Incentive Mechanism Design for Federated Learning and Unlearning

Ningning Ding
Northwestern University, USA
ningning.ding@northwestern.edu

Zhenyu Sun
Northwestern University, USA
zhenyusun2026@u.northwestern.edu

Ermin Wei
Northwestern University, USA
ermin.wei@northwestern.edu

Randall Berry
Northwestern University, USA
rberry@northwestern.edu

ABSTRACT

To protect users' *right to be forgotten* in federated learning, federated unlearning aims at eliminating the impact of leaving users' data on the global learned model. The current research in federated unlearning mainly concentrated on developing effective and efficient unlearning techniques. However, the issue of incentivizing valuable users to remain engaged and preventing their data from being unlearned is still under-explored, yet important to the unlearned model performance. This paper focuses on the incentive issue and develops an incentive mechanism for federated learning and unlearning. We first characterize the leaving users' impact on the global model accuracy and the required communication rounds for unlearning. Building on these results, we propose a four-stage game to capture the interaction and information updates during the learning and unlearning process. A key contribution is to summarize users' multi-dimensional private information into one-dimensional metrics to guide the incentive design. We show that users who incur high costs and experience significant training losses are more likely to discontinue their engagement through federated unlearning. The server tends to retain users who make substantial contributions to the model but has a trade-off on users' training losses, as large training losses of retained users increase privacy costs but decrease unlearning costs. The numerical results demonstrate the necessity of unlearning incentives for retaining valuable leaving users, and also show that our proposed mechanisms decrease the server's cost by up to 53.91% compared to state-of-the-art benchmarks.

CCS CONCEPTS

• **Computing methodologies** → **Model development and analysis**; **Machine learning**; • **Security and privacy** → **Economics of security and privacy**.

KEYWORDS

incentive mechanism, federated learning, federated unlearning

MobiHoc '23, October 23–26, 2023, Washington, DC, USA

ACM ISBN 978-1-4503-9926-5/23/10…$15.00
https://doi.org/10.1145/3565287.3610269

ACM Reference Format:
Ningning Ding, Zhenyu Sun, Ermin Wei, and Randall Berry. 2023. Incentive Mechanism Design for Federated Learning and Unlearning. In *International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '23), October 23–26, 2023, Washington, DC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3565287.3610269

1 INTRODUCTION

1.1 Background and Motivations

Federated learning is a promising distributed machine learning paradigm, in which multiple users collaborate to train a shared model under the coordination of a central server [13]. This approach allows users to keep their local data on their own devices and only share the intermediate model parameters, which helps protect their raw data. However, despite these measures, it may not provide sufficient privacy guarantees [14, 18].

For privacy reasons, one desirable property of a federated learning platform is the users' "right to be forgotten" (RTBF), which has been explicitly stated in the European Union General Data Protection Regulation (GDPR) [19] and the California Consumer Privacy Act (CCPA) [7]. That is, a user has the right to request deletion of his private data and its impact on the trained model, if he no longer desires to participate in the platform. Users may seek to leave a platform for a variety of reasons. For example, they may feel that the benefits from the platform are not sufficient to compensate for their potential privacy leakage from participation. Furthermore, until they participate in the platform, they may not have full knowledge of these benefits and costs due to incomplete information about other users' data. For instance, users' privacy costs in federated learning depend on how unique their data is [8], which they can infer from their training loss after training [6].

To remove data from a trained federated learning model, the concept of *federated unlearning* has recently been proposed [12]. In this concept, after some users request to revoke their data, staying users will perform additional training or calculation to eliminate the impact of leaving users' data and obtain an unlearned model. A simple yet costly approach is to retrain the model from scratch with the requested data being removed from the training dataset [1]. To be more efficient and effective, existing literature (e.g., [6, 11, 25]) focused on alternative federated unlearning methods that obtain a model similar (in some distance metrics) to a retrained model with lower computational costs. However, these studies usually assumed that users are willing to participate in federated learning and unlearning. This assumption may not be realistic

without proper incentives since users incur various costs during the training process (e.g., time, energy, and privacy costs). Our goal in this paper is to develop incentive mechanisms to help retain valuable leaving users and create a sustainable learning platform for both the users and the server.

There are several challenges for designing an incentive mechanism for federated learning and unlearning. First, different leaving users will lead to different unlearned model performances and unlearning costs, the relationship among which is still an open problem. Second, it is difficult for the server to design incentives for a large number of heterogeneous users, when users have multi-dimensional private information (e.g., training costs and privacy costs) and unknown information (e.g., users' training losses before federated learning). Third, unlearning incentives for retaining valuable leaving users require careful design. High incentives may encourage strategic users to intentionally request revocation to obtain retention rewards, while low incentives may fail to retain valuable users. It is also crucial for the server to distinguish between high-quality leaving users (e.g., with rare and valuable data) and low-quality ones (e.g, with erroneous data), both of which can lead to high training losses. Fourth, both learning and unlearning incentives affect the server's and users' payoffs but are determined in different stages - before or after federated learning. Meanwhile, there are different information asymmetry levels in each stage, as the federated learning process can reveal some information such as users' training losses and contributions.

The above discussion motivates us to answer the following interesting question: *Considering leaving users' impact, what is the server's optimal incentive mechanism for federated learning and unlearning, when heterogeneous users have strategic data revocation decisions and multi-dimensional private and unknown information?*

1.2 Contributions

We summarize our key contributions below.

- *Incentive mechanism design for federated learning and unlearning.* We propose a four-stage Stackelberg game to analyze the optimal incentives of the server and the optimal strategies of users within this game. To the best of our knowledge, this is the first analytical study of incentive mechanisms for federated learning and unlearning.
- *Theoretical characterization of global model accuracy and unlearning communication rounds.* We theoretically derive bounds on the global model optimality gap given non-IID data for a federated learning algorithm (Scaffold [9]) and the number of global communication rounds required for a federated unlearning method.
- *Optimal incentives and revocation decisions under multi-dimensional incomplete information.* Due to the complex interaction, users' multi-dimensional private information, and dynamically updated knowledge, the server's optimization problem in Stage I of the four-stage game is highly complex. We summarize users' multi-dimensional heterogeneity into several one-dimensional metrics and develop an efficient algorithm with linear complexity, to handle the exponentially large number of possible cases involved in optimal mechanism design. We also identify and analyze a supermodular game among the users to obtain their optimal data revocation decisions.
- *Insights and Performance Evaluation.* We show that high costs and training losses motivate users to leave, while the server will retain the leaving users who make significant contributions to model accuracy but not necessarily low training losses, as small losses of retained users will reduce privacy costs yet increase unlearning costs. We numerically show that compared with state-of-the-art benchmarks, our proposed incentive mechanism decreases the server's cost by up to 53.91%. Moreover, the results demonstrate that it is beneficial for the server to retain valuable leaving users and jointly optimize the federated learning and unlearning incentive mechanisms.

1.3 Related Work

The concept of *machine unlearning*, which refers to the process of removing the impact of a data sample from a trained model, was first introduced by Cao et al. in 2015 [2]. Most related literature was about centralized machine unlearning (e.g., [1]), in which the unlearned model (not retrained from scratch) was trained on summarized (e.g., aggregates of summations) or partitioned subsets rather than individual training samples. As a result, the model only needed to be updated on the subset(s) of data that are associated with the requested samples. Centralized unlearning methods are not suited to federated learning, due to (i) lack of direct data access, (ii) the fact that the global model is updated based on the aggregated rather than the raw gradients, and (iii) the possibility that different users may have similar training samples [6]. This motivated the emergence of *federated unlearning*.

Only a few studies proposed federated unlearning mechanisms using methods such as gradient subtraction (e.g., [11, 12]), gradient scaling (e.g., [6]), or knowledge distillation (e.g., [25]). Albeit with good numerical performance, there is no theoretical guarantee of these proposed federated unlearning methods. To fill this gap, we propose theoretical bounds on the model optimality gap and communication rounds for one approach to federated unlearning.

Furthermore, there is a wide spectrum of literature on incentive mechanisms for various systems, including crowdsensing (e.g., [26]), wireless networks (e.g., [29]), data trading (e.g., [22]), and energy sharing (e.g., [20]). Some important work studied incentive mechanism design for federated learning to discourage valuable clients from leaving (e.g., [4, 23, 27, 28]). However, very few of them considered users' multi-dimensional private information (e.g., [4]), and none of them incorporated the unique aspects of federated unlearning (e.g., unlearning costs) or the dynamics of users' payoffs (e.g., pre-/post-training and before/after some users leave). This paper is the first to focus on incentive mechanism design for both federated learning and unlearning.

