Towards Accurate and Stronger Local Differential Privacy for Federated Learning with Staircase Randomized Response

Matta Varun
Indian Institute of Technology
Kharagpur, India

Shuya Feng
University of Connecticut
Storrs, CT, USA

Han Wang
University of Kansas
Lawrence, KS, USA

Shamik Sural
Indian Institute of Technology
Kharagpur, India

Yuan Hong
University of Connecticut
Storrs, CT, USA

ABSTRACT

Federated Learning (FL), a privacy-preserving training approach, has proven to be effective, yet its vulnerability to attacks that extract information from model weights is widely recognized. To address such privacy concerns, Local Differential Privacy (LDP) has been applied to FL: perturbing the weights trained for the local model by each client. However, besides high utility loss on the randomized model weights, we identify a new inference attack to the existing LDP method, that can reconstruct the original value from the noisy values with high confidence. To mitigate these issues, in this paper, we propose the Staircase Randomized Response (SRR)-FL framework, which assigns higher probabilities to weights closer to the true weight, reducing the distance between the true and perturbed data. This minimizes the noise for maintaining the same LDP guarantee, leading to better utility. Compared to existing LDP mechanisms (e.g., Generalized Randomized Response) on the FL, SRR-FL can further provide a more accurate privacy-preserving training model, and enhance the robustness against the inference attack while ensuring the same LDP guarantee. Furthermore, we also use the parameter shuffling method for privacy amplification. The efficacy of SRR-FL has been validated on widely used datasets MNIST, Medical-MNIST and CIFAR-10, demonstrating remarkable performance.[1]

CCS CONCEPTS

• **Security and privacy** → *Privacy-preserving protocols*; • **Computing methodologies** → *Machine learning algorithms.*

KEYWORDS

Local differential privacy, federated learning, client-level LDP

ACM Reference Format:
Matta Varun, Shuya Feng, Han Wang, Shamik Sural, and Yuan Hong. 2024. Towards Accurate and Stronger Local Differential Privacy for Federated Learning with Staircase Randomized Response. In *Proceedings of the Fourteenth ACM Conference on Data and Application Security and Privacy (CODASPY '24), June 19–21, 2024, Porto, Portugal.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3626232.3653279

[1]Code is available at https://github.com/matta-varun/SRR-FL.

CODASPY '24, June 19–21, 2024, Porto, Portugal

ACM ISBN 979-8-4007-0421-5/24/06
https://doi.org/10.1145/3626232.3653279

1 INTRODUCTION

Deep Neural Networks (DNNs) have significantly impacted various sectors by leveraging distributed data for insight generation and innovation. The challenge lies in training these complex models across numerous decentralized devices while ensuring data privacy [52, 62]. Federated Learning (FL) [27, 34, 66] offers a viable solution by allowing model training across clients under the coordination of a central server, without sharing raw data. This method ensures privacy by only exchanging model weights for aggregation, predicated on the trust in the aggregator's discretion. However, recent studies have shown that attackers could potentially infer sensitive and important information about the training data under certain conditions by having access to these model weight updates [4, 21, 40, 64]. The adversary can infer clients' sensitive information from their local model weights via inference attacks, such as membership inference attacks [46] (whether a sample is in the training dataset or not), property-inference attacks [41, 42] (e.g., inferring the ratio of males vs. females), and data reconstruction attacks [39, 63].

To enhance FL's privacy protection in light of recent vulnerabilities, integrating Differential Privacy (DP) [10, 11] has become a prominent strategy to protect data with provable indistinguishability against inferences with arbitrary background knowledge. Despite their advancements, DP-enhanced FL models [6, 23, 29, 49, 61] rely on a reliable aggregator for privacy, which could be a weak point if compromised. Addressing this, Local Differential Privacy (LDP) [13, 24] offers a robust alternative by perturbing data client-side, thereby safeguarding privacy even with a potentially untrustworthy aggregator. However, LDP's effectiveness depends on its randomization algorithm's indistinguishability rather than its resistance to specific inference attacks, leading to possible discrepancies in performance against such attacks [45]. While LDP ensures the theoretical conditions for privacy, its practical defense against inference in FL scenarios might not be as strong, indicating the need for further refinement in LDP applications.

While incorporating LDP in the FL framework strengthens privacy guarantees, as the number of federated rounds increases, privacy leakage is accumulated. Then, with the limited privacy budget and the high dimensionality of weights in each round, a challenge lies in the high variance of perturbed model weights due to the

diverse weight ranges in DNNs, leading to the subpar performance of the trained model. Chamikara et al. [3] discussed the sample-level privacy protection for FL, and Truex et al. [50] provided the exponential-based LDP, which considers the distance between different model weights and assigns the perturbation probabilities to improve the performance. However, their work cannot provide the strong notion of client-level LDP guarantee ("indistinguishability for all the input data provided by all the clients"). Recently, Sun et al. [48] addressed the privacy guarantee for client-level LDP, and proposed a data perturbation algorithm with adaptive ranges and a parameter shuffling mechanism to reduce privacy budget accumulation. However, due to the limited output domain in [48], input can be accurately inferred from the noisy output even if the LDP guarantee holds, as demonstrated through our experiments.

Given the shortcomings of existing LDP applications in FL, a need emerges for a method that tightly secures client privacy without sacrificing model accuracy. Thus, we introduce the Staircase Randomized Response-based Federated Learning (SRR-FL) framework, which is uniquely characterized by its strategy for converting the infinite numerical domain into a finite discrete one, partitioning the weight space into multiple groups, and having different perturbation probabilities for different groups.[2]

The intuition behind assigning higher perturbation probabilities to values closer to the true weight is to enhance the utility of the randomization while guaranteeing LDP. By reducing the expected distortion introduced by the perturbation, the adverse effects on the model's accuracy and convergence properties are minimized.

Therefore, the main contributions of this work can be summarized as follows:

(1) To our best knowledge, we take the first first to investigate the attack performance of FL with LDP. Then, we propose the first LDP framework for FL (named SRR-FL), providing a scalable and robust solution, enabling strict client-level privacy, improving the accuracy of the training model and enhancing the robustness against the inference attack.

(2) Specifically, SRR-FL addresses the challenge of high variance among noisy weights in DNNs by introducing adaptive ranges for data perturbation. It takes the first step to allocate higher perturbation probabilities to values closer to the true value, reducing the noise for better utility.

(3) Moreover, the SRR-FL framework mitigates the issue of privacy budget explosion in LDP in FL by integrating a parameter shuffling mechanism. This preserves data privacy and significantly reduces the global privacy budget (ϵ) across multiple federated rounds.

(4) The proposed SRR-FL framework is empirically evaluated on MNIST, Medical-MNIST and CIFAR-10 datasets, demonstrating its effectiveness in enhancing model performance while ensuring stringent privacy protection. Besides this, we also demonstrate the effectiveness of our inference attack on certain existing LDP in FL works and present how SRR-FL mitigates the risk of such attacks.

[2]The staircase randomized response mechanism [55] has demonstrated the state-of-the-art performance on the utility and privacy trade-off for location data with a discrete domain by considering the distance between the true location and the perturbed location while adhering to LDP principles.

The rest of the paper is organized as follows. Section 2 reviews some preliminaries, and Section 3 shows the vulnerability of the existing work on the client-level LDP for federated learning. Section 4 presents the details for the SRR-FL framework. Section 5 analyzes the privacy and utility of the SRR-FL. Section 6 demonstrates the experimental results. Section 7 reviews the related work, and Section 8 concludes the paper.

2 PRELIMINARIES

In this section, we first describe federated learning, then define our privacy notion for federated learning, the limitations of existing work, and finally provide a general overview of our proposed approach SRR-FL.

2.1 Federated Learning

Federated Learning is an approach to training deep neural models and being increasingly adopted for its widespread utility. Works such as [27], [2] and [35] have increased the attention towards FL. In a federated learning system, there are multiple edge devices containing independent datasets and a server that acts as an aggregator. First, all the client devices agree with the server on a deep neural network architecture to be trained. The server then generates a random set of parameters for the model and shares them with all the client devices who in turn initialize their local models with the same. When the federated rounds begin, in each round, the clients train the local models, update the model parameters with the local training data and upload the updated model parameters to the aggregating server. Then, the server combines all the weights updated from the client devices and create a new global model update by taking some form of weighted average. Finally, the new global model is distributed to the clients again and the next federated round begins. The process between the clients and aggregator is repeated either for a pre-determined number of federated rounds or until the model reaches a certain performance threshold.

