Towards Fair and Privacy-Preserving Federated Deep Models

Lingjuan Lyu, *Member, IEEE*, Jiangshan Yu, Karthik Nandakumar, *Senior Member, IEEE*, Yitong Li, Xingjun Ma, Jiong Jin, *Member, IEEE*, Han Yu, and Kee Siong Ng

Abstract—The current standalone deep learning framework tends to result in overfitting and low utility. This problem can be addressed by either a centralized framework that deploys a central server to train a global model on the joint data from all parties, or a distributed framework that leverages a parameter server to aggregate local model updates. Server-based solutions are prone to the problem of a single-point-of-failure. In this respect, collaborative learning frameworks, such as federated learning (FL), are more robust. Existing federated learning frameworks overlook an important aspect of participation: fairness. All parties are given the same final model without regard to their contributions. To address these issues, we propose a decentralized Fair and Privacy-Preserving Deep Learning (FPPDL) framework to incorporate fairness into federated deep learning models. In particular, we design a local credibility mutual evaluation mechanism to guarantee fairness, and a three-layer onion-style encryption scheme to guarantee both accuracy and privacy. Different from existing FL paradigm, under FPPDL, each participant receives a different version of the FL model with performance commensurate with his contributions. Experiments on benchmark datasets demonstrate that FPPDL balances fairness, privacy and accuracy. It enables federated learning ecosystems to detect and isolate low-contribution parties, thereby promoting responsible participation.

Index Terms—Federated learning, privacy-preserving, deep learning, fairness, encryption

1 Introduction

Deep learning has become an important technology to deal with challenging real-world problems such as image classification and speech recognition. Empirical evidence has shown that deep learning models can benefit significantly from large-scale datasets [1]. However, large-scale datasets are not always available for a new domain, due to the significant time and effort required for data collection and annotation [2], [3]. Moreover, training complex deep networks on large-scale datasets is computationally expensive and may not be feasible for a single party in practice. Therefore, there is a high demand to perform deep learning in a collaborative manner among a group of parties.

This trend is motivated by the fact that the data owned by a single party may be very homogeneous, resulting in overfitting which negatively impacts accuracy when the model is applied to previously unseen data, i.e., poor generalizability. Utilizing data from diverse parties to train deep models can help mitigate this problem. However, collaborative model training may not be viable due to privacy concerns. Federated learning (FL), which incorporates privacy preservation techniques into collaborative model training, offers a potential solution to this challenge [4].

In the current federated learning paradigm [5], all participants receive the same federated model at the end of collaborative model training regardless of their contributions. This makes the paradigm vulnerable to free-riding participants. For example, several banks may want to work together to build model to predict the creditworthiness of small and medium enterprises. However, but larger banks with more data maybe reluctant to train their local model based on high quality local data for fear of smaller banks benefiting from the shared FL model and eroding its market share [4]. Without the guarantee of privacy and the promise of collaborative fairness, participants with high quality and large datasets may be discouraged from joining federated learning, thereby negatively affect the formation of a healthy FL ecosystem. Existing research on fairness mostly focuses on protecting sensitive attributes or reducing the variance of the prediction distribution across participants [6], [7]. The problem of treating federated learning participants fairly remains open [4].

In this paper, we address the problem of treating FL participants fairly based on their contributions to build a healthy FL ecosystem. We refer to the proposed framework as the decentralized Fair and Privacy-Preserving Deep Learning (FPPDL) framework. Unlike existing work such as [8] which uses monetary rewards to incentivize good behaviour, our

- *L. Lyu is with The Department of Computer Science, National University of Singapore, Singapore 119077. E-mail: lyulj@comp.nus.edu.sg.*
- *J. Yu is with the Faculty of Information Technology, Monash University, Clayton, VIC 3800, Australia. E-mail: jiangshan.yu@monash.edu.*
- *K. Nandakumar is with IBM Singapore Lab, Singapore, 018983. E-mail: nkarthik@sg.ibm.com.*
- *Y. Li and X. Ma are with the School of Computing and Information Systems, The University of Melbourne, Melbourne 3010, Australia. E-mail: yitongl4@student.unimelb.edu.au, xingjun.ma@unimelb.edu.au.*
- *J. Jin is with the School of Software and Electrical Engineering, Swinburne University of Technology, Hawthorn, VIC 3122, Australia. E-mail: jiongjin@swin.edu.au.*
- *H. Yu is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore, 639798. E-mail: han.yu@ntu.edu.sg.*
- *K.S. Ng is with the Software Innovation Institute, Australian National University, Canberra, ACT 0200, Australia. E-mail: keesiong.ng@anu.edu.au.*

Manuscript received 3 Oct. 2019; revised 1 May 2020; accepted 16 May 2020. Date of publication 21 May 2020; date of current version 1 June 2020. (Corresponding author: Lingjuan Lyu and Han Yu.) Recommended for acceptance by S. Chen. Digital Object Identifier no. 10.1109/TPDS.2020.2996273

TABLE 1
Comparing Different Deep Learning Frameworks

Frameworks	Standalone	Centralized [18], [19]	Distributed [5], [10], [11], [20]	Decentralized [12], [13], [14], [15], [16], [17]	Decentralized (our FPPDL)
Architecture	Fig. 1a	Fig. 1b	Fig. 1c	Fig. 1d	Fig. 1d
Global model	No	Yes	Yes	Depends	Depends
Local models	Yes	No	Yes	Yes	Yes
Collaborative Fairness	NA	NA	No	No	Yes
Quality control	NA	No	No	No	Yes

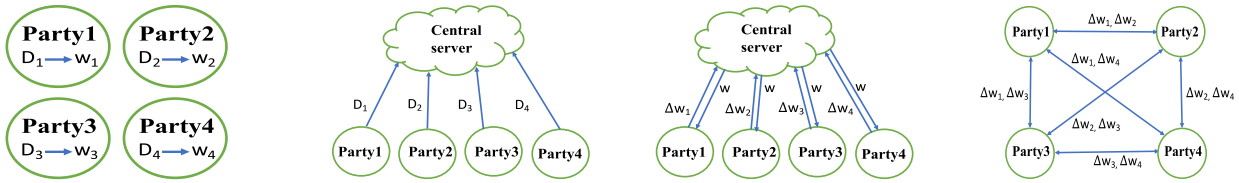


Fig. 1. (a): Standalone framework. (b): Centralized framework. (c): Distributed framework. (d): Decentralized framework.

proposed solution fundamentally changes the current FL paradigm so that participants may not receive the same FL model in the end. Instead, each of them will receive a final FL model with performance reflecting their individual contributions to the federation. FPPDL does not require participants to trust each other or any third party. It records all operations, including uploading and downloading *differentially private artificial samples* and *encrypted model updates*, as transactions through blockchain technology. Through mutual evaluations of local credibility that considers the relative contribution of each party during both initial benchmarking and privacy-preserving collaborative deep learning, FPPDL achieves collaborative fairness. For privacy preservation, instead of leveraging differential privacy at the cost of utility, we propose a three-layer onion-style encryption scheme to guarantee accuracy and privacy.

To the best of our knowledge, this paper is the first to achieve collaborative fairness in federated learning through adjusting the level of performance of the version of the FL model allocated to each participant based on his contribution. Extensive experiments based on two benchmark datasets under three realistic settings demonstrate that FPPDL achieves high fairness, delivers comparable accuracy to existing centralized and distributed deep learning frameworks, and outperforms standalone deep learning.

In terms of the threat model, FPPDL adopts an honest-but-curious setting: each party is assumed to be curious in inferring sensitive information of other parties; and yet, it is assumed to be honest in operations. This setting is reasonable as the main incentive for parties to participant in the collaborative system is to get better local models compared to their standalone models without any collaboration. Moreover, in our scenario parties are considered as organisations such as financial or biomedical institutions acting with responsibilities by laws. However, we also discuss how our local credibility mutual evaluation mechanism can help prevent certain malicious behaviours of the insider attacker, and resist the outsider attacker in Section 7.

The rest of this paper is organized as follows. Section 2 reviews the existing deep learning frameworks, and the related literatures on privacy-preserving collaborative deep learning, and fairness in federated learning which are major problems we aim to tackle. Section 5 presents technical details of the proposed FPPDL framework. Section 6 evaluates the performance of FPPDL in terms of accuracy and fairness for different SGD frameworks under different settings, followed by discussions in Section 7. Section 8 concludes the paper and points out potential future research directions.

2 Related Work

In this section, we review relevant literature on deep learning frameworks, privacy preservation and fairness in federated learning to position our research in relation to existing research.

2.1 Overview of Deep Learning Frameworks

In general, deep learning frameworks fall into the following categories: *Standalone framework*; server-based frameworks including *Centralized framework* and *Distributed framework*; and *Decentralized framework*. In particular, in distributed framework and decentralized framework, parties are all involved in the global or consensus model improvement process. Hence, we refer to them as collaborative deep learning frameworks. A comparison among different deep learning frameworks is provided in Table 1.

Standalone Framework. Parties individually train standalone models on their local training data without any collaboration (Fig. 1a). However, standalone models might fail to generalise to the unseen data.

Centralized Framework. Participants pool their data into a centralized server to train a global model (Fig. 1b). This centralized framework is very effective, but it violates data privacy as all participants' data are exposed to the server.

Distributed Framework. Dean *et al.* [9] first introduced the concept of distributed deep learning, where parties collaboratively train a model by sharing local model updates with a parameter server. Distributed learning had been extensively studied in [5], [10], [11].

It should be noted that both the centralized framework and the distributed framework require a central server to mediate the training process, which makes them susceptible to the following issues: (1) Party policies: due to privacy concern, parties may not want to cede control to an untrusted server; (2) Single-point-of-failure: if the central server fails or is shut down for maintenance, the whole network stops working.

Decentralized Framework. the above issues in the central server-based frameworks can be addressed by a decentralized framework [12], [13], [14], [15], [16], [17], which parallelizes the computation among all parties (Fig. 1d). In particular, Kuo *et al.* [12] first proposed a decentralized machine learning framework: ModelChain, which integrates Blockchain with privacy-preserving machine learning. [15], [16] investigated privacy-preserving deep learning on blockchain. [17] studied the problem of fairness in load sharing in blockchain-based privacy-preserving learning, which is different from the collaborative fairness in our work. Specifically, their decentralized architecture is developed under a less secure setting, where one site can access the models of all the other sites. [13], [14] utilized differential privacy for privacy-preserving machine learning on blockchain. However, [14] had pointed out that the proposed algorithms in [13] cannot guarantee privacy-preserving properties correctly as they did not consider composition for a repeated additive-noise mechanism.

In summary, existing collaborative frameworks (distributed or decentralized) focus on how to learn a global consensus model with higher accuracy than individual standalone models. In reality, some parties may contribute more compared with other parties, while others may contribute nearly nothing or even negatively. The reason is that data owned by different parties may be of different quality, and there may exist unpredictable random errors during data collection and storage. On the other hand, parties may choose only to use a limited part of its data for collaborative model training.

2.2 Privacy Preserving Collaborative Learning

As pointed out by Shokri *et al.* [11], centralized deep learning commonly comes with many privacy concerns. Specifically, all the sensitive training data are revealed to a third party; data owners have no control over the learning objective; the learned model is not directly available to data owners. To mitigate these privacy risks, Gilad-Bachrach *et al.* [18] developed CryptoNets to run deep learning on homomorphically encrypted data. However, CryptoNets assumes that neural network model has been trained beforehand, hence their system is mainly used to provide encrypted outputs to users. In contrast, SecureML [20] conducts privacy-preserving learning via secure multiparty computation (SMC), where data owners need to process, encrypt and/or secret-share their data among two non-colluding servers in the initial setup phase. SecureML allows data owners to train various models on their joint data without revealing any information beyond the outcome. However, such an approach incurs high computational and communication costs [21], [22].

The most relevant work is Distributed Selective Stochastic Gradient Descent (DSSGD) [11]. To preserve privacy, instead of explicitly sharing training data, each party computes and shares (with the PS) its local model gradients based on local training data, while updating its local model by downloading the most-updated parameters from the PS. To further mitigate privacy leakage from the shared model updates, each party adds noise to local model updates to ensure per-parameter differential privacy. However, their system requires a central parameter server to mediate training process. Therefore, it suffers from the common issues in the central server-based frameworks.