The rest of the paper is organized as follows. In Section 2, we characterize the models of federated learning and unlearning. We describe the system model in Section 3 and calculate the optimal incentive mechanisms in Section 4. Simulation results are presented in Section 5, and we conclude in Section 6.

2 CHARACTERIZATION OF FEDERATED LEARNING AND UNLEARNING MODELS

Before modeling the game-theoretic interaction between the server and the users in the next section, we first discuss federated learning and unlearning models in this section as a preliminary. Specifically, we specify the learning and unlearning objectives in Sections 2.1 and 2.2, respectively. Then, we derive bounds on global model accuracy and federated unlearning time in Section 2.3.

2.1 Federated Learning Objective

Consider an example of data (x_a, y_a), where x_a is the input (e.g., an image) and y_a is the label (e.g., the object in the image). The objective of learning is to find the proper model parameter w that can predict the label y_a based on the input x_a. Let us denote the prediction value as $\tilde{y}(x_a; w)$. The gap between the prediction $\tilde{y}(x_a; w)$ and the ground truth label y_a is characterized by the prediction loss function $f_a(w)$. If user i selects a set of local data with data size d_i to train the model, the loss function of user $i \in \mathcal{I}$ is the average prediction loss on all his training data:

$$F_i(w) = \frac{1}{d_i} \sum_{a=1}^{d_i} f_a(w). \tag{1}$$

The purpose of federated learning is to compute the model parameter w by using all users' local data. The optimal model parameter w^* minimizes the global loss function, which is an average of all users' loss functions [9, 15]:[1]

$$w^* = \arg\min_w \frac{1}{I} \sum_{i \in \mathcal{I}} F_i(w). \tag{2}$$

2.2 Federated Unlearning Objective

A federated learning process maps users' data into a model space, while a federated unlearning process maps a learned model, users' data set, and the data set that is required to be forgotten into an unlearned model space. The goal of federated unlearning is to make the unlearned model have the same distribution as the retrained model (i.e., retrained from scratch using the remaining data).[2]

A natural method for federated unlearning is to let the remaining users (excluding leaving users) continue training from the learned model w^*, until it converges to a new optimal model parameter $\tilde{w}^*$ that minimizes the global loss function of remaining users:

$$\tilde{w}^* = \arg\min_w \frac{1}{I - I_{leave}} \sum_{i \in \mathcal{I} \setminus \mathcal{I}_{leave}} F_i(w), \tag{3}$$

where $\mathcal{I}_{leave}$ is the set of users who leave the system through federated unlearning. This method is typically more efficient than training from scratch, as the minimum point may not change much after some users leave.

2.3 Model Accuracy and Unlearning Time

Given the objectives of federated learning and unlearning, we analyze the model accuracy gap and unlearning time in the following.

It has been shown that many federated learning algorithms (e.g., FedAvg [13]) suffer from significant communication overhead [10]. Scaffold [9] can mitigate this issue by incorporating an additional correction term based on gradient tracking techniques during local updates. Thus, we use Scaffold as the federated learning algorithm when deriving the optimality gap of the global model.[3] In each local iteration of the algorithm, every user computes a mini-batch gradient with batch size s_i. A batch or minibatch refers to equally sized subsets of the training dataset over which the gradient is calculated. In this paper, we consider the widely adopted setting that users' batch sizes $\{s_i\}_{i \in \mathcal{I}}$ are in the same proportion to their data sizes $\{d_i\}_{i \in \mathcal{I}}$ (i.e., $s_i = \iota d_i, \forall i \in \mathcal{I}, \iota \in (0, 1)$) [1, 4, 17].

The following proposition presents a bound on the optimality gap for the global model trained with Scaffold:

Proposition 1. *Suppose each user's loss function F_i is μ-strongly-convex and L-Lipschitz-smooth. Consider the federated learning algorithm Scaffold with the local iteration number of user i denoted by K_i and local step size denoted by η_i. Setting $\bar{\eta} = \eta_i K_i \leq \frac{1}{12L}$, we have*

$$\mathbb{E}\|w_{t+1} - w^*\|^2 \leq (1 - \frac{\mu\bar{\eta}}{2})\mathbb{E}\|w_t - w^*\|^2 + \frac{22\bar{\eta}^2\sigma^2}{I} \sum_{i \in \mathcal{I}} \frac{1}{s_i}, \tag{4}$$

where w_{t+1} and w_t represent the model parameter after global round $t+1$ and t, respectively, s_i is user i's local batch size, and σ^2 is the variance bound of each data sample.[4] Moreover, by selecting $\bar{\eta} = \frac{c}{t+1}$ for some $0 < c \leq \frac{1}{12L}$, we have that the expected optimality gap of the global model satisfies:

$$\mathbb{E}\|w_t - w^*\|^2 \leq \frac{1}{t+1}\left(\frac{b(c)\sigma^2}{I} \sum_{i \in \mathcal{I}} \frac{1}{s_i} + \|w_0 - w^*\|^2\right), \tag{5}$$

where $b(\cdot)$ is a function of c.

The proof of Proposition 1 is given in Appendix A in the technical report [5]. As a large optimality gap $\|w_t - w^*\|^2$ means a high accuracy loss of the global model, Proposition 1 presents a relationship between the expected global model accuracy loss and the users' data sizes. As shown in (5), the expected accuracy loss of the global model decreases in the users' training batch sizes $\{s_i\}_{i \in \mathcal{I}}$ (and thus data sizes $\{d_i\}_{i \in \mathcal{I}}$). Moreover, we explain two asymptotic cases of (5) for better understanding. When the initial point is optimal (i.e., $w_0 = w^*$), the bound does not go to zero due to sample randomness. When batch size s_i is large enough, the randomness is then highly reduced and the bound is only controlled by the initialization of the algorithm, i.e., the farther the initial point w_0 is from the optimal solution w^*, the more iterations are needed.

Then, after applying the result in Proposition 1 to the natural unlearning model introduced in Section 2.2, we have the following proposition about federated unlearning rounds:

Proposition 2. *Consider the same conditions of Proposition 1 with diminishing step size $\bar{\eta}$ and suppose*

$$b(c) \leq \frac{1}{(I - I_{leave})\mu^2} \frac{\left(\sum_{i \in \mathcal{I}_{leave}} \|\nabla F_i(w^*)\|\right)^2}{\sum_{i \in \mathcal{I} \setminus \mathcal{I}_{leave}} \frac{1}{s_i}\sigma^2}.$$

It will require

$$T_{unlearn} \geq \frac{2(I-1)}{\epsilon^2\mu^2} \sum_{i \in \mathcal{I}_{leave}} \|\nabla F_i(w^*)\|^2 - 1 \tag{6}$$

rounds of communication to guarantee $\mathbb{E}\|w_{T_{unlearn}} - \tilde{w}^*\| \leq \epsilon$ when starting from the original learned model w^*, where the new model $\tilde{w}^*$ is defined in (3).

The proof of Proposition 2 is given in Appendix B in the technical report [5]. Each user's gradient $\|\nabla F_i(w^*)\|$ can represent his training loss (denoted as ℓ_i) because the calculated gradient increases in the loss. Hence, Proposition 2 reveals the relationship between the number of communication rounds required for federated unlearning and the training losses of leaving users. As indicated in (6), a larger total training loss of the leaving users $\sum_{i \in \mathcal{I}_{leave}} \ell_i^2$ (i.e., a

[1] This model treats each user equally. Some papers (e.g., [13]) adopted another objective, a weighted sum of all users' losses, where the weights (i.e. $d_i / \sum_{i=1}^{I} d_i$) reflect the differences in data size. The two objectives are equivalent when users' data sizes are the same. Our results can be easily extended to the weighted case.

[2] The distribution is due to the randomness in the training process (e.g. randomly sampled data and random ordering of batches).

[3] We can derive similar results for Fedavg if we additionally assume that local data distributions satisfy bounded heterogeneity.

[4] To estimate the true gradient $\nabla F_i(w)$, we uniformly sample one data point to generate a gradient estimate $g_i(w)$ and assume $\mathbb{E}\|g_i(w) - \nabla F_i(w)\|^2 \leq \sigma^2$ for any w.