2.2 Local Differential Privacy

Initial applications of differential privacy to federated learning [4, 16, 18, 21, 36, 40, 64] involved a central trusted aggregator utilizing the standard model of DP and not the local model. To mitigate the possibility of an untrustworthy aggregator, differential privacy was applied to FL with local noise injection in recent works [25, 30, 45]. Essentially, Local Differential Privacy (LDP) [13, 24] is a stronger privacy model of DP [9–11] to conduct private data collection and was adopted by organizations such as Google RAPPOR [13] and Microsoft Telemetry [8] for their data collection. LDP does not require a trusted data aggregator and it perturbs data locally before sending it to the server for processing or analysis. Specifically, instead of sharing the raw values, users use an LDP mechanism Φ to perturb the raw value w locally and instead share $\Phi(w)$. This prevents an attacker from successfully carrying out attacks with the shared values. This can be formalized as follows,

Definition 1 (ϵ-LDP). *A randomized algorithm Φ satisfies ϵ-local differential privacy, if and only if for any pair of inputs $w_1, w_2 \in D$ and any output y of Φ,*

$$\frac{Pr[\Phi(w_1) = y]}{Pr[\Phi(w_2) = y]} \leq e^{\epsilon}$$

The strength of privacy guaranteed by Φ is determined by the privacy parameter ϵ, also known as the privacy budget. A lower value of ϵ indicates stronger privacy guarantees.

3 VULNERABILITY IN CURRENT CLIENT-LEVEL LDP MECHANISM FOR FL

Although federated learning succeeds in preventing direct access to the private data of clients, recent works have proven that inference attacks can be carried out to infer the underlying training data using model parameters or the outputs of the trained model [40, 46, 51].

To the best of our knowledge, [48] was the only paper that achieves acceptable model performance for client-level LDP at a highly affordable privacy budget. This can be attributed to their fairly simple data perturbation algorithm consisting of only two output values that are independent of the specific input. This led to the introduction of a mathematical guarantee of "zero bias" in the model weights, thereby providing great model performance. However, in spite of their client anonymization technique and ϵ-LDP guarantees, we found certain situations where the server would be able to infer information about a common property in datasets across the clients, which may be private to the individual clients. We will explain in detail how the attack works and how our proposed SRR-FL algorithm mitigates the risk of this problem drastically.

3.1 Design of the Inference Attack

In LDP-FL [48], the data perturbation algorithm is given by,

$$\text{LDP-FL} : M(w) = \begin{cases} c + r \cdot \frac{e^\epsilon+1}{e^\epsilon-1}, & \text{w. prob. } \frac{(w-c)(e^\epsilon-1)+r(e^\epsilon+1)}{2r(e^\epsilon+1)}, \\ c - r \cdot \frac{e^\epsilon+1}{e^\epsilon-1}, & \text{w. prob. } \frac{-(w-c)(e^\epsilon-1)+r(e^\epsilon+1)}{2r(e^\epsilon+1)} \end{cases} \quad (1)$$

As mentioned before, it only has two possible output values, that are independent of the input and only depend on the privacy budget ϵ, the central value of the domain c, and the radius of the domain r. Suppose that a high number of clients conduct federated learning with a high client selection rate (you select a big fraction of the clients to proceed with training in every round) and there exists a common property P across the datasets distributed over the clients which always lead the model weight at a particular location to take up a value very close to w after every training round. This common property can be in the form of presence of a particular kind of training samples in the datasets across the clients. In such a situation, all participating clients would invoke the LDP-FL data perturbation algorithm with nearly the same input value and would only output one of the two expected output values.

These perturbed weights would provide privacy protection for the clients. However, the untrusted server can now evaluate the distribution of these perturbed weights and store them. Over the several federated rounds being conducted, the server would keep storing all the updates from clients. After a fairly large number of rounds, the server will use the historical data it has collected over the rounds and can, with a high probability, conclude that the raw weight would have been. This is because the server knows the values of c, r and ϵ, and with the repeatedly confirmed distribution of the noisy weight updates, it can work out the equations of the data perturbation algorithm to approximately determine the original weight value w. Let

$$p = \frac{\text{number of } (c + r \cdot \frac{e^\epsilon+1}{e^\epsilon-1}) \text{ updates}}{\text{total number of updates}}$$

where p is the fraction of one of the updates for a given weight w in the model. Then,

$$w^i_{\text{predicted}} = c + \frac{2r \cdot (e^\epsilon + 1) \cdot p - r \cdot (e^\epsilon + 1)}{e^\epsilon - 1} \quad (2)$$

where $w^i_{\text{predicted}}$ is the predicted value for w in the i^{th} federated round.

The exact mechanism through which the server verifies or validates its prediction for the current weight is as follows. Let us denote the two output values from the LDP-FL perturbation mechanism as L and R. For example, the server initially has no historical data with it and only has one round's data, which is the frequency of L and frequency of R from the clients after round one. At this stage the server computes p as $p = \frac{R}{L+R}$ and uses this value to compute $w^1_{\text{predicted}}$. Note that the sum of the number of responses for L and R is equal to the product of the fraction of clients selected in each round (f_r) and the total number of clients (n). When the next round of federated learning begins, the server follows the steps below to predict and validate the current weight value.

- **Step 1**: The server computes the frequency of L and R responses for the latest round. It pushes these results into it's historical data for this particular weight.
- **Step 2**: The server randomly splits the historical data into two samples of equal size. Let us denote them as sample_1 and the sample_2.
- **Step 3**: The server computes L^i_1 as the mean of all L frequency values in the entries of sample_1, and R^i_1 as the mean of all R frequency values in the entries of sample_1. Similarly, L^i_2 and R^i_2 are computed from sample_2.
- **Step 4**: The server computes the value $w^i_{\text{predicted\_1}}$ using p^i_1 (which is computed using L^i_1 and R^i_1) and the value $w^i_{\text{predicted\_2}}$ using p^i_2 (which is computed using L^i_2 and R^i_2). The smaller the difference between $w^i_{\text{predicted\_1}}$ and $w^i_{\text{predicted\_2}}$, the more confidence with which the server concludes that the actual w value is either the same or very close to $w^i_{\text{predicted}}$ (computed using overall p (L^i and R^i)).

3.2 Performance of Inference Attack

To assess the scope and strength of this kind of attack, we simulate the attack on the LDP protocol following the design of [48], with varying numbers of clients, domain sizes, privacy budgets and number of rounds. The metric for attack performance is the error in prediction with respect to the domain size, which is given by the ratio of the absolute difference between the original and predicted mean weight and the size of the domain. Note that a lower value denotes better attack performance. The attack setting is detailed in the Appendix. As shown in Table 1, with $\epsilon = 1$, after 50 federated rounds, the server was able to filter down the location of the original w value to 0.46% of the domain. This means that if our domain had 1000 possible values for w, our attack filtered it down to nearly

4 possible values. As the ϵ becomes larger, the predicted weight is closer to the true weight since the prediction error is less (e.g., 0.28%). Therefore, as long as there exists a global property among the datasets, that leads a given set of weights to converge to the same value every time, the design of the perturbation algorithm in [48] is unable to protect those weights from leaking with an affordable value of ϵ. These predicted weight values can be sufficient for the attacker to achieve the membership inference attacks [46] and infer sensitive information about the local datasets on the client side, which entirely compromises the motivation behind using local differentially private FL. Increasing the input domain size might raise the absolute error in our prediction, but it is known that a larger input domain results in higher variance in noisy weights, leading to poor model performance. More detailed results are given in the Appendix.

Table 1: Prediction error for inference attacks on [48] vs. ϵ (w.r.t. domain size)

ϵ	1	2	3	5	8	10
rounds=50	0.46%	0.33%	0.31%	0.28%	0.28%	0.29%

Summary. The root cause of this problem is that the server knows that there are only two possible outputs from the data perturbation algorithm, where the count distribution of two outputs causes privacy leakage. With a high number of clients, the server is able to roughly reconstruct the probability distribution of the LDP mechanism which would have been used by the clients to perturb the data. Once the probabilities are known, it is a matter of re-working the equations to arrive at w. Thus, the output domain is very important in protecting the weight privacy of clients. SRR-FL, as discussed in the following section, combats this problem by considering the entire input domain as possible output values.

4 FRAMEWORK OF SRR-FL

4.1 Overview of SRR-FL

We now illustrate the major components of SRR-FL. SRR-FL system is made up of N edge devices hosting independent training data and one server that acts as an aggregator of the parameter updates and conducts the training process in a federated fashion.

(1) **Local Update**: The clients first receive updated model weights from the server along with any other relevant messages. These weights are used to update the local models. The devices then start training the models in parallel using their local datasets and optimize their model weights. After the training finishes, the model weights are perturbed using the Staircase Randomized Response scheme (discussed from 4.2). The perturbed model weights would satisfy ϵ-LDP and make weights indistinguishable accordingly. These perturbed weights are sent to the server using a stripping and shuffling scheme, which keeps the identity of the incoming updates anonymous. The details of the shuffling mechanism for federated learning are given in Section 4.5.

