ULDP-FL: Federated Learning with Across-Silo User-Level Differential Privacy

Fumiyuki Kato*
Kyoto University
fumilemon79@gmail.com

Li Xiong
Emory University
lxiong@emory.edu

Shun Takagi
Kyoto University
shun021677@gmail.com

Yang Cao
Tokyo Institute of Technology
cao@c.titech.ac.jp

Masatoshi Yoshikawa
Osaka Seikei University
yoshikawa-mas@osaka-seikei.ac.jp

ABSTRACT

Differentially Private Federated Learning (DP-FL) has garnered attention as a collaborative machine learning approach that ensures formal privacy. Most DP-FL approaches ensure DP at the record-level within each silo for cross-silo FL. However, a single user's data may extend across multiple silos, and the desired user-level DP guarantee for such a setting remains unknown. In this study, we present ULDP-FL, a novel FL framework designed to guarantee user-level DP in cross-silo FL where a single user's data may belong to multiple silos. Our proposed algorithm directly ensures user-level DP through per-user weighted clipping, departing from group-privacy approaches. We provide a theoretical analysis of the algorithm's privacy and utility. Additionally, we improve the utility of the proposed algorithm with an enhanced weighting strategy based on user record distribution and design a novel private protocol that ensures no additional information is revealed to the silos and the server. Experiments on real-world datasets show substantial improvements in our methods in privacy-utility trade-offs under user-level DP compared to baseline methods. To the best of our knowledge, our work is the first FL framework that effectively provides user-level DP in the general cross-silo FL setting.

PVLDB Reference Format:
Fumiyuki Kato, Li Xiong, Shun Takagi, Yang Cao, and Masatoshi Yoshikawa. ULDP-FL: Federated Learning with Across-Silo User-Level Differential Privacy. PVLDB, 17(11): 2826 - 2839, 2024.
doi:10.14778/3681954.3681966

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Emory-AIMS/uldp-fl.

1 INTRODUCTION

Federated Learning (FL) [35] is a collaborative machine learning (ML) scheme in which multiple parties train a single global model without sharing training data. FL has attracted industry attention [43, 44] as concerns about the privacy of training data have become more serious, as exemplified by GDPR [19]. It should be noted that FL itself does not provide rigorous privacy protection for the trained model [41, 53]. *Differentially Private FL* (DP-FL) [16, 22] further guarantees a formal privacy for trained models based on differential privacy (DP) [12].

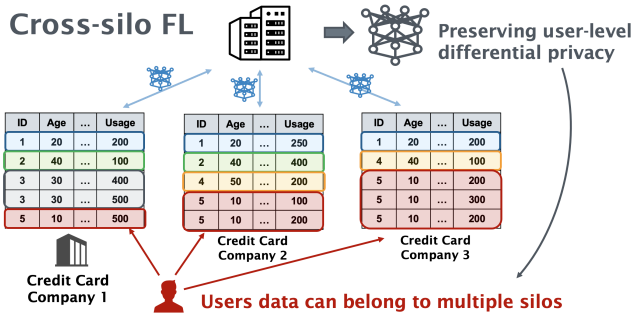


Figure 1: In cross-silo FL, records belonging to the same user can exist across silos, e.g., a user can use several credit card companies. In this study, we investigate how to train models satisfying *user-level* DP in this setting.

Although DP is the de facto standard in the field of statistical privacy protection, it has a theoretical limitation. The standard DP definition takes a single record as a unit of privacy. This can easily break down in realistic settings where one user may provide multiple records, theoretically deteriorating the privacy loss bound of DP. This can be intuitively understood from the following example. User-level DP effectively protects privacy by restricting the influence of multiple records from a single user (e.g., multiple transactions in credit card history or one patient is diagnosed with the same disease at multiple hospitals), preventing the disclosure of his/her distinctive patterns, i.e., features and/or trends. This contrasts with record-level DP, which falls short in addressing the cumulative privacy risks associated with aggregating these records. To address this, the notion of *user-level DP* has been studied [4, 28, 31, 49]. In user-level DP, all records belonging to a single user are considered as a unit of privacy, which is a stricter definition than standard DP. We distinguish user-level DP from *group-privacy* [13], which considers any k records as privacy units. User-level DP has also been studied in the FL context [15, 16, 22, 34, 36]. However, these studies focus on the cross-device FL setting, where one user's data belongs to a single device only.

*Work partially done while visiting Emory University.

Proceedings of the VLDB Endowment, Vol. 17, No. 11 ISSN 2150-8097.
doi:10.14778/3681954.3681966

Cross-silo FL [30, 32, 33, 42] is a practical variant of FL in which a relatively small number of silos (e.g., hospitals or credit card companies) participate in training rounds. In cross-silo FL, unlike in cross-device FL, a single user can have multiple records across silos, as shown in Figure 1. Existing cross-silo DP-FL studies [30, 32, 33] have focused on record-level DP for each silo; user-level DP across silos has not been studied. Therefore, an important research question arises: *How do we design an FL framework that guarantees user-level DP across silos in cross-silo FL?*

A baseline solution for guaranteeing user-level DP is to bound user contributions (number of records per user) as in [4, 49] and then utilize group-privacy property of DP [13]. Group-privacy simply extends the indistinguishability of the record-level DP to multiple records. We can convert any DP algorithm to group-privacy version of DP (see Lemma 5, 6 later), which we formally define as Group DP (GDP). However, this approach can be impractical due to the super-linear privacy bound degradation of conversion to GDP and the need to appropriately limit the maximum number of user records (group size) in a distributed environment. In particular, the former issue is a fundamental limitation for DP and highlights the need to develop algorithms that directly satisfy user-level DP without requiring conversion to GDP.

In this study, we present ULDP-FL, a novel cross-silo FL framework, designed to directly ensure user-level DP through per-user weighted clipping and an effective weighting strategy. Additionally, we propose a novel private weighted aggregation protocol for implementing the weighting algorithm. This protocol ensures the private information of each silo is protected from both the server and other silos. The limitation of previous methods that relied solely on the Paillier cryptosystem for private weighted summation [3] is that the raw data is visible to the party with the secret key. We overcome this limitation by using a combination of several cryptographic techniques such as Paillier, Secure Aggregation [7], and Multiplicative blinding [9]. To the best of our knowledge, our work is the first FL framework that effectively provides user-level DP across silos in the general cross-silo FL setting (as in Figure 1).

The contributions of this work are summarized as follows:

- We propose the ULDP-FL framework and design baseline algorithms capable of achieving user-level DP across silos. The baseline algorithms limit the maximum number of records per user and use group-privacy property of DP.
- We then propose ULDP-AVG/SGD algorithms that directly satisfy user-level DP by implementing user-level weighted clipping within each silo, effectively bounding user-level sensitivity even when a single user may have unbounded number of records across silos. We provide theoretical analysis, including privacy bound and convergence analysis.
- We evaluate our proposed method and baseline approaches through comprehensive experiments on various real-world datasets. The results underscore that our proposed method yields superior trade-offs between privacy and utility compared to the baseline approaches.
- We further design an effective method by refining the weighting strategy for individual user-level clipping bounds. Since this approach may lead to additional privacy leaks, we develop a novel private weighted aggregation protocol employing several cryptographic techniques. We evaluate the extra computational overhead of the proposed private protocol using real-world benchmark scenarios.

2 BACKGROUND & PRELIMINARIES

2.1 Cross-silo Federated learning

In this work, we consider the following cross-silo FL scenario. We have a central aggregation server and a set of silos S participating in all rounds. In each round, the server aggregates models from all silos and then redistributes the aggregated models. Each silo $s \in S$ optimizes a local model f_s, which is the expectation of a loss function $F(x; \xi)$ that may be non-convex, where $x \in \mathbb{R}^d$ denotes the model parameters and ξ denotes the data sample, and the expectation is taken over local data distribution $\mathcal{D}_s$. In cross-silo FL, we optimize this global model parameter cooperatively across all silos. Formally, the overarching goal in FL can be formulated as follows:

$$\min_x \left\{ f(x) := \frac{1}{|S|} \sum_{s \in S} f_s(x) \right\}, \; f_s(x) := \mathbb{E}_{\xi \sim \mathcal{D}_s} F(x; \xi). \tag{1}$$

Additionally, in our work, we have user set U across all datasets across silos, where each record belongs to one user $u \in U$, and each user may have multiple records in one silo and across multiple silos. Each silo s has local objectives for each user u, $f_{s,u} := \mathbb{E}_{\xi \sim \mathcal{D}_{s,u}} F(x; \xi)$, where $\mathcal{D}_{s,u}$ is the data distribution of s and u. In round $t \in [T]$ in FL, the global model parameter is denoted as x_t.

Note that this modeling is clearly different from cross-device FL in that there is no constraint that one user should belong to one device. Records from one user can belong to multiple silos. For example, the same customer may use several credit card companies, etc. Additionally, all silos participate in all training rounds, unlike the probabilistic participation in cross-device FL [36], and the number of silos $|S|$ is small, around 2 to 100 [23].

2.2 Differential Privacy

Differential privacy. DP [12] is a rigorous mathematical privacy definition that quantitatively evaluates the degree of privacy protection for outputs.

DEFINITION 1 ((ϵ, δ)-DP). *A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{Z}$ satisfies (ϵ, δ)-DP if, for any two input databases $D, D' \in \mathcal{D}$ s.t. D' differs from D in at most one record and any subset of outputs $Z \subseteq \mathcal{Z}$, it holds that*

$$\Pr[\mathcal{M}(D) \in Z] \leq \exp(\epsilon) \Pr[\mathcal{M}(D') \in Z] + \delta. \tag{2}$$

We call databases D and D' as *neighboring* databases. The maximum difference of the output for any neighboring database is referred to as *sensitivity*. We label the original definition as *record-level* DP because the neighboring databases differ in only one record.

Rényi differential privacy. Rényi DP (RDP) [38] is a variant of approximate DP based on Rényi divergence. RDP is preferable because it is easy to use for *Gaussian mechanism* [38] and has a tighter bound than the standard composition theorems. The following lemmas give the bounds of the RDP for a typical mechanism and are used to further convert it to an original DP bound.

Definition 2 ((α, ρ)-RDP [38]). *Given a real number $\alpha \in (1, \infty)$ and privacy parameter $\rho \geq 0$, a randomized mechanism $\mathcal{M}$ satisfies (α, ρ)-RDP if for any two neighboring datasets $D, D' \in \mathcal{D}$ s.t. D' differs from D in at most one record, we have that $D_\alpha(\mathcal{M}(D)||\mathcal{M}(D')) \leq \rho$ where $D_\alpha(\mathcal{M}(D)||\mathcal{M}(D'))$ is the Rényi divergence between $\mathcal{M}(D)$ and $\mathcal{M}(D')$ and is given by*

$$D_\alpha(\mathcal{M}(D)||\mathcal{M}(D')) := \frac{1}{\alpha - 1} \log \mathbb{E}\left[\left(\frac{\mathcal{M}(D)}{\mathcal{M}(D')}\right)^\alpha\right] \leq \rho,$$

where the expectation is taken over the output of $\mathcal{M}(D)$.