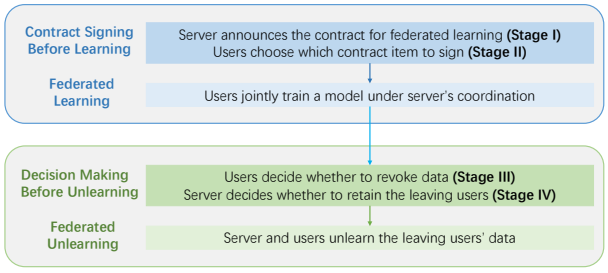

Figure 1: Framework of federated learning and unlearning system with incentive mechanisms.

larger $\sum_{i\in\mathcal{I}_{leave}} \|\nabla F_i(w^*)\|^2$) requires more communication rounds $T_{unlearn}$ to achieve unlearning.

We will apply the derived results about model accuracy loss and unlearning rounds in building the system model in the next section.

3 SYSTEM MODEL

We consider a federated learning and unlearning system consisting of a set of heterogeneous users with private data and a central server. As illustrated in Fig. 1, the server first incentivizes users as workers to participate in a federated learning phase through a contract. However, some users may later choose to revoke their data and leave the system. In response, the server can provide further incentives to retain valuable users. Upon the final exit of some users from the system, the remaining users collectively execute an algorithm to unlearn the leaving users' data.

In the following, we first divide the heterogeneous users into different types for the convenience of incentive design, then formulate a multi-stage game between the strategic server and users, and finally specify the payoffs of the server and the users (i.e., their optimization objectives), respectively.

3.1 User Type

We consider a set $\mathcal{I} \triangleq \{1, 2, ..., I\}$ of users in the system with two-dimensional private information: marginal cost for training effort θ and marginal perceived privacy cost ξ. We refer to a user with (θ_j, ξ_j) as a type j user. We further assume that the I users belong to a set $\mathcal{J} \triangleq \{1, 2, ..., J\}$ of J types. Each type j has I_j users, with $\sum_{j\in\mathcal{J}} I_j = I$. The total number of users I and the number of each type I_j are public information, but each user's specific type is private information.[5]

Under private information, it is difficult for the server to predict users' strategies. To this end, we propose to design a contract mechanism for the server to elicit information.

3.2 Games and Strategies

We use a four-stage Stackelberg game to model the interaction between the server and users.

- Stage I: The server designs a federated learning incentive contract $\boldsymbol{\phi} \triangleq \{\phi_j\}_{j\in\mathcal{J}}$, which contains J contract items (one for each user type). Each contract item $\phi_j \triangleq \left(d_j, r_j^L\right)$ specifies the relationship between the required data size d_j of each type-j user (for local computation) and the corresponding learning reward r_j^L.

[5]The server can have knowledge about statistics of type information through market research and past experiences, but it is hard for it to know each user's private type.

Table 1: The Server and Users' Knowledge in Different Stages

Stage	Known	Unknown
Server in Stage I	$\mathcal{J}, \{I_j\}_{j\in\mathcal{J}}$	$\{\theta_i, \xi_i, \ell_i, v_i\}_{i\in\mathcal{I}}$
User in Stage II	his own type (θ_i, ξ_i)	other users' types, $\{\ell_i, v_i\}_{i\in\mathcal{I}}$
User in Stage III	his own type (θ_i, ξ_i), $\{\ell_i\}_{i\in\mathcal{I}}$	other users' types, $\{v_i\}_{i\in\mathcal{I}}$
Server in Stage IV	$\mathcal{J}, \{I_j\}_{j\in\mathcal{J}}$, $\{\theta_i, \xi_i, \ell_i, v_i\}_{i\in\mathcal{I}}$	\

- Stage II: Users decide which contract item to choose. Then, they jointly implement the federated learning algorithm (Scaffold).
- Stage III: Users decide whether to revoke data after federated learning. We denote a user i's revocation decision as
$$x_i = \begin{cases} 0, & \text{if user } i \text{ does not revoke data,} \\ 1, & \text{if user } i \text{ revokes his data,} \end{cases} \tag{7}$$
and denote the set of users who revoke their data as $\mathcal{I}_u$. If a type-j user revokes his data, then he needs to fully return the reward r_j^L to the server.[6] We consider that the server will announce users' training losses $\{\ell_i\}_{i\in\mathcal{I}}$ (without specifying users) after federated learning to help users decide whether to revoke data.[7]
- Stage IV: The server decides the set of leaving users to retain $\mathcal{I}_r$ and designs the corresponding retention incentives $\left\{r_i^U\right\}_{i\in\mathcal{I}_r}$, such that those receiving the retention incentives will choose to stay in the system and those without will leave.[8] The remaining users and server collectively implement federated unlearning.

In Stage III, we use $\ell_i = ||\nabla F_i(w_T)||$ to represent the training loss, where w_T is the solution obtained after T iterations of Scaffold. We assume T is large enough, such that w_T and w^* are close. A large ℓ_i implies the federated solution is far away from the minimizer of local loss function F_i and therefore a larger training loss.

After federated learning, the server and users have more information in Stages III and IV compared with Stages I and II. For example, the users will know their training losses $\{\ell_i\}_{i\in\mathcal{I}}$. The server can evaluate the users' contribution to the global model (denoted by $\{v_i\}_{i\in\mathcal{I}}$), and it will know each user's type by observing users' contract item choices. We summarize their knowledge about some key information in the four stages in Table 1 and list the key notations in this paper in Table 2.[9]

Moreover, in Stage IV, the server has enough information to know whether the users will accept the retention incentives. Therefore, we do not model a Stage V in which the users decide to accept or not accept the retention incentives. After that, as in Fig. 1, the staying users perform federated unlearning under the server's coordination, which makes staying users sustain unlearning costs. We will specify the payoffs and costs of the server and users in each stage of the game in the next subsection.

[6]If there is no such return policy, every user can first participate to get rewards and then revoke data to reduce costs, resulting in a catastrophic failure of model training collaboration and a huge cost to the server.

[7]It is not obvious that a strategic server would make such an announcement, but it can be stipulated by regulations for protecting users' right to be forgotten. If we do not make this assumption, the problem will be even simpler. As we shall see in the analysis in Section 4.2, we just need to replace other users' training losses $\{\ell_k\}_{k\in\mathcal{I}}$ in (19) with the same expected loss $\mathbb{E}[\ell]$ and solve the problem through a similar approach.

[8]In this case, $\mathcal{I}_u \backslash \mathcal{I}_r$ is the set of users who finally leave the system, and $\mathcal{I} \backslash (\mathcal{I}_u \backslash \mathcal{I}_r)$ is the set of users who finally stay.

[9]As analyzing the four-stage game is complicated, this paper does not model the information update in a fully Bayesian framework but specifies plausible beliefs that the players hold in each stage.

Table 2: Key Notations

θ_j	Marginal training cost of type-j users
ξ_j	Marginal perceived privacy cost of type-j users
I_j	Number of type-j users
$j/\mathcal{J}$	Index/Set of user types in the system
$i/\mathcal{I}$	Index/Set of users in the system
$\mathcal{I}_u$	Set of users who revoke their data in Stage III
$\mathcal{I}_r$	Set of users who are retained by the server in Stage IV
ϕ_j	Contract item designed for type-j users
d_j	Required data size for each type-j user in the contract
r_j^L	Learning reward for each type-j user in the contract
r_i^U	Unlearning reward (retention incentive) for user i
x_i	User i's data revocation decision
p_j	Historical revocation rate of type-j users
q_j	Historical retention rate of type-j users
T	Number of communication rounds of federated learning
λ	Coefficient related to unlearning communication rounds
ϱ	Coefficient related to expected accuracy loss
γ	Server's weight on incentive rewards
v_i	User i's contribution to global model accuracy
ℓ_i	User i's training loss (representing $\|\|\nabla F_i(x^*)\|\|$)

3.3 Payoffs

At each stage, every user or the server seeks to maximize his expected payoff (or minimize his expected cost) based on his current knowledge. As knowledge updates occur between stages, the payoffs of the users or the server (maximization or minimization objectives respectively) take different forms in each stage.

3.3.1 Server's Payoff in Stage I. The server's objective in Stage I is to minimize the sum of the expected accuracy loss of the global model and the expected total incentive rewards for users.

