DP-BREM: Differentially-Private and Byzantine-Robust Federated Learning with Client Momentum

Xiaolan Gu
University of Arizona
xiaolang@arizona.edu

Ming Li
University of Arizona
lim@arizona.edu

Li Xiong
Emory University
lxiong@emory.edu

Abstract

Federated Learning (FL) allows multiple participating clients to train machine learning models collaboratively while keeping their datasets local and only exchanging the gradient or model updates with a coordinating server. Existing FL protocols are vulnerable to attacks that aim to compromise data privacy and/or model robustness. Recently proposed defenses focused on ensuring either privacy or robustness, but not both. In this paper, we focus on simultaneously achieving differential privacy (DP) and Byzantine robustness for cross-silo FL, based on the idea of learning from history. The robustness is achieved via client momentum, which averages the updates of each client over time, thus reducing the variance of the honest clients and exposing the small malicious perturbations of Byzantine clients that are undetectable in a single round but accumulate over time. In our initial solution DP-BREM, DP is achieved by adding noise to the aggregated momentum, and we account for the privacy cost from the momentum, which is different from the conventional DP-SGD that accounts for the privacy cost from the gradient. Since DP-BREM assumes a trusted server (who can obtain clients' local models or updates), we further develop the final solution called DP-BREM$^+$, which achieves the same DP and robustness properties as DP-BREM without a trusted server by utilizing secure aggregation techniques, where DP noise is securely and jointly generated by the clients. Both theoretical analysis and experimental results demonstrate that our proposed protocols achieve better privacy-utility tradeoff and stronger Byzantine robustness than several baseline methods, under different DP budgets and attack settings.

1 Introduction

Federated learning (FL) [25] is an emerging paradigm that enables multiple clients to collaboratively learn models without explicitly sharing their data. The clients upload their local model updates to a coordinating server, which then shares the global average with the clients in an iterative process. This offers a promising solution to mitigate the potential privacy leakage of sensitive information about individuals (since the data stays local with each client), such as typing history, shopping transactions, geographical locations, and medical records. However, recent works have demonstrated that FL may not always provide sufficient *privacy* and *robustness* guarantees. In terms of privacy leakage, exchanging the model updates throughout the training process can still reveal sensitive information [4, 27] and cause deep leakage such as pixel-wise accurate image recovery [41, 44], either to a third-party (including other participating clients) or the central server. In terms of robustness, the decentralization design of FL systems opens up the training process to be manipulated by malicious clients, aiming to either prevent the convergence of the global model (a.k.a. Byzantine attacks) [3, 14, 38], or implant a backdoor trigger into the global model to cause targeted misclassification (a.k.a. backdoor attacks) [2, 37].

To mitigate the privacy leakage in FL, Differential Privacy (DP) [12, 13] has been adopted as a rigorous privacy notion. Existing frameworks [16, 24, 26] applied DP in FL to provide *client-level* privacy under the assumption of a trusted server: whether a client has participated in the training process cannot be inferred by a third party from the released global model. Other works in FL [24, 35, 39, 42] focused on *record-level* privacy: whether a data record at a client has participated during training cannot be inferred by the server or other adversaries that have access to the model updates or the global model. Record-level privacy is more relevant in cross-silo (as versus cross-device) FL scenarios, such as multiple hospitals collaboratively learn a prediction model for COVID-19, in which case what needs to be protected is the privacy of each patient (corresponding to each record in a hospital's dataset). In this paper, we focus on cross-silo FL with *record-level* DP, where each client possesses a set of raw records, and each record corresponds to an individual's private data.

To defend against Byzantine attacks, robust FL protocols are proposed to ensure that the training procedure is robust to a fraction of potentially malicious clients. This problem has received significant attention from the commu-

nity. Most existing approaches replace the averaging step at the server with a robust aggregation rule, such as the median [5, 9, 28, 40]. However, recent state-of-the-art attacks [3, 33, 38] have demonstrated the failure of the above robust aggregators. Furthermore, a recent work [20] shows that there exist realistic scenarios where these robust aggregators fail to converge, even if there are no Byzantine attackers and the data distribution is identical (i.i.d.) across the clients, and proposed a new solution called Learning From History (LFH) to address this issue. LFH achieves robustness via client momentum with the motivation of averaging the updates of each client over time, thus reducing the variance of the honest clients and exposing the small malicious perturbations of Byzantine clients that are undetectable in a single round but accumulate over time.

In this paper, we focus on achieving record-level DP and Byzantine robustness simultaneously in cross-silo FL. Existing FL protocols with DP-SGD [1] do not achieve the robustness property intrinsically. Directly implementing an existing robust aggregator over the privatized client gradients will lead to poor utility because these aggregators (such as median [5, 28, 40]) usually have large sensitivity and require large DP noise, leading to poor utility. It is desirable to achieve DP guarantees based on average-based aggregators. Although LFH [20] is an average-based Byzantine-robust FL protocol, it aggregates client momentum instead of gradient, thus it is non-trivial to achieve DP on top of LFH. We show that a direct combination of LFH with DP-SGD momentum has several limitations, leading to both poor utility and robustness. Therefore, we aim to address these limitations in our solution.

To achieve an enhanced privacy-utility tradeoff, we start our problem from an assumption that the server is trusted and developed a **D**ifferentially-**P**rivate and **B**yzantine-**R**obust f**E**derated learning algorithm with client **M**omentum (**DP-BREM**), which essentially is a DP version of the Byzantine-robust method LFH [20]. Instead of adding DP noise to the gradient and then aggregating momentum as post-processing, we add DP noise to the aggregated momentum with carefully computed sensitivity to account for the privacy cost. Since the noise is added to the final aggregate (instead of intermediate local gradient), our basic solution DP-BREM maintains the non-private LFH's robustness as much as possible, which we show both theoretically (via convergence analysis) and empirically (via experimental results). Then, we relax our trust assumption to a malicious server (for privacy only) and develop our final solution DP-BREM$^+$. It utilizes secure multiparty computation (MPC) techniques, including secure aggregation and secure noise generation, to achieve the same DP and robustness guarantees as in DP-BREM. In Table 1, we compare DP-BREM and DP-BREM$^+$ with existing approaches (or the variants) that achieve both DP and Byzantine robustness (DDP-RP [36] and DP-RSA [43] are described in Sec. 7). These approaches will be evaluated and compared in experiments. Our main contributions are:

1) We propose DP-BREM, a novel differentially private and Byzantine-robust FL protocol that adds DP noise to aggregated client momentum with computed sensitivity. Our privacy analysis (Theorem 1) accounts for momentum, differing from conventional DP-SGD which accounts for gradient. Our convergence analysis (Theorem 3) shows minimal convergence rate sacrifice for DP compared to the baseline.

2) Considering that DP-BREM is developed under the assumption of a trusted server, we propose the final solution called DP-BREM$^+$ (Section 5), which achieves the same privacy and robustness properties as DP-BREM, even under a *malicious* server (for privacy only), using secure multiparty computation techniques. DP-BREM$^+$ is built based on the framework of secure aggregation with verifiable inputs (SAVI) [30], but extends it to guarantee the integrity of DP noise via a novel secure distributed noise generation protocol. Our extended SAVI protocol is general enough to be applied to other DP and robust FL protocols that are average-based.

3) We demonstrate our protocols' effectiveness through extensive experiments on MNIST, CIFAR-10, and FEMNIST datasets (Section 6), showing improved utility with the same record-level DP guarantees and strong robustness against Byzantine clients under state-of-the-art attacks, compared to baseline methods.

2 Preliminaries

2.1 Differential Privacy (DP)

Differential Privacy (DP) is a rigorous mathematical framework for the release of information derived from private data. Applied to machine learning, a differentially private training mechanism allows the public release of model parameters with a strong privacy guarantee: adversaries are limited in what they can learn about the original training data based on analyzing the parameters, even when they have access to arbitrary side information. The formal definition is as follows:

Definition 1 ((ε,δ)-DP [12, 13])**.** *For $\varepsilon \in [0,\infty)$ and $\delta \in [0,1)$, a randomized mechanism $\mathcal{M}: \mathcal{D} \to \mathcal{R}$ with a domain $\mathcal{D}$ (e.g., possible training datasets) and range $\mathcal{R}$ (e.g., all possible trained models) satisfies (ε,δ)-Differential Privacy (DP) if for any two neighboring datasets $D, D' \in \mathcal{D}$ that differ in only one record and for any subset of outputs $S \subseteq \mathcal{R}$, it holds that*

$$\mathbb{P}[\mathcal{M}(D) \in S] \leqslant e^{\varepsilon} \cdot \mathbb{P}[\mathcal{M}(D') \in S] + \delta$$

where ε and δ are privacy parameters (or privacy budget), and a smaller ε and δ indicate a more private mechanism.

Gaussian Mechanism. A common paradigm for approximating a deterministic real-valued function $f: \mathcal{D} \to \mathbb{R}$ with a differentially-private mechanism is via additive noise calibrated to f's sensitivity s_f, which is defined as the maximum of the absolute distance $|f(D) - f(D')|$. The Gaussian Mechanism is defined by $\mathcal{M}(D) = f(D) + \mathcal{N}(0, s_f^2 \cdot \sigma^2)$, where

$\mathcal{N}(0, s_f^2 \cdot \sigma^2)$ is noise drawn from a Gaussian distribution. It was shown that $\mathcal{M}$ satisfies (ε, δ)-DP if $\delta \geqslant \frac{4}{5} e^{-(\sigma\varepsilon)^2/2}$ and $\varepsilon < 1$ [13]. Note that we use an advanced privacy analysis tool proposed in [11], which works for all $\varepsilon > 0$.

DP-SGD Algorithm. The most well-known differentially-private algorithm in machine learning is DP-SGD [1], which introduces two modifications to the vanilla stochastic gradient descent (SGD). First, a *clipping step* is applied to the gradient so that the gradient is in effect bounded for a finite sensitivity. The second modification is *Gaussian noise augmentation* on the summation of clipped gradients, which is equivalent to applying the Gaussian mechanism to the updated iterates. The privacy accountant of DP-SGD is shown in the Appendix of our full-version paper [17].

2.2 Federated Learning (FL) with DP

Federated Learning (FL) [19, 25] is a collaborative learning setting to train machine learning models. We consider the horizontal cross-silo FL setting, which involves multiple clients, each holding their own private dataset of the same set of features, and a central server that implements the aggregation. Unlike the traditional centralized approach, data is not stored at a central server; instead, clients train models locally and exchange updated parameters with the server, which aggregates the received local model parameters and sends them to the clients. Based on the participating clients and scale, federated learning can be classified into two types: *cross-device* FL where clients are typically mobile devices and the client number can reach up to a scale of millions; *cross-silo* FL (our focus) where clients are organizations or companies and the client number is relatively small (e.g., within hundreds).

FL with DP. In FL, the *neighboring datasets* D and D' in Definition 1 can be defined at two distinct levels: *record-level* and *client-level*. In cross-device FL, each device usually stores one individual's data, and then the whole device's data should be protected. It corresponds to client-level DP, where D' is obtained by adding or removing one client/device's whole training dataset from D. In cross-silo FL, each record corresponds to one individual's data, then record-level DP should be provided, where D' is obtained by adding or removing a single training record/example from D. Since we consider cross-silo FL, achieving *record-level* DP is our privacy goal.

2.3 Byzantine Attacks and Defenses

In a Byzantine attack, the adversary aims to destroy the convergence of the model. Due to the decentralization design, FL systems are vulnerable to Byzantine clients, who may not follow the protocol and can send arbitrary updates to the server. Also, they may have complete knowledge of the system and can collude with each other. Most state-of-the-art defense mechanisms [5, 9, 28, 40] play with median statistics of client gradients. However, recent attacks [3, 38] have empirically demonstrated the failure of the above robust aggregations.

LFH: Non-private Byzantine-Robust Defense. Recently, Karimireddy et al. [20] showed that most state-of-the-art robust aggregators require strong assumptions and may not converge even in the complete absence of Byzantine attackers. Then, they proposed a new Byzantine-robust scheme called "learning from history" (LFH) that essentially utilizes two simple strategies: *client momentum* (during local update) and *centered clipping* (during server aggregation). In each iteration t, client C_i receives the global model parameter $\boldsymbol{\theta}_{t-1}$ from the server, and computes the local gradient of the random dataset batch $\mathcal{D}_{i,t} \subseteq \mathcal{D}_i$ by

$$\boldsymbol{g}_{t,i} = \frac{1}{|\mathcal{D}_{i,t}|} \sum\nolimits_{\boldsymbol{x} \in \mathcal{D}_{i,t}} \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{x}, \boldsymbol{\theta}_{t-1}) \tag{1}$$

where $\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{x}, \boldsymbol{\theta}_{t-1})$ is the per-record gradient w.r.t. the loss function $\ell(\cdot)$. The client momentum can be computed via

$$\boldsymbol{m}_{t,i} = (1-\beta)\boldsymbol{g}_{t,i} + \beta \boldsymbol{m}_{t-1,i} \tag{2}$$

where $\beta \in [0, 1)$. After receiving $\boldsymbol{m}_{t,i}$ from all clients, the server implements aggregation with centered clipping via

$$\boldsymbol{m}_t = \boldsymbol{m}_{t-1} + \frac{1}{n} \sum\nolimits_{i=1}^{n} \mathsf{Clip}_C(\boldsymbol{m}_{t,i} - \boldsymbol{m}_{t-1}) \tag{3}$$

where $\mathsf{Clip}_C(\cdot)$ with scalar $C > 0$ is the clipping function:

$$\mathsf{Clip}_C(\boldsymbol{x}) := \boldsymbol{x} \cdot \min\{1, C/\|\boldsymbol{x}\|\} \tag{4}$$

and $\|\boldsymbol{x}\|$ is the L2-norm of any vector $\boldsymbol{x}$. The clipping operation $\mathsf{Clip}_C(\boldsymbol{m}_{t,i} - \boldsymbol{m}_{t-1})$ essentially bounds the distance between client's local momentum $\boldsymbol{m}_{t,i}$ and the previous aggregated momentum $\boldsymbol{m}_{t-1}$, thus restricts the impact from Byzantine clients. Then, the global model $\boldsymbol{\theta}_t$ can be updated by $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta_t \boldsymbol{m}_t$ with learning rate η_t. The convergence rate under Byzantine attacks is shown by the following lemma.