Lemma 1 (RDP composition [38]). *If $\mathcal{M}_1$ satisfies (α, ρ_1)-RDP and $\mathcal{M}_2$ satisfies (α, ρ_2), then their composition $\mathcal{M}_1 \circ \mathcal{M}_2$ satisfies $(\alpha, \rho_1 + \rho_2)$-RDP.*

Lemma 2 (RDP to DP conversion [5]). *If $\mathcal{M}$ satisfies (α, ρ)-RDP, then it also satisfies (ρ', δ)-DP for any $0 < \delta < 1$ such that*

$$\rho' = \rho + \log\frac{\alpha - 1}{\alpha} - \frac{\log\delta + \log\alpha}{\alpha - 1}.$$

Lemma 3 (RDP Gaussian mechanism [38]). *If $f : D \to \mathbb{R}^d$ has $\ell 2$-sensitivity Δ_f, then the Gaussian mechanism $G_f(\cdot) := f(\cdot) + \mathcal{N}(0, I\sigma^2\Delta_f^2)$ is $(\alpha, \alpha/2\sigma^2)$-RDP for any $\alpha > 1$.*

Lemma 4 (RDP for sub-sampled Gaussian mechanism [48]). *Let $\alpha \in \mathbb{N}$ with $\alpha \geq 2$ and $0 < q < 1$ be a ratio of sub-sampling operation $Samp_q$. Let $G'_f(\cdot) := G_f \circ Samp_q(\cdot)$ be a sub-sampled Gaussian mechanism. Then, G'_f is $(\alpha, \rho'(\alpha, \sigma))$-RDP where*

$$\rho'(\alpha, \sigma) \leq \frac{1}{\alpha - 1} \log\left(1 + 2q^2\binom{\alpha}{2} \min\{2(e^{1/\sigma^2} - 1), e^{1/\sigma^2}\} + \sum_{j=3}^{\alpha} 2q^j \binom{\alpha}{j} e^{j(j-1)/2\sigma^2}\right).$$

In general, we can compute tighter numerical bounds along with the closed-form upper bounds described above [39, 48]. In particular, RDP computation frameworks such as Opacus [52] use analysis in [39] with Poisson sampling on records.

Group differential privacy. To extend privacy guarantees to multiple records, group-privacy [13] has been explored as a solution. We refer to the group-privacy version of DP as Group DP (GDP).

Definition 3 ((k, ϵ, δ)-GDP). *A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{Z}$ satisfies (k, ϵ, δ)-GDP if, for any two input databases $D, D' \in \mathcal{D}$, s.t. D' differs from D in at most k records and any subset of outputs $Z \subseteq \mathcal{Z}$, Eq. (2) holds.*

GDP is a versatile privacy definition, as it can be applied to existing DP mechanisms without modification.

To convert DP to GDP, it is known that any $(\epsilon, 0)$-DP mechanism satisfies $(k, k\epsilon, 0)$-GDP [13]. However, in the case of any $\delta > 0$, δ increases super-linearly [24], leading to a much larger ϵ.

Lemma 5 (Group privacy conversion (record-level DP to GDP) [24]). *If f is (ϵ, δ)-DP, for any two input databases $D, D' \in \mathcal{D}$ s.t. D' differs from D in at most k records and any subset of outputs $Z \subseteq \mathcal{Z}$, it holds that*

$$\Pr[f(D) \in Z] \leq \exp(k\epsilon) \Pr[f(D') \in Z] + ke^{(k-1)\epsilon}\delta.$$

It means when f is (ϵ, δ)-DP, f satisfies $(k, k\epsilon, k\exp^{(k-1)\epsilon}\delta)$-GDP.

Also, we can compute GDP using group-privacy property of Rényi DP [38]. First, we calculate the RDP of the algorithm, then convert it to group version of RDP, and subsequently to GDP.

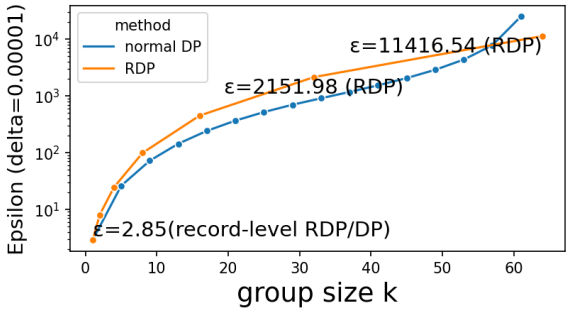


Figure 2: Group-privacy conversion results.

Lemma 6 (Group-privacy of RDP (record-level DP to GDP) [38]). *If $f : D \to \mathbb{R}^d$ is (α, ρ)-RDP, $g : D' \to D$ is k-stable and $\alpha \geq 2^{k+1}$, then $f \circ g$ is $(\alpha/2^k, 3^k\rho)$-RDP.*

Here, group-privacy property is defined using a notion of k-stable transformation [37]. $g : D' \to D$ is k-stable if $g(A)$ and $g(B)$ are neighboring in D implies that there exists a sequence of length $c+1$ so that $D_0 = A, ..., D_c = B$ and all (D_i, D_{i+1}) are neighboring in D'. This privacy notion corresponds to $(k, \cdot, \cdot)$-GDP in Definition 3.

Here, to highlight the significant privacy degradation of GDP, we conduct a pre-experiment and show the converted privacy bounds for increasing group sizes. Figure 2 illustrates a numerical comparison of the group-privacy conversion from DP to GDP with normal DP (Lemma 5) and RDP (Lemma 6). To compute the final GDP privacy bounds, we repeatedly run the Gaussian mechanism with $\sigma = 5.0$ and a sampling rate of 0.01 for 10^5 iterations, emulating a typical DP-SGD [1] (which is the common mechanism for DP ML) setup. We use a fixed $\delta = 10^{-5}$ and vary the group size $k = 1, 2, 4, 8, 16, 32, 64$. To compute GDP, RDP for sub-sampled Gaussian mechanisms is calculated according to [48]. We then compute GDP using group-privacy of RDP by Lemma 6. For normal DP, the computed RDP is converted to normal DP by Lemma 2, and then to GDP by Lemma 5.

When converting from normal DP to GDP, computing the final ϵ at a fixed δ is challenging. In Lemma 2, the output ϵ (denoted as ϵ_{l2}) depends on the input δ (denoted as δ_{l2}), and the final δ (denoted as δ_{l5}) output by Lemma 5 depends on both ϵ_{l2} and δ_{l2}. Therefore, we repeatedly select δ_{l2} in a binary search manner, compute ϵ_{l2} and δ_{l2}, and finally report the ϵ when the difference between δ_{l2} and $\delta = 10^{-5}$ is sufficiently small (accuracy by 10^{-8}) as the ϵ of GDP[1]. Note that this method does not guarantee achieving the optimal ϵ for the given δ, but it finds a reasonable ϵ.

In the figure, we plot various group sizes, k, on the x-axis and ϵ of GDP at a fixed $\delta = 10^{-5}$ on the y-axis. Significantly, the results indicate that as the group size k increases, ϵ grows rapidly, highlighting a considerable degradation in the privacy bound of GDP. For instance, with $\epsilon = 2.85$ at record-level ($k = 1$), the value reaches 2100 for only $k = 32$, and 11400 at $k = 64$. While there might be some looseness in the group-privacy conversion of RDP compared to normal DP for some small group sizes, the difference is relatively

[1]The implementation is in the function `get_normal_group_privacy_spent()` in https://github.com/Emory-AIMS/uldp-fl/blob/main/src/noise_utils.py

minor (roughly three times at most). The drastic change in normal DP with large group size is due to the numerical instability in our conversion procedure[1]. RDP's conversion is easier to compute with a fixed δ. Hence, we utilize RDP's conversion in the experiments.

2.3 Differentially Private FL

DP has been applied to the FL paradigm, with the goal of ensuring that the trained model satisfies DP. A popular DP variant in the context of cross-device FL is user-level DP (also known as client-level DP) [2, 16, 22]. Informally, this definition ensures indistinguishability for device participation and has demonstrated a favorable privacy-utility trade-off if the number of clients are sufficient, even with large-scale models [36]. These studies often employ secure aggregation [6, 7] to mitigate the need for trust in other parties during FL model training. This is achieved by allowing the server and other silos to only access appropriately perturbed models after aggregation, often referred to as Distributed DP [8, 22]. In particular, *shuffling*-based variants have recently gained attention [8, 29, 46] and are being deployed in FL [18], which also provides user-level DP. All of these studies assume that a single device holds all records for a single user, i.e., cross-device FL. However, in a cross-silo setting, this definition does not extend meaningful privacy protection to individual users when they possess multiple records across silos.

Another DP definition in cross-silo FL offers record-level DP within each silo [30, 32, 33], referred to as *Silo-specific sample-level* or *Inter-silo record-level DP*. These studies suggest that record-level DP can guarantee user-level DP through group-privacy. However, they cannot account for settings where a single user can have records across multiple silos. As far as we know, no method exists for training models that satisfy user-level DP in cross-silo FL where a single user's records can extend across multiple silos. More fine-grained comparison between DP variants in FL can be seen in the Appendix of our full paper [27].

3 ULDP-FL FRAMEWORK

3.1 Trust model and Assumptions

We assume that all (two or more) silos and an aggregation server are *semi-honest* (or *honest-but-curious*). This is a typical assumption in prior works [7, 45]. In our study, aggregation is performed using secure aggregation to ensure that the server only gains access to the model after aggregation [22]. All communications between the server and silos are encrypted with SSL/TLS, and third parties with the ability to snoop on communications cannot access any information except for the final trained model. We assume that there is no collusion, which is reasonable given that silos are socially separate institutions (such as different hospitals or companies). Additionally, in our scenario, we assume that record linkage [47] across silos has already been completed, resulting in shared common user IDs. Both the server and the silos are aware of the total number of users $|U|$ and the number of silos $|S|$. When user (device)-level sub-sampling is employed for DP amplification, only the server is permitted to know the sub-sampling results for each round [2, 22]. Note that all these assumptions do not affect the privacy guarantees of the final model released to external users.