The disadvantages of having a centralized parameter server can be summarized as follows:

1) Privacy leakage: As evidenced in [23], local data information may be leaked to an honest-but-curious PS, even if only a small portion of local model updates is released to the PS. In particular, a PS can infer the true data or label of the participates with non-negligible probability for the local neural network with only one neuron. The above observations similarly hold for general neural networks, with both cross-entropy and squared-error cost functions. Even for general neural networks with regularization, the released local gradients can still reveal the truth value.
2) Vulnerability to active adversaries: Most distributed learning frameworks assume that all the parties are honest. In reality, if a party turns out to be malicious, it can sabotage the learning process by spoofing random samples to infer information about the victim party's private data.

A special case of distributed deep learning is federated learning [4]. In FL, to preserve privacy of individual model updates, Bonawitz *et al.* [24] proposed a practical secure aggregation protocol, which is proven to be secure under the honest-but-curious and active adversary settings, even if an arbitrarily chosen subset of users drop out at any time. In particular, secure multi-party computation is leveraged to compute sums of model parameter updates from individual users' devices in a secure manner, which comes at the cost of extra computation and communication overheads. Another more efficient method is to use differential privacy by enabling the server to add the tailored noise to the weighted-average user updates to guarantee user-level privacy [25]. However, the default trusted Google server is entitled to see all users' update clearly, aggregate individual updates and add noise to the aggregation. Thus, their method is not preferred when the server is not a trusted party. We also remark that when the server is untrusted, the weighted aggregation becomes unrealistic, as the server may not know the data size of each party for weight calculation. Instead, the proposed FPPDL allows each party to integrate other parties' updates based on their local credibility and sharing level.

2.3 Fairness in Federated Learning

Existing approach for promoting collaborative fairness among federated learning participants is based on incentive schemes. In general, participants shall receive payoffs that is commensurate with their contributions. Equal division is an example of egalitarian profit-sharing [26]. Under this scheme, the available total payoff at a given round is equally divided among all participants. Under the Individual profit-sharing

scheme [26], each participant i's own contribution to the collective (assuming the collective only contains i) is used to determine his share of the total payoff.

The Labour Union game [27] profit-sharing scheme determines a participant's share of the total payoff based on his marginal contribution to the utility of the collective formed by his predecessors (i.e., each participant's marginal contribution is computed based on the same sequence as they joined the federation). The Fair-value game scheme [27] is a marginal loss-based scheme. Under this scheme, a participant's share of the total payoff is determined by the sequence following which the participants leave a federation. The Shapley game profit-sharing scheme [27] is also a marginal contribution-based scheme. Unlike the Labour Union game, Shapley game aims to eliminate the effect of the participants joining the collective in different sequences in order to more fairly estimate their marginal contributions to the collective. Thus, it averages the marginal contribution for each participant under all different permutations of him joining the collective relative to other participants. This approach is computationally expensive.

For gradient-based federated learning approaches, the gradient information can be regarded as a type of data. However, in these cases, output agreement-based rewards are hard to apply as mutual information requires a multi-task setting which is usually not present in such cases. Thus, among these three categories of schemes, model improvement is the most relevant way of designing rewards for federated learning. There are two emerging federated learning incentive schemes focused on model improvement.

A scheme which pays for marginal improvements brought about by model updates was proposed in [28]. The sum of improvements might result in overestimation of contribution. Thus, the proposed approach also includes a model for correcting the overestimation issue. This scheme ensures that payment is proportional to model quality improvement, which means the budget for achieving a target model quality level is predictable. It also ensures that data owners who submit model updates early receive a higher reward. This motivates them to participate even in early stages of the federated model training process.

In addition to the contributions made by participants, [8] proposed a joint objective optimization-based approach to take costs and waiting time into account in order to achieve additional notions of fairness when distributing payoffs to FL participants. Different from the aforementioned approaches, the proposed FPPDL framework does not utilize monetary payoffs to achieve fair treatment of FL participants. Instead, it allocates each of them a different version of the FL model with performance commensurate with his contributions. This represents a alternative paradigm to existing federated learning in which all participants receive the same final FL model.

3 Preliminaries

In this section, we introduce key technologies which form the building blocks of the proposed FPPDL framework, including differential privacy, homomorphic encryption, and blockchain.

3.1 Differential Privacy

Differential privacy 1 [29] trades off privacy and accuracy by perturbing the data in a way that is (i) computationally efficient, (ii) does not allow an attacker to recover the original data, and (iii) does not severely affect utility.

Definition 1. *A randomized mechanism $\mathcal{M}$: $\mathcal{D} \rightarrow \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-differential privacy if for all two neighbouring inputs $D, D' \in \mathcal{D}$ that differ in one record and for any measurable subset of outputs $S \subseteq \mathcal{R}$ it holds that*

$$\Pr\{\mathcal{M}(D) \in S\} \leq \exp(\epsilon) \cdot \Pr\{\mathcal{M}(D') \in S\} + \delta.$$

Furthermore $\mathcal{M}$ is said to preserve (pure) ϵ-differential privacy if $\delta = 0$.

The formal definition of differential privacy has two parameters: privacy budget ϵ measures the privacy leakage; and δ bounds the probability that the privacy loss exceeds ϵ. The values of (ϵ, δ) are accumulated as the algorithm repeatedly accesses the private data [30].

3.2 Homomorphic Encryption

Homomorphic encryption is a form of encryption that is widely used to derive the aggregate in a secure manner. Existing homomorphic encryption techniques include fully homomorphic encryption, somewhat homomorphic encryption and partially homomorphic encryption. Fully homomorphic encryption can support arbitrary computation on ciphertexts, but is less efficient [31]. On the other hand, somewhat homomorphic encryption and partially homomorphic encryption only support a limited number of operations [32].

However, all these techniques generally result in longer ciphertext than the plaintext, incurring extra communication costs. To address this issue in this paper, we take inspirations from stream ciphers [33] to develop efficient homomorphic-ciphertext compression, which also allows additive homomorphic operation over ciphertexts encrypted under different parties' key streams. More details are provided in Section 5.2.1.

3.3 Blockchain

Blockchain is a decentralized (i.e., a peer-to-peer, non-intermediated) system that is maintained by all the participants in the system. There are two types of blockchains, namely permissionless blockchain and permissioned blockchain. With permissionless blockchains, such as Bitcoin [34], participants can join and leave at anytime and the number of participants is not pre-defined nor fixed. With permissioned blockchains (a.k.a. consortium blockchains), such as IBM's Hyperledger Fabric, participants require permissions from the system to join or leave. The set of participants are normally predefined [35].

For the application with a relatively stable set of participants, a permissioned blockchain is preferred. It can serve as a distributed key-value store, where a fault tolerance (a.k.a. Byzantine agreement) scheme is required for reaching consensus on the global state. Blockchain is well known for its transparency, accountability and robustness–data and all operations are recorded on the blockchain in an append-only manner and are accessible by all the participants. Intuitively, the incremental characteristic of federated deep learning

TABLE 2
List of Symbols

Symbol	Meaning
D_i, M_i	local training data and local model of party i
SD_i	μ DPGAN samples randomly chosen by party i
p_i, d_i	points and gradients download budget of party i
c_i^j, $c_i^{j'}$	local credibility and updated local credibility of party j given by party i
u_i	number of DPGAN samples released by party i
d_i^j	number of meaningful gradients of party j released to party i
λ_j	sharing level of party j
$sacc_j, acc_j$	standalone and final model accuracy of party j
$\Delta \boldsymbol{w}_j$	gradient vector of party j
$\Delta \tilde{\boldsymbol{w}}_j^i$	masked gradient vector of party j shared with party i by filling the remaining $\lvert\Delta \boldsymbol{w}_{j^i}\rvert - d_i^j$ gradients with 0
$\boldsymbol{w}_i$	parameter of party i at previous round
$\boldsymbol{w}_i'$	updated parameter of party i at current round
n	number of participating parties
c_{th}	lower bound of the credibility threshold
C	credible party set with local credibility above c_{th} agreed by $2/3$ parties
m_j	number of matches between majority labels and party j's predicted labels
(sk_i', pk_i')	party i's key pair for signing and verification, respectively
k_i	party i's keystream used in the first layer of three-layer onion-style encryption
fsk	fresh symmetric encryption key used in the second layer of three-layer onion-style encryption
(sk_i, pk_i)	party i's key pair for decryption and encryption in the third layer of three-layer onion-style encryption
Enc	homomorphic encryption
$Senc$	symmetric key encryption
$Aenc$	public key encryption
E	number of local training epochs in each round
B, lr	local batch size, local learning rate

makes it suitable for leveraging Blockchain. However, a reasonable approach to integrate Blockchain with privacy-preserving deep learning needs to be developed.

4 The FPPDL Framework

This section describes the design of our proposed decentralized Fair and Privacy-Preserving Deep Learning framework, and an investigation of Blockchain as the decentralized architecture for FPPDL. Table 2 presents the list of symbols used in this paper and their meanings for easy readability.

4.1 Design Objectives

4.1.1 Privacy Preservation

In FPPDL, we assume parties do not trust each other or any third party. Hence, parties may not be willing to share their information when training a joint model without the promise of privacy protection. Under FPPDL, instead of sharing the original data or model parameters, each party leverages Differentially Private GAN (DPGAN) to publish differentially private local samples for mutual evaluation during the initial benchmarking phase. Then, they encrypt the shared gradients using the proposed three-layer onion-style encryption scheme to preserve privacy during collaborative deep model training.

4.1.2 Fairness

Since our focus here is to distribute different variants of the final FL model to participants based on their contributions, the notion of fairness most relevant for our purpose is *Fairness through Awareness*. Under this notion, individuals who are similar with respect to a similarity metric defined for a particular task should receive a similar outcome [36]. A party with high-contribution should be rewarded more than a party with low-contribution party. Moreover, we clarify that the low-contribution parties are not malicious, i.e., they follow the protocol honestly and aim to benefit from other parties' data, but contribute lowly, or even nearly nothing or negatively. Under the collaborative model training scenario, we define collaborative fairness as:

Definition 2. *Collaborative fairness. In collaborative learning systems, a high-contribution party is deserved to be rewarded with a better performing local model than a low-contribution party. Specially, in IID setting, fairness can be quantified by the correlation coefficient between the contributions by different parties and their respective final model accuracies.*

Considering these two goals, we design a local credibility mutual evaluation mechanism to enforce fairness in FPPDL, where participants trade their information in an "earn-and-pay" way using their "points". The local credibility and points of each participant are initialized through an initial benchmarking phase, and updated through privacy-preserving collaborative deep model training.

The basic idea is that participants can earn points by contributing their information to other participants. Then, they can use the earned points to trade information with other participants. Thus, participants are encouraged to upload more samples or gradients to earn more points (as long as it is within the limit of their sharing levels), and use these points to download more gradients from others. All trades are recorded as immutable transactions in a blockchain, providing transparency and auditability. In particular, FPPDL ensures fairness during download and upload processes as follows:

- *Download*: Since one party might contribute differently to different parties, the credibility of this party might be different from the view of different parties. Therefore, each party i records a private local credibility list for all parties sorted in descending order of their credibility values. The higher the credibility of party j in party i's credibility list, the more likely party i will download gradients from party j, and consequently, more points will be rewarded to party j by party i.
- *Upload*: Once a party receives download request for its local gradients, it can determine how many meaningful gradients to send back based on both the download request from the requester and its own sharing level.

4.2 Blockchain-Based Architecture

To develop a decentralized architecture for FPPDL, we incorporate the privacy-preserving deep learning algorithm into a private Blockchain using Blockchain 2.0, which is only available to the participating parties. Compared with

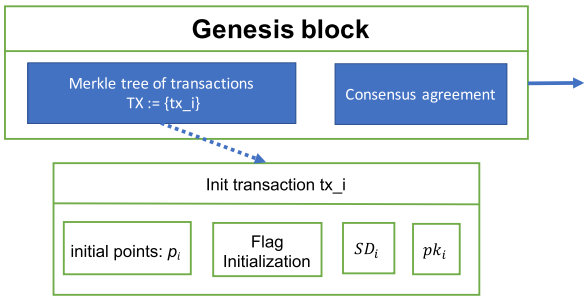


Fig. 2. *An example structure of the genesis block*. It mainly contains two key components, one is a set of init transactions organized as leaves of a Merkle tree; and the other one is the consensus agreement reached by the participants through the underlying consensus protocol (e.g., PBFT or PoS), which is specific to the deployed Blockchain. The pk_i' in the init transaction is a signature verification key of party i.

the current server-based architecture, FPPDL inherits the peer-to-peer architecture of Blockchain, allowing each party to remain modular while interoperating with others. In addition, instead of ceding control to the central server, each party keeps full control of its own data. Moreover, Blockchain provides the native ability to automatically coordinate the joining and departure of each party, further facilitating independence and modularity of the federation. Blockchains, with no single point of failure, also enhances robustness. Here, we design two types of blocks in the Blockchain for FPPDL, namely the *init block* and the *operation block*.

