A Robust Game-Theoretical Federated Learning Framework With Joint Differential Privacy

Lefeng Zhang, Tianqing Zhu, Ping Xiong, Wanlei Zhou, *Senior Member, IEEE*, and Philip S. Yu, *Fellow, IEEE*

Abstract—Federated learning is a promising distributed machine learning paradigm that has been playing a significant role in providing privacy-preserving learning solutions. However, alongside all its achievements, there are also limitations. First, traditional frameworks assume that all the clients are voluntary and so will want to participate in training only for improving the model's accuracy. However, in reality, clients usually want to be adequately compensated for the data and resources they will use before participating. Second, today's frameworks do not offer sufficient protection against malicious participants who try to skew a jointly trained model with poisoned updates. To address these concerns, we have developed a more robust federated learning scheme based on joint differential privacy. The framework provides two game-theoretic mechanisms to motivate clients to participate in training. These mechanisms are dominant-strategy truthful, individual rational, and budget-balanced. Further, the influence an adversarial client can have is quantified and restricted, and data privacy is similarly guaranteed in quantitative terms. Experiments with different training models on real-word datasets demonstrate the effectiveness of the proposed approach.

Index Terms—Differential privacy, game theory, federated learning

1 Introduction

The rapidly developing techniques in artificial intelligence (AI) are greatly improving traditional machine learning methods and bringing enormous potential for future innovation along with them. The past few years have witnessed a boom in AI solutions in computer vision, speech recognition, medical diagnosis, and more [1]. However, although computing power has generally increased, so has the amount of data needed to build a high-performing model. As such, it is no longer practical to collect sufficient data and train a model with a single processing unit. Federated learning [2], a promising learning paradigm for distributed settings, is being used to overcome this issue. In federated learning, each client trains a local model using their own data, and updates the model's parameters or gradients to the server. The server aggregates all the parameters from clients and update the global model round by round. In most federated learning schemes, only the model's parameters are shared to the server, hence each client's data are kept invisible and anonymized to other parties and their privacy can be protected to some extent.

Yet, despite these benefits, federated learning still faces several critical challenges. The first and foremost one corresponds to clients' incentive, which is usually overlooked by most federated learning frameworks [3]. Incentive refers to the rewards or gains of clients that encourage them to participate in a training, or the inducement that make them behave actively and reliably in the training process [4]. The incentive issue seems redundant in federated learning as each client has access to the final trained model and actually benefits from it. However, researches has demonstrated that this is far from being enough [5], [6], [7]. In fact, being a part of a federated learning scheme demands a great deal of overhead on the part of the client, and so their excessive consumption of time, energy, bandwidth, etc., may need to be compensated. More importantly, nowadays data are regarded as valuable and private assets, which can rarely be contributed for free. Hence, the traditional assumption that all the clients are voluntary to participate in the federated learning is somewhat over-optimistic. From this perspective, an incentive mechanism is an indispensable element of a federated learning system.

In addition to the incentive problem, malicious participants attempting to poison the jointly learned model is another issue. Researchers have demonstrated that federated learning is generally vulnerable to manipulation by adversarial clients, like model and data poisoning [8], [9] in particular. In this paper, we focused on dirty-label data poisoning attack, where a malicious participant introduces a number of data samples with modified labels in the training set to cause a mis-classification of the global model. As

- *Lefeng Zhang and Tianqing Zhu are with the Centre for Cyber Security and Privacy and the School of Computer Science, University of Technology, Sydney, NSW 2007, Australia. E-mail: lefeng.e.zhang@gmail.com, Tianqing.Zhu@uts.edu.au.*
- *Ping Xiong is with the Zhongnan University of Economics and Law, Wuhan 430073, China. E-mail: pingxiong@zuel.edu.cn.*
- *Wanlei Zhou is with the City University of Macau, Macau, China. E-mail: wlzhou@cityu.mo.*
- *Philip S. Yu is with the Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607 USA. E-mail: Psyu@uic.edu.*

Manuscript received 31 Aug. 2021; revised 7 Dec. 2021; accepted 23 Dec. 2021. Date of publication 4 Jan. 2022; date of current version 7 Mar. 2023.
This work was supported in part by ARC Discovery Project under Grants DP190100981 and DP200100946 from the Australian Research Council, and in part by NSF under Grants III-1763325, III-1909323, III-2106758, and SaTC-1930941.
(Corresponding author: Tianqing Zhu.)
Recommended for acceptance by W. Zhang.
Digital Object Identifier no. 10.1109/TKDE.2021.3140131

clients' data are never shared in federated learning, the server cannot verify the data used in each local training. Thus, dirty-label data poisoning attack provides a convenient way for malicious clients to manipulate the global model. Therefore, framework designs need to be more robust and defense strategies need to be more reliable.

As a last point, federated learning works without disclosing the clients' raw data. However, their private information can still be compromised by inferring information from the parameter updates [10]. Hence, an optimal security strategy against poisoning attacks should also provide protection against privacy breaches through inference.

In this paper, we proposed a game-theoretic framework to advance federated learning, aiming at guiding client selection and limiting the influence of adversaries. The framework not only introduces incentives and compensation for participation, but also improves defenses against poisoning attacks. Specifically, our solutions treat clients as purveyors of training power, and the server is wired to behave like a buyer. Various allocation and payment rules are designed to guarantee truthfulness and individual rationality during the "buying" process, i.e., the process of selecting clients to participate in training. Further, should a particular client be adversarial, the mechanism applies joint differential privacy to limit its impact. To demonstrate the effectiveness of this framework, we conducted experiments with several real-world datasets and different learning models. The main contributions of this paper therefore include:

- A jointly differentially private federated learning framework that limits the impact of adversarial clients and provides a rigorous privacy guarantee.
- Two truthful game-theoretic mechanisms the server can use to select participating clients for training.
- Theoretical proofs and analyses of the proposed mechanisms. The client selection mechanisms satisfy game-theoretic properties and are robust against strategic manipulation by clients.
- Experimental simulations that show the the rationale behind the mechanisms. Further experiments with real-world datasets demonstrate the effectiveness of the proposed federated learning scheme.

The rest of the paper is organized as follows. In Section 2, we review the literature on differentially-private federated learning and differentially-private mechanism design. Section 3 introduces some necessary background knowledge and formally defines the problems to be addressed. In Section 4, we detail the client selection mechanisms and learning framework. Section 5 presents theoretical analyses of the game-theoretic properties and the differential privacy guarantee. The experimental results and related discussions are given in Section 6, and Section 7 concludes this paper.

2 Related Work

2.1 Review of Related Work

2.1.1 Differential Privacy in Federated Learning

Federated learning is a promising distributed machine learning framework, where the clients train a model locally using their own private data and then submit the parameters to a central server for aggregation. However, although federated learning protects data privacy to an extent, clients' information can still be inferred from the parameter updates [8], [10]. Differential privacy, as a rigorous privacy model, has been widely used to address this issue.

Wei *et al.* [11] used a Gaussian mechanism to perturb the actual parameters client-side before sending them to the server for aggregation. They calibrated the appropriate levels of noise for uploading and broadcasting, and theoretically analyzed the convergence bound with respect to the level of privacy protection offered. The strategy is robust to adversaries who eavesdrop between the server and clients.

Hu *et al.* [12] presented a differentially private federated learning framework to train personalized models for users in distributed IoT architectures. They formulated federated learning as a multitask optimization problem, using a Gaussian mechanism to protect the information in dual variables. The trained model takes advantage of each user's data so as to maintain a high level of personalization.

Hao *et al.* [13] developed a privacy-preserving deep federated learning scheme based on gradient descent. To protect the parameter update process, they combined differential privacy with homomorphic encryption. Each local gradient is randomized by adding Laplace noise and then encrypted using a Pallier algorithm. Integrating encryption techniques stops collusion between the server and users.

Zhao *et al.* [14] explored the notion that unreliable participants who hold low-quality data and may negatively impact learning accuracy. Hence, they employed an exponential mechanism to select high-quality participants, and applied functional mechanism expand the loss function. To protect privacy, Laplace noise is added to the coefficients of the approximated polynomial formula.

2.1.2 Differential Privacy and Game Theory

Game theory involves strategic decision-making by rational players. As a rigorous model, differential privacy provides many desirable properties for solving game-theoretic problems, such as truthfulness and stability [15], [16].

Li *et al.* [17] adopted differential privacy in the smart grid response system, presenting a private double auction mechanism for allocating energy resources. The electric vehicles act as both buyers and sellers, while a microgrid control center decides the winners and corresponding payments. A Gaussian mechanism protects bidders' valuations, and the valid price is computed based on noisy data.

Lin *et al.* [18] implemented an exponential mechanism in a mobile crowdsensing system to select the winning bidders while protecting users' bidding privacy. They modeled the interactions between crowdsensing platforms and mobile users as a reverse auction, proposing both single-bid and multi-bid frameworks for different crowdsensing scenarios. They also designed linear and logarithmic score functions to achieve near-optimal social cost minimization.

Ye *et al.* [19] developed a differentially private cyber deception game that misleads attackers by providing inaccurate system information. In their game, the number of systems is perturbed by adding Laplace noise [20], while the system configurations are obfuscated based on the exponential mechanism [21]. Their model reduces the influence

of the number of systems on network security and limits the impact of inference attacks from rational attackers.

With some games, implementing classic full differential privacy is too rigorous to allow for a non-trivial solution [22]. A relaxed privacy definition, called joint differential privacy, has been developed to suit these situations.