(2) **Server Update**: In the $k_i^{th}, i \in [1, N]$ federated round, the server randomly selects m edge devices to go ahead in this round and sends them the current global model (model on the server) weights (where $m \leq N$). These m devices then perform local updates independently. After receiving the noisy values, aggregation is done by the server to update the global model weights. The updated weights are also returned back and shared with these edge devices so that they update their local models. Our approach assumes that the identity of the edge devices is unknown to the server, hence they remain anonymous.

4.2 Staircase Randomized Response

Recall that LDP-FL [48] applies LDP to federated learning, which has only two perturbation probabilities (Equation 1). As a result, the lack of fine-grained perturbation probabilities and outputs can limit the optimization of utility. Motivated by the current limitations in privacy protection and utility maximization, we propose an LDP framework for federated learning, named the Staircase Randomized Response Federated Learning (SRR-FL). This new approach balances strict privacy protection with enhanced utility. SRR-FL benefits from a novel Staircase Randomized Response (SRR) [55] mechanism. The SRR mechanism differentiates by allocating various perturbation probabilities across different value groups within the domain, an approach proven successful in location-based services. The intuition behind the mechanism is that assigning higher perturbation probabilities to values closer to the true value enhances the randomization scheme's performance and utility while maintaining ϵ-LDP guarantees. In order to implement this, it is necessary to assess and quantify the distance between the input value w and the output value y. As a result, the set of perturbation probabilities should be predetermined for all possible output values y given a specific input value w.

We formally define the perturbation probabilities of all possible output values from a given input value w as follows based on SRR [55]. For any input value w in the given domain D, all possible output values can be partitioned into m groups $G_1(w), G_2(w), \ldots, G_m(w)$ based on their distance from w. It is important to note that the partitioning here is dependent on the input value w. Therefore, for each input value w, the m groups and their perturbation probabilities will be computed as:

$$\text{SRR} : \forall w \in D, p(y|w) = \begin{cases} \alpha_1(w), & \text{if } y \in G_1(w) \\ \vdots & \vdots \\ \alpha_m(w), & \text{if } y \in G_m(w) \end{cases} \quad (3)$$

where $\alpha_1(w), \alpha_2(w), ..., \alpha_m(w)$ are the perturbation probabilities of the m groups partitioned from w based on their distance. The difference between the perturbation probabilities of two adjacent groups is also the same among $\alpha_1(w), ..., \alpha_m(w)$. Note that the sum of perturbation probabilities for each input value w should satisfy: $\Sigma_{i\in[1,m]}\Sigma_{y\in G_i(w)} p(y|w) = 1$.

Revised SRR for FL (SRR-FL). There are two key steps of the SRR mechanism: (1) the computation of optimal group numbers m value; (2) the group partitioning scheme with encoding the coordinates into bits by a hierarchical structure. The SRR mechanism deals with a finite set of discrete locations, which is similar to most LDP mechanisms. However, in this work, the domain of model weights is unknown and can be any real value. Thus the domain should be an infinite real value. If LDP were directly applied to an infinite domain,

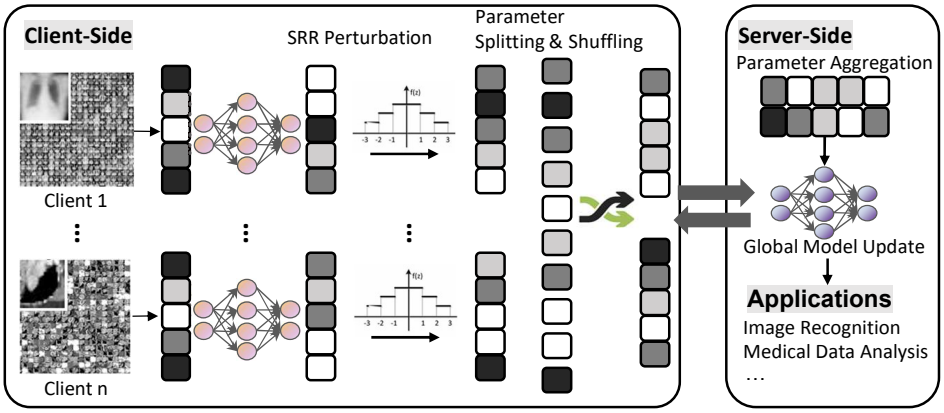


Figure 1: Overview of SRR-FL: There are n clients, each of which holds some local training data. The local models are trained on local data, perturbed with SRR, and uploaded to the server (after shuffling). The server aggregates these updates and derives the global model, which can be shared with the clients. These steps repeat until convergence.

the utility would be sacrificed significantly as a result of the large domain. To overcome this limitation, we convert the continuous range of values under consideration into a set of discrete values using some parameters and have different schemes to compute the perturbation probabilities, discussed in Section 4.3. Besides, the mechanism SRR in [55] can only protect the values of users, which can still lead to side-channel attacks identifying users/devices based on the other information of a system (e.g., time, power) [26, 54]. However, SRR-FL could provide client-level protection that protects both the value and identity of each device.

4.3 Domain Discretization and Partition

Given the model weights are real values in our application, a distinct approach is required to categorize the domain into groups for a specific weight w. Initially, the domain of infinite real numbers is transformed into a finite set of discrete values, facilitating the application of the LDP mechanism through domain conversion parameters.

Specifically, we first consider the parameters c and r, which limits our initial domain spanning $(-\infty, \infty)$ to $(c - r, c + r)$. The values of c and r can be varied for a specific application. We assume that the values of a particular model weight always fall in the proposed range and accordingly decide c and r for it. Varying these parameters for different weights in the model helps us prevent high variance among perturbed weight values which is a commonly prevalent issue when applying LDP to deep neural networks. For weights that do not vary hugely, r can be set to a lower value as compared to weights that vary by huge margins.

Using parameter r makes the considered domain finite in terms of boundaries, but still does not make the domain discrete. This motivates us to consider a precision parameter p, using which we divide the domain $[c - r, c + r]$ into a set of discrete values starting from $c - r$ up to $c + r$ with a precision of 10^{-p}. Therefore, the parameter p converts the continuous range $[c - r, c + r]$ into a set of discrete values in the same range with a precision limited to the p^{th} decimal place. To avoid dealing with floating point numbers, we can map each value in this range to an integer in the range $(c \cdot 10^p - r \cdot 10^p, c \cdot 10^p + r \cdot 10^p)$.

Note that we approximate c and r to the nearest decimal with precision 10^p before determining the domain. Using both these parameters converts the continuous infinite real domain to a discrete finite set of real values. Existing LDP mechanisms based on randomized response can now be intuitively applied to the transformed domain. To apply SRR, we still need to partition this transformed domain into m groups based on the distance from input value w.

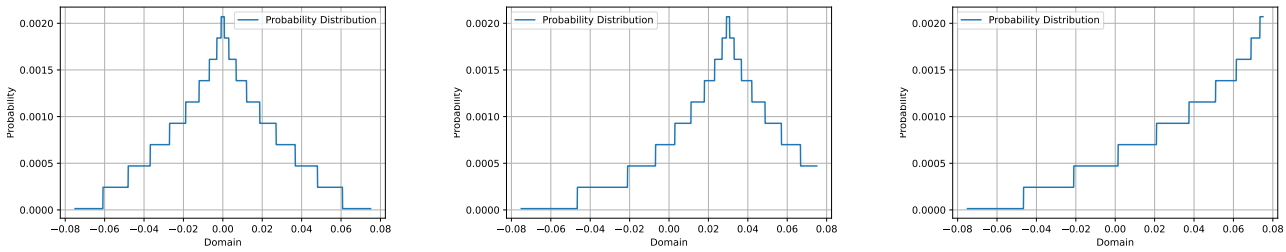
Utilizing numerical differences to measure the distance between values in our domain allows for flexibility in assigning group sizes. The naive approach would be to divide the domain into m equally-sized groups. However, this approach does not harness the true capability of SRR. Hence, the sizes of groups will be considered to be an arithmetic progression, with the size increasing by a constant value $\Delta(d)$ from the closest group to the farthest group to w. Formally, the sizes of the groups are going to be of the form:

$$|G_j(w)| = |G_1(w)| + z \cdot \Delta(d) \tag{4}$$

where $j \in [1, m]$ and $z = 0, 1, 2, ..., m-1$. $|G_1(w)|$ denotes the size of the closest group to w and $\Delta(d)$ denotes the constant group size difference between adjacent groups. Hence the m groups would be $G_1(w), G_2(w), ..., G_m(w)$, where the group size increases by $\Delta(d)$ from left to right. This approach increases the size of farther groups whose perturbation probability is already diminished. This effectively decreases the perturbation probabilities for values distant from w by a large margin, thereby further improving the utility of SRR in its application to FL. The value of $\Delta(d)$ has to be strictly less than an upper limit for a given m, p and r to prevent $|G_1(w)|$ from being zero. The maximum of $\Delta(d)$ can be computed as:

$$\Delta(d)_{max} = \frac{2 \cdot (2 \cdot r \cdot 10^p + 1)}{m \cdot (m - 1)} \tag{5}$$

Once a suitable $\Delta(d)$ has been determined, the sizes of all groups for a given m, r and p can be computed. The size of the closest



(a) Distribution for $w = 0$ **(b) Distribution for $w = 0.03$** **(c) Distribution for $w = 0.075$**

Figure 2: Example of perturbation probability distribution in SRR-FL ($c = 0.0$, $r = 0.075$) for different input values.

group $|G_1(w)|$ can computed as follows,

$$|G_1(w)| = \frac{1}{2} \cdot \left[\frac{2 \cdot (2 \cdot r \cdot 10^p + 1)}{m} - (m-1) \cdot \Delta(d) \right] \tag{6}$$

Consequently $|G_2(w)|, |G_3(w)|, \cdots, |G_m(w)|$ can be calculated by adding $\Delta(d)$ successively.