First, we specify the expected model accuracy loss, which depends on the data of users who finally stay in the system. Since the server cannot predict which users will leave and who to retain due to the lack of information in Stage I, it can only base its decision on user distribution expectations. Specifically, we assume that according to the historical experience and market statistics, the server knows the probability of a type-j user revoking his data (i.e., his revocation rate) p_j and the probability that a type-j user who wants to revoke data is retained (i.e., his retention rate) q_j, where p_j and q_j are independent. Following Proposition 1, we model the server's expected accuracy loss after federated unlearning as:

$$\frac{\varrho}{T}\sum_{j\in\mathcal{J}} I_j(1-p_j+p_jq_j)\frac{1}{d_j}, \tag{8}$$

where T is the number of communication rounds of federated learning, ϱ is a coefficient related to the sample variance, and $1-p_j+p_jq_j$ is the percentage of type j users remaining in the system in the end. This captures that the expected model accuracy loss decreases in the data sizes of all staying users.[10]

The server's payoff also includes the cost of all rewards it pays to users, which comprises the initial contract announced in stage I and incentives offered to encourage leaving users to remain in stage IV. If all users choose to participate in the contract and choose their corresponding contract items,[11] the expected total learning reward is $\sum_{j\in\mathcal{J}} I_j(1-p_j+p_jq_j)r_j^L$. Note that if a type-$j$ user successfully revokes his data, he needs to fully return the reward r_j^L to the server. The server's expected incentive for retaining leaving users is $\mathbb{E}[\sum_{i\in\mathcal{I}_r} r_i^U]$, which depends on p, q, and training losses and will be calculated through backward induction in Section 4.4.

Combining these terms, the server's expected cost in Stage I is

$$\begin{aligned} W^{s-1} =& \frac{\varrho}{T}\sum_{j\in\mathcal{J}} I_j(1-p_j+p_jq_j)\frac{1}{d_j} \\ &+\gamma\left(\sum_{j\in\mathcal{J}} I_j(1-p_j+p_jq_j)r_j^L+\mathbb{E}\Big[\sum_{i\in\mathcal{I}_r} r_i^U\Big]\right), \end{aligned} \tag{9}$$

where γ is how much weight the server puts on the incentive reward payments compared to the model accuracy loss. A smaller γ means that the server is less concerned about minimizing the incentive rewards and more concerned about reducing the accuracy loss.

3.3.2 Users' Payoffs in Stage II. In the overall game, there are three possible outcomes for a user (not revoke data, revoke and retained, revoke and not retained). However, in this stage, a user does not have enough information to know which outcome will realize, so he must calculate his expected payoff by considering three cases:

- *Case (a): not revoke.* With probability $1-p_j$, a type-j user will not revoke his data after federated learning. In this case, his expected payoff is the difference between the learning reward r_j^L and costs (including the learning cost, privacy cost, and unlearning cost):
$$U_{j,a}^{s-2} = r_j^L - \theta_j d_j T - \xi_j \mathbb{E}[\ell_j] d_j - \mathbb{E}\left[\theta_j d_j \lambda \sum_{i\in\mathcal{I}_u\setminus\mathcal{I}_r} \ell_i^2\right], \tag{10}$$
where $\theta_j d_j T$ is the total learning cost in T rounds. As we consider that each user's sampled data size in each local round is proportional to his total data size, the learning cost is linear in his data size d_j (e.g., [1, 4, 17]). Similarly, in the unlearning cost $\theta_j d_j \lambda \sum_{i\in\mathcal{I}_u\setminus\mathcal{I}_r} \ell_i^2$, the $\lambda \sum_{i\in\mathcal{I}_u\setminus\mathcal{I}_r} \ell_i^2$ models the number of communication rounds for unlearning, which increases in the leaving users' training losses (according to Proposition 2).[12] A type j user's perceived privacy cost $\xi_j\mathbb{E}[\ell_j]d_j$ increases in his expected training loss $\mathbb{E}[\ell_j]$ and data size d_j. As a high training loss ℓ_j reflects a large distance of user j's data from the average of other users' distribution, we use it to measure the uniqueness of a user. Thus, the model captures that the privacy cost increases in the uniqueness and size of one's training data (e.g., [3, 16]). As each user cannot know his exact training loss ℓ_j before federated learning, we assume that he estimates the expected loss using the public distribution (with mean $\mathbb{E}[\ell_j]$ and variance $D(\ell_j)$).
- *Case (b): revoke but retained.* With probability p_jq_j, a type-j user will revoke his data after federated learning but will be retained by the server through more incentives r_j^U. In this case, his expected payoff is the difference between total rewards (including both learning and unlearning incentives) and costs:

[10] As the server aims to incentivize users to contribute data in federated learning, we only model the impact of data sizes and omit the independent term about initial point w_0 in (5). Since we consider that users' batch sizes $\{s_i\}_{i\in\mathcal{I}}$ are in the same proportion to their data sizes $\{d_i\}_{i\in\mathcal{I}}$, it is equivalent to substitute s_i with d_i in (5).

[11] As we shall see in Section 4.4, we will design the contract to ensure that each user will participate (i.e., individual rationality) and choose the contract item designed for his type (i.e., incentive compatibility).

[12] We use the simplified model of (6) in Proposition 2 to capture the key relationship between the unlearning communication rounds $T_{unlearn}$ and leaving users' training losses (represented by $\|\nabla F_i(w^*)\|$).

$$U_{j,b}^{s-2} = r_j^L + \mathbb{E}\left[r_j^U\right] - \theta_j d_j T - \xi_j \mathbb{E}[\ell_j] d_j - \mathbb{E}\left[\theta_j d_j \lambda \sum_{i \in \mathcal{I}_u \setminus \mathcal{I}_r} \ell_i^2\right]. \quad (11)$$

The unlearning incentive r_j^U will be determined by the server in Stage IV based on users' training losses, contributions, and data revocation, which are unknown in this stage. Thus, each user can only calculate the expectation of the unlearning incentive.

- *Case (c): revoke and not retained.* With probability $p_j(1-q_j)$, a type-j user will revoke his data and will not be retained by the server, i.e., the user's data will be unlearned. The user needs to return the reward r_j^L to the server but will not incur any privacy cost or unlearning cost. In this case, his expected payoff is

$$U_{j,c}^{s-2} = -\theta_j d_j T, \quad (12)$$

which is the sunk training cost from federated learning.

In summary, a type-j user's expected payoff in Stage II is

$$U_j^{s-2} = (1-p_j) U_{j,a}^{s-2} + p_j q_j U_{j,b}^{s-2} + p_j (1-q_j) U_{j,c}^{s-2}. \quad (13)$$

If $U_j^{s-2} \geq 0$, the type-j user will choose to participate in the federated learning in Stage II.

3.3.3 Users' Payoffs in Stage III. After federated learning, each user i has knowledge about his training loss ℓ_i. If user i chooses not to revoke his data, his expected payoff in Stage III is (updating (10) in Case (a) with the realized training loss ℓ_i):

$$U_{i,a}^{s-3} = r_i^L - \theta_i d_i T - \xi_i \ell_i d_i - \mathbb{E}\left[\theta_i d_i \lambda \sum_{k \in \mathcal{I}_u \setminus \mathcal{I}_r} \ell_k^2\right]. \quad (14)$$

The reason for using expectation here is that users do not know the set of retained users $\mathcal{I}_r$ determined in Stage IV. Users' expected payoffs of Cases (b) and (c) in Stage III follow the same approach (i.e., updating (11) and (12) with the realized training loss ℓ_i).

Note that users of the same type may have different training losses and thus different payoffs, so the payoff in Stage III is user-specific instead of type-specific. Moreover, after some users leave, the remaining users' training losses may change as the global model will be updated. Since users cannot accurately predict their future expected loss even if they know all users' current losses, we assume that each user still approximates his future expected loss as equal to his current loss.

3.3.4 Server's Payoff in Stage IV. When some users want to leave the system, it is important for the server to know their contributions to the global model for retaining valuable users.

A fair and effective method to compute a user's contribution to a coalition is the Shapley value [24]. Wang et al. [21] introduced a related concept called federated Shapley value to evaluate each user's contribution in a federated learning setting. The federated Shapley value for user i, denoted as v_i, is calculated by the server during the federated learning process and is unknown to the users.

Once obtaining users' contributions (federated Shapley values), the server can calculate its realized cost in Stage IV. This cost is the sum of two factors: the realized accuracy loss, which is estimated by the sum of federated Shapley values of all users who remain in the system, and the realized incentives.

$$W^{s-4} = \sum_{i \in \mathcal{I} \setminus (\mathcal{I}_u \setminus \mathcal{I}_r)} v_i + \gamma \left(\sum_{i \in \mathcal{I} \setminus (\mathcal{I}_u \setminus \mathcal{I}_r)} r_i^L + \sum_{i \in \mathcal{I}_r} r_i^U \right). \quad (15)$$

The first term in (15) represents the model accuracy loss, the second is the learning reward paid to all remaining users for participation in federated learning, and the last term is the total retention incentive. The additivity property of federated Shapley values allows the server to compare all the possible sets of users to retain and find the optimal one. Note that a smaller federated Shapley value is better, as it means a larger contribution to the accuracy of the global model, and the federated Shapley values can be negative.

