CPFed: Communication-Efficient and Privacy-Preserving Federated Learning

Rui Hu, *Student Member, IEEE,* Yanmin Gong, *Member, IEEE,* and Yuanxiong Guo, *Senior Member, IEEE*

***Abstract*—Federated learning is a machine learning setting where a set of edge devices iteratively train a model under the orchestration of a central server, while keeping all data locally on edge devices. In each iteration of federated learning, edge devices perform computation with their local data, and the local computation results are then uploaded to the server for model update. During this process, the challenges of privacy leakage and communication overhead arise due to the extensive information exchange between edge devices and the server. In this paper, we develop CPFed, a communication-efficient and privacy-preserving federated learning method, to solve the above challenges. CPFed integrates three key components: (1) periodic averaging where local computation results at edge devices are only periodically averaged at the server; (2) Gaussian mechanism where edge devices randomly perturb their local computation results before sending the results to the server; and (3) secure aggregation where the perturbed local computation results are homomorphically encrypted before being sent to the server. CPFed can address both the communication efficiency and privacy leakage challenges in federated learning while achieving high model accuracy. We provide an end-to-end privacy guarantee of CPFed and analyze its theoretical convergence rates for both convex and non-convex models. Through extensive numerical experiments on real-world datasets, we demonstrate the effectiveness and efficiency of our proposed method.**

I. Introduction

With the development of Internet-of-Things (IoT) technologies, smart devices with built-in sensors, Internet connectivity, and programmable computation capability have proliferated and generated huge volumes of data at the network edge over the past few years. These data can be collected and analyzed to build machine learning models that enable a wide range of intelligent services, such as personal fitness tracking [1], traffic monitoring [2], smart home security [3], and renewable energy integration [4]. However, data are often sensitive in many services, like the heart rate monitored by smart watches, and can leak a lot of personal information about the users. Due to the privacy concern, users would not be willing to share their data, prohibiting the deployment of these intelligent services. *Federated Learning* is a novel machine learning paradigm where a group of edge devices collaboratively learn a shared model under the orchestration of a central server without sharing their training data. It mitigates many of the privacy risks resulting from the traditional, centralized machine learning paradigm, and has received significant attention recently.

Although promising, federated learning faces several challenges, among which communication overhead is a major one [5]. Specifically, at each iteration of federated learning, edge devices download the shared model from the server and compute updates to it using their own datasets, and then these updates will be gathered by the server to renew the shared model. Although only model updates are transmitted between edge devices and the server instead of the raw data, such updates could contain hundreds of millions of parameters in modern machine learning models such as deep neural networks, resulting in high bandwidth usage per iteration. Moreover, many federated learning schemes require many iterations to achieve a high model accuracy, and hence the communication of the whole training process is expensive. Since most edge devices are resource-constrained, the bandwidth between the server and edge devices is rather limited, especially during up-link transmissions. Therefore, it is crucial to make federated learning communication-efficient.

Besides communication efficiency, privacy leakage is another core challenge in federated learning [5]. Although in federated learning edge devices keep their data locally and only exchange ephemeral model updates which contain less information than raw data, this is not sufficient to guarantee data privacy. For example, by observing the model updates from an edge device, it is possible for the adversary to recover the private dataset in that device using reconstruction attack [6] or infer whether a sample is in the dataset of that device using membership inference attack [7]. Especially, if the server is not fully trusted, it can easily infer the private information of edge devices from the received model updates during the training by employing existing attack methods. Therefore, how to protect against those advanced privacy attacks and provide rigorous privacy guarantee for each participant in federated learning without a fully trusted server is challenging and needs to be addressed.

In order to motivate and retain edge devices in federated learning, it is desirable to achieve both communication efficiency and data privacy guarantee. Many prior efforts have considered either communication efficiency [8], [9], [10], [11] or privacy [12], [13] in federated learning, but not both. In this paper, we propose a novel distributed learning scheme called **C**ommunication-efficient and **P**rivacy-preserving **Fed**erated learning (CPFed) that both reduces communication cost and provides formal privacy guarantee without assuming a fully trusted server. To save the communication cost, we reduce the number of communications from two perspectives. We limit the number of participating devices per iteration through device selection, and then decrease the number of iterations

R. Hu and Y. Gong are with the Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX, 78249 USA (e-mail: {rui.hu, yanmin.gong}@utsa.edu).

Y. Guo is with the Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX, 78249 USA (e-mail: yuanxiong.guo@utsa.edu).

via allowing selected devices to perform multiple iterations before sending their computation results out. Utilizing client selection and more local computation can significantly save the communication cost, however, it is hard to provide the rigorous convergence analysis and will have an impact on the privacy guarantee. To preserve the privacy of devices without a fully trusted server, we leverage the concept of local differential privacy and ask each device to add random noise to perturb its local computation results before transmission. However, combined with our communication-reduction strategy directly, local differential privacy adds too much noise to the model updates and leads to low model accuracy. In our proposed scheme, we use a secure aggregation protocol with low communication overhead to aggregate devices' model updates, which improves the model accuracy under the same differential privacy guarantee.

In summary, the main contributions of this paper are as follows.

- We propose a novel federated learning scheme called CPFed for communication-efficient and differentially private learning over distributed data without a fully trusted server.
- CPFed reduces the number of communications by allowing partial devices to participate the training at each iteration and communicate with the server periodically.
- Without much degradation of the model accuracy, CPFed rigorously protects the data privacy of each device by integrating secure aggregation and differential privacy techniques.
- Instead of providing only per-iteration differential privacy guarantee, we tightly account the end-to-end privacy loss of CPFed using zero-concentrated differential privacy.
- We perform convergence analysis of CPFed for both strongly-convex and non-convex loss functions and conduct extensive evaluations based on the real-world dataset.

The rest of the paper is organized as follows. Related work and background on privacy notations used in this paper are described in Section VIII and Section II, respectively. Section III introduces the system setting and the problem formulation. Section IV presents our CPFed learning scheme and the corresponding algorithm. The privacy guarantee and convergence property of CPFed is rigorously analyzed in Section V and Section VI, respectively. Finally, Section VII shows the evaluation results based on the real-world dataset, and Section IX concludes the paper.

II. Preliminaries

In what follows, we briefly describe the basics of differential privacy and their properties. Differential privacy (DP) is a rigorous notion of privacy and has become the de-facto standard for measuring privacy risk.