Therefore, for every layer in our deep neural network, we fix the parameters m, c, r and p independently. This allows for a flexible data perturbation mechanism that is computationally efficient and quick while keeping strong local differential privacy guarantees.

4.4 Staircase Perturbation Probabilities

There exist previous works applying the notion of a staircase mechanism in differential privacy [17]. Notice that the Probability Density Function (PDF) of SRR has a similar shape to the PDF in the above work, which utilizes the staircase mechanism for different groups to satisfy ϵ-DP. Recall that in our mechanism the possible output values can be partitioned into m groups based on their distance to the input value w. Now the formal definition of the perturbation probabilities of the output values based on w is as follows.

For any two output values y_1 and y_2 belonging to neighboring groups, $y_1 \in G_j(w)$ and $y_2 \in G_{j+1}(w)$, we have the perturbation probability $q(y_1|x) = q(y_2|x) + \Delta(w)$, where $\Delta(w) \in [0, 1)$ is the constant probability difference between values in any neighboring groups for a given input w. This ensures that probabilities are highest for values nearest to w and decrease for those increasingly distant within the domain. Therefore, The perturbation probability for an input w to an output y decreases as y moves to further groups with respect to w.

Let us denote $\alpha_{max}(w)$ and $\alpha_{min}(w)$ as the maximum and minimum probabilities among the set of perturbation probabilities for m groups respectively. Then we have $\alpha_{max}(w) = \alpha_1(w)$ and $\alpha_{min}(w) = \alpha_m(w)$. Note that $k = \frac{\alpha_{max}(w)}{\alpha_{min}(w)} \geq 1$. Thus,

$$\Delta(w) = \frac{\alpha_{min}(w) \cdot (k-1)}{m-1} \tag{7}$$

Notice that the sum of perturbation probabilities for all possible outputs is 1. It is worth noting that, in SRR for locations [55], the sizes of the groups $|G_1(w)|, |G_2(w)|, \cdots, |G_m(w)|$ may be different for different location w. Unlike it, the sizes of the groups $|G_1(w)|, |G_2(w)|, \cdots, |G_m(w)|$ are the same for different $w \in D$ in SRR-FL. Thus k should be bounded by e^ϵ (privacy bound in Section 5.1). To fully utilize the privacy budget, the value of k should be equal to e^ϵ.

Therefore, given the differences of perturbation probabilities for output values in different groups in Equation 7, sizes of different groups, the perturbation probabilities can be derived as follows,

$$\alpha_{min}(w) = \frac{m-1}{(m-1) \cdot d \cdot e^\epsilon - (k-1) \cdot \Sigma_{j=2}^{m}[(j-1) \cdot |G_j(w)|]}$$
$$\alpha_{max}(w) = e^\epsilon \cdot \alpha_{min}(w) \tag{8}$$
$$\alpha_i(w) = \alpha_1(w) - (i-1) \cdot \Delta(w)$$

where d is the domain size which is equal to $2 \cdot r \cdot 10^p + 1$.

Adaptive Range Data Perturbation in SRR-FL. Given the weights W of a model, our algorithm M returns a perturbed W^*, by randomizing each weight in W using SRR-FL (subject to the parameter constraints of the weight). Every $w_i \in W$ have the parameters c_i, r_i, p_i, m_i that are globally agreed upon. Then, our mechanism M perturbs the weights of a model with Equation 3. Specifically, with the given fixed parameters m, r, c, p, we can learn the finite domain size and the size of each group $|G_1(w)|, |G_2(w)|, \cdots, |G_m(w)|$ with Equation 4, 5, and 6. The perturbation probabilities $\alpha_1(w), \alpha_2(w), \cdots, \alpha_m(w)$ can be determined with corresponding group size and given ϵ. The example of perturbation probability distribution in SRR-FL is illustrated in Figure 2. Thus, the perturbation probability distribution is constructed completely. Note that the group sizes in our mechanism are independent of the input value w and hence remain constant given that we fix our domain range and precision. After each round, these parameter values m, r, c, p can be re-evaluated for every model weight and changed according to any new information.

4.5 Parameter Shuffling

In federated learning, the requirement for clients to send parameter gradient updates to the server across several rounds, with SRR-FL preprocessing each update, significantly increases privacy budgets due to sequential composition [11]. This escalation is more pronounced over numerous training rounds, weakening privacy guarantees. A solution involves anonymizing the link between updates and their originating clients across rounds, thus improving privacy by disassociating consecutive updates from individual clients.

The implementation of this strategy varies based on server-client coordination.

Note that there exist previous works [18, 20, 32] that have employed model shuffling, which works fairly well for simple neural networks. However, when LDP is applied to more complex deep neural networks participating in FL, this mechanism fails to prevent the explosion of the privacy budget due to the high dimensionality of the model parameters. To address this issue, a parameter shuffling mechanism, as introduced in [48], is applied. This mechanism is implemented through a two-step process. First, the model is split into its constituent weights, and each weight is then tagged with an identification marker that uniquely identifies the weight location in the model structure. The next step involves sampling a random latency t from the uniform distribution $U(0, T)$, where $T > 0$. This latency is sampled for each and every weight and the client waits for $T_{\max} + t_j$ time before sending the j^{th} model parameter. Here, $T_{\max} = max_j(T^j_{\text{comp. time}} + T^j_{\text{comm. time}})$, where $T^j_{\text{comp. time}}$ stands for the local computation time of the j^{th} client and $T^j_{\text{comm. time}}$ stands for the communication delay of the j^{th} client. t_j is the random variable sampled from the uniform distribution $U(0, T)$, where T is agreed upon between the clients before the iteration starts. In other words, $T_{\max}$ is the time the slowest client would take to respond back when there is no parameter shuffling. Therefore, the delay of the updates would depend upon the slowest client in our system. Note that we are only adding a constant value of a random variable sampled from a uniform distribution and hence the delays are still random and uniformly distributed. The value of $T_{\max}$ can also be computed before the start of each round by using the client's report of their communication and computation times based on their communication settings and hardware capabilities.

The approach can support a small T value, as long as the clients and the server support it enough to add necessary randomness to the shuffling. However, it is better to set it to a slightly larger value to accommodate things such as synchronization issues, randomness in local computation and unforeseen communication delays.

5 PRIVACY UTILITY AND ANALYSIS

5.1 Privacy Analysis

For the perturbation mechanism for model weights, we have the following privacy bound.

Theorem 5.1. *With the given m, k, d, Staircase Randomized Response (SRR) satisfies ϵ-local differential privacy in each round, where*

$$\epsilon = \log(k) \tag{9}$$

Proof. In SRR-FL, for any pair of inputs w_1, w_2 from our domain and output value y, the maximum perturbation probability $q(y|w_1)$ (the maximum probability of y being sampled from the input w_1) is given by $\alpha_{max}(w_1)$, which is the output value sampled from the closest group w.r.t. w_1. Similarly, the minimum perturbation probability $q(y|w_2)$ (the minimum probability of y being sampled from the input w_2) is given by $\alpha_{min}(w_2)$, which is when the output value is from the farthest group with respect to w_2. Therefore, the maximum value of $\frac{P[\Phi(w_1)=y]}{P[\Phi(w_2)=y]}$ is given by $\frac{\alpha_{max}(w_1)}{\alpha_{min}(w_2)}$. We know that according to our definition of perturbation probabilities and group sizes, the values α_{max} and α_{min} are independent of the input value w. We also know that $\alpha_{max}(w) = k \cdot \alpha_{min}(w)$. Hence, the value of $\frac{P[\Phi(w_1)=y]}{P[\Phi(w_2)=y]}$ is bounded by k, which means ϵ is $\log(k)$. □

Hence, given our target privacy budget ϵ, we can calculate the perturbation probability ratio k as e^{ϵ}. Note that ϵ alone does not determine the privacy, but parameters r and p also determine the scale of the domain d using which the raw value has been randomized.