Hsu *et al.* [23] first proposed joint differential privacy to privately compute a Walrasian equilibrium. They employed differentially private counters to indicate the prices during the allocation so that agents need only look at the number of counts to figure out their bidding status. The counters increase in an ascending manner, which means agents adjust their bidding strategy by inferring the general prices of others in conjunction with their own valuations.

Huang *et al.* [24] studied the behavior of non-myopic bidders in repeated auctions. They proposed jointly differentially private learning algorithms against non-myopic bidders by increasing the cost of deviation. In this study, they employed a differentially private tree-aggregation scheme [25] to record the noisy cumulated gains of each price. Joint differential privacy controls the information revealed to each bidder, and thus further limit agents' influences in future round auctions.

2.2 Discussion of Related Works

The reviewed works have two common limitations. First, they do not incentivize clients to participate and do not consider the monetary budget constraint of the server. Second, their game-theoretic models do not involve adversarial clients within the system, who may behave strategically to influence the performance of the trained model.

In terms of the lack of client incentives, these differentially private methods are based on the common assumption that all clients are or want to be involved in the training process. However, this is not true in practice [26], especially when the collaboratively trained model is commercialized in markets [3]. Without satisfactory compensation for their data and resource use, many clients are not willing to participate in federated learning. Besides, the server may only have a limited monetary budget with which to select participants. The question of how to best use server resources to select clients is a topic worthy of further discussion.

As for privacy and security, in existing game-theoretic models, differential privacy is used either to protect a client's private information or to stabilize the outcome of the model. No differentially private solution considers an outcome where a client is the adversary. Nor do these solutions provide an adequate defense strategy to stop an adversarial client from negatively impacting the training.

To this end, we developed two dominant-strategy truthful mechanisms for the server to select participating clients – mechanisms that foil attempts at strategic manipulation and provide a privacy guarantee over the clients' data. At the same time, we considered situations where an adversarial client has been selected to participate and designed a jointly differentially private framework to limit that client's impact. We focused on only one malicious client in this paper. On the one hand, one malicious client can cause significant impact on the global model's performance [9], it is not trivial to defend against one client's malicious manipulation. On the other hand, the joint differential privacy actually limits one single individual's influence on the mechanism's output. To defend against multiple malicious clients using differential privacy, more relaxed definitions should be proposed and new differentially private mechanisms should be designed accordingly. We leave the exploration of multi malicious client scenario in our future work.

3 Preliminaries

3.1 Federated Learning

Consider a general federated learning framework consisting of one server S and n clients $\{C_1, C_2, \ldots, C_n\}$. Each client C_i possesses a certain amount of data, denoted as D_i, and is equipped with a computing device. The server S wishes to learn a global model with the data distributed across these clients. A client participating in the training process must find the parameter θ that minimizes a given loss function. The loss function for client C_i with data $d_i = |D_i|$ is

$$F_i(\theta) = \frac{1}{d_i} \sum_{j \in D_i} f_j(\theta),$$

where $f_j(\theta)$ is the loss function for the data sample j. The server aggregates the model parameters θ_j from each client by computing their weighted average

$$\theta = \sum_{i=1}^{n} \frac{d_i}{d} \theta_i, \quad d = \sum_{i=1}^{n} d_i.$$

The server broadcasts aggregated model parameters to each client, who updates their local model and starts the next round of training. The learning process is therefore formulated as an optimization problem, aimed at finding the minimizer of the global loss function

$$\theta^* = \arg\min_{\theta} \frac{d_i}{d} F_i(\theta).$$

Clients iteratively learn a machine leaning model according to a schedule set by the server. After enough rounds of training and parameter aggregation, the solution to the optimization problem θ^* converges to the optimal value globally. Because the server S aggregates the local parameters, clients do not need to communicate with each other.

3.2 Game Theory

Game theory is a study of mathematical models of conflict and cooperation between intelligent rational decision-makers [27]. Game-theoretic solutions have been widely applied across many fields of artificial intelligence [28], [29].

In federated learning, the server needs to recruit a number of clients to perform a training task. However, given that the clients each have a different dataset and diverse computation abilities, the energy and time costs etc. for them to participate in the training also varies. Hence, our framework treats the client selection process as an auction game, where clients submit their cost c_i as a bid and the server decides the winners and payments p_i based on its budget B. If a client is selected by the server, its utility is $u_i = p_i - c_i$, and $u_i = 0$ otherwise.

Clients are assumed to be rational and self-interested. In other words, each client only cares about its own utility and tries to maximize it, even if that means misreporting information. However, the server is incentivized to attract as many participating clients as possible with its limited budget B. Specifically, the client selection mechanism should satisfy the following game-theoretic properties:

Definition 1. *(Dominant-strategy truthful) A client selection mechanism $\mathcal{M}$ is dominant-strategy truthful if, for every client, truthfully reporting his cost c_i is a dominant strategy. That is, given c_i and all possible $c_i' \in \mathbb{R}$*

$$u_i(c_i, c_{-i}) \geq u_i(c_i', c_{-i}), \tag{1}$$

where c_{-i} represents the strategy profile of clients except for the ith component, $c = (c_i, c_{-i})$.

Truthfulness means that no client can improve their utility by misrepresenting their data, which is an important property in developing game-theoretic solutions.

Definition 2. *(Individual rationality) A mechanism $\mathcal{M}$ is individual rational for each client if the clients gain non-negative utility by participating in $\mathcal{M}$*

$$u_i = p_i - c_i \geq 0. \tag{2}$$

Individual rationality guarantees the baseline utility of each client. If participating in a mechanism possibly leads to negative utility, clients may decline to participate at all.

Lastly, from the server's perspective, its total payments must stay within its predefined budget.

Definition 3. *(Budget balance) A client selection mechanism $\mathcal{M}$ is budget-balanced if the total amount paid by the server does not exceed its budget. Formally, given m selected clients, each receives a payment p_i, then*

$$\sum_{i=1}^{m} p_i \leq B. \tag{3}$$

3.3 Differential Privacy

Differential privacy is a rigorous mathematical privacy model that limits the influence of each individual on an aggregated output. It has been widely explored in fields like machine learning, data publishing, and mechanism design [30]. Recently, differential privacy has also been proven to provide useful game-theoretic properties in AI fields [31].

The aim of differential privacy is to bound the differences in a query between neighboring datasets. Two datasets D and D' are defined as neighboring datasets if they are identical except for one record. A query function $f : D \to \mathcal{R}$ maps dataset D to an output range $\mathbb{R}$. A mechanism $\mathcal{M}$ satisfies differential privacy if, for any pair of datasets, the probability distribution of $f(D)$ and $f(D')$ on the output space are almost identical. The formal definition of differential privacy is as follows:

Definition 4. *(Differential privacy [20]) A randomized mechanism $\mathcal{M}$ gives (ϵ, δ)-differential privacy for any pair of neighboring datasets D and D', and for every possible set of outcome $S \subseteq \mathcal{R}$, if $\mathcal{M}$ satisfies*

$$\Pr[\mathcal{M}(D) \in S] \leq \exp(\epsilon) \cdot \Pr[\mathcal{M}(D') \in S] + \delta, \tag{4}$$

where the parameter ϵ refers to the privacy budget.

Sensitivity captures the maximum difference f may have on D and D', which, in turn, determines the magnitude of the perturbation required to privately answering a query.

Definition 5. *(l_2-sensitivity [20]) Given a pair of neighboring datasets D and D', the l_2-sensitivity of a query $f : D \to \mathcal{R}^n$ is defined as*

$$\Delta f = \max_{D,D'} ||f(D) - f(D')||_2. \tag{5}$$

One of the most widely used differentially private mechanisms for numeric query is the Gaussian mechanism, which adds noise following a Gaussian distribution to the results of a numeric query.

Definition 6. *(Gaussian Mechanism [32]) Given any function $f : D \to \mathcal{R}^n$ over a dataset D, the Gaussian mechanism, which adds Gaussian distributed noise to each coordinate of the output, provides an (ϵ, δ)-differential privacy guarantee. The noise is scaled to $\mathcal{N}(0, \sigma^2)$ with the parameter σ satisfying*

$$\sigma \geq \sqrt{2 \ln \frac{1.25}{\delta}} \cdot \frac{\Delta_2 f}{\epsilon}. \tag{6}$$

For non-numeric queries, the exponential mechanism is designed to provide differential privacy guarantee, paired with an application-dependent score function $y(D, r)$.

Definition 7. *(Exponential mechanism [21]) Let $y(D, r)$ be a score function that measures the quality of output $r \in \mathcal{R}$ over dataset D. The exponential mechanism $\mathcal{M}$ satisfies ϵ-differential privacy if*

$$\mathcal{M} = \left\{ \text{return } r \text{ with probability} \propto \frac{\epsilon y(D, r)}{2\Delta y} \right\}. \tag{7}$$

Joint differential privacy is a relaxation of classic full differential privacy. It guarantees that the joint distribution of an outcome given to all other clients $j \neq i$ will be insensitive to client i's reported data.

Definition 8. *(Joint Differential Privacy [23]) A randomized method $\mathcal{M}$ gives (ϵ, δ)-joint differential privacy for any pair of neighboring datasets D and D', and for every possible subset of outcome $S_{-i} \subseteq \mathcal{R}^{n-1}$, if $\mathcal{M}$ satisfies*

$$\Pr[\mathcal{M}(D)_{-i} \in S_{-i}] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(D')_{-i} \in S_{-i}] + \delta. \tag{8}$$

An important connection between classic differential privacy and joint differential privacy is the billboard lemma. The billboard lemma states that if each client's output is a function only of an ϵ-differentially private computation and their own private data, then the overall mechanism satisfies ϵ-joint differential privacy.