(ϵ, δ)-Differential Privacy [14]. (ϵ, δ)-DP is the classic DP notion with the following definition:

Definition 1 ((ϵ, δ)-DP). *A randomized algorithm $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{O}$ is (ϵ, δ)-differentially private if for any two adjacent datasets $D, D' \subseteq \mathcal{D}$ that differ in at most one data sample and any subset of outputs $\mathcal{S} \subseteq \mathcal{O}$, it satisfies that:*

$$\Pr[\mathcal{M}(D) \in \mathcal{S}] \leq e^{\epsilon} \Pr[\mathcal{M}(D') \in \mathcal{S}] + \delta. \tag{1}$$

The above definition reduces to ϵ-DP when $\delta = 0$. Here the parameter ϵ is also called the privacy budget. Given any function f that maps a dataset $D \in \mathcal{D}$ into a scalar $o \in \mathbb{R}$, we can achieve (ϵ, δ)-DP by adding Gaussian noise $\mathcal{N}(0, \sigma^2)$ to the output scalar o, where the noise magnitude σ is proportional to the sensitivity of f, given as $\Delta_2(f) := \|f(D) - f(D')\|_2$.

ρ-Zero-Concentrated Differential Privacy [15]. ρ-zero-concentrated differential privacy (ρ-zCDP) is a relaxed version of (ϵ, δ)-DP. zCDP has a tight composition bound and is more suitable to analyze the end-to-end privacy loss of iterative algorithms. To define zCDP, we first define the privacy loss. Given an output $o \in \mathcal{R}$, the privacy loss Z of the mechanism $\mathcal{M}$ is a random variable defined as:

$$Z := \log \frac{\Pr[\mathcal{M}(D) = o]}{\Pr[\mathcal{M}(D') = o]}. \tag{2}$$

zCDP imposes a bound on the moment generating function of the privacy loss Z. Formally, a randomized mechanism $\mathcal{M}$ satisfies ρ-zCDP if for any two adjacent datasets $D, D' \subseteq \mathcal{D}$, it holds that for all $\alpha \in (1, \infty)$,

$$\mathbb{E}[e^{(\alpha-1)Z}] \leq e^{(\alpha-1)\rho}. \tag{3}$$

Here, (3) requires the privacy loss Z to be concentrated around zero, and hence it is unlikely to distinguish D from D' given their outputs. zCDP has the following properties [15]:

Lemma 1. *Let $f : x \to \mathbb{R}$ be any real-valued function with sensitivity $\Delta_2(f)$, then the Gaussian mechanism, which returns $f(x) + \mathcal{N}(0, \sigma^2)$, satisfies $\Delta_2(f)^2/(2\sigma^2)$-zCDP.*

Lemma 2. *Suppose two mechanisms satisfy ρ_1-zCDP and ρ_2-zCDP, then their composition satisfies $\rho_1 + \rho_2$-zCDP.*

Lemma 3. *If $\mathcal{M}$ is a mechanism that provides ρ-zCDP, then $\mathcal{M}$ is $(\rho + 2\sqrt{\rho \log(1/\delta)}, \delta)$-DP for any $\delta > 0$.*

III. System Modeling and Problem Formulation

A. Federated Learning System

Consider a federated learning setting that consists of a central server and n clients which are able to communicate with the server. Each of n clients has a local dataset $D_i = \{\xi_1^i, \ldots, \xi_m^i\}$, a collection of m datapoints from its edge device. The clients want to collaboratively learn a shared model $\boldsymbol{\theta} \in \mathbb{R}^d$ using their data under the orchestration of the central server. Due to the privacy concern and high latency of uploading all local datapoints to the server, federated learning allows clients to train the model while keeping their data locally. Specifically, the shared model $\boldsymbol{\theta}$ is learned by minimizing the overall empirical risk of the loss on the union of all local datasets, that is,

$$\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) := \frac{1}{n} \sum_{i=1}^{n} f_i(\boldsymbol{\theta}) \text{ with } f_i(\boldsymbol{\theta}) := \frac{1}{m} \sum_{\xi \in D_i} l(\boldsymbol{\theta}, \xi). \tag{4}$$

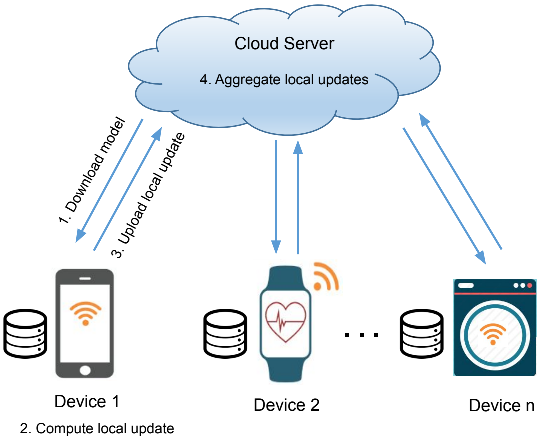

Fig. 1. System architecture of federated learning.

Here, f_i represents the local objective function of client i, $l(\boldsymbol{\theta};\xi)$ is the loss of the model $\boldsymbol{\theta}$ at a datapoint ξ sampled from local dataset D_i.

In federated learning, the central server is responsible for coordinating the training process across all clients and maintaining the shared model $\boldsymbol{\theta}$. The system architecture of federated learning is shown in Fig. 1. At the beginning of each iteration, clients download the shared model $\boldsymbol{\theta}$ from the server and compute local updates on $\boldsymbol{\theta}$ using their local datasets. Then, each client uploads its computed result to the server, where the received local results are aggregated to update the shared model $\boldsymbol{\theta}$. This procedure repeats until certain convergence criteria are satisfied.

B. Threat Model

We assume that the adversary here can be the "honest-but-curious" central server or clients in the system. The central server will honestly follow the designed training protocol, but are curious about the client's private data and may infer it from the shared message. Furthermore, some malicious clients could collude with the central server or each other to infer private information about a specific client. Besides, the adversary can also be the passive outside attacker. These attackers can eavesdrop all shared messages in the execution of the training protocol but will not actively inject false messages into or interrupt the message transmission. Malicious clients who, for instance, may launch data pollution attacks by lying about their private datasets or returning incorrect computed results to disrupt the training process are out of the scope of this paper and will be left as our future work.

C. Design Goals

Our goal is to design a scheme that enables multiple clients to jointly learn an accurate model for a given machine learning task with low communication cost. Moreover, the differential privacy guarantee for each client should be provided without sacrificing much accuracy of the trained model.

IV. Proposed CPFed Scheme

In this section, we propose our method called CPFed to address the communication overhead and privacy leakage issue in federated learning. In what follows, we first describe how to save the communication cost of training the model, using the periodic averaging method. Then we discuss how to preserve the data privacy of each client in the system with differential privacy. Next, we improve the accuracy of our method with secure aggregation. Finally, we present the overall algorithm that captures all these components.