4 OPTIMAL INCENTIVE MECHANISM

In this section, we analyze an optimal incentive mechanism for federated learning and unlearning. Based on backward induction, we will derive the optimal strategies from Stage IV to Stage I in Sections 4.1-4.4, respectively.

4.1 Server's Retention Strategies in Stage IV

Given the server's contract $\boldsymbol{\phi}$ in Stage I, the users' contract item choices in Stage II, and the users' revocation decisions $\mathcal{I}_u$ in Stage III, the server needs to determine which users to retain $\mathcal{I}_r$ and the corresponding retention incentives $\{r_i^U\}_{i \in \mathcal{I}_r}$ in Stage IV.

As we discussed in Section 3.3.4, the server seeks to minimize the cost in (15) in Stage IV, which can be formulated as follows:

Problem 1 (Server's Optimization Problem in Stage IV).

$$\min \sum_{i \in \mathcal{I} \setminus (\mathcal{I}_u \setminus \mathcal{I}_r)} v_i + \gamma \left(\sum_{i \in \mathcal{I} \setminus (\mathcal{I}_u \setminus \mathcal{I}_r)} r_i^L + \sum_{i \in \mathcal{I}_r} r_i^U \right) \quad (16a)$$

$$\text{s.t. } r_i^U + r_i^L - \theta_i d_i T - \xi_i \ell_i d_i - \theta_i d_i \lambda \sum_{k \in \mathcal{I}_u \setminus \mathcal{I}_r} \ell_k^2 \geq -\theta_i d_i T, \forall i \in \mathcal{I}_r \quad (16b)$$

$$\text{var. } \mathcal{I}_r \subseteq \mathcal{I}_u, \{r_i^U\}_{i \in \mathcal{I}_r}. \quad (16c)$$

The constraint (16b) is to ensure that the retention incentives are enough to make the target users stay in the system. The left-hand side of the constraint is a user i's payoff after accepting the retention incentive (including unlearning reward, learning reward, learning cost, privacy cost, and unlearning cost), and the right-hand side is his payoff of not accepting (i.e., he has to return the learning reward to the server and only has sunk learning cost).

The following proposition presents the solution to Problem 1.

Proposition 3. *The server's optimal set of users to retain is*

$$\mathcal{I}_r^* = \arg \min_{\mathcal{I}_r \subseteq \mathcal{I}_u} \sum_{i \in \mathcal{I}_r} \left(v_i + \gamma \theta_i d_i \lambda \sum_{k \in \mathcal{I}_u \setminus \mathcal{I}_r} \ell_k^2 + \gamma \xi_i \ell_i d_i \right), \quad (17)$$

and the optimal retention incentives are

$$r_i^{U*} = \theta_i d_i \lambda \sum_{k \in \mathcal{I}_u \setminus \mathcal{I}_r^*} \ell_k^2 + \xi_i \ell_i d_i - r_i^L, \forall i \in \mathcal{I}_r^*. \quad (18)$$

The proof of Proposition 3 is given in Appendix C in the technical report [5]. Proposition 3 highlights a trade-off regarding the retention of users and their training losses. Users who have larger training losses incur higher privacy costs and thus require higher incentives to retain (indicated by $\gamma \xi_i \ell_i d_i$ in (17)). However, retaining such users also helps reduce the unlearning costs since the objective in (17) increases with the aggregated loss of the leaving users. Furthermore, the server has the incentive to retain users who contribute more to the model accuracy, which corresponds to

smaller values of v_i. Additionally, users with smaller marginal costs θ_i and ξ_i are also desirable to reduce unlearning incentives.[13]

4.2 Users' Revocation Decisions in Stage III

Considering the server's optimal retention strategies in Stage IV, each user i decides whether to revoke his data in Stage III given the information announced in Stages I and II.

Based on the server's optimal retention incentives (18) and the user's payoffs in Stage III (i.e., the updated (11) and (12) with realized losses), a user i's payoff after revoking data is $-\theta_i d_i T$, regardless of whether the user is retained by the server or not. Thus, user i's expected payoff in Stage III can be rewritten as

$$U_i^{s-3}(x_i; x_{-i}) = x_i\,(-\theta_i d_i T) + (1-x_i)\left[r_i^L - \theta_i d_i T - \xi_i \ell_i d_i - \theta_i d_i \lambda \sum_{k\in\mathcal{I}} x_k (1-q)\ell_k^2\right], \tag{19}$$

where $x_{-i} = \{x_k\}_{k\in\mathcal{I}\setminus\{i\}}$ is the revocation decisions of all users except user i and $q = \mathbb{E}[q_j]$ is the expected retention rate of all users, as users do not know each other's type.[14] As shown in (19), each user's payoff depends on the other users' revocation decisions, so users engage in a non-cooperative game in Stage III.

We formally define users' non-cooperative sub-game as follows.

Sub-Game 1 (Users' Revocation Sub-Game in Stage III).

- *Players: all users in set $\mathcal{I}$.*
- *Strategy space: each user $i \in \mathcal{I}$ decides whether to revoke his data, i.e., $x_i \in \{0,1\}$ (0: not revoke, 1: revoke).*
- *Payoff function: each user $i \in \mathcal{I}$ maximizes his payoff in* (19).

The following proposition characterizes the Nash equilibrium (NE) of Sub-Game 1:

Proposition 4. *Sub-Game 1 is a supermodular game, where pure NE exists but may not be unique. Algorithm 1 converges to one NE.*

Algorithm 1: Users' optimal revocation decisions

Input : $\{r_i^L, \xi_i, \ell_i, d_i, \theta_i\}_{i\in\mathcal{I}}, \lambda, q$
Output: Optimal revocation decisions $\{x_i^*\}_{i\in\mathcal{I}}$

1 Initialize $x_i^* \leftarrow 0, i \in \mathcal{I}$;
2 **while** $\exists x_i^* = 0$ & $r_i^L - \xi_i\ell_i d_i - \theta_i d_i \lambda(1-q)\sum_{k\in\mathcal{I}\setminus\{i\}} x_k \ell_k^2 < 0$ **do**
3 $\quad x_i^* \leftarrow 1, \forall i$ satisfying conditions in line 2;

The proof of Proposition 4 is given in Appendix D in the technical report [5]. Based on Algorithm 1, we can find the set of users who revoke data in one NE, i.e., $\mathcal{I}_u^* = \{i : x_i^* = 1, i \in \mathcal{I}\}$. Basically, Algorithm 1 corresponds to doing best response updates of the users starting from all users choosing not to revoke (i.e., 0). It is well known that for supermodular games, these updates will converge monotonically to a NE. Algorithm 1 will terminate within I iterations.[15] The resulting equilibrium strategies and insights will be illustrated through simulation in Section 5.2.1.

4.3 Users' Contract Item Choices in Stage II

Based on the analysis in Stages III and IV, a type-j user's expected payoff in Stage II (13) can be rewritten as:

$$U_j^{s-2} = (1-p_j) r_j^L - \kappa_j d_j, \tag{20}$$

where

$$\kappa_j \triangleq (1-p_j)\xi_j \mathbb{E}[\ell_j] + \theta_j T + \theta_j (1-p_j)\lambda \sum_{m\in\mathcal{J}} I_m p_m (1-q_m)\left(\mathbb{E}[\ell_m]^2 + D(\ell_m)\right), \tag{21}$$

and $D(\ell_m)$ is the variance of type-m users' training losses.

Each type-j user in Stage II will choose a contract item that gives him a maximum non-negative expected payoff, leading to the constraints that the server needs to consider in Stage I.

4.4 Server's Contract in Stage I

In Stage I, the server designs a contract to minimize its expected cost, considering the results in Stages II-IV.