Algorithm 1. Initial Benchmarking

Input: number of participating parties n**, C={1,...,n}**
Output: local credibility and points of all parties
1: **Pre-train aprior models**: Each party i trains standalone model M_i and local DPGAN based on its local training data.
2: **Sharing level initialization**: During initialization, party i randomly selects and releases u_i artificial samples generated by local DPGAN to any party j, sharing level is autonomously determined as $\lambda_i = u_i/|D_i|$, where $|D_i|$ is local training data size of party i.
3: **Local credibility initialization**: Party j labels the received artificial samples by its local model M_j, then returns the predicted labels back to party i. Meanwhile, party i also labels its own DPGAN samples using M_i. Afterwards, party i applies majority voting to all the predicted labels, then initializes the local credibility of party j as $c_i^j = \frac{m_j}{u_i}$, where m_j is the number of matches between majority labels and party j's predicted labels, and u_i is the number of DPGAN samples released by party i. The detailed explanation is elaborated in Section 5.1.1.
4: **Local credibility normalization**: $c_i^j = \frac{c_i^j}{\sum_{j \in C \setminus i} c_i^j}$
if $c_i^j < c_{th}$ **then**
party i reports party j as a low-contribution party
end if
5: **Credible party set**: If the majority of parties report party j as low-contribution, Blockchain removes party j from the credible party set C and all parties run step 4 again.
6: **Points initialization to download gradients**:
$p_i = \lambda_i * |\boldsymbol{w}_i| * (n-1)$.

An init block initializes benchmarking of the usefulness of each party's training data, as a set of init transactions. An init transaction contains the initial points that the transaction creator earned, its contributed DPGAN samples, and its public key that will be used for authenticating future transactions. The genesis block (i.e., the first block) of the Blockchain is an init block, which contains the initial points and local credibility values to all the participants according to their relative contributions, as stated in Algorithm 1. If any party joins or adds new data during later updates, a new init block will be created and added to the existing Blockchain.

An operation block contains a set of transactions defining the UPLOAD operation and/or DOWNLOAD operation. All UPLOAD and DOWNLOAD transactions are signed by their creator using the private key associated with the public key recorded in the init transaction. An UPLOAD operation commits that a data owner has uploaded local model gradients to the party who sent a download request. A DOWNLOAD operation states that a participant is committed an order to request some local model updates from other participants. Upon receiving a DOWNLOAD transaction, Blockchain miners verify its signature, check if the requester has enough balance points to download the number of requested gradients, and record the verified transactions in an operation block. Once the DOWNLOAD transaction is recorded in the Blockchain, the requested local model gradients will be encrypted and uploaded by the owner to a publicly accessible storage, and re-encrypted using the recipient's public key defined in the DOWNLOAD transaction.

The privacy of local model gradients is protected through a three-layer onion-style encryption scheme (see Section 5.2). The first layer encrypts the local model gradients through our proposed symmetric key based homomorphic encryption (Algorithm 3), which allows each party to learn the aggregate of the received gradients without revealing individual gradients, i.e., party obliviousness. The second and third layers present a standard hybrid encryption process: the second layer uses a freshly generated symmetric key fsk to re-encrypt the first layer ciphertext, and the third layer encrypts fsk with the requester party i's public key pk_i. In this way, we minimize the required computational cost incurred by the asymmetric key based encryption. The commitment of the uploaded encrypted local model gradients (e.g., hash value of the ciphertext, as presented in Fig. 3) will be included in the UPLOAD transaction.

In our private Blockchain, only the requester who pays can read the plaintext. Others can verify that this transaction has happened, but cannot read the plaintext. When a requester blames a data uploader, the data uploader reveals the plaintext as evidence. In this case, the requester will be forced to pay a fine that it deposits when filing a claim if it is shown to be an dishonest claim. Once an UPLOAD transaction is recorded in the Blockchain, the points will be automatically transferred from the requester to the uploader. An example of INITIALIZE and DOWNLOAD transaction stored by the Blockchain are shown in Figs. 2 and 3, respectively. For our application scenario, we expect a relatively stable and small set of participants, such as financial institutions acting with legal liabilities, which falls under the umbrella of horizontal federated learning (HFL) involving business participants [37]. This allows us to adopt a permissioned blockchain.

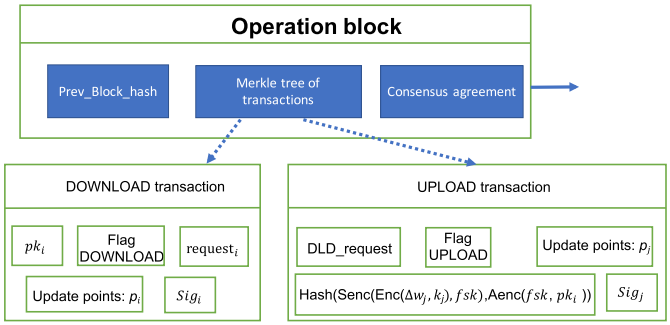


Fig. 3. *An example structure of the operation block*. It mainly contains three key components, namely, the hash value Prev_Block_hash of the previous block, a set of UPLOAD/DOWNLOAD transactions organized as a Merkle tree, and the consensus agreement of this block. In particular, a Prev_Block_hash links the current block to the previous one, and the request in the UPLOAD transaction acts as a reference to the associated DOWNLOAD transaction. pk_i in the DOWNLOAD transaction is the public key that will be used in the last layer of our three-layer onion-style encryption scheme, $request_i$ is a unique request ID of this transaction and will be referenced in the corresponding UPLOAD transaction via DLD_request, and Sig_i is the signature on this transaction. Enc, $Senc$, and $Aenc$ refer to homomorphic encryption, symmetric key encryption, and public key encryption, respectively.

5 Implementation of FPPDL

This section details the two-stage implementation of FPPDL to enforce both fairness and privacy. These include how to initialize local credibility values, sharing levels and points through initial benchmarking, and how to update local credibility values and points in the privacy-preserving collaborative deep learning phase, followed by the quantification of fairness. The two-stage implementation is shown in Fig. 4.

5.1 Initial Benchmarking

The proposed initial benchmarking algorithm aims to assess the quality of local training data of each participant via mutual evaluation without looking at the raw data before collaborative model training starts. The algorithm works as follows: each participant trains a DPGAN based on its local training data to generate artificial samples. However, these generated samples will not disclose the true sensitive examples, as well as the true distribution of data, but only a few implicit density estimation within a modest privacy budget used in DPGAN. Each participant publishes individually generated artificial samples based on its individual sharing level without releasing labels. All the other participants produce predictions for the received artificial samples using their pre-trained standalone models and send the predicted labels back to the party who generated these samples.

The aim of sharing artificial DPGAN samples is two-fold:

1) To obtain prior information about individual models before collaborative learning starts. If a participant does not have a reasonable amount of training data to produce a decent model, it will perform poorly during the initial evaluation phase. Therefore, other participants will be cautious when sharing gradients with it.
2) To obtain a rough estimate of data distribution of other participants. Two participants can mutually benefit only if their data distributions are different, but with some degree of overlap. Suppose that two participants A and B have published almost identical artificial samples, it means that their training data distributions are almost identical. In this case, the updates from B are unlikely to increase the accuracy

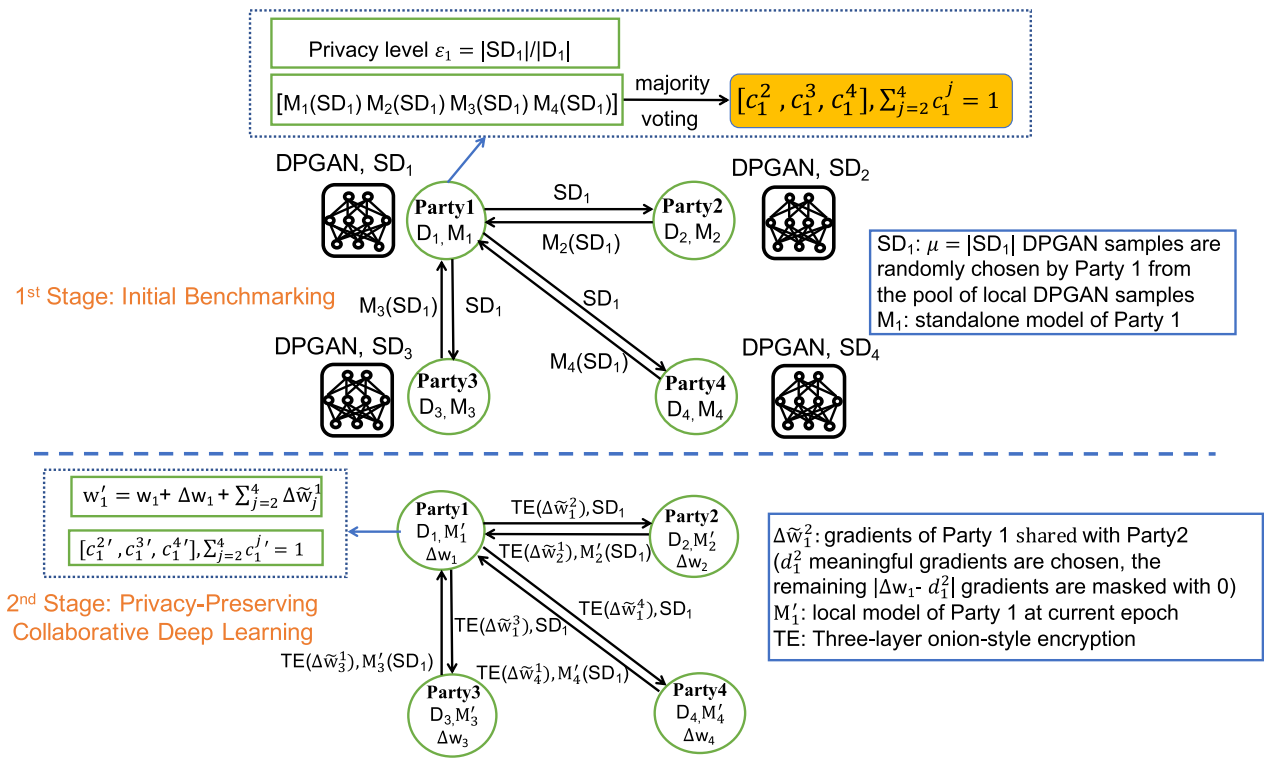


Fig. 4. Two-stage implementation of FPPDL.

of model A and vice versa. Therefore, during the subsequent communication rounds, A and B should avoid downloading updates from each other. Other participants can choose to download updates from either A or B but not both. On the other hand, suppose that participants A and B have completely different data distributions, the updates from B are unlikely to increase the accuracy of model A and vice versa. Thus, during the subsequent rounds, A and B should also avoid downloading updates from each other. Furthermore, suppose that A's data distribution is different from that of all the other participants, all these participants should try to avoid A. This automatically takes care of the scenario where a honest participant publishes some gradients, while all the other honest participants assign very low credibility to the publisher. In this case, the data distribution of the publisher is completely different from that of the other participants. Hence it is reasonable to reduce the credibility of the publisher.

We next describe the detailed procedures of initial benchmarking in Algorithm 1, including: local credibility initialization, and sharing level and points initialization.

5.1.1 Local Credibility Initialization

For local credibility initialization, each party compares the majority voting of all the combined labels with a particular party's predicted labels to evaluate the effect of this party. It relies on the fact that the majority voting of all the combined labels reflects the outcome of the majority of parties, while the predicted labels of party j only reflects the outcome of party j.