A. Improving Communication Efficiency with Periodic Averaging

In the vanilla distributed stochastic gradient descent (SGD) approach that solves Problem (4), the server collects the gradients of local objectives from all clients and updates the shared model using a gradient descent iteration given by

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \eta \sum_{i=1}^{n} \nabla f_i(\boldsymbol{\theta}^t), \tag{5}$$

where $\boldsymbol{\theta}^t$ represents the shared model at iteration t, η is the stepsize, and $\nabla f_i(\boldsymbol{\theta}^t) := \frac{1}{m}\sum_{\xi\in D_i} \nabla l(\boldsymbol{\theta}^t, \xi)$ represents the gradient of local objective function f_i based on the local dataset D_i.

Above distributed-SGD method, however, requires many rounds of communication between clients and the server [5]. Federated learning systems are potentially comprised of a massive number of devices, e.g., millions of smartphones, and communication can be slower than local computation by many orders of magnitude. Therefore, a large number of communication rounds will lead to inefficient training. More precisely, assume the number of iterations for training the model is K and at each iteration client $i \in [n]$ shares its local gradient with the server to update the model. Then, the total number of communications is nK.

To save the communication cost, we propose a communication-reduction method which reduces the number of communication round and the involved clients per round simultaneously, as shown in Fig. 2. In our method, the server first selects a bunch of clients uniformly at random and then lets the selected clients perform multiple iterations to minimize the local objectives before sending their local computation results to the server. Specifically, at round t, a set of r clients Ω_t are selected to download the current shared model $\boldsymbol{\theta}^t$ from the server and perform τ local iterations on $\boldsymbol{\theta}^t$. Let $\boldsymbol{\theta}_i^{t,s}$ denote the local model of client i at s-th local iteration of the t-th round. At each local iteration $s = 0, \ldots, \tau-1$, client i updates its model by

$$\boldsymbol{\theta}_i^{t,s+1} = \boldsymbol{\theta}_i^{t,s} - \eta g(\boldsymbol{\theta}_i^{t,s}), \tag{6}$$

where $g(\boldsymbol{\theta}_i^{t,s}) := \frac{1}{B}\sum_{\xi\in X_i} \nabla l(\boldsymbol{\theta}_i^{t,s}, \xi)$ represents the mini-batch stochastic gradient computed based on a batch of B datapoints X_i sampled from the local dataset D_i. Note that when $s = 0$, the local model $\boldsymbol{\theta}_i^{t,s} = \boldsymbol{\theta}^t$ for all clients in Ω_t. After τ local iterations, the selected clients upload their local models to the server where the shared model is updated by

$$\boldsymbol{\theta}^{t+1} = \frac{1}{r}\sum_{i\in\Omega_t} \boldsymbol{\theta}_i^{t,\tau}. \tag{7}$$

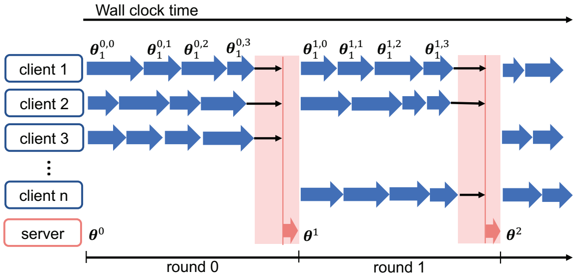

Fig. 2. Illustration of communication-reduction strategies for $\tau = 4$ and $r = 3$. Blue, black, red arrows represent gradient computation, model uploading, and model aggregation respectively.

In this case, each client is selected to communicate to the server with probability r/n and only needs to periodically communicate for K/τ times in total. Hence, the number of communication rounds is reduced by a factor of $r/n\tau$.

B. Preventing Privacy Leakage with Differential Privacy

Aforementioned communication-reduction method is able to prevent the direct information leakage of clients via keeping the raw data locally, however, it could not prevent more advanced attacks [6], [7] that infer private information of local training data by observing the messages communicated between clients and the server. According to our threat model described in Section III-B, clients and the server in the system are "honest-but-curious", and attackers outside the system can eavesdrop the transmitted messages. These attackers are able to obtain the latest shared model $\boldsymbol{\theta}^t$ sent from the server to clients and the local models $\{\boldsymbol{\theta}_i^{t,\tau}\}_{i\in\Omega_t}$ sent from clients to the server, both of which contain private information of clients' training data. Our goal is to prevent the privacy leakage from these two types of messages with differential privacy.

In our setting, a straightforward approach would be the Gaussian mechanism. Specifically, each client $i \in \Omega_t$ adds enough Gaussian noise into the shared information (i.e., the local model to be uploaded) directly before releasing it. In this case, attackers would not be able to learn much about an individual sample in D_i from the received massages. Accordingly, at each local iteration, client $i \in \Omega_t$ updates its local model by

$$\boldsymbol{\theta}_i^{t,s+1} = \boldsymbol{\theta}_i^{t,s} - \eta\left(g(\boldsymbol{\theta}_i^{t,s}) + \mathbf{b}_i^{t,s}\right), \tag{8}$$

where $\mathbf{b}_i^{t,s}$ is the Gaussian noise sampled at s-th local iteration of the t-th round from the distribution $\mathcal{N}(0, \sigma^2\mathbf{1}_d)$. Here, the local model $\boldsymbol{\theta}_i^{t,\tau}$ will preserve a certain level of differential privacy guarantee for client i, which is proportional to the size of noise σ. Due to the post-processing property of differential privacy [14], the sum of local models, i.e., the updated model $\boldsymbol{\theta}^{t+1}$, preserves the same level of differential privacy guarantee for client i as the local model.

C. Improving Model Accuracy with Secure Aggregation

Although differential privacy can be achieved using Gaussian mechanism, the accuracy of the learned model will degrade significantly. At each round of the training, all uploaded local models are exposed to the attacker, leading to a large amount of information leakage. However, we observe that the server only needs to know the average values of the local models. Intuitively, one can reduce the privacy loss of clients by hiding the individual local models and restricting the server to receive only the sum of local models, without disturbing the learning process. This can be achieved via a secure aggregation protocol so that the server can only decrypt the sum of the encrypted local models of selected clients without knowing each client's local model. In the following, we design such a protocol based on secrete sharing, which is efficient in terms of the amortized computation and communication overhead across all communication rounds. The effect of secure aggregation in reducing privacy loss will be rigorously analyzed in Section V.