Lemma 1 (Convergence Rate of LFH [20])**.** *With some parameter tuning, the convergence rate of the Byzantine-robust algorithm LFH is asymptotically (ignoring constants and higher order terms) of the order*

$$\frac{1}{T} \sum\nolimits_{t=1}^{T} \mathbb{E}\|\nabla \ell(\boldsymbol{\theta}_{t-1})\|^2 \lesssim \sqrt{\frac{\rho^2}{T} \frac{1+|\mathcal{B}|}{n}} \tag{5}$$

where $\ell(\cdot)$ is the loss function, T is the total number of training iterations, $|\mathcal{B}|$ is the number of Byzantine clients, n is the number of all clients, and ρ is a parameter that quantifies the variance of honest clients' stochastic gradients:

$$\mathbb{E}\|\boldsymbol{g}_{t,i} - \mathbb{E}[\boldsymbol{g}_{t,i}]\|^2 \leqslant \rho^2 \tag{6}$$

Interpretation of Lemma 1. When there are no Byzantine clients, LFH recovers the optimal rate of $\frac{\rho}{\sqrt{nT}}$. In the presence of a $|\mathcal{B}|/n$ fraction of Byzantine clients, the rate has an additional term $\rho\sqrt{\frac{|\mathcal{B}|/n}{T}}$, which depends on the fraction $|\mathcal{B}|/n$ but does not improve with increasing clients.

3 Problem Statement and Motivation

3.1 Problem Statement

System Model. Our system model follows the general setting of Fed-SGD [25]. There are multiple parties in the FL system: one aggregation server and n participating clients $\{\mathsf{C}_1, \cdots, \mathsf{C}_n\}$. The server holds a global model $\boldsymbol{\theta}_t \in \mathbb{R}^d$ and each client C_i, $i \in \{1, \cdots, n\}$ possesses a private training dataset $\mathcal{D}_i$. The server communicates with each client through a secure (private and authenticated) channel. During the iterative training process, the server broadcasts the global model in the current iteration to all clients and aggregates the received gradient/momentum from all clients (or a subset of clients) to update the global model until convergence. The final global model is returned after the training process as the output.

Threat Model. The considered adversary aims to perform a 1) privacy attack and/or 2) Byzantine attack with the following threat model, respectively.

1) Privacy Attack. Following the conventional FL setting, we assume the server has no access to the client's local training data, but may have an incentive to infer clients' private information. In our initial solution called DP-BREM, we assume a trusted server that can obtain clients' local models/updates. The adversary is a third party or the participating clients (can be any set of clients) who have access to the intermediate and final global models and may use them to infer the private data of an honest client C_i. Hence, the privacy goal is to ensure the global model (and its update) satisfies DP. In our final solution DP-BREM$^+$, in addition to third parties and clients, the adversary also includes the server that tries to infer additional information from the local updates (and may deviate from the protocol for privacy inference). Such a model is also adopted in previous work [30]. Following [30], we assume a minority of malicious clients who can deviate from the protocol arbitrarily.

2) Byzantine Attack. Recall that the goal of Byzantine attacks is to destroy the convergence of the global model (discussed in Section 2.3). We only consider malicious clients as the adversaries for Byzantine attacks because the server's primary goal is to train a robust model, thus no incentive to implement Byzantine attacks. These malicious clients (assumed to be a minority of all participating clients) can deviate from the protocol arbitrarily and have full control of both their local training data and their submission to the servers, but do not influence other honest clients.

Objectives. The goal of this paper is to achieve both record-level DP and Byzantine robustness at the same time. We aim to provide high utility (i.e., high accuracy of the global model) with the required DP guarantee under the existence of Byzantine attacks from malicious clients. Our ultimate privacy goal is to provide DP guarantees against an untrusted server and other clients, but we start by assuming a trusted server first in our initial solution.

3.2 Challenges and Baseline

Challenges. Replacing the average-based aggregator with median-based or complex robust aggregators increases DP sensitivity. Achieving both DP and Byzantine robustness with high utility is challenging because these methods result in significantly larger DP sensitivity than averaging, as illustrated in Example 1.

Example 1 (Sensitivity Computation: Average vs. Median)**.** *Consider a dataset with 5 samples: $\mathcal{D} = \{1,3,5,7,9\}$, and its neighboring dataset $\mathcal{D}'$ is obtained by changing one value in $\mathcal{D}$ with at most 1, such as $\mathcal{D}' = \{1,3,\mathbf{4},7,9\}$. Then, the sensitivity of average-query is $\max_{\mathcal{D},\mathcal{D}'} |\mathsf{avg}(\mathcal{D}) - \mathsf{avg}(\mathcal{D}')| = 1/5 = 0.2$. However, the sensitivity of median-query is $\max_{\mathcal{D},\mathcal{D}'} |\mathsf{median}(\mathcal{D}) - \mathsf{median}(\mathcal{D}')| = 1$. Moreover, when increasing the size of the dataset, the sensitivity of the average query will be reduced (and then less noise to be added), while the sensitivity of the median query is the same.*

DP-LFH: baseline via direct combination of LFH and DP-SGD. As shown in Section 2.3, the Byzantine-robust scheme LFH [20] utilizes an average-based aggregator, which can be regarded as a non-private robust solution to address the disadvantage of the median-based aggregator. A straightforward method to add DP protection on top of LFH is to combine it with the DP-SGD algorithm. However, LFH requires each client to compute the local momentum $\boldsymbol{m}_{t,i}$ for server aggregation, while DP-SGD aggregates gradients and accounts for the privacy cost via the composition of iterative gradient update. In LFH, since the gradient is computed only on the client-side, a straightforward solution to integrate DP is to use DP-SGD at each client to privatize the local gradient, and then compute the momentum from the privatized gradient (thus there is no additional privacy cost due to post-processing). Formally, client C_i computes

$$\boldsymbol{g}_{t,i} = \frac{1}{|\mathcal{D}_{i,t}|} \sum_{\boldsymbol{x} \in \mathcal{D}_{i,t}} \mathsf{Clip}_R(\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{x}, \boldsymbol{\theta}_{t-1})) + \mathcal{N}(0, R^2\sigma^2 \mathbf{I}_d), \quad (7)$$

where $\mathbf{I}_d$ is an identity matrix with size $d \times d$ (d is the model size, i.e., $\boldsymbol{\theta}_t \in \mathbb{R}^d$), the record-level clipping $\mathsf{Clip}_R(\cdot)$ restricts the sensitivity when adding/removing one record from the local dataset, and Gaussian noise $\mathcal{N}(0, R^2\sigma^2\mathbf{I}_d)$ introduces DP property on $\boldsymbol{g}_{t,i}$. Since DP is immune to post-processing, the remaining steps can follow the original LFH without additional privacy costs. This baseline solution DP-LFH achieves record-level DP against an untrusted server but has limitations, leading to poor privacy-utility tradeoff and robustness.

Limitation 1: large aggregated noise. Since each client locally adds DP noise, the overall noise after aggregation is larger than the case of the central setting under the same privacy budget ε since only the server adds noise in the central setting. Therefore, DP-LFH has a poor privacy-utility tradeoff.

Table 1: Comparison of FL approaches with DP and Byzantine-robustness

Approaches	Differential Privacy (DP) [§]				Byzantine Robustness
	Trust Assumption of Server	Noise Generator	Perturbation Mechanism	Standard Deviation of Noise in Aggregate	Mechanism
DP-FedSGD [26] with both record and client norm clippings	trusted	server	$\sum_i g_i + \mathcal{N}(0,\sigma^2)$	σ	client norm clipping
CM [40] with DP noise	trusted	server	$\text{median}(\{g_i\}_{i=1}^n) + \mathcal{N}(0,\sigma^2)$	σ	coordinate-wise median (CM)
DDP-RP [36] [◇]	honest-but-curious	clients (distributively)	$\sum_i [g_i + \mathcal{N}(0, \frac{\sigma^2}{\tau})]$	$\sqrt{\frac{n}{\tau}} \cdot \sigma$	element-wise range proof
DP-RSA [43]	untrusted	client	$\sum_i [\text{sign}(g_i) + \mathcal{N}(0,\sigma^2)]$	$\sqrt{n} \cdot \sigma$	aggregation of sign-SGD
DP-LFH (baseline in Sec. 3.2)	untrusted	client	$\sum_i [m_i + \mathcal{N}(0,\sigma^2)]$	$\sqrt{n} \cdot \sigma$	LFH [20]: client momentum and centered clipping
DP-BREM (our initial solution)	trusted	server	$\sum_i m_i + \mathcal{N}(0,\sigma^2)$	σ	
DP-BREM$^+$ (our final solution) [†]	untrusted	clients (jointly)			

[§] We demonstrate the privacy-utility tradeoff by comparing the standard deviation of DP noise on the aggregation, with smaller values indicating less negative impact on utility. Note that different approaches use different aggregation strategies, where g_i is the local gradient and m_i is the local momentum.

[◇] DDP-RP assumes an honest-but-curious server and ensures distributed DP (DDP) with secure aggregation. Clients add partial noise with a smaller standard deviation based on the number of honest clients, τ, resulting in a better privacy-utility tradeoff than local DP (LDP).

[†] DP-BREM$^+$ matches DP-BREM's DP and robustness guarantees with a different server trust assumption. It achieves central DP without a trusted server, as clients securely generate and add noise using the proposed noise generation and secure aggregation protocols.

Limitation 2: large impact on Byzantine robustness. Since the DP noise is added locally to each client's gradient before momentum-based clipping, it leads to a negative impact on Byzantine robustness: the noisy client momentum $\boldsymbol{m}_{t,i}$ has larger variance than the noise-free one, which leads to larger bias and variance on the clipping step $\text{Clip}_C(\boldsymbol{m}_{t,i} - \boldsymbol{m}_{t-1})$. Furthermore, this impact will be enlarged when there are more Byzantine clients, which is explained as follows. Since the parameter ρ^2 defined in (6) quantifies the variance of client's gradient, and the DP noise is added to the local gradient in (7), the parameter ρ of the convergence rate shown in (5) is replaced by $\rho + \sqrt{d}\sigma$ (ignoring constants) for DP-LFH, i.e., the convergence rate of DP-LFH is asymptotic of the order

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\|\nabla \ell(\boldsymbol{\theta}_{t-1})\|^2 \lesssim \sqrt{\frac{(\rho+\sqrt{d}\sigma)^2}{T}\frac{1+|\mathcal{B}|}{n}} \quad (8)$$

Therefore, either a large d (i.e., large model) or a large σ (i.e., small privacy budget ε) will enlarge the impact from Byzantine clients due to the order $O(\sqrt{d\sigma^2|\mathcal{B}|})$ of convergence rate. We note that Guerraoui et al.'s work [18] also shares a similar insight: they show that DP with local noise and Byzantine robustness are incompatible, especially when the dimension of model parameters d is large.

Limitation 3: no privacy amplification from client-level sampling due to momentum. According to the recursive representation $\boldsymbol{m}_{t,i} = (1-\beta)\boldsymbol{g}_{t,i} + \beta \boldsymbol{m}_{t-1,i}$, client C_i's momentum in t-th iteration $\boldsymbol{m}_{t,i}$ is essentially a weighted summation of all previous privatized client gradients:

$$\boldsymbol{m}_{t,i} = (1-\beta)(\boldsymbol{g}_{t,i} + \beta \boldsymbol{g}_{t-1,i} + \cdots + \beta^{t-2}\boldsymbol{g}_{2,i}) + \beta^{t-1}\boldsymbol{g}_{1,i} \quad (9)$$

where $\boldsymbol{g}_{1,i}, \boldsymbol{g}_{2,i}, \cdots, \boldsymbol{g}_{t,i}$ are already privatized via local noise. Assume the server samples a subset of clients for aggregation in each iteration. If client C_i's momentum $\boldsymbol{m}_{t,i}$ is not selected in the t-th iteration, the aggregate is independent of $\boldsymbol{g}_{t,i}$. However, in a later iteration (i.e., $\tau > t$), if client C_i's momentum $\boldsymbol{m}_{\tau,i}$ is included, it depends on $\boldsymbol{g}_{t,i}$ according to (9). Thus, we must account for the privacy cost of $\boldsymbol{g}_{t,i}$ in all iterations. Sampling clients offers no privacy amplification, resulting in high privacy costs or low utility.