When designing the contract, the server needs to ensure that each user achieves a non-negative payoff, so that the user will accept the corresponding contract item. Moreover, since the server does not know each user's type in Stage I, the server also needs to make a user choose the contract item intended for him (i.e., the user does not misreport his type).[16] In other words, a contract is feasible if and only if it satisfies Individual Rationality (IR) and Incentive Compatibility (IC) constraints:

Definition 1 (Individual Rationality). *A contract is individually rational if each type-j user receives a non-negative payoff by accepting the contract item $\phi_j = \left(d_j, r_j^L\right)$ intended for his type, i.e.,*

$$(1-p_j) r_j^L - \kappa_j d_j \geq 0, \forall j \in \mathcal{J}. \tag{22}$$

Definition 2 (Incentive Compatibility). *A contract is incentive compatible if each type-j user maximizes his own payoff by choosing the contract item $\phi_j = \left(d_j, r_j^L\right)$ intended for his type, i.e.,*

$$(1-p_j) r_j^L - \kappa_j d_j \geq (1-p_j) r_m^L - \kappa_j d_m, \forall j, m \in \mathcal{J}. \tag{23}$$

Considering the constraints in Definitions 1 and 2, the server in Stage I seeks to design the contract $\boldsymbol{\phi} = \{(d_j, r_j^L)\}_{j\in\mathcal{J}}$ to minimize its expected cost in (9), which is rewritten as follows after combining the results in Stages II-IV:

[13] Note that in (17), the server may not only include users with a negative value in the brackets, as retaining some users with positive values may reduce the server's objective through the aggregated losses. This is an integer programming problem with complexity $O(2^{I_u})$. When the number of leaving users I_u is large, the server can reduce the complexity by classifying the leaving users into several categories to retain, each category with similar contributions and costs.

[14] Here we use the historical retention rate q to calculate the expected payoffs instead of the retention rate obtained in Stage IV (i.e., $|\mathcal{I}_r^*|/|\mathcal{I}_u|$). This is because users do not know their federated Shapley values and cannot calculate $\mathcal{I}_r^*$. If they calculate the expectation $\mathbb{E}[\mathcal{I}_r^*]$ based on type statistics, according to (17), the result will be user type retention instead of user retention (e.g., retain all type-i users and not retain all type-j users regardless of different data distributions and losses of the same type of users), which is not true. Conversely, historical rates ranging between $[0,1]$ allow for more realistic partial retention of same-type users. Therefore, we assume that the users have a belief at this stage in the retention rate which is the same as the historical rate. In the following analysis in Stages I and II, we will also use the historical rates for calculating the expected cost/payoffs for similar reasons.

[15] We can also initialize all the users' decisions as 1 and check whether there exists a user who wants to change his action from 1 to 0 for payoff improvement. If the equilibrium is the same as that found by Algorithm 1, it is the unique NE, as Game 1 is a supermodular game.

[16] Revelation principle demonstrates that if a social choice function can be implemented by an arbitrary mechanism, then the same function can be implemented by an incentive-compatible-direct-mechanism (i.e. in which users truthfully report types) with the same equilibrium outcome. Thus, requiring IC will simplify the mechanism design without affecting optimality.

Problem 2.

$$\begin{aligned}\min \sum_{j\in\mathcal{J}}\Bigg(&\frac{\varrho I_j(1-p_j+p_jq_j)}{Td_j}+\gamma I_j(1-p_j)r_j^L\\&+\gamma I_jp_jq_j(\alpha\theta_j+\xi_j\mathbb{E}[\ell_j])d_j\Bigg),\\ \text{s.t. } &(1-p_j)r_j^L-\kappa_jd_j\geq 0,\forall j\in\mathcal{J},\\ &(1-p_j)r_j^L-\kappa_jd_j\geq(1-p_j)r_m^L-\kappa_jd_m,\forall j,m\in\mathcal{J},\\ \text{var. } &\left\{\left(d_j,r_j^L\right)\right\}_{j\in\mathcal{J}},\end{aligned}\tag{24}$$

where

$$\alpha\triangleq\lambda\sum_{j\in\mathcal{J}}I_jp_j(1-q_j)\left(\mathbb{E}[\ell_j]^2+D(\ell_j)\right).\tag{25}$$

Solving Problem 2 involves two challenges. First, users' multi-dimensional heterogeneity leads to a challenging multi-dimensional contract design for the server. We will simplify the analysis by summarizing users' multi-dimensional heterogeneity into several one-dimensional metrics, to guide the server's design of the optimal rewards and data sizes in the contract. Second, as the total number of IR and IC constraints is large (i.e., J^2), it is challenging to obtain the optimal contract directly. To overcome such a complexity issue, we transform the constraints into a smaller number of equivalent ones. Next, we first derive the server's optimal reward $\{r_j^{L*}(\boldsymbol{d})\}_{j\in\mathcal{J}}$ for any given data size $\boldsymbol{d}=\{d_j\}_{j\in\mathcal{J}}$ (Lemma 1) in Section 4.4.1. Then, we calculate the optimal data size $\boldsymbol{d}^*$ (Proposition 5 and Theorem 1) in Section 4.4.2.

4.4.1 Optimal Rewards in Contract. Without loss of generality, we assume that users are indexed in ascending order of

$$\begin{aligned}\pi_j&\triangleq\frac{\kappa_j}{1-p_j}\\&=\xi_j\mathbb{E}[\ell_j]+\frac{\theta_jT}{1-p_j}+\theta_j\lambda\sum_{m\in\mathcal{J}}I_mp_m(1-q_m)\left(\mathbb{E}[\ell_m]^2+D(\ell_m)\right),\end{aligned}$$

which can be regarded as a type-j user's aggregated marginal cost. That is,

$$\pi_1\leq\pi_2\leq\ldots\leq\pi_J.\tag{26}$$

The following Lemma 1 characterizes the server's optimal learning rewards for any feasible data size:

Lemma 1. *For any given data size $\boldsymbol{d}=\{d_j\}_{j\in\mathcal{J}}$ (even if it is not optimal), the unique optimal reward for a type j user is:*

$$r_j^{L*}(\boldsymbol{d})=\begin{cases}\pi_jd_j, & \text{if } j=J;\\ \pi_jd_j+\sum_{m=j+1}^{J}(\pi_m-\pi_{m-1})d_m, & \text{if } j=1,\ldots,J-1.\end{cases}\tag{27}$$

The proof of Lemma 1 is given in Appendix E in the technical report [5]. Lemma 1 indicates that all user types except the boundary type J will obtain positive expected payoffs (type-J users receive zero expected payoff), which can be interpreted as the *information rent* in economics due to information asymmetry.

4.4.2 Optimal Data Sizes in Contract. Based on Lemma 1, we can significantly simplify Problem 2 but still need to derive the optimal values of J variables $\{d_j\}_{j\in\mathcal{J}}$ under J constraints $d_1\geq\ldots\geq d_J\geq 0$.

For the convenience of presentation, we define

$$A_j\triangleq\frac{\varrho I_j(1-p_j+p_jq_j)}{T},\tag{28}$$

$$\begin{aligned}B_j\triangleq&\gamma I_j\left(p_jq_j\left(\alpha\theta_j+\xi_j\mathbb{E}[\ell_j]\right)+\left(1-p_j\right)\pi_j\right)\\&+\sum_{m=1}^{j-1}\gamma I_m(1-p_m)(\pi_j-\pi_{j-1}).\end{aligned}\tag{29}$$

Based on these two metrics, we first present two special cases of the optimal data sizes, which we call all-independent and all-dependent.

Proposition 5. *Two special cases of the optimal data sizes follow:*

- All-independent. *If*

$$\frac{A_1}{B_1}\geq\frac{A_2}{B_2}\geq\ldots\geq\frac{A_J}{B_J},\tag{30}$$

then the optimal data sizes in the contract are

$$d_j^*=\sqrt{\frac{A_j}{B_j}},j\in\mathcal{J}.\tag{31}$$

- All-dependent. *If*

$$\frac{\sum_{m\in\mathcal{J}}A_m}{\sum_{m\in\mathcal{J}}B_m}>\frac{\sum_{m=1}^{j}A_m}{\sum_{m=1}^{j}B_m},\forall j=1,2,\ldots,J-1,\tag{32}$$

then the optimal data sizes in the contract are

$$d_j^*=\sqrt{\frac{\sum_{m\in\mathcal{J}}A_m}{\sum_{m\in\mathcal{J}}B_m}},j\in\mathcal{J}.\tag{33}$$

The proof of Proposition 5 is given in Appendix F in the technical report [5]. The all-independent case means that if $\{A_j/B_j\}_{j\in\mathcal{J}}$ follow a descending order, then the optimal data size for each type-j user only depends on his own parameters (A_j,B_j). The condition for the all-dependent case means that for any type j, there always exists at least one type $m>j$ with A_m/B_m larger than A_j/B_j (i.e., not in descending order). In this case, each type's optimal data size depends on all types' parameters $\{(A_j,B_j)\}_{j\in\mathcal{J}}$.

Next, we give an efficient algorithm to compute the optimal data sizes in any possible case based on the insights in Proposition 5.