3.2 Privacy definition

In contrast to GDP, which offers indistinguishability for any k records, user-level DP [16, 28] provides a more reasonable user-level indistinguishability regardless of the number of the records. While [16] focuses solely on a cross-device FL context, we re-establish user-level DP (ULDP) in the cross-silo setting as follows:

Definition 4 ((ϵ, δ)-User-Level DP (ULDP)). *A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{Z}$ satisfies (ϵ, δ)-ULDP if, for any two input databases across silos $D, D' \in \mathcal{D}$, s.t. D' differs from D in at most one user's records, and any $Z \subseteq \mathcal{Z}$, Eq. (2) holds.*

The fundamental difference of user-level DP from record-level DP lies in the definition of the neighboring databases. The user-level neighboring database inherently defines *user-level sensitivity*. Additionally, it is important to emphasize that the input database D represents the comprehensive database spanning across silos.

If the number of records per user in the database is less than or equal to k, it is clear that GDP is a generalization of ULDP, and the following proposition holds.

Proposition 1. *If a randomized mechanism $\mathcal{M}$ is (k, ϵ, δ)-GDP with input database D in which any user has at most k records, the mechanism $\mathcal{M}$ with input database D also satisfies (ϵ, δ)-ULDP.*

One drawback of GDP is the challenge of determining the appropriate value for k. Setting k to the maximum number of records associated with any individual user could lead to introducing excessive noise to achieve the desired privacy protection level. On the other hand, if a smaller k is chosen, the data of users with more than k records must be excluded from the dataset, potentially introducing bias and compromising model utility. In this context, while several studies have analyzed the theoretical utility for a given k [28, 31] and theoretical considerations for determining k have been partially explored in [4], it still remains an open problem. In contrast, ULDP does not necessitate the determination of k. Instead, it requires designing a specific ULDP algorithm.

3.3 Baseline methods: ULDP-NAIVE/GROUP

Table 1 summarizes symbols used in later algorithms in the paper. Please note that all omitted proofs for the following theorems can be found in the Appendix of the full paper [27].

ULDP-NAIVE. We begin by describing two baseline methods. The first method is ULDP-NAIVE (described in Algorithm 1), a straightforward approach using substantial noise. It works similarly to DP-FedAVG [36], where each silo locally optimizes with multiple epochs, computes the model update (delta), clips by C, and adds Gaussian noise. The original DP-FedAVG adds Gaussian noise with variance $\sigma^2 C^2$. In ULDP-NAIVE, since a single user may contribute to the model delta of all silos, the sensitivity across silos is $C|S|$ for the aggregated model delta, hence it needs to scale up the noise as $\sigma^2 C^2 |S|$ (Line 14) such that the aggregated result from $|S|$ silos satisfies required DP. Compared to DP-FedAVG, which focuses on cross-device FL, the number of model delta samples (number of silos as opposed to the number of devices) in our setting is very small, resulting in larger variance. Thus, ULDP-NAIVE satisfies ULDP but at a significant sacrifice in utility. The aggregation is performed using secure aggregation and is assumed to be so in the following algorithms.

Table 1: Notation Table for Algorithms.

Symbol	Description
η_l	Local learning rate.
η_g	Global learning rate.
σ	Noise parameter. (Noise multiplier.)
C	Clipping bound.
T	Total number of rounds.
Q	Number of local epochs.
$w_{s,u}$	Weight for user u in silo s.
$\mathbf{W}$	Matrix of weights for users and silos. $\mathbf{W} = (\mathbf{w}_1, ..., \mathbf{w}_{\lvert S\rvert})$: matrix with weight for user u and silo s, and $\forall u \in U$, $w_{s,u} \in \mathbf{w}_s$ and $\sum_{s\in S} w_{s,u} = 1$
x_0	Initial model.
x_t	Model at round t.
Δ_t^s	Model update from silo s at round t.
$g_{t,q}^{s,u}$	Stochastic gradient for user u in silo s at round t and epoch q.
$\Delta_t^{s,u}$	Model update for user u in silo s at round t.
$\tilde{\Delta}_t^{s,u}$	Clipped and weighted model update for user u in silo s at round t.
$g_t^{s,u}$	Stochastic gradient for user u in silo s at round t.
$\tilde{g}_t^{s,u}$	Clipped and weighted gradient for user u in silo s at round t.
$\mathcal{N}(0, b)$	Gaussian noise vector with mean 0 and variance b.
q	User-level sub-sampling probability.

Algorithm 1 ULDP-NAIVE

Input: η_l, η_g: local and global learning rates, σ: noise parameter, C: clipping bound, T: #rounds, Q: #local epochs

1: **procedure** SERVER
2: Initialize model x_0
3: **for** each round $t = 0, 1, \ldots, T-1$ **do**
4: **for** each silo $s \in S$ **do**
5: $\Delta_t^s \leftarrow \text{CLIENT}(x_t, C, \sigma, \eta_l)$
6: $x_{t+1} \leftarrow x_t + \eta_g \frac{1}{|S|} \sum_{s\in S} \Delta_t^s$
7: **procedure** CLIENT(x_t, C, σ, η_l)
8: $x_s \leftarrow x_t$
9: **for** epoch $q = 0, 1, \ldots, Q-1$ **do**
10: Compute stochastic gradients $g_{t,q}^{(s)}$ ▷ $\mathbb{E}[g_{t,q}^{(s)}] = \nabla f_s(x_s)$
11: $x_s \leftarrow x_s - \eta_l g_{t,q}^{(s)}$
12: $\Delta_t \leftarrow x_t - x_s$
13: $\tilde{\Delta}_t \leftarrow \Delta_t \cdot \min\left(1, \frac{C}{\|\Delta_t\|_2}\right)$ ▷ clipping with C
14: $\Delta_t' \leftarrow \tilde{\Delta}_t + \mathcal{N}(0, I\sigma^2 C^2 |S|)$ ▷ based on user-level sensitivity
15: **return** Δ_t'

THEOREM 1. *For any $0 < \delta < 1$ and $\alpha > 1$, given noise multiplier σ, ULDP-NAIVE satisfies $(\epsilon = \frac{T\alpha}{2\sigma^2} + \log((\alpha-1)/\alpha) - (\log\delta + \log\alpha)/(\alpha-1), \delta)$-ULDP after T rounds.* (The actual ϵ is numerically calculated by selecting the optimal α so that ϵ is minimized.)

ULDP-GROUP-k. We introduce a second baseline, ULDP-GROUP-k, utilizing group DP (described in Algorithm 2), which limits each user's records to a given k while satisfying (k, ϵ, δ)-GDP. As Proposition 1 implies, this ensures (ϵ, δ)-ULDP. The algorithm achieves GDP by firstly performing DP-SGD [1] (Line 9) and converting from record-level DP within each silo. The core principle of the algorithm is similar to that of [30]. Before executing DP-SGD, it is essential to constrain the number of records per user to k (Line 8). This is accomplished by employing flags, denoted as $\mathbf{B}$, which indicate the records to be used for training (i.e., $b^s u, i = 1$), with a total of k records for each user across all silos (i.e., $\forall u, \sum_{s,i} b_{u,i}^s \leq k$). These flags must be consistent across all rounds. We disregard the privacy concerns in generating these flags as this is a baseline method.

Algorithm 2 ULDP-GROUP-k

Input: η_l, η_g: local and global learning rates, σ: noise parameter, D_s: training dataset of silo s, C: clipping bound, T: #rounds, Q: #local epochs, k: group size, γ: sampling rate, $\mathbf{B}$: flags for limit contribution s.t. for each matrix $\mathbf{b}^s \in \mathbf{B}$ if $b_{u,i}^s = 1$ the user u's i-th record in silo s is used, otherwise the record is excluded

1: **procedure** SERVER
2: Initialize model x_0
3: **for** each round $t = 0, 1, \ldots, T-1$ **do**
4: **for** each silo $s \in S$ **do**
5: $\Delta_t^s \leftarrow \text{CLIENT}(x_t, C, \sigma, \eta_l, \gamma, \mathbf{b}^s)$
6: $\theta_{t+1} \leftarrow \theta_t + \eta_g \frac{1}{|S|} \sum_{s\in S} \Delta_t^s$
7: **procedure** CLIENT$(x_t, C, \sigma, \eta_l, \gamma, \mathbf{b}^s)$
8: $D_s' \leftarrow$ filter D_s by $\mathbf{b}^s$
9: $x_t^Q \leftarrow$DP-SGD$(\theta_t, D_s', C, \sigma, \eta_l, \gamma, Q)$ ▷ Algorithm 1 in [1]
10: $\Delta_{t+1} \leftarrow x_t^Q - x_t$
11: **return** Δ_{t+1}

THEOREM 2. *If flags $\mathbf{B}$ is given privately, for any $0 < \delta < 1$, any integer k to the power of 2 and $\alpha > 2^{k+1}$, ULDP-GROUP-k satisfies $(3^k\rho + \log((\frac{\alpha}{2^k} - 1)/\frac{\alpha}{2^k}) - (\log\delta + \log\frac{\alpha}{2^k})/(\frac{\alpha}{2^k} - 1), \delta)$-ULDP where $\rho = \max_{s\in S} \rho_s$ s.t. for each silo $s \in S$, DP-SGD of local subroutine satisfies (α, ρ_s)-RDP.*

While ULDP-GROUP shares algorithmic similarities with existing record-level DP cross-silo FL frameworks [30], it presents weaknesses from several perspectives: (1) Significant degradation of privacy bounds due to the group-privacy conversion (DP to GDP). (2) The challenge of determining an appropriate group size k [4], which requires substantial insights into data distribution across silos and might breach the trust model. The determination of the flags $\mathbf{B}$ can also be problematic. (3) The use of group-privacy to guarantee ULDP necessitates removing records from the training dataset, potentially introducing bias and causing utility degradation [4, 14]. Our next proposed method aims to address these challenges.