4 DP-BREM

To address DP-LFH's limitations, we propose DP-BREM, a differentially-private LFH variant assuming a trusted server that generates DP noise. DP-BREM maintains robustness of LFH and uses a different privacy accountant (Theorem 1) than DP-SGD. We also provide convergence analysis (Theorem 3) showing minimal deviation from LFH. We further relax the server trust assumption in DP-BREM$^+$ (Section 5) by using secure multiparty computation for secure aggregation and joint noise generation, achieving the same DP and robustness guarantees without a trusted server.

4.1 Algorithm Design

The illustration of our design is shown in Figure 1, and the algorithm is shown in Algorithm 1, where all clients need to implement local updates (in Line-3), but only a subset of their momentum vectors are aggregated by the server (in Line-4). The details of client updates and server aggregation are described below.

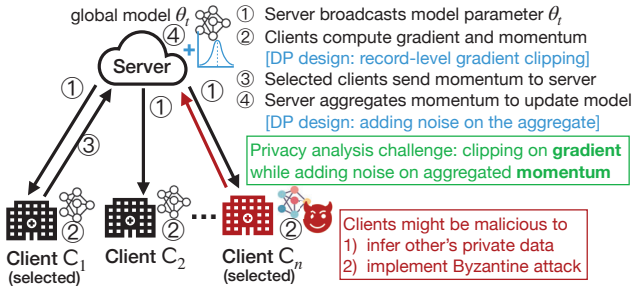

Figure 1: Illustration of our DP-BREM algorithm.

Client Updates. The client C_i first samples a random batch $\mathcal{D}_{i,t}$ from the local dataset $\mathcal{D}_i$ with sampling rate p_i, clips the per-record gradient $\nabla_{\boldsymbol{\theta}}\ell(\boldsymbol{x},\boldsymbol{\theta}_{t-1})$ by R and multiplies the sum by a constant factor $\frac{1}{p_i|\mathcal{D}_i|}$ to get the averaged gradient

$$\bar{\boldsymbol{g}}_{t,i} = \frac{1}{p_i|\mathcal{D}_i|}\sum\nolimits_{\boldsymbol{x}\in\mathcal{D}_{i,t}} \mathsf{Clip}_R(\nabla_{\boldsymbol{\theta}}\ell(\boldsymbol{x},\boldsymbol{\theta}_{t-1})) \tag{10}$$

where $\mathsf{Clip}_R(\cdot)$ is the clipping function defined in (4), but is used here to bound the sensitivity for DP (refer to DP-SGD discussed in Section 2.1). $\mathcal{D}_{i,t}$ represents a random subset obtained via subsampling from client C_i's dataset. This subsampling is essential to apply privacy amplification, enabling the privacy accountant to derive a tight privacy budget ε. Note that the batch size $|\mathcal{D}_{i,t}|$ is random and $\mathbb{E}[|\mathcal{D}_{i,t}|] = p_i|\mathcal{D}_i|$. Then, the local momentum can be computed by

$$\tilde{\boldsymbol{m}}_{t,i} = \begin{cases} \bar{\boldsymbol{g}}_{t,i}, & \text{if } t=1 \\ (1-\beta)\bar{\boldsymbol{g}}_{t,i} + \beta\tilde{\boldsymbol{m}}_{t-1,i}, & \text{if } t>1 \end{cases} \tag{11}$$

where $\beta\in[0,1)$ is the momentum parameter.

Server Aggregation. The server implements centered clipping with clipping parameter $C>0$ to bound the difference between client momentum $\tilde{\boldsymbol{m}}_{t,i}$ and the previous noisy global momentum $\tilde{\boldsymbol{m}}_{t-1}$ for robustness. Then, it adds Gaussian noise with standard deviation $R\sigma$ (thus the variance is $R^2\sigma^2$) to the sum of clipped terms to get the noisy global momentum $\tilde{\boldsymbol{m}}_t$

$$\tilde{\boldsymbol{m}}_t = \tilde{\boldsymbol{m}}_{t-1} + \frac{1}{|I_t|}\left[\sum\nolimits_{i\in I_t}\mathsf{Clip}_C(\tilde{\boldsymbol{m}}_{t,i}-\tilde{\boldsymbol{m}}_{t-1}) + \mathcal{N}(0,R^2\sigma^2\mathbf{I}_d)\right] \tag{12}$$

where $\mathbf{I}_d$ is an identity matrix with size $d\times d$, and only the sampled clients in I_t (which is obtained in Line-2 of Algorithm 1 with sampling rate q) are aggregated in t-th iteration. Note that adding noise $\mathcal{N}(0,R^2\sigma^2\mathbf{I}_d)$ to the *summation* of clipped client momentum $\sum_{i\in I_t}\mathsf{Clip}_C(\tilde{\boldsymbol{m}}_{t,i}-\tilde{\boldsymbol{m}}_{t-1})$ is equivalent to adding noise $\frac{1}{|I_t|}\mathcal{N}(0,R^2\sigma^2\mathbf{I}_d)$ to the *average* result $\frac{1}{|I_t|}\sum_{i\in I_t}\mathsf{Clip}_C(\tilde{\boldsymbol{m}}_{t,i}-\tilde{\boldsymbol{m}}_{t-1})$. Then, the server updates the global model $\boldsymbol{\theta}_t$ with learning rate η_t

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta_t\tilde{\boldsymbol{m}}_t \tag{13}$$

Remark: clipping bounds and sampling rates. In our algorithm, we use two clipping bounds and two sampling rates.

Algorithm 1 DP-BREM

Input: Initialization $\boldsymbol{\theta}_0\in\mathbb{R}^d$, clipping bounds R and C, learning rate η_t of the global model, total number of iterations T, client-level sampling rate q, record-level sampling rate p_i.

1: **for** $t=1,\cdots,T$ **do**
2: The server broadcasts the previous model $\boldsymbol{\theta}_{t-1}$ to all clients $\{\mathsf{C}_i\}_{i=1}^n$ and selects a subset of client index $I_t\subseteq\{1,\cdots,n\}$, where each client is selected with probability q.
3: Each client C_i for $i\in\{1,\cdots,n\}$ implements the local updates via (10) and (11), while only selected clients need to send the local momentum $\boldsymbol{m}_{t,i}$ (for $i\in I_t$) to the server.
4: The server aggregates received clients' momentum (only for $i\in I_t$) with centered clipping and DP noise via (12), then updates the global model $\boldsymbol{\theta}_t$ via (13).
5: **end for**

Output: The final model parameter $\boldsymbol{\theta}_T$.

For clipping bounds, each client uses record-level bound R to bound the *per-record* gradient in (10) for a finite sensitivity in record-level DP; while the server uses client-level bound C to bound the difference between *client momentum* $\tilde{\boldsymbol{m}}_{t,i}$ and the previous noisy global momentum $\tilde{\boldsymbol{m}}_{t-1}$ in (12), which achieves Byzantine robustness as in LFH. For sampling rates, the client C_i samples a batch of records $\mathcal{D}_{i,t}$ from the local dataset $\mathcal{D}_i$ with sampling rate p_i, which provides privacy amplification for DP from *record-level* sampling; while the server samples a subset of clients with sampling rate q (where the sampled clients set is denoted by I_t), which provides privacy amplification for DP from *client-level* sampling.

Remark: comparison with non-private LFH. Comparing with the original non-private Byzantine-robust method LFH [20] (see Section 2.3), our differentially-private version has three differences. First, comparing with (1), the client gradient in (10) is computed by averaging the *clipped* per-record gradient (with clipping bound R), which bounds the sensitivity of final aggregation when adding/removing one record from the local dataset. Second, comparing with (3), the server adds Gaussian noise when computing the aggregated global momentum $\tilde{\boldsymbol{m}}_t$ in (12) to guarantee DP. Third, instead of aggregating all clients' momentum, our method also considers aggregating a subset of them, reflected by the index set I_t in (12). It provides additional privacy amplification from *client-level* sampling with sampling rate q. Note that the original privacy amplification is provided by *record-level* sampling.

4.2 Privacy Analysis

Before presenting the final privacy analysis of DP-BREM, we first show how we compute the sensitivity for the summation of clipped client momentum in (12) for privacy analysis of one iteration, shown in Lemma 2. We note that clients may have different sizes of local datasets $\mathcal{D}_i$ and can use different record-level sampling rates p_i, thus the record-level sensitivity (denoted by S_i) for different clients can be different.

Lemma 2 (DP Sensitivity). *We use $\|\cdot\|$ to denote L2-norm $\|\cdot\|_2$. In the t-th round, denote the query function $Q_t(\mathcal{D}) := \sum_{j\in I_t} \mathsf{Clip}_C(\boldsymbol{m}_{t,j} - \tilde{\boldsymbol{m}}_{t-1})$, where $\tilde{\boldsymbol{m}}_{t-1}$ is public and $\mathcal{D} = \{\mathcal{D}_j\}_{j\in I_t}$. Consider the neighboring dataset $\mathcal{D}' = \{\mathcal{D}_j\}_{j\neq i, j\in I_t} \cup \mathcal{D}_i'$ that differs in one record from client C_i's local data ($i \in I_t$), i.e., $|\mathcal{D}_i - \mathcal{D}_i'| = 1$, then the sensitivity with respect to client C_i is*

$$S_i := \max_{\mathcal{D},\mathcal{D}'} \|Q_t(\mathcal{D}) - Q_t(\mathcal{D}')\| = \min\left\{2C, \frac{R}{p_i|\mathcal{D}_i|}\right\} \quad (14)$$

Proof. (Sketch) According to (10), the sensitivity of $\bar{\boldsymbol{g}}_{t,i}$ is $\frac{R}{p_i|\mathcal{D}_i|}$ because each clipped term $\mathsf{Clip}_R(\cdot)$ has bounded L2-norm, i.e., $\|\mathsf{Clip}_R(\cdot)\| \leqslant R$. Then, due to the recursive representation of local momentum in (11), the sensitivity of $\boldsymbol{m}_{t,i}$ is $\frac{R}{p_i|\mathcal{D}_i|}$. Finally, the client-level clipping $\mathsf{Clip}_C(\cdot)$ introduces another upper bound for the sensitivity. Refer to the Appendix of our full-version paper [17] for detailed proof. □

Remark: comparison with the privacy accountant of DP-SGD momentum. As discussed in Section 3.2, the privacy accountant of DP-SGD with momentum (i.e., account for privacy cost of gradient, then do post-processing for momentum) requires clients to add noise in the *local gradients*, which leads to poor utility especially when Byzantine attacks exist. In Lemma 2, we account for the privacy cost of *aggregated momentum*, where the sensitivity is carefully computed from the bounded record-level gradient. Therefore, our scheme solves the three limitations shown in Section 3.2, which is explained as follows. First, only the server adds noise (which is the same as the central setting), thus the privacy-utility tradeoff is not impacted. Second, the noise is added after the centered clipping $\mathsf{Clip}_C(\tilde{\boldsymbol{m}}_{t,i} - \tilde{\boldsymbol{m}}_{t-1})$, thus it only introduces unbiased error. We also show that (in Section 4.3) the impact from the added noise is separate from the impact from Byzantine attacks, as versus the impact from the local noise is enlarged with Byzantine attacks in DP-LFH (see Section 3.2). Third, since privacy is accounted on momentum, and only the aggregated momentum leaks privacy, our solution enjoys privacy amplification from client-level sampling.

The final privacy analysis of DP-BREM is shown in Theorem 1. It presents how to compute the privacy budget ε and privacy parameter δ when the parameters (such as T, σ, q, etc.) of the algorithm are given. We use an advanced privacy accountant tool called Gaussian DP (GDP) [11], then convert it to (ε, δ)-DP. Note that in our privacy analysis, clients can use different record-level sampling rates p_i, thus different sensitivity S_i shown in (14). Therefore, the final privacy budget (denoted by ε_i) of DP-BREM may be different for different clients, which provides personalized privacy if these parameters are different for each client.

Theorem 1 (Privacy Analysis). *DP-BREM (in Algorithm 1) satisfies record-level (ε_i, δ)-DP for an honest client C_i with ε_i and δ satisfying*

$$\delta = \Phi\left(-\frac{\varepsilon_i}{\mu_i} + \frac{\mu_i}{2}\right) - e^{\varepsilon_i} \cdot \Phi\left(-\frac{\varepsilon_i}{\mu_i} - \frac{\mu_i}{2}\right), \quad (15)$$

where $\Phi(\cdot)$ denotes the cumulative distribution function (CDF) of standard normal distribution, and μ_i is defined by

$$\mu_i = q p_i \sqrt{T(e^{1/(2\sigma_i^2)} - 1)}, \quad \text{with } \sigma_i = \sigma \cdot \max\left\{\frac{R}{2C}, p_i|\mathcal{D}_i|\right\} \quad (16)$$

Proof. This result is obtained by the composition of multiple iterations and the privacy amplification from sampling. See Appendix A for the detailed proof. □

4.3 Convergence Analysis

Before presenting the final convergence analysis of our solution, we first show the aggregation error for one iteration in Theorem 2.