Lemma 1. *(*Billboard Lemma *[33]) Suppose $\mathcal{M} : D \to \mathcal{R}$ is (ϵ, δ)-differentially private. Consider any set of functions $f_i : D_i \times \mathcal{R} \to \mathcal{R}'$, where D_i is the portion of the database containing i's data, then the mechanism $\mathcal{M}'$ that outputs to each client i: $f_i(D_i, \mathcal{M}(D))$ is (ϵ, δ)-jointly differentially private.*

TABLE 1
The Main Notations Used in the Paper

Notations	Explanation
n	the number of clients
m	the number of server selected clients
ϵ_L, ϵ_E	privacy budgets
c_{-i}	vector c except for the ith component
Δf	the sensitivity of a query
C_i	the ith client
θ_i	the model parameters trained by client C_i
p_i, c_i	the payment, cost of client C_i
u_i	the utility of client C_i
d_i	the amount of training data used by client C_i
(agg, sum)	the aggregated parameters
g_i	the partitioned groups of clients
y	the score function in an exponential mechanism
B	the server's monetary budget
∇f	the gradient vector of function f
$\mathcal{N}(0, \sigma)$	the Gaussian distribution with mean 0 and std. σ

3.4 Problem Definition

In this paper, we address three issues in federated learning. First, we need to find a dominant-strategy truthful client selection mechanism, such that the server can select participating clients while minimizing the risk of strategic manipulation. Second, we need to consider situations where one adversarial client has "slipped through the cracks" and been selected to participate in the training: we need to find a way to limit the impact of this adversarial client. Third and last, the solutions we derive must protect each client's data privacy. Formal definitions of these problems follow, with the main notations used summarized in Table 1.

Consider a server S that wishes to recruit m out of n clients to collaboratively train a global model. Each client C_i has a database D_i that contains $d_i = |D_i|$ data records. The server S publishes the training task and each client submits their estimated costs c_i, which is the minimum amount of payment that incentivizes them to participate in the training. Given a cost vector $(c_i, c_2, \ldots, c_n)$, the server must select m participating clients while satisfying the game-theoretic properties outlined in Definitions 1-3. Moreover, the m clients selected include one adversarial client, who will try to skew the trained model by submitting dirty parameters during the training process. The server cannot distinguish which client is adversarial but wishes to reduce its impact as much as possible.

For the purposes of this paper, we assume that the server S is not trustworthy. As a result, no client can submit their trained parameters directly to the server for fear of a privacy breach. Moreover, we also assume that the server knows the amount of data d_i used by each client. This assumption is without loss of generality because in many federated learning scenarios d_i is not regarded as private information. For example, in a training of default prediction model, the central authority (e.g., bank regulator) actually knows the amount of registered members in each organization (e.g., banks). And, many input recommendation models in mobile phone are locally trained based on a length-fixed prefixes of user's previous inputs (e.g., 5 words). Even if d_i cannot be known, we can simply set $d_i = 1$, which removes d_i's influence in the proposed mechanisms. As d_i is not involved in the derivation of game-theoretic properties, the proposed mechanisms still work without knowing an actual d_i.

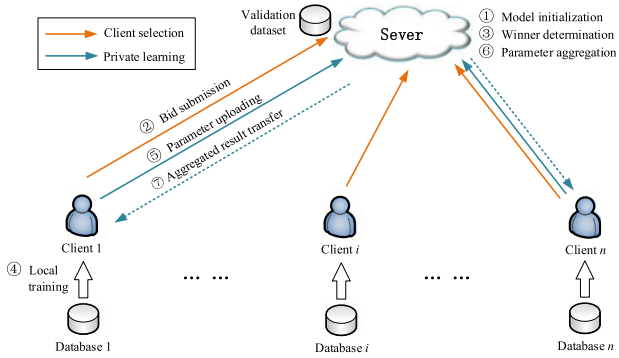


Fig. 1. Our jointly differentially private federated learning framework.

4 A Jointly Differentially-Private Federated Learning Mechanism

4.1 Overview of the Mechanism

The proposed framework consists of a central server and multiple clients, as pictured in Fig. 1.

We provide an example to describe the workflow of the proposed framework. Consider there are $n = 15$ clients $\{C_1, C_2, \ldots, C_{15}\}$, each with their own private database that might be used to collaboratively train a machine learning model. A central server S is responsible for selecting participating clients and aggregating clients' submitted parameter updates. To arrive at a global model, each party in the framework must execute the following steps in succession.

① *Model initialization.* The server S initializes the model that is to be trained and publishes necessary information about this model to all clients, e.g., the number of parameters, the network structure, etc.

② *Bid submission.* Each client C_i receives the model's information from the server, and evaluates their cost c_i of participating in the training. The cost can be evaluated by considering the energy consumption, communication cost and the valuation of their training dataset [6], [34]. Each client then submits their evaluated costs c_i as a bid to the server S.

③ *Winner determination.* The server S employs a dominant-strategy truthful mechanism to decide the winning clients and corresponding payments. It then publishes the result of the bidding process. The winning clients then prepare for the coming local training (e.g., Client 1 in Fig. 1). Clients that were not selected are not included in any further training. For the purposes of this example, we assume that clients $\{C_1, C_2, \ldots, C_{10}\}$ are winners selected by the server.

④ *Local training.* Each winner downloads the global model from the server S and performs local training using their own device and data D_i. In each round, the locally trained model parameters of client C_i are denoted as θ_i.

⑤ *Parameter uploading.* The winners add local perturbation to their parameters θ_i, and then upload the perturbed parameters $\hat{\theta}_i$ to the server S according to

predefined uploading rules. Each θ_i is perturbed with Gaussian noise $v_i \sim \mathcal{N}(0, \sigma)$, i.e., $\hat{\theta}_i = \theta_i + v_i$.

⑥ *Parameter aggregation.* The server constructs client groups $g_i = \{C_j\}_{j \neq i}$, each contains all the winning clients except for a particular one. The server then employs an exponential mechanism to select a group g_i^*, and computes $(agg = \sum d_i \hat{\theta}_i, sum = \sum d_i)$ for $C_i \in g_i^*$. The probability of being selected is determined using the model's accuracy evaluated on the server's validation dataset. For example, if $g_1 = \{C_2, \ldots, C_{10}\}$ is selected by the mechanism, then the aggregated parameters will be $(agg = \sum_{i=2}^{10} d_i \hat{\theta}_i, sum = \sum_{i=2}^{10} d_i)$.

⑦ *Aggregated result transfer.* The server S transfers the aggregated parameters (agg, sum) to each client. The clients update their local model by $\theta_i = \frac{agg + d_i \theta_i}{sum + d_i}$, and return to Step ④ for the next round of training. For example, Client 1 would update its local parameters with the formula $\theta_1 = \frac{agg + d_1 \theta_1}{sum + d_1}$.

After each round, the server S tests the accuracy of the global model using the newest θ_i selected by the exponential mechanism. Once the accuracy of the global model reaches a satisfactory threshold, the server stops the training. Each client will receive their payment after the training stops.

The proposed framework involves an auction process compared to traditional ones. Suppose there are n candidate clients participating in the bidding, each client submits their evaluated cost to the server, and the server publishes the result after determining winners. During the auction process, the communication overhead is $O(n)$.

We note that, non-myopic rational adversaries may bear the loss incurred in the client selection process and strategically underbid their cost to make themselves selected. This happens when an adversary's expected future gain by attack outweighs its previous loss in client selection. Such scenarios are complicated to analyze because it is hard to quantify the actual gain of an adversary without introducing more assumptions about their (may be heterogeneous) utility function. To address this issue, we introduced joint differential privacy in the private learning stage, which limits the influence of an adversary.

4.2 The Client Selection Stage

The keys to the client selection stage (i.e., Steps ②∼③) are the client selection mechanisms, which are designed to choose appropriate clients while incentivizing truthful reporting. As the proposed framework is built for the possibility that an adversarial client may be a participant, both normal and adversarial clients are considered.

- *Normal clients.* Normal clients are those benign clients who participate in the training without any intention of sabotaging the trained model. These clients are rational and wish to maximize their utility u_i. Therefore, to get a higher payment, a normal client may strategically misreport their expected costs c_i to the server S either by over- or underbidding the amount. In addition, normal clients will follow the predefined rules about parameter uploading, and they care about their privacy.
- *Adversarial clients.* Adversarial clients want to sabotage the trained model in some way. For example, they may upload random noise to reduce the model's accuracy [35], or train their local model with incorrectly labeled data so as to skew the global update [9]. Like normal clients, adversarial clients can also strategically overbid or underbid their costs c_i to obtain higher payments. However, it is generally difficult to determine how much they will benefit from misleading a global model in this way. As a result, we have not included a specific representation of utility for adversarial clients.

Importantly, the server cannot distinguish between the two types of clients. Thus, the client selection mechanisms must be dominant-strategy truthful to avoid strategical manipulation. We present two mechanisms that satisfies this requirement. The first mechanism is dominant-strategy truthful, individual rational and envy-free, which means that each client likes its outcome at least as well as the outcome of anyone else. The second mechanism is also dominant-strategy truthful and individual rational, besides, it satisfies the server's specific optimization requirements for selecting clients. In addition, both mechanisms are budget-balanced.