3.4 Advanced methods: ULDP-AVG/SGD

To directly satisfy ULDP without using group-privacy, we design ULDP-AVG and ULDP-SGD (described in Algorithm 3) . These are the same as the relationship between (DP-)FedAVG and (DP-)FedSGD [36]. In most cases, FedAVG is better in communication-cost and privacy-utility trade-offs. FedSGD might be preferable only when we have fast networks. In the following analysis, we focus

Algorithm 3 ULDP-AVG / ULDP-SGD

Input: η_l, η_g: local and global learning rates, σ: noise parameter, C: clipping bound, T: total round, Q: #local epochs, $\mathbf{W} = (\mathbf{w}_1, ..., \mathbf{w}_{|S|})$: matrix with weight for user u and silo s, and $\forall u \in U$, $w_{s,u} \in \mathbf{w}_s$ and $\sum_{s\in S} w_{s,u} = 1$

1: **procedure** SERVER
2: Initialize model x_0
3: **for** each round $t = 0, 1, \ldots, T-1$ **do**
4: **for** each silo $s \in S$ **do**
5: $\Delta_t^s \leftarrow \text{CLIENT}(x_t, \mathbf{w}_s, C, \sigma, \eta_l)$
6: $x_{t+1} \leftarrow x_t + \eta_g \frac{1}{|U||S|} \sum_{s\in S} \Delta_t^s$
7: /* **Client algorithm for ULDP-AVG** */
8: **procedure** CLIENT$(x_t, \mathbf{w}_s, C, \sigma, \eta_l)$ ▷ For ULDP-AVG
9: **for** user $u \in U$ **do** ▷ per-user training with $\mathcal{D}_{s,u}$
10: $x_t^{s,u} \leftarrow x_t$
11: **for** epoch $q = 0, 1, \ldots, Q-1$ **do**
12: Compute stochastic gradients $g_{t,q}^{s,u}$
13: ▷ $\mathbb{E}[g_{t,q}^{s,u}] = \nabla f_{s,u}(x_t^{s,u})$
14: $x_t^{s,u} \leftarrow x_t^{s,u} - \eta_l g_{t,q}^{s,u}$
15: $\Delta_t^{s,u} \leftarrow x_t^{s,u} - x_t$
16: $\tilde{\Delta}_t^{s,u} \leftarrow w_{s,u} \cdot \Delta_t^{s,u} \cdot \min\left(1, \frac{C}{\|\Delta_t^{s,u}\|_2}\right)$
17: $\Delta_t^s \leftarrow \sum_{u\in U} \tilde{\Delta}_t^{s,u} + \mathcal{N}(0, I\sigma^2C^2/|S|)$
18: **return** Δ_t^s
19: /* **Client algorithm for ULDP-SGD** */
20: **procedure** CLIENT$(x_t, \mathbf{w}_s, C, \sigma)$ ▷ For ULDP-SGD
21: **for** user $u \in U$ **do**
22: Compute stochastic gradients $g_t^{s,u}$
23: $\tilde{g}_t^{s,u} \leftarrow w_{s,u} \cdot g_t^{s,u} \cdot \min\left(1, \frac{C}{\|g_t^{s,u}\|_2}\right)$
24: $g_t^s \leftarrow \sum_{u\in U} \tilde{g}_t^{s,u} + \mathcal{N}(0, I\sigma^2C^2/|S|)$
25: **return** g_t^s

on ULDP-AVG since it essentially generalizes ULDP-SGD which has only a single SGD step and shares the gradients.

Intuitively, ULDP-AVG limits each user's contribution to the global model by training the model for each user in each silo and performing per-user per-silo clipping across all silos with globally prepared clipping weights. In each round, ULDP-AVG computes model delta using a per-user dataset in each silo to achieve ULDP: selecting a user (Line 9), training local model with Q epochs using only the selected user's data (Lines 11-14), calculating model delta (Line 15) and clipping the delta (Line 16). These clipped deltas $\Delta_t^{s,u}$ are then weighted by $w_{s,u}$ (Line 16) and summed for all users (Line 16). As long as the weights $w_{s,u}$ satisfy constraints $\forall u \in U$, $w_{s,u} > 0$ and $\sum_{s\in S} w_{s,u} = 1$, each user's contribution, or *sensitivity*, to the delta aggregation $\sum_{s\in S} \Delta_t^s$ is limited to C at most. This allows ULDP-AVG to provide user-level privacy. We will discuss better ways to determine $\mathbf{W}$ later, but a simple way is to set $w_{s,u} = 1/|S|$. Compared to DP-FedAVG, ULDP-AVG increases computational cost due to per-user local training iteration but keeps communication costs the same, which is likely acceptable in the cross-silo FL setting.

Algorithm 4 ULDP-AVG with user-level sub-sampling

Input: η_l, η_g, σ, C, T, Q, $\mathbf{W}$, q: user-level sub-sampling probability

1: **procedure** SERVER
2: Initialize model x_0
3: **for** each round $t = 0, 1, \ldots, T-1$ **do**
4: $U_t \leftarrow$ Poisson sampling from U with probability q
5: **for** each silo $u \in U_t$ **do**
6: **for** each silo $s \in S$ **do**
7: $w_{s,u} \leftarrow 0$ ▷ set 0 if user is not sampled.
8: **for** each silo $s \in S$ **do**
9: $\Delta_t^s \leftarrow \text{CLIENT}(x_t, \mathbf{w}_s, C, \sigma, \eta_l)$ ▷ same as ULDP-AVG
10: $x_{t+1} \leftarrow x_t + \eta_g \frac{1}{q|U||S|} \sum_{s\in S} \Delta_t^s$

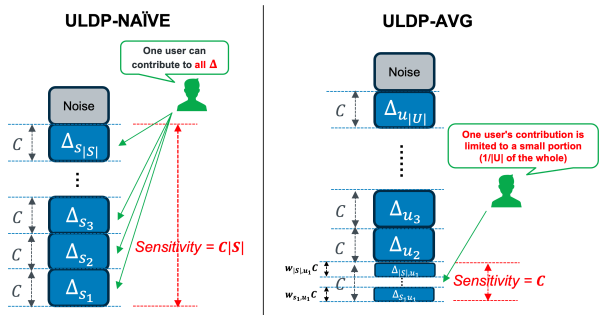


Figure 3: An intuitive illustration of the difference between ULDP-NAIVE and ULDP-AVG. In ULDP-NAIVE, every user can contribute to whole model deltas. In ULDP-AVG, one user's contribution is limited to a small portion, i.e., $1/|U|$ of the whole model delta, which reduces user-level sensitivity.

THEOREM 3. *For any $0 < \delta < 1$ and $\alpha > 1$, given noise multiplier σ, ULDP-AVG satisfies $(\epsilon = \frac{T\alpha}{2\sigma^2} + \log((\alpha-1)/\alpha) - (\log\delta + \log\alpha)/(\alpha-1), \delta)$-ULDP after T rounds.*

REMARK 1. For further privacy amplification, we introduce user-level sub-sampling, which can make RDP smaller according to sub-sampled amplification theorem (Lemma 4) [48]. User-level sub-sampling must be done globally across silos. This sub-sampling can be implemented in the central server by controlling the weight $\mathbf{W}$ for each round, i.e., all users not sub-sampled are set to 0 as shown in Algorithm 4. This may violate privacy against the server but does not affect the DP when the final model is provided externally as discussed in C.3 of [2]. Our following experimental results demonstrate the effectiveness of user-level sub-sampling.

Comparison to baselines. Compared to ULDP-GROUP, ULDP-AVG satisfies ULDP without group-privacy, thus avoiding the large privacy bound caused by group-privacy conversion, the need to choose a group size k, and removing records. ULDP-AVG can be used for an arbitrary number of records per user. Also, we illustrate the intuitive difference between ULDP-NAIVE and ULDP-AVG in Figure 3. Fundamentally, per-user clipping can be viewed as cross-user FL (instead of cross-silo FL), ensuring that each user contributes only to their user-specific portion of the aggregated model updates (i.e., $\sum_{s\in S} \tilde{\Delta}_t^{s,u}$) instead of the entire aggregated update (i.e., $\sum_{s\in S} \Delta_t^s$), thereby reducing sensitivity (as illustrated in

Figure 3). The user contributes only $1/|U|$ of the entire aggregated model update, which is especially effective when $|U|$ is large, as in cross-silo FL (i.e., $|S| \ll |U|$). Moreover, computing the model delta at the user level leads to lower Gaussian noise variances due to large $|U|$, while it also introduces new biases. The overhead due to such biases can also be seen in the convergence analysis, motivating a better weighting strategy to reduce this overhead.

Convergence analysis (sketch)[2]**.** Here, we present a high-level and intuitive summary of the convergence characteristics of ULDP-AVG, compared to existing methods. ULDP-AVG partially recaptures the standard convergence bounds of FedAVG by considering user-silo pairs as participants and by setting the global and local learning rates under specific conditions. However, compared to FedAVG, an additional noise term for DP and a bias due to user-silo granularity are introduced, which hinder convergence. The former, a noise term, is also present in DP-FedAVG and can diminish as the number of users increases. The latter bias term can be minimized through the strategic weighting of weights in ULDP-AVG. Thus, a more creative use of weights, as will be explained in the next section, could result in improved convergence.

4 ENHANCED WEIGHTING AND PRIVATE WEIGHTING PROTOCOL

4.1 The weighting protocol

The bias we observed in the convergence analysis on ULDP-AVG (described in detail in Remark in Appendix of the full paper [27]) is due to the fact that we employed uniform clipping weights in the ULDP-AVG algorithm (Algorithm 3 Line 16), i.e., for any $s \in S$ and $u \in U$, we set $w_{s,u} = 1/|S|$, as a simple solution without privacy violation. Now we propose an enhanced weighting strategy that aims to reduce the bias. We set a weight $w_{s,u}^{opt}$ for $C_{s,u}$ according to the number of records for user u in silo s, following the heuristic that a gradient computed from a large number of records yields a better estimation that is closely aligned with the average. This results in smaller bias. That is, let $n_{s,u}$ be the number of records for user u in silo s, we set the weight as follows:

$$w_{s,u}^{opt} := \frac{n_{s,u}}{\sum_{s \in |S|} n_{s,u}}. \tag{3}$$

We empirically demonstrate the effectiveness of this strategy later.

Private weighting protocol. Given the above weighting strategy that relies on number of user records per user per silo, the crucial question arises: *how can this be implemented without violating privacy?* A central server could aggregate histograms encompassing the user population (number of records per user) within each silo's dataset. Subsequently, the server could compute the appropriate weights for each silo and distribute these weights back to the respective silos. However, it raises significant privacy concerns, since the histograms are directly shared with the server. Moreover, when the server broadcasts the weights back to the silos, it enables an estimation of the entire histogram of users across all the silos, posing a similar privacy risk against other silos. In essence, the privacy protection is necessary in both directions. This is challenging. Additive homomorphic encryption, such as Paillier cryptosystem, is often used for this situation [3], but it is impossible to securely compute inverse values (for the weight in Eq. (3)). Note that unlimited records per user make DP impractical for protection.

[2]Please refer to the Appendix of the full paper [27] for a rigorous result where we theoretically analyze the convergence of ULDP-AVG to compare with existing methods.

Algorithm 5 Encode and Decode

```
1:  procedure Encode(x, P, n)                          ▷ e.g., P = 10^-10
2:      /* to turn floating point into fixed point */
3:      x ← x/P                                        ▷ compute as floating point
4:      x ← x as integer
5:      x ← x (mod n)                                  ▷ to map integer ℤ into finite field 𝔽_n
6:      return x
7:  procedure Decode(x, P, C_LCM, n)
8:      /* to map finite field 𝔽_n number into integer ℤ */
9:      if x > n//2 then                               ▷ // means integer division
10:         x ← x − n
11:     else
12:         x ← x
13:     /* compute as floating point */
14:     x ← x/C_LCM                                    ▷ to remove C_LCM factor
15:     x ← xP                                         ▷ to recover original magnitude
16:     return x
```

To address this privacy issue, we design a novel private weighting protocol to securely aggregate the user histograms and also to securely perform local training and model aggregation. The protocol leverages well-established cryptographic techniques, including secure aggregation [7, 45], the Paillier cryptosystem [3], and multiplicative blinding [9]. Intuitively, the protocol employs multiplicative blinding to hide user histograms against the server while allowing the server to compute inverses of blinded histograms to compute the weights (Eq. (3)). Subsequently, the server employs the Paillier encryption to conceal the inverses of blinded histograms because the silo knows the blinded masks. This enables the server and silos to compute private weighted sum aggregation with its additive homomorphic property.