Theorem 2 (Aggregation Error). *Denote $\boldsymbol{m}_t^* := \frac{1}{|\mathcal{H}|}\sum_{i\in\mathcal{H}} \boldsymbol{m}_{t,i}$ as the ground truth aggregated raw momentum, where $\boldsymbol{m}_{t,i}$ is the client momentum computed from gradient* without *record-level clipping. Assume the local momentum of all honest clients $\{\boldsymbol{m}_{t,i}\}_{i\in\mathcal{H}}$ are i.i.d. with expectation $\boldsymbol{\mu} := \mathbb{E}[\boldsymbol{m}_{t,i}]$, and the variance is bounded (in terms of L2-norm)*

$$\mathbb{E}\|\boldsymbol{m}_{t,i} - \boldsymbol{\mu}\|^2 \leqslant \rho^2 \quad (17)$$

After the following parameter tuning of the clipping bounds:

$$R \propto O\left(\rho\sqrt{n/(|\mathcal{B}| + \sqrt{d}\sigma/q)}\right), \quad C \propto O(R) \quad (18)$$

we have the following aggregation error due to clipping, DP noise, and Byzantine clients:

$$\mathbb{E}\|\tilde{\boldsymbol{m}}_t - \boldsymbol{m}_t^*\|^2 \leqslant O\left(\frac{\rho^2(|\mathcal{B}| + \sqrt{d}\sigma/q)}{n}\right) \quad (19)$$

where $|\mathcal{B}|$ is the number of Byzantine clients, d is the dimension of model parameter $\boldsymbol{\theta}_t$, σ is the noise multiplier (for DP) shown in (12), q is the client-level sampling rate shown in Line-2 of Algorithm 1, and ρ is defined in (17).

Proof. (Sketch) Directly bounding $\mathbb{E}\|\tilde{\boldsymbol{m}}_t - \boldsymbol{m}_t^*\|^2$ is not easy, thus we utilize the upper bounds of $\mathbb{E}\|\tilde{\boldsymbol{m}}_t - \boldsymbol{\mu}\|^2$ and $\mathbb{E}\|\boldsymbol{\mu} - \boldsymbol{m}_t^*\|^2$ to get the final result, where $\boldsymbol{\mu} := \mathbb{E}[\boldsymbol{m}_{t,i}]$ is the expected local momentum (we assume clients' local momentum are i.i.d.). When upper bounding $\mathbb{E}\|\tilde{\boldsymbol{m}}_t - \boldsymbol{\mu}\|^2$, we decompose errors into three types: honest clients' error (from clipping randomness and bias), Byzantine clients' error (from perturbation), and DP noise error. Optimizing parameters C and R minimizes the total error. See the Appendix of our full-version paper [17] for the detailed proof and the formal version of (18) and (19). □

Interpretation of Theorem 2. The value of $\mathbb{E}\|\tilde{\boldsymbol{m}}_t - \boldsymbol{m}_t^*\|^2$ quantifies the aggregation error, i.e., how the aggregated privatized momentum $\tilde{\boldsymbol{m}}_t$ (with clipping, DP noise, and Byzantine clients' impact) differs from the "pure" momentum aggregation $\boldsymbol{m}_t^*$, where only honest clients participate and without clipping and DP noise. According to (19), the aggregation error is proportional to ρ^2 and $\frac{|\mathcal{B}|}{n} + \frac{\sqrt{d}\sigma}{nq}$, where ρ^2 quantifies the variance of honest clients' local momentum, $\frac{|\mathcal{B}|}{n}$ is the fraction of Byzantine clients, and $\frac{\sigma}{nq} = O(1/\varepsilon)$ for ε-DP. In other words, the aggregation error will be enlarged when: honest clients' variance is large, or the Byzantine attacker corrupts more clients, or the training model is complex (i.e., the model dimension d is large), or we need stronger privacy (i.e., a smaller ε), or the number of clients n is small. Furthermore, due to the format of $\frac{|\mathcal{B}|}{n} + \frac{\sqrt{d}\sigma}{nq}$, the impact from DP noise is independent of the increase of Byzantine clients $|\mathcal{B}|$ (versus Limitation 2 of DP-LFH in Section 3.2). On the other hand, according to the parameter tuning in (18), we could theoretically set a smaller record-level clipping bound R when σ, d, and $|\mathcal{B}|$ are large, or ρ and n are small. The tuning of client-level clipping bound C should be adjusted according to the value of R. Recall that R is for DP, while C is for robustness.

By following the convergence analysis in [20] and using the result in (19), we have the convergence rate shown below.

Theorem 3 (Convergence Rate of DP-BREM)**.** *The convergence rate of DP-BREM in Algorithm 1 is asymptotically (ignoring constants and higher order terms) of the order*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla\ell(\boldsymbol{\theta}_{t-1})\|^2 \lesssim \sqrt{\frac{\rho^2}{T}\frac{|\mathcal{B}| + (1+\sqrt{d}\sigma)/q}{n}} \tag{20}$$

where $\ell(\cdot)$ is the loss function, T is the total number of training iterations, and other parameters are the same as in (19).

Proof. See Appendix B. □

Remark: comparison with LFH and DP-LFH. The convergence rate of the non-private LFH, DP-LFH, and the proposed solution DP-BREM, showing in (5), (8), and (20) respectively, are summarized in Table 2. Though both DP-LFH and DP-BREM pay an additional term of $\sqrt{d}\sigma/q$ to get the DP property, they have different impacts on the convergence. As discussed in Limitation 2 of Section 3.2, the additional term $\sqrt{d}\sigma/q$ of DP-LFH (due to DP noise added to clients' gradient) is on the term ρ, thus it will enlarge the impact of Byzantine clients (i.e., the term $|\mathcal{B}|$). However, the additional term $\sqrt{d}\sigma/q$ of our solution DP-BREM (due to DP noise added to the aggregated momentum) is on the term $1+|\mathcal{B}|$, which has a squared-root order. Therefore, DP noise only has a limited impact on the convergence of DP-BREM when there are Byzantine clients. We will validate the above theoretical analysis via experimental results in Section 6.

Table 2: Comparison of Convergence Rate

	Where to add noise	Convergence Rate
LFH [20]	None	$O(\rho\sqrt{1+\|\mathcal{B}\|})$
DP-LFH	Clients' gradients	$O\left((\rho+\sqrt{d}\sigma)\sqrt{1+\|\mathcal{B}\|}\right)$
DP-BREM	Aggregated momentum	$O\left(\rho\sqrt{1+\|\mathcal{B}\|+\sqrt{d}\sigma}\right)$

5 DP-BREM$^+$ with Secure Aggregation

The private and robust FL solution DP-BREM (in Section 4) assumes a *trusted* server which can access clients' momentum. In this section, we propose DP-BREM$^+$, which assumes a *malicious* server and utilizes secure aggregation techniques, achieving the same DP and robustness guarantees as DP-BREM. As discussed in Section 3.1, we consider the server as malicious only for data privacy, while clients are malicious for both data privacy and Byzantine attacks.

5.1 Challenges

Considering the server is malicious for data privacy, the noisy aggregate of momentum with centered clipping shown in (12) must be implemented securely with the goals of 1) privacy, i.e., each party, including clients and the server, learns nothing but the differentially-private output; and 2) integrity, i.e., the output is correctly computed. Since the noisy aggregated momentum of the previous iteration $\tilde{\boldsymbol{m}}_{t-1}$ already satisfies DP, we can regard it as public information and only need to focus on securely computing the term $\sum_{i\in I_t}\mathsf{Clip}_C(\tilde{\boldsymbol{m}}_{t,i} - \tilde{\boldsymbol{m}}_{t-1}) + \mathcal{N}(0, R^2\sigma^2\mathbf{I}_d)$ in (12).

Secure Aggregation with Verified Inputs (SAVI). The key crypto technique we leverage to achieve the above objectives is SAVI [30], which is a type of protocols that securely aggregate only well-formed inputs. The security goals include both *privacy* and *integrity*. Specifically, privacy means that no party should be able to learn anything about the raw input of an honest client, other than what can be learned from the final aggregation result. Integrity means that the output of the protocol returns the correct aggregate of well-formed input, where 1) an input u passes the input integrity check with a public validation predicate $\mathsf{Valid}(\cdot)$ if and only if $\mathsf{Valid}(u) = 1$, and 2) the aggregation is correctly computed. An instantiation of the SAVI protocol is EIFFeL [30] (described in the Appendix of our full-version paper [17]).

Challenge: Secure Generation of Gaussian Noise. A SAVI protocol can potentially solve the problem of securely aggregating the clipped vectors (by enforcing a norm-bound on the client momentum difference). However, the Gaussian noise $\mathcal{N}(0, R^2\sigma^2\mathbf{I}_d)$ needs to be securely generated and aggregated as well. In DP-BREM with a trusted server, the Gaussian noise $\mathcal{N}(0, R^2\sigma^2\mathbf{I}_d)$ is generated by the server to guarantee DP. However, when the server is assumed as malicious, the added Gaussian noise for DP cannot be directly

generated by the server.

A straightforward approach is to use a semi-honest server, as proposed in [31], to generate DP noise and manage the privacy engine. However, relying on another non-colluding server may be impractical, so we assume only a single server. An alternative is Distributed DP [34], where clients locally generate Gaussian noise. The aggregated noise follows a Gaussian distribution with an enlarged standard deviation, ensuring DP through cryptographic techniques. This method, however, has two limitations: it requires more noise to achieve the same privacy level due to potential collusion among malicious clients, and the robustness is compromised as malicious clients can generate arbitrary local noise.

A possible solution to address the first limitation is to *jointly* generate Gaussian noise as in [29], where no party learns or controls the true value of the noise (or a portion of the noise). However, the protocol in [29] is designed only for additive secret sharing schemes, which only works for honest-but-curious parties and does not tolerate malicious parties. Moreover, in [29], the Gaussian noise is jointly generated by honest-but-curious and non-colluding parties, which does not address the second limitation as the clients can be malicious in our threat model discussed in Section 3.1.

Overview of DP-BREM$^+$. To achieve secure aggregation with verified inputs and secure Gaussian noise generation under the threat model of a malicious server and malicious minority of clients, our DP-BREM$^+$ 1) leverages an existing SAVI protocol called EIFFeL [30] to achieve secure input validation; and 2) introduces a new protocol to achieve secure noise generation that is compatible with EIFFeL. The idea of *jointly* generating Gaussian noise in DP-BREM$^+$ is inspired by [29], but our design is based on Shamir's secret sharing [32] with robust reconstruction by following the design in EIFFeL, thus guarantees security under malicious minority. We present the preliminaries of Shamir's secret sharing and EIFFeL protocol in the Appendix of our full-version paper [17].

5.2 Design of DP-BREM$^+$

As discussed in Section 5.1, the main task of DP-BREM$^+$ is to securely compute the term $\sum_{i\in I_t} \mathsf{Clip}_C(\tilde{\boldsymbol{m}}_{t,i} - \tilde{\boldsymbol{m}}_{t-1}) + \mathcal{N}(0, R^2\sigma^2\mathbf{I}_d)$ shown in (12). After computing local momentum $\tilde{\boldsymbol{m}}_{t,i}$ via (11), each client C_i first implements centered clipping to get $\boldsymbol{z}_i := \mathsf{Clip}_C(\tilde{\boldsymbol{m}}_{t,i} - \tilde{\boldsymbol{m}}_{t-1})$, which is the private input for validation and aggregation.

Three-Phase Design. In DP-BREM$^+$, clients and the server jointly implement three phases: 1) secure input validation to validate the client momentum is properly centered clipped by C, 2) secure noise generation, where clients generate shares of Gaussian noise which can be aggregated in Phase 3 to ensure DP, and 3) aggregation of valid inputs and noise to obtain the noisy global model. We assume the arithmetic circuit is computed over a finite field $\mathbb{F}_{2^K}$. The illustration of

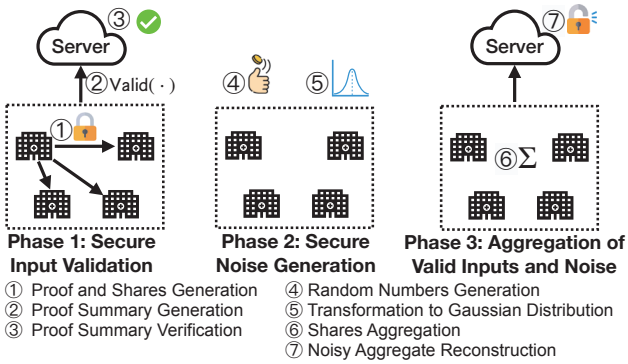

Figure 2: Illustration of DP-BREM$^+$ (see Appendix C for detailed steps ①-⑦)

DP-BREM$^+$ is shown in Figure 2. Due to limited space, we present the detailed steps ①-⑦ in Appendix C.