Theorem 1. *For a fixed J, there are 2^{J-1} possible cases of the optimal data sizes depending on the values of $\{(A_j,B_j)\}_{j\in\mathcal{J}}$. For any given $\{(A_j,B_j)\}_{j\in\mathcal{J}}$, the unique optimal data sizes can be calculated by Algorithm 2.*

The proof of Theorem 1 is given in Appendix G in the technical report [5]. The computation complexity of Algorithm 2 is $O(\sum_{x=1}^{X}J_x)$, which is no larger than $O(J)$. We can interpret Algorithm 2 as greedily merging non-descending types based on A_j/B_j, so that all merged types have $\sum_j A_j/\sum_j B_j$ in a descending order.[19] The optimal data sizes of the merged types are the same and follow the dependent form (33) in Proposition 5, while the optimal data sizes of the not-merged types follow the independent form (31), as illustrated in footnote 17.

[16] For example, if $\frac{A_1}{B_1}\geq\frac{A_4}{B_4}\geq\frac{A_3}{B_3}\geq\frac{A_2}{B_2}\geq\frac{A_5}{B_5}\geq\frac{A_8}{B_8}\geq\frac{A_6}{B_6}\geq\frac{A_7}{B_7}$, then $X=2$, $\mathcal{J}_1=\{2,3,4\}$, and $\mathcal{J}_2=\{6,7,8\}$.

[17] In the example of footnote 16. If $\frac{A_6}{B_6}\geq\frac{A_7+A_8}{B_7+B_8}$, then $\mathcal{J}_2$ can be divided into two subsets $\{6\}$ and $\{7,8\}$. The optimal data sizes in this example are $d_j^*=\sqrt{\frac{A_j}{B_j}}$, $j=1,5,6$, $d_j^*=\sqrt{\frac{A_2+A_3+A_4}{B_2+B_3+B_4}}$, $j=2,3,4$, and $d_j^*=\sqrt{\frac{A_7+A_8}{B_7+B_8}}$, $j=7,8$.

[19] In the example of footnotes 16 and 17, $\frac{A_1}{B_1}\geq\frac{A_2+A_3+A_4}{B_2+B_3+B_4}\geq\frac{A_5}{B_5}\geq\frac{A_6}{B_6}\geq\frac{A_7+A_8}{B_7+B_8}$.

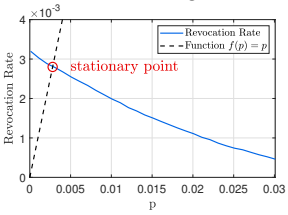

Figure 2: Revocation rate $|\mathcal{I}_u^*|/I$ versus historical revocation rate p.

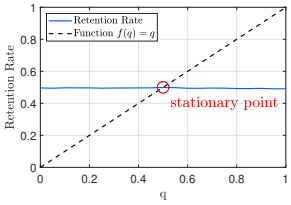

Figure 3: Retention rate $|\mathcal{I}_r^*|/|\mathcal{I}_u^*|$ versus historical revocation rate q.

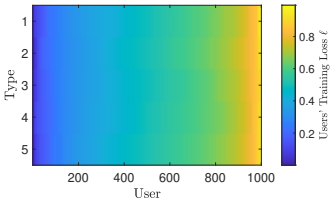

Figure 4: Users' training losses $\{\ell_i\}_{i\in\mathcal{I}}$.

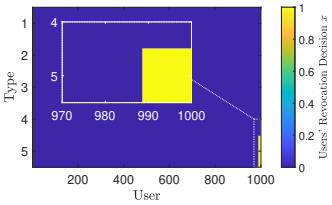

Figure 5: Users' optimal revocation decisions $\{x_i^*\}_{i\in\mathcal{I}}$.

Algorithm 2: Optimal data sizes in contract

Input : Parameters $\{(A_j, B_j)\}_{j\in\mathcal{J}}$ indexed based on (26)
Output: Optimal data sizes $\{(d_j^*)\}_{j\in\mathcal{J}}$

Initialize $d_j^* \leftarrow \sqrt{\frac{A_j}{B_j}}, j \in \mathcal{J}$;
Find all non-descending types $\{j : \exists m > j, \frac{A_m}{B_m} > \frac{A_j}{B_j}$ or $\exists m < j, \frac{A_m}{B_m} < \frac{A_j}{B_j}\}$;
Put each group of non-descending types that have adjacent indexes into one auxiliary set $\mathcal{J}_x$;
$X \leftarrow$ the number of these auxiliary sets;[17] // i.e., $x \in \{1, 2, ..., X\}$
for $x = 1; x \le X; x++$ **do**
  check($\mathcal{J}_x$);[18] // divide each auxiliary set $\mathcal{J}_x$ into subsets $\{\mathcal{J}_x^y\}$ that satisfy (32)
Function check($\mathcal{J}$):
  Reindex the types in $\mathcal{J}$ with $1_\mathcal{J}, 2_\mathcal{J}, ..., J_\mathcal{J}$.
  if $|\mathcal{J}| \neq 1$ **then**
    flag←1;
    for $m = 1_\mathcal{J}$ *to* $(J-1)_\mathcal{J}$ **do**
      if $\frac{\sum_{j\in\mathcal{J}} A_j}{\sum_{j\in\mathcal{J}} B_j} \le \frac{\sum_{j=1_\mathcal{J}}^{m} A_j}{\sum_{j=1_\mathcal{J}}^{m} B_j}$ // $\mathcal{J}$ does not satisfy (32) **then**
        flag←0;
        $d_j^* = \sqrt{\frac{\sum_{n=1_\mathcal{J}}^{m} A_n}{\sum_{n=1_\mathcal{J}}^{m} B_n}}, j \in \{1_\mathcal{J}, ..., m\}$;
        check($\{m+1, ..., J_\mathcal{J}\}$);
        break;
    if *flag=1* **then**
      $d_j^* = \sqrt{\frac{\sum_{m\in\mathcal{J}} A_m}{\sum_{m\in\mathcal{J}} B_m}}, j \in \mathcal{J}$;// $\mathcal{J}$ satisfies (32)

5 SIMULATIONS

In this section, we use simulations to evaluate the performance of our mechanism. In Section 5.1, we specify our experiment setting. In Section 5.2, we validate the users' and the server's optimal strategies and compare our mechanism with state-of-the-art benchmarks.

5.1 Experiment Setting

We consider $J = 5$ types of users with marginal training costs $\boldsymbol{\theta} = [1, 4, 6, 9, 10]$ and marginal perceived privacy costs $\boldsymbol{\xi} = [0.8, 1.7, 1.4, 2.2, 1.2] \times 10^3$.[20] Each type has $I_j = I/J = 1000$ users. Heterogeneous users' training losses follow a truncated normal distribution $N(0.5, 0.2)$ over the support $[0, 1]$, and users' federated Shapley values follow a normal distribution $N(5\times10^{-5}, 0.04)$.[21] Users perform $T = 100$ rounds of federated learning, and the unlearning rounds coefficient $\lambda = 4$. The server's accuracy loss coefficient $\varrho = 1$ and its weight on the incentives $\gamma = 10^{-10}$ (to balance different units of incentives and model accuracy loss).

We perform experiments to find the appropriate values of historical revocation rate p and retention rate q. As shown in Fig. 2 and Fig. 3, when we set different values of p and q, both the realized revocation rate $|\mathcal{I}_u^*|/I$ and retention rate $|\mathcal{I}_r^*|/|\mathcal{I}_u^*|$ at the equilibrium have a stationary point, i.e., $(2.8\times10^{-3}, 2.8\times10^{-3})$ in Fig. 2 and $(0.5, 0.5)$ in Fig. 3, respectively. Therefore, we take the historical revocation rate $p = 0.28\%$ and the historical retention rate $q = 50\%$ in the following simulations.

5.2 Experiment Results

5.2.1 Users' Revocation Decisions and Server's Retention Decision. As shown in Fig. 4, we rank each type of users in ascending order of their training losses for the convenience of presenting insights.

Fig. 5 shows that at the equilibrium, users with larger aggregated marginal costs π (i.e., type 5) and training losses ℓ (i.e., user 986-1000) are more likely to revoke their data. This is because (i) users with larger costs receive smaller learning incentives from the server in the contract (Lemma 1); (ii) they do not know their high training losses before federated learning and their realized privacy costs (training losses) significantly exceed their expectations.

Fig. 6 illustrates the server's optimal retention decision. We rank the users who want to revoke their data in ascending order of their federated Shapley values $\{v_i\}_{i\in\mathcal{I}_u}$. Users with smaller federated Shapley values are more likely to be retained by the server, as smaller Shapley values represent larger contributions to the global model accuracy. Users with smaller training losses have lower privacy costs and may require fewer incentives from the server, compared to users with larger losses. However, Fig. 6 shows that the server does not necessarily retain users with smaller training losses. This is because given a fixed set of users, reducing the total training losses of retained users means increasing the total losses of leaving users, resulting in higher unlearning costs (Proposition 3).