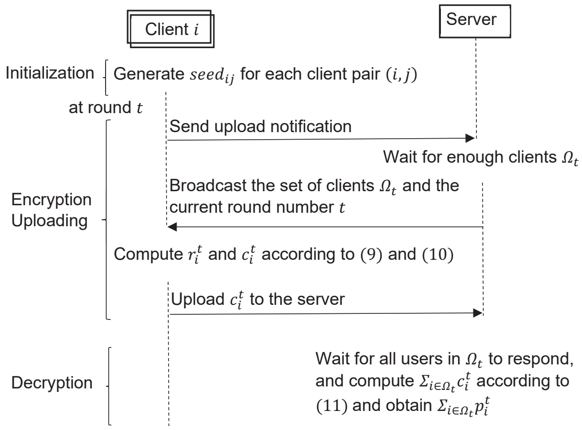
Client i
Server
Initialization
Generate $seed_{ij}$ for each client pair (i, j)
at round t
Send upload notification
Wait for enough clients Ω_t
Encryption Uploading
Broadcast the set of clients Ω_t and the current round number t
Compute r_i^t and c_i^t according to (9) and (10)
Upload c_i^t to the server
Decryption
Wait for all users in Ω_t to respond, and compute $\Sigma_{i\in\Omega_t} c_i^t$ according to (11) and obtain $\Sigma_{i\in\Omega_t} p_i^t$

Fig. 3. Basic protocol for efficient secure aggregation in CPFed.

In our setting, a secure aggregation protocol should be able to 1) hide individual messages for clients, 2) recover the sum of individual messages of a random set of clients at each round, and 3) incur low communication cost for participating clients. Denote by p_i^t the plaintext message of client i (i.e., local model parameters $\boldsymbol{\theta}_i^{t,\tau}$) that needs to be uploaded to the server. Our proposed protocol involves few interactions between clients and the server during each round and consists of the following two main steps:

- *Encryption uploading:* Clients in Ω_t upload their own encrypted local models $\{c_i^t\}_{i\in\Omega_t}$ to the server.
- *Decryption:* The server decrypts the sum of the messages received from clients in Ω_t.

The basic idea of the protocol is to protect the message p_i^t of client i by hiding it with a random number r_i^t in the plaintext space, i.e., $c_i^t = p_i^t + r_i^t$. However, the challenge here is how to remove the random number r_i^t from the received ciphertext at the server part. To this end, we require that all the r_i^t will sum up to 0, i.e., $\sum_{i\in\Omega_t} r_i^t = 0$, which prevents the attacker from recovering each individual message p_i^t but enables the server to recover $\sum_{i\in\Omega_t} p_i^t$. However, this requires the clients to communicate with each other in order to generate such secrets $\{r_i^t\}_{i\in\Omega_t}$, which is inefficient in terms of communication overhead.

To save the communication overhead, we introduce a pseudorandom function (PRF) G here. The PRF G takes a random seed $seed_{i,j}$ that both client i and j agree on during initialization and the round number t, and outputs a different pseudorandom number $G(seed_{i,j}, t)$ at each round. Client i could calculate the shared secret r_{ij}^t without interacting with client j at each round as long as they both use the same seed and round number, and thus each client could calculate r_i^t without interactions. This procedure greatly reduces the amortized communication overhead of our protocol over multiple rounds.

The detailed protocol is depicted in Fig. 3. All clients need to go through an initialization step upon enrollment which involves pairwise communications with all other clients (which can be facilitated by the server) to generate a random seed $seed_{ij}$. After this initialization step, all enrolled clients could upload their messages through the encryption uploading step. At each round, only a subset of selected clients would upload their messages. Clients send a notification signal to the server once they are ready to upload their local models, and the server waits until receiving notifications from enough clients. The server then broadcasts the information Ω_t to all clients in Ω_t. Client $i \in \Omega_t$ would first compute its secret at the current round as follows:

$$r_i^t = \sum_{j \in \Omega_t \setminus \{i\}} \left(r_{ij}^t - r_{ji}^t \right), \tag{9}$$

where $r_{ij}^t = G(seed_{i,j}, t)$ is a secret known by both client i and j. Client i could then generate the ciphertext for p_i^t by

$$c_i^t = p_i^t + r_i^t. \tag{10}$$

In the decryption step, the server receives $\{c_i^t\}_{i \in \Omega_t}$ from all selected clients. The server could then recover the sum of plaintext messages from clients in Ω_t as follows:

$$\begin{aligned} \sum_{i \in \Omega_t} c_i^t &= \sum_{i \in \Omega_t} p_i^t + \sum_{i \in \Omega_t} \sum_{j \in \Omega_t \setminus \{i\}} (r_{ij}^t - r_{ji}^t) \\ &= \sum_{i \in \Omega_t} p_i^t. \end{aligned} \tag{11}$$

Note that in the above protocol, we assume all clients in Ω_t have stable connection to the server. In the rest of the paper, for the ease of expression, we use $E(\boldsymbol{\theta}_i^{t,\tau})$ to denote the encryption of the local model parameters $\boldsymbol{\theta}_i^{t,\tau}$.

D. The Overall Scheme of CPFed

The overall scheme of our CPFed is summarized in Algorithm 1. Our scheme consists of T communication rounds and during each round, a set of clients is selected to perform τ local iterations, which results in $K = T\tau$ iterations in total. More precisely, at each round $t = 0, \ldots, T-1$, the server first picks $r \leq n$ clients uniformly at random which we denote by Ω_t. The server then broadcasts its current shared model $\boldsymbol{\theta}^t$ to all the clients in Ω_t and each client $i \in \Omega_t$ performs τ local iterations using its local dataset $\mathcal{D}_i$ according to (8). Note that clients in Ω_t start with a common model initialization, i.e., $\boldsymbol{\theta}_i^{t,0} = \boldsymbol{\theta}^t$, at the beginning of each round. After τ local iterations, each client in Ω_t uploads an encrypted local model $E(\boldsymbol{\theta}_i^{t,\tau})$ to the server. The server finally aggregates the encrypted messages to compute the next shared model, and the procedure repeats for T rounds.