4.2.1 A Simple, Envy-Free Client Selection Mechanism

The aim of the client selection mechanism $\mathcal{M}_1$ is to guarantee game-theoretic properties, such as truthfulness and individual rationality. After collecting the costs c_i from each client i, the server runs Algorithm 1 to decide the winners and corresponding payments. The algorithm first computes the unit prices of each piece of data $q_i = c_i/d_i$ at client i, and sorts them into increasing order. Then it finds the largest integer $m \in [n]$ that satisfies $q_m \leq \frac{B}{\sum_{i=1}^{m} d_i}$. The winning clients are C_i, $i \in [m]$ and their payments are determined by $p_i = \min\{\frac{B}{\sum_{i=1}^{m} d_i}, q_{m+1}\} \cdot d_i$. Any clients who are not selected to participate receives zero payment, i.e., $p_i = 0$ for $i \in [m+1, n]$, and are no longer considered.

Algorithm 1. Truthful Client Selection Mechanism $\mathcal{M}_1$

Input: cost c_i from each client, the server's total budget B.
Output: winning clients for FL training, the payment p_i for each client.

for $i \leq n$ **do**
  $q_i \leftarrow \frac{c_i}{d_i}$.
end for
sort q_i in an increasing order, $q_1 \leq q_2 \leq \cdots \leq q_n$, breaking ties arbitrarily.
find the largest $m \in [n]$ that satisfies $q_1 \leq q_2 \leq \cdots \leq q_m$ and $q_m \leq \frac{B}{\sum_{i \in S} d_i}$, breaking ties arbitrarily.
for $1 \leq i \leq m$ **do**
  $p_i \leftarrow \min\{\frac{B}{\sum_{i \in S} d_i}, q_{m+1}\} \cdot d_i$.
end for
for $m < i \leq n$ **do**
  $p_i \leftarrow 0$.
end for

The design rationale of $\mathcal{M}_1$ is as follows. What makes a client attractive to a server is low costs and a large amount of data offered. Therefore, the heuristic trades off between

these two properties by considering the data's unit price. In other words, the server S *buys* the *goods* (data) from the *sellers* (clients) according to the unit *selling price* of their goods. The detailed game-theoretic properties of the mechanism are proofed and discussed in Section 5.

4.2.2 Client Selection With the Server's Requirement

Although mechanism $\mathcal{M}_1$ satisfies the desired game-theoretic properties, it does not consider the server's specific requirements. To treat server S as a data buyer in a market, we must assume a specific goal that is driving its data purchases. For example, the server may wish to reach an accuracy threshold with the minimum amount of spending, or to achieve the highest possible accuracy with a fixed monetary budget [34].

Algorithm 2. Truthful Client Selection Mechanism $\mathcal{M}_2$

Input: cost c_i from each client, the server's total budget B.
Output: winning clients for FL training, the payment p_i for each client.

for $i \leq n$ **do**
  $r_i \leftarrow \frac{d_i}{c_i}$.
end for
sort r_i in an decreasing order, $r_1 \geq r_2 \geq \cdots \geq r_n$, breaking ties arbitrarily.
$paid \leftarrow 0$, $selected \leftarrow \emptyset$, $i = 1$.
while $paid + c_i \leq B$ **do**
  $selected \leftarrow selected \cup \{i\}$.
  $paid \leftarrow paid + c_i$.
  $i \leftarrow i + 1$.
end while
if $\sum_{j \in selected} d_j \geq d_{j+1}$ **then**
  Output $selected$.
else
  Output $\{i^*\}$ that satisfies $i^* = \arg\max_i d_i$.
end if
for $i \in selected$ **do**
  $p_i \leftarrow \int_0^{c_i} z \cdot \frac{\mathrm{d}}{\mathrm{d}z} alloc_i(z, c_{-i}) \mathrm{d}z$.
end for

However, in our framework, the server S only knows the amount of data each client holds, it cannot assess the quality of the data. From the server's perspective, guaranteeing the performance of the trained model demands as much training data as it can possibly buy with its money. As such, it is reasonable to formulate the client selection problem as a social welfare maximization problem, where the social welfare is defined as the total amount of data the server can possibly buy. Formally, the problem is described as follows:

$$\max \sum_{i=1}^{n} x_i \cdot d_i$$
$$\text{s.t. } \sum_{i=1}^{n} x_i \cdot c_i \leq B, \ \forall i, x_i \in \{0, 1\}. \tag{9}$$

This formulation is a typical $0/1$ knapsack problem, where d_i represents each client's weight, and c_i is the corresponding size. The client selection process is therefore a variant of the knapsack auction, where the size is the private information of bidders, and the weights are public.

Knapsack problems are NP-hard; therefore, an approximation algorithm [36] is used to find winning clients, and the corresponding payments are determined following Myerson's lemma [37]. The details of the mechanism are shown in Algorithm 2.

The design rationale of $\mathcal{M}_2$ is as follows. The algorithm approximately solves the knapsack problem and decides the winning clients. The payment p_i for winner i is then determined using the critical bid – the infimum of the bids i could make and continue to win. The allocation function $alloc_i$ describes the allocation status of client i (i.e., win or lose) given the other clients' bids are fixed. The game-theoretic properties of the mechanism are discussed in Section 5.

The difference between mechanisms $\mathcal{M}_1$ and $\mathcal{M}_2$ follows. Mechanism $\mathcal{M}_1$ is easier to compute. It allows the server to decide winning clients by simply sorting the unit prices of the received bids, and the desired game-theoretic properties are satisfied without extra argumentation. In comparison, mechanism $\mathcal{M}_2$ is based on an optimization problem, which may be hard to solve, e.g., NP-hard problems. Moreover, the incentive compatibility of mechanism $\mathcal{M}_2$ also needs further proof, i.e., the monotonicity of client selection rule. Generally, $\mathcal{M}_2$ leads to more computational cost of the server. In practice, the choice of the mechanism depends on the server's requirements.

Remark 1. In mechanism $\mathcal{M}_2$, the heuristic used to solve the knapsack problem is a typical 2-approximation algorithm. Therefore, according to the Myerson's lemma, we can deduce a truthful polynomial time mechanism that achieves at least 50% of the optimal social welfare. Note that there are advanced approximation algorithms that can achieve a higher social welfare while still producing a monotone allocation rule. For example, the truthful polynomial time approximation scheme (PTAS) indeed produces at least a $(1-\beta)$-fraction of the optimal social welfare and runs in time polynomial in n and $1/\beta$ [38]. But we do not discuss them further. It suffices to provide a feasible solution here.

Remark 2. In the client selection stage, maximizing the amount of training data need not necessarily be a requirement for the server. This federated learning framework supports various server requirements. For example, the server may want to involve more clients in the training to increase diversity in the data sources. With different requirements, the optimization problem will also vary. In fact, the optimization problem itself does not influence the game-theoretic properties. What really matter is whether the (approximate) solution of the optimization problem leads to a monotone client selection rule [37]. The Myerson's lemma works as a linkage between the optimization solution and game-theoretic properties. Therefore, the solution of the optimization problem should be carefully designed.

4.3 The Private Learning Stage

The private learning stage spans Steps ④∼⑦ of the proposed framework. The following explanation is based on the same assumption that one of the clients participating in the training is adversarial. Intuitively, the ideal way to deal

with an adversarial client is to simply pick it out and throw it away. However, as the server cannot judge which clients are adversarial and which are not, the second-best method is to limit the impact that an adversary can have on the global model. Differential privacy, which limits the influence of any individual on the final aggregated output, is a useful tool to achieve this goal.

Unfortunately, traditional full differential privacy imposes its bounds over the whole client set, which would here make the aggregated parameters almost independent of each client's submission. This would impair the contribution of high-caliber clients. As a result, the relaxed definition offered by joint differential privacy better suits our needs. The rest of this section outlines the process of guaranteeing joint differential privacy during parameter aggregation process.

In Step ④, the winning clients download the global model with the initialized parameters from the server S, and perform local training using their own data and device. After a certain number of local iterations, each client uploads its locally trained parameters to the server S for aggregation. However, as the parameter set still contains sensitive information [8], [10], these data must be sanitized before transmission.

Hence, in Step ⑤, each client uses mechanism $\mathcal{M}_3$ to perturb their parameters with randomized noises drawn from Gaussian distribution. The noise is calibrated to a sensitivity of $\Delta_2 f$ and a privacy budget ϵ. The actual value of $\Delta_2 f$ depends on specific training model. For example, with linear regression, the sensitivity would be derived using the support bound and the norms of the parameters [39]. Once sanitized, the parameters $\hat{\theta}_i$ are sent to the server.

Algorithm 3. Local Perturbation Mechanism $\mathcal{M}_3$

Input: the private data θ_i of client i, privacy budget ϵ_L, sensitivity $\Delta_2 f$, clipping threshold C.
Output: perturbed data $\hat{\theta}_i$.
1: sample $Z_i \sim \mathcal{N}(0, \sigma)$, where $\sigma = \sqrt{2 \ln \frac{1.25}{\delta}} \cdot \frac{\Delta_2 f}{\epsilon_L}$
2: $\theta_i \leftarrow \theta_i / \max\{1, \frac{\theta_i}{C}\}$.
3: $\hat{\theta}_i \leftarrow \theta_i + Z_i$.
4: **output** $\hat{\theta}_i$.

In Step ⑥, the server S receives the noisy parameters and uses the parameter aggregation mechanism $\mathcal{M}_4$ to produce feedback for each client. The most widely used parameter aggregation method in traditional federated learning frameworks is a weighted average with respect to the client's data, specifically, $\theta = \sum_{i=1}^{m} d_i \hat{\theta}_i / \sum_{i=1}^{m} d_i$. However, this approach incorporates an adversarial client's bad updates alongside all the normal clients' good updates. Hence, to reduce this negative influence and achieve joint differential privacy, the server evaluates the performance of parameters following a leave-one-out strategy, where an exponential mechanism is employed to pick up a high-quality output. Algorithm 4 shows the details.