For example, in the case of party i initializing local credibility list for other parties, party i broadcasts its DPGAN samples to other parties, who label these samples using their pre-trained standalone models, and send the corresponding predicted labels back to party i. Meanwhile, party i also labels its own artificial samples using its pre-trained standalone model, then combines all parties' predicted labels as a label matrix with total n columns, where each column corresponds to one party's predicted labels. Party i then initializes the local credibility of party j as $c_i^j = \frac{m_j}{u_i}$, where m_j is the number of matches between the majority labels and party j's predicted labels, and u_i is the number of DPGAN samples released by party i. Afterwards, party i normalizes c_i^j within [0, 1].

If the majority of parties report that the local credibility of one party is lower than the threshold c_{th}, implying a potentially low-contribution party, it will be banned from the local credibility lists of all parties. Here, c_{th} is mainly used to detect and isolate the low-contribution party, and it should be agreed by the majority of parties. However, it should not be too small or too large as fairness and accuracy may be affected. If it is too small, it might allow low-contribution party to into the collaborative learning system without being detected and isolated. If it is too large, it might ban most participants from the system. In the following update process, party i is more likely to download gradients from more credible participants, while download less, even ignoring those published by less credible parties.

5.1.2 Sharing Level and Points Initialization

Sharing level is denoted by the the upper bound of the number of samples or gradients one party can share with others. Based on the number of artificial samples u_i that party i publishes at the beginning, a suitable sharing level of party i can be automatically estimated as $\lambda_i = u_i/|D_i|$, where D_i is the local training data of party i. Points are initialized as follows:

$$p_i = \lambda_i * |\boldsymbol{w}_i| * (n-1), \tag{1}$$

where λ_i is the sharing level of party i (i.e., the higher, the more data one party would like to share), $|\boldsymbol{w}_i|$ is the number of model parameters, and n is the number of parties. The points gained from initial benchmarking will be used to download gradients in the following collaborative learning process, and the number of gradients i can downloaded depends on both the local credibility and sharing level of the party from which it is requesting.

5.1.3 Differentially Private GAN (DPGAN)

During initial benchmarking, although each party only releases a small amount of unlabeled samples, it may still disclose privacy of local training data. The approach of generating samples under differential privacy with generative adversarial network (GAN) offers a solution to this problem. Under FPPDL, we train a *Differentially Private GAN* by adding tailored noise to the gradients during DPGAN learning [38] at each party.

In the context of a GAN, the discriminator is the only component that accesses the private real data. Therefore, we only need to train the discriminator under differential privacy. The differential privacy guarantee of the entire GAN directly follows because the computations of the generator are simply post-processing from the discriminator. The main idea follows the post-processing property of differential privacy [29], as stated in Lemma 1.

To counter the stability and scalability issues of training DPGAN models, we apply multi-fold optimization strategies, including weight clustering, adaptive clipping and warm starting, which significantly improve both training stability and utility [38]. Unlike PATE [39], where privacy loss is proportional to the amount of data needed to be labeled in public test data, differentially private generator can generate infinite number of samples for the intended analysis, while rigorously guaranteeing (ϵ, δ)-differential privacy of training data. Without loss of generality, we exemplify DPGAN in the context of the improved WGAN framework [40] and let each party generates a total of 1,000 artificial samples. As demonstrated in [38], DPGAN is able to synthesize both grey and RGB image with inception scores fairly close to the real data and samples generated by regular GANs without any privacy protection.

Lemma 1. *Let algorithm $\mathcal{A} : \mathbb{R}^n \rightarrow \mathbb{R}$ be a randomized algorithm that is (ϵ, δ)-differentially private. Let $f : \mathbb{R} \rightarrow \mathbb{R}'$ be an arbitrary randomized mapping. Then $f \circ \mathcal{A} : \mathbb{R}^n \rightarrow \mathbb{R}'$ is (ϵ, δ)-differentially private.*

Meanwhile, it is well-known that larger amount of training data causes less privacy loss, and allows for more iterations within a moderate privacy budget [30]. Due to the scarcity of

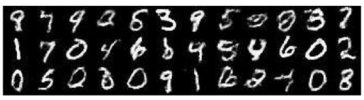


Fig. 5. Generated DPGAN samples with $\epsilon = 4, \delta = 10^{-5}$ using the augmented 60 000 MNIST examples of one party who owns 600 original MNIST examples.

training data of each party, data augmentation is exploited to expand local data size of each party to 100 times, which allows DPGAN to generate realistic samples within a moderate privacy budget. In particular, we augment original data with rotation range of 1 and width shift range and height shift range of 0.01.

In our study, we use moments accountant described in [30] to track the spent privacy over the course of training. Our DPGAN is able to generate realistic MNIST samples with $\epsilon = 4$ and $\delta = 10^{-5}$, as shown in Fig. 5. Note that each party can individually train DPGAN and generate massive DPGAN samples offline without affecting collaboration.

5.2 Privacy-Preserving Collaborative Deep Learning

Algorithm 2 summarizes the steps for the privacy-preserving collaborative deep learning in each communication round, including how to update points as per upload/download, how to preserve privacy of individual model updates using three-layer onion-style encryption followed by parameter and local credibility update, and credible party set maintenance by the Blockchain system. In particular, the gradients download budget of party i, i.e., d_i, is closely related with how many points p_i party i has in each communication round. More concretely, d_i should not exceed p_i, otherwise, party i will not have enough points to pay for the gradients provided by other parties. Moreover, d_i can be dynamically determined based on the existing points p_i in each communication round. For simplicity, we initialize $d_i = p_i$ in each communication round, but how many gradients can be downloaded will be dependent on both the local credibility list of the requester and sharing levels of the requested parties, which can be referred to Section 4.1. In the following sections, we will focus on the most important details for parameter update, three-layer onion-style encryption, and local credibility update.

5.2.1 Parameter Update With Homomorphic Encryption

Sharing gradients can prevent direct exposure of the local data, but may indirectly disclose local data information. To further prevent potential privacy leakage from sharing gradients and facilitate gradients aggregation during the collaborative learning process, we use additive homomorphic encryption such that each party can only decrypt the sum of all the received encrypted gradients. Specifically, Vernam cipher or one-time pad (OTP) has been mathematically proved to be completely secure, which cannot be broken given enough ciphertext and time. Therefore, we use simple and provably secure OTP for additively homomorphic encryption that allows efficient aggregation of encrypted data [41], [42]. The main idea of forming the ciphertext is to combine the keystream with the plaintext digits. Meanwhile, rather than XOR operation typically found in stream ciphers, which is unsecured under the frequency analysis attacks, our encryption scheme uses modular addition (+), and is hence very efficient [41]. The security relies on two important features: (1) the keystream changes from one message to another; and (2) all the operations are performed modulo a large integer M [41].

Algorithm 2. Privacy-Preserving Collaborative Deep Learning

Input: C, c_i^j, p_i, p_j, d_i, λ_j, $\boldsymbol{w}_i$, $\Delta \boldsymbol{w}_j$
Output: updated points p_j', p_i', **parameters** w_i', **and local credibility** $c_i^{j'}$

1: **Trade gradients as per download requests, local credibility, and sharing level; Party points update**: In each communication round, party i aims to download total $d_i = p_i$ gradients from all parties in C, while party $j \in C \setminus i$ can at most provide $\lambda_j \times |\Delta \boldsymbol{w}_j|$ gradients, one point is consumed/rewarded for each download and upload. Each party i updates local model parameters based on the gradients of party $j \in C \setminus i$ as follows:
for $j \in C \setminus i$ **do**
$d_i^j = min(c_i^j * d_i, \lambda_j * |\Delta \boldsymbol{w}_j|)$, $p_j' = p_j + d_i^j$, $p_i' = p_i - d_i^j$
$\Delta \boldsymbol{w}_{j^i} = \Delta \boldsymbol{w}_j$, party j first chooses d_i^j meaningful gradients from $\Delta \boldsymbol{w}_{j^i}$ according to "largest values" criterion: sort gradients in $\Delta \boldsymbol{w}_{j^i}$ and choose d_i^j of them, starting from the largest, then masks the remaining $|\Delta \boldsymbol{w}_{j^i}| - d_i^j$ gradients with 0 as $\Delta \tilde{\boldsymbol{w}}_j^i$.
end for
2: **Three-layer onion-style encryption**: Party j follows Algorithm 3 to encrypt the masked gradients $\Delta \tilde{\boldsymbol{w}}_j^i$ with its keystream k_j as $c = Enc(\Delta \tilde{\boldsymbol{w}}_j^i, k_j)$, and re-encrypts the encrypted gradients c with a fresh symmetric encryption key fsk as $Senc(c, fsk)$, the symmetric encryption key of the second layer is encrypted in the third layer by the receiver party i's public key pk_i as $Aenc(fsk, pk_i)$. Finally, the two-layer encrypted gradients $Senc(c, fsk)$ and the encrypted fresh symmetric encryption key $Aenc(fsk, pk_i)$ are sent to party i;
3: **Parameter update**: party i uses the paired secret key sk_i to decrypt the received encrypted fresh symmetric encryption key as fsk, then uses fsk to decrypt the two-layer encrypted gradients as $c = Enc(\Delta \tilde{\boldsymbol{w}}_j^i, k_j)$, finally decrypts the sum of all the received gradients using homomorphic property and updates local parameters by integrating all its plain gradients $\Delta \boldsymbol{w}_i$ as: $\boldsymbol{w}_i' = \boldsymbol{w}_i + \Delta \boldsymbol{w}_i + Dec(\sum_{j \in C \setminus i} Enc(\Delta \tilde{\boldsymbol{w}}_j^i, k_j), -k_i) = \boldsymbol{w}_i + \Delta \boldsymbol{w}_i + \sum_{j \in C \setminus i} \Delta \tilde{\boldsymbol{w}}_j^i$, where $\boldsymbol{w}_i$ is party i's local parameters at previous communication round.
4: **Local credibility update**: party i randomly selects and releases u_i artificial samples to any party j for labelling, mutual evaluation is repeated by following Step 3 of Algorithm 1 to calculate local credibility of party j at current communication round as $c_i^{j'}$. Party i updates local credibility of party j by integrating its historical credibility as: $c_i^{j'} = 0.2 * c_i^j + 0.8 * c_i^{j'}$, where c_i^j is the local credibility of party j at previous communication round.
5: **Local credibility normalization**: $c_i^{j'} = \frac{c_i^{j'}}{\sum_{j \in C} c_i^{j'}}$
if $c_i^{j'} < c_{th}$ **then**
party i reports party j as a low-contribution party
end if
6: **Credible party set**: If the majority of parties report party j as low-contribution, Blockchain removes party j from credible party set C and all parties run Step 5 again.

The detailed procedure for homomorphic encryption is presented in Algorithm 3. In practice, if $p = \max(x_i)$, M is derived as $M = 2^{\lceil log_2(p \times n) \rceil}$. All computations in the

remainder of this paper are modulo M unless otherwise stated. However, all the original floating-point values need to be mapped to the integer domain by using Scaling, Rounding, Unscaling (SRU) algorithm [42]. A pseudorandom keystream k can be generated by a secure pseudo random function (PRF) by implementing a secure stream cipher, such as Trivium [43], keyed with each party's keystream k_i and a unique message ID. For encryption purpose, the secret keys are pre-computed through a trusted setup, which can be performed by a trusted dealer or through a standard SMC protocol.

For example, a trusted key managing authority can generate these keystreams in each communication round, but the generated keystreams cannot be used more than once. The trusted setup generates non-zero random shares of 0: $\sum_{i\in C} k_i = 0$, such that each participant $i \in C$ obtains a keystream k_i. Note that if the Blockchain removes party j from the credible party set C, a new credible party set C should be constructed.

Algorithm 3. Homomorphic Encryption Scheme

Setup
1: A trusted dealer randomly generates $|C|$ keystreams: $k_1, \ldots, k_{|C|} \in [0, M-1]$, such that $\sum_{i\in C} k_i \pmod M = 0$, where M is a large integer.
2: Party i obtains keystream k_i.
Enc(m, k)
1: Represent message m as integer $m \in [0, M-1]$.
2: Let k be a randomly generated keystream, where $k \in [0, M-1]$.
3: Compute $c = Enc(m, k) = m + k$.
Dec(c, k)
1: $Dec(c, k) = c - k$.
AggrDec(k_i)
1: Let $c_j = Enc(m_j, k_j)$, where $j \in C \setminus i$.
2: Party i uses $-k_i = \sum_{j\in C\setminus i} k_j$ to decrypt the aggregation of other parties as follows: $Dec(\sum_{j\in C\setminus i} c_j, -k_i) = \sum_{j\in C\setminus i} c_j - \sum_{j\in C\setminus i} k_j = \sum_{j\in C\setminus i} m_j$.