Algorithm 1 CPFed Algorithm

```
Input: number of rounds $T$, round length $\tau$, number of
    selected clients $r$, stepsize $\eta$
for $t = 0$ to $T - 1$ do
    Server uniformly selects $r$ clients denoted by $\Omega_t$;
    Server broadcasts $\boldsymbol{\theta}^t$ to all clients in $\Omega_t$;
    for all clients in $\Omega_t$ in parallel do
        $\boldsymbol{\theta}_i^{t,0} \leftarrow \boldsymbol{\theta}^t$;
        for $s = 0$ to $\tau - 1$ do
            Randomly sample a batch of datapoints $X_i$ with
            size $B$ from the local dataset $D_i$;
            $g(\boldsymbol{\theta}_i^{t,s}) \leftarrow \frac{1}{B} \sum_{\xi \in X_i} \nabla l(\boldsymbol{\theta}_i^{t,s}, \xi)$;
            $\boldsymbol{\theta}_i^{t,s+1} \leftarrow \boldsymbol{\theta}_i^{t,s} - \eta \left( g(\boldsymbol{\theta}_i^{t,s}) + \mathcal{N}(0, \sigma^2 \mathbf{1}_d) \right)$;
        end for
        Generate encrypted local model $c_i^t = E(\boldsymbol{\theta}_i^{t,\tau})$ using
        the secure aggregation protocol and send it to the
        server;
    end for
    Server updates $\boldsymbol{\theta}^{t+1} \leftarrow \frac{1}{r} \sum_{i \in \Omega_t} c_i^t$.
end for
```

V. Privacy Analysis

As we mentioned before, our goal of using differential privacy techniques is to prevent the attacker outside the system or the "honest-but-curious" server and clients from learning sensitive information about the local data of a client. Under the secure aggregation protocol, the local model is encrypted and the attacker will only obtain the sum of local models. Thus, as long as the sum of local models is differentially private, we can prevent the attacks launched by the attacker.

Instead of using DP directly, we use zCDP to tightly account the end-to-end privacy loss of CPFed and then convert it to a DP guarantee. Accordingly, we first show that the sum of local models achieves zCDP at each round, then we account the overall zCDP guarantee after T rounds. To do so, we compute the sensitivity of the stochastic gradient of client i at a single local iteration (as given in Corollary 1) and the sensitivity of $\sum_{i \in \Omega_t} \boldsymbol{\theta}_i^{t,\tau}$ (as given in Lemma 4).

Corollary 1. *The sensitivity of the stochastic gradient $g(\boldsymbol{\theta}_i^{t,s})$ of client i at the s-th local iteration of the t-th round is bounded by $2L/B$.*

Proof: For client i, given any two neighboring datasets X_i and X_i' of size B that differ only in the j-th data sample, the sensitivity of the stochastic gradient computed at each local iteration in Algorithm 1 can be computed as

$$\begin{aligned} &\|g(\boldsymbol{\theta}_i^{t,s}; X_i) - g(\boldsymbol{\theta}_i^{t,s}; X_i')\|_2 \\ &\qquad = \frac{1}{B} \|\nabla l(\boldsymbol{\theta}_i^{t,s}; \xi_j) - \nabla l(\boldsymbol{\theta}_i^{t,s}; \xi_j')\|_2. \end{aligned}$$

Since the loss function $l(\cdot)$ is L-Lipschitz continuous, the sensitivity of $g(\boldsymbol{\theta}_i^{t,s})$ can be estimated as $\Delta_2(g(\boldsymbol{\theta}_i^{t,s})) \leq 2L/B$. ■

Lemma 4. *The sensitivity of the sum of uploaded local models $\sum_{i \in \Omega_t} \boldsymbol{\theta}_i^{t,\tau}$ at round t is bounded by $2\eta\tau L/B$.*

Proof: Without adding noise, the local model of client $i \in \Omega_t$ after τ local iterations at round t can be written as

$$\boldsymbol{\theta}_i^{t,\tau} = \boldsymbol{\theta}_i^{t,0} - \eta g(\boldsymbol{\theta}_i^{t,0}) - \cdots - \eta g(\boldsymbol{\theta}_i^{t,\tau-1}). \tag{12}$$

According to the sensitivity of $g(\boldsymbol{\theta}_i^{t,s})$ given in Corollary 1, we have that

$$\begin{aligned}\Delta_2(\boldsymbol{\theta}_i^{t,\tau}) = \eta \Big\| & g(\boldsymbol{\theta}_i^{t,0}; X_i^{t,0}) - g(\boldsymbol{\theta}_i^{t,0}; X_i^{t,0'}) + \ldots \\ & + g(\boldsymbol{\theta}_i^{t,\tau-1}; X_i^{t,\tau-1}) - g(\boldsymbol{\theta}_i^{t,\tau-1}; X_i^{t,\tau-1'}) \Big\| \\ \leq & \frac{2\eta\tau L}{B}.\end{aligned}$$

Now, it is easy to find that the sensitivity of the sum of uploaded local models is $\Delta_2(\sum_{i\in\Omega_t} \boldsymbol{\theta}_i^{t,\tau}) = \Delta_2(\boldsymbol{\theta}_i^{t,\tau}) \leq 2\eta\tau L/B$. ■

By Lemma 1 and 4, we can obtain the zCDP guarantee at each round if we can measure the magnitude of noise added on the sum of local models. According to Algorithm 1, Gaussian noise is added to the stochastic gradient. Thus, after τ local iterations, client $i \in \Omega_t$ obtains a noisy local model that is

$$\begin{aligned}\boldsymbol{\theta}_i^{t,\tau} &= \boldsymbol{\theta}_i^{t,0} - \eta[g(\boldsymbol{\theta}_i^{t,0}) + \mathcal{N}(0, \sigma^2 \mathbf{1}_d)] - \eta[g(\boldsymbol{\theta}_i^{t,1}) \\ &\quad + \mathcal{N}(0, \sigma^2 \mathbf{1}_d)] - \cdots - \eta[g(\boldsymbol{\theta}_i^{t,\tau-1}) + \mathcal{N}(0, \sigma^2 \mathbf{1}_d)] \\ &= \boldsymbol{\theta}_i^{t,0} - \eta g(\boldsymbol{\theta}_i^{t,0}) - \cdots - \eta g(\boldsymbol{\theta}_i^{t,\tau-1}) + \mathcal{N}(0, \tau\eta^2\sigma^2 \mathbf{1}_d).\end{aligned}$$

The server will receive r such local models at each round and each of them contains an independent Gaussian noise drawn from the distribution $\mathcal{N}(0, \tau\eta^2\sigma^2\mathbf{1}_d)$. Therefore, we have that

$$\sum_{i\in\Omega_t} \left(\boldsymbol{\theta}_i^{t,\tau} + \mathcal{N}(0, \tau\eta^2\sigma^2\mathbf{1}_d)\right) = \sum_{i\in\Omega_t} \boldsymbol{\theta}_i^{t,\tau} + \mathcal{N}(0, r\tau\eta^2\sigma^2\mathbf{1}_d),$$

where we can see the magnitude of Gaussian noise added on the sum of uploaded local models is $\sqrt{r\tau}\eta\sigma$. By Lemma 1 and Lemma 4, each round of Algorithm 1 achieves $\frac{2\tau L^2}{rB^2\sigma^2}$-zCDP for each selected client. Finally, we compute the overall privacy guarantee for a client after T rounds of training and give the (ϵ, δ)-DP guarantee in Theorem 1.