After receiving the noisy parameters, the server S first constructs m client groups, each contains $m-1$ different clients. The server then computes the weighted average of parameters within each group, and employs an exponential mechanism to select an aggregated result as final output. The score function is the model's accuracy evaluated on the server's validation dataset, namely $y = Acc_D(\hat{\theta}_i)$. The score function represents the goodness of aggregated parameters from each client group. We note that the accuracy is not the only way to measure the goodness of clients' updates, there are also important factors, such as the number of data used in training, that can be adopted in the score function to further weight the importance of each group. We use the model accuracy here for simplicity. The sensitivity of the score function is $\Delta y = 1/(m-1)$ because the model's accuracy is within interval [0,1]. In this way, the adversarial client can only influence the output of parameter aggregation *with some probability*, which is related to the quality of parameters it submits. Moreover, although the adversarial client's current submission could be selected in an aggregation, there is still a chance that its next submission be rejected in the next round. The negative impact of the adversarial client's submission is rigorously limited by differential privacy.

Algorithm 4. Parameter Aggregation Mechanism $\mathcal{M}_4$

Input: the noisy data $\hat{\theta}_i$ of each client, privacy budget ϵ_E
Output: aggregated data (agg, sum) for each client
1: **construct** m client groups $g_i = \{Cj\}_{j \neq i}$.
2: **for** each group g_i **do**
3: $agg_i \leftarrow \sum_{j \in g_i} d_j \hat{\theta}_j$, $sum_i \leftarrow \sum_{j \in g_i} d_j$.
4: $\hat{\theta}_i = agg_i / sum_i$.
5: **end for**
6: **for** $1 \leq i \leq m$ **do**
7: **compute** $y \leftarrow Acc_D(\hat{\theta}_i)$.
8: **end for**
9: $\Delta y \leftarrow \frac{1}{m-1}$.
10: **pick up** a g_i^* with probability $\propto \frac{\epsilon_E \cdot y(D,g)}{2\Delta y}$.
11: **for** $1 \leq i \leq m$ **do**
12: **send** client C_i the aggregated data pair (agg_{i^*}, sum_{i^*}) computed from g_i^*.
13: **end for**

In Step ⑦, each client $i \in [m]$ receives the aggregated parameter (agg, sum) from the server S. The data pair needs to be further processed to satisfy joint differential privacy. Specifically, client i first computes $d_i \theta_i$ using its true data θ_i, and then combines the private $d_i \theta_i$ with the public data pair to update its local parameters, as shown in Algorithm 5. Once combined, each client can test the new parameters with their local data, and continue the training process to for the next round of updates.

The intuition behind the private training stage is as follows. According to the billboard lemma (Lemma 1), to achieve joint differential privacy, we need to generate a differentially private output as signal. Then, if each client's result is only a function of the signal and their own data, the overall algorithm is jointly differentially private. Here, the data pair (agg, sum) selected by the exponential mechanism is treated as signal, and each client's parameter is updated by combining the agg and their true data θ_i. Therefore, the new parameters of all the clients together satisfy joint differential privacy – namely, no matter what data a client reports, the joint distribution of the other clients' new parameters is almost the same. A complete proof of the privacy guarantee is presented in Section 5.

Algorithm 5. Jointly Differentially Private Local Update $\mathcal{M}_5$

Input: the data pair (agg, sum), C_i's true parameters θ_i
Output: jointly differentially private parameter for client i
1: $\theta_i \leftarrow \frac{agg+d_i\theta_i'}{sum+d_i}$.
2: **test** the new parameters θ_i using client i's local dataset.
3: **generate** the parameters for next round of submission.

5 Theoretical Analysis

Our theoretical analysis consists of two parts: a game-theoretic analysis of the client selection strategies (namely $\mathcal{M}_1$ and $\mathcal{M}_2$) and a privacy analysis of the private learning stage. The client selection analysis explores the game-theoretic properties $\mathcal{M}_1$ and $\mathcal{M}_2$ provide, such as truthfulness and individual rationality. The private learning analysis examines the privacy guarantee promised by the mechanism with respect to specific privacy budget constraints.

5.1 Analysis of the Client Selection Stage

The analysis begins with proof that the mechanism $\mathcal{M}_1$ satisfies desired game-theoretic properties.

Theorem 1. *The mechanism $\mathcal{M}_1$ is individual rational. That is, each client gets non-negative utility $u_i = p_i - c_i \geq 0$ by participating in $\mathcal{M}_1$.*

Proof. By the payment rule, we know that $q_m \leq \frac{B}{\sum_{i\in S} d_i}$ and $q_{m+1} \geq q_m$. Thus, $\min\{\frac{B}{\sum_{i\in S} d_i}, q_{m+1}\} \geq q_m \geq q_i$ for $i \in S$. For each selected clients, $u_i = p_i - c_i = p_i - q_i \cdot d_i = \min\{\frac{B}{\sum_{i\in S} d_i}, q_{m+1}\} \cdot d_i - q_i \cdot d_i \geq 0$. For the clients who are not selected, their payment is $p_i = 0$, and their cost is $c_i = 0$, either. Therefore, $u_i \geq 0$ for every $i \in [n]$. □

Theorem 2. *The mechanism $\mathcal{M}_1$ is dominant-strategy truthful; clients do not benefit from mis-reporting.*

Proof. Suppose client i reports c_i' instead of c_i, and the position of $q_i' = \frac{c_i'}{d_i}$ in the sequence $\{q_1, \ldots, q_n\}$ changes from k to k', the payment changes from p_i to p_i'.

Situation 1. If client i is selected by reporting c_i truthfully, that means, $k \leq m$ and $q_k \leq q_m$.

- If client i reports a fake $c_i' < c_i$, then $q_i' \leq q_i$, and the position of client i moves up in the sequence of $\{q_1, \ldots, q_n\}$, i.e., $k' \leq k \leq m$. In this situation, client i is still selected, according to the payment rule, $p_i' = p_i$, and the utility of agent i will not change.
- If client i reports a fake $c_i' > c_i$, then $q_i' \geq q_i$, the position of client i moves back in the sequence of $\{q_1, \ldots, q_n\}$, i.e., $k' \geq k$. If $k \leq k' \leq m$, client i is still selected, according to the payment rule, $p_i' = p_i$. If $k \leq m \leq k'$, then client i would not be selected, according to the payment rule, $p_i' = 0$. In summary, $p_i \geq p_i'$ and client utility $p_i - c_i \geq p_i' - c_i$.

Situation 2. If client i is not selected by reporting c_i truthfully, that means, $k > m$ and $q_k > q_m$.

- If client i reports a fake $c_i' > c_i$, then $q_i' \geq q_i$, the position of client i moves back in the sequence of $\{q_1, \ldots, q_n\}$, i.e., $k' \geq k > m$. In this situation, client i will not be selected, $p_i' = p_i = 0$.
- If client i reports a fake $c_i' < c_i$, then $q_i' \leq q_i$, and the position of client i will move up in the sequence of $\{q_1, \ldots, q_n\}$, i.e., $k' \leq k$. If $m \leq k' \leq k$, then client i will not be selected and the payment $p_i' = p_i = 0$. If $k' \leq m \leq k$, then client i would be selected, according to the payment rule, $p_i' = \min\{\frac{B}{\sum_{i\in S} d_i}, q_{m+1}\} \cdot d_i$. In this case, $p_i' = \min\{\frac{B}{\sum_{i\in S} d_i}, q_{m+1}\} \cdot d_i \leq q_{m+1} \cdot d_i \leq q_k \cdot d_i = c_i$. As a result, $u_i = p_i - c_i \leq 0$. That means, although client i was selected after misreporting $c_i' < c_i$, the payment it receives will not cover the cost it incurs by participating in the training.

Given these scenarios, we conclude that clients will not benefit from mis-reporting. □

Theorem 3. *The mechanism $\mathcal{M}_1$ is budget-balanced, the total payment paid by the server does not exceeds its total monetary budget.*

Proof. By the payment rule, we know that for the selected clients, the total payment they receive is $\sum_{i\in S} p_i = \sum_{i\in S} \min\{\frac{B}{\sum_{i\in S} d_i}, q_{m+1}\} \cdot d_i \leq B$, and the payments for unselected clients are 0, therefore, $\sum_{i=1}^{n} p_i \leq B$. □

Before showing properties of mechanism $\mathcal{M}_2$, we first introduce Myreson's lemma.

Definition 9. *(Implementable allocation rule) An allocation rule* alloc *for a single-parameter environment is implementable if we can find a payment rule p such that the mechanism $\mathcal{M} = (alloc, p)$ is truthful.*

Definition 10. *(Monotone allocation rule) An allocation rule* alloc *for a single-parameter environment is monotone if for every client $i \in [n]$ and bids b_{-i} of the other clients, the allocation $alloc_i(z, b_{-i})$ to client i is nondecreasing in bid z.*

Intuitively, this monotonicity means the winner in an auction still wins (or wins more) by bidding at a higher value (or equivalently, a lower cost).

