Practical Differentially Private and Byzantine-resilient Federated Learning

ZIHANG XIANG, King Abdullah University of Science and Technology, Saudi Arabia
TIANHAO WANG, University of Virginia, USA
WANYU LIN, The Hong Kong Polytechnic University, China
DI WANG, King Abdullah University of Science and Technology, Saudi Arabia

Privacy and Byzantine resilience are two indispensable requirements for a federated learning (FL) system. Although there have been extensive studies on privacy and Byzantine security in their own track, solutions that consider both remain sparse. This is due to difficulties in reconciling privacy-preserving and Byzantine-resilient algorithms.

In this work, we propose a solution to such a two-fold issue. We use our version of differentially private stochastic gradient descent (DP-SGD) algorithm to preserve privacy and then apply our Byzantine-resilient algorithms. We note that while existing works follow this general approach, an in-depth analysis on the interplay between DP and Byzantine resilience has been ignored, leading to unsatisfactory performance. Specifically, for the random noise introduced by DP, previous works strive to reduce its impact on the Byzantine aggregation. In contrast, we leverage the random noise to construct an aggregation that effectively rejects many existing Byzantine attacks.

We provide both theoretical proof and empirical experiments to show our protocol is effective: retaining high accuracy while preserving the DP guarantee and Byzantine resilience. Compared with the previous work, our protocol 1) achieves significantly higher accuracy even in a high privacy regime; 2) works well even when up to 90% distributive workers are Byzantine.

CCS Concepts: • **Computing methodologies → Cooperation and coordination**; **Distributed artificial intelligence**.

Additional Key Words and Phrases: Federated Learning, Byzantine Security, Differential Privacy

ACM Reference Format:
Zihang Xiang, Tianhao Wang, Wanyu Lin, and Di Wang. 2023. Practical Differentially Private and Byzantine-resilient Federated Learning. *Proc. ACM Manag. Data* 1, 2, Article 119 (June 2023), 26 pages. https://doi.org/10.1145/3589264

1 INTRODUCTION

Federated Learning (FL), a learning framework for preserving the privacy of distributed data [38], has thrived during the past few years. To comply with the privacy regulations such as General Data Protection Regulation (GDPR) [25], variants of FL frameworks have been widely studied, and recently adopted in industry, such as Apple's "FE&T" [48], Google's Gboard [24], and Alibaba's

Authors' addresses: Zihang Xiang, zihang.xiang@kaust.edu.sa, King Abdullah University of Science and Technology, P.O.Box 4700, Thuwal, Saudi Arabia, 23955-6900; Tianhao Wang, tianhao@virginia.edu, University of Virginia, Charlottesville, VA, USA; Wanyu Lin, wanylin@comp.polyu.edu.hk, The Hong Kong Polytechnic University, , Hong Kong, China; Di Wang, di.wang@kaust.edu.sa, King Abdullah University of Science and Technology, P.O.Box 4700, Thuwal, Saudi Arabia, 23955-6900.

2836-6573/2023/6-ART119
https://doi.org/10.1145/3589264

FederatedScope [70]. In an FL system, there are several local workers, each holding a dataset for local training, and a server aggregating gradient vectors from workers for global model updates.

However, current FL frameworks that seemingly can protect privacy (because the original data never leaves the local workers) are in fact vulnerable to various privacy attacks, such as membership inference attacks [55] (tries to infer whether some data samples are used in training) and model inversion attacks [77] ("reverse-engineer" sensitive data samples through gradients). These vulnerabilities drive the community to design methods that can further preserve the privacy of data held by workers. Among the privacy-enhancing techniques [22, 23, 52, 60], differential privacy (DP) [20] is a rigorous mathematical scheme that allows for rich statistical and machine learning analysis and is becoming the *de facto* notion for data privacy. Many methods have been proposed to tackle the problems of integrating DP into machine learning/deep learning from different perspectives [1, 12, 34, 46, 47, 56, 61, 62, 64]. More recently, DP has been adopted in the FL setting [2, 26, 59, 66, 73].

Besides privacy risks, FL systems are also vulnerable to adversarial manipulations from Byzantine workers, which could be fake workers injected by an attacker or genuine workers compromised by an attacker. Specifically, in a Byzantine attack, the adversary intends to sabotage the collective efforts by sending false information, such as contrived Byzantine gradients [6, 69]. To mitigate this issue, recent work proposes Byzantine-resilient machine learning approaches, such as diagnosing and rejecting gradients with abnormal features [9, 11, 15, 49, 51].

Tremendous progress on privacy and Byzantine resilience have been seen in their own track. However, all of them are not applicable to the more practical scenario where a privacy attacker is also Byzantine (a double-role attacker). Being aware of that, some recent work started to focus on such an issue yet provided unsatisfactory answers. Some of them fail to ensure both DP and Byzantine resilience simultaneously [30], while some other work tries to explore optimal parameter setups but still end up with a much-limited solution [29]. We also notice that some work [76] tries to combine existing variants in both tracks to side-step the seeming incompatibility of DP and Byzantine security, however, their resistance is retained only when the privacy level is low and the portion of Byzantine clients is small.

Contributions: We observe that previous solutions fail to give a satisfactory answer for a common reason: neither the DP algorithm nor the Byzantine defending method is designed against both risks simultaneously. Our contribution is how we start from a co-design to form a DP and Byzantine-resilient solution, proving the synergy of combined DP and Byzantine resilience.

1) Co-design: Since random noise introduced by DP impairs the effectiveness of existing Byzantine-resilient aggregation rules, previous works tend to limit the impact of randomness by increasing the data batch size [29, 30]. In contrast, we leverage random noise to aid Byzantine aggregation: we use small batch size and accordingly construct our first-stage aggregation which effectively rejects many existing attacks.

Moreover, previous works continue to use the standard DP-SGD [1] to bound gradient sensitivity by clipping, which involves manually tuning the clipping parameter. In contrast, we ensure bounded sensitivity by normalizing, and it enables our second-stage aggregation, which provides a final sound filtering.

2) Cherry on top: We are the first to find out that bounding gradient sensitivity by normalizing is more suitable for DP learning although normalizing itself is not new[1]. Specifically, we analyze its theoretical implication and also leverage it to construct a learning protocol that saves quadratic

[1]Some concurrent work [10, 18, 19, 71] on DP learning use an operation similar to normalizing with different considerations.

efforts[2] in hyper-parameter-tuning for DP learning, where a smaller amount of queries on gradient computation is more favored.

In the final evaluation, we conduct experiments to first show our contribution to DP solution and Byzantine aggregation in their own track. Then, for the core aim to preserve both privacy and Byzantine security, our experimental results show that in addition to having the DP guarantee, our protocol also remains robust against strong attackers when there are up to 90% distributive workers are Byzantine. We have released our code in supplementary material [3].

2 BACKGROUND AND PRELIMINARIES

2.1 Federated Learning

In a typical setting of machine learning, we have a training dataset $\{x_1, x_2, \cdots, x_m\}$ where each x_i contains a feature vector and a label, and we also have a loss function f. We aim to find the best model parameters w from a parameter space Θ which minimizes the following function through stochastic optimization:

$$\min_{w\in\Theta} F(w) = \mathbb{E}_{x\sim\mathcal{P}}\left[f(x; w)\right], \tag{1}$$

In FL, suppose there are n workers and the i-th worker has local and private data D_i, then training in FL happens in a distributed manner. Specifically, in the t-th iteration, we have:

1) *Model broadcasting:* The server broadcasts the current model parameters w^{t-1} to all workers.

2) *Local gradient computation:* After receiving the model sent by the server, each worker will use his/her private data and the model w^{t-1} to compute his/her gradient vector $g_i^t = \nabla f(D_i; w^{t-1}) := \frac{1}{|D_i|}\sum_{x\in D_i} \nabla f(x; w^{t-1})$. Note that workers can also compute their gradients and update their model N times locally, and report the difference between the model they get locally and the last model they receive from the server. In our framework, we take $N = 1$ and this is due to the constraints of DP-SGD protocol which will be discussed later. Extending to the cases where $N > 1$ will be left for future study.

3) *Gradient aggregation and model update:* The server will perform an aggregation step (denoted by function **Aggregation**) on the gradient vectors reported by workers and use the result $g^t = \textbf{Aggregation}(g_1^t, g_2^2, \cdots, g_n^t)$ to update the model by $w^t = w^{t-1} - \eta g^t$, where η is the learning rate. Note that there are variants of aggregation strategies, e.g., $g^t = \sum_i \frac{|D_i|}{\sum_j |D_j|} g_i^t$ [44].

2.2 Differential Privacy for Deep Learning

Definition 1 (Differential Privacy [20]). *Given a data universe $\mathcal{X}$, we say that two datasets $D, D' \subseteq \mathcal{X}$ are neighbors if they differ by only one data sample, which is denoted as $D \sim D'$. A randomized algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private if for all neighboring datasets D, D' and for all events S in the output space of $\mathcal{A}$, we have* $\Pr(\mathcal{A}(D) \in S) \leq e^{\epsilon}\Pr(\mathcal{A}(D') \in S) + \delta$.

An (ϵ, δ)-DP mechanism typically adds calibrated noise to the output of a query. In this paper we mainly use the Gaussian mechanism to guarantee (ϵ, δ)-DP:

Definition 2 (Gaussian Mechanism). *Given any function $q : \mathcal{X}^n \to \mathbb{R}^d$, the Gaussian mechanism is defined as $q(D) + \xi$ where $\xi \sim \mathcal{N}(0, \frac{2\Delta_2^2(q)\log(1.25/\delta)}{\epsilon^2} I_d)$, where where $\Delta_2(q)$ is the ℓ_2-sensitivity of the function q,* i.e., $\Delta_2(q) = \sup_{D\sim D'} ||q(D) - q(D')||_2$. *Gaussian mechanism satisfies (ϵ, δ)-DP when $\epsilon \leq 1$.*

[2]Instead of tuning learning rate η and clip threshold C for different ϵ, our approach only needs to turn η for any instance of ϵ.

[3]https://github.com/zihangxiang/-Practical-Differentially-Private-and-Byzantine-resilient-Federated.git.

Another notable property of DP is that DP is closed under postprocessing, *i.e.,* if we post-process the output of an (ϵ, δ)-DP algorithm, then the whole procedure will still be (ϵ, δ)-DP.

DP in deep learning: Differentially Private SGD (DP-SGD) is a widely used method in machine learning to ensure DP [1, 7, 57]. It modifies the SGD-based methods by adding Gaussian noise to perturb the (stochastic) gradient in each iteration of the training, *i.e.,* in the centralized setting, during the t-th iteration DP-SGD will compute a noisy gradient as follows:

$$g^t = \frac{1}{|B|}\Big(\sum_{x_i \in B} \hat{g}_i^t + \mathcal{N}\left(0, \sigma^2 C^2 I\right)\Big), \tag{2}$$

where B is a subsampled data batch used to compute the gradient, σ is the noise multiplier, $\hat{g}_i^t$ is the gradient vector computed by feeding one data sample to w^{t-1} which is the current model before the t-th iteration, and g^t is the (noisy) gradient used to update the model. The main reason here we use $\hat{g}_i^t$ instead of the original gradient vector is that we wish to make the term $\sum \hat{g}_i^t$ have bounded ℓ_2-sensitivity so that we can use the Gaussian mechanism to ensure DP. The most commonly used approach to get a $\hat{g}_i^t$ is clipping the gradient: $\hat{g}_i^t = \nabla f(x_i; w^{t-1}) \min\{1, \frac{C}{\|\nabla f(x_i; w^{t-1})\|_2}\}$ *i.e.*, each gradient vector is clipped by C (scale those whose ℓ_2-norm is greater than C to be C exactly and leave the rest untouched). Since the ℓ_2-sensitivity of $\sum \hat{g}_i^t$ is bounded by C, after the clipping, we can add Gaussian noise to ensure DP. To prove the (ϵ, δ)-DP property of DP-SGD, there has been a line of research [4, 28, 45, 65, 74, 78]. We use *TensorFlow Privacy* [58] to search for noise multiplier given ϵ and δ. Other than the DP-SGD framework, we note that there exists others work [16] tackling the privacy issue in FL by extending the PATE framework [46]. However, their work does not apply to the case where Byzantine workers exist.