Theorem 1. *If the Gaussian noise $\mathbf{b}_i^{t,s}$ in Algorithm 1 are sampled from $\mathcal{N}(0, \sigma^2\mathbf{1}_d)$, then Algorithm 1 achieves (ϵ, δ)-DP for each client in the system after T rounds of training, where*

$$\epsilon = \frac{2T\tau L^2}{nB^2\sigma^2} + 2\sqrt{\frac{2T\tau L^2}{nB^2\sigma^2}\log\frac{1}{\delta}}. \tag{13}$$

Proof: It is proved that each round of Algorithm 1 achieves $\frac{2\tau L^2}{rB^2\sigma^2}$-zCDP for the client in Ω_t. Due to the client selection, not all clients will upload their models to the server at round t. If their models are not sent out, they do not lose their privacy at that round. Indeed, every client in the system only participates the training with probability r/n at each round. Therefore, by Lemma 2, the overall zCDP guarantee of each client in the system after T rounds of training is $\frac{2T\tau L^2}{nB^2\sigma^2}$. Theorem 1 then follows by Lemma 3. ■

VI. Convergence Analysis

In this section, we present our main theoretical results on the convergence properties of the proposed CPFed algorithm. We first consider strongly convex loss functions and state the convergence rate of the CPFed for such losses in Theorem 2. Then, in Theorem 3, we present the convergence rate of the CPFed for non-convex losses.

Before stating our results, we give some assumptions for both convex and non-convex cases.

Assumption 1 (Smoothness)**.** *The loss function l is L-smooth, i.e., for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, we have $l(\mathbf{y}) \leq l(\mathbf{x}) + \langle \nabla l(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{L}{2}\|\mathbf{y} - \mathbf{x}\|^2$.*

Assumption 2 (Unbiased gradients)**.** *The local stochastic gradients $\nabla l(\mathbf{x}, \xi)$ with $\xi \in D_i$ are unbiased, i.e., for any $\mathbf{x} \in \mathbb{R}^d$ and $i \in [n]$, $\mathbb{E}\left[\nabla l(\mathbf{x}, \xi)\right] = \nabla f_i(\mathbf{x})$.*

Assumption 3 (Bounded divergence)**.** *The local stochastic gradients will not diverge a lot from the exact gradient, i.e., for any $\mathbf{x} \in \mathbb{R}^d$ and $i \in [n]$, $\mathbb{E}\left[\|g(\mathbf{x}) - \nabla f(\mathbf{x})\|^2\right] \leq \beta$.*

The condition in Assumption 1 implies that the local loss function f_i and the global loss function f are L-smooth. The condition on the bias of stochastic gradients in Assumption 2 is customary for SGD algorithms. Assumption 3 ensures that the divergence between local stochastic gradients is bounded. This condition is assumed in most federated learning settings. Under Assumption 2 and 3, we obtain Lemma 5 which bounds the divergence of local gradients. Note that in the rest of this paper we consider the stochastic gradient with batch size $B = 1$, but it is easy to extend our conclusions to the stochastic gradient descent with larger batch size.

Lemma 5. *The variance of local stochastic gradient is bounded, and the local gradient will not diverge a lot from the exact gradient, i.e., for any $\mathbf{x} \in \mathbb{R}^d$ and $i \in [n]$:*

(i) $\mathbb{E}\left[\|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\|^2\right] \leq \frac{\beta}{m}$;

(ii) $\mathbb{E}\left[\|g(\mathbf{x}) - \nabla f_i(\mathbf{x})\|^2\right] \leq \beta - \frac{\beta}{m}$.

Proof: In the lemma, (i) is an immediate result of Assumption 2 and 3 together with the fact that the noise of the stochastic gradient scales down with the sample size. To prove (ii), we use the fact that

$$\begin{aligned}\mathbb{E}\left[\|g(\mathbf{x}) - \nabla f_i(\mathbf{x})\|^2\right] = {} & \mathbb{E}\left[\|g(\mathbf{x}) - \nabla f(\mathbf{x})\|^2\right] \\ & - \mathbb{E}\left[\|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\|^2\right],\end{aligned}$$

and therefore (ii) is obvious. ■

To prove the convergence of CPFed, we first represent the update rule of CPFed in a general manner. In Algorithm 1, the total number of iterations is K, i.e., $K = T\tau$. At iteration k where $k = t\tau + s$, each client i evaluates the stochastic gradient $g(\boldsymbol{\theta}_i^k)$ based on its local dataset and updates current model $\boldsymbol{\theta}_i^k$. Thus, n clients have different versions $\boldsymbol{\theta}_1^k, \ldots, \boldsymbol{\theta}_n^k$ of the model. After τ local iterations, clients upload their encrypted local models to the server and then update their models with the updated shared model downloaded from the server, i.e., $\frac{1}{r}\sum_{i\in\Omega_k} S(\boldsymbol{\theta}_i^k)$ with $(k \mod \tau = 0)$, where $\Omega_k = \Omega_t$ since Ω_t does not change during the local iteration.

Now, we can present a virtual update rule that captures all the features in Algorithm 1. Define matrices $\boldsymbol{\Theta}^k, \mathbf{G}^k, \mathbf{B}^k \in$

$\mathbb{R}^{d\times n}$ for $k=0,\dots,K-1$ that concatenate all local models, gradients and noises:

$$\begin{aligned}\boldsymbol{\Theta}^k &:= \left[\boldsymbol{\theta}_1^k, \boldsymbol{\theta}_2^k, \dots, \boldsymbol{\theta}_n^k\right],\\ \mathbf{G}^k &:= \left[g(\boldsymbol{\theta}_1^k), g(\boldsymbol{\theta}_2^k), \dots, g(\boldsymbol{\theta}_n^k)\right],\\ \mathbf{B}^k &:= \left[\mathbf{b}_1^k, \mathbf{b}_2^k, \dots, \mathbf{b}_n^k\right].\end{aligned}$$