Lemma 2. *(Myerson's lemma [37]) For single parameter environments, the following three claims hold.*

1) *An allocation rule is implementable if and only if it is monotone.*
2) *If an allocation rule* alloc *is monotone, then there exists a unique payment rule p such that the mechanism $\mathcal{M} = (alloc, p)$ is truthful, assuming that a zero bid implies a zero payment.*
3) *The payment rule is given as*

$$p(b_i, b_{-i}) = b_i \text{alloc}_i(b_i, b_{-i}) - \int_0^{b_i} alloc_i(z, b_{-i}) \mathrm{d}z. \tag{10}$$

Theorem 4. *The mechanism $\mathcal{M}_2$ is dominant-strategy truthful; clients will not benefit from mis-reporting.*

Proof. According to Myerson's lemma, the core of the proof is to show that the allocation rule in mechanism $\mathcal{M}_2$ is

monotone. Note that, in our framework, the server S *buys* data from clients, which is performed as a reversed auction. Therefore, we need to show that the winning client still wins by lowering its reported costs.

For any $(d, c) \in \mathbb{R}^n_{\geq 0}$, and $\forall i$, let $c'_i < c_i$. Denote $c' = (c'_i, c_{-i})$, $T = \mathcal{M}_2(d, c)$, and $T' = \mathcal{M}_2(d, c')$. Denote $Q = selected(d, c)$, and $Q' = selected(d, c')$.

We first show that $i \in Q \Rightarrow i \in Q'$. This is because Q and Q' are the prefixes of clients with cumulated costs less than B in the ordered sequence of r_i. When client i lowers its costs from c_i to c'_i, the position of the client only moves up in the sequence. Therefore, client i is still in the prefix Q' if it was in the prefix Q.

Based on the observation that $i \in Q \Rightarrow i \in Q'$, we know that if $T = Q$ and $T' = Q'$, the mechanism is monotone. In addition, note that if $i \in Q$, then $\sum_{j \in Q'} d'_j \geq \sum_{j \in Q} d_j$. Therefore, if $T = Q$ and $T' = Q$, the mechanism is also monotone.

Finally, if $i \in T$, $i = i^*$ and $d_i > \sum_{j \in Q} d_j$, in this case, we must also have $i \in T'$. Client i remains the one with the highest d_i, and thus the output is either $T' = \{i^*\}$ or $T' = Q'$ with $i \in Q'$.

Therefore, we conclude that $i \in T \Rightarrow i \in T'$. That is, client i still wins by lowering its bid. In other words, the allocation rule produced by mechanism $\mathcal{M}_2$ is monotone. According to the Myerson's lemma, mechanism $\mathcal{M}_2$ is truthful, and the payment is computed by Equation (10). □

5.2 Analysis of the Private Learning Stage

Before discussing the privacy guarantee of the mechanisms, we need to introduce some theorems that are widely used in proofs of differential privacy's mathematical properties.

Lemma 3. *(Parallel Composition [40]) Given a set of mechanisms $\mathcal{M} = \{\mathcal{M}_1, \ldots, \mathcal{M}_m\}$, if $\mathcal{M}_i$ provides ϵ_i privacy guarantee on a disjointed subset of the entire dataset, $\mathcal{M}$ will provide $\max\{\epsilon_1, \ldots, \epsilon_m\}$-differential privacy.*

Lemma 4. *(Sequential Composition [40]) Suppose a set of mechanisms $\mathcal{M} = \{\mathcal{M}_1, \ldots, \mathcal{M}_m\}$ are sequentially performed on a dataset, and each $\mathcal{M}_i$ satisfies ϵ_i differential privacy, $\mathcal{M}$ will provide $\sum_i \epsilon_i$-differential privacy.*

Differential privacy is immune to post-processing. Applying an arbitrary data-independent mapping function f to a differentially private mechanism $\mathcal{M}$ does not influence its privacy guarantee.

Lemma 5. *(Post-processing [15]) Suppose $\mathcal{M} : D \to \mathcal{R}$ is a (ϵ, δ)-differentially private mechanism. Let $f : \mathcal{R} \to \mathcal{R}'$ be an arbitrary randomized mapping. Then $f \circ \mathcal{M} : D \to \mathcal{R}'$ is (ϵ, δ)-differentially private.*

Theorem 5. *The parameter updating mechanisms ($\mathcal{M}_3$, $\mathcal{M}_4$, $\mathcal{M}_5$) satisfies $(\epsilon_L + \epsilon_E)$-joint differential privacy.*

Proof. The mechanism $\mathcal{M}_3$ has each client add Gaussian distributed noise to their data θ_i, which can be thought of as using a Gaussian mechanism on a disjoint partition of a database $\{\theta_i\}^m$. According to the theory of parallel composition (Lemma 3), the sequence $\{\hat{\theta}_1, \ldots, \hat{\theta}_m\}$ satisfies ϵ_L-differential privacy.

In mechanism $\mathcal{M}_4$, the sever computes $d_i\hat{\theta}_i$ using the noisy parameters $\hat{\theta}_i$ provided by each client, which are post processing operations on an ϵ_L-differentially private computation $\hat{\theta}_i$. According to the theory of post-processing (Lemma 5), $\{d_j\hat{\theta}_j\}^m$ satisfies ϵ_L-differential privacy. The server employs an exponential mechanism to select high-quality parameter aggregations, which is based on ϵ_L-differentially private $\{d_j\hat{\theta}_j\}^m$. According to the theory of post-processing (Lemma 5) and the theory of sequential composition (Lemma 4), this process satisfies $(\epsilon_L + \epsilon_E)$-differential privacy.

Mechanism $\mathcal{M}_5$ has each client update their parameters θ_i as a function of their private information θ_i and an $(\epsilon_L + \epsilon_E)$-differentially private signal (i.e., the result of the exponential mechanism). According to the billboard lemma (Lemma 1), the results $\{\theta_i\}^m$ satisfy $(\epsilon_L + \epsilon_E)$-joint differential privacy. □

6 Experiments and Analysis

6.1 Experimental Setup

The two datasets selected were the $MINST$ [41] and $CIFAR-10$ [42] datasets. The $MINST$ dataset contains 60,000 training images and 10,000 test images of handwritten numerical digits. Each image is grayscale and normalized to 28×28 pixels. The $CIFAR-10$ dataset consists of 60,000 32×32 color images spanning 10 classes; 50,000 of them are training images and 10,000 are for testing. With the $MINST$ dataset, we trained a CNN with two convolution layers, and two fully connected layers with dropout regularization. The network for the $CIFAR-10$ dataset comprised six convolution layers with ReLU units and max-pooling layers. The learning rate for both networks was 0.001 and the momentum was 0.9.

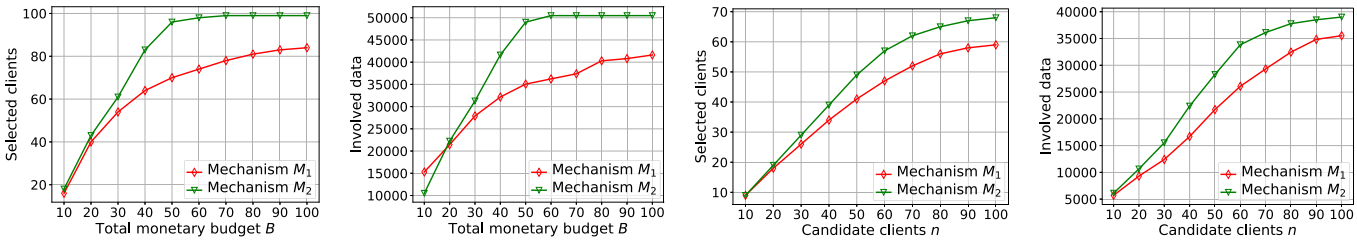
As a baseline, we used the traditional parameter aggregation method that is common to federated learning, where the server simply computes a weighted average of the parameters submitted by clients. We also included one adversarial client whose aim was to skew the global model via a dirty-label attack [9]. Hence, with the $MINST$ dataset, the adversary changes the labels of all its training images to the digit 2. With the $CIFAR-10$ dataset, it changes all the labels to cat. For all other clients, the data is uniformly sampled from different classes. Each client perturbs their data locally using the Gaussian mechanism. The privacy budgets used in the mechanisms were set to $\epsilon = \epsilon_L = \epsilon_E$.

6.2 Experimental Results

6.2.1 The Effectiveness of the Proposed Methods

The first simulation was designed to examine how differing privacy budgets influence client selection and the incentive to participate. Here, we randomly generated the client's private costs in the range of [1,1000], and the amount of data they hold is in the interval [1,1000]. Different distributions of client's cost will not influence the game-theoretic properties of proposed mechanisms, thus we only limited the range of clients' private costs here. We also varied the server's total monetary budget from 10 units to 100 units.

Fig. 2a shows the change in the number of clients selected with respect to the server's total monetary budget. In each simulation, $n = 100$ clients submit bids to participate in the



(a) The number of selected clients vs. the server's monetary budget

(b) The number of involved data vs. the server's monetary budget

(c) The number of selected clients vs. the number of candidate clients

(d) The number of involved data vs. the number of candidate clients

Fig. 2. The performance of the method with various server budgets and numbers of candidate clients.

training. Initially, a large number of clients were selected because the higher the monetary budget, the more clients the server can attract. In fact, with a sufficiently large monetary budget, the server was able to involve almost every client in the training, and so, at higher budget levels, the number of clients selected did not change much.

Fig. 2b shows the amount of data provided by the selected clients. The results show similar tendencies to Fig. 2a. In addition, mechanism $\mathcal{M}_2$ (Algorithm 2) is designed with the aim of maximizing the amount of training data. As a result, it converges faster than the envy-free $\mathcal{M}_1$.