The details of the private weighting protocol are explained in Protocol 1 (with encode and decode schemes in Algorihtm 5). The protocol consists of a setup phase, which is executed only once during the entire training process, and a weighting phase, which is executed in each round of training. In the setup phase, as depicted in (a-c) of Protocol 1, the server generates a key-pair for Paillier encryption, while the silos establish shared random seeds through a Diffie–Hellman (DH) key exchange via the server. Subsequently, in steps of (d-f), the blinded inverses of the user histogram are computed. In the weighting phase, (a) the server prepares the encrypted weights, (b) the silos compute user-level weighted model deltas in the encrypted world, and (c) the server recovers the aggregated value. It is important to note that in the Paillier cryptosystem, the plaintext x exists within the additive group modulo n, while encrypted data (denoted as $Enc_{\mathrm{p}}(x)$) belongs to the multiplicative group modulo n^2 with an order of n. The system allows for operations such as addition of ciphertexts and scalar multiplication and addition on ciphertexts.

Private user-level sub-sampling. Note that our assumption so far is that the results of user-level sub-sampling (i.e., whether a user is sampled) are open to the aggregation server. This is also the case in

Protocol 1 Private Weighting Protocol

Inputs: Silo $s \in S$ that holds an dataset with $n_{s,u}$ records for each user $u \in U$. $\mathcal{A}$ is central aggregation server. $N_{\max}$ is upper bound on the number of records per user, e.g., 2000. P is precision parameter, e.g, 10^{-10}. λ is security parameter, e.g., 3072-bit security.

(1) **Setup.**
   (a) $\mathcal{A}$ generates Paillier keypairs (PK, SK) with the given security parameter λ and sends the public key PK to all silos. All silos s generate DH keypairs (pk_s, sk_s) with the same parameter λ and transmit their respective public key pk_s to $\mathcal{A}$. Both $\mathcal{A}$ and all silos compute C_{LCM}, which is the least common multiple of all integers up to $N_{\max}$. The modulus n included in PK is used for the finite field $\mathbb{F}_n$ by $\mathcal{A}$ and all silos.
   (b) After receiving all pk_s, $\mathcal{A}$ broadcasts all DH public keys pk_s to all s. All s compute shared keys $sk_{s,s'}$ from sk_s and received public keys $pk_{s'}$ for all $s' \in S$.
   (c) Silo 0 ($\in S$) generates a random seed R and encrypts R using $sk_{0,s'}$ to obtain $Enc(R)$ and sends $Enc(R)$ to s' via $\mathcal{A}$ for all s'. All $s \in S \setminus 0$ receive and decrypt $Enc(R)$ with $sk_{s,0}$ and get R as a shared random seed.
   (d) All s generate multiplicative blind masks $r_u \in \mathbb{F}_n$ with the same R and compute blinded histogram as $B(n_{s,u}) \equiv r_u n_{s,u} \pmod n$ for all $u \in U$.
   (e) All s generate pair-wise additive masks $r^u_{s,s'} \in \mathbb{F}_n$ employing $sk_{s,s'}$ for all s' and u, with $r^u_{s,s'} = r^u_{s',s}$. Subsequently, they calculate the doubly blinded histogram as $B'(n_{s,u}) \equiv B(n_{s,u}) + \sum_{s<s'} r^u_{s,s'} - \sum_{s>s'} r^u_{s,s'} \pmod n$. All s send $B'(n_{s,u})$ to $\mathcal{A}$. $\mathcal{A}$ aggregates these contributions to compute $B(N_u) \equiv \sum_{s\in S} B'(n_{s,u}) \pmod n$ for each u, denoting $N_u = \sum_{s\in S} n_{s,u}$.
   (f) $\mathcal{A}$ computes the inverse of $B(N_u)$ as $B_{\text{inv}}(N_u) = B(N_u)^{-1}$ for each u. This is the multiplicative inverse on $\mathbb{F}_n$, which is efficiently computed by the Extended Euclidean algorithm.

(2) **Weighting for each training round t.**
   (a) $\mathcal{A}$ encrypts $B_{\text{inv}}(N_u)$ using Paillier's public key PK, resulting in $Enc_{\text{p}}(B_{\text{inv}}(N_u))$ for all u. If user-level sub-sampling is required, the server performs Poisson sampling with a given probability q for each user before the encryption. For non-selected users, $B_{\text{inv}}(N_u)$ is set to 0. If we require user-level sub-sampling, we perform Poisson sampling with given probability q on the server for each user before the Paillier's encryption and set $B_{\text{inv}}(N_u) = 0$ for all users not selected. Subsequently, $\mathcal{A}$ broadcasts all $Enc_{\text{p}}(B_{\text{inv}}(N_u))$ to all silos.
   (b) In each s, following the approach of ULDP-AVG, the clipped model delta $\tilde{\Delta}^{s,u}_t$ is computed for each user u. The weighted clipped model delta is then calculated as
   $$Enc_{\text{p}}(\tilde{\Delta}^{s,u}_t) = \text{Encode}(\tilde{\Delta}^{s,u}_t, P, n) n_{s,u} r_u C_{\text{LCM}} Enc_{\text{p}}(B_{\text{inv}}(N_u)).$$
   Let the Gaussian noise be z^s_t, we then compute $z'_s = \text{Encode}(z^s_t, P, n) C_{\text{LCM}}$. Note that we need to approximate real number $\tilde{\Delta}^{s,u}_t$ and z^s_t on a finite field using Encode (described in Algorithm 5). Lastly, we compute the summation $Enc_{\text{p}}(\Delta^s_t) = \sum_{u\in U} Enc_{\text{p}}(\tilde{\Delta}^{s,u}_t) + z'_s$.
   (c) In each s, random pair-wise additive masks are generated, and secure aggregation is performed on $Enc_{\text{p}}(\Delta^s_t)$ mirroring the steps in 1.(f). Then, $\mathcal{A}$ gets $\sum_{s\in S} Enc_{\text{p}}(\Delta^s_t)$. $\mathcal{A}$ decrypts it with Paillier's secret key SK and decodes it by $\text{Decode}(\sum_{s\in S} \Delta^s_t, P, C_{\text{LCM}}, n)$ and recovers the aggregated value.
   (d) Steps 2.(a) through 2.(c) are repeated for each training round.

Protocol 1. However, it could be hidden by combining the two-party verifiable sampling scheme with 1-out-of-P Oblivious Transfer (OT) as described in [25]. As an overview, for each user u, the server creates $P-1$ dummy data $Enc_{\text{p}}(0)$ for $Enc_{\text{p}}(B_{\text{inv}}(N_u))$ described in the step 2.(a) of Protocol 1. When the client performs OT on this data, the selection probability of $Enc_{\text{p}}(B_{\text{inv}}(N_u))$ is $\frac{1}{P}$ and that of $Enc_{\text{p}}(0)$ is $\frac{P-1}{P}$. The selection of $Enc_{\text{p}}(B_{\text{inv}}(N_u))$ means that the user is not sampled by the user-level sub-sampling. In this way, the server does not know which data was retrieved by the client from the OT, and the client cannot know the sampling result due to the Paillier encryption. However, the expressed probability is likely to be less strict because it can only represent discrete probability distributions. This process requires extra computational costs for both the server and the silo, proportional to the number of users, and should not be included if it is not necessary.

4.2 Theoretical analysis

We provide a theoretical analysis of this private weighting protocol (Protocol 1) in terms of correctness and privacy.

Correctness. The protocol must compute the correct result that is the same as non-secure method. To this end, we consider the correctness of the aggregated data obtained in each round.

Theorem 4 (Correctness of Protocol 1). *Let $\sum_{s\in S} \Delta^s_t$ with non-secure method be Δ and the one with the Protocol 1 be Δ_{sec}, our goal is formally stated as $\Pr[|\Delta - \Delta_{sec}|_\infty > P] < negl$, where P is a precision parameter and $negl$ signifies a negligible value.*

Privacy. In the protocol, both the central server and the silos do not get more information than what is available in the original ULDP-AVG while we perform the enhanced weighting strategy.

Theorem 5 (Privacy of Protocol 1). *None of the parties learns $n_{s,u}$ other than their own users from the protocol.*

5 EXPERIMENTS

In this section, we report the results of the experimental evaluation of our proposed methods. We design experiments to answer the following questions:

- How much does our proposed method improve the privacy-utility trade-offs from baselines in terms of ULDP?
- How effective are enhanced weighting strategies and user-level sub-sampling in enhancing ULDP-AVG?
- How efficient is the proposed private weight protocol? Can it work for real-world data?

All of our experimental source code and settings are available[3].

5.1 Settings

We evaluate the privacy-utility trade-offs of the proposed methods (ULDP-AVG/ULDP-AVG-w/SGD), along with the baselines (ULDP-NAIVE/GROUP-k) and a non-private baseline (FedAVG with two-sided learning rates [50], denoted by DEFAULT). In ULDP-AVG/SGD, we set the weights as $w_{s,u} = 1/|S|$ for all s and u, the one using $w^{opt}_{s,u}$ is referred to as ULDP-AVG-w. Regarding ULDP-GROUP-k, flags **B** are generated for existing records to minimize waste on

[3] https://github.com/Emory-AIMS/uldp-fl

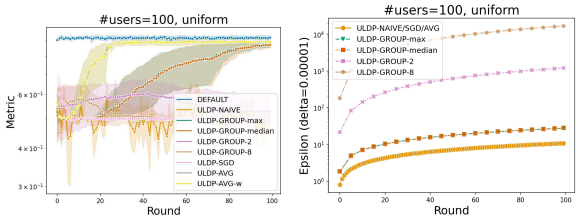
filtered out records, despite the potential privacy concerns. Various values, including the maximum number of user records (ULDP-GROUP-max), the median (ULDP-GROUP-median), 2, and 8, are tested as group size k and we report GDP using group-privacy conversion of RDP. In particular, ULDP-GROUP-max would represent an upper bound on the utility achieved by record-level DP in each silo (such as [33] and [30]), since there are no deleted records. In cases where k is not a power of 2, the computed ϵ is reported for the largest power of 2 below k, showcasing the lower bound of GDP to underscore that ϵ is large. The hyperparameters, including global and local learning rates η_g, η_l, clipping bound C, and local epoch Q, are set individually for each method. Execution times are measured on macOS Monterey v12.1, Apple M1 Max Chip with 64GB memory with Python 3.9 and 3072-bit security. Most of the results are averaged over 5 runs and the colored area in the graph represents the standard deviation.