2.3 Byzantine Attacks in FL

The FL protocol is vulnerable to Byzantine attacks, as each worker can report a malicious gradient vector to deteriorate the model performance [6, 69] and even bias the model in a specific way [13, 41]. Several Byzantine-robust approaches are proposed to tackle different attacks [9, 11, 15, 31, 49, 51, 72]. And there are also new advanced Byzantine attacks that try to bypass such defenses [5, 6, 14, 15, 35, 53, 69].

A Taxonomy: To understand the features of those Byzantine attacks, we summarise three dimensions to capture their properties.

1) Objective: There are generally 2 types of objectives: *a*) Denial-of-service attack (also called *untargeted* or convergence prevention) [6] that tries to destroy the training process and makes the model unusable. *b*) Backdoor attack [41] that tries to poison the training data to make the model predict intended results on inputs with specific triggers while still behaving normally on other inputs.

2) Capability: Existing attacks assume different attackers' power. Some work assumes that the Byzantine attacker is *omniscient*, *i.e.*, the attacker knows what the honest workers send to the server [21, 69], while others assume the attacker does not know the honest workers' data [6]. In both cases, the attacker knows the aggregation rules.

3) Specificity of targeted defenses: a) Some attacks are defense-specific, *i.e.*, and they are tailored for specific defense methods [6, 69]. However, it is unclear whether such attacks still remain effective against other or new defense protocols. *b*) Some other attacks are universal: these attacks are either defense-agnostic [11, 76] or have a meta-method [21] that can be instantiated to attack almost all existing defense strategies upon knowing the defense rules.

Instantiating Existing Attacks: We briefly introduce some existing Byzantine attacks which have been considered in previous work. They can be categorized by the above 3 dimensions:

1) *Gaussian attack* [51, 76] uploads pure Gaussian noise trying to hurt utility. 2) *Label-flipping attack* [11, 21] first poisons the local dataset by flipping the original label I to $H-1-I$ (H is the total number of classes, $I = 0, 1, \cdots, H-1$ is the label) and then follows the FL protocol. Note that we can also adopt other ways to perform the label flipping (such as randomly flipping to a different label). In fact, the way to flip the label does not matter as long as it tries to reduce the overall accuracy. 3) *Optimized Local Model Poisoning attack* [21], the state-of-the-art Byzantine attack method that can be accordingly instantiated for a specific Byzantine defense method in an adversarial way given the Byzantine defense protocol first.

For ease of discussion, we use "Byzantine attacker" to refer to a master attacker which can inject several fake workers into the system and control all of them. Hence, in this sense, attackers who can send malicious uploads to the server can possibly collude.

3 PROBLEMS AND EXISTING SOLUTIONS

3.1 Problem Setting

Attacker: FL is indeed exposed to threats from two kinds of attackers: *privacy attacker* and *Byzantine attacker*. Note that we will not discuss specific *privacy attack*s and only focus on *Byzantine attack*s for the following reasons.

Protecting privacy with DP: From an information-theoretical view, DP guarantees privacy in the worst case by limiting the maximum amount of information that any privacy attacker can extract even with side information and unlimited computational resource [20]. It has also been shown that by tuning ϵ small enough [50], adopting DP effectively rejects strong privacy attackers [55]. Following the privacy settings as in the previous work [29, 30, 76], we focus on the item-level privacy for each worker's dataset; in this case, the gradient needs to be privatized (by adding random noise) before being uploaded to the server.

Focusing on Byzantine-resilience: We are not interested in tuning ϵ to test the algorithm's strength to defend against privacy attacks. Since our algorithms are guaranteed to be (ϵ, δ)-DP theoretically, *we only need to focus on Byzantine attacks*.

In other words, we treat DP as one of the basic properties that an FL system should possess. In fact, leveraging our tailored DP protocol to defend against Byzantine attacks is one of our novelties.

Byzantine attacker:

We specify the Byzantine attacker according to the taxonomy mentioned above:

- *Objective:* Our Byzantine attacker is trying to perform Denial-of-Service (DoS) attacks.
- *Capability:* Our Byzantine attacker is omniscient; it knows all the data held by honest workers, information sent by honest workers, and the aggregation rules. We assume such capability in our framework to show that our protocol still works even when facing such a strong adversary.
- *Specificity of targeted defense:* We consider a stronger version of universal attacks. Our protocol will be made public and the attacker is allowed to instantiate his attack on our protocol.

In other words, we are interested in defending the stronger *untargeted* attacks and we leave backdoor attacks as future work.

Defender: Privacy is guaranteed through DP, and each user applies DP to protect its local data. To defend against Byzantine attacks, the server needs to design new aggregation rules such that false gradients from Byzantine workers are excluded. Here we assume the server possesses *a small amount* of labeled data samples which are kept secret from attackers. Let $\mathbb{X}$ be the the data space and $\mathbb{Y}$ be the label space in a classification task. We do not require the server to have direct access to the local-hold data, instead, we assume the data space $\mathbb{X}_{aux}$ from which the auxiliary data is

Methods	Privacy	$> 50\%$-Resilience
Krum [9]	✗	✗
Coordinate-wise Median [72]	✗	✗
Trimmed Mean [72]	✗	✗
Bulyan [31]	✗	✗
Zhu et al. [75]	✗	✗
FLTrust [11]	✗	✓
Rachid et al. [29]	✓	✗
Xu et al. [42]	✓	✗
Heng et al. [76]	✓	✗
Our work	✓	✓

Table 1. Comparison with previous work. For privacy, ✓ means the method is guaranteed to be DP while the ✗ means the converse; for $> 50\%$-Resilience, ✓ means the method remains resilient when the number of Byzantine attackers exceeds half of the total while ✗ means the method is no longer effective under such majority Byzantine attack.

sampled is the same as that of local-hold data. Notably, this additional assumption is reasonable in real applications as getting such a tiny amount of data is relatively cheap, and there is work on DP learning [8, 27, 32, 39, 46, 47, 63, 79] and Byzantine-resilient learning [11, 51] making this assumption. In our experiments, we simulate obtaining such data by randomly drawing $2C$ sampling from a validation set where C is equal to the number of classes of that dataset (e.g., for the MNIST dataset, $C = 10$, thus 20 auxiliary data samples will suffice). It is also helpful to consider whether the server-own data and local-hold data follow the same label distribution $\mathbb{Y}$ (with a slight abuse of notation), accordingly, we also conduct experiments for *i.i.d.* case (distributions on $\mathbb{Y}$ are the same for both) and *non-i.i.d.* case (distributions on $\mathbb{Y}$ are different).

We also assume the server knows the truth that at least γn workers are honest among all n workers. It is notable that in the paper we do not need to place any restriction on γ, while previous work [21, 72] needs to assume that $\gamma > 0.5$.

In conclusion, each local worker adopts DP to protect privacy, hence, for one worker, the privacy attacker can be anyone (including the server) except itself. The Byzantine attacker (disguised as some local workers) contrives its upload and tries to destroy the training to make the model has low utility. As a Byzantine defender, the server is honest-but-curious, *i.e.*, it wants to have a model with good utility, thus it follows the protocol. However, it may try to infer sensitive information from the uploads sent by local workers.

3.2 Existing Solutions

In Table 1, we summarize the previous methods on privacy and Byzantine security issues in FL. As we can see from the table, our method can defend against more than 50% Byzantine workers while also achieving DP. In the following, we will provide more details and discuss the limitations of previous methods.

1) The following lines of work only focus on defending against Byzantine attacks: We first recall some existing solutions to Byzantine resilience. There is a line of work focusing on designing robust aggregation rules for corrupted gradients [9, 11, 15, 31, 31, 49, 51, 72] including Krum [9], RFA [49], coordinate-wise median [72], and Trimmed Mean [72]. We summarise the detail of these four methods in supp. material. for interested readers. In general, the first two methods involve computing pair-wise distance between vectors while the latter two concentrate on robust aggregation on each coordinate of vectors. Due to their intrinsic limitations, all of these methods are

only applicable when the majority of workers (> 50%) are honest. Recently, Zhu et al. [75] propose improvements to existing Byzantine-resilient methods to provide a certain level of resilience.

Recently, there is work showing that it is possible to leverage the knowledge of clean gradient computed from non-private auxiliary data [11, 51] to help the aggregation. Common behavior in such methods is that the server weights each uploaded gradient according to their similarity compared to the gradient computed by server-own auxiliary data.

All of the above methods have no DP guarantee because they are designed to defend against Byzantine attacks ignoring privacy attacks. Hence, local datasets' privacy is at risk.

2) The following lines of work try to apply Byzantine defending methods on top of DP output: Some recent work investigates the problem of maintaining both DP and Byzantine resilience in federated learning [29, 30]. Specifically, they study methods of directly combing DP-SGD with some existing aggregation methods such as Krum, *i.e.,* by applying the aggregation on the noisy gradient.

Difficulties in reconciling DP and Byzantine-resilient protocols: Previous work shows that for these methods, to become Byzantine-resilient when DP noise is injected, the fraction of Byzantine workers must decrease with $\sqrt{d}$, where d is the size of the model if the batch size is not large enough [29, 30]. This means such methods achieve good performance only when the number of Byzantine workers is small. Their experiments also verify that this type of method is unsatisfactory by showing that the testing accuracy deteriorates significantly even for a small model learned on a simple dataset [30]. Another line of work is based on robust stochastic model aggregation on the local workers' gradients. In these methods, the gradients of each worker are compressed into signs (1 for non-negative and -1 for negative) with DP [42, 76], however, all of them remain effective only under < 50% Byzantine attack.

4 OUR APPROACH

Observations and lessons learned : Existing solutions apply off-the-shelf Byzantine methods on top of noisy gradient to explore optimal parameter setups. They fail to reach satisfactory performance because neither the Byzantine defending protocol nor the DP protocol is designed for the scenario where privacy and Byzantine resilience are both needed.

4.1 Method Overview

We first re-design the DP protocol, there are two notable properties we enforce in our DP protocol: 1) small training batch size for each worker; 2) use normalizing instead of clipping to bound per-example gradient norm. *We are not considering privacy and Byzantine resilience in a separate manner*. We design our first-stage aggregation based on the first property and design a second-stage aggregation based on the second. As will be seen later, the first property together with the first-stage aggregation trivially yet effectively rejects some existing attacks. The second property together with the second-stage aggregation effectively rejects more advanced attacks which bypasses our first-stage aggregation. As a cherry on top, in our DP protocol, such two properties themselves enable efficient hyper-parameter tuning.