Phase 1: Secure Input Validation. The validation function for an input $\boldsymbol{z}_i$ considered in DP-BREM$^+$ is defined as $\mathsf{Valid}(\boldsymbol{z}_i) := \mathbb{1}(\|\boldsymbol{z}_i\| \leqslant C)$, where $\mathsf{Valid}(\boldsymbol{z}_i) = 1$ if and only if the condition $\|\boldsymbol{z}_i\| \leqslant C$ holds. Since honest clients compute $\boldsymbol{z}_i = \mathsf{Clip}_C(\tilde{\boldsymbol{m}}_{t,i} - \tilde{\boldsymbol{m}}_{t-1})$, verifying whether $\boldsymbol{z}_i$ is well-formed, with bounded L2-norm via $\mathsf{Valid}(\cdot)$, for all clients ensures centered clipping of client momentum $\tilde{\boldsymbol{m}}_{t,i}$ (to achieve robustness as DP-BREM). We follow the design in EIFFeL [30] for secure input validation, which returns the validation result $\mathsf{Valid}(\boldsymbol{z}_i)$ (either 1 or 0) for client C_i's private input $\boldsymbol{z}_i$, corresponding to steps ①, ②, and ③ shown in Figure 2. Then, clients and the server can jointly verify all inputs $\{\boldsymbol{z}_i\}_{i\in I_t}$, and obtain the set of valid inputs I_{Valid}, where $\mathsf{Valid}(\boldsymbol{z}_i) = 1$ for all $i \in I_{\mathsf{Valid}}$. In the later step, only inputs in I_{Valid} are aggregated.

Phase 2: Secure Noise Generation. We develop a new protocol for secure distributed Gaussian noise generation, which returns the shares (held by each client) of a random vector $\boldsymbol{\xi}$ of length d from the Gaussian distribution $\mathcal{N}(0, R^2\sigma^2\mathbf{I}_d)$, corresponding to steps ④ and ⑤ shown in Figure 2. The shares of noise can be reconstructed into a single Gaussian noise (for ensuring DP) with the guarantee that no parties know or control the generated noise, which protects the information of private inputs after the noisy aggregate is released.

Phase 3: Aggregation of Valid Inputs and Noise. Finally, the server and clients can aggregate the valid inputs (obtained in Phase 1) and the generated Gaussian noise (obtained in Phase 2) by implementing steps ⑥ and ⑦ shown in Figure 2, ensuring nothing except the noisy aggregate can be learned.

Remark on Efficiency. DP-BREM$^+$'s usage of EIFFeL's secure input validation is due to efficiency considerations. Instead of having clients perform clipping and using secure input validation, one alternative is to use standard secure multi-party computation (MPC) for the clipping and aggregation. However, doing this under MPC would result in a very large computation/communication overhead due to the multiplication, min-operation, division, and L2-norm computation in the clipping operation $\mathsf{Clip}_C(\cdot)$ defined in (4). In contrast, the

secure input validation protocol only requires the verifiers to check all the multiplication gates very efficiently with just one identity test. The compatibility with secure input validation is one of the advantages of DP-BREM.

Complexity. According to EIFFeL [30], the computation/communication complexity of secure aggregation with input validation is $O(mnd)$ for clients and $O(n^2 + md\min\{n, m^2\})$ for the server in terms of the number of clients n, number of malicious clients m, and data dimension d. For the proposed secure noise generation (only clients are involved), the computation/communication complexity for total n clients is $O(mnd)$.

5.3 Security Analysis

In comparison, EIFFeL [30] is a secure aggregation protocol with verified inputs (without guaranteeing DP), while our solution DP-BREM$^+$ is a secure noisy aggregation protocol with verified inputs and jointly generated Gaussian noise, which provides DP on the aggregated results. Therefore, the only difference is the Gaussian noise that will be aggregated to the final result. We show the formal security guarantee of DP-BREM$^+$ in the following theorem.

Theorem 4 (Security Guarantees of DP-BREM$^+$)**.** *For the validation function* $\mathsf{Valid}(\cdot)$ *considered in Section 5.2, given a security parameter* κ*, the secure noisy aggregation protocol in DP-BREM*$^+$ *satisfies:*

1) Integrity. *For a negligible function* $negl(\cdot)$*, the output of the protocol returns the noisy aggregate of a subset of clients* I_{Valid} *and Gaussian noise* $\boldsymbol{\xi}$*, such that all clients in* I_{Valid} *have well-formed inputs:*

$$\Pr[output = \sum\nolimits_{i\in I_{\mathsf{Valid}}} \boldsymbol{z}_i + \boldsymbol{\xi}] \geqslant 1 - negl(\kappa)$$

where random vector $\boldsymbol{\xi} \sim \mathcal{N}(0, R^2\sigma^2\mathbf{I}_d)$*, and* $\mathsf{Valid}(\boldsymbol{z}_i) = 1$ *for all* $i \in I_{\mathsf{Valid}}$*. Note that the set* I_{Valid} *contains all honest clients (denoted by* I_H*) and the malicious clients who submitted well-formed input (denoted by* I_M^**), i.e.,* $I_{\mathsf{Valid}} = I_H \cup I_M^*$*.*

2) Privacy. *For a set of malicious clients* I_M *and a malicious server* S*, there exists a probabilistic polynomial-time (P.P.T.) simulator* $\mathsf{Sim}(\cdot)$ *such that:*

$$\mathsf{Real}\left(\{z_i\}_{i\in I_H}, \Omega_{I_M\cup\mathsf{S}}\right) \equiv_{\mathsf{C}} \mathsf{Sim}\left(\sum\nolimits_{i\in I_H} \boldsymbol{z}_i + \boldsymbol{\xi}, I_H, \Omega_{I_M\cup\mathsf{S}}\right)$$

where $\{z_i\}_{i\in I_H}$ *denotes the input of all the honest clients,* Real *denotes a random variable representing the joint view of all the parties in the protocol's execution,* $\Omega_{I_M\cup\mathsf{S}}$ *indicates a polynomial-time algorithm implementing the "next-message" function of the parties in* $I_M \cup \mathsf{S}$ *(see [30, Appendix 11.5]), and* $\equiv_{\mathsf{C}}$ *denotes computational indistinguishability. In summary, the server and clients learn nothing besides the final aggregated result.*

Proof. See the Appendix of our full-version paper [17]. □

6 Experimental Evaluation

In this section, we demonstrate the effectiveness of the proposed DP-BREM/DP-BREM$^+$ on achieving both good privacy-utility tradeoff and Byzantine robustness via experimental results on MNIST [23], CIFAR-10 [22], and FEMNIST [8] datasets with non-IID setting (refer to Appendix D for more details on the datasets and model architectures). Note that MNIST and CIFAR-10 have 10 classes, while FEMNIST includes 62 classes. All experiments are developed via PyTorch[1].

Byzantine Attacks. We consider four existing Byzantine attacks in our experiments, including ALIE ("a little is enough") [3], IPM (inner-product manipulation) [38], LF (label-flipping), and the state-of-the-art MTB ("manipulating-the-Byzantine") [33]. Refer to Appendix D for more details.

Compared Methods. We compare the performance of six approaches against Byzantine attacks, including DP-BREM/$^+$ (our approach)[2], a variant of DP-FedSGD [26] with both record and client norm clipping, DDP-RP [36], DP-RSA [43], a variant of CM [40] with DP noise, and DP-LFH. The comparison (on trust assumption and mechanism overview) of these approaches is provided in Table 1, and Appendix D shows more details of each approach. In summary, DP-BREM/$^+$, DP-FedSGD, and DP-CM add central noise to the aggregation, but DP-BREM$^+$ does not require a trusted server due to the secure aggregation technique. DDP-RP adds *partial* local noise to the client's update with secure aggregation. DP-RSA and DP-LFH add local noise to the client's update. We fix $\delta = 10^{-6}$ for (ε, δ)-DP in all experiments. For the setting of other parameters, refer to Appendix E.

Evaluation Metric. We evaluate the testing accuracy of the global model within T iterations. Considering the accuracy curve might be unstable under Byzantine attacks, we average the accuracy between $0.9T$ and T as the final accuracy for comparison. Note that both DP noise and Byzantine attacks reduce the accuracy. A protocol achieves good Byzantine robustness if its accuracy does not decrease too much with an increased number of Byzantine clients.

6.1 Robustness Evaluation with DP

We consider a fixed privacy budget ε and implement each attack with different percentages of Byzantine clients $\delta_B = \frac{|\mathcal{B}|}{n}$ for the four attacks, and compare the accuracy among all approaches. We note that a complex dataset requires a more sophisticated model architecture and makes it more challenging to maintain good utility in the presence of DP and Byzantine attacks. Therefore, in our experiments with CIFAR-10 (which

[1] Our source code is available at https://github.com/xiaolangu/DP-BREM

[2] Since DP-BREM$^+$ achieves the same DP and robustness guarantees as DP-BREM, we did not perform the empirical experiments with secure aggregation because the accuracy results will be exactly the same as DP-BREM. We use DP-BREM/$^+$ to denote both DP-BREM and DP-BREM$^+$, and the implementation follows Algorithm 1.

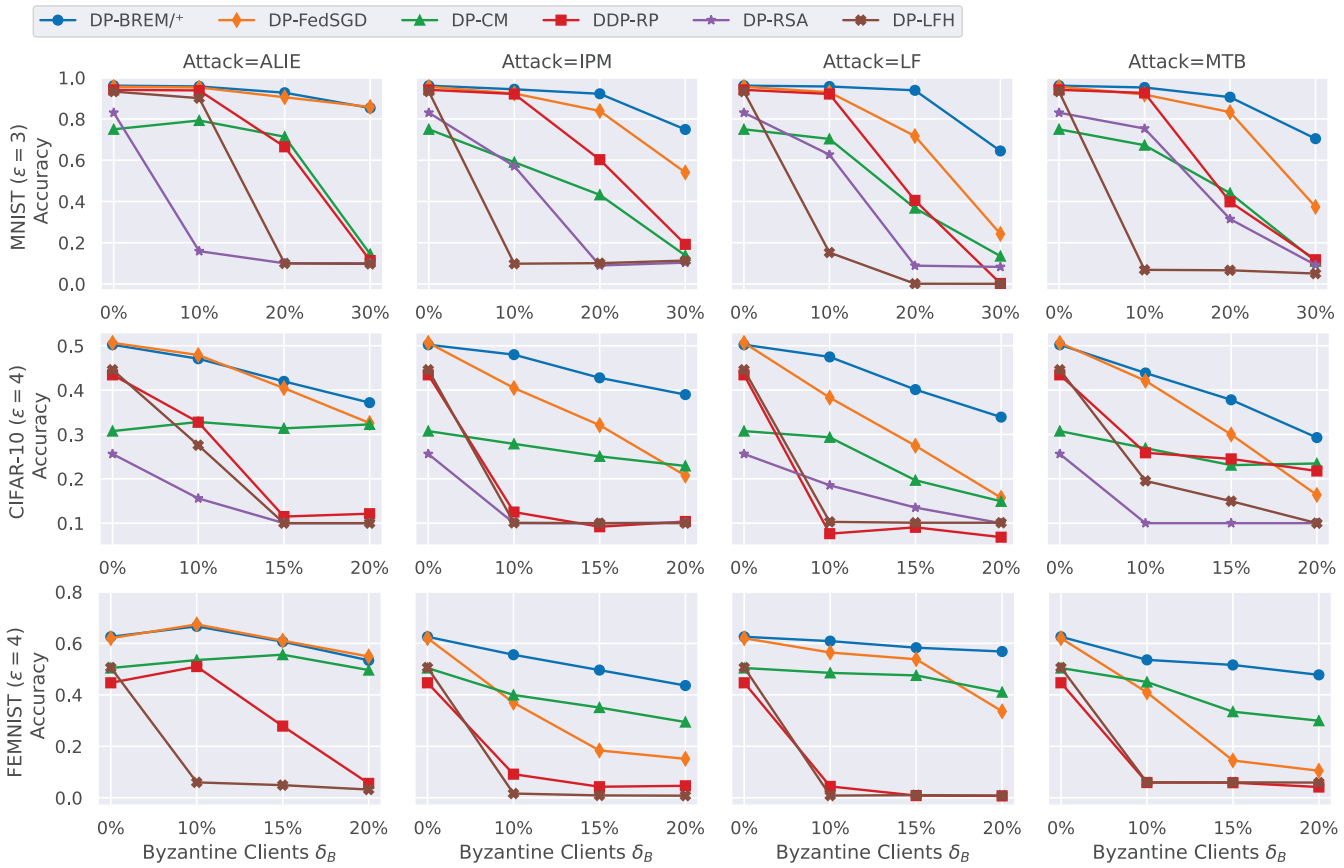

Figure 3: With fixed privacy budget ε, varying the percentage of Byzantine clients δ_B for three datasets.

has three color channels) and FEMNIST (which includes 62 classes), we use slightly larger ε values and a smaller number of Byzantine clients. These choices are still within a reasonable range. Previous papers, such as [1] and [42], also used larger privacy budgets for the CIFAR-10 dataset compared to the MNIST dataset. The results for MNIST (with $\varepsilon = 3$), CIFAR-10 (with $\varepsilon = 4$), and FEMNIST (with $\varepsilon = 4$) datasets are shown in Figure 3. Compared to the results on the MNIST dataset (with 10 classes), the accuracy on the FEMNIST dataset is lower due to the larger number of classes. Though the detailed results vary under different attacks and across three datasets, we have some general observations:

1) When there is no attack, i.e., $\delta_B = 0$, DP-BREM/$^+$ achieves almost the same accuracy as DP-FedSGD, indicating the Byzantine-robust design (client momentum with centered clipping) has almost no impact on the utility in this case.