Model parameter of party i is updated as per gradients-encrypted SGD as follows:

$$\begin{aligned} \boldsymbol{w}_i' &= \boldsymbol{w}_i + \Delta\boldsymbol{w}_i + Dec(\textstyle\sum_{j\in C\setminus i} Enc(\Delta\tilde{\boldsymbol{w}}_j^i, k_j), -k_i) \\ &= \boldsymbol{w}_i + \Delta\boldsymbol{w}_i + \textstyle\sum_{j\in C\setminus i} \Delta\tilde{\boldsymbol{w}}_j^i, \end{aligned}$$

where Enc and Dec correspond to encryption and decryption operations in Algorithm 3, $\boldsymbol{w}_i$ is the local parameters of party i at previous round, $\Delta\tilde{\boldsymbol{w}}_j^i$ is the masked gradient vector of party j shared with party i, where only d_i^j gradients are meaningful, i.e., d_i^j elements of total $|\Delta\tilde{\boldsymbol{w}}_j^i|$ elements are kept intact, while the remaining $|\Delta\tilde{\boldsymbol{w}}_j^i| - d_i^j$ elements are nullified as 0. The second equality follows the homomorphic addition property, thus participant i can get the updated $\boldsymbol{w}_i'$ correctly after decryption, without having access to either $\Delta\tilde{\boldsymbol{w}}_j^i$ or $\Delta\boldsymbol{w}_j$. FPPDL ensures party obliviousness by ensuring that each participant knows nothing but the sum of its received gradients in each communication round, and cannot infer any information about other participants' data.

5.2.2 Three-Layer Onion-Style Encryption

However, as all parties need to store different encrypted gradients that are meant to be sent to different parties on Blockchain for commitment, all the encrypted gradients are also accessible to all parties. Applying public-key encryption on top of homomorphic encryption for authentication [42] can address this problem. However, as the released gradient vector is high-dimensional, encrypting gradient vector is both computation and communication expensive.

Therefore, we propose a three-layer onion-style encryption scheme. The first layer protects local model gradients by using symmetric key keystream k_j for homomorphic encryption, as presented in Algorithm 3. The second layer and the third layer are classic hybrid encryption, as used in OpenPGP [44] for instance. In particular, in the second layer, a fresh symmetric encryption key fsk will be generated and used to re-encrypt the ciphertext of the first layer, and then the fresh symmetric key is encrypted by using the receiver's public key pk_i in the third layer. In this way, the encryption of high-dimensional data becomes very effective, and the receiver could be authenticated as well: only the receiver who has the corresponding secret key sk_i paired with the public key pk_i can decrypt the two-layer encrypted gradients committed on the Blockchain.

5.2.3 Local Credibility Update

Instead of using the standalone models as in the local credibility initialization, during each round of collaborative learning, each party randomly selects and shares a subset of DPGAN samples as per individual sharing level, then calculates the local credibility of other parties based on the returned labels, which are evaluated by using its updated local model at current round. The mutual evaluation follows the same procedure as in Step 3 of Algorithm 1. Finally, local credibility of each party is updated by integrating its historical local credibility as per Step 4 of Algorithm 2. In this way, local credibility of each party can be adaptively updated, reflecting more accurately how one party contributes to different parties during collaborative learning.

5.3 Quantification of Fairness

In collaborative learning system, collaborative fairness should be quantified from the point of view of the whole system. In this work, we quantify collaborative fairness through the correlation coefficient between party contributions (i.e., standalone model accuracy which characterizes the learning capability of each party on its own local data, and sharing level, which characterizes the sharing willingness of each party) and party rewards (i.e., final model accuracies of different parties).

Specifically, we take party contributions as the X-axis, which represents the contributions of different parties from the system view. In particular, in Setting 2, we characterize different parties' contributions by their sharing levels and standalone model accuracies, as the party who is less private and has local data with better generalization empirically contributes more. In Setting 1 and Setting 3, we characterize different parties' contributions by their standalone model accuracies, as the party who has local data with better generalization empirically contributes more. Moreover, in Setting 3, the party with more local data typically yields higher standalone model accuracy in IID scenarios. In summary, the X-axis can be expressed by Equation (2), where λ_j and

$sacc_j$ denote the sharing level and standalone model accuracy of party j respectively

$$\boldsymbol{x} = \begin{cases} \left\{\frac{\lambda_1}{\sum \lambda_j}, \ldots, \frac{\lambda_n}{\sum \lambda_j}\right\} + \left\{\frac{sacc_1}{\sum sacc_j}, \ldots, \frac{sacc_n}{\sum sacc_j}\right\}, & \text{Setting 2} \\ \{sacc_1, \ldots, sacc_n\}, & \text{Setting 1\&3} \end{cases} \tag{2}$$

Similarly, we take party rewards (i.e., final model accuracies of different parties) as the Y-axis, as expressed by Equation (3), where acc_j denotes the final model accuracy of party j

$$\boldsymbol{y} = \{acc_1, \ldots, acc_n\}. \tag{3}$$

As the Y-axis measures local model performance of different parties after collaboration, it is expected to be positively correlated with the X-axis to deliver good fairness. Hence, we formally quantify collaborative fairness in Equation (4)

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y}, \tag{4}$$

where $\bar{x}$ and $\bar{y}$ are the sample means of $\boldsymbol{x}$ and $\boldsymbol{y}$, s_x and s_y are the corrected standard deviations. The range of fairness is within [-1, 1], with higher values implying good fairness. Conversely, negative coefficient implies poor fairness.

6 Experimental Evaluation

In this section, we evaluate the performance of the proposed FPPDL framework by comparing it against the state of the art on real-world datasets.

6.1 Datasets

We implement experiments on two benchmark image datasets. The first is the MNIST dataset[1] for handwritten digit recognition consisting of 60,000 training examples and 10,000 test examples. Each example is a 32 x 32 gray-level image [11], with digits locating at the center of the image. The second is the SVHN dataset[2] of house numbers obtained from Google's street view images, which contains 600,000 training examples, from which we use 100,000 for training and 10,000 for testing. Each example is a 32 x 32 centered image with three channels (RGB). SVHN is more challenging as most of the images are noisy, and contain distractors at the sides. The size of the input layer of neural networks for MNIST and SVHN are 1,024 and 3,072, respectively. The objective is to classify the input as one of 10 possible digits within ["0"-"9"], thus the size of the output layer is 10. We normalize the training examples by subtracting the average and dividing by the standard deviation of training examples. For reproducibility purposes, our code will be made available here: https://github.com/lingjuanlv/FPPDL.

6.2 Baselines

We demonstrate the effectiveness of our proposed FPPDL framework by comparison with the following three frameworks. In all frameworks, stochastic gradient descent (SGD) is applied to each party.

1. http://yann.lecun.com/exdb/mnist/
2. http://ufldl.stanford.edu/housenumbers/

1) *Standalone* framework: which assumes parties train standalone models on local training data without any collaboration. This framework delivers maximum privacy, but minimum utility, because each party is susceptible to falling into local optima when training alone.
2) *Centralized* framework: which allows a trusted server to have access to all participants' data in the clear, and train a global model on the combined data using standard SGD. Hence, it is a privacy-violating framework.
3) *Distributed* framework: which enables parties to train independently and concurrently, and chooses a fraction of parameters to be uploaded at each iteration. In particular, as shown in [11], Distributed Selective SGD achieves even higher accuracy than the centralized SGD because updating only a small fraction of parameters at each round acts as a regularization technique to avoid overfitting. Hence, we take DSSGD for the analysis of the distributed framework. As DSSGD with round robin parameter exchange protocol results in the highest accuracy [11] and facilitates fairness calculation, we follow the round robin protocol for DSSGD, where participants run SSGD sequentially, each downloads a fraction of the most updated parameters from the server, runs local training, and uploads selected gradients; the next party follows in the fixed order. Gradients are uploaded according to the "largest values" criterion.

6.3 Experiment Setup

For local model architecture, we consider two popular neural network architectures: *multi-layer perceptron* (MLP) and *convolutional neural network* (CNN), which are the same as in [11]. For local model training, we set the learning rate as 0.001, learning rate decay as 1e-7, and mini-batch size as 1. In addition, to reduce the impact of different initializations and avoid non-convergence, each party is initialized with the same parameter w_0, then local training is run on individual training data to update local model parameter $\boldsymbol{w}_i$. To boost fairness, we let each party individually train 10 epochs before collaborative learning starts. For all experiments, we empirically set the local credibility threshold as $c_{th} = \frac{1}{|C|-1} * \frac{2}{3}$ via grid search, where $|C|$ is the number of alive parties, i.e., credible parties in the system. Next, we investigate three realistic IID settings as follows:

Setting 1: Same sharing level, same data size: in the first case, sharing level of each party is set as 0.1, i.e., each party only releases 10 percent meaningful gradients during collaboration. For each party, we randomly sample 1 percent of the entire database as the local training data of each party, i.e., 600 examples for MNIST and 1000 examples for SVHN, this setting is the same as Shokri *et al.* [11] when the upload rate of each party equals 0.1;

Setting 2: Different sharing level, same data size: in the second case, sharing level of each party is randomly sampled from [0.1, 0.5], and parties release meaningful gradients as per individual sharing level during collaboration. For each participant, we randomly sample 1 percent of the entire database as local training data as above.

TABLE 3
Fairness of Distributed Framework and Our FPPDL Over MNIST Dataset, With Different Model Architectures, Different Party Numbers (P-k) and Different Settings as Described in Section 6.3:

	Setting 2				Setting 3			
	Distributed		FPPDL		Distributed		FPPDL	
	CNN	MLP	CNN	MLP	CNN	MLP	CNN	MLP
P4	-0.68	0.30	**0.89**	**0.92**	-0.97	0.05	**0.98**	**0.96**
P15	0.20	-0.15	**0.76**	**0.82**	0.03	-0.07	**0.90**	**0.83**
P30	-0.02	0.02	**0.79**	**0.85**	0.13	0.01	**0.75**	**0.63**
P50	-0.16	-0.05	**0.75**	**0.67**	0.14	-0.07	**0.72**	**0.60**

TABLE 4
Fairness of Distributed Framework and Our FPPDL Over SVHN Dataset, With Different Model Architectures, Different Party Numbers (P-k) and Different Settings

	Setting 2				Setting 3			
	Distributed		FPPDL		Distributed		FPPDL	
	CNN	MLP	CNN	MLP	CNN	MLP	CNN	MLP
P4	0.27	0.26	**0.78**	**0.76**	0.28	0.20	**0.97**	**0.93**
P15	0.16	0.19	**0.77**	**0.71**	-0.13	0.16	**0.87**	**0.88**
P30	-0.14	0.12	**0.68**	**0.65**	-0.15	-0.27	**0.67**	**0.78**
P50	-0.25	-0.37	**0.67**	**0.66**	-0.23	0.15	**0.65**	**0.69**

Setting 3: Different data size, same sharing level: in the third case, we simulate the case where different parties have different data size. In particular, for MNIST dataset, we randomly partition total {2,400, 9,000, 18,000, 30,000} examples among {4,15,30,50} parties respectively. Similarly, for SVHN dataset, total {4,000, 15,000, 30,000, 50,000} examples are randomly partitioned among {4,15,30,50} parties respectively. The sharing level of each party is fixed to 0.1.

Remark. In Setting 1 and Setting 2, the purpose of allocating 600 MNIST examples or 1000 SVHN examples for each party is to fairly compare with Shokri *et al.* [11], in which each party is allocated with 600 MNIST examples or 1000 SVHN examples (small number of local examples to simulate data scarity which necessitates collaboration). Therefore, for MNIST, we simulate the total examples of 2,400 (4 parties) up to 30,000 (50 parties). For larger datasets like 300,000 examples, it would require 500 parties, imposing heavy requirement on real deployment, while delivering similar analysis as in Section 6.4. We also remark that our Setting 2 and Setting 3 are relatively conservative, by increasing the contribution diversity among parties, for example, sampling sharing level from [0, 1] instead of [0, 0.5], partitioning data size among parties in a more imbalanced way, our FPPDL can definitely results in higher fairness.