4.2 Modifying DP Protocol

Our DP protocol is summarized in Algorithm 1. The two notable properties compared with vanilla DP-SGD [1] are: 1) different from existing works that adopt big batch size ($10^2 - 10^6$) [3, 19], we adopt small batch size b_c (typically 8 or 16). Note that small batch size is essential for our first-stage aggregation (see Section 4.3 for details); 2) the second, is to replace the clipping operation in vanilla DP-SGD by normalization, vanilla DP-SGD method clips the gradients by multiplying

Algorithm 1 Private and Secure Learning

Input: initial model w^0, number of iteration T, learning rate η, datasets held by n workers $\{D_i | i = 1, 2, \cdots, n\}$, gradient momentum β, noise multiplier σ, batch size b_c, loss function $f(;)$

Each worker i initializes a size-b_c momentum list $\phi_i^0 = [0, \cdots, 0]$
for $t = 1, 2, \cdots, T$ **do**
  Server broadcasts model w^{t-1} to all workers
  for $i = 1, 2, \cdots, n$ **do in parallel**
    Sample a size-b_c mini-batch d_i
    for $j = 1, 2, \cdots, b_c$ **do in parallel**
      $g_j \leftarrow \nabla f(x_j \in d_i; w^{t-1})$
      $\phi_i^t[j] \leftarrow (1-\beta) g_j + \beta \phi_i^{t-1}[j]$
    end for
    $g_i^t \leftarrow \frac{1}{b_c} \left(\sum_{j \in [b_c]} \frac{\phi_i^t[j]}{\|\phi_i^t[j]\|_2} + \mathcal{N}(0, \sigma^2 I) \right)$
    Upload g_i^t to server, then $\phi_i^t[j] \leftarrow g_i^t$
  end for
  $G_s^t \leftarrow$ **FilterGradient**$(\{g_i^t | i = 1, 2, \cdots, n\}, w^{t-1})$
  $w^t \leftarrow w^{t-1} - \eta \frac{1}{n} \sum_{g \in G_s^t} g$
end for

Output: learned target model w^T

the gradient vector g by $Factor = \min\{1, \frac{C}{\|g\|_2}\}$ (C is called the clipping threshold). We modify the multiplication factor to $Factor = \frac{1}{\|g\|_2}$, which normalizes the gradients to be of unit length. Also note that inspired by [17], to have a better convergence behavior, the gradients are processed with momentum.

We will see in the following why normalizing enables efficient hyper-parameter tuning. In later sections about defending against Byzantine attacks, we will also see that using small batch size is essential in our first-stage aggregation, and normalizing also plays an important role in the second-stage aggregation.

Normalization helps hyper-parameter tuning: We now introduce Theorem 1 which supports our hyper-parameter tuning strategy. Based on our DP protocol, consider a simpler case where there is no Byzantine attacker and we only have one honest worker. The model update (without momentum) in the t-th iteration has the following form:

$$w^t = w^{t-1} - \frac{\eta}{|B^t|} \left(\sum_{g^t \in B^t} \frac{g^t}{\|g^t\|} + z \right), \tag{3}$$

where B^t is the current local batch of per-example gradient (we fix the batch size to be $|B^t| = b_c$), $z \sim \mathcal{N}(0, \sigma^2 I)$ is the DP noise and $g^t = \nabla f(x; w^{t-1})$. Since g^t is derived from only a batch of samples, there is a sampling error. Denoting the sampling error by ξ^t, we can rewrite $\nabla f(x; w^{t-1})$ as $\nabla F(w^{t-1}) + \xi^t$, where $\nabla F(w^{t-1})$ is the gradient of the stochastic function. With some assumptions[4] that: 1) The stochastic function F is bounded below with $F(w) > 0$; 2) F has L-Lipschitz continuous gradient (defined in Assumption 1); 3) The random vector $g^t = \nabla F(w^{t-1}) + \xi^t$ has bounded variance, *i.e.*, $\mathbb{E} \left\| \xi^t \right\|^2 \leq v^2$ with some v. We have the following result:

[4]In deep neural networks we always have $F(w) > 0$, and for the L-Lipschitz continuous and bounded variance assumptions, they have been commonly used in the previous work for convergence analysis [10, 17, 71, 73].

Theorem 1. *(Convergence Behavior) Given a learning rate η, and the model is updated according to Equation 3 (gradient is normalized), we have*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left\|\nabla F(w^t)\right\| \leq \underbrace{\frac{3F(w^0)}{T\eta}+\frac{3L\eta}{2}\left(1+\frac{\sigma^2 d}{b_c^2}\right)}_{M}+8\nu.$$

Proof. See supp. material. □

By the above result, we can see that it is sufficient to minimize the term M, whose expression provides valuable guidance on choosing T and η that need to be set before training. If the magnitude of the noise and the batch size b_c satisfy $\frac{\sigma^2 d}{b_c^2} \gg 1$, then by setting the learning rate as

$$\eta = \frac{1}{\sigma}\sqrt{\frac{2F(w^0)b_c^2}{TLd}}, \tag{4}$$

we have $M \approx \sqrt{\frac{36F(w^0)Ld\sigma^2}{2b_c^2T}}$. Note that since we always have $\sigma = \Omega(\frac{q\sqrt{T\log(1/\delta)}}{\epsilon})$ where $q = \frac{b_c}{|D|}$ is the sampling rate ($|D|$ is the size of data) [1]. Thus, we have $M = \Omega\left(\frac{1}{\epsilon|D|}\sqrt{F(w^0)Ld\log(1/\delta)}\right)$. This implies that: 1) The lower bound of M is getting worse when ϵ becomes smaller; 2) We get this optimal bound by relating T and η via Equation (4). If we fix one, we can potentially get the other one analytically instead of going through inefficient hyper-parameter tuning. In practice, we fix T first and decide the learning rate η. With T fixed, Equation (4) suggests that the optimal learning rate should be set inversely proportional to the DP noise multiplier σ, and this leads to our efficient hyper-parameter tuning strategy which outperforms existing methods. Note that the previous analysis was built on the assumption that $\frac{\sigma^2 d}{b_c^2} \gg 1$. Thus, to satisfy the assumption, we can either use a bigger model (increase d) or adopt a smaller batch size. Hence, using a small batch size is preferred for our method and differs from existing work as mentioned before.

Hence, our DP approach saves quadratic efforts and is truly beneficial for DP learning. This is beyond only considering the running-time complexity.

4.3 First-stage Aggregation

Design strategy: Inspecting existing work on Byzantine resilience, the uploads (d-dimension vectors) by Byzantine attackers are arbitrary in $\mathbb{R}^d$. Hence, a single faulty inclusion on a malicious upload can totally destroy model updates. As a strategy, enforcing some constraints on the subspace where any upload should lie will be beneficial to defend against attacks. From a high-level perspective, our refactored DP protocol together with first-stage aggregation does the job of "*constrain*"; our second-stage aggregation does the job of "*complementary aggregation*".

Specifically, our choice of small batch size leads to the phenomenon that DP noise dominates for each upload, which leads to some expected statistical properties. Another phenomenon is that although for each worker DP noise is dominating, we can still achieve good utility overall. The reason is that the server takes the average of all aggregated uploads; such an operation reduces the DP noise variance and averages gradients to its non-zero expectation. Therefore, we can still get good utility as long as the number of honest workers is sufficiently large. As will be shown in the experiments section, 10-20 honest workers will suffice and such a number is smaller than that in previous work of FL systems [11, 21].

Forming first-stage aggregation: As we can see from Algorithm 1, an honest worker will upload $g = \tilde{g} + z$ to the server where $\tilde{g}$ is the sum of some normalized terms, and z is the DP noise.

If DP noise is dominating ($\|z\| \gg \|\tilde{g}\|$), we can approximately treat vector g as each coordinate of g is sampled from $\mathcal{N}(0, \sigma^2)$. We then have the following conclusion.

Norm test: Note that $\frac{\|g\|^2}{\sigma^2}$ follows the chi-squared distribution with degree d. As d is very large, by Central Limit Theorem, we can safely approximate the distribution of $\|g\|^2$ as Gaussian distribution: $\mathcal{N}(\sigma^2 d, 2\sigma^4 d)$. Hence, $\|g\|^2$ falls in the interval $\left[\sigma^2 d - 3\sigma^2\sqrt{2d}, \sigma^2 d + 3\sigma^2\sqrt{2d}\right]$ almost surely.[5]

Similarly, we can also conclude other statistical results for higher-order moments leveraging the property of Gaussian distribution. However, the real situation only allows us to use a limited number of these statistics. To further enhance the efficiency and the soundness of such checking, we will leverage non-parametric test methods which test the hypothesis that given samples follow a reference distribution. We leverage Kolmogorov–Smirnov test (KS test) [37] as described below.

KS test: Treat each coordinate of g as a sample and the null hypothesis is that these samples are sampled from the same distribution $\mathcal{N}(0, \sigma^2)$. Suppose that we are currently testing on upload g with d coordinates (d-dimension vector) and we denote the i-th coordinate of g as $g[i]$. KS test will 1) compute the empirical Cumulative Distribution Function as:

$$C_d(x) = \frac{1}{d}\sum_{i=1}^{d} \mathbf{1}_{g[i]<x},$$

where $\mathbf{1}_{g[i]<x}$ is the indicator function that takes value on 1 if $g[i] < x$ and 0 otherwise; 2) compute the KS statistics $D_{KS} = \sup_x |C_d(x) - \Phi_\sigma(x)|$ where $\Phi_\sigma(x)$ is the CDF of $\mathcal{N}(0, \sigma^2)$; 3) compute the P-value by D_{KS} from Kolmogorov D-statistic table [43] and there are many off-the-shelf libraries that can compute it. If the *P-value* is smaller than 0.05,[6] we reject the null hypothesis (the server then treats g as one malicious upload that is not sampled from $\mathcal{N}(0, \sigma^2)$ and rejects it).

Algorithm 2 FirstAGG(g)

Input: g, the upload to be tested

1: **if** $\|g\| < \sqrt{\sigma^2 d - 3\sigma^2\sqrt{2d}}$ or $\|g\| > \sqrt{\sigma^2 d + 3\sigma^2\sqrt{2d}}$ **then** $g \leftarrow 0$
2: **end if**
3: **if** $KS(g) < 0.05$ **then** $g \leftarrow 0$
4: **end if**

Output: g

KS test confines Byzantine subspace: After we set the significance level for the P-value, we are essentially placing an upper bound on D_{KS} (if D_{KS} is too large, the corresponding P-value will be small enough to make g to be rejected). And from the definition of D_{KS}, we know that such an upper bound applies to $|C_d(x) - \Phi_\sigma(x)|$ for any $x \in \mathbb{R}$. This can also be interpreted as that the curve of $C_d(x)$ will fall into a band bounded by an upper envelop $E_u(x) = min(1, \Phi_\sigma(x) + D_{KS})$ and a lower envelop $E_l(x) = max(0, \Phi_\sigma(x) - D_{KS})$. If we set the significance level strictly enough, D_{KS} will be small enough such that the band will be narrow, requiring that $C_d(x)$ almost aligns with $\Phi_\sigma(x)$.

Formally, since $C_d(x)$ is a step function (with d steps) that is monotonously increasing with d steps ($\frac{i}{d} \to \frac{i+1}{d}$ for $i = 0, 1, \cdots, d-1$), we have the following theorem.

[5]Such interval is narrow, because $\frac{\sigma^2\sqrt{2d}}{\sigma^2 d} \ll 1$ when d is very large. And by the 68-95-99.7 rule [67], we set such an interval to span three s.t.d. around the center so that a benign gradient falls into this interval with 99.7% probability approximately.
[6]We use the widely adopted significance level.