Fig. 2c shows the change in the number of selected clients when the number of candidates submitting bids changes, where the server's monetary budget is fixed. The results show that both $\mathcal{M}_1$ and $\mathcal{M}_2$ selected more clients as the number of candidates increased. However, because the total monetary budget is fixed, there came a sort of saturation point where it was simply impossible to add more clients given that each has a base cost of more than 0. A similar tendency can also be observed in Fig. 2d, which shows the amount of data provided by clients with respect to the number of candidates. When the number of candidates reaches 90, the amount of data stabilized at around 35,000 units for $\mathcal{M}_1$ and 39,000 units for $\mathcal{M}_2$. Notably, because $\mathcal{M}_2$ works to solve a social welfare maximization problem (Equation (9)), it always incorporated more training data and selected more clients.

6.2.2 Our Methods versus Traditional Methods

Our next experiment involved a comparison between the proposed method and traditional method, both under attack, with varied privacy budget. The accuracy of traditional method without attack was used as a baseline. This simulation included $m = 20$ clients, one of which was an adversary in the two attack scenarios. The data was evenly distributed among clients and performance was assessed in terms of the global model's accuracy. The results are shown in Fig. 3.

The result patterns were similar for both datasets. With a small privacy budget, the clients had to inject a good deal of noise, which impacted accuracy. But, as the privacy budget increased, less noise was needed and accuracy improved. The "no attack" method was obviously the most accurate, followed by our method under attack, and then the traditional method. It is worth pointing out that once the privacy budget reaches a certain level, noise is no longer the main factor influencing accuracy; thus, the curves flatten. However, even at this point, our method was still more accurate than the traditional method.

In Fig. 3, we observe that the malicious client does not cause a serious decrease in the global model's accuracy. This is because the malicious client only changes the label of one class in its local training. In addition, the malicious client cannot have access to the number of training data used by other clients, its update is more likely to be detected directly by the stealth metrics of the server.

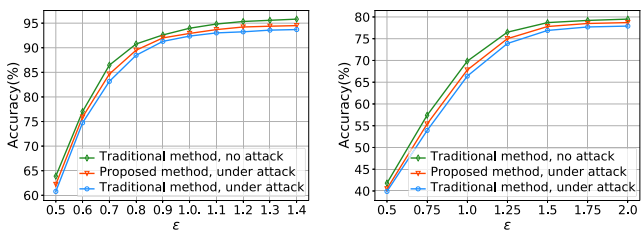
6.2.3 Model Accuracy With Privacy Budget

Figs. 4a and 4b show the performance of the method with different privacy budgets. With a larger privacy budget, each client injects less noise into the submitted parameters. As a result, the accuracy of the model increases. Similar variation tendencies can also be observed in Figs. 3a and 3b. Besides, with a small number of clients, involving more clients in the training reduces the impact of the adversary, which also improves model accuracy.

However, we note that the privacy budget ϵ should be set carefully. On the one hand, a smaller ϵ means that any single client has less influence over other clients' data, which makes the learning framework more robust to adversarial manipulations. On the other hand, a small ϵ also flattens the probability distribution generated by the exponential mechanism, and further influences the model's performance. Therefore, although smaller ϵ protects the client's data privacy better, introducing more randomizations in the exponential mechanism impacts the model's accuracy. Hence, it is important to make the right tradeoff between robustness and performance for different situations.

6.2.4 Model Accuracy and the Number of Clients

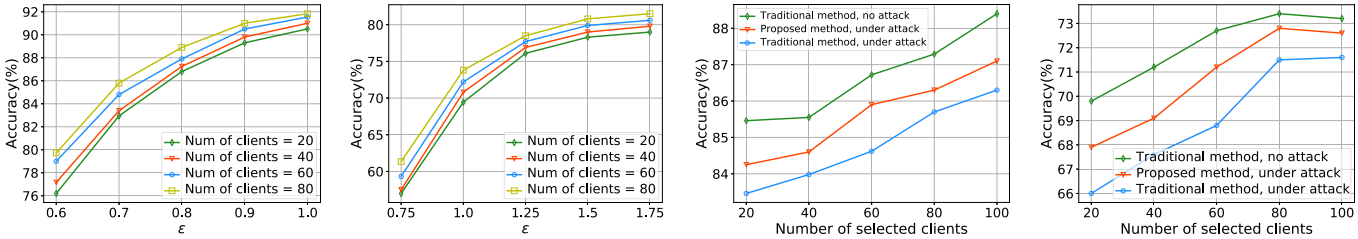
Figs. 4c and 4d show the model accuracy with different numbers of clients. In Fig. 4c, we see that the accuracy of the



(a) Accuracy of the global model vs. privacy budget, MNIST dataset.

(b) Accuracy of the global model vs. privacy budget, CIFAR-10 dataset.

Fig. 3. A comparison between our method and traditional methods.



(a) The accuracy of the global model vs. the privacy budget, MNIST dataset.

(b) The accuracy of the global model vs. the privacy budget, CIFAR-10 dataset.

(c) The accuracy of the global model vs. the number of client, MNIST dataset.

(d) The accuracy of the global model vs. the number of client, CIFAR-10 dataset.

Fig. 4. The change in global model accuracy with various privacy budgets and numbers of clients.

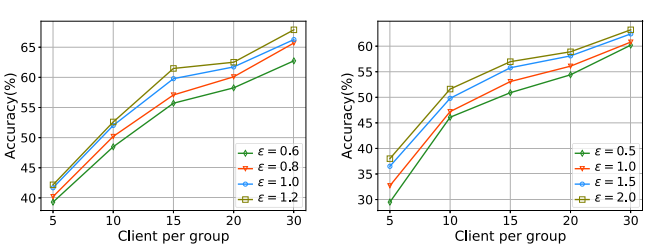
global model increases with an increase in participating clients. This is because involving more benign clients in the training process reduces the impact of the adversary. In other words, incorporating more clients increases the number of benign clients within each candidate groups, and the weighted average of parameters in each candidate group becomes more accurate. By contrast, Fig. 4d shows that the accuracy of the model first increases but then flattens (and even decreases) as more clients participate. The reason is that the adversarial client is only in one candidate group, and the score of this group is at most as good as that of the other groups. Therefore, the exponential mechanism is making a selection over an almost uniform distribution when the number of clients is very large, and thus the global model's accuracy fluctuates.

What Fig. 4d demonstrates is that involving more clients is not always beneficial to the training, because incorporating more clients cannot continuously improve the global model's accuracy, but recruiting more clients in the training requires the server to spend more monetary budgets. Thus, again, a balance is needed – this time, between the number of participating clients and model accuracy.

6.2.5 *Model Accuracy and the Server's Grouping Strategy*

Fig. 5 presents the influence of the server's grouping strategies on model accuracy. In Algorithm 4, the server applies leave-one-out strategy to construct different client groups. In this experiment, we took $m = 60$ clients including 1 adversary, and had the server randomly partition them into groups where each contains 5, 10, 15, 20 and 30 clients. From the figures for both datasets, we can see that the model accuracy increases as the number of clients in each group increases, but the accuracy in these scenarios are lower than that obtained using the leave-one-out strategy. This is because averaging over more clients reduces the variance of the noise on the averaged value.

Note, however, that this observation only holds when each client has a similar amount of training data. In our framework, the amount of data used in the training process is considered as the weight of each client in parameter aggregation. So, if the adversary locally trains the model with much more data than the other clients, the impact of the poisoned dirty label data might be more substantial.



(a) The accuracy of the global model vs. clients per group, MNIST dataset.

(b) The accuracy of the global model vs. clients per group, CIFAR-10 dataset.

Fig. 5. The change in global model accuracy with various numbers of clients allocated to each group.

6.3 Discussion

In these experiments, we considered the influence of exactly one adversarial participant in the training process. With the traditional method, the adversary was able to cause a $4\% \sim 5\%$ decrease in model accuracy via its dirty-label attack. However, under our joint differential privacy regime, the adversary's impact was limited to a $1\% \sim 2\%$ decrease.

Our experiments validate the effectiveness of the privacy budget as a means of controlling both the noise level in privacy protection and the influence of an adversary. We also examined the game-theoretic properties of the proposed mechanisms, and demonstrated the effectiveness of the framework in terms of privacy budget, the number of clients and the server's group strategy. In addition, an appropriate tradeoff between accuracy and privacy should be carefully considered when setting privacy budgets.

7 Conclusion

In this paper, we proposed a robust federated learning framework that addresses the practical problem of incentivizing clients to participate in federated learning. As a solution, we suggested two novel, game-theoretic mechanisms that formulate client selection as an auction game. The clients report their costs as bids and the server uses different payment strategies to maximize its objectives. The overall framework is jointly differentially private, which limits the impact of adversarial clients. Additionally, we have conducted simulations with real-world datasets to validate the framework's performance. The results demonstrate that under our scheme, an adversarial client has little impact on the global model's accuracy.

In future work, we plan to explore more complex federated learning scenarios where more than one adversarial client has been selected to participate in the training. At present, joint differential privacy limits the impact of a single adversarial client. With more adversarial clients

involved, we intend to explore ways to extend and/or relax the current definition of differential privacy, and develop new federated learning frameworks based on these revised definitions. In addition, we plan to design one or more client selection mechanisms based on different kinds of auctions where the dataset information is invisible to the server.

REFERENCES

[1] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Comput. Surv.*, vol. 54, no. 6, Jul. 2021, Art. no. 131.