5.2.2 Comparison with Benchmarks. We compare our incentive mechanism with two benchmarks to evaluate the performance.

- No Retention Incentive (NRI): the server does not retain users who want to revoke their data.
- Limited Look Ahead (LLA) (adapted from [4]): the server first optimizes the incentive mechanism for federated learning without considering the unlearning part, and then designs the retention incentive in unlearning (i.e., separate optimization).

[20] Different orders of magnitude are to balance different units of users' training costs and privacy costs.

[21] In future work, we will use real-world datasets to calculate users' true training losses and federated Shapley values. The simulation data here can also demonstrate our results. As in Appendix H in the technical report [5], we further validate that if we change the simulation setting, we will obtain similar experiment results and insights.

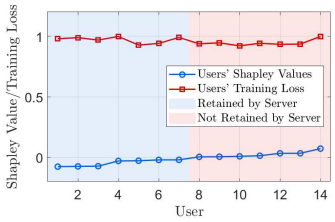

Figure 6: Server's optimal retention decisions $\mathcal{I}_r^*$.

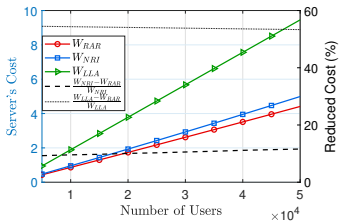

Figure 7: Server's cost comparison of NRI, LLA, and RAR.

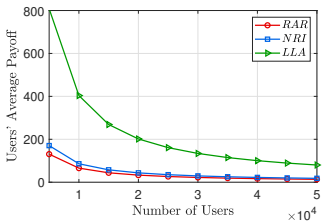

Figure 8: Users' average payoff comparison of NRI, LLA, and RAR.

- Our proposed incentive mechanism (RAR): the server is Rational in jointly optimizing both federated learning and unlearning And designs Retention incentive to retain valuable leaving users.

Fig. 7 shows the server's costs in the three mechanisms under different numbers of users. Our proposed RAR reduces the server's cost by around 53.91% (black dotted line) compared with LLA. The reduced cost of RAR compared with NRI can reach 11.59% (black dashed line) and will increase in the number of users, as the server retains more valuable users when the number of users increases. Therefore, it is beneficial for the server to retain valuable leaving users and make joint optimization of federated learning and unlearning incentive mechanisms. As the objective of our incentive mechanism design is to minimize the server's cost, the server's cost reduction is at the expense of users' payoffs (as shown in Fig. 8).

6 CONCLUSION

To the best of our knowledge, this paper is the first study to focus on the important issue of incentive design for federated learning and unlearning. We derive theoretical bounds on the global model optimality gap of Scaffold and the number of communication rounds of natural federated unlearning. Our approach tackles a challenging problem in incentive design, by summarizing users' multi-dimensional heterogeneity into one-dimensional metrics and developing an efficient algorithm for an exponentially large number of possible cases. We identify what types of users will leave the system or be retained by the server. The experiments demonstrate the superior performance of our proposed incentive mechanism and the benefits of unlearning incentives for retaining leaving users. We will design incentive mechanisms for federated unlearning to maximize social welfare in future work.

ACKNOWLEDGEMENT

This work was supported in part by the National Science Foundation under grant ECCS-2030251.

REFERENCES

[1] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning. In *IEEE Symposium on Security and Privacy (SP)*.

[2] Yinzhi Cao and Junfeng Yang. 2015. Towards making systems forget with machine unlearning. In *IEEE Symposium on Security and Privacy*.

[3] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. 2013. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports* 3, 1 (2013), 1–5.

[4] Ningning Ding, Zhixuan Fang, and Jianwei Huang. 2020. Optimal contract design for efficient federated learning with multi-dimensional private information. *IEEE Journal on Selected Areas in Communications* 39, 1 (2020), 186–200.

[5] Ningning Ding, Zhenyu Sun, Ermin Wei, and Randall Berry. 2023. Techinical Report. https://www.dropbox.com/s/f2b2z8t9kajs5ax/Appendix.pdf?dl=0.

[6] Xiangshan Gao, Xingjun Ma, Jingyi Wang, Youcheng Sun, Bo Li, Shouling Ji, Peng Cheng, and Jiming Chen. 2022. VeriFi: Towards Verifiable Federated Unlearning. *arXiv preprint arXiv:2205.12709* (2022).

[7] Elizabeth Liz Harding, Jarno J Vanto, Reece Clark, L Hannah Ji, and Sara C Ainsworth. 2019. Understanding the scope and impact of the California Consumer Privacy Act of 2018. *Journal of Data Protection & Privacy* 2, 3 (2019), 234–253.

[8] Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. 2019. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488* (2019).

[9] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*.

[10] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2019. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189* (2019).

[11] Gaoyang Liu, Xiaoqiang Ma, Yang Yang, Chen Wang, and Jiangchuan Liu. 2021. Federaser: Enabling efficient client-level data removal from federated learning models. In *2021 IEEE/ACM 29th International Symposium on Quality of Service*.

[12] Yang Liu, Zhuo Ma, Ximeng Liu, and Jianfeng Ma. 2020. Learn to forget: User-level memorization elimination in federated learning. *arXiv preprint arXiv:2003.10933* (2020).

[13] H B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Artificial Intelligence and Statistics*.

[14] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. 2021. A survey on security and privacy of federated learning. *Future Generation Computer Systems* 115 (2021), 619–640.

[15] Reese Pathak and Martin J Wainwright. 2020. FedSplit: An algorithmic framework for fast federated optimization. *Advances in Neural Information Processing Systems* 33 (2020), 7057–7066.

[16] Daniele Romanini, Sune Lehmann, and Mikko Kivelä. 2021. Privacy and uniqueness of neighborhoods in social networks. *Scientific reports* 11, 1 (2021), 20104.

[17] N. Tran, W. Bao, Al. Zomaya, and C. Hong. 2019. Federated Learning over Wireless Networks: Optimization Model Design and Analysis. In *IEEE Conference on Computer Communications (INFOCOM)*.

[18] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou. 2019. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*.

[19] Paul Voigt and Axel Von dem Bussche. 2017. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10, 3152676 (2017), 10–5555.

[20] Jianxiao Wang, Haiwang Zhong, Junjie Qin, Wenyuan Tang, Ram Rajagopal, Qing Xia, and Chongqing Kang. 2019. Incentive mechanism for sharing distributed energy resources. *Journal of Modern Power Systems and Clean Energy* 7, 4 (2019), 837–850.

[21] Tianhao Wang, Johannes Rausch, Ce Zhang, Ruoxi Jia, and Dawn Song. 2020. A principled approach to data valuation for federated learning. In *Federated Learning*. Springer, 153–167.

[22] Weina Wang, Lei Ying, and Junshan Zhang. 2016. The value of privacy: Strategic data subjects, incentive mechanisms and fundamental limits. In *ACM International Conference on Measurement and Modeling of Computer Science*.

[23] Zhiyuan Wang, Lin Gao, and Jianwei Huang. 2022. Socially-optimal mechanism design for incentivized online learning. In *IEEE Conference on Computer Communications (INFOCOM)*.

[24] Eyal Winter. 2002. The shapley value. *Handbook of game theory with economic applications* 3 (2002), 2025–2054.

[25] Chen Wu, Sencun Zhu, and Prasenjit Mitra. 2022. Federated Unlearning with Knowledge Distillation. *arXiv preprint arXiv:2201.09441* (2022).

[26] Hong Xie and John CS Lui. 2014. Modeling crowdsourcing systems: Design and analysis of incentive mechanism and rating system. *ACM SIGMETRICS Performance Evaluation Review* 42, 2 (2014), 52–54.

[27] Yufeng Zhan, Peng Li, Zhihao Qu, Deze Zeng, and Song Guo. 2020. A learning-based incentive mechanism for federated learning. *IEEE Internet of Things Journal* (2020).

[28] Meng Zhang, Ermin Wei, and Randall Berry. 2021. Faithful edge federated learning: Scalability and privacy. *IEEE Journal on Selected Areas in Communications* 39, 12 (2021), 3790–3804.

[29] Nan Zhao, Ying-Chang Liang, and Yiyang Pei. 2018. Dynamic contract incentive mechanism for cooperative wireless networks. *IEEE transactions on vehicular technology* 67, 11 (2018), 10970–10982.