5.1.1 Datasets. Datasets used in the evaluation comprise real-world open datasets, including Credicard [21], well-known image dataset MNIST [11], and two benchmark medical datasets for cross-silo FL [42], HeartDisease and TcgaBrca. Creditcard is a tabular dataset for credit card fraud detection from Kaggle. We undersample the dataset and use about 25K training data and a neural network with about 4K parameters. For MNIST, we use a convolutional neural network (CNN) with about 20K parameters, 60K training data and 10K evaluation data, and assigned silos and users to all of the training data. For HeartDisease and TcgaBrca, we use the same setting such as number of silos (4 and 6), data assignments to the silos, models, etc. as shown in [42]. These two datasets are small and the model has less than 100 parameters.

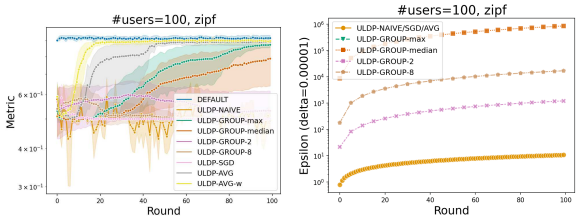
For all datasets, we need to link all records to each user and silo. We allocate the records to users and silos as follows.

Record allocation for MNIST and Creditcard. We designed two different record distribution patterns, *uniform* and *zipf*, to model how user records are scattered across silos in the MNIST and Creditcard datasets. Both distributions take the number of users $|U|$ and the number of silos $|S|$. It associates each record with a user and a silo. (1) In uniform, every record is assigned to a user with equal probability, and likewise, each record is assigned to a silo with equal probability. (2) zipf combines two types of Zipf distributions. First, the distribution of the number of records per user follows a Zipf distribution. Then, for each user, the numbers of records are assigned to different silos based on another Zipf distribution. Each of the two Zipf distributions takes a parameter α that determines the concentration of the numbers. In the experiments, we used $\alpha = 0.5$ for the first distribution and $\alpha = 2.0$ for the second distribution. This choice is rooted in the observation that the concentration of user records is not as high as the concentration in the silos selected by each user. For Creditcard and MNIST, the number of silos $|S|$ is fixed at 5. We used 100, 1000 for Creditcard as $|U|$ and 100, 10000 for MNIST. For MNIST, we can require each user to have only 2 labels at most for non-i.i.d.

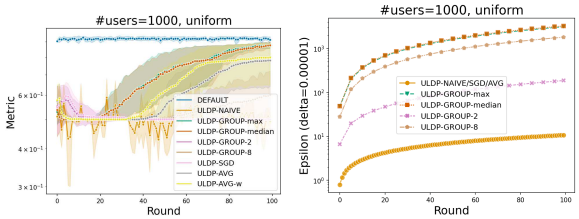
Record allocation for HeartDisease and TcgaBrca. For the HeartDisease and TcgaBrca datasets, we adopted the same two distributions *uniform* and *zipf* as mentioned above. In the benchmark datasets HeartDisease and TcgaBrca, all records are already allocated to silos and the number of records of each silo is fixed.



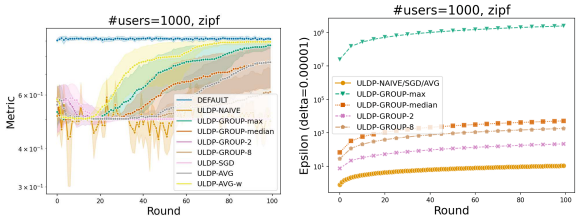
(a) $n \approx 246$ ($|U| = 100$), uniform.



(b) $n \approx 246$ ($|U| = 100$), zipf.



(c) $n \approx 25$ ($|U| = 1000$), uniform.



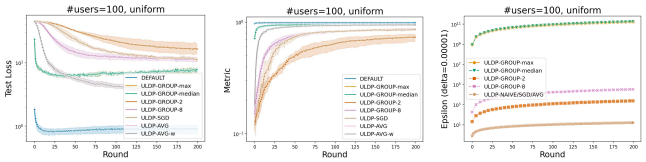
(d) $n \approx 25$ ($|U| = 1000$), zipf.

Figure 4: Privacy-utility trade-offs on Creditcard dataset: Test Accuracy (Left), Privacy (Right).

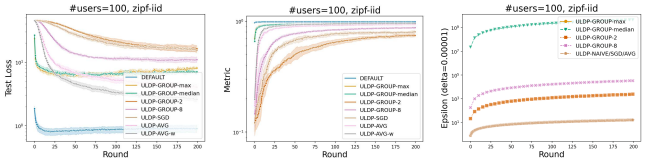
Therefore, the design of the user-record allocation is slightly different. (1) In uniform, all records belong to one of the users with equal probability without allocation to silos. (2) In zipf, the number of records for a user is first generated according to a Zipf distribution, and 80% of the records are assigned to one silo, and the rest to the other silos with equal probability. The priority of the silo is chosen randomly for each user. We used $\alpha = 0.5$ for the parameter of the Zipf. In TcgaBrca, Cox-Loss is used for loss function [42], which needs more than two records for calculating valid loss and we set more than two records for each silo and user for per-user clipping of ULDP-AVG.

5.2 Results

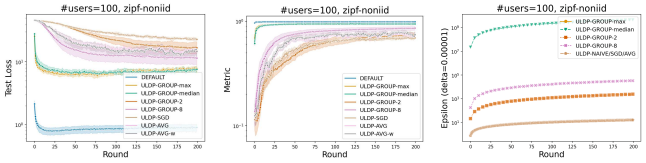
Privacy-utility trade-offs under ULDP. Figures 4 show the utility and privacy evaluation results on Creditcard. The average number of records per user (denoted as n) in entire silos and the distribution



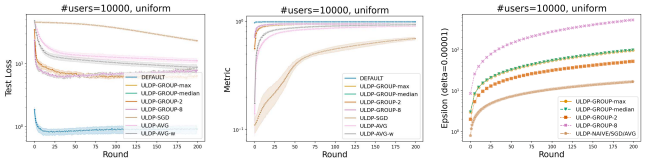
(a) $n \approx 600$ ($|U| = 100$), uniform.



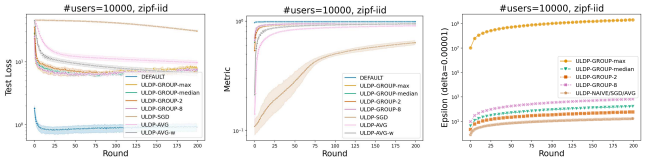
(b) $n \approx 600$ ($|U| = 100$), zipf, iid.



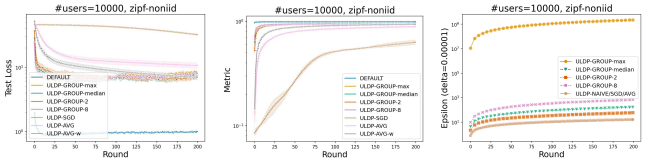
(c) $n \approx 600$ ($|U| = 100$), zipf, non-iid.



(d) $n \approx 6$ ($|U| = 10000$), uniform, iid.



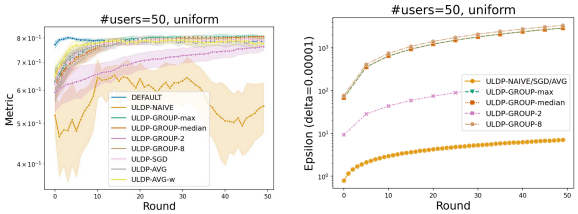
(e) $n \approx 6$ ($|U| = 10000$), zipf, iid.



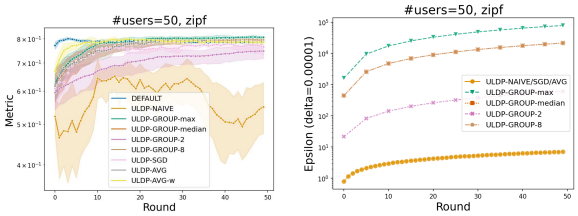
(f) $n \approx 6$ ($|U| = 10000$), zipf, non-iid.

Figure 5: Privacy-utility trade-offs on MNIST dataset: Test Loss (Left), Accuracy (Middle), Privacy (Right).

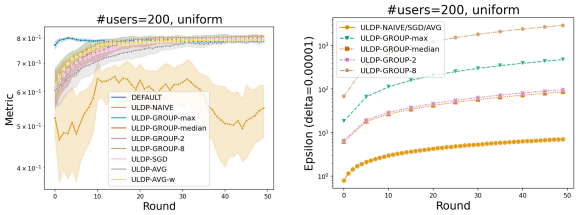
changes for each figure. All experiments used a fixed noise parameter $\sigma = 5.0$ and $\delta = 10^{-5}$, utility metrics (Accuracy for Creditcard) are displayed on the left side and accumulated privacy consumption ϵ for ULDP are depicted on the right side. Note that the privacy bounds for ULDP-GROUP-k are derived from the local DP-SGD and depend on not only the group size k but also the size of the local training dataset.



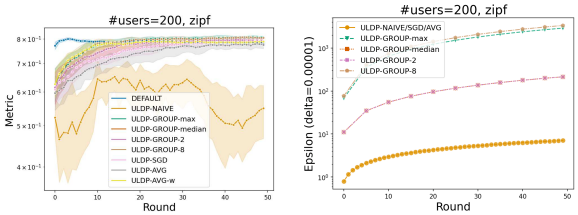
(a) $n \approx 10$ ($|U| = 50$), uniform.



(b) $n \approx 10$ ($|U| = 50$), zipf.



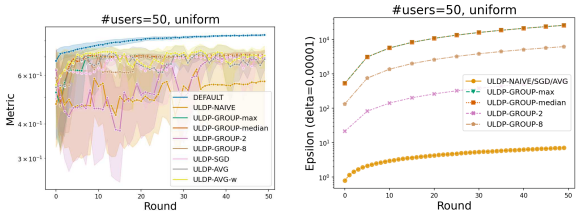
(c) $n \approx 2.5$ ($|U| = 200$), uniform.



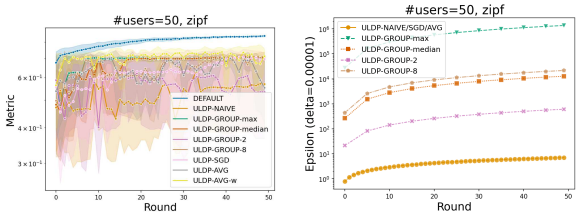
(d) $n \approx 2.5$ ($|U| = 200$), zipf.