6.4 Experimental Results

For collaborative fairness comparison, we only analyze our FPPDL and the distributed framework using DSSGD, neglecting centralized framework and standalone framework, because parties cannot get access to the trained global model in the centralized framework, while parties do not collaborate in the standalone framework. Tables 3 and 4 list the calculated fairness of the distributed framework and our FPPDL over MNIST and SVHN datasets, with different architectures, different party numbers and different settings, as detailed in Section 6.3. In particular, we omit the results for setting 1 with the same sharing level and same data size, as fairness is a less concerned problem in this setting. All the fairness results for setting 2 and setting 3 are averaged over five trails to reduce the impact of different initialization in each trail.

As is evidenced by the high positive values of fairness, with most of them above 0.5, FPPDL achieves reasonably good fairness, confirming the intuition behind fairness: the party who is less private and has more training data delivers higher accuracy. In contrast, the distributed framework exhibits bad fairness with significantly lower values than that of FPPDL in all cases, and even negative values in some cases, manifesting the lack of fairness in the distributed framework. This is because in the distributed framework, all the participating parties can derive similarly well models, no matter how much one party contributes.

System-Level Convergence. For accuracy comparison, following [11], we report the best accuracy when running the distributed framework using DSSGD and our FPPDL on MNIST dataset. For DSSGD, we adopted round robin protocol, and set the upload rate as 0.1 ($\theta_u = 0.1$) [11], which is equivalent to our Setting 1, we omit the learning curves of DSSGD in Setting 2 and Setting 3 as they approximate the learning curve in Setting 1. Figs. 6 and 7 present the accuracy trajectories when running different frameworks over MNIST with MLP and CNN architectures. The x-axis corresponds to epochs (communication rounds) (1 round=1 epoch, when the number of local epochs $E = 1$), and y axis corresponds to the maximum accuracy achieved by all parties in each round, hence the curve of our FPPDL is not necessarily associated with a particular party, but it is expected that the highest accuracy is achieved by the most contributive party in our FPPDL, as demonstrated by the individual convergence in Figs. 9, 10 and 11.

Note that the convergences of the standalone framework in Setting 1 and Setting 2 are the same, as these two settings share the same data shard. It can be observed that FPPDL did not change the overall behavior of convergence in all settings, while achieving comparable accuracy to the non-private frameworks, and delivering both fairness and privacy. We notice that our FPPDL achieves slightly slower convergence rate and more fluctuations (especially in early stages of convergence) compared to the distributed framework, this is partly attributed to the individual training of 10 epochs before collaborative learning starts, as we found that collaboration from the state of 10 epochs of local training results in better fairness than the collaboration from the beginning.

Another important reason is that to strike a good balance between computational efficiency, communication cost and convergence rate, we enforce parties to share their local model updates after each epoch of local training ($E = 1$), where the shared gradients is the average of the gradients over the whole local training data, rather than a single example, a mini-batch or multiple local epochs, which may also affect convergence. We hypothesise that the convergence rate is also closely related with our chosen hyperparameters $B = 1, E = 1, lr = 0.001$ (E: number of local training epochs in each communication round; B: local batch size; lr: local learning rate). Better convergence can be achieved by

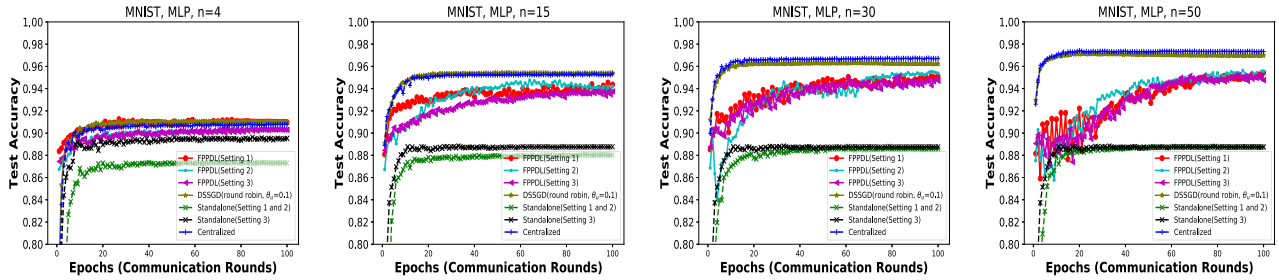


Fig. 6. System convergence for MNIST MLP. Collaboration involves different number of parties in $\{4, 15, 30, \text{and } 50\}$.

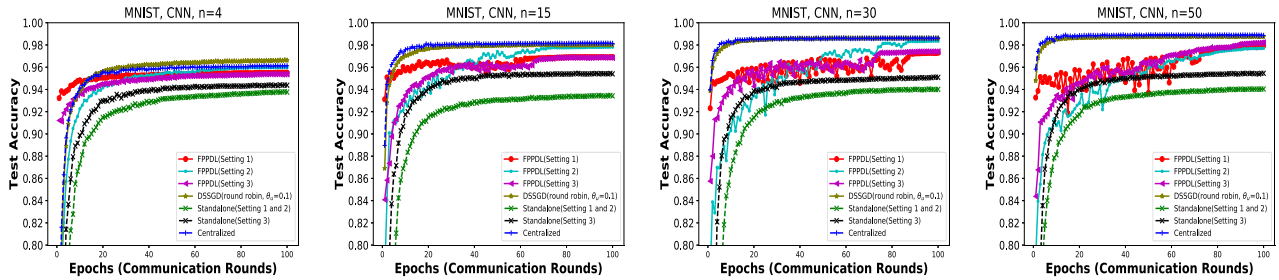
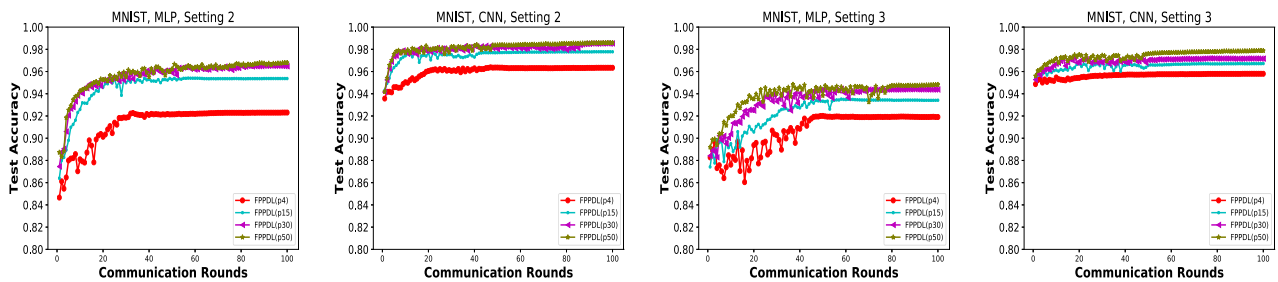
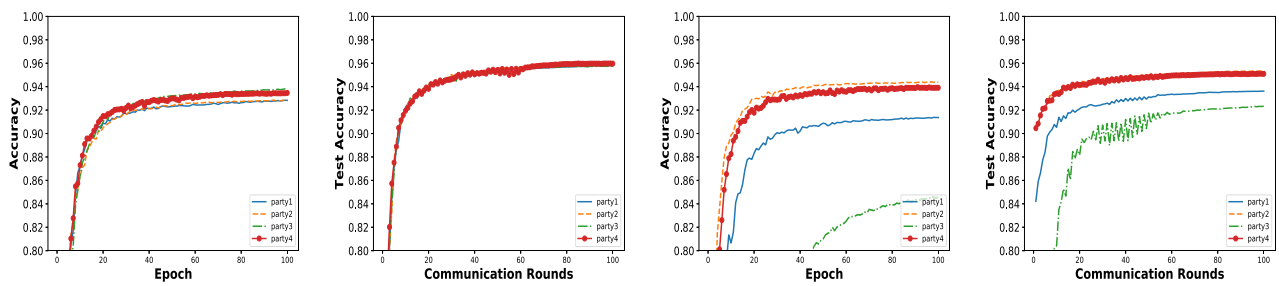


Fig. 7. System convergence for MNIST CNN. Collaboration involves different number of parties in $\{4, 15, 30, \text{and } 50\}$.



(a) FPPDL Setting 2 (MLP) (b) FPPDL Setting 2 (CNN) (c) FPPDL Setting 3 (MLP) (d) FPPDL Setting 3 (CNN)

Fig. 8. System convergence for MNIST MLP and CNN using our FPPDL in Setting 2 and Setting 3 (B=10, E=5, and lr=0.15).

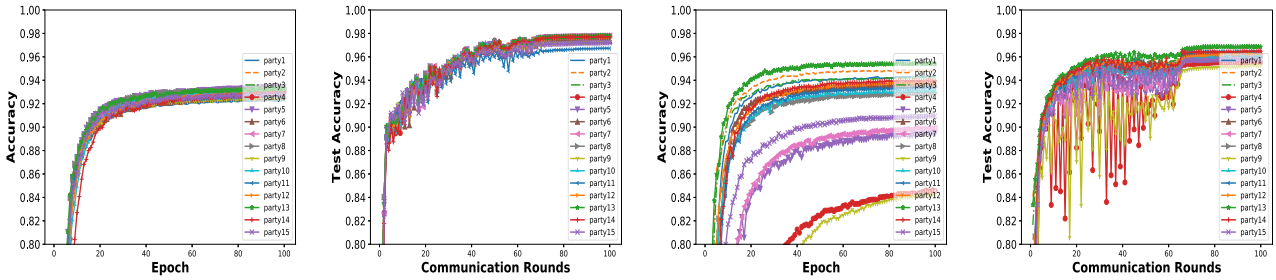


(a) Standalone Setting 2 (b) FPPDL Setting 2 (c) Standalone Setting 3 (d) FPPDL Setting 3

Fig. 9. Individual convergence for MNIST CNN using Standalone framework and our FPPDL (P4, B=1, E=1, lr=0.001).

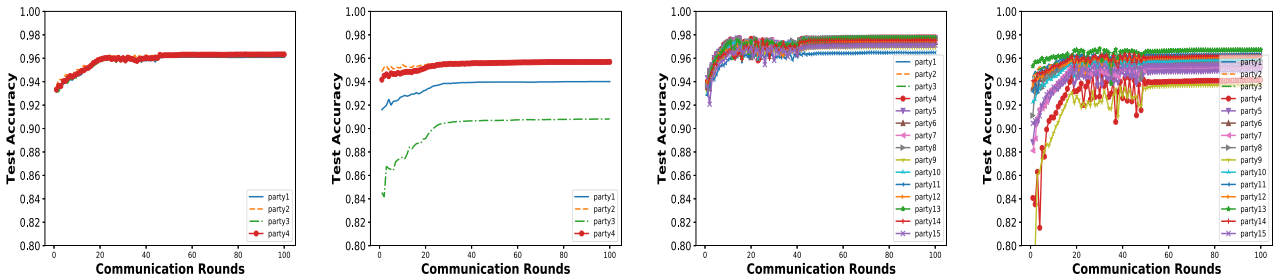
varying the amount of local computation per communication round, local batch size or the learning rate, as indicated in Figs. 8 and 11 by using B=10, E=5, lr=0.15.

Individual Convergence. To investigate the impact of our FPPDL on individual convergence, Figs. 9 and 10 further depict the accuracy trajectory of each party when running Standalone framework and our FPPDL with CNN architecture over MNIST across 100 communication rounds. For the sake of brevity, we only report experimental results obtained for the collaboration among 4 parties and 15 parties in Setting 2 and Setting 3. It can be observed that our FPPDL consistently delivers better accuracy than any standalone model obtained by any individual party, at the cost of slower convergence and more fluctuation. However, most parties can converge within the first 20 rounds, except those with lower standalone accuracy. For example, in Fig. 10d, party 4 and party 9 encounter higher fluctuations compared with the other parties with higher standalone accuracy. More importantly, these



(a) Standalone Setting 2 (b) FPPDL Setting 2 (c) Standalone Setting 3 (d) FPPDL Setting 3

Fig. 10. Individual convergence for MNIST CNN using Standalone framework and our FPPDL (P15, B=1, E=1, lr=0.001).