Theorem 2. *(Byzantine Resilience) If we sort all coordinates of an upload vector into a sequence by increasing order, to pass the **KS** test, the k-th (we count from 1 to d) element must fall into the interval:*

$$\left[E_u^{-1}\left(\frac{k}{d}\right), E_l^{-1}\left(\frac{k-1}{d}\right)\right] \tag{5}$$

Proof. Let the k-th element be x_k, according to the definition of $C_d(x_)$, then we must have $C_d(x_k) = \frac{k-1}{d}$ and $C_d(x_k + s) = \frac{k}{d}$ where s is some small real number. To pass the KS test, $\frac{k-1}{d}$ must be above $E_l(x_k)$ and $\frac{k}{d}$ must be below $E_u(x_k)$, consequently, x_k falls into the above interval. □

In essence, our first-stage aggregation enforces that the attacker's upload must lie in the subspace of $\mathbb{R}^d$ as described by Equation 5. This is different from existing work on Byzantine security, where the attacker's upload can be arbitrary in $\mathbb{R}^d$ [6, 11, 21, 51, 69].

Ensuring $\|z\| \gg \|\tilde{g}\|$: Our first-stage aggregation only keeps those uploads that follow our DP protocol (approximately with the form $g = \tilde{g} + z$) and this validity builds on the assumption that $\|z\| \gg \|\tilde{g}\|$. The good news is that we can always control $\frac{\|z\|}{\|\tilde{g}\|}$. Recall that $\|z\| \approx \sigma\sqrt{d}$ and $\tilde{g}$ is just the sum of b_c norm-bounded vectors, hence, before the training, we can compute $\frac{\|z\|}{\|\tilde{g}\|}$. To increase $\frac{\|z\|}{\|\tilde{g}\|}$, we can either 1) use a bigger model (increase d); or 2) adopt a smaller batch size for local workers. Thus, as highlighted in Section 4.2, using a small batch size is one of our technical details that differs from vanilla DP-SGD.

4.4 Second-stage Aggregation

In this part, we present our second-stage aggregation which does the job of "*complementary aggregation*". In total, our first-stage and second-stage aggregation constitute our final protocol shown in Algorithm 3.

As a complement: According to the resilience analysis for our first-stage aggregation, any acceptable upload is confined to lie in a special subspace described by Equation 5. To deceive our first-stage aggregation, the Byzantine attacker can also enforce its upload has the same form as $g = g' + z$ with $\|z\| \gg \|g'\|$ where z is the DP noise and g' is malicious component. Now the question is:

Is there an effective way for the server to differentiate benign uploads from Byzantine uploads based on the different nature of $\tilde{g}$ and g'?

The answer is yes if the server can get some estimate on the true gradient. To have such a capability, we assume that the server has access to some auxiliary data that can be used to compute the gradient during the training. Our empirical finding shows that *two samples per class* are enough for our second-stage aggregation to be effective. The intuition is that benign $\tilde{g}$ should update the model towards roughly the same direction as the true gradient ∇F while the malicious one does not. Quantitatively speaking, with high confidence, $\mathbb{E}\langle \nabla F, \tilde{g}\rangle > \mathbb{E}\langle \nabla F, g'\rangle$. And the server can use the gradient of non-private data to approximate ∇F.

Theoretical motivation: For simplicity, based on our DP protocol, at a certain iteration, considering one honest worker's upload is[7]:

$$g = \frac{\nabla f(x; w)}{\|\nabla f(x; w)\|} + z = \frac{\nabla F(w) + \xi}{\|\nabla F(w) + \xi\|} + z \tag{6}$$

where $z \sim \mathcal{N}(0, \sigma^2 I)$ is the DP noise and $\nabla f(x; w)$ can be written as $\nabla F(w) + \xi$ where ξ is random noise due to data sampling. We compute the expectation of the inner product between g and the true gradient $\nabla F(w)$, which has the following inequality (the proof is given by A.5.1 in supp. material.).

[7]We assume the batch size is 1. The iteration number is omitted for ease of notation as it is clear from the context.

$$\begin{aligned}\mathbb{E}\left\langle \nabla F(w), g\right\rangle =& \mathbb{E}\left\langle \nabla F(w), \frac{\nabla F(w)+\xi}{\|\nabla F(w)+\xi\|}\right\rangle \\ \geq& \frac{\mathbb{E}\|\nabla F(w)\|}{3}-\frac{8\mathbb{E}\|\xi\|}{3},\end{aligned} \tag{7}$$

where the expectation is taken over the randomness of the data sampling and DP noise. Note that Equation (6) is the special case where the honest worker only uses one data sample to compute the gradient and in the general case where many data samples are used, Equation (7) still holds (expectation is linear with respect to sum operation). Again, we do not have such bound if using clipping.

We then consider $\mathbb{E}\left\langle \nabla F, g'\right\rangle$ for the attacks we consider: 1) for Gaussian attack, $\mathbb{E}\left\langle \nabla F(w), g\right\rangle = 0$; 2) for Label-flipping attack, we hypothesize that $\mathbb{E}\left\langle \nabla F(w), g\right\rangle \leq 0$ as such Byzantine gradient is to destroy our learning; 3) for Optimized Local Model Poisoning attack, we have $\mathbb{E}\left\langle \nabla F(w), g\right\rangle < 0$ as this is the goal of such attack. In total, for the three attacks, we have $\mathbb{E}\left\langle \nabla F(w), g\right\rangle \leq 0$.

As $\mathbb{E}\|\xi\|^2$ is bounded, we can be confident that at least at the early phase of training, $\mathbb{E}\|\nabla F(w)\|$ is large enough to satisfy $\frac{\mathbb{E}\|\nabla F(w)\|}{3}-\frac{8\mathbb{E}\|\xi\|}{3}>0$. Our empirical result give positive evidence on the correctness of $\frac{\mathbb{E}\|\nabla F(w)\|}{3}-\frac{8\mathbb{E}\|\xi\|}{3}>0$. Thus, we can use this to filter Byzantine uploads and this is the foundation for our second-stage aggregation.

Algorithm 3 FilterGradient$(\{g_i^t | i=1,2,\cdots,n\}, w_{k-1})$

Input: gradients from each worker i at the t-th iteration g_i^t, model w^{t-1}, server-hold dataset D_p, the loss function $f(;)$, server-maintained score list $\mathbb{S}$, server's belief of honest worker ratio γ

1: **for** $i=1,2,\cdots,n$ **do in parallel**
2: $\quad g_i^t \leftarrow$ **FirstAGG**(g_i^t)
3: **end for**
4: Server computes $g_s^t \leftarrow \nabla f(D_p; w^{t-1})$
5: Server initialize $\mathbb{S}_{tmp} = [0,0,\ldots,0]$
6: **for** $i=1,2,\cdots,n$ **do in parallel**
7: $\quad \mathbb{S}_{tmp}[i] = \left\langle g_i^t, g_s^t\right\rangle$ ▷ Motivated by our analysis in Section 4.4
8: **end for**
9: $\hat{\mu} \leftarrow$ average of top $\lceil \gamma n \rceil$ scores in $\mathbb{S}_{tmp}$
10: **for** $i=1,2,\cdots,n$ **do in parallel**
11: $\quad \mathbb{S}_{tmp}[i] \leftarrow 0$ if $\mathbb{S}_{tmp}[i] < \hat{\mu}$
12: $\quad \mathbb{S}[i] \leftarrow \mathbb{S}[i] + \mathbb{S}_{tmp}[i]$
13: **end for**
14: Select those upload inside $\{g_i^t | i=1,2,\cdots,n\}$ which correspond to top $\lceil \gamma n \rceil$ scores inside $\mathbb{S}$ to form set G_s^t

Output: G_s^t

4.5 Final Byzantine-resilient Protocol

Combining all stages: According to the above statements, we design our second-stage aggregation which is shown in line 4-14 in Algorithm 3. In line 4, the server gets an estimation on ∇F by computing the gradient using some non-private data; Since the server has a prior belief that at least $\lceil \gamma n \rceil$ workers are honest, in line 9, server gets the average on top $\lceil \gamma n \rceil$ inner product scores among all scores computed by current upload of each worker (line 6-8). This average is used as the threshold to suppress scores lower than it to zero in line 11. By processing all scores by using the threshold, we can suppress all scores corresponding to Byzantine uploads and preserve the benign ones; the processed scores are accumulated in line 12 to be used to differentiate benign uploads

from Byzantine ones which is described in line 14. Then the selected vectors are returned for the model update.

Novelties: Our first-stage aggregation is the first aggregation rule leveraging the aforementioned DP properties. Leveraging auxiliary data to aid Byzantine aggregation is not new [11, 51], nonetheless, our second-stage aggregation differs from all existing approaches in 1) theoretical support: we have a solid theoretical explanation while previous work stands on heuristics; 2) differentiation metric: we use inner product while previous work use cosine similarity (cosine similarity never leads to the lower bound in Equation 7); 3) the way to integrate any upload into model update: by a unifying language, in our protocol, the weight assigned to any upload is binary (1 or 0), while existing work use real-valued weights according to computed similarities, we find that when DP is enforced (noise is added to the upload), assigning real-valued weights to any upload results in further biasing gradient which leads to rubbish model update.

4.6 Byzantine Attacks to Our Protocol

Recall that our attacker is the stronger version. In consistency with such consideration and to test the limit of our protocol's resilience, we stand in the perspective of an attacker and form possible attacks based on our already-released protocol.

Attacker's response: First, the attacker has to pass our first-stage aggregation to possibly have a malicious impact. Hence, he only has 2 possible guidelines in general:

Guideline 1: The attacker must first generate a d-element ordered sequence according to Theorem 2. Then, the attacker will form any permuted version of such sequence to be malicious. If the attacker is content with any permutation, this would be *Gaussian attack* [51, 76] as mentioned before. If the attacker aims to find any particular order, he will fail because it incurs $O(d!)$ computation complexity.

Guideline 2: Just like the honest workers' upload, the attacker can make his malicious upload has the form (or can be decomposed to such form)as $g = g' + z$ with $\|z\| \gg \|g'\|$ to pass the KS test.

For completeness, we will not only test on *Gaussian attack* but also test on other attacks which comply with Guideline 2. Following previous work, we will include *Label-flipping attack* [11, 21] and *Optimized Local Model Poisoning attack* [21]. The former has been described in previous sections and we will explain how to form the latter attack in the following. The attacker forms Optimized Local Model Poisoning attack by the following meta procedure:

- Infer the aggregated result g_r by applying the aggregation rule on all benign uploads;
- Based on the result and the aggregation rule, he forms his Byzantine upload which passes the aggregation and makes the final aggregated result have the inverse direction compared to g_r.

Accordingly, the goal of the adversary is to pass the first-stage aggregation to be possibly malicious further. We formally summarize such strategy as the following optimization problem:

$$\begin{aligned} \min_{\{g_{M_i}\}} \quad & S_c\left(\sum g_{M_i} + \sum g_{B_j}, \sum g_{B_j}\right) \\ s.t. \quad & \left\|\textbf{FirstAGG}(g_{M_i})\right\| > 0, \end{aligned} \tag{8}$$

where the constraint means that the Byzantine upload can pass our first-stage aggregation, $\{g_{M_i}\}$ are all Byzantine uploads by the Byzantine attacker, and $\{g_{B_j}\}$ are all benign uploads by honest workers. The function $S_c(A, B) = \frac{AB}{\|A\|\|B\|}$ calculates the cosine similarity between two vectors. Suppose all benign uploads are $g_{B_1}, g_{B_2}, \cdots, g_{B_m}$ by B_m honest workers and all Byzantine uploads are $g_{M_1}, g_{M_2}, \cdots, g_{M_n}$ by M_n Byzantine workers. According to Equation (8), the attacker aims to reach the following goal:

$$\sum g_{M_i} = -(1+\lambda)\sum g_{B_j} \tag{9}$$

where $\lambda > 0$ is a positive number. This leads to the term $\sum g_{M_i} + \sum g_{B_j} = -\lambda \sum g_{B_j}$ results in the inverse direction compared to $\sum g_{B_j}$. By setting:

$$g_{M_1} = g_{M_2} = \cdots = -\frac{(1+\lambda)}{M_n}\sum g_{B_j} \tag{10}$$

this goal is reached. And setting $\lambda = \frac{M_n}{\sqrt{B_m}} - 1$ will let Byzantine uploads pass our first-stage aggregation (one can check that all malicious and benign upload behaves the same when applying our first-stage aggregation on them).