Figure 6: HeartDisease.

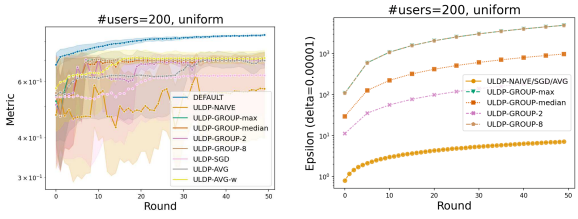
Overall, the proposed method ULDP-AVG/SGD achieves competitive utility with fast convergence and high accuracy, while achieving considerably small privacy bounds, which means the significantly better privacy-utility trade-offs compared to baselines. We observe that the baseline method, ULDP-NAIVE, has low accuracy and that ULDP-GROUP-k requires much larger privacy budgets, which is consistent with the analysis on the conversion of group privacy described earlier. The convergence speed of ULDP-AVG is faster than that of ULDP-SGD, which is the same as that of DP-FedAVG/SGD. Nevertheless, there is still a gap between ULDP-AVG and the non-private method (DEFAULT) in terms of convergence speed and ultimately achievable accuracy, as a price for privacy. Also, as shown in Figure 4c, for small n (i.e., a large number of users), ULDP-GROUP-max/median show higher accuracy than ULDP-AVG. This is likely due to the overhead from finer datasets at user-level, which increases the bias compared to DP-FedAVG, as seen in the theoretical convergence analysis for ULDP-AVG.



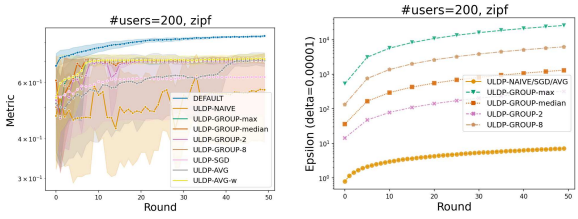
(a) $n \approx 17$ ($|U| = 50$) uniform.



(b) $n \approx 17$ ($|U| = 50$), zipf.



(c) $n \approx 4$ ($|U| = 200$), uniform.



(d) $n \approx 4$ ($|U| = 200$), zipf.

Figure 7: TcgaBrca.

Figure 5, 6, and 7 show privacy-utility trade-offs on MNIST, HeartDisease, and TcgaBrca respectively. All experiments used a fixed noise parameter (noise multiplier) $\sigma = 5.0$ and $\delta = 10^{-5}$, utility metrics (Accuracy for HeartDisease and MNIST, C-index for TcgaBrca) are plotted on the left side, and accumulated privacy consumption ϵ for ULDP are plotted on the right side. For clarity, the test loss is shown on the left-hand side for MNIST. The average number of records per user (denoted as n) in entire silos and the distribution (uniform/zipf) changes for each figure.

In all datasets, consistently, ULDP-AVG is competitive in terms of utility, ULDP-AVG-w shows much faster convergence, and ULDP-SGD shows slower convergence. ULDP-NAIVE achieves a low privacy bound; however, its utility is much lower than other methods. ULDP-GROUP-k show reasonably high utility, especially in settings where n is small. This is because the records to be removed due to the number of records per user being over group size k is small. However, the privacy bound of ULDP-GROUP-k is very large. Note

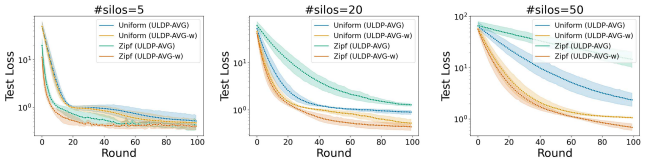


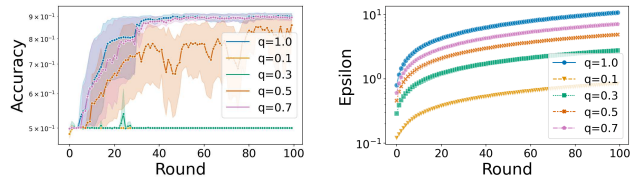
Figure 8: Test loss of Creditcard: Weighting method is effective, especially in skewed distribution in many silos.

that the privacy bounds for ULDP-GROUP-k are derived from the local DP-SGD and depend on not only the group size k but also the size of the local training dataset. The exceptions are cases where the local data set size is large and the number of records per user is very small as in Figures 5d, 5e, and 5f. In these cases, ULDP-GROUP-2 achieves a reasonably small privacy bound. In other words, if the number of user records is fixed at one or two in the scenario, and the number of training records is large (it is advantageous for ULDP-GROUP because the record-level sub-sampling rate in DP-SGD becomes small), it could be better to use ULDP-GROUP.

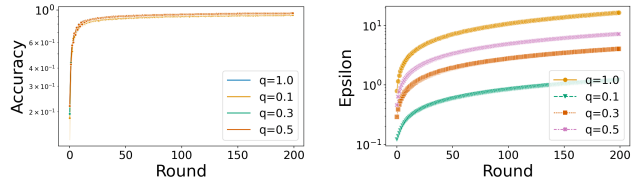
Effect of non-IID data. The results for MNIST non-i.i.d and $|U| = 100$ case highlight a weak point of ULDP-AVG. Note that non-i.i.d here is at user-level and DEFAULT and ULDP-GROUP are less affected by non-i.i.d because they train per silo rather than per user. As Figure 5c shows, the convergence of ULDP-AVG is worse compared to other results. It suggests that ULDP-AVG may emphasize the bad effects of user-level non-i.i.d. distribution, which was not an issue with normal cross-silo FL because the gradient is not computed at the user level as in ULDP-AVG. This is less problematic when the number of users is large as shown in Figure 5f. This is due to the relatively smaller effect of individual user overfitting caused by non-i.i.d. distribution as the number of users increases. A potential way to address it could be to adapt existing non-iid cross-device FL methods [17] to the user-level in the silo (e.g., turning client-level clustering into user-level clustering). However, it should be non-trivial, since it should not violate ULDP.

Effectiveness of enhanced weighting strategy. To highlight the effectiveness of the enhanced weighting strategy, Figure 8 shows the test losses of the Creditcard dataset on different record distributions with ULDP-AVG and ULDP-AVG-w. We present the results with various numbers of silos: 5, 20, and 50. The need for the better weighting strategy is emphasized by the distribution of the records and the number of silos $|S|$. When there are large skews in the user records, as in the Zipf distribution, giving equal weights (i.e., ULDP-AVG) results in inefficiency and opens up a large gap from ULDP-AVG-w. This trend becomes even more significant as $|S|$ increases because all weights become smaller in ULDP-AVG.

Effect of user-level sub-sampling. We evaluate the effect of user-level sub-sampling. Figure 9a illustrates how user-level sub-sampling affects the privacy-utility trade-offs on the Creditcard dataset with 1000 users. We report the test accuracy and ULDP privacy bounds for various sampling rates $q = 0.1, 0.3, 0.5, 0.7, 1.0$. Basically, a tighter privacy bound is obtained at the expense of utility. As the results show, the degradation of utility due to sub-sampling could be acceptable to some extent (e.g., $q = 0.7$) and there could be an optimal point for each setting. Figure 9b illustrates how



(a) Creditcard.



(b) MNIST.

Figure 9: User-level sub-sampling achieves a more competitive privacy-utility trade-off.

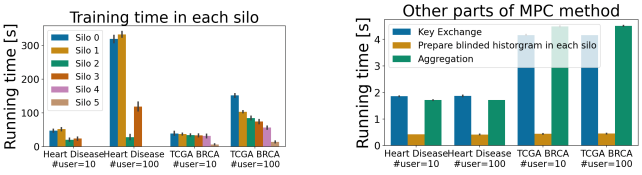


Figure 10: With a small model, the private weighting protocol has a practical execution time.

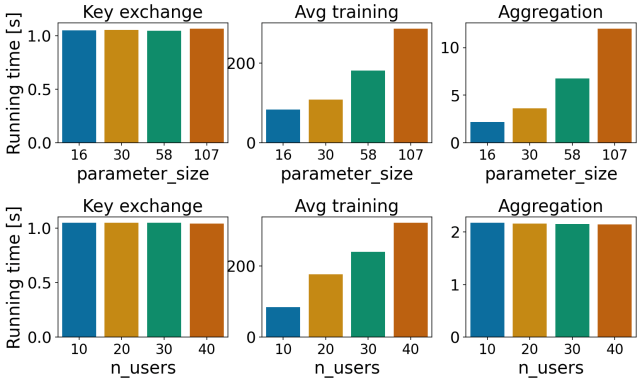


Figure 11: The dominant execution time grows linearly with parameter size (Top) and/or the number of users (Bottom).

user-level sub-sampling affects the privacy-utility trade-offs on MNIST with 10000 users, with sampling rates $q = 0.1, 0.3, 0.5, 1.0$. The results show that while privacy is greatly improved, there is less degradation in utility. This is due to the fact that there are a sufficient number of users, i.e., 10000. In the case of a larger user base, the effect of sub-sampling is greater and more important.

Overhead of private weighting protocol. We evaluate execution time with the private weighting protocol (Protocol 1). Figure 10 shows the execution times on HeartDisease and TcgaBrca with the number of users 10 and 100, respectively, and a skewed (zipf) distribution. These two benchmark scenarios of cross-silo FL from [42] use small models. The left figure shows the time required for local training in each silo, and the right figure shows the execution time for key exchange, preparation of blinded histograms, and aggregation. As shown, the execution time of local training is dominant and it increases with a larger number of users. Overall, it shows realistic execution times under these benchmark scenarios.

Figure 11 shows the execution times of the private weighting protocol with an artificial dataset with 10000 samples and a model with 16 parameters, 20 users, and 3 silos as default, in order to show the impact of parameter size and number of users. These are on a considerably small scale. The major time-consuming parts of the protocol are key exchange, training in each silo, and aggregation on the server. The top and bottom three figures show the execution times on each part of the protocol with varying parameter sizes from 16 to 107 and number of users from 10 to 40, respectively. The execution time of local training is averaged by silos. The dominant part is the local training, which is considered to be an overhead due to the computation with the Paillier encryption, growing linearly with parameter size and the number of users. The larger parameter size increases the aggregation time on the server as well. Our implementation is based on the Python library [10], which itself could be made faster by software implementation or hardware accelerators [51]. However, it can be challenging to apply to larger models, such as DNNs, because the execution time increases linearly with parameter size. Therefore, extending the proposed method to deep models with millions of parameters is a future challenge. It may be possible to replace such software-based encryption methods by using hardware-assisted Trusted Execution Environments, which have recently attracted attention in the FL field [26, 40].