If client i is not selected to upload its model at iteration k, $\boldsymbol{\theta}_i^k = g(\boldsymbol{\theta}_i^k) = \mathbf{b}_i^k = \mathbf{0}_d$. Besides, define matrix $\mathbf{J}^{\Omega_k} \in \mathbb{R}^{n\times n}$ with element $\mathbf{J}_{i,j}^{\Omega_k} = \frac{1}{r}$ if $i \in \Omega_k$ and $\mathbf{J}_{i,j}^{\Omega_k} = 0$ otherwise. Unless otherwise stated, $\mathbf{1}^k \in \mathbb{R}^n$ is a column vector of size n with element $\mathbf{1}_i^k = 1$ if $i \in \Omega_k$ and $\mathbf{1}_i^k = 0$ otherwise. To capture periodic averaging, we define $\mathbf{J}^k$ as

$$\mathbf{J}^k := \begin{cases}\mathbf{J}^{\Omega_k}, k \bmod \tau = 0\\ \mathbf{1}_{n\times n}, \text{ otherwise.}\end{cases}$$

where $\mathbf{1}_{n\times n}$ is a $n\times n$ identity matrix. Then a general update rule of CPFed can be expressed as follows:

$$\boldsymbol{\Theta}^{k+1} = \left(\boldsymbol{\Theta}^k - \eta(\mathbf{G}^k + \mathbf{B}^k)\right)\mathbf{J}^k. \tag{14}$$

Note that the secure aggregation does not change the sum of local models. Multiplying $(1/r)\mathbf{1}^k$ on both sides of (14), we have

$$\frac{\boldsymbol{\Theta}^{k+1}\mathbf{1}^k}{r} = \frac{\boldsymbol{\Theta}^k\mathbf{1}^k}{r} - \eta\left(\frac{\mathbf{G}^k\mathbf{1}^k}{r} + \frac{\mathbf{B}^k\mathbf{1}^k}{r}\right). \tag{15}$$

Then define the averaged model at iteration k as

$$\hat{\boldsymbol{\theta}}^k := \frac{\boldsymbol{\Theta}^k\mathbf{1}^k}{r} = \frac{1}{r}\sum_{i\in\Omega_k}\boldsymbol{\theta}_i^k.$$

After rewriting (15), one yields

$$\hat{\boldsymbol{\theta}}^{k+1} = \hat{\boldsymbol{\theta}}^k - \eta\left(\frac{1}{r}\sum_{i\in\Omega_k} g(\boldsymbol{\theta}_i^k) + \mathbf{b}_i^k\right). \tag{16}$$

Since client is picked at random to preform updating at each round, and $g(\boldsymbol{\theta}_i^k)$ is the stochastic gradient computed on one data sample $\xi \in \mathcal{D}_i$. We can see that the randomness in our federated learning system comes from the client selection, stochastic gradient, and Gaussian noise. In the following, we bound the expectation of several intermediate random variables, which we denote by $\mathbb{E}_{\{\Omega_k,\xi,\mathbf{b}_i^k|i\in[n]\}}[\cdot]$. For ease of expression, we use $\mathbb{E}[\cdot]$ instead of $\mathbb{E}_{\{\Omega_k,\xi,\mathbf{b}_i^k|i\in[n]\}}[\cdot]$ in the rest of the paper, unless otherwise stated. Specifically, as given in Lemma 6 and Lemma 7, we compute the upper bound of the expectation of the perturbed stochastic gradients and the network error that captures the divergence between local models and the averaged model.

Lemma 6. *The expectation and variance of the averaged perturbed stochastic gradients at iteration k are*

$$\mathbb{E}\left[\frac{1}{r}\sum_{i\in\Omega_k}\left(g(\boldsymbol{\theta}_i^k) + \mathbf{b}_i^k\right)\right] = \frac{1}{n}\sum_{i=1}^n \nabla f_i(\boldsymbol{\theta}_i^k), \tag{17}$$

and

$$\mathbb{E}\left[\left\|\frac{1}{r}\sum_{i\in\Omega_k}\left(g(\boldsymbol{\theta}_i^k) + \mathbf{b}_i^k\right) - \frac{1}{n}\sum_{i=1}^n \nabla f_i(\boldsymbol{\theta}_i^k)\right\|^2\right] \leq d\sigma^2 + \beta - \frac{\beta}{m} + \frac{4(n-r)^2}{n^3}\sum_{i=1}^n \left\|\nabla f_i(\boldsymbol{\theta}_i^k)\right\|^2. \tag{18}$$

Proof: To simplify the notation, we set $\mathcal{G}^k := \frac{1}{r}\sum_{i\in\Omega_k}\left(g(\boldsymbol{\theta}_i^k) + \mathbf{b}_i^k\right)$. Given Assumption 2, we have

$$\begin{aligned}\mathbb{E}\left[\mathcal{G}^k\right] &= \sum_{\substack{\Omega\in[n],\\|\Omega|=r}} P_r(\Omega_k = \Omega)\left(\frac{1}{r}\sum_{i\in\Omega_k}\mathbb{E}\left[g(\boldsymbol{\theta}_i^k) + \mathbf{b}_i^k\right]\right)\\ &= \frac{1}{r}\frac{1}{\binom{n}{r}}\binom{n-1}{r-1}\sum_{i=1}^n \nabla f_i(\boldsymbol{\theta}_i^k) = \frac{1}{n}\sum_{i=1}^n \nabla f_i(\boldsymbol{\theta}_i^k).\end{aligned}$$

Here, $\mathbb{E}[\mathbf{b}_i^k] = \mathbf{0}_d$ since $\mathbf{b}_i^k \sim \mathcal{N}(0, \sigma^2\mathbf{1}_d)$. To derive the variance of $\mathcal{G}^k$, we use the conclusions in Lemma 5. let $\overline{\mathcal{G}^k} := \mathbb{E}\left[\mathcal{G}^k\right]$, we have