2) After increasing δ_B, our DP-BREM/$^+$ has the smallest accuracy decrease, indicating its success in providing Byzantine robustness. However, the accuracy of DP-LFH reduces sharply, demonstrating that the large aggregated local DP noise makes the robust aggregator more vulnerable to Byzantine attacks, which is consistent with our discussions of Limitation 2 in Section 3.2.

3) Though DP-FedSGD has client-level gradient clipping, which can restrict malicious clients' impact, it is still vulnerable to some types of Byzantine attacks (such as IPM and MTB) under larger δ_B values.

4) CM with DP noise (or DP-CM) has a relatively small accuracy decrease for a relatively small δ_B. It is the benefit of the median-based robust aggregator. But the sensitivity is larger than the average-based aggregators, as discussed in Example 1, the aggregated DP noise is too large to obtain a high accuracy, even when $\delta_B = 0$.

5) DDP-RP is more vulnerable to LF attack because it only checks the element-wise range. Also, the model replacement strategy in LF attack is more likely to change the positions that have small values in benign gradient vectors.

6) DP-RSA has relatively poor accuracy compared to other approaches, even when $\delta_B = 0$. This is caused by the sign-SGD aggregator, which only aggregates element-wise *signs* rather than the full precision gradient, leading to large information loss. Moreover, the local DP noise makes Byzantine attacks easier to succeed. We note that DP-RSA does not converge for the FEMNIST dataset (possibly caused by the sign aggregation), even without DP noise and Byzantine clients, and thus we do not present the results for this dataset.

7) Under the ALIE attack, it is possible for a small number of Byzantine clients to improve the accuracy of the model compared to the scenario without any Byzantine clients. For instance, an ALIE attack with 10% Byzantine clients can

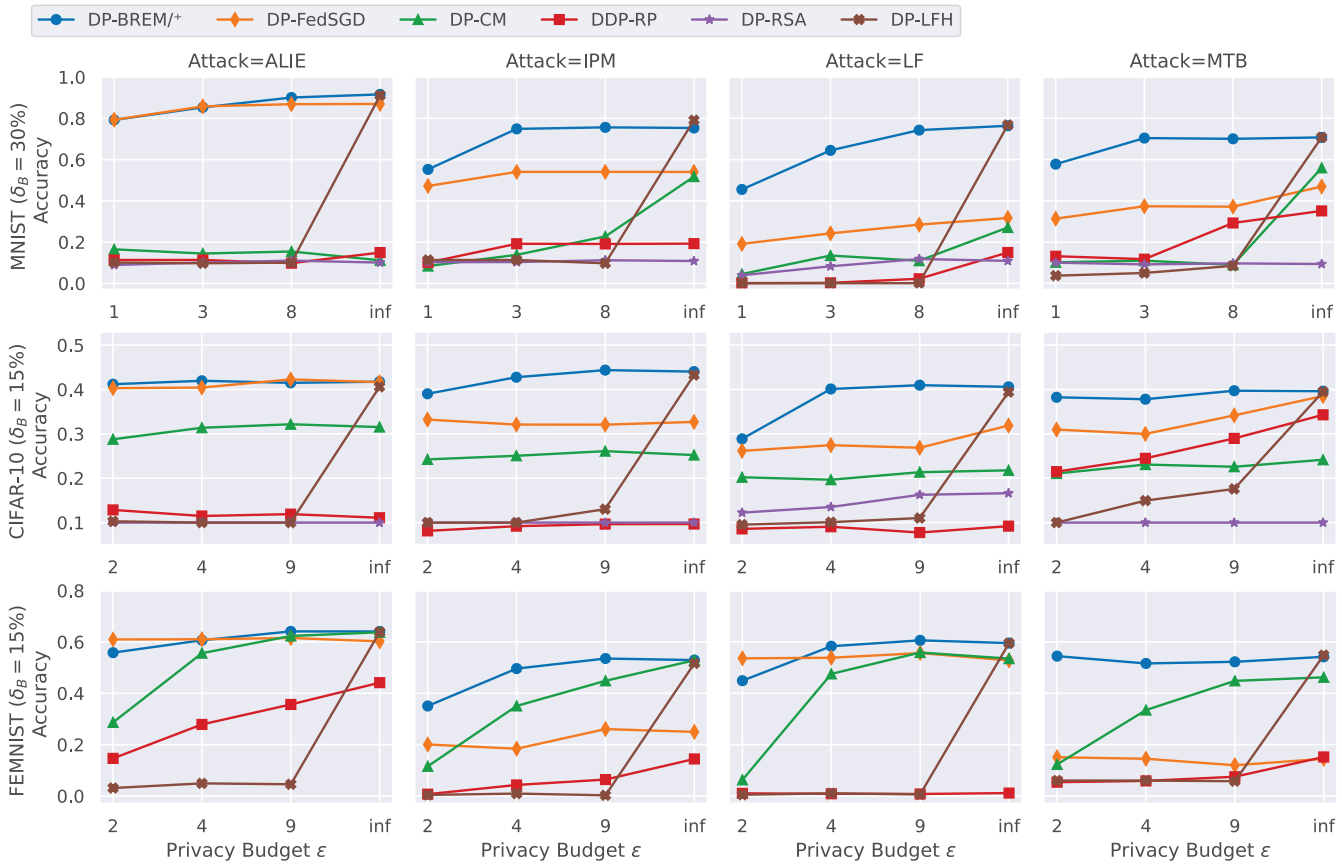

Figure 4: With fixed percentage of Byzantine clients δ_B, varying privacy budget ε for three datasets.

achieve higher accuracy than the case with 0% Byzantine clients across all defense aggregators except DP-LFH. This improvement occurs because the ALIE attack generates malicious gradients that are close to the averaged good gradients but deviate slightly using a scaling factor. This factor is determined based on the total number of clients and the proportion of Byzantine clients, designed to bypass any anomaly detection mechanisms employed by the central server. Consequently, when the number of Byzantine clients is relatively small, the malicious gradients can enhance model accuracy compared to benign gradients, where the record-level clipping is used to achieve DP.

6.2 Privacy-Utility Tradeoff under Attack

We consider a fixed percentage of Byzantine clients δ_B for each attack under different values of privacy budget ε, and compare the accuracy of all approaches. The results for MNIST (with $\delta_B = 30\%$), CIFAR-10 (with $\delta_B = 15\%$), and FEMNIST (with $\delta_B = 15\%$) datasets are shown in Figure 4. For all three datasets, we consider four different levels of privacy, where $\varepsilon = \text{inf}$ means the standard deviation of DP noise is 0. However, we still implement record-level clipping to illustrate how the noise affects the results while keeping other settings including the clipping step the same.

It's essential to highlight that while the privacy-utility curve is generally monotonic in the absence of Byzantine attacks, this may not hold under Byzantine attacks due to two sources of perturbation. When malicious perturbation dominates, the impact of DP noise on utility is typically minimal. Additionally, different defense aggregators exhibit varying sensitivities to malicious perturbation and DP noise across various datasets, even when the number of malicious clients and ε values are the same. Consequently, observations can vary across defense aggregators, attacks, and datasets (with different parameters). For example, DP noise has a very small (or almost negligible) impact on DP-FedSGD compared to DP-BREM. This could be because DP-FedSGD aggregates more information than the momentum-based solution, leading to a better signal-to-noise ratio (SNR) and thus greater robustness to DP noise. However, the attack has a more significant impact on DP-FedSGD.

Though the detailed results vary under different attacks and across the three datasets, DP-BREM/$^+$ generally achieves the highest accuracy among almost all approaches, especially under IPM and MTB attacks. The only exception is when $\varepsilon = 2$ for the FEMNIST dataset, where the accuracy of DP-BREM/$^+$ is lower than that of DP-FedSGD. This is because the client momentum in DP-BREM/$^+$ restricts the information that can be learned from each new iteration, making the increased DP

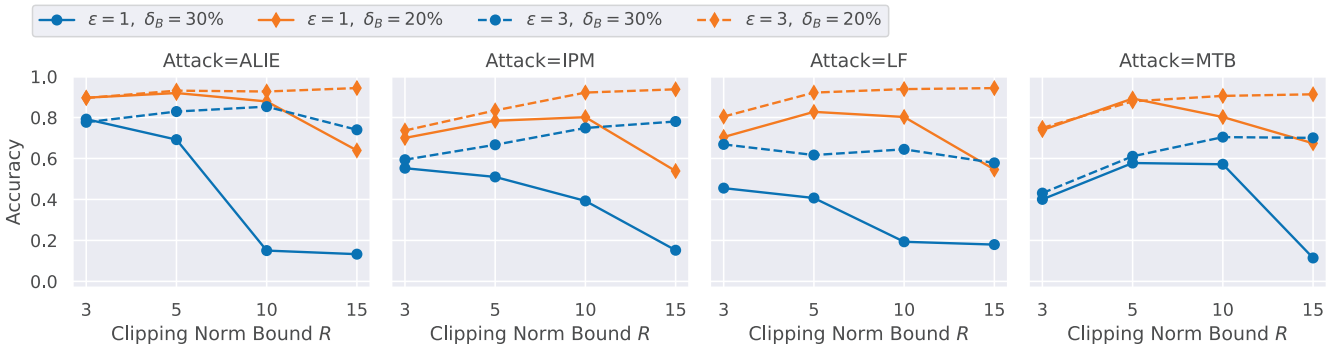

Figure 5: MNIST: Varying record-level clipping bound R for DP-BREM under different settings.

Table 3: Running time[1] (in milliseconds) per round per client on the MNIST dataset.

Batch Size	Baseline (FedSGD)	FedSGD+DP (efficient[2])	DP-BREM (DP+robust)	FedSGD+DP (inefficient[3])
30	11.80	13.31	13.72	41.06
60	18.23	19.79	20.27	76.70
120	31.22	33.18	33.70	149.32

[1] Our GPU device is NVIDIA Tesla P100-PCIE-16GB. Using other GPU devices may have different results.

[2] By default, our implementation uses *efficient* per-record gradient clipping by following Opacus library's implementation with parallel clipping and optimized einsum (refer to https://opacus.ai/api/_modules/opacus/optimizers/optimizer.html#DPOptimizer)

[3] To illustrate the improvement of *efficient* clipping, we also show the results of the *inefficient* implementation, which clips per-record gradient sequentially and without using optimized einsum.

noise have a greater impact on the model's accuracy.

Note that when $\sigma = 0$ (i.e., $\varepsilon = \text{inf}$), both DP-BREM/$^+$ and DP-LFH reduce to LFH, thus they have the same results in this case. We can observe that with a moderate privacy budget, such as $\varepsilon \geqslant 2$, DP noise only has a negligible impact on the accuracy. But if ε is too small, such as $\varepsilon = 1$ for the MNIST dataset in Figure 4, DP-BREM/$^+$ suffers a relatively larger impact (but still acceptable) from DP noise. Note that when there exist Byzantine attacks, reducing the DP noise to $\sigma = 0$ (i.e., $\varepsilon = \text{inf}$) does not significantly improve the accuracy of DP-BREM/$^+$ compared with $\varepsilon < \text{inf}$, because Byzantine clients' perturbations largely impact the performance. However, the accuracy of DP-LFH is greatly reduced when $\varepsilon < \text{inf}$, since the local DP noise impacts the robustness of the aggregator. This observation is consistent with our theoretical analysis in Limitation 2 of DP-LFH (Section 3.2).

6.3 Other Results

Efficiency Evaluation of DP and Byzantine Robustness. We note that DP and Byzantine Robustness designs in our solution only introduce a small computation overhead, because 1) the clipping step of DP can be implemented efficiently; 2) our robustness is essentially a clipped summation of client momentum without any complex computations. Due to limited resources, we implemented the distributed training of FL on a single machine (by running all the clients and the server code sequentially). We evaluate the efficiency of DP-BREM via the running time (per round per client) on the MNIST dataset. The results shown in Table 3 indicate that the DP noise and Byzantine robustness only incur 8% $\sim$ 16% additional running time (depending on batch size).

Table 4: Model accuracy when varying C of DP-BREM/$^+$ with $\varepsilon = 4$ under IPM and MTB attacks on FEMNIST dataset.

δ_B	$C = 0.5$	$C = 1$	$C = 2$	$C = 3$	$C = 4$
0%	0.622	**0.647**	0.625	0.621	0.627
IPM 10%	0.407	0.524	**0.555**	0.528	0.514
IPM 20%	0.060	0.305	**0.436**	0.413	0.392
MTB 10%	0.591	**0.605**	0.535	0.525	0.545
MTB 20%	**0.554**	0.537	0.477	0.426	0.426

Impact of R in DP-BREM/$^+$. Figure 5 shows how the accuracy changes w.r.t. the record-level clipping bound R in DP-BREM/$^+$. The results demonstrate that when there are fewer Byzantine clients (i.e., smaller δ_B) or the noise multiplier σ is smaller (i.e., larger ε), we need to set a larger R to obtain better accuracy. This observation is consistent with the theoretical analysis of parameter tuning discussed in Theorem 2 and its interpretation.