Privacy Guarantee with Parameter Shuffling. Parameter Shuffling is an important aspect of SRR-FL as it removes the curse of high dimensionality among deep neural networks with respect to privacy budget explosion. With our shuffling approach, the parameters are split and anonymously uploaded to the server with random latencies sampled in between. This ensures that the server cannot link any two updates as being from the same client, either within the same round or across different rounds, which relieves the sequential composition. Additionally, parameter shuffling also provides privacy amplification [12], further enhancing the privacy of clients' data. Shuffling after local randomization can adjust the privacy bound of central model (which can be converted from LDP) to (ϵ', δ)-DP in the server side, where $\epsilon' = 12\epsilon\sqrt{\frac{\log(1/\delta)}{N}}$ and N denotes the number of clients. Recent studies [14, 20] have refined the assessment of this amplified privacy, enabling more precise privacy budget management. Incorporating these enhanced privacy bounds with sophisticated privacy accounting methods [56, 60] further diminishes the cumulative privacy budget, significantly bolstering data protection in federated learning environments.

Mitigating the Risks against the Designed Inference Attack. Even if the number of clients is so high as to match the size of our current input domain, it is practically impossible to estimate the frequency of all possible output values and re-construct the probability distribution. SRR-FL also facilitates shaping of the input and output domain through precision parameter p and increasing it exponentially growing the concerned domain and preventing the server from carrying out such attacks, even from updates of multiple rounds. Additionally, the fact of "zero bias" to the weight values in [48] was also leveraged in our attack. SRR does not have such a guarantee and the bias introduced by SRR-FL depends on the weight values w_i, which is unknown to the server. Hence, SRR-FL not only offers impressive model performance but does so with affordable privacy guarantees that are robust and difficult to break.

5.2 Utility Analysis

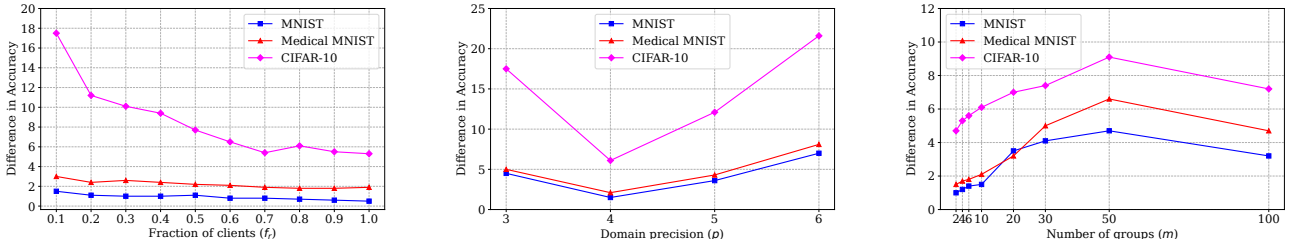
Our utility analysis focuses on the error bound, ensuring that the model maintains high accuracy with strict LDP guarantee.

Theorem 5.2. *The expected value of L_1 error bound for any client in one round of SRR-FL is bounded by $L_1 \leq r$.*

Proof. The expected value of the L_1 error bound for a parameter w on a client i is given by,

$$E[|w_i - w_i^*|] = \sum_{j \in [1,m], w_i^* \in G_j} (|w_i - w_i^*| \cdot \alpha_j(w_i))$$

This is L_1 error bound for one parameter w_i on the i^{th} client. To get the L_1 error bound for any client in one round, we take the



(a) Effect of f_r across datasets **(b) Effect of p across datasets** **(c) Effect of m across datasets**

Figure 3: Effect of parameters f_r (fraction of clients selected), p (domain precision) and m (number of partition groups in SRR-FL) across different datasets.

maximum overall clients' L_1 error bounds for w. That is, for N clients, the L_1 error bound would be,

$$L_1 = \max_{i=1}^{N}(E[|w_i - w_i^*|])$$

Hence, we consider the case where the value $E[|w_i - w_i^*|]$ will be the maximum in case that w_i is equal to either of the extreme values in our clipped discrete domain. That is, when $w_i = c - r$ or when $w_i = c + r$. Without loss of generality, let us assume $w_i = c + r$. Then, the value of $E[|w_i - w_i^*|]$ turns out to be

$$E[|w_i - w_i^*|] = \sum_{j\in[1,m]], w_i^*\in G_j(w_i)} \left(|w_i - w_i^*| \cdot \alpha_j(w_i)\right)$$

$$= \sum_{j\in[1,m]]} \left(\sum_{w\in G_j(c+r)} \left(|c + r - w| \cdot \alpha_j\right) \right)$$

In the worst setting of SRR-FL (that is when $k = 1$), all the perturbation probabilities converge to a single value, that is, $\alpha_1 = \alpha_2 = \cdots = \alpha_m = \frac{1}{d}$, where d represents the size of the domain. Then the expression simplifies into,

$$E[|w - w^*|] = \frac{1}{d} \cdot \sum_{w\in D} (|c + r - w|) = \frac{1}{d} \cdot \sum_{i=0}^{i=2\cdot r\cdot 10^p} (i \cdot 10^{-p})$$

$$= \frac{1}{2 \cdot r \cdot 10^p + 1} \cdot \sum_{i=0}^{i=2\cdot r\cdot 10^p} (i \cdot 10^{-p}) = r$$

Therefore, in the worst case, the expected value of L_1 is upper bounded by r. Thus, this completes the proof. □

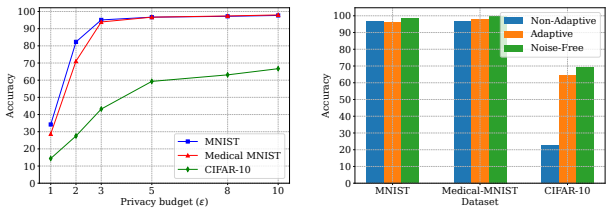
6 EXPERIMENTS

6.1 Experimental Settings

Our SRR-FL framework was evaluated using three benchmark image datasets: MNIST [7], Medical-MNIST [65], and CIFAR-10 [28], focusing on image classification tasks. We utilized a two-layer neural network for MNIST and adapted VGG-like models for Medical-MNIST and CIFAR-10. Experiments varied parameters to assess their impact on SRR-FL, ensuring equal class distribution across clients' datasets. The setup followed existing settings used in work [34] for client number N and selection ratio f_r, with all model weights confined to specified domains for consistency. We explored the framework's utility through four experimental groups, examining classifying accuracy over federated rounds, client numbers, and privacy budgets (ϵ is used for each round before shuffling; the total privacy loss can be significantly reduced compared to sequential composition after applying shuffling [12] and advanced composition or budget accountants, e.g., [14, 20, 56, 60]), and the influence of various parameters. Comparisons against the GRR (which is a commonly adopted LDP mechanism with some optimality, e.g., for relatively large ϵ) and a noise-free scenario highlighted SRR-FL's enhanced utility and privacy.

6.2 Performance on Parameters

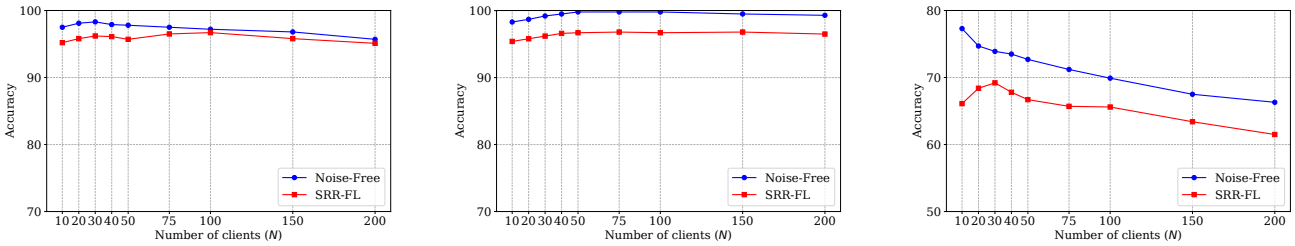
In our initial group of experiments, we assessed how selecting different fractions of clients (f_r), domain precision (p), and partition group numbers (m) in SRR-FL influence model accuracy compared to a noise-free model, fixing $N = 100$. As shown in Figure 3a, increasing f_r notably improves CIFAR-10 performance but has a minor effect on MNIST and Medical-MNIST. Optimal domain precision was found at $p = 4$ for all datasets, optimizing SRR-FL's performance (see Figure 3b). Furthermore, adjusting m revealed that accuracy benefits from lower m values, particularly beyond $m = 75$ (see Figure 3c). Consequently, future tests will utilize high f_r, $p = 4$, and low m.