Note that to be able to perform such an attack, we need $M_n > \sqrt{B_m}$ (because $\lambda > 0$), that is, such a strong attack only exists when the number of Byzantine workers is sufficiently large.

Note that we do not simulate the Optimized Local Model Poisoning attack compromising our second-stage aggregation because the attacker must know the serve-hold auxiliary dataset, and this means that the attacker must fully control the server which is unrealistic. Also note that, for the goal in Equation (8), we set it to be the inverse to the sum of all benign uploads. This is because such a goal leads to an efficient solution that can be tolerated by the attacker. In fact, the attacker can choose its goal freely as long as the constraint in Equation (8) is satisfied. However, other goals may not lead to an efficient solution. For instance, if the attacker chooses it to be orthogonal, the attacker is faced with the hard problem as discussed in Guideline 1.

Discussion on the adaptive attack: There exists another attack that copies benign uploads by honest workers for some iterations and suddenly turns to be malicious after that. We call this attack as *adaptive attack*. The way the attacker is malicious can be any instantiation of the previous three attacks we mentioned before. We will also include this attack in our experiment for completeness. Note that although Optimized Local Model Poisoning attack seems to be more advanced than Gaussian attack and Label-flipping attack, it is unclear which attack is most successful on our protocol before the experiment.

Discussion on excluded attacks: Optimized Local Model Poisoning attack performs well on attacking various existing Byzantine defense methods [21]. Another similar recent work [54] adopts the attacking intuition (the meta procedure mentioned above) of the Optimized Local Model Poisoning attack in other cases where the attacker's power is more limited (the attacker is weaker). Hence, here we only adopt the Optimized Local Model Poisoning attack in our experiments.

To the best of our knowledge, many other attacks can be trivially defended by our protocol, such as the attacks that have been considered in the existing work: "A little" attack [6] and "Inner" attack [69]. "A little" attack involves estimating the coordinate-wise mean and the s.t.d. of benign uploads to form its attack. However, our learning protocol enforces that the DP noise is dominating, hence knowing benign uploads gains the attacker no useful information when forming "A little" attack. Most importantly, naively applying such an attack will end up being rejected by first-stage aggregation. This shows the power of our protocol.

4.7 Discussions

First-stage aggregation provides critical robustness: Only using our second-stage aggregation to aggregate all worker's uploads is not enough, because, due to randomness, it is not guaranteed that Byzantine upload will never be selected for model update, and selected Byzantine upload could destroy our model in just one iteration as it is arbitrary. In contrast, there exists no such concern when we apply our first-stage aggregation, according to previous resilience analysis for our first-stage aggregation, it enforces any upload (including malicious ones) g which passes the filtering to have the form $g = \hat{g} + z$ with $\|z\| \gg \|\hat{g}\|$ where $\hat{g}$ is strictly norm-bounded and z is the

DP noise. For all malicious uploads, strictly norm-bounded $\hat{g}$ means their detrimental impact is bounded.

DP-Byzantine-robustness interaction: we do not consider DP and Byzantine-robustness in isolation. Instead, our whole protocol is formed by leveraging each other's properties.

As mentioned previously in our design strategy, we use our first-stage rule to "*constrain*" the way that any upload should behave by re-designing our DP protocol so that any Byzantine upload violating it will be immediately rejected. Hence, other than only protecting privacy, this refactoring on DP also provides the first-stage Byzantine-robustness. To deal with those Byzantine uploads that pass our first-stage aggregation, we further design our second-stage rule to do the "*complementary aggregation*" by leveraging the properties of our refactored DP protocol. In total, our privacy protocol and the robust aggregation rule are aware of each other, leading to a solution that is both privacy-preserving and Byzantine-resilient.

5 THEORETICAL GUARANTEES

We provide theoretical guarantees on privacy, utility, and Byzantine robustness of our protocol in this section. For convenience, we assume that the dataset of each worker has the same size which is denoted as $|D|$, and the size of non-private data held by the server is $|D_P|$. We also denote $w^* = \arg\min_{w\in\Theta} F(w)$.

Privacy guarantee: We have the following privacy guarantee.

Theorem 3. *(Privacy Guarantee) There exist constants c_1 and c_2 such that given the sampling rate $q = \frac{b_c}{|D|}$ and the number of iteration steps T. For each worker, Algorithm 1 is (ϵ, δ)-DP for any $\delta > 0$ and $\epsilon < c_1 q^2 T$ if $\sigma \geq c_2 \frac{q\sqrt{T \ln \frac{1}{\delta}}}{\epsilon}$.*

Proof. See A.5.2 in supp. material. □

Utility and Byzantine robustness: Theorem 4 shows the utility and robustness of Byzantine resilience of our algorithm. Before formally introducing Theorem4, we present some assumptions, which are commonly used in the previous work on optimization and Byzantine-robust learning [11, 15].

Assumption 1. *The expected loss function $F(w)$ is μ-strongly convex and differentiable over the space Θ with L-Lipschitz continuous gradient. Formally, we have the following for any $w, \widehat{w} \in \Theta$:*

$$F(\widehat{w}) \geq F(w) + \langle \nabla F(w), \widehat{w} - w \rangle + \frac{\mu}{2}\|\widehat{w} - w\|^2$$
$$\|\nabla F(w) - \nabla F(\widehat{w})\| \leq L\|w - \widehat{w}\|.$$

Moreover, the empirical loss function $f(D, w) := \frac{1}{|D|}\sum_{x\in D} f(x; w)$ is L_1-Lipschitz continuous with high probability. Formally, for any $\zeta \in (0, 1)$, there exists an L_1 such that:

$$\Pr\left(\sup_{w,\widehat{w}\in\Theta: w\neq\widehat{w}} \frac{\|\nabla f(D, w) - \nabla f(D, \widehat{w})\|}{\|w - \widehat{w}\|} \leq L_1\right) \geq 1 - \frac{\zeta}{3}$$

Assumption 2. *The gradient of the empirical loss function $\nabla f(D, w^*)$ at the optimal global model w^* is bounded. Moreover, the gradient difference $h(D, w) = \nabla f(D, w) - \nabla f(D, w^*)$ for any $w \in \Theta$ is bounded. Specifically, there exist positive constants σ_1 and γ_1 such that for any unit vector $\boldsymbol{v}$, $\langle \nabla f(D, w^*), \boldsymbol{v} \rangle$ is sub-exponential with σ_1 and γ_1; and there exist positive constants σ_2 and γ_2 such that for any $w \in \Theta$ with $w \neq w^*$ and any unit vector $\boldsymbol{v}$, $\langle h(D, w) - \mathbb{E}[h(D, w)], \boldsymbol{v} \rangle / \|w - w^*\|$ is*

sub-exponential with σ_2 and γ_2. Formally, for all $|\tau| \leq 1/\gamma_1, |\tau| \leq 1/\gamma_2$, we have:

$$\sup_{\boldsymbol{v}\in\boldsymbol{B}} \mathbb{E}\left[\exp\left(\tau\left\langle\nabla f\left(D, w^*\right), \boldsymbol{v}\right\rangle\right)\right] \leq e^{\sigma_1^2\tau^2/2}$$
$$\sup_{w\in\Theta,\boldsymbol{v}\in\boldsymbol{B}} \mathbb{E}\left[\exp\left(\frac{\tau\langle h(D,w)-\mathbb{E}[h(D,w)],\boldsymbol{v}\rangle}{\|w-w^*\|}\right)\right] \leq e^{\sigma_2^2\tau^2/2}$$

where $\boldsymbol{B}$ is the unit sphere $\boldsymbol{B} = \{\boldsymbol{v} : \|\boldsymbol{v}\| = 1\}$

Note that the strongly convex and Lipschitz continuous conditions in Assumption 1 are widely adopted in the convergence analysis of optimization algorithms, and these conditions indicate the largest eigenvalue of the Hessian matrix of the loss function is between μ and L. Assumption 2 indicates that the gradient $\nabla f(D, w^*)$ is quite close to its expectation $\mathbb{E}[\nabla f(D, w^*)] = 0$, and the difference $h(D, w)$ concentrates to its expectation with high probability.

Theorem 4. *For an arbitrary number of Byzantine workers, the difference between the global model learned by Algorithm 1 and the optimal global model w^* under no attacks is bounded. Specifically, if the parameter space $\Theta \subseteq B(0, r\sqrt{d})$,* i.e.*, it is contained in a ball with radius $r\sqrt{d}$ and $\nabla F(w^*) = 0$. Set σ as in Theorem 3, $T = O\left(\frac{1}{\rho}\ln\left(\sqrt{n}|D|\sqrt{|D_0|}\right)\right)$ and $\eta_{t-1} \leq \left\|g_s^{t-1}\right\|_2 \eta_0$ with fixed $\eta_0 \leq \frac{\mu}{2L^2}$ in the t-th iteration in Algorithm 1, then if $n, |D_p|$ and $|D|$ are sufficiently large and η_0 is sufficiently small such that*

$$\sqrt{n} \geq \tilde{\Omega}\left(\frac{\sqrt{d\ln\frac{1}{\delta}}}{\epsilon|D|}\cdot\max\left\{\sqrt{\ln\frac{1}{\xi}}, \frac{\ln\frac{1}{\xi}}{r\rho\sqrt{|D_p|}}\right\}\right) \tag{11}$$

and $\frac{\eta_0\sigma_1}{\sqrt{|D_p|}} \leq O\left(\frac{r\sqrt{d}}{\rho}\right)$ with $0 < \rho < 1$. Then, with probability at least $1-\xi$ with $\xi \in (0, 1)$, we have:

$$\|w_T - w^*\| \leq \tilde{O}\left(\frac{1}{\rho^2}\frac{d\ln\frac{1}{\zeta}\sqrt{\ln\frac{1}{\delta}}\sigma_1}{|D|\sqrt{nb_s}\epsilon} + \frac{1}{\rho}\frac{\sigma_1\sqrt{d\ln\frac{1}{\zeta}}}{\sqrt{|D_p|}}\right), \tag{12}$$

where the Big-$\tilde{O}$ and Big-$\tilde{\Omega}$ notations omit other logarithmic terms. Here $\rho = 1 - \sqrt{1-\frac{u^2}{4L^2}} - 32\eta_0\Delta_2 - 3\eta_0 L$ with $\Delta_2 = \sigma_2\sqrt{\frac{2}{|D_p|}}\sqrt{K_1+K_2}$ with $K_1 = d\log\frac{\max\{L,L_1\}}{\sigma_2}$ and $K_2 = \frac{d}{2}\log\frac{|D_p|}{d} + \log\frac{6\sigma^2 r\sqrt{|D_p|}}{\gamma_2\sigma_1\zeta}$.

Proof. In A.5.3 in supp. material. □

Theorem 4 is on the robustness. Briefly speaking, as can be seen from Equation (12), if $|D|$, n and $|D_p|$ are large enough (nonetheless, our experiment shows that only a small number of non-private data will suffice), with some iteration number and stepsize, even there is an arbitrary number of Byzantine workers, the final model we get will be close to the optimal model (measured by the l_2 distance) with high probability.