6 CONCLUSION

This study aimed to integrate user-level DP into FL, providing practical privacy guarantees for the trained model in general cross-silo scenarios. We proposed the first cross-silo ULDP FL framework where a user can have multiple records across silos. We designed an algorithm using per-user weighted clipping to directly satisfy ULDP instead of group-privacy. In addition, we developed an enhanced weighting strategy that improves the utility of our proposed method and a novel protocol that performs it privately. Finally, we demonstrated the effectiveness of the proposed method on several real-world datasets and showed that it performs significantly better than existing methods. We also verified that our proposed private protocol works in realistic time in existing cross-silo FL benchmark scenarios. For future work, research into the more scalable private protocols would be considered. Also, it would be an independent and interesting direction to empirically compare the privacy protection of user/record-level DP in FL in terms of particular attack aspects such as user/record-level membership inference [20].

ACKNOWLEDGMENTS

This work was supported by NIH R01LM013712, R01ES033241, NSF CNS-2124104, CNS-2125530, IIS-2302968, JST SICORP JPMJSC2107, JST CREST JPMJCR21M2, JST PRESTO JPMJPR23P5, JSPS KAKENHI JP22H03595, JP23K24851, JP21K19767.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[2] Naman Agarwal, Peter Kairouz, and Ziyu Liu. 2021. The skellam mechanism for differentially private federated learning. *Advances in Neural Information Processing Systems* 34 (2021), 5052–5064.

[3] Andreea B. Alexandru and George J. Pappas. 2022. Private Weighted Sum Aggregation. *IEEE Transactions on Control of Network Systems* 9, 1 (2022), 219–230. https://doi.org/10.1109/TCNS.2021.3094788

[4] Kareem Amin, Alex Kulesza, Andres Munoz, and Sergei Vassilvtiskii. 2019. Bounding user contributions: A bias-variance trade-off in differential privacy. In *International Conference on Machine Learning*. PMLR, 263–271.

[5] Borja Balle, Gilles Barthe, Marco Gaboardi, Justin Hsu, and Tetsuya Sato. 2020. Hypothesis testing interpretations and renyi differential privacy. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2496–2506.

[6] James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. 2020. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1253–1269.

[7] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1175–1191.

[8] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. 2019. Distributed differential privacy via shuffling. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38*. Springer, 375–403.

[9] Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. 2007. Efficient and secure comparison for on-line auctions. In *Information Security and Privacy: 12th Australasian Conference, ACISP 2007, Townsville, Australia, July 2-4, 2007. Proceedings 12*. Springer, 416–430.

[10] data61. [n.d.]. https://github.com/data61/python-paillier.

[11] Li Deng. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142.

[12] Cynthia Dwork. 2006. Differential privacy. In *Proceedings of the 33rd international conference on Automata, Languages and Programming-Volume Part II*. Springer-Verlag, 1–12.

[13] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[14] Alessandro Epasto, Mohammad Mahdian, Jieming Mao, Vahab Mirrokni, and Lijie Ren. 2020. Smoothly bounding user contributions in differential privacy. *Advances in Neural Information Processing Systems* 33 (2020), 13999–14010.

[15] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Shuang Song, Kunal Talwar, and Abhradeep Thakurta. 2020. Encode, shuffle, analyze privacy revisited: Formalizations and empirical evaluation. *arXiv preprint arXiv:2001.03618* (2020).

[16] Robin C Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially private federated learning: A client level perspective. *NIPS 2017 Workshop: Machine Learning on the Phone and other Consumer Devices* (2017).

[17] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2020. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems* 33 (2020), 19586–19597.

[18] Antonious Girgis, Deepesh Data, Suhas Diggavi, Peter Kairouz, and Ananda Theertha Suresh. 2021. Shuffled model of differential privacy in federated learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2521–2529.

[19] Michelle Goddard. 2017. The EU General Data Protection Regulation (GDPR): European regulation that has a global impact. *International Journal of Market Research* 59, 6 (2017), 703–705.

[20] Bargav Jayaraman and David Evans. 2019. Evaluating differentially private machine learning in practice. In *Proceedings of the 28th USENIX Conference on Security Symposium* (Santa Clara, CA, USA) *(SEC'19)*. USENIX Association, USA, 1895–1912.

[21] Kaggle. 2018. Credit Card Fraud Detection dataset. https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud. Accessed: 2023-08-03.

[22] Peter Kairouz, Ziyu Liu, and Thomas Steinke. 2021. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *International Conference on Machine Learning*. PMLR, 5201–5212.

[23] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning* 14, 1–2 (2021), 1–210.

[24] Gautam Kamath. 2020. CS 860 : Algorithms for Private Data Analysis Fall 2020 Lecture 5 — Approximate Differential Privacy. http://www.gautamkamath.com/CS860notes/lec5.pdf. [Online; accessed 23-June-2023].

[25] Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. 2021. Preventing manipulation attack in local differential privacy using verifiable randomization mechanism. In *Data and Applications Security and Privacy XXXV: 35th Annual IFIP WG 11.3 Conference, DBSec 2021, Calgary, Canada, July 19–20, 2021, Proceedings 35*. Springer, 43–60.

[26] Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. 2023. Olive: Oblivious Federated Learning on Trusted Execution Environment against the Risk of Sparsification. *Proc. VLDB Endow.* 16, 10 (aug 2023), 2404–2417. https://doi.org/10.14778/3603581.3603583

[27] Fumiyuki Kato, Li Xiong, Shun Takagi, Yang Cao, and Masatoshi Yoshikawa. 2023. ULDP-FL: Federated Learning with Across Silo User-Level Differential Privacy. *arXiv preprint arXiv:2308.12210* (2023).

[28] Daniel Levy, Ziteng Sun, Kareem Amin, Satyen Kale, Alex Kulesza, Mehryar Mohri, and Ananda Theertha Suresh. 2021. Learning with user-level privacy. *Advances in Neural Information Processing Systems* 34 (2021), 12466–12479.

[29] Seng Pei Liew, Tsubasa Takahashi, Shun Takagi, Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. 2022. Network shuffling: Privacy amplification via random walks. In *Proceedings of the 2022 International Conference on Management of Data*. 773–787.

[30] Ken Liu, Shengyuan Hu, Steven Z Wu, and Virginia Smith. 2022. On privacy and personalization in cross-silo federated learning. *Advances in Neural Information Processing Systems* 35 (2022), 5925–5940.

[31] Yuhan Liu, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Michael Riley. 2020. Learning discrete distributions: user vs item-level privacy. *Advances in Neural Information Processing Systems* 33 (2020), 20965–20976.

[32] Andrew Lowy, Ali Ghafelebashi, and Meisam Razaviyayn. 2023. Private nonconvex federated learning without a trusted server. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 5749–5786.

[33] Andrew Lowy and Meisam Razaviyayn. 2023. Private Federated Learning Without a Trusted Server: Optimal Algorithms for Convex Losses. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=TVY6GoURrw

[34] H Brendan McMahan, Galen Andrew, Ulfar Erlingsson, Steve Chien, Ilya Mironov, Nicolas Papernot, and Peter Kairouz. 2018. A general approach to adding differential privacy to iterative training procedures. *arXiv preprint arXiv:1812.06210* (2018).

[35] H Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. 2016. Federated learning of deep networks using model averaging. *arXiv preprint arXiv:1602.05629* (2016).

[36] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963* (2017).

[37] Frank D McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. 19–30.

[38] Ilya Mironov. 2017. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*. IEEE, 263–275.

[39] Ilya Mironov, Kunal Talwar, and Li Zhang. 2019. R\'enyi differential privacy of the sampled gaussian mechanism. *arXiv preprint arXiv:1908.10530* (2019).

[40] Fan Mo, Hamed Haddadi, Kleomenis Katevas, Eduard Marin, Diego Perino, and Nicolas Kourtellis. 2021. PPFL: privacy-preserving federated learning with trusted execution environments. In *Proceedings of the 19th annual international conference on mobile systems, applications, and services*. 94–108.

[41] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*. IEEE, 739–753.

[42] Jean Ogier du Terrail, Samy-Safwan Ayed, Edwige Cyffers, Felix Grimberg, Chaoyang He, Regis Loeb, Paul Mangold, Tanguy Marchand, Othmane Marfoq, Erum Mushtaq, Boris Muzellec, Constantin Philippenko, Santiago Silva, Maria Teleńczuk, Shadi Albarqouni, Salman Avestimehr, Aurélien Bellet, Aymeric Dieuleveut, Martin Jaggi, Sai Praneeth Karimireddy, Marco Lorenzi, Giovanni Neglia, Marc Tommasi, and Mathieu Andreux. 2022. FLamby: Datasets and Benchmarks for Cross-Silo Federated Learning in Realistic Healthcare Settings. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 5315–5334.

[43] Matthias Paulik, Matt Seigel, Henry Mason, Dominic Telaar, Joris Kluivers, Rogier van Dalen, Chi Wai Lau, Luke Carlson, Filip Granqvist, Chris Vandevelde, et al. 2021. Federated Evaluation and Tuning for On-Device Personalization: System Design & Applications. *arXiv preprint arXiv:2102.08503* (2021).

[44] Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. 2019. Federated learning for emoji prediction in a mobile keyboard. *arXiv preprint arXiv:1906.04329* (2019).

[45] Jinhyun So, Ramy E Ali, Başak Güler, Jiantao Jiao, and A Salman Avestimehr. 2023. Securing secure aggregation: Mitigating multi-round privacy leakage in

federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 9864–9873.

[46] Shun Takagi, Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. 2023. From Bounded to Unbounded: Privacy Amplification via Shuffling with Dummies. In *2023 IEEE 36th Computer Security Foundations Symposium (CSF)*. IEEE, 457–472.

[47] Dinusha Vatsalan, Ziad Sehili, Peter Christen, and Erhard Rahm. 2017. Privacy-preserving record linkage for big data: Current approaches and research challenges. *Handbook of big data technologies* (2017), 851–895.

[48] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. 2019. Subsampled rényi differential privacy and analytical moments accountant. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 1226–1235.

[49] Royce J Wilson, Celia Yuxin Zhang, William Lam, Damien Desfontaines, Daniel Simmons-Marengo, and Bryant Gipson. 2020. Differentially private SQL with bounded user contribution. *Proceedings on privacy enhancing technologies* 2020, 2 (2020), 230–250.

[50] Haibo Yang, Minghong Fang, and Jia Liu. 2021. Achieving Linear Speedup with Partial Worker Participation in Non-IID Federated Learning. *Proceedings of ICLR* (2021).

[51] Zhaoxiong Yang, Shuihai Hu, and Kai Chen. 2020. FPGA-based hardware accelerator of homomorphic encryption for efficient federated learning. *arXiv preprint arXiv:2007.10560* (2020).

[52] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, et al. 2021. Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint arXiv:2109.12298* (2021).

[53] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2020. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610* (2020).