(a) Effect of ϵ **(b) Effect of adaptive setting**

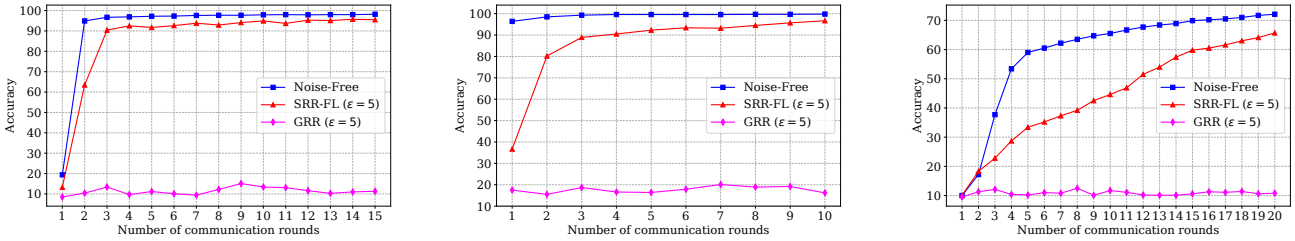
Figure 4: Effect of privacy parameter ϵ and adaptive setting across different datasets.

The second set of experiments varied the privacy budget (ϵ) to observe accuracy changes, demonstrating improved performance with higher ϵ values across datasets (Figure 4a). For MNIST, SRR-FL achieved 94.3% accuracy at $\epsilon = 3$ and 96.2% at $\epsilon = 5$. Medical-MNIST also reached up to 94% accuracy at $\epsilon = 3$. Examining the adaptive setting's impact, SRR-FL showed significant improvements, particularly for CIFAR-10, where accuracy jumped from 22.4% to 64.2% with adaptive ranges (Figure 4b). This highlights the effectiveness of adaptive range settings, especially benefiting CIFAR-10 performance, aligning closely with noise-free model accuracy.



(a) Effect of N on MNIST (b) Effect of N on Medical-MNIST (c) Effect of N on CIFAR-10

Figure 5: Comparison results of number of clients (N) across different datasets.



(a) LDP mechanisms on MNIST (b) LDP mechanisms on Medical-MNIST (c) LDP mechanisms on CIFAR-10

Figure 6: Comparison results of different LDP mechanisms with federated learning across different datasets.

6.3 Comparative Evaluation of Models

In the third group of experiments, we first compare the performance of the proposed SRR-FL with the noise-free model and see the utility with varying client numbers N, as shown in Figure 5. The accuracy of SRR-FL is all beyond 95% with different client numbers for MNIST and Medical-MNIST datasets. The accuracies of SRR-FL are all beyond 60% with different client numbers for the CIFAR-10 dataset. It is worth noting that the accuracy difference between the SRR-FL and noise-free model for all three datasets is very slight no matter what the client numbers are. Thus, we can conclude that the parameter N slightly affects the SRR-FL performance.

To compare the performance of SRR-FL with another LDP mechanism for FL, we fix the client number $N = 100$ and see the performance of the proposed SRR-FL with the noise-free model and GRR model, varying communication round number R. As shown in Figure 6a, the accuracy of the proposed SRR-FL increases with more communication rounds and has a similar trend with the noise-free model. Specifically, SRR-FL can achieve 96.2% accuracy on MNIST, which is close to noise-free setting (97.9% accuracy) at around the same number of communication rounds. The GRR scheme for FL is not able to retain this level of accuracy since the accuracy is around 10% to 20%. Thus the performance of SRR-FL outperforms the existing GRR mechanism in federated learning. We can observe the same trend in the Medical-MNIST and CIFAR-10 datasets, shown in Figure 6b and 6c.

Although we have above similar trends among three datasets, on the CIFAR-10 dataset, SRR-FL can only achieve 64.2% accuracy with a privacy budget of $\epsilon = 5$ after 20 rounds. For reference, noise-free FL achieved 69.5% accuracy in the same number of rounds. Thus, you can see the accuracy increases very slowly for the CIFAR dataset compared to the other two datasets. For SRR-FL, it generally requires 3 and 10 communication rounds to achieve near-optimal accuracy of the noise-free model for MNIST and Medical-MNIST datasets. This can be justified as it is trivial to notice that more complex model training on complex datasets in FL would require more communication rounds to converge to an optimal model.

It is worth noting that Bhowmick et al. [1], the first to apply LDP with FL, suffered with high variance from their approach and as a result required a very high privacy budget (around $\epsilon = 500$ for MNIST) and also many communication rounds ($R = 200$). [50] used α-CLDP as their LDP mechanism and achieved 86.93% accuracy on the FMNIST dataset. However, their approach also suffered from an exploding privacy budget for achieving this performance, leading to a very poor privacy guarantee. With strict privacy protection, the accuracy should be worse. Focusing on a shuffled model for LDP-FL, the work by [19] achieved an accuracy of approximately 76.7% on the MNIST dataset with a privacy budget of $\epsilon = 5$, without prioritizing accuracy enhancement.

6.4 System Performance

SRR-FL includes a pre-computation step where depending on the parameters, the server would need to compute the perturbation probabilities, group sizes and group boundaries for every possible input x. However, unlike native SRR [55], in our framework, the group sizes are independent of the input given, which makes the pre-computation step comparatively easier. It is also important to note that the pre-computation step is usually carried out before the federated learning steps start, and therefore would not affect the

latencies that could occur during FL. In our experiments, we were able to finish the pre-computation step within a few seconds even for domains with a high precision value.

The latency would be during the communication between clients and server, which is bounded by $n_w \cdot T$, where n_w is the number of parameters and T is the value we are using for the uniform distribution to sample latencies. An experiment on MNIST (10 communication rounds) could be completed within 60 - 90 minutes.

The models designed for Medical-MNIST and CIFAR-10 were much larger, with the model for Medical-MNIST consisting of $5,611,878$ trainable parameters and the model for CIFAR-10 consisting of $2,169,770$ trainable parameters. For both, perturbing all the model weights also causes a considerable latency on top of training. On average, we observed an average latency of 0.698 seconds on the client side for MNIST ($20,490$ trainable parameters), an average latency of 75.98 seconds on the client side for CIFAR-10 ($2,169,770$ trainable parameters) and an average latency of 158.45 seconds for Medical-MNIST dataset ($5,611,878$ trainable parameters). As seen from the results, the latency introduced by the application of SRR-FL on the client side varies linearly with the model size.

In the pre-computation step for SRR on the server, additional memory is utilized to store computed values, which might exceed the requirements of other LDP implementations in Federated Learning FL. However, this extra memory usage is minimal, especially when considering the server's role in managing and storing all incoming model updates from various clients.

7 RELATED WORK

Federated learning, aimed at protecting user data privacy, faces significant security and privacy challenges. Yang et al. [66] proposed secure federated learning to address these issues, underscoring the necessity for robust security in distributed learning environments. Bonawitz et al. [2] further examined the high-level design challenges and potential solutions in secure federated learning, identifying open problems and future research directions.

In the past decade, Differential Privacy (DP) [10, 11] has been recognized as the de facto rigorous privacy solution for learning algorithms [38, 53] in many different domains, such as numeric data [58], texts [22], videos [57], graphs [44], trajectories [31], and streaming data [15]. Thus, in the context of federated learning, there are existing works [5, 37, 47, 50, 67] that leverage DP to preserve instance-level DP in federated learning while [18, 68] preserve client-level DP in FL. Despite its effectiveness, DP's formal quantification of privacy leakage remains a topic for further refinement. To address the limitations of DP-FL, which relies on a trusted aggregator, LDP has been proposed as a more secure alternative. In LDP-FL, data is perturbed locally on each client before being sent to the aggregator, providing privacy protection even if the aggregator is compromised. Chamikara et al. [33] proposed a new LDP FL protocol, which is designed for industrial settings with untrusted entities. [1] presented practicable approaches to large-scale locally private model training that were previously impossible, showing theoretically and empirically that the proposed work can fit large-scale image classification and language models with little degradation in utility. However, these two works do not consider the composition of privacy budget with multiple rounds. Wang et al. [59] is also based on LDP, but it needs assumption on the training data (e.g., the topic distribution of users) for deriving the perturbation probabilities (otherwise, the LDP would be violated). Sav et al. [43] proposed FedPAQ, a communication-efficient FL method using periodic averaging and quantization. Collectively, these studies showcase the diverse applications and progress in the field.

8 CONCLUSION

Applying LDP technique to federated learning is a good solution to protect the model weights from each client. However, LDP technique may sacrifice the utility of federated learning while preserving privacy. In this paper, we first expose the privacy leakage in the work [48] (the best performing LDP in FL work to the best of our knowledge) and argue for a more effective LDP mechanism balancing optimal performance with stringent privacy. We introduce the SRR-FL framework, enhancing LDP in FL and demonstrating better performance and robust privacy compared to existing methods. Our findings highlight SRR-FL's significant advantages in both utility and privacy protection.