6 EXPERIMENTAL RESULTS

6.1 Datasets and System Settings

Datasets and models: We conduct experiments on MNIST [40], Colorectal [36], Fashion [68], and USPS [33]. Details of all these benchmark datasets with various properties are summarised in supp. material.. Details of neural network setup are also in supp. material. Each experiment is repeated with different random seeds {1, 2, 3} and we report the min. max. and mean. All of our experiments are conducted under the same base learning rate $\eta_b = 0.2$ (which will be explained later), batch size $b_c = 16$ and momentum $\beta = 0.1$. We set the number of epochs $T = \lceil 10|D|/b_c\rceil$ for Colorectal and USPS, $T = \lceil 8|D|/b_c\rceil$ for MNIST and Fashion.

Data sample distribution: We consider both *i.i.d.* and *non-i.i.d.* settings. To be specific, *i.i.d.* is the case where each worker's local dataset follows the same distribution as the whole data population while *non-i.i.d.* is the case where each worker's local dataset's distribution is arbitrary [44]. We simulate both settings following previous work [11, 21, 29, 76], and details are presented in supp. material.

For generating server-own auxiliary data, we only randomly sample 2 data samples per class from the validation dataset. As mentioned earlier, obtaining such a tiny amount of data is easy. Note that generating such auxiliary data is totally agnostic to the distribution of the whole data population while such data still enables our protocol's effectiveness (as will be confirmed by our experiments). Once the auxiliary data is generated, it is vacuous to compare the distribution of such auxiliary data to distributions of any other datasets, because the size of our auxiliary data is micro.

Byzantine setup: We fix the number of honest workers (20 for MNIST and Fashion, 10 for Colorectal and USPS), and vary the number of Byzantine workers (0%, 20%, 40%, 60%, 90% of total).

Privacy settings: We do experiments on different privacy settings $\epsilon = \{2^{-3}, 2^{-2}, 2^{-1}, 2^{0}, 2^{1}\}$ while fixed $\delta = 1/|D_i|^{1.1}$, where $|D_i|$ is the size of the local dataset possessed by worker i.

Reference Accuracy:

The *Reference Accuracy* is the testing accuracy of FL under the scenario where no Byzantine threat exists and FL only adopts DP (not adopting any Byzantine defense method). Compare any private and Byzantine-resilient protocol's performance to the *Reference Accuracy*, many useful conclusions can be drawn:

- **Side-effect:** Apply a protocol under the scenario where there are no Byzantine threats, by comparing it with *Reference Accuracy*, we know how much "side-effect" caused by that protocol. The ideal case is that we expect the "medicine" causes no additional harm to the "patient" with no "illness".
- **Efficacy:** For the scenario where there is a certain number of attackers, by comparing with *Reference Accuracy*, we know how effectively a protocol defends the attack. The ideal case is that the "medicine" eradicates the "illness" (under such case, the performance should be the same as *Reference Accuracy*).

6.2 Claims and Experimental Evidence

All of the attacks we consider have been tested. Based on the observation that our protocol remains resilient across all attacks and due to space limitation, we arbitrarily only present results for *Label-flipping attack* under *i.i.d.* in the main body. All additional results for other attacks we consider under both *i.i.d.* and *non-i.i.d* settings is in supp. material.

A quick overview: We provide 7 claims with their corresponding evidence. By comparing with previous work, claim 1-2 show our contribution to DP learning and Byzantine resilience in their own track. Most importantly, recall our core aim is to ensure privacy and Byzantine resilience simultaneously, we use claims 3-7 to show its effectiveness.

1) Normalizing is better than vanilla DP-SGD (which uses clipping) at 1) gaining Byzantine resilience; 2) efficiently tuning hyper-parameter for DP deep learning.

Evidence: A thought experiment will suffice. Recall that clipping is essentially normalizing gradient vectors with ℓ_2-norm greater than C to be C exactly and leaving those vectors with ℓ_2-norm smaller than C untouched. If we are guaranteed that all gradient vectors' ℓ_2-norm is greater than C, then normalizing and clipping only differ in the learning rate scale and are essentially equivalent to each other, *i.e.*, clipping with $C = 2, \eta = 0.1$ is the same as normalizing (to be unit ℓ_2-norm) with $\eta = 0.2$. This means that if we have that guarantee, clipping will also enjoy the lower bound in

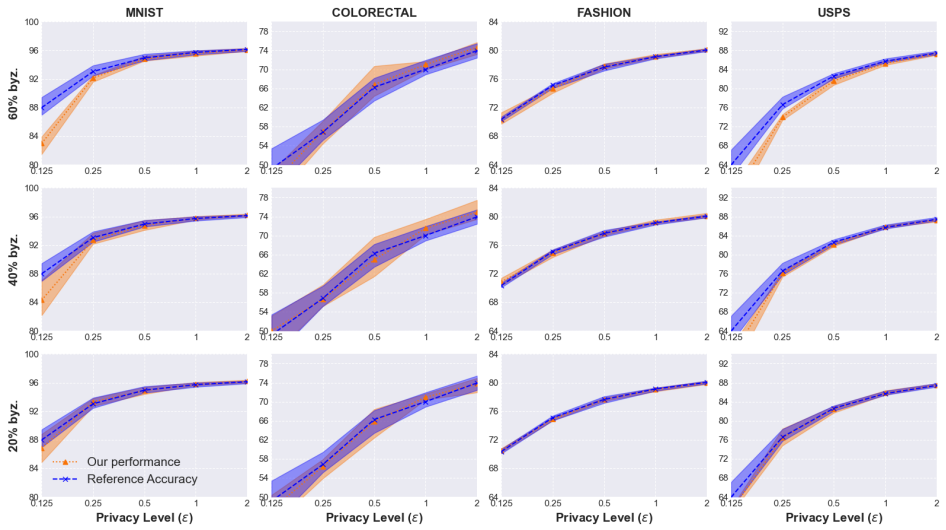

Fig. 1. Byzantine-resilient performance (testing accuracy) under Label-flipping attack. The experiment is conducted under 3 different attacking levels (20%, 40%, 60% of the total workers are Byzantine).

Equation 7 for gaining Byzantine resilience and will also enjoy our analysis for hyper-parameter tuning in Theorem 1.

However, *it is unfeasible to get a prior bound* on the gradient vector's norm for arbitrary deep-learning neural networks. Meanwhile, it is unclear whether clipping could lead to similar theoretical results which serve our purpose. Adopting normalizing circumvents such issues.

2) Our protocol outperforms existing solutions.

Evidence: We compare our protocol to previous work with the same aim (ensure privacy and Byzantine security simultaneously). We will show that our tailored Byzantine aggregation with DP outperforms previous solutions whose methodology is to naively apply off-the-shelf Byzantine aggregation with DP. And our result shows the contribution of our Byzantine aggregation rule.

For a fair comparison, we provide the results for the scenario where our privacy level is similar and our attacker is the same compared with existing solutions.

Method	Byz./ Privacy	[6] attack	[69] attack
[29]	40%, $\epsilon = 3.46$	.61	.75
	20%, $\epsilon = 7.58$	.78	.79
Ours	60%, $\epsilon = 2.00$	.79 ± .010	.80 ± .010
	40%, $\epsilon = 2.00$	.80 ± .005	.80 ± .005

(a) Testing accuracy comparison with existing work [29] on Fashion.

Method	Byz./ Privacy	Gaussian attack
[76]	10%, $\epsilon = .21$	.20
	10%, $\epsilon = .40$	.43
Ours	60%, $\epsilon = .125$	.86 ± .010
	40%, $\epsilon = .125$	.86 ± .010

(b) Testing accuracy comparison with existing work [76] on MNIST.

Comparison with [29]: We compare our results with [29] in Table 2a. We can see from Table 2a that [29] only reaches 61% accuracy under 40% "a little" Byzantine attack [6] in the privacy setting ($\epsilon = 3.46, \delta = 1.2 \times 10^{-4}$). We also notice that under the same privacy setting but a different Byzantine attack, [29] achieves 75% testing accuracy, and [29] makes comments that "a little" attack is stronger against their defense.

Applying our Byzantine defense method under the same attacks, we get around 80% testing accuracy when there are 60%, 40% Byzantine workers in the privacy setting ($\epsilon = 2, \delta = 1.4 \times 10^{-4}$).

Thus, we can gain much more utility compared to [29] even when the majority worker are Byzantine and with even better privacy guarantee (we are ensuring $(\epsilon = 2, \delta = 1.4 \times 10^{-4})$-DP instead of $(\epsilon = 3.46, \delta = 1.2 \times 10^{-4})$-DP). The utility we gain is also better than [29] under its weakest attack with a much weaker privacy guarantee: $(\epsilon = 7.58, \delta = 1.2 \times 10^{-4})$-DP.

Comparison with [76]: As can be seen from Table 2b, the method in [76] reaches 43% testing accuracy on MNIST when there are only 10% Byzantine workers under the privacy setting ($\epsilon = 0.4, \delta = 0$). As a comparison, our learning protocol provides 86% testing accuracy when there are 60% Byzantine workers under privacy setting ($\epsilon = 0.125, \delta = 1.4 \times 10^{-4}$). We gain much more utility even when the majority of workers are Byzantine, which is impossible for [76] to accomplish due to their intrinsic limitation of aggregation methods.

	MNIST			COLOR.			FASHION			USPS		
	$\epsilon =$			$\epsilon =$			$\epsilon =$			$\epsilon =$		
	$\frac{1}{8}$	$\frac{1}{2}$	2	$\frac{1}{8}$	$\frac{1}{2}$	2	$\frac{1}{8}$	$\frac{1}{2}$	2	$\frac{1}{8}$	$\frac{1}{2}$	2
RA	.88	.95	.96	.49	.66	.74	.69	.77	.80	.64	.82	.87
zero	.85	.94	.96	.44	.67	.74	.69	.77	.80	.58	.81	.87

Table 3. Experimental result on the test for the "side-effect" our protocol brings. *RA* stands for *Reference Accuracy* and *zero* stands for the scenario where all original 60% Byzantine workers turn to behave honestly (hence we have *zero* attackers) while our protocol is still applied. The performance (testing accuracy) results from taking the average of three runs with different seeds.

3) Our protocol brings no "side-effect" even when there is no Byzantine attack.

Evidence: we design the following experiment to test whether our protocol brings any "side-effect". Let 60% of workers be Byzantine, however, those Byzantine workers do not perform any attack. Instead, they behave just like all honest workers. The server still follows its prior belief that only 40% of workers are trustworthy. Our results are shown in Table 3.

We can see that other than at the extreme privacy level ($\epsilon = 1/8 = .125$), our protocol's performance is almost identical to the *Reference Accuracy*, hence incurring no "side-effect". We indeed observe a noticeable accuracy drop when at $\epsilon = 1/8 = .125$, this is because in such extreme case, noise becomes so overwhelming that the training itself is not stable.

4) Our protocol eradicates Byzantine attacks if not facing extreme privacy requirements.

Evidence: Figure 1 shows the performance of our method. The testing accuracy almost always aligns with the Reference Accuracy. Such a phenomenon can be observed not only across different privacy levels but also across different datasets.

The most discrepant results are observed for USPS and MNIST datasets when there are 60% Byzantine workers in the high privacy regime with $\epsilon = 0.125$. This is because, at this extreme privacy level, significant noise is added to the gradient, We are getting less confident in differentiating benign uploads from Byzantine ones when we are at such a high privacy level, fortunately, our first-stage aggregation guarantees that malicious upload has limited detrimental impact even if selected.

We also observe that results for Colorectal present a larger variance than the rest datasets. This is because the dataset size is much smaller (only 5,000 samples in total) than the rest and it is limited by the intrinsic limitation that DP learning requires large-scale data.

5) Our protocol remains robust against majority attack.