Impact of C in DP-BREM/$^+$. We use the fixed client-level clipping bound C for each dataset in previous experiments. Table 4 illustrates how varying values of C (while keeping the default and fixed R) can influence model accuracy. In the absence of Byzantine attacks, the value of C has a relatively small impact on the model accuracy. However, in the presence of Byzantine attacks, the effect of C varies depending on the nature of the attack. For instance, attacks like the IPM attack, which deviate significantly from benign gradients, benefit from a slightly larger C as it allows more useful information (from benign clients) to be retained. Conversely, for attacks like the MTB attack, which aim to evade detection by aligning more closely with benign gradients, a slightly smaller C can improve accuracy by reducing the impact of the attack on the aggregated gradient.

Impact of q in DP-BREM/$^+$. In previous experiments, we set client-level sampling rate $q = 1$ by default. As discussed

Table 5: Model accuracy when varying q of DP-BREM/$^+$ with $\varepsilon = 2$ under MTB attack on the CIFAR-10 dataset.

δ_B	$q = 1$	$q = 0.8$	$q = 0.6$	$q = 0.4$	$q = 0.2$
0%	0.503	**0.525**	0.504	0.491	0.485
10%	0.435	0.434	**0.465**	0.449	0.438
20%	0.255	0.284	0.297	**0.328**	0.241

in Sec. 4.1, aggregating a subset I_t of clients in (12) is one of the major differences from LFH. Table 5 demonstrates the utility improvement by optimizing q under different attack percentages δ_B. Intuitively, without attacks, a smaller q enhances privacy amplification, reducing the required σ for a given ε in DP; however, too small a q increases aggregation variance. Under Byzantine attacks, a smaller q mitigates attack impact as only a subset of Byzantine clients are aggregated. Thus, with higher δ_B the optimal q (highlighted in Table 5) decreases.

7 Related Work

Due to limited space, we only discuss the most relevant defenses below and put other related work in the Appendix of our full-version paper [17]. Other works either only achieve DP or Byzantine robustness (but not both), or combine secure aggregation with Byzantine robustness without realizing DP.

Wang et al. [36] proposed DDP-RP, an FL scheme offering Distributed DP (via encryption) and robustness (via range-proof technologies). However, this scheme only verifies if local model weights are within a bounded range, providing weak robustness. Our solution, in contrast, employs client momentum and centered clipping for Byzantine robustness with provable convergence. Zhu et al. [43] uses *sign* aggregation for robustness, thus each client has limited impact, and adds DP noise to local gradients before sign operations. This method suffers from information loss, resulting in degraded convergence, and only accounts for the privacy cost of one iteration, underestimating the overall cost. Our solution, based on original SGD with momentum, considers the privacy cost of all iterations. Experimental results show that DP-BREM outperforms both approaches.

8 Conclusions

This paper aims to achieve FL in the cross-silo setting with both DP and Byzantine robustness. We first proposed DP-BREM, a DP version of LFH-based FL protocol with a robust aggregator based on client momentum, where the server adds noise to the aggregated momentum. Then we further developed DP-BREM$^+$ which relaxes the server's trust assumption, by combining secure aggregation techniques with verifiable inputs and a new protocol for secure joint noise generation. DP-BREM$^+$ achieves the same DP and robustness guarantees as DP-BREM, under a malicious server (for privacy) and malicious minority clients. We theoretically analyze the error and convergence of DP-BREM, and conduct extensive experiments that empirically show the advantage of DP-BREM/$^+$ in terms of privacy-utility tradeoff and Byzantine robustness over five baseline protocols. In the future, we will extend our work to other types of robust aggregators.

Acknowledgments

The authors would like to thank the anonymous reviewers and the shepherd for their valuable comments and suggestions. Li Xiong was partly supported by NSF grants CNS-2124104, CNS-2125530, IIS-2302968, and NIH grants R01LM013712, R01ES033241.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.

[2] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[3] Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[4] Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. Protection against reconstruction and its applications in private federated learning. *arXiv preprint*, 2018.

[5] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *NeurIPS*, 2017.

[6] George EP Box and Mervin E Muller. A note on the generation of random normal deviates. *The annals of mathematical statistics*, 1958.

[7] Zhiqi Bu, Jinshuo Dong, Qi Long, and Weijie J Su. Deep learning with gaussian differential privacy. *Harvard Data Science Review*, 2020(23), 2020.

[8] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint*, 2018.

[9] Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. In *ACM on Measurement and Analysis of Computing Systems*, 2017.

[10] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2017.

[11] Jinshuo Dong, Aaron Roth, and Weijie J Su. Gaussian differential privacy. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2019.

[12] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference (TCC)*, 2006.

[13] Cynthia Dwork, Aaron Roth, et al. *The algorithmic foundations of differential privacy.* Now Publishers, 2014.

[14] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to byzantine-robust federated learning. In *USENIX Security Symposium*, 2020.

[15] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *IEEE Annual Symposium on Foundations of Computer Science (SFCS)*, pages 427–438, 1987.

[16] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint*, 2017.

[17] Xiaolan Gu, Ming Li, and Li Xiong. DP-BREM: differentially-private and byzantine-robust federated learning with client momentum. *arXiv preprint*, 2023.

[18] Rachid Guerraoui, Nirupam Gupta, Rafaël Pinot, Sébastien Rouault, and John Stephan. Differential privacy and byzantine resilience in sgd: Do they add up? In *ACM Symposium on Principles of Distributed Computing*, 2021.

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

[20] Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. Learning from history for byzantine robust optimization. In *International Conference on Machine Learning (ICML)*, 2021.

[21] Marcel Keller. MP-SPDZ: A versatile framework for multi-party computation. In *CCS*, 2020.

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

[23] Yann LeCun. The mnist database of handwritten digits. 1998.

[24] Jeffrey Li, Mikhail Khodak, Sebastian Caldas, and Ameet Talwalkar. Differentially private meta-learning. In *International Conference on Learning Representations (ICLR)*, 2020.

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

[26] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *ICLR*, 2018.

[27] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *IEEE Symposium on Security and Privacy (S&P)*, 2019.

[28] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. In *ICML*, 2018.

[29] Sikha Pentyala, Davis Railsback, Ricardo Maia, Rafael Dowsley, David Melanson, Anderson Nascimento, and Martine De Cock. Training differentially private models with secure multiparty computation. *arXiv preprint*, 2022.

[30] Amrita Roy Chowdhury, Chuan Guo, Somesh Jha, and Laurens van der Maaten. EIFFeL: Ensuring integrity for federated learning. In *CCS*, 2022.

[31] Amrita Roy Chowdhury, Chenghong Wang, Xi He, Ashwin Machanavajjhala, and Somesh Jha. Crypte: Crypto-assisted differential privacy on untrusted servers. In *ACM Special Interest Group on Management of Data (SIGMOD)*, 2020.

[32] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[33] Virat Shejwalkar and Amir Houmansadr. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *Network and Distributed System Security (NDSS)*, 2021.

[34] Elaine Shi, TH Hubert Chan, Eleanor Rieffel, Richard Chow, and Dawn Song. Privacy-preserving aggregation of time-series data. In *NDSS*, 2011.

[35] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. A hybrid approach to privacy-preserving federated learning. In *ACM Workshop on Artificial Intelligence and Security*, 2019.

[36] Fayao Wang, Yuanyuan He, Yunchuan Guo, Peizhi Li, and Xinyu Wei. Privacy-preserving robust federated learning with distributed differential privacy. In *IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2022.

[37] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris S Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. In *NeurIPS*, 2020.

[38] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Fall of empires: Breaking byzantine-tolerant sgd by inner

product manipulation. In *Uncertainty in Artificial Intelligence*, 2020.

[39] Runhua Xu, Nathalie Baracaldo, Yi Zhou, Ali Anwar, and Heiko Ludwig. Hybridalpha: An efficient approach for privacy-preserving federated learning. In *ACM Workshop on Artificial Intelligence and Security*, 2019.

[40] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *ICML*, 2018.

[41] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. In *IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2021.

[42] Qinqing Zheng, Shuxiao Chen, Qi Long, and Weijie Su. Federated f-differential privacy. In *AISTATS*, 2021.

[43] Heng Zhu and Qing Ling. Bridging differential privacy and byzantine-robustness via model aggregation. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

[44] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *NeurIPS*, 2019.

A Proof of Theorem 1 (Privacy Analysis)

Proof. Since the added Gaussian noise in (12) has standard deviation $R\sigma$, and the aggregation sensitivity is shown in (14), then the noise multiplier (defined by the ratio between Gaussian noise's standard deviation and the sensitivity) is

$$\sigma_i = \frac{R\sigma}{S_i} = \max\left\{\frac{R\sigma}{2C}, \sigma p_i|\mathcal{D}_i|\right\} = \sigma \cdot \max\left\{\frac{R}{2C}, p_i|\mathcal{D}_i|\right\}$$

Also, due to the client-level sampling (i.e., each client was selected by the server w.p. q) and record-level sampling (i.e., each record was selected by client C_i w.p. p_i), the overall sampling rate is qp_i. Then, by applying the privacy accountant of Gaussian DP [11] (also shown in the Appendix of our full-version paper [17]), DP-BREM satisfies μ_i-GDP with μ_i shown in (16). Finally, by converting μ_i-GDP to (ε_i, δ)-DP, we get (15), which finishes the proof. □

Remark: privacy accountant in practice. Eq. (15) provides the formula of δ when ε_i is given and μ_i is computed from (16). In practice, however, we need to compute the value of privacy budget ε_i with a fixed δ, where δ is conventionally set to be less than $1/n$. In our experiments, we utilize the computation tool[3] in [7] to solve ε_i from (15). For the value of σ_i in (16), we usually have $p_i|\mathcal{D}_i| > \frac{R}{2C}$ in practice, then $\sigma_i = \sigma p_i|\mathcal{D}_i|$. In this case, the clipping bounds R and C are just hyperparameters that may affect the utility of the algorithm, but has no influence on the privacy analysis.

[3] https://github.com/woodyx218/Deep-Learning-with-GDP-Pytorch

B Proof of Theorem 3 (Convergence Rate)

Proof. The proof of DP-BREM's convergence rate is based on the result of DP-BREM's aggregation error shown in Theorem 2, and LFH's convergence rate derived from LFH's aggregation error. Note that all differences between DP-BREM and LFH, including per-record clipping and the DP noise, are reflected by the aggregation error. Comparing with the aggregation error of $O(\rho^2|\mathcal{B}|/n)$ (ignoring constants and higher order terms) in LFH [20, Lemma 9], our aggregation error shown in (19) replaces the term $|\mathcal{B}|$ by $|\mathcal{B}| + \sqrt{d}\sigma/q$, which means a slower convergence due to DP noise. Then, following the result in [20, Theorem VI] and its informal version in (5), we get the convergence rate of our algorithm as in (20). Note that our aggregation utilizes a client-level sampling rate q, i.e., approximate nq clients participate in the aggregation for one iteration. We need to replace the term of $\frac{1}{n}$ in (5) by $\frac{1}{nq}$ in (20). □

C Detailed Steps of DP-BREM$^+$ in Figure 2

① Proof and Shares Generation: $\boldsymbol{z}_i, \mathsf{Valid}(\cdot) \to [\boldsymbol{z}_i]_j, [\pi_i]_j \ (\forall j \neq i)$. For generating the proof, client C_i first evaluates the circuit $\mathsf{Valid}(\cdot)$ on its private input $\boldsymbol{z}_i$ to obtain the value of every wire in the arithmetic circuit corresponding to the computation of $\mathsf{Valid}(\boldsymbol{z}_i)$, then uses these wire values to generate the proof π_i (refer to [10, 30] for the detailed format). Then, client C_i splits the private input $\boldsymbol{z}_i$ and proof π_i to generate shares $[\boldsymbol{z}_i]_j$ and $[\pi_i]_j$ $(\forall j \neq i)$, and send them to other clients $\{\mathsf{C}_j\}_{\forall j \neq i}$ via Shamir's secret sharing.

② Proof Summary Computation: $[\boldsymbol{z}_i]_j, [\pi_i]_j \ (\forall j \neq i) \to [\sigma_i]_j \ (\forall j \neq i)$. Each client except C_i first verifies the validity of the received secret shares via verifiable secret shares [15], and then locally constructs the shares of every wire in $\mathsf{Valid}(\boldsymbol{z}_i)$ via affine operations on the shares $[\boldsymbol{z}_i]_j$ and$[\pi_i]_j$ to get the shares of proof summary $[\sigma_i]_j$ (refer to [30] for the detailed format), which will be sent to the server.

③ Proof Summary Verification: $[\sigma_i]_j \ (\forall j \neq i) \to \mathsf{Valid}(\boldsymbol{z}_i)$. After receiving shares of proof summary $[\sigma_i]_j (\forall j \neq i)$ from clients $\{\mathsf{C}_j\}_{\forall j \neq i}$, the server recovers the value of σ_i via robust reconstruction, which is resilient to incorrect shares submitted by the malicious clients, and then checks the values in proof summaries. Finally, the validation result $\mathsf{Valid}(\boldsymbol{z}_i) = 1$ if and only if σ_i has the correct value.