ACKNOWLEDGMENTS

The authors sincerely thank all the reviewers for their constructive comments. This work is supported in part by the National Science Foundation (NSF) under Grants No. CNS-2308730, CNS-2302689, CNS-2319277, and CMMI-2326341.

REFERENCES

[1] Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. 2018. Protection against reconstruction and its applications in private federated learning. *arXiv preprint arXiv:1812.00984* (2018).
[2] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. *Proceedings of machine learning and systems* 1 (2019), 374–388.
[3] Mahawaga Arachchige Pathum Chamikara, Dongxi Liu, Seyit Camtepe, Surya Nepal, Marthie Grobler, Peter Bertok, and Ibrahim Khalil. 2022. Local Differential Privacy for Federated Learning. In *ESORICS*, Vol. 13554. Springer, 195–216.
[4] Cangxiong Chen and Neill DF Campbell. 2021. Understanding training-data leakage from gradients in neural networks for image classification. *arXiv preprint arXiv:2111.10178* (2021).
[5] Anda Cheng, Peisong Wang, Xi Sheryl Zhang, and Jian Cheng. 2022. Differentially private federated learning with local regularization and sparsification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10122–10131.
[6] Olivia Choudhury, Aris Gkoulalas-Divanis, Theodoros Salonidis, Issa Sylla, Yoonyoung Park, Grace Hsu, and Amar Das. 2019. Differential privacy-enabled federated learning for sensitive health data. *arXiv:1910.02578* (2019).
[7] Li Deng. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142.
[8] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting telemetry data privately. *Advances in Neural Information Processing Systems* 30 (2017).
[9] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *Advances in Cryptology - EUROCRYPT 2006*, Serge Vaudenay (Ed.). 486–503.
[10] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.
[11] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. 9, 3–4 (2014).
[12] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. 2019. Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. (2019).
[13] Úlfar Erlingsson, Aleksandra Korolova, and Vasyl Pihur. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. *CoRR* abs/1407.6981 (2014). arXiv:1407.6981 http://arxiv.org/abs/1407.6981

[14] Vitaly Feldman, Audra McMillan, and Kunal Talwar. 2023. Stronger privacy amplification by shuffling for rényi and approximate differential privacy. In *SODA*. SIAM, 4966–4981.

[15] Shuya Feng, Meisam Mohammady, Han Wang, Xiaochen Li, Zhan Qin, and Yuan Hong. 2024. DPI: Ensuring Strict Differential Privacy for Infinite Data Streaming. In *2024 IEEE Symposium on Security and Privacy (SP)*.

[16] Jiahui Geng, Yongli Mou, Feifei Li, Qing Li, Oya Beyan, Stefan Decker, and Chunming Rong. 2021. Towards general deep leakage in federated learning. *arXiv preprint arXiv:2110.09074* (2021).

[17] Quan Geng, Peter Kairouz, Sewoong Oh, and Pramod Viswanath. 2015. The Staircase Mechanism in Differential Privacy. *IEEE Journal of Selected Topics in Signal Processing* 9, 7 (2015), 1176–1184.

[18] Robin C. Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially Private Federated Learning: A Client Level Perspective. *CoRR* abs/1712.07557 (2017). arXiv:1712.07557 http://arxiv.org/abs/1712.07557

[19] Antonious Girgis, Deepesh Data, Suhas Diggavi, Peter Kairouz, and Ananda Theertha Suresh. 2021. Shuffled Model of Differential Privacy in Federated Learning. In *AISTATS*.

[20] Antonious M Girgis, Deepesh Data, Suhas Diggavi, Ananda Theertha Suresh, and Peter Kairouz. 2021. On the renyi differential privacy of the shuffle model. In *CCS*. 2321–2341.

[21] Haimei Gong, Liangjun Jiang, Xiaoyang Liu, Yuanqi Wang, Lei Wang, and Ke Zhang. 2022. Recover User's Private Training Image Data by Gradient in Federated Learning. *Sensors* 22, 19 (2022), 7157.

[22] Yuan Hong, Jaideep Vaidya, Haibing Lu, Panagiotis Karras, and Sanjay Goel. 2015. Collaborative Search Log Sanitization: Toward Differential Privacy and Boosted Utility. *IEEE Trans. Dependable Secur. Comput.* 12, 5 (2015), 504–518.

[23] Rui Hu, Yuanxiong Guo, Hongning Li, Qingqi Pei, and Yanmin Gong. 2020. Personalized federated learning with differential privacy. *IEEE Internet of Things Journal* 7, 10 (2020), 9530–9539.

[24] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2011. What can we learn privately? *SIAM J. Comput.* 40, 3 (2011), 793–826.

[25] Muah Kim, Onur Günlü, and Rafael F Schaefer. 2021. Federated learning with local differential privacy: Trade-offs between privacy, utility, and communication. In *ICASSP*. IEEE, 2650–2654.

[26] Paul C Kocher. 1996. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO'96*. Springer, 104–113.

[27] Jakub Konecnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* (2016).

[28] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[29] Yiwei Li, Tsung-Hui Chang, and Chong-Yung Chi. 2020. Secure federated averaging algorithm with differential privacy. In *MLSP*. IEEE, 1–6.

[30] Zhuotao Lian, Weizheng Wang, and Chunhua Su. 2021. COFEL: Communication-efficient and optimized federated learning with local differential privacy. In *ICC*.

[31] Bingyu Liu, Shangyu Xie, Han Wang, Yuan Hong, Xuegang Ban, and Meisam Mohammady. 2021. VTDP: Privately Sanitizing Fine-Grained Vehicle Trajectory Data With Boosted Utility. *IEEE Trans. Dependable Secur. Comput.* 18, 6 (2021), 2643–2657.

[32] Ruixuan Liu, Yang Cao, Hong Chen, Ruoyang Guo, and Masatoshi Yoshikawa. 2021. Flame: Differentially private federated learning in the shuffle model. In *AAAI*, Vol. 35. 8688–8696.

[33] Pathum Chamikara Mahawaga Arachchige, Dongxi Liu, Seyit Camtepe, Surya Nepal, Marthie Grobler, Peter Bertok, and Ibrahim Khalil. 2022. Local differential privacy for federated learning. In *ESORICS*. Springer, 195–216.

[34] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*. 1273–1282.

[35] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. 2016. Federated Learning of Deep Networks using Model Averaging. *CoRR* abs/1602.05629 (2016). arXiv:1602.05629 http://arxiv.org/abs/1602.05629

[36] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017. Learning Differentially Private Language Models Without Losing Accuracy. *CoRR* abs/1710.06963 (2017). arXiv:1710.06963 http://arxiv.org/abs/1710.06963

[37] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *ICLR*.

[38] Meisam Mohammady, Shangyu Xie, Yuan Hong, Mengyuan Zhang, Lingyu Wang, Makan Pourzandi, and Mourad Debbabi. 2020. R2DP: A Universal and Automated Approach to Optimizing the Randomization Mechanisms of Differential Privacy for Utility Metrics with No Known Optimal Distributions. In *CCS*.

[39] Seung Ho Na, Hyeong Gwon Hong, Junmo Kim, and Seungwon Shin. 2022. Closing the Loophole: Rethinking Reconstruction Attacks in Federated Learning from a Privacy Standpoint. In *ACSAC*.

[40] 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 *IEEE SP*. 739–753.

[41] Sayedeh Leila Noorbakhsh, Binghui Zhang, Yuan Hong, and Binghui Wang. 2024. Inf2Guard: An Information-Theoretic Framework for Learning Privacy-Preserving Representations against Inference Attacks. *USENIX Security* (2024).

[42] Mathias PM Parisot, Balazs Pejo, and Dayana Spagnuelo. 2021. Property inference attacks on convolutional neural networks: Influence and implications of target model's complexity. *arXiv preprint arXiv:2104.13061* (2021).

[43] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. 2020. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *AISTATS*. PMLR, 2021–2031.

[44] Sina Sajadmanesh and Daniel Gatica-Perez. 2021. Locally Private Graph Neural Networks. In *CCS*. 2130–2145.

[45] Ahmed Salem, Giovanni Cherubin, David Evans, Boris Köpf, Andrew Paverd, Anshuman Suri, Shruti Tople, and Santiago Zanella-Béguelin. 2023. SoK: Let the Privacy Games Begin! A Unified Treatment of Data Inference Privacy in Machine Learning. In *2023 IEEE Symposium on Security and Privacy (SP)*. 327–345.