(a) FPPDL Setting 2 (P4) (b) FPPDL Setting 3 (P4) (c) FPPDL Setting 2 (P15) (d) FPPDL Setting 3 (P15)

Fig. 11. Individual convergence for MNIST CNN using our FPPDL in P4 and P15 (B=10, E=5, lr=0.15).

TABLE 5
MNIST Accuracy [%] After 100 Communication Rounds, Achieved by *Centralized*, *Standalone*, *Distributed* (DSSGD Without DP, Round Robin, $\theta_u = 10\%$), and FPPDL (Three Settings as Described in Section 6.3) Frameworks Using MLP and CNN Architectures

Framework	MLP				CNN			
	P4	P15	P30	P50	P4	P15	P30	P50
Centralized	91.68	95.17	96.28	96.85	96.58	98.19	98.52	98.58
Distributed	91.67	95.17	96.33	97.35	96.25	98.04	98.63	98.83
Standalone (Setting 1&2)	87.39	88.06	88.64	88.80	93.81	93.46	94.04	94.05
Standalone (Setting 3)	89.61	88.83	89.57	89.52	94.42	95.44	95.11	95.45
FPPDL (Setting 1)	90.13	94.42	94.88	95.57	95.93	97.19	97.62	98.07
FPPDL (Setting 2)	91.92	95.70	95.94	96.23	95.50	97.34	97.84	98.14
FPPDL (Setting 3)	90.75	94.37	94.75	95.21	95.23	97.50	97.82	98.22

P-k indicates there are k parties in the experiments.

figures confirm that our FPPDL enforces all parties to converge to different local models, which are better than their standalone models without any collaboration, thereby offering fairness as claimed.

To speed up convergence and alleviate fluctuations, we further experiment with larger number of local epochs, larger local batch size, and higher learning rate. As corroborated by Fig. 11, by setting $B = 10, E = 5, lr = 0.15$, each party can converge faster, without affecting both accuracy and fairness. For example, for P15 in Fig. 10d, it needs 65 communication rounds for all parties to converge using $B = 1, E = 1, lr = 0.001$, while it only needs 50 communication rounds using $B = 10, E = 5, lr = 0.15$ in Fig. 11d. However, this faster convergence and less fluctuations come at the cost of local computation at each party.

Table 5 provides the accuracy results we obtain when running different frameworks on MNIST dataset of {4,15,30,50} parties for different neural network architectures. For all frameworks, we report the best accuracy the system can achieve across all rounds. In particular, in our FPPDL, fairness enables each party to get a different local model after collaborative learning, and we expect that the most contributive party derives a local model with maximum accuracy approximating the non-private centralized and distributed frameworks. Similarly, Table 6 provides the accuracy on SVHN dataset. For both MNIST and SVHN datasets using CNN and MLP architectures, we show the worst accuracy for standalone SGD (minimum utility, maximum privacy). In particular, FPPDL obtains comparable accuracy (less than 2 percent) to both the centralized framework and the distributed framework using DSSGD without differential privacy, and consistently achieves higher accuracy than the standalone SGD. For example, as shown in Table 5, for MNIST dataset of 50 parties with CNN model, our FPPDL achieves 98.07-98.22 percent test accuracy under different settings, which is higher than the standalone SGD 94.05 percent, and comparable to 98.83

TABLE 6
SVHN Accuracy [%] After 100 Communication Rounds, Achieved by *Centralized*, *Standalone*, *Distributed* (DSSGD Without DP, Round Robin, $\theta_u = 10\%$), and FPPDL (Three Settings as Described in Section 6.3) Frameworks Using MLP and CNN Architectures

Framework	MLP				CNN			
	P4	P15	P30	P50	P4	P15	P30	P50
Centralized	75.40	83.08	85.77	87.15	90.50	91.88	93.42	95.44
Distributed	78.34	85.49	87.64	89.21	91.78	93.03	95.75	96.19
Standalone (Setting 1&2)	57.85	58.77	57.90	59.18	80.24	80.74	81.29	81.60
Standalone (Setting 3)	59.05	59.13	60.09	60.22	81.57	81.92	82.06	82.31
FPPDL (Setting 1)	73.74	82.55	84.86	86.51	90.07	91.18	92.74	94.83
FPPDL (Setting 2)	74.16	82.67	85.25	86.57	89.91	91.15	92.59	95.18
FPPDL (Setting 3)	74.57	82.95	85.37	86.34	89.53	91.03	93.13	94.89

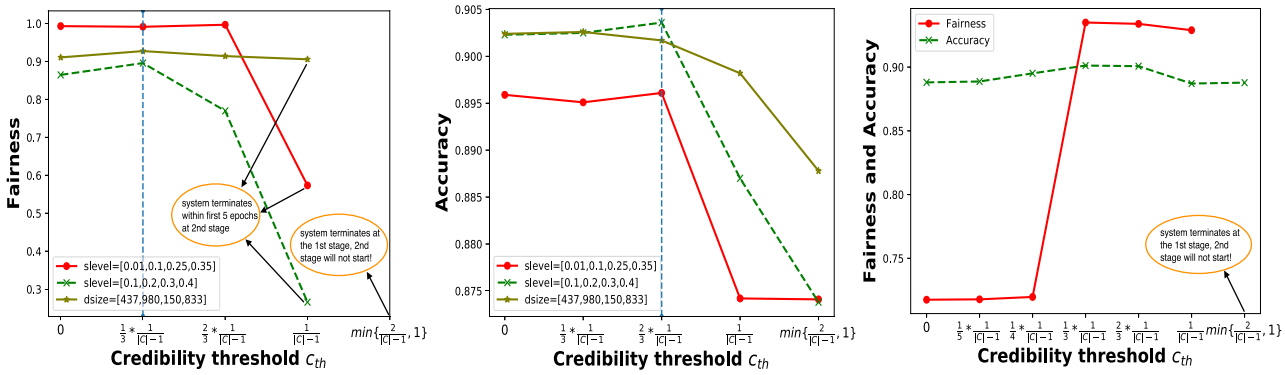


Fig. 12. How c_{th} affects fairness and accuracy in normal and malicious settings. First two figures correspond to setting 2 with sharing level of [0.01,0.1,0.25,0.35] and [0.1,0.2,0.3,0.4], and setting 3 with data shard of [437,980,150, and 833] among four honest parties. The last figure simulates setting 3 with data shard of [437,980,150, and 833]) among four honest parties and one more malicious party indicated in Section 7.

percent of the distributed framework using DSSGD without differential privacy, and 98.58 percent of the centralized framework.

The above fairness results in Tables 3 and 4, and accuracy results in Tables 5 and 6 demonstrate that *our proposed framework FPPDL achieves reasonable fairness, at the expense of a tiny decrease in model utility*.

Moreover, to investigate how fairness and accuracy change with the local credibility threshold c_{th}, we implement a four-party scenario (P4) under both normal settings (Setting 2 and Setting 3 in Section 6.3) and malicious setting (1 malicious party as indicated in Section 7). As shown in Fig. 12, both fairness and accuracy can keep relatively high values when c_{th} is within $[\frac{1}{3} * \frac{1}{|C|-1}, \frac{2}{3} * \frac{1}{|C|-1}]$. In contrast, too small $c_{th} < \frac{1}{3} * \frac{1}{|C|-1}$ allows even the malicious party to sneak into the collaborative learning system without being detected and isolated, resulting in lower fairness, as manifested by the last figure of Fig. 12. On the contrary, too large c_{th} might isolate most participants in the system. For example, $c_{th} = \frac{1}{|C|-1}$ will terminate the system within the first 5 rounds during the second stage of collaborative learning, resulting in both lower fairness and accuracy; and $c_{th} = min\{\frac{2}{|C|-1}, 1\}$ will terminate the system after the first stage, and second stage of collaborative learning will never start, thus there is no collaborative fairness. These results validate our hypothesis in Section 5.1.1 and provide empirical support on our chosen $c_{th} = \frac{1}{|C|-1} * \frac{2}{3}$ in Section 6.3.

Complexity Analysis. The main communication cost occurs when each party sends its encrypted gradients to the other $(n-1)$ parties, resulting in $(n-1) * L$ ciphertexts, where n and L are the number of parties and the size of the released gradients (the encrypted symmetric key size is negligible compared with the encrypted gradients). Therefore, our framework is applicable to practical applications to businesses [37], such as biomedical or financial institutions where the number of parties is limited. On the other hand, the main computation cost occurs at each party who needs to train a local DPGAN during initial benchmarking, compute local gradients, and conduct three-layer onion-style encryption during collaborative deep learning. However, all parties can individually train their DPGAN models offline before collaborative deep learning starts, and all parties can individually train local models in parallel, hence deep learning computation cost is not an obstacle for those parties with enough computational power. Moreover, our encryption scheme using stream ciphers and hybrid encryption is relatively efficient, because encrypting a short plaintext (i.e., the symmetric key) requires only one asymmetric operation, while encrypting a longer message (released gradients) would in theory require many asymmetric operations.

7 Discussions

Data Augmentation and Collaboration. To facilitate credibility initialization, we apply data augmentation to expand local data size to *help DPGAN generate reliable samples within a moderate privacy budget*. However, data augmentation is intended to increase the amount of training data using information inherent in local training data, and thus improve the

generalizability of local model, while not helpful for generalizing to unseen data. In other words, it cannot represent global distribution, and this explains why parties still need collaboration for better utility even after data augmentation. By using DPGAN, it not only preserves privacy of the original data, but also preserves privacy of the augmented data that are similar to the original data.

Fairness and Privacy. With three-layer onion-style encryption, privacy is better preserved without compromising utility. We ensure fairness from two ways: (i) during initial benchmarking, parties generate DPGAN samples based on their local training data, which are then evaluated by other parties' standalone models to mutually initialize the local credibilities of other parties; and (ii) during collaborative learning process, each party randomly selects and shares a subset of DPGAN samples as per individual sharing level at each round of communication, then updates the local credibility values for other parties who evaluate the received DPGAN samples using their local models at current round. Therefore, local credibility of each party keeps changing, reflecting more accurate relative contribution and thus possessing better fairness. Differentially private training of deep models provides another alternative solution by releasing gradients after each epoch or several epochs of local training, thus enabling each party to verify the claims of other parties and update their local credibility values as per the received gradients during collaborative learning process. One obstacle is that differentially private models may significantly reduce utility for small ϵ values.

Attacker Prevention. Although the capability of detecting and isolating malicious parties is not the main focus of this paper, we next discuss how our design can help prevent certain behaviours of inside attacker, and resist the outside attacker as a by-product of FPPDL.

For an inside attacker who is a participant in the decentralized system, we specially consider an interesting case: a free-rider without any data, and we remark that this free-rider belongs to the category of low-contribution party. During initialization, this free-rider may choose to send the fake information to other parties. For example, it may randomly sample from 10 classes as predicted labels for the received DPGAN samples, then release them to the corresponding party who publishes these DPGAN samples and requests labels. When the publisher receives the returned random labels from the free-rider and detects that most of them are not aligned with the majority voting, i.e., $\frac{m_j}{u_i} \ll c_{th}$, then the free-rider will be reported as a "low-contribution" party. If the majority of parties report the free-rider as "low-contribution", then the Blockchain rules out the free-rider from the credible party set, and all parties would terminate the collaboration with the free-rider. In this way, such a malicious party is isolated from the beginning, while the collaboration among the remaining parties will not be affected. Even though the free-rider might succeed in initialization somehow, its local credibility would be significantly lower compared with the other honest parties.

To further detect and isolate this malicious party during the collaborative learning process, we repeat mutual evaluation at each round of collaborative learning by using samples generated at the initialization phase, i.e., each party randomly selects and shares a subset of DPGAN samples as per individual sharing level in each round of collaborative learning, then updates the local credibility values of other parties by comparing the majority labels with the received labels output by the local models of other parties in current round of training. Hence, the chance of the survival of the malicious party is significantly reduced, thus it will not dominate the whole system. Note that the lower bound of the acceptable credibility threshold can be agreed by the system requirement. For the outsider attacker like the eavesdropper who aims to steal the exchanged information by eavesdropping on the communication channels among parties, differential privacy used in the first stage and three-layer onion-style encryption applied in the second stage inherently prevent the success of this attack.