④ Random Numbers Generation: $l, d \to \{([u_k]_j, [v_k]_j)\}_{k=1}^{\lceil d/2 \rceil} \ (\forall j)$. In this step, clients jointly generate the shares of $\lceil d/2 \rceil$-pairs of random numbers $\{(u_k, v_k)\}_{k=1}^{\lceil d/2 \rceil}$, where all of them are i.i.d. from uniform distribution in the range $[0, 1]$. Denote l as the fractional precision of the power 2 ring representation of real numbers. To obtain the share of one random number u, each client C_i $(\forall i)$ generates l random bits in the binary filed $\mathbb{F}_2$,

denoted by a binary vector $\boldsymbol{b}_i$ with length l, then generate and distributes the shares $[\boldsymbol{b}_i]_j$ to other clients (via Shamir's secret sharing). After receiving all shares from other clients, each client C_j $(\forall j)$ locally adds these shares to get $[\boldsymbol{b}]_j = [\sum_i \boldsymbol{b}_i]_j \in \mathbb{F}_2^l$, where vector $\boldsymbol{b} \in \mathbb{F}_2^l$ is actually the bitwise XOR of vectors $\{\boldsymbol{b}_i\}_{\forall i}$ because the computation is implemented in the binary field $\mathbb{F}_2^l$. We define the binary vector $\boldsymbol{b}$ as the binary representation of the fractional part of $u \in [0,1]$. Note that the Shamir's secret sharing scheme of Phase 1 is implemented in a finite filed $\mathbb{F}_{2^K}$, where $K > l$. Therefore, the client C_j can locally compute the arithmetic share $[u]_j \in \mathbb{F}_{2^K}$ from the share of binary representation $[\boldsymbol{b}]_j \in \mathbb{F}_2^l$. Since all possible discrete values with power 2 ring representation evenly span the range $[0,1]$, the generated random real number u is uniformly distributed in $[0,1]$.

⑤ Transformation to Gaussian Distribution: $\{([u_k]_j, [v_k]_j)\}_{k=1}^{\lceil d/2 \rceil}$ $(\forall j) \rightarrow [\boldsymbol{\xi}]_j$ $(\forall j)$. For each pair of (u_k, v_k), clients can jointly compute a secret sharing of $a_k = \sqrt{-2\ln(u_k)} \cdot \cos(2\pi v_k)$ and of $b_k = \sqrt{-2\ln(u_k)} \cdot \sin(2\pi v_k)$ by utilizing Secure Multiparty Computation (MPC) protocols [21] that guarantees security (i.e., privacy and integrity) with malicious minority. According to Box and Muller Transformation [6], a_k and b_k are i.i.d. random variables from the Gaussian distribution with mean 0 and variance 1. Then, by locally implementing secure multiplication with a constant (i.e., $R\sigma$), a_k and b_k are i.i.d random numbers following a Gaussian distribution with the desired standard deviation of $R\sigma$. Finally, by concatenating shares of d numbers in $\{(a_k, b_k)\}_{k=1}^{\lceil d/2 \rceil}$, clients obtains the shares of random vector $\boldsymbol{\xi}$ with length d from Gaussian distribution $\mathcal{N}(0, R^2\sigma^2\mathbf{I}_d)$.

⑥ Shares Aggregation: $\{[\boldsymbol{z}_i]_j\}_{i \in I_{\mathsf{Valid}}}, [\boldsymbol{\xi}]_j$ $(\forall j) \rightarrow [\sum_{i \in I_{\mathsf{Valid}}} \boldsymbol{z}_i + \boldsymbol{\xi}]_j$ $(\forall j)$. Due to the linearity of Shamir's secret sharing scheme, each client C_j can locally compute the share of the noisy aggregate by adding the shares of all valid inputs and the share of Gaussian noise: $[\sum_{i \in I_{\mathsf{Valid}}} \boldsymbol{z}_i + \boldsymbol{\xi}]_j = \sum_{i \in I_{\mathsf{Valid}}} [\boldsymbol{z}_i]_j + [\boldsymbol{\xi}]_j$, and sends that share to the server.

⑦ Noisy Aggregate Reconstruction: $[\sum_{i \in I_{\mathsf{Valid}}} \boldsymbol{z}_i + \boldsymbol{\xi}]_j$ $(\forall j) \rightarrow \sum_{i \in I_{\mathsf{Valid}}} \boldsymbol{z}_i + \boldsymbol{\xi}$. After receiving all shares of the noisy aggregate, the server recovers it using robust reconstruction.

D Experimental Setup

FL Implementation. Due to limited resources, we simulate the distributed training of FL by running a single machine sequentially for clients and the server. The real-world implementation of FL is out of the scope of this paper.

Datasets (non-IID) and Model Architecture. We use three datasets for our experiments: MNIST [23] CIFAR-10 [22] and FEMNIST [8], where the number of total clients is $n = 100$ for the former two datasets, and $n = 400$ for FEMNIST dataset. Note that the MNIST and CIFAR-10 datasets only have 10 classes, while the FEMNIST dataset has 62 classes (including 10 digits, 26 lowercase letters, and 26 uppercase letters). For the MNIST dataset, we use the CNN model from PyTorch example[4]. For the CIFAR-10 dataset, we use the CNN model from the TensorFlow tutorial[5], like the previous works [26, 42]. To simulate the heterogeneous data distributions, we make non-i.i.d. partitions of the datasets, which is a similar setup as [42] and is described below. For the FEMNIST dataset, we use a CNN model with 2 convolution layers and 2 fully connected layers.

1) Non-IID MNIST: The MNIST dataset contains 60,000 training images and 10,000 testing images of 10 classes. There are 100 clients, each holds 600 training images. We sort the training data by digit label and evenly divide it into 400 shards. Each client is assigned four random shards of the data, so that most of the clients have examples of three or four digits.

2) Non-IID CIFAR-10: The CIFAR-10 dataset contains 50,000 training images and 10,000 test images of 10 classes. There are 100 clients, each holds 500 training images. We sample the training images for each client using a Dirichlet distribution with hyperparameter 0.9.

3) Non-IID FEMNIST: The FEMNIST dataset is pre-partitioned based on the writer of the characters, simulating a non-IID scenario. Each client's local dataset consists of samples written by individual users, introducing variability in handwriting styles. We use the TensorFlow-Federated API[6] to load the first 400 partitions, representing data from 400 clients. Unlike the MNIST dataset, which includes only digits, FEMNIST includes both digits and uppercase and lowercase letters, spanning 62 classes (10 digits + 52 letters).

Byzantine Attacks. We consider four different Byzantine attacks in our experiments.

1) ALIE ("a little is enough") [3]. The attacker uses the empirical variance (estimated from the data of corrupted clients) to determine the perturbation range, in which the attack can deviate from the mean without being detected or filtered out.

2) IPM (inner-product manipulation) [38]. The attacker manipulates the submitted gradient to be the negative direction of the mean of other honest clients' gradients, thus the negative inner-product of the true gradient and the aggregation prevents the descent of the loss. Note that the original IPM attack assumes the *omniscient* attacker (i.e., knows the data/gradient of all other clients), which is contradicted to our assumption that the attacker only has access to the data of the corrupted clients (otherwise, the privacy is already leaked and no need to provide DP). Thus, in the experiments, we use the data of corrupted clients to estimate the aggregated gradient of honest clients, and then manipulate the inner-product (i.e., non-omniscient attack).

3) LF (label-flipping). The attacker modifies the labels of

[4] https://github.com/pytorch/opacus

[5] https://www.tensorflow.org/tutorials/images/cnn

[6] https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets/emnist/load_data

all examples of corrupted clients' data and computes a new gradient, then uses a gradient replacement strategy (similar to [2]) to enhance the impact on the global model. Specifically, the attacker computes a benign gradient $\mathbf{g}_{\text{benign}}$ with non-flipped labels and also a bad gradient $\mathbf{g}_{\text{bad}}$ with flipped labels. Finally, each malicious client submits the difference $\mathbf{g}_{\text{bad}} - \mathbf{g}_{\text{benign}}$ to the server to achieve the goal that the aggregated global gradient (averaged over all clients) is close to $\mathbf{g}_{\text{bad}}$.

4) MTB ("manipulating-the-Byzantine") [33]. The attacker computes a benign reference aggregate using some benign data samples obtained from corrupted clients, then computes a malicious perturbation vector, and an optimized scaling factor to get the malicious update with the goal of evading detection by robust aggregation algorithms. The optimization of the scaling factor can be tailored or agnostic to the aggregator. Considering our scheme and the baselines do not detect malicious clients, we use the agnostic setting (including min-max and min-sum) for simplicity because tailoring MTB attack to all defense aggregators is nontrivial. In our experiments, we implement the min-max attack since it has a larger impact on the global model.

Byzantine Defenses with DP. We compare the performance of our approaches with the following five competitors against Byzantine attacks. All of them satisfy record-level DP via record-level clipping and DP noise added to the local gradient/momentum. Note that privacy budget ε in Theorem 1 is the same for different clients because clients have the same size of local datasets $|\mathcal{D}_i|$ and same record-level sampling rate (i.e., same $|\mathcal{D}_i|$ and p_i for different clients C_i).

1) DP-FedSGD. Note that the original DP-FedSGD in [26] clips the client gradient to achieve *client-level* DP. For a fair comparison, we also implement record-level gradient clipping on top of the original DP-FedSGD to guarantee record-level DP. Though DP-FedSGD is not designed for robustness, its client-level clipping can restrict malicious clients' capability, thus providing some level of Byzantine robustness. We take this as a baseline to illustrate that client-level clipping can provide some level of robustness, but may not be enough to defend against strong attackers (either advanced attack strategy or a larger number of malicious clients).

2) DP-CM. As a baseline that adds DP to median-based robust aggregators (discussed in Section 3.2), we implement the Byzantine-robust aggregator Coordinate-wise Median (CM) [40] with DP noise added to the median result. Note that only DP-CM uses median-based aggregation, while other methods use average-based aggregation. As discussed in Section 3.1 and Example 1, the median-based aggregation has large sensitivity and poor privacy-utility tradeoff.

3) DDP-RP [36]. By leveraging encryption techniques, DDP-RP guarantees Distributed DP with secure aggregation. It allows clients to add smaller noise in the local gradient than the Local DP, with the knowledge of the lower bound of trusted clients, thus providing enhanced privacy-utility tradeoff than local DP protocols. To guarantee Byzantine robustness, DDP-RP uses range-proof (RP) technologies to securely verify whether the local model/gradient weights are in a (predefined) bounded range.

4) DP-RSA [43]. It replaces the *value* aggregation to *sign* aggregation, which provides robustness because each client has limited impact on the aggregation. The DP noise is added to the local gradient before the sign operation.

5) DP-LFH. The baseline (Section 3.2) directly combines DP-SGD based momentum with LFH. Each client adds DP noise to the local gradient, and then computes the local momentum to be aggregated with centered clipping by the server.

E Parameters in Experiments

Basic Parameters.

- Total number of iterations T: 1000 for MNIST and FEMNIST; 2000 for CIFAR-10.
- Learning rate η_t: For MNIST and FEMNIST datasets, η_t is linearly reduced from 0.1 to 0.01 w.r.t. iterations. For CIFAR-10 dataset, η_t is linearly reduced from 0.05 to 0.0025 w.r.t. iterations.

DP-related Parameters.

- Record-level sampling rate p_i: 0.05 for all i on MNIST and CIFAR-10; 0.1 for all i on FEMNIST (because each client has fewer data records).
- Client-level sampling rate q: the default value is 1. We evaluate the influence of q (from 0.2 to 1) on the accuracy in Table 5.
- Record-level clipping bound R: linearly reduced from R_0 to $0.3R_0$ w.r.t. iterations. Note that in Figure 5, the different value of R in x-axis is the value of the above R_0. For MNIST and FEMNIST, we set $R_0 = 10$ by default. For CIFAR-10, we set $R_0 = 20$ by default, but $R_0 = 15$ only for the case of $\varepsilon = 2$ in Figure 4.
- Privacy parameter δ in DP: 10^{-6}
- Noise multiplier σ: For MNIST (with $T = 1000$ and each client has $|\mathcal{D}|_i = 60000/100 = 600$ examples), $\sigma \in \{0.15, 0.06, 0.029, 0\}$ for $\varepsilon \in \{1, 3, 8, \text{inf}\}$. For CIFAR-10 (with $T = 2000$ and each client has $|\mathcal{D}|_i = 50000/100 = 500$ examples), $\sigma \in \{0.14, 0.077, 0.042, 0\}$ for $\varepsilon \in \{2, 4, 9, \text{inf}\}$. For FEMNIST (with $T = 1000$ and each client has around 300 examples), $\sigma \in \{0.16, 0.09, 0.047, 0\}$ for $\varepsilon \in \{2, 4, 9, \text{inf}\}$.

Robustness-related Parameters.

- Client-level clipping bound C (only for DP-BREM and DP-LFH): linearly reduced from C_0 to $0.3C_0$ w.r.t. iterations, where $C_0 = 1$ for MNIST, $C_0 = 5$ for CIFAR-10, and $C_0 = 2$ for FEMNIST.
- Momentum parameter $\beta = 0.9$ for all three datasets.