Evidence: Results are shown in Figure 2. We can observe that even when 90% workers are Byzantine (we have also simulated more stringent cases where 95%, 99% workers are Byzantine, results can be found in *supp. material*), similar results can be observed compared with the cases where there are 60%, 40%, 20% Byzantine workers. We observe a noticeable accuracy drop for certain

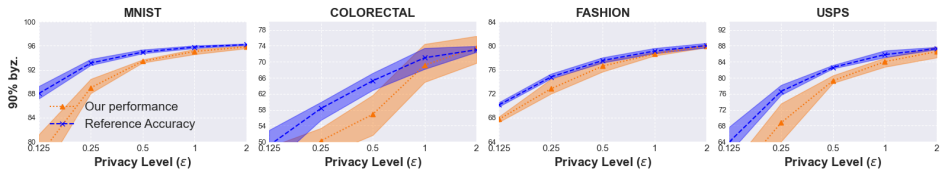

Fig. 2. Byzantine-resilient performance (testing accuracy) when 90% workers are Label-flipping Byzantine attackers.

datasets when $\epsilon = 0.125$ and $\epsilon = 0.25$ due to overwhelming random noise which guarantees high privacy. For $\epsilon \geq 0.5$, we still gain privacy and Byzantine resilience without hurting too much performance.

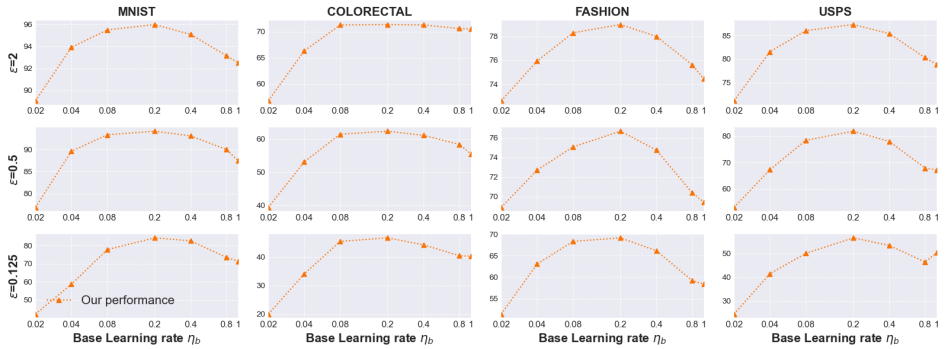

Fig. 3. Our hyper-parameter tuning results when facing 60% Label-flipping attackers.

6) As a cherry on top, our protocol enables efficient hyper-parameter tuning by saving quadratic efforts.

Evidence: For a typical DP deep learning task, vanilla DP-SGD's running task spans on the 3-dimensional tuple (η, C, ϵ). In contrast, adopting normalizing together with our tuning strategy only needs to tune η for one arbitrary ϵ. That is, we only need to tune the learning rate η_b for one privacy level ϵ with the corresponding noise multiplier σ_b, then we can use the learning rate $\eta = \frac{\eta_b \sigma_b}{\sigma}$ for any other privacy level with noise multiplier σ. We call η_b and σ_b at the privacy level we are tuning as *"base learning rate"* and *"base noise multiplier"*.

To evaluate the effectiveness of such a strategy, it suffices to confirm that if we find the optimal base learning rate for one privacy level, we also find the optimal learning rate for other privacy levels by setting the learning rate according to such a strategy. In this sense, we first choose the base case of $\sigma_b = 0.79$ (corresponding to $\epsilon = 2$). Then, for each privacy level, we tune the learning rate with respect to different base learning rates (the actual learning rate is computed according to the above strategy). In our experiment, we vary the base learning rate among $\{0.02, 0.04, 0.08, 0.2, 0.4, 0.8, 1\}$, so the actual learning rate will be $\{\frac{0.02\sigma_b}{\sigma}, \frac{0.04\sigma_b}{\sigma}, \frac{0.08\sigma_b}{\sigma}, \frac{0.2\sigma_b}{\sigma}, \frac{0.4\sigma_b}{\sigma}, \frac{0.8\sigma_b}{\sigma}, \frac{\sigma_b}{\sigma}\}$ for a specific privacy level with noise multiplier σ.

Results on MNIST are shown in Figure 3, and we can see that for all the privacy levels we considered, the optimal point is the same ($\frac{0.2\sigma_b}{\sigma}$ for MNIST), and a similar phenomenon can also be observed on the other three datasets.

7) Our protocol remains resilient against adaptive attack.

TTBB	MNIST		COLOR.		FASHION		USPS	
	$\epsilon =$		$\epsilon =$		$\epsilon =$		$\epsilon =$	
	2	.125	2	.125	2	.125	2	.125
0	.96	.82	.74	.45	.80	.68	.86	.60
.2	.96	.82	.74	.41	.80	.68	.86	.60
.4	.96	.81	.73	.45	.80	.68	.86	.57
.6	.96	.81	.73	.44	.80	.69	.86	.57
.8	.96	.82	.73	.43	.80	.69	.86	.60

Table 4. Under Label-flipping attack with different TTBB.

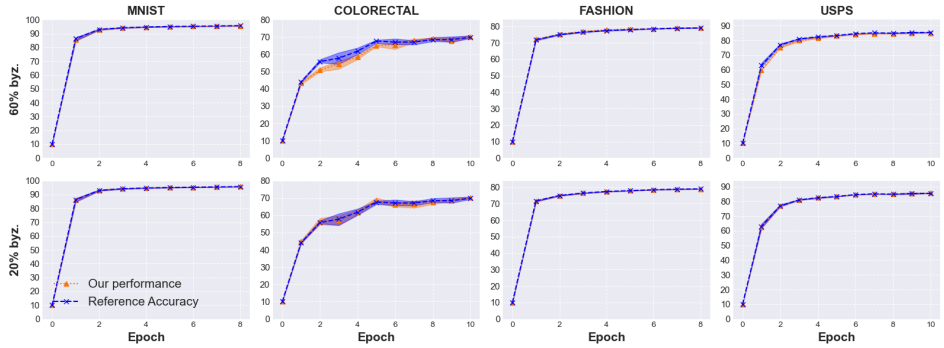

Fig. 4. Byzantine-resilient convergence curves (testing accuracy) under Label-flipping attack (considering 20%, 60% of the total workers are Byzantine, fixing $\epsilon = 1$).

Evidence: Our robust and private learning framework is also resilient to *adaptive attack*. We evaluate that by letting 60% Byzantine workers be honest via copying the uploads of some random honest workers from the beginning of training and turning to Byzantine at different iterations to see if they can possibly have a significant impact. Results are shown in Table 4. The first column represents the Time To Be Byzantine (TTBB), *i.e.*, if the total iteration is T, 0.2 TTBB means that Byzantine workers behave honestly within the first $0.2T$ iterations and then start to send Byzantine uploads thereafter.

We can see that no matter when the Byzantine workers start to be Byzantine, they all have a negligible impact on the testing accuracy except for the case with extreme privacy requirements. We notice that there are some mild performance fluctuations when $\epsilon = 0.125$ for Colorectal and USPS, again, due to the large variance of DP noises.

γ	MNIST		COLOR.		FASHION		USPS	
	$\epsilon =$		$\epsilon =$		$\epsilon =$		$\epsilon =$	
	$\frac{1}{8}$	2	$\frac{1}{8}$	2	$\frac{1}{8}$	2	$\frac{1}{8}$	2
20%	.86	.95	.48	.73	.66	.78	.64	.85
35%	.87	.96	.47	.74	.69	.79	.63	.86
50% *(exact)*	.88	.96	.49	.74	.69	.80	.64	.87
65%	.85	.96	.45	.73	.70	.79	.56	.87
80%	.83	.95	.34	.74	.69	.79	.54	.85

Table 5. γ is treated as a prior belief in this experiment and we study the effect when there is a mismatch between such belief and the truth. We fix the setting that 50% workers are honest and vary γ. For the case where the belief is exactly the truth ($\gamma = 50\%$), we denote it as "exact". All results are obtained by taking the average of three runs with different random seeds.

6.3 More Experimental Results

Convergence behavior: The convergence curve is presented in Figure 4. As can be seen in Figure 4, the training converges in the first several epochs. The convergence behavior of our protocol aligns well with "*Reference Accuracy*" even when we have 60% Byzantine workers. Similar to previous results in our CLAIM 4, We observe a larger variance for Colorectal than that of the rest datasets. As expected, this is due to its significantly small dataset size and the nature of training with DP.

Ablation study on γ: Recall that previously we assumed the server knows that at least γn workers are honest, what if γ is only a (prior) belief rather than the truth, and moreover, what if there is a mismatch between such belief and the truth? We further conduct an ablation study on γ if it is only a belief. We can see from Table 5 that, in the case where 50% workers are honest, as long as the server is conservative ($\gamma \leq 50\%$), we can still retain robustness. In contrast, we observe a notable utility drop for Colorectal and USPS under privacy level $\epsilon = .125$ when the server radically believes that 80% workers are honest, this is because in our protocol, being radical (γ is greater than the true honest portion) means the server tends to aggregate malicious uploads. Hence, the more radical, the worse the utility is expected to be. Based on such observation, the learned lesson is that we can always have robustness if we are not facing extreme privacy requirements and a conservative γ is set.

7 CONCLUSION

In this paper, with the aim to ensure both DP and Byzantine resilience for FL systems, we developed a learning protocol resulting from a co-design principle. We refactor the DP-SGD algorithm and tailor the Byzantine aggregation process towards each other to form an integrated protocol. For our DP-SGD variant, the small batch size property enables our first-stage Byzantine aggregation which trivially rejects many existing Byzantine attacks; the normalization technique enables our second-stage aggregation which provides a final sound filtering. As a cherry on top, normalizing also enables our efficient hyper-parameter tuning strategy which saves quadratic efforts. We also provide theoretical explanations behind the efficacy of our protocol.

In the experiment part, we first provide evidence to support our contribution claim to both DP learning and Byzantine security tracks in separation, we then provide evidence of the effectiveness of our protocol tackling the two-fold issue, *i.e.*, an FL system needs to be privacy-preserving and Byzantine-resilient simultaneously. We have shown that our protocol does not incur "side-effects" to a system with no Byzantine attacker, and we have also seen that our protocol remains Byzantine-resilient even when there are up to 90% distributive workers being Byzantine.

ACKNOWLEDGMENTS

Di Wang and Zihang Xiang were supported by BAS/1/1689-01-01, URF/1/4663-01-01, FCC/1/1976-49-01, RGC/3/4816-01-01, and REI/1/4811-10-01 of King Abdullah University of Science and Technology (KAUST) and KAUST-SDAIA Center of Excellence in Data Science and Artificial Intelligence. Tianhao Wang is supported by NSF-2220433 and NSF-2213700. Wanyu Lin is supported by NSFC-P0040473 and Hong Kong Research Grants Council General Research Fund P0041813.

REFERENCES

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

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

[3] Rohan Anil, Badih Ghazi, Vineet Gupta, Ravi Kumar, and Pasin Manurangsi. 2021. Large-scale differentially private bert. *arXiv preprint arXiv:2108.01624* (2021).

[4] Shahab Asoodeh, Jiachun Liao, Flavio P Calmon, Oliver Kosut, and Lalitha Sankar. 2021. Three variants of differential privacy: Lossless conversion and applications. *IEEE Journal on Selected Areas in Information Theory* 2, 1 (2021), 208–222.

[5] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2938–2948.

[6] Gilad Baruch, Moran Baruch, and Yoav Goldberg. 2019. A little is enough: Circumventing defenses for distributed learning. *Advances in Neural Information Processing Systems* 32 (2019).

[7] Raef Bassily, Adam Smith, and Abhradeep Thakurta. 2014. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE, 464–473.