[46] Reza Shokri, Marco Stronati, and Vitaly Shmatikov. 2016. Membership Inference Attacks against Machine Learning Models. *CoRR* abs/1610.05820 (2016). arXiv:1610.05820 http://arxiv.org/abs/1610.05820

[47] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. 2017. Federated multi-task learning. *Advances in neural information processing systems* 30 (2017).

[48] Lichao Sun, Jianwei Qian, and Xun Chen. 2021. LDP-FL: Practical Private Aggregation in Federated Learning with Local Differential Privacy. In *IJCAI*.

[49] Aleksei Triastcyn and Boi Faltings. 2019. Federated learning with bayesian differential privacy. In *IEEE Big Data*. 2587–2596.

[50] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. 2020. LDP-Fed: Federated learning with local differential privacy. In *Proceedings of the third ACM international workshop on edge systems, analytics and networking*. 61–66.

[51] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. 2019. Demystifying membership inference attacks in machine learning as a service. *IEEE transactions on services computing* 14, 6 (2019), 2073–2089.

[52] Muhammad Habib ur Rehman, Ahmed Mukhtar Dirir, Khaled Salah, Ernesto Damiani, and Davor Svetinovic. 2021. TrustFed: A framework for fair and trustworthy cross-device federated learning in IIoT. *IEEE Transactions on Industrial Informatics* 17, 12 (2021), 8485–8494.

[53] Jaideep Vaidya, Basit Shafiq, Anirban Basu, and Yuan Hong. 2013. Differentially Private Naive Bayes Classification. In *WI*. 571–576.

[54] Huanyu Wang and Elena Dubrova. 2021. Federated learning in side-channel analysis. In *ICISC*. Springer, 257–272.

[55] Han Wang, Hanbin Hong, Li Xiong, Zhan Qin, and Yuan Hong. 2022. L-SRR: Local Differential Privacy for Location-Based Services with Staircase Randomized Response. In *CCS'22*. 2809–2823. https://doi.org/10.1145/3548606.3560636

[56] Han Wang, Jayashree Sharma, Shuya Feng, Kai Shu, and Yuan Hong. 2022. A Model-Agnostic Approach to Differentially Private Topic Mining. In *KDD*.

[57] Han Wang, Shangyu Xie, and Yuan Hong. 2020. VideoDP: A Flexible Platform for Video Analytics with Differential Privacy. *Proc. Priv. Enhancing Technol.* 2020, 4 (2020), 277–296.

[58] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. 2019. Collecting and Analyzing Multidimensional Data with Local Differential Privacy. In *ICDE*. IEEE, 638–649.

[59] Yansheng Wang, Yongxin Tong, and Dingyuan Shi. 2020. Federated latent dirichlet allocation: A local differential privacy based framework. In *AAAI*.

[60] 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.

[61] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *TIFS* (2020).

[62] Jiahao Xie, Chao Zhang, Zebang Shen, Weijie Liu, and Hui Qian. 2021. Efficient cross-device federated learning algorithms for minimax problems. *arXiv preprint arXiv:2105.14216* (2021).

[63] Shangyu Xie and Yuan Hong. 2021. Reconstruction Attack on Instance Encoding for Language Understanding. In *EMNLP*.

[64] Xiaoyun Xu, Jingzheng Wu, Mutian Yang, Tianyue Luo, Xu Duan, Weiheng Li, Yanjun Wu, and Bin Wu. [n. d.]. Information leakage by model weights on federated learning. In *PPMLP'20*.

[65] Jiancheng Yang, Rui Shi, and Bingbing Ni. 2021. MedMNIST Classification Decathlon: A Lightweight AutoML Benchmark for Medical Image Analysis. In *IEEE ISBI*. 191–195.

[66] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *TIST* 1, 2 (2019), 1–19.

[67] Yuxiang Yang, Junzhe Jia, Justin Zhan, Chaoyi Pang, Senzhang Li, Haibin Lu, and Yanmin Chen. 2020. Federated Learning with Differential Privacy: Algorithms and Performance. *TIFS* (2020).

[68] Xinwei Zhang, Xiangyi Chen, Mingyi Hong, Zhiwei Steven Wu, and Jinfeng Yi. 2022. Understanding clipping for federated learning: Convergence and client-level differential privacy. In *ICML*.

APPENDIX

Detailed Evaluation for the Inference Attack

The Setting of Attack. Extensive experiments were conducted to simulate the inference attack on [48] (detailed in Section 3.1). In the simulation of attack, we assume that the value w in the input domain is approximately the value to which all models should converge after every round and that the mean of values across all models would be very close to w. Then, by considering the factor that model weights may not always be equal to w for various reasons, a tolerance parameter is set to ensure the mean of w_i's across models is close to w and the model weight of each client varies from w. Then, n random numbers were generated around w with a standard deviation of σ. These values represent the value of this particular weight that the n models converged to across the clients. Then, the LDP perturbation algorithm is applied to all these weights independently. The new vector of values is the noisy values arriving at the server. From here, the server would conduct the attack steps as discussed in Section 3.1.

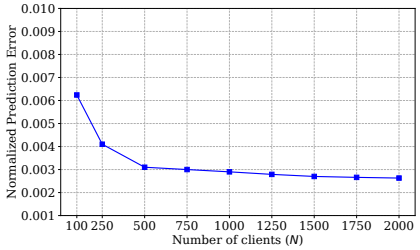
Evaluation Metric. Recall from Section 3.1, a smaller difference between $w^i_{\text{predicted\_1}}$ and $w^i_{\text{predicted\_2}}$ represents higher confidence for the server in the current round. The reason for this can be intuitively understood. In the ideal scenario, the difference would be zero, which implies that the randomly generated and split distribution of sample_1 and sample_2 are the same. This reassures the current attack on the server side is very strong. Consequently, the confidence score CS_i for the i'th round is introduced to quantify the deviation between the two sample predictions, delineated as follows:

$$\text{CS}_i = \frac{1}{0.01 + (|w^i_{\text{predicted\_2}} - w^i_{\text{predicted\_1}}|)/(2 \cdot r)} \tag{10}$$

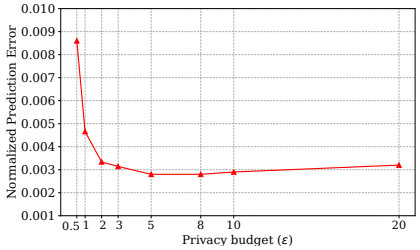
where the difference is divided by $2 \cdot r$ to normalize the difference with respect to the domain size. We can see that this metric always gives us a value between 0 and 100, indicating the confidence of the server in the attack prediction.

Experimental Results. In our experiments, we investigate the attack performance by varying the client number, privacy budget and the variance of model weights among all clients. As shown in Figure 7a, with around 750 clients, we were able to track down the value of the original non-noisy w_{mean} to be inside a range comprising 0.3% of the original domain. It can be seen that by increasing n, we are able to better track down the value of the original weight. The effect of privacy budget ϵ on the success of our attacks is minimal. Even with a privacy budget of $\epsilon = 1$, our attack was able to approximate the original weight back with an error in prediction (adjusted to domain size) of 0.46%, which is very successful. The performance of our attack deteriorates with reducing ϵ from $\epsilon = 1$, but the attack performance is still good even with very small privacy budget. Figure 7c shows that, with varying weight values among clients, our attack can accurately estimate the global non-private weight mean against the LDP scheme in [48].

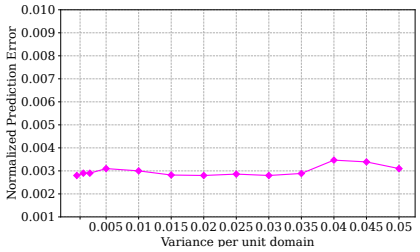
Since we simulate the weight value of clients in our attack, it is important to analyze how the variance of weights affects the attack performance. As demonstrated in Figure 8, besides the prediction



(a) Effect of N on the attack



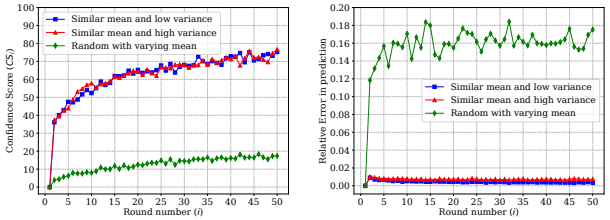
(b) Effect of ϵ on the attack



(c) Effect of variance per unit on the attack

Figure 7: Results for the effect of different parameters on the outcome of the attack.

error, the Confidence Score (CS_i) is also used to assess attack outcomes across different weight distributions. A high and increasing CS_i correlates with the similar mean of weights over rounds, enhancing the attack performance. This finding demonstrates our attack can work well in our proposed cases of federated learning.



(a) Effect on Score **(b) Effect on error**

Figure 8: Effect of the distribution of weight values across clients on the confidence score (CS_i) computed on the server side and the relative error in prediction.