, pp. 3–14, 2016.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," nature, vol. 521, no. 7553, p. 436, 2015.

[5] P. Danaee, R. Ghaeini, and D. A. Hendrix, "A deep learning approach for cancer detection and relevant gene identification," in PACIFIC SYMPOSIUM ON BIOCOMPUTING 2017. World Scientific, 2017, pp. 219–229.

[6] S. Gupta, W. Zhang, and F. Wang, "Model accuracy and runtime tradeoff in distributed deep learning: A systematic study," in Data Mining (ICDM), 2016 IEEE 16th International Conference on. IEEE, 2016, pp. 171–180.

[7] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman, "Project adam: building an efficient and scalable deep learning training system," in Usenix Conference on Operating Systems Design and Implementation, 2016, pp. 571–582.

[8] T. Chen and S. Zhong, "Privacy-preserving backpropagation neural network learning," IEEE Transactions on Neural Networks, vol. 20, no. 10, p. 1554, 2009.

[9] A. Bansal, T. Chen, and S. Zhong, "Privacy preserving back-propagation neural network learning over arbitrarily partitioned data," Neural Computing Applications, vol. 20, no. 1, pp. 143–150, 2011.

[10] J. Yuan and S. Yu, "Privacy preserving back-propagation learning made practical with cloud computing," IEEE Transactions on Parallel Distributed Systems, vol. 25, no. 1, pp. 212–221, 2014.

[11] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in Allerton Conference on Communication, Control, and Computing, 2015, pp. 909–910.

[12] P. Li, J. Li, Z. Huang, C. Z. Gao, W. B. Chen, and K. Chen, "Privacy-preserving outsourced classification in cloud computing," Cluster Computing, no. 1, pp. 1–10, 2017.

[13] Q. Zhang, L. Yang, and Z. Chen, "Privacy preserving deep computation model on cloud for big data feature learning," IEEE Transactions on Computers, vol. 65, no. 5, pp. 1351–1362, 2016.

[14] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2017, pp. 1175–1191.

[15] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in Security and Privacy (SP), 2017 IEEE Symposium on. IEEE, 2017, pp. 19–38.

[16] Y. Aono, T. Hayashi, L. Wang, S. Moriai et al., "Privacy-preserving deep learning via additively homomorphic encryption," IEEE Transactions on Information Forensics and Security, vol. 13, no. 5, pp. 1333–1345, 2018.

[17] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2017, pp. 587–601.

[18] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Inference attacks against collaborative learning," arXiv preprint arXiv:1805.04049, 2018.

[19] B. Hitaj, G. Ateniese, and F. Pérez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2017, pp. 603–618.

[20] T. Orekondy, S. J. Oh, B. Schiele, and M. Fritz, "Understanding and controlling user linkability in decentralized learning," arXiv preprint arXiv:1805.05838, 2018.

[21] A. Pyrgelis, C. Troncoso, and E. De Cristofaro, "Knock knock, who's there? membership inference on aggregate location data," arXiv preprint arXiv:1708.06145, 2017.

[22] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," arXiv preprint arXiv:1807.00459, 2018.

[23] "Health insurance portability and accountability act," https://www.hhs.gov/hipaa/index.html.

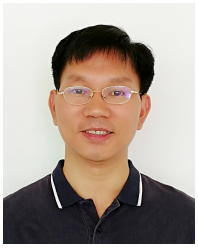
[24] G. Heigold, V. Vanhoucke, A. Senior, P. Nguyen, M. Ranzato, M. Devin, and J. Dean, "Multilingual acoustic models using distributed deep neural networks," in Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE, 2013, pp. 8619–8623.

[25] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[26] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments

from bitcoin," in Security and Privacy (SP), 2014 IEEE Symposium on. IEEE, 2014, pp. 459–474.
[27] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in Security and Privacy (SP), 2013 IEEE Symposium on. IEEE, 2013, pp. 397–411.
[28] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in Security and Privacy (SP), 2016 IEEE Symposium on. IEEE, 2016, pp. 839–858.
[29] S. Haykin, Neural networks: a comprehensive foundation. Prentice Hall PTR, 1994.
[30] H. Cui, G. R. Ganger, and P. B. Gibbons, "Scalable deep learning on distributed gpus with a gpu-specialized parameter server," pp. 1–16, 2016.
[31] H. Ma, F. Mao, and G. W. Taylor, "Theano-mpi: A theano-based distributed training framework," CoRR, pp. 800–813, 2016.
[32] Poseidon: An Efficient Communication Architecture for Distributed Deep Learning on GPU Clusters.
[33] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin, "Distributed deep learning models for wireless signal classification with low-cost spectrum sensors," CoRR, vol. abs/1707.08908, 2017.
[34] Distributed deep learning on edge-devices: Feasibility via adaptive compression, 2017.
[35] J. Dean, G. Corrado, Monga et al., "Large scale distributed deep networks," in Advances in neural information processing systems, 2012, pp. 1223–1231.
[36] N. Vasilache, J. Johnson, M. Mathieu, S. Chintala, S. Piantino, and Y. LeCun, "Fast convolutional nets with fbfft: A gpu performance evaluation," arXiv preprint arXiv:1412.7580, 2014.
[37] R. Wu, S. Yan, Y. Shan, Q. Dang, and G. Sun, "Deep image: Scaling up image recognition," arXiv preprint arXiv:1501.02876, vol. 7, no. 8, 2015.
[38] M. Lin, S. Li, X. Luo, and S. Yan, "Purine: A bi-graph based deep learning framework," arXiv preprint arXiv:1412.6249, 2014.
[39] L. Chen, P. Koutris, and A. Kumar, "Model-based pricing for machine learning in a data marketplace," arXiv preprint arXiv:1805.11450, 2018.
[40] A. B. Kurtulmus and K. Daniel, "Trustless machine learning contracts; evaluating and exchanging machine learning models on the ethereum blockchain," arXiv preprint arXiv:1802.10185, 2018.
[41] S. Micali, "Algorand: The efficient and democratic ledger," arXiv preprint arXiv:1607.01341, 2016.
[42] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in Proceedings of the 26th Symposium on Operating Systems Principles. ACM, 2017, pp. 51–68.
[43] M. Belenkiy, M. Chase, C. C. Erway, J. Jannotti, A. Küpçü, and A. Lysyanskaya, "Incentivizing outsourced computation," in Proceedings of the 3rd international workshop on Economics of networked systems. ACM, 2008, pp. 85–90.
[44] J.-S. Weng, J. Weng, 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, https://eprint.iacr.org/2018/679.
[45] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution." HASP@ ISCA, vol. 10, 2013.
[46] P.-A. Fouque, G. Poupard, and J. Stern, "Sharing decryption in the context of voting or lotteries," in International Conference on Financial Cryptography. Springer, 2000, pp. 90–104.
[47] T. Nishide and K. Sakurai, "Distributed paillier cryptosystem without trusted dealer," in International Workshop on Information Security Applications. Springer, 2010, pp. 44–60.
[48] I. Bentov and R. Kumaresan, "How to use bitcoin to design fair protocols," in International Cryptology Conference. Springer, 2014, pp. 421–439.
[49] R. Kumaresan and I. Bentov, "How to use bitcoin to incentivize correct computations," in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014, pp. 30–41.
[50] M. Ben-Or and A. Hassidim, "Fast quantum byzantine agreement," in Proceedings of the thirty-seventh annual ACM symposium on Theory of computing. ACM, 2005, pp. 481–485.
[51] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," in Proceedings of the sixth annual ACM Symposium on Principles of distributed computing. ACM, 1987, pp. 1–12.
[52] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains," Ph.D. dissertation, 2016.
[53] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in International Conference on the Theory and Applications of Cryptographic Techniques. Springer, 1999, pp. 223–238.
[54] B. Schoenmakers and M. Veeningen, "Universally verifiable multiparty computation from threshold homomorphic cryptosystems," in International Conference on Applied Cryptography and Network Security. Springer, 2015, pp. 3–22.
[55] "Corda: an open source distributed ledger platform," https://docs.corda.net/.
[56] W. Gavin, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project Yellow Paper, vol. 151, 2014.
[57] C. J. B. Yann LeCun, Corinna Cortes, "The mnist database of handwritten digits," http://yann.lecun.com/exdb/mnist/.

Jia-Si Weng received the B.S. degree in software engineering from South China Agriculture University in June 2016. Currently, she is a Ph.D. student with School of Information Science and Technology in Jinan University. Her research interests include applied cryptography, Blockchain, network security, etc.



Jian Weng received B.S. and M.S. degrees in computer science from South China University of Technology, in 2000 and 2004, respectively, and Ph.D. degree in computer science from Shanghai Jiao Tong University, in 2008. From April 2008 to March 2010, he was a postdoc with School of Information Systems in Singapore Management University. Currently, he is a professor and executive dean with School of Information Science and Technology in Jinan University. He has published more than 100 papers in international conferences and journals, such as CRYPTO, EUROCRYPT, ASIACRYPT, TCC, PKC, CT-RSA, IEEE TPAMI, IEEE TDSC, IEEE TIFS, etc. He served as PC co-chairs or PC members for more than 30 international conferences. He is currently on the editor board of IEEE Transactions on Vehicular Technology. He has won the 2014 cryptographic innovation award from Chinese Association for Cryptographic Research.

Yue Zhang received his M.S. degree and B.S. degree from Xi'an University of Posts & Telecommunications in 2014 and 2016, respectively. Since September 2016, he is a Ph.D. student with School of Information Science and Technology in Jinan University. His research interests include Blockchain, system security, android security, etc.

Weiqi Luo received his B.S. degree and M.S. degree from Jinan University in 1982 and 1985 respectively, and Ph.D. degree from South China University of Technology in 1999. Currently, he is a professor with School of Information Science and Technology in Jinan University. His research interests include network security, big data, artificial intelligence, etc. He has published more than 100 high-quality papers in international journals and conferences.