[8] Raef Bassily, Om Thakkar, and Abhradeep Guha Thakurta. 2018. Model-agnostic private learning. *Advances in Neural Information Processing Systems* 31 (2018).

[9] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 118–128.

[10] Zhiqi Bu, Yu-Xiang Wang, Sheng Zha, and George Karypis. 2022. Automatic Clipping: Differentially Private Deep Learning Made Easier and Stronger. *arXiv preprint arXiv:2206.07136* (2022).

[11] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. 2020. Fltrust: Byzantine-robust federated learning via trust bootstrapping. *arXiv preprint arXiv:2012.13995* (2020).

[12] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. 2011. Differentially private empirical risk minimization. *Journal of Machine Learning Research* 12, 3 (2011).

[13] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* (2017).

[14] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* (2017).

[15] Yudong Chen, Lili Su, and Jiaming Xu. 2017. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1, 2 (2017), 1–25.

[16] Christopher A Choquette-Choo, Natalie Dullerud, Adam Dziedzic, Yunxiang Zhang, Somesh Jha, Nicolas Papernot, and Xiao Wang. 2021. Capc learning: Confidential and private collaborative learning. *arXiv preprint arXiv:2102.05188* (2021).

[17] Ashok Cutkosky and Harsh Mehta. 2020. Momentum improves normalized sgd. In *International Conference on Machine Learning*. PMLR, 2260–2268.

[18] Rudrajit Das, Abolfazl Hashemi, Sujay Sanghavi, and Inderjit S Dhillon. 2021. Privacy-Preserving Federated Learning via Normalized (instead of Clipped) Updates. *arXiv preprint arXiv:2106.07094* (2021).

[19] Soham De, Leonard Berrada, Jamie Hayes, Samuel L Smith, and Borja Balle. 2022. Unlocking high-accuracy differentially private image classification through scale. *arXiv preprint arXiv:2204.13650* (2022).

[20] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.

[21] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local Model Poisoning Attacks to {Byzantine-Robust} Federated Learning. In *29th USENIX Security Symposium (USENIX Security 20)*. 1605–1622.

[22] Fangcheng Fu, Yingxia Shao, Lele Yu, Jiawei Jiang, Huanran Xue, Yangyu Tao, and Bin Cui. 2021. VF^{2}Boost: Very Fast Vertical Federated Gradient Boosting for Cross-Enterprise Learning. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 563–576. https://doi.org/10.1145/3448016.3457241

[23] Fangcheng Fu, Huanran Xue, Yong Cheng, Yangyu Tao, and Bin Cui. 2022. BlindFL: Vertical Federated Machine Learning without Peeking into Your Data. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 1316–1330. https://doi.org/10.1145/3514221.3526127

[24] gboard [n. d.]. Federated Learning: Collaborative Machine Learning without Centralized Training Data. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html.

[25] gdpr [n. d.]. General Data Protection Regulation (GDPR). https://gdpr.eu/what-is-gdpr/.

[26] Robin C Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557* (2017).

[27] Aditya Golatkar, Alessandro Achille, Yu-Xiang Wang, Aaron Roth, Michael Kearns, and Stefano Soatto. 2022. Mixed Differential Privacy in Computer Vision. *arXiv preprint arXiv:2203.11481* (2022).

[28] Sivakanth Gopi, Yin Tat Lee, and Lukas Wutschitz. 2021. Numerical Composition of Differential Privacy. *arXiv preprint arXiv:2106.02848* (2021).

[29] Rachid Guerraoui, Nirupam Gupta, Rafael Pinot, Sebastien Rouault, and John Stephan. 2021. Combining Differential Privacy and Byzantine Resilience in Distributed SGD. *arXiv preprint arXiv:2110.03991* (2021).

[30] Rachid Guerraoui, Nirupam Gupta, Rafael Pinot, Sebastien Rouault, and John Stephan. 2021. Differential Privacy and Byzantine Resilience in SGD: Do They Add Up?. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*. 391–401.

[31] Rachid Guerraoui, Sébastien Rouault, et al. 2018. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*. PMLR, 3521–3530.

[32] Jihun Hamm, Yingjun Cao, and Mikhail Belkin. 2016. Learning privately from multiparty data. In *International Conference on Machine Learning*. PMLR, 555–563.

[33] Jonathan J. Hull. 1994. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence* 16, 5 (1994), 550–554.

[34] Roger Iyengar, Joseph P Near, Dawn Song, Om Thakkar, Abhradeep Thakurta, and Lun Wang. 2019. Towards practical differentially private convex optimization. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 299–316.

[35] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. 2018. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 19–35.

[36] Jakob Nikolas Kather, Cleo-Aron Weis, Francesco Bianconi, Susanne M Melchers, Lothar R Schad, Timo Gaiser, Alexander Marx, and Frank Gerrit Zöllner. 2016. Multi-class texture analysis in colorectal cancer histology. *Scientific reports* 6, 1 (2016), 1–11.

[37] Andrey Kolmogorov. 1933. Sulla determinazione empirica di una lgge di distribuzione. *Inst. Ital. Attuari, Giorn.* 4 (1933), 83–91.

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

[39] Alexey Kurakin, Steve Chien, Shuang Song, Roxana Geambasu, Andreas Terzis, and Abhradeep Thakurta. 2022. Toward training at imagenet scale with differential privacy. *arXiv preprint arXiv:2201.12328* (2022).

[40] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[41] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning Attack on Neural Networks. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society. http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_03A-5_Liu_paper.pdf

[42] Xu Ma, Xiaoqian Sun, Yuduo Wu, Zheli Liu, Xiaofeng Chen, and Changyu Dong. 2022. Differentially Private Byzantine-robust Federated Learning. *IEEE Transactions on Parallel and Distributed Systems* (2022).

[43] George Marsaglia, Wai Wan Tsang, and Jingbo Wang. 2003. Evaluating Kolmogorov's distribution. *Journal of statistical software* 8 (2003), 1–4.

[44] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[45] Ilya Mironov, Kunal Talwar, and Li Zhang. 2019. R\'enyi Differential Privacy of the Sampled Gaussian Mechanism. *arXiv preprint arXiv:1908.10530* (2019).

[46] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. 2016. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755* (2016).

[47] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. 2018. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908* (2018).

[48] Matthias Paulik, Matt Seigel, Henry Mason, Dominic Telaar, Joris Kluivers, Rogier van Dalen, Chi Wai Lau, Luke Carlson, Filip Granqvist, Chris Vandevelde, et al. 2021. Federated evaluation and tuning for on-device personalization: System design & applications. *arXiv preprint arXiv:2102.08503* (2021).

[49] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. 2019. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445* (2019).

[50] Md Atiqur Rahman, Tanzila Rahman, Robert Laganière, Noman Mohammed, and Yang Wang. 2018. Membership Inference Attack against Differentially Private Deep Learning Model. *Trans. Data Priv.* 11, 1 (2018), 61–79.

[51] Jayanth Regatti, Hao Chen, and Abhishek Gupta. 2020. Bygars: Byzantine sgd with arbitrary number of attackers. *arXiv preprint arXiv:2006.13421* (2020).

[52] Xuebin Ren, Liang Shi, Weiren Yu, Shusen Yang, Cong Zhao, and Zongben Xu. 2022. LDP-IDS: Local Differential Privacy for Infinite Data Streams. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 1064–1077. https://doi.org/10.1145/3514221.3526190

[53] Benjamin IP Rubinstein, Blaine Nelson, Ling Huang, Anthony D Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and J Doug Tygar. 2009. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*. 1–14.

[54] Virat Shejwalkar and Amir Houmansadr. 2021. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS*.

[55] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.

[56] Adam Smith, Abhradeep Thakurta, and Jalaj Upadhyay. 2017. Is interaction necessary for distributed private learning?. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 58–77.

[57] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. 2013. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*. IEEE, 245–248.

[58] tfp [n. d.]. TensorFlow Privacy Git repository. https://github.com/tensorflow/privacy.

[59] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. 2020. LDP-Fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*. 61–66.

[60] Chenghong Wang, Johes Bater, Kartik Nayak, and Ashwin Machanavajjhala. 2022. IncShrink: Architecting Efficient Outsourced Databases using Incremental MPC and Differential Privacy. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 818–832. https://doi.org/10.1145/3514221.3526151

[61] Di Wang, Changyou Chen, and Jinhui Xu. 2019. Differentially private empirical risk minimization with non-convex loss functions. In *International Conference on Machine Learning*. PMLR, 6526–6535.

[62] Di Wang, Marco Gaboardi, Adam Smith, and Jinhui Xu. 2020. Empirical risk minimization in the non-interactive local model of differential privacy. *Journal of machine learning research* 21, 200 (2020).

[63] Di Wang, Lijie Hu, Huanyu Zhang, Marco Gaboardi, and Jinhui Xu. 2019. Estimating smooth GLM in non-interactive local differential privacy model with public unlabeled data. *arXiv preprint arXiv:1910.00482* (2019).

[64] Di Wang, Minwei Ye, and Jinhui Xu. 2017. Differentially private empirical risk minimization revisited: Faster and more general. *Advances in Neural Information Processing Systems* 30 (2017).

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

[66] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3454–3469.

[67] Wikipedia contributors. 2022. 68–95–99.7 rule — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=68%E2%80%9395%E2%80%9399.7_rule&oldid=1097113055 [Online; accessed 24-July-2022].

[68] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).

[69] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. 2020. Fall of empires: Breaking Byzantine-tolerant SGD by inner product manipulation. In *Uncertainty in Artificial Intelligence*. PMLR, 261–270.

[70] Yuexiang Xie, Zhen Wang, Daoyuan Chen, Dawei Gao, Liuyi Yao, Weirui Kuang, Yaliang Li, Bolin Ding, and Jingren Zhou. 2022. FederatedScope: A Comprehensive and Flexible Federated Learning Platform via Message Passing. *arXiv preprint arXiv:2204.05011* (2022).

[71] Xiaodong Yang, Huishuai Zhang, Wei Chen, and Tie-Yan Liu. 2022. Normalized/Clipped SGD with Perturbation for Differentially Private Non-Convex Optimization. *arXiv e-prints* (2022), arXiv–2206.

[72] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*. PMLR, 5650–5659.

[73] Xinwei Zhang, Xiangyi Chen, Mingyi Hong, Zhiwei Steven Wu, and Jinfeng Yi. 2021. Understanding Clipping for Federated Learning: Convergence and Client-Level Differential Privacy. *arXiv preprint arXiv:2106.13673* (2021).

[74] Qinqing Zheng, Jinshuo Dong, Qi Long, and Weijie Su. 2020. Sharp Composition Bounds for Gaussian Differential Privacy via Edgeworth Expansion. In *International Conference on Machine Learning*. PMLR, 11420–11435.

[75] Banghua Zhu, Lun Wang, Qi Pang, Shuai Wang, Jiantao Jiao, Dawn Song, and Michael I Jordan. 2022. Byzantine-Robust Federated Learning with Optimal Statistical Rates and Privacy Guarantees. *arXiv preprint arXiv:2205.11765* (2022).

[76] Heng Zhu and Qing Ling. 2022. Bridging Differential Privacy and Byzantine-Robustness via Model Aggregation. *arXiv preprint arXiv:2205.00107* (2022).

[77] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in Neural Information Processing Systems* 32 (2019).

[78] Yuqing Zhu and Yu-Xiang Wang. 2019. Poission subsampled rényi differential privacy. In *International Conference on Machine Learning*. PMLR, 7634–7642.
[79] Yuqing Zhu, Xiang Yu, Manmohan Chandraker, and Yu-Xiang Wang. 2020. Private-knn: Practical differential privacy for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11854–11862.

Received October 2022; revised January 2023; accepted February 2023