We recognize that our current design may not be resistant to all the malicious parties who can arbitrarily deviate from the protocol, sending incorrect and/or arbitrarily chosen messages to honest parties, aborting, omitting messages, and sharing their entire view of the protocol with each other. For example, a malicious party who aims to compromise other parties' local model integrity (prevent other parties from learning reasonable models) may adaptively or alternatively adjust its behaviour by behaving normally during releasing DPGAN samples to avoid being detected and kicked out, while poisoning the second stage by sending random local gradients or local gradients with the embedded backdoor behavior to the requester. However, in this case, this malicious party is unlikely to obtain a reasonable local model or steal any party's personal information.

Moreover, to prevent the success of the poisoning attack, one potential solution is to let each party repeat local prediction process on its hold-out validation set by using individually aggregated gradients. Each party will release a signal to indicate whether its aggregated gradients can give a reasonable accuracy result, or help improve prediction on local validation set, if majority party report local validation accuracy lower than a threshold, or negative gain on local validation accuracy, then the system terminates to avoid being further poisoned. We leave this open problem to our future work, and our current design is mainly for the business applications, where parties act with legal liabilities.

8 Conclusion and Future Work

This paper proposes FPPDL, a decentralized privacy-preserving deep learning framework with fairness considerations. Our enhanced framework shows the following properties: (1) it inherently resolves the relevant issues in the server-based frameworks, and investigates Blockchain for decentralization; (2) it makes the first investigation on the research problem of collaborative fairness in deep learning, by introducing a notion of local credibility and transaction points, which are initialized by initial benchmarking, and updated during privacy-preserving collaborative deep learning; (3) it combines *Differentially Private GAN* and a three-layer onion-style encryption scheme to guarantee both accuracy and privacy; (4) it provides a viable solution to detect and reduce the impact of low-contribution parties in the system. The experimental results demonstrate that our FPPDL achieves comparable accuracy to both the centralized and distributed selective SGD framework without differential privacy, and always delivers better results than the standalone framework, confirming the applicability of our proposed framework.

A number of avenues for further work are attractive. In particular, we would like to study how to quantify fairness in Non-IID setting, and investigate more malicious behaviours and byzantine or sybil adversary in the decentralized system. We also expect to deploy our system into a wide spectrum of real-world applications.

Acknowledgments

This work was supported, in part, by IBM PhD Fellowship; ANU Translational Fellowship; Nanyang Assistant Professorship (NAP); and NTU-WeBank JRI (NWJ-2019-007). The authors would like to thank Prof. Benjamin Rubinstein, Dr. Kumar Bhaskaran, and Prof. Marimuthu Palaniswami for their insightful discussions. This research was undertaken using the LIEF HPC-GPGPU Facility hosted at the University of Melbourne. This Facility was established with the assistance of LIEF Grant LE170100200.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[2] Y. Wang *et al.*, "Iterative learning with open-set noisy labels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8688–8696.
[3] X. Ma *et al.*, "Dimensionality-driven learning with noisy labels," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3361–3370.
[4] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, *Federated Learning*. San Rafael, CA, USA: Morgan & Claypool, 2019.
[5] H. B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017, pp. 1273–1282.
[6] R. Cummings, V. Gupta, D. Kimpara, and J. Morgenstern, "On the compatibility of privacy and fairness," in *Proc. 27th Conf. User Model. Adaptation Personalization*, 2019, pp. 309–315.
[7] M. Jagielski *et al.*, "Differentially private fair learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3000–3008.
[8] H. Yu *et al.*, "A fairness-aware incentive scheme for federated learning," in *Proc. 3rd AAAI/ACM Conf. AI Ethics Soc.*, 2020, pp. 393–399.
[9] J. Dean *et al.*, "Large scale distributed deep networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1223–1231.
[10] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *Proc. 23rd Int. Conf. Neural Inf. Process. Syst.*, 2010, pp. 2595–2603.
[11] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1310–1321.
[12] T.-T. Kuo and L. Ohno-Machado, "Modelchain: Decentralized privacy-preserving healthcare predictive modeling framework on private blockchain networks," 2018, *arXiv:1802.01746*.
[13] X. Chen, J. Ji, C. Luo, W. Liao, and P. Li, "When machine learning meets blockchain: A decentralized, privacy-preserving and secure design," in *Proc. IEEE Int. Conf. Big Data*, 2018, pp. 1178–1187.
[14] H. Kim, S.-H. Kim, J. Y. Hwang, and C. Seo, "Efficient privacy-preserving machine learning for blockchain network," *IEEE Access*, vol. 7, pp. 136 481–136 495, 2019.
[15] X. Zhu, H. Li, and Y. Yu, "Blockchain-based privacy preserving deep learning," in *Proc. Int. Conf. Inf. Secur. Cryptol.*, 2018, pp. 370–383.
[16] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Dependable Secure Comput.*, 2019.
[17] T.-T. Kuo, R. A. Gabriel, and L. Ohno-Machado, "Fair compute loads enabled by blockchain: Sharing models by alternating client and server roles," *J. Amer. Med. Informat. Assoc.*, vol. 26, no. 5, pp. 392–403, 2019.
[18] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 201–210.
[19] O. Ohrimenko *et al.*, "Oblivious multi-party machine learning on trusted processors," in *Proc. 25th USENIX Conf. Secur. Symp.*, 2016, pp. 619–636.
[20] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. IEEE Symp. Security Privacy*, 2017, pp. 19–38.
[21] L. Lyu, X. He, Y. W. Law, and M. Palaniswami, "Privacy-preserving collaborative deep learning with application to human activity recognition," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 1219–1228.
[22] L. Lyu, J. C. Bezdek, X. He, and J. Jin, "Fog-embedded deep learning for the Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 15, no. 7, pp. 4206–4215, Jul. 2019.
[23] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1333–1345, May 2018.
[24] K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1175–1191.
[25] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *Proc. Int. Conf. Learn. Representations*, 2018.
[26] S. Yang, F. Wu, S. Tang, X. Gao, B. Yang, and G. Chen, "On designing data quality-aware truth estimation and surplus sharing method for mobile crowdsensing," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 4, pp. 832–847, Apr. 2017.
[27] S. Gollapudi, K. Kollias, D. Panigrahi, and V. Pliatsika, "Profit sharing and efficiency in utility games," in *Proc. 25th Annu. Eur. Symp. Algorithms*, 2017, pp. 1–16.
[28] A. Richardson, A. Filos-Ratsikas, and B. Faltings, "Rewarding high-quality data via influence functions," 2019, *arXiv:1908.11598*.
[29] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends® Theor. Comput. Sci.*, vol. 9, no. 3/4, pp. 211–407, 2014.
[30] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 308–318.
[31] C. Gentry and D. Boneh, *A Fully Homomorphic Encryption Scheme*, vol. 20. Stanford, CA, USA: Stanford Univ., 2009.
[32] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Proc. Annu. Cryptol. Conf.*, 2012, pp. 643–662.
[33] A. Canteaut *et al.*, "Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression," *J. Cryptology*, vol. 31, no. 3, pp. 885–916, 2018.
[34] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: http://bitcoin.org/bitcoin.pdf
[35] C. Natoli, J. Yu, V. Gramoli, and P. J. E. Verissimo, "Deconstructing blockchains: A comprehensive survey on consensus, membership and structure," 2019, *arXiv:1908.08316*.
[36] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," 2019, *arXiv:1908.09635*.
[37] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," 2020, *arXiv: 2003.02133*.
[38] X. Zhang, S. Ji, and T. Wang, "Differentially private releasing via deep generative model," 2018, *arXiv: 1801.01594*.
[39] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," in *Proc. Int. Conf. Learn. Representations*, 2017.
[40] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 214–223.
[41] C. Castelluccia, E. Mykletun, and G. Tsudik, "Efficient aggregation of encrypted data in wireless sensor networks," in *Proc. 2nd Annu. Int. Conf. Mobile Ubiquitous Syst. Netw. Serv.*, 2005, pp. 109–117.
[42] L. Lyu, K. Nandakumar, B. Rubinstein, J. Jin, J. Bedo, and M. Palaniswami, "PPFA: Privacy preserving fog-enabled aggregation in smart grid," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3733–3744, Aug. 2018.
[43] C. De Canniere and B. Preneel, "Trivium," in *New Stream Cipher Designs*. Berlin, Germany: Springer, 2008, pp. 244–266.
[44] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer, "OpenPGP message format," Tech. Rep. RFC 2440, 1998.

Lingjuan Lyu (Member, IEEE) received the PhD degree from the University of Melbourne, Australia. She is currently a research fellow with the Department of Computer Science, National University of Singapore, Singapore. Her current research interests span machine learning, privacy, fairness, and edge intelligence. Her work was supported by an IBM PhD Fellowship.

Jiangshan Yu received the PhD degree from the University of Birmingham, United Kingdom, in 2016. He is currently associate director (research) at Monash Blockchain Technology Centre, Monash University, Australia. Previously, he was a research associate at SnT, University of Luxembourg, Luxembourg. The focus of his research has been on design and analysis of cryptographic protocols, cryptographic key management, blockchain consensus, and ledger-based applications. In particular, His recent research challenges the soundness of the foundational security models and design principles of existing blockchain systems, where the blockchain ecosystem of hundreds of billions of dollars is based upon. He won numerous prestigious awards, including Dean's Research Impact Award (2019) and the Chinese Government Award for Outstanding Scholar Abroad (1 percent worldwide, 2016).

Karthik Nandakumar (Senior Member, IEEE) received the BE degree from Anna University, Chennai, India, in 2002, the MS degrees in computer science and statistics, in 2005 and 2007, respectively and the PhD degree in computer science from Michigan State University, East Lansing, Michigan, in 2008, and the MSc degree in management of technology from National University of Singapore, Singapore, in 2012. He is a research staff member at IBM Research, Singapore. Prior to joining IBM, in 2014, he was a scientist at Institute for Infocomm Research, A*STAR, Singapore for more than six years. His research interests include computer vision, statistical pattern recognition, biometric authentication, image processing, machine learning, and blockchain.

Yitong Li received the BS degree from Shanghai Jiao Tong University, China. He is currently working toward the PhD degree with the School of Computing and Information Systems, the University of Melbourne, Australia. His research interests cover privacy and adversarial learning with NLP applications. He has publications in ACL, EMNLP, NAACL, etc.

Xingjun Ma received the ME degree from Tsinghua University, China, and the PhD degree from the University of Melbourne, Australia. He is currently a research fellow with the University of Melbourne, Australia. His research interests cover adversarial machine learning and robust supervised/weakly-supervised learning. He has publications in ICML, ICLR, CVPR, IJCAI, AAAI, ICCV, etc.

Jiong Jin (Member, IEEE) received the BE degree with first class honours in computer engineering from Nanyang Technological University, Singapore, in 2006, and the PhD degree in electrical and electronic engineering (EEE) from the University of Melbourne, Australia, in 2011. From 2011 to 2013, he was a research fellow with the Department of EEE, University of Melbourne, Australia. He is currently a senior lecturer with the School of Software and Electrical Engineering, Faculty of Science, Engineering and Technology, Swinburne University of Technology, Melbourne, Australia. His research interests include network design and optimization, edge computing and distributed systems, robotics and automation, and cyber-physical systems and Internet of Things as well as their applications in smart manufacturing, smart transportation, and smart cities.

Han Yu received the BEng (hons) degree and PhD degree from the School of Computer Science and Engineering (SCSE), Nanyang Technological University (NTU), Singapore, in 2007 and 2014, respectively. He is currently a Nanyang assistant professor (NAP) at SCSE, Nanyang Technological University, Singapore. From 2015 to 2018, he held the prestigious Lee Kuan Yew post-doctoral Fellowship (LKY PDF) at the Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY). His research focuses on the ethics of artificial intelligence and federated learning. He co-authored the book "Federated Learning" - the first monograph on the topic of federated learning.

Kee Siong Ng received the PhD degree from the Australian National University, Australia and has more than 15 years of experience in industry and government. He is an associate professor with the newly formed Software Innovation Institute at the Australian National University (ANU), Australia, and one of the first two Translational Fellows appointed through ANU's Entrepreneurial Academic Scheme.