[2] Google-Research, "Federated learning: Collaborative machine learning without centralized training data," Apr. 2017. [Online]. Available: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

[3] Y. Zhan, J. Zhang, Z. Hong, L. Wu, P. Li, and S. Guo, "A survey of incentive mechanism design for federated learning," *IEEE Trans. Emerg. Topics Comput.*, early access, Mar. 03, 2021, doi: 10.1109/TETC.2021.3063517.

[4] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10 700–10 714, Dec. 2019.

[5] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6360–6368, Jul. 2020.

[6] T. H. T. Le *et al.*, "An incentive mechanism for federated learning in wireless cellular network: An auction approach," *IEEE Trans. Wireless Commun.*, vol. 20, no. 8, pp. 4874–4887, Aug. 2021.

[7] M. Tang and V. W. S. Wong, "An incentive mechanism for cross-silo federated learning: A public goods perspective," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[8] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. 23rd Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2938–2948.

[9] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 634–643.

[10] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 3–18.

[11] K. Wei *et al.*, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 3454–3469, 2020.

[12] R. Hu, Y. Guo, H. Li, Q. Pei, and Y. Gong, "Personalized federated learning with differential privacy," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9530–9539, Oct. 2020.

[13] M. Hao, H. Li, G. Xu, S. Liu, and H. Yang, "Towards efficient and privacy-preserving federated deep learning," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–6.

[14] L. Zhao, Q. Wang, Q. Zou, Y. Zhang, and Y. Chen, "Privacy-preserving collaborative deep learning with unreliable participants," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 1486–1500, 2020.

[15] C. Dwork and A. Roth, *The Algorithmic Foundations of Differential Privacy*. Boston, MA, USA: Now, 2014.

[16] L. Zhang, T. Zhu, P. Xiong, W. Zhou, and P. S. Yu, "More than privacy: Adopting differential privacy in game-theoretic mechanism design," *ACM Comput. Surv.*, vol. 54, no. 7, Jul. 2021, Art. no. 136.

[17] D. Li, Q. Yang, W. Yu, D. An, Y. Zhang, and W. Zhao, "Towards differential privacy-based online double auction for smart grid," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 971–986, 2020.

[18] J. Lin, D. Yang, M. Li, J. Xu, and G. Xue, "Frameworks for privacy-preserving mobile crowdsensing incentive mechanisms," *IEEE Trans. Mobile Comput.*, vol. 17, no. 8, pp. 1851–1864, Aug. 2018.

[19] D. Ye, T. Zhu, S. Shen, and W. Zhou, "A differentially private game theoretic approach for deceiving cyber adversaries," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 569–584, 2021.

[20] C. Dwork, "Differential privacy," in *Automata, Languages and Programming*, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Eds. Berlin, Germany: Springer, 2006, pp. 1–12.

[21] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *Proc. 48th Annu. IEEE Symp. Found. Comput. Sci.*, 2007, pp. 94–103.

[22] K. Nissim, R. Smorodinsky, and M. Tennenholtz, "Approximately optimal mechanism design via differential privacy," in *Proc. 3rd Innov. Theor. Comput. Sci. Conf.*, 2012, pp. 203–213.

[23] M. Kearns, M. Pai, A. Roth, and J. Ullman, "Mechanism design in large games: Incentives and privacy," in *Proc. 5th Conf. Innov. Theor. Comput. Sci.*, 2014, pp. 403–410.

[24] Z. Huang, J. Liu, and X. Wang, "Learning optimal reserve price against non-myopic bidders," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 2042–2052.

[25] T.-H. H. Chan, E. Shi, and D. Song, "Private and continual release of statistics," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 3, Nov. 2011, Art. no. 26.

[26] W. Y. B. Lim *et al.*, "Hierarchical incentive mechanism design for federated machine learning in mobile networks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9575–9588, Oct. 2020.

[27] R. B. Myerson, *Game Theory: Analysis of Conflict*. Cambridge, MA, USA: Harvard Univ. Press, 1991.

[28] D. Tian, J. Zhou, Y. Wang, Z. Sheng, X. Duan, and V. C. M. Leung, "Channel access optimization with adaptive congestion pricing for cognitive vehicular networks: An evolutionary game approach," *IEEE Trans. Mobile Comput.*, vol. 19, no. 4, pp. 803–820, Apr. 2020.

[29] A. S. Chivukula, X. Yang, W. Liu, T. Zhu, and W. Zhou, "Game theoretical adversarial deep learning with variational adversaries," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 11, pp. 3568–3581, Nov. 2021.

[30] Z. Chen, T. Ni, H. Zhong, S. Zhang, and J. Cui, "Differentially private double spectrum auction with approximate social welfare maximization," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 11, pp. 2805–2818, Nov. 2019.

[31] T. Zhu, D. Ye, W. Wang, W. Zhou, and P. Yu, "More than privacy: Applying differential privacy in key areas of artificial intelligence," *IEEE Trans. Know. Data Eng.*, early access, Aug. 04, 2020, doi: 10.1109/TKDE.2020.3014246.

[32] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory Cryptogr. Conf.*, 2006, pp. 265–284.

[33] J. Hsu, Z. Huang, A. Roth, T. Roughgarden, and Z. S. Wu, "Private matchings and allocations," in *Proc. 46th Annu. ACM Symp. Theory Comput.*, 2014, pp. 21–30.

[34] A. Ghosh and A. Roth, "Selling privacy at auction," in *Proc. 12th ACM Conf. Electron. Commerce*, 2011, pp. 199–208.

[35] L. Chen, H. Wang, Z. Charles, and D. Papailiopoulos, "DRACO: Byzantine-resilient distributed training via redundant gradients," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 903–912.

[36] A. Mu'alem and N. Nisan, "Truthful approximation mechanisms for restricted combinatorial auctions," *Games Econ. Behav.*, vol. 64, no. 2, pp. 612–631, 2008.

[37] R. B. Myerson, "Optimal auction design," *Math. Oper. Res.*, vol. 6, no. 1, pp. 58–73, 1981.

[38] P. Briest, P. Krysta, and B. Vöcking, "Approximation techniques for utilitarian mechanism design," in *Proc. 37th Annu. ACM Symp. Theory Comput.*, 2005, pp. 39–48.

[39] R. Cummings, S. Ioannidis, and K. Ligett, "Truthful linear regression," in *Proc. 28th Conf. Learn. Theory*, 2015, pp. 448–483.

[40] F. McSherry, "Privacy integrated queries: An extensible platform for privacy-preserving data analysis," *Commun. ACM*, vol. 53, no. 9, pp. 89–97, Sep. 2010.

[41] Y. LeCun, "The MNIST database of handwritten digits," 1998. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[42] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., Univ. Toronto, 2009.

Lefeng Zhang received the BEng and MEng degrees from the Zhongnan University of Economics and Law, Wuhan, China, in 2016 and 2019, respectively. He is currently working toward the PhD degree with the University of Technology Sydney, Sydney, Australia. His research interests include game theory and privacy preserving.

Tianqing Zhu received the BEng and MEng degrees from Wuhan University, Wuhan, China, in 2000 and 2004, respectively, and the PhD degree in computer science from Deakin University, Geelong, Australia, in 2014. She is currently an associate professor with the School of Computer Science, University of Technology Sydney, Australia. Before that, she was a lecture with the School of Information Technology, Deakin University, Australia, from 2014 to 2018. Her research interests include privacy preserving, data mining, and network security.

Ping Xiong received the BEng degree from Lanzhou Jiaotong University, Lanzhou, China, in 1997, and the MEng and PhD degrees from Wuhan University, Wuhan, China, in 2002 and 2005, respectively. He is currently the professor with the School of Information and Security Engineering, Zhongnan University of Economics and Law, China. His research interests include network security, data mining, and privacy preservation.

Wanlei Zhou (Senior Member, IEEE) received the BEng and MEng degrees in computer science and engineering from the Harbin Institute of Technology, Harbin, China, in 1982 and 1984, respectively, the PhD degree in computer science and engineering from the Australian National University, Canberra, Australia, in 1991, and the DSc (a higher doctorate degree) degree from Deakin University, Geelong, Australia, in 2002. He is currently the vice rector (Academic Affairs) and dean with the Institute of Data Science, City University of Macau, Macao SAR, China. Before joining City University of Macau, he held various positions including the head of the School of Computer Science, University of Technology Sydney, Australia, the Alfred Deakin professor, chair of information technology, associate dean, and head of the School of Information Technology, Deakin University, Australia. He also served as a lecturer with the University of Electronic Science and Technology of China, a system programmer in HP at Massachusetts, USA; a lecturer with Monash University, Melbourne, Australia; and a lecturer with the National University of Singapore, Singapore. His research interests include security, privacy, and distributed computing. He has published more than 400 papers in referred international journals and referred international conferences proceedings, including many articles in IEEE transactions and journals.

Philip S. Yu (Fellow, IEEE) received the BS degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, and the MS and PhD degrees in electrical engineering from the Stanford University, Stanford, California. He is currently a distinguished professor in computer science with the University of Illinois at Chicago and also currently the Wexler chair of information technology. He was a recipient of ACM SIGKDD 2016 Innovation Award for his influential research and scientific contributions on mining, fusion, and anonymization of Big Data, the IEEE Computer Society's 2013 Technical Achievement Award for pioneering and fundamentally innovative contributions to the scalable indexing, querying, searching, mining, and anonymization of Big Data. He was the editor-in-chiefs of *ACM Transactions on Knowledge Discovery from Data*, from 2011 to 2017, and *IEEE Transactions on Knowledge and Data Engineering*, from 2001 to 2004.