$$\begin{aligned}&\mathbb{E}\left[\left\|\mathcal{G}^k - \overline{\mathcal{G}^k}\right\|^2\right]\\ &= \mathbb{E}\left[\left\|\frac{1}{r}\sum_{i\in\Omega_k} g(\boldsymbol{\theta}_i^k) + \mathbf{b}_i^k - \nabla f_i(\boldsymbol{\theta}_i^k)\right\|^2\right]\\ &\quad + \mathbb{E}\left[\left\|\frac{1}{r}\sum_{i\in\Omega_k}\nabla f_i(\boldsymbol{\theta}_i^k) - \frac{1}{n}\sum_{i=1}^n \nabla f_i(\boldsymbol{\theta}_i^k)\right\|^2\right]\\ &\leq \sum_{\substack{\Omega\in[n],\\|\Omega|=r}} P_r(\Omega_k = \Omega)\frac{1}{r}\sum_{i\in\Omega_k}\mathbb{E}\left[\left\|g(\boldsymbol{\theta}_i^k) + \mathbf{b}_i^k - \nabla f_i(\boldsymbol{\theta}_i^k)\right\|^2\right]\\ &\quad + 2\sum_{\substack{\Omega\in[n],\\|\Omega|=r}} P_r(\Omega_k = \Omega)\left(\frac{1}{r} - \frac{1}{n}\right)^2 r\sum_{i\in\Omega_k}\left\|\nabla f_i(\boldsymbol{\theta}_i^k)\right\|^2\\ &\quad + 2\sum_{\substack{\Omega\in[n],\\|\Omega|=r}} P_r(\Omega_k = \Omega)\frac{1}{n^2}(n-r)\sum_{i\notin\Omega_k}\left\|\nabla f_i(\boldsymbol{\theta}_i^k)\right\|^2\\ &\leq \frac{1}{n}\sum_{i=1}^n \mathbb{E}\left[\left\|g(\boldsymbol{\theta}_i^k) - \nabla f_i(\boldsymbol{\theta}_i^k)\right\|^2\right] + \frac{1}{n}\sum_{i=1}^n \mathbb{E}\left[\left\|\mathbf{b}_i^k\right\|^2\right]\\ &\quad + \frac{2(n-r)}{n^2}\frac{1}{\binom{n}{r}}\left(\binom{n}{r} - \binom{n-1}{r-1}\right)\sum_{i=1}^n \left\|\nabla f_i(\boldsymbol{\theta}_i^k)\right\|^2\\ &\quad + \frac{2(n-r)^2}{rn^2}\frac{1}{\binom{n}{r}}\binom{n-1}{r-1}\sum_{i=1}^n \left\|\nabla f_i(\boldsymbol{\theta}_i^k)\right\|^2\\ &\leq d\sigma^2 + \beta - \frac{\beta}{m} + \frac{4(n-r)^2}{n^3}\sum_{i=1}^n \left\|\nabla f_i(\boldsymbol{\theta}_i^k)\right\|^2.\end{aligned}$$

■

Lemma 7. *Assume $k = t\tau + s$, the expected network error at iteration k is bounded as follows:*

$$\mathbb{E}\left[\frac{1}{r}\sum_{i\in\Omega_k}\left\|\hat{\boldsymbol{\theta}}^k - \boldsymbol{\theta}_i^k\right\|^2\right] \leq 2s\eta^2(d\sigma^2 + \beta - \frac{\beta}{m})$$

$$+ 4s\eta^2 \frac{3n^2 + 2(n-r)^2}{n^3} \sum_{h=0}^{s-1}\sum_{i=1}^{n} \left\|\nabla f_i(\boldsymbol{\theta}_i^{t\tau+h})\right\|^2 . \qquad (19)$$

Proof: Since $k = t\tau + s$ and all clients in Ω_k start from the same model received from the server $\boldsymbol{\theta}^{t\tau}$ to update, i.e., $\hat{\boldsymbol{\theta}}^{t\tau} = \boldsymbol{\theta}_i^{t\tau} = \boldsymbol{\theta}^{t\tau}, \forall i \in \Omega_k$. For client $i \in \Omega_k$, we have

$$\boldsymbol{\theta}_i^k = \boldsymbol{\theta}_i^{t\tau} - \eta \sum_{h=0}^{s} g(\boldsymbol{\theta}_i^{t\tau+h}) + \mathbf{b}_i^{t\tau+h}. \qquad (20)$$

Given that $\hat{\boldsymbol{\theta}}^k = 1/r \sum_{i\in\Omega_k} \boldsymbol{\theta}_i^k$, one yields $\forall j \in \Omega_k$,

$$\begin{aligned}
\left\|\hat{\boldsymbol{\theta}}^k - \boldsymbol{\theta}_j^k\right\|^2 &\leq 2\eta^2 \left\|\frac{1}{r}\sum_{i\in\Omega_k}\sum_{h=0}^{s-1} g(\boldsymbol{\theta}_i^{t\tau+h}) + \mathbf{b}_i^{t\tau+h}\right\|^2 \\
&\quad + 2\eta^2 \left\|\sum_{h=0}^{s-1} g(\boldsymbol{\theta}_j^{t\tau+h}) + \mathbf{b}_j^{t\tau+h}\right\|^2 \\
&\leq 2s\eta^2 \sum_{h=0}^{s-1} \left\|\frac{1}{r}\sum_{i\in\Omega_k} g(\boldsymbol{\theta}_i^{t\tau+h}) + \mathbf{b}_i^{t\tau+h}\right\|^2 \\
&\quad + 2s\eta^2 \sum_{h=0}^{s-1} \left\|g(\boldsymbol{\theta}_j^{t\tau+h}) + \mathbf{b}_j^{t\tau+h}\right\|^2
\end{aligned}$$

where we use the inequality $\|\sum_{i=1}^n \mathbf{a}_i\|^2 \leq n \sum_{i=1}^n \|\mathbf{a}_i\|^2$. By Lemma 6 and the fact that $\mathbb{E}[(\mathbf{X} - \mathbb{E}[\mathbf{X}])^2] = \mathbb{E}[\mathbf{X}^2] - \mathbb{E}[\mathbf{X}]^2$, we have that

$$\begin{aligned}
&\mathbb{E}\left[\left\|\frac{1}{r}\sum_{i\in\Omega_k} g(\boldsymbol{\theta}_i^{t\tau+h}) + \mathbf{b}_i^{t\tau+h}\right\|^2\right] \\
&\leq d\sigma^2 + \beta - \frac{\beta}{m} + \frac{4(n-r)^2 + n^2}{n^3} \sum_{i=1}^{n} \left\|\nabla f_i(\boldsymbol{\theta}_i^{t\tau+h})\right\|^2 ,
\end{aligned}$$

which shows that the upper bound of $\|\hat{\boldsymbol{\theta}}^k - \boldsymbol{\theta}_j^k\|^2$ is not related to the index of client j. Thus, the expected network error at iteration k is

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{r}\sum_{j\in\Omega_k} \left\|\hat{\boldsymbol{\theta}}^k - \boldsymbol{\theta}_j^k\right\|^2\right] &\leq 4s\eta^2 \Big(sd\sigma^2 + s\beta - \frac{s\beta}{m} \\
&\quad + \frac{2(n-r)^2 + 2(n-1)^2 + n^2}{n^3} \sum_{h=0}^{s-1}\sum_{i=1}^{n} \left\|\nabla f_i(\boldsymbol{\theta}_i^{t\tau+h})\right\|^2\Big),
\end{aligned}$$

and Lemma 7 is finally obtained by relaxing the constant of the second term. ■

A. Convex Setting

This subsection describes the convergence rate of CPFed for smooth and strongly convex loss functions. The strong convexity is defined as follows:

Assumption 4. *The loss function l is λ-strongly convex if for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ we have $\|\nabla l(\mathbf{x}) - \nabla l(\mathbf{y})\| \geq \lambda\|\mathbf{x} - \mathbf{y}\|$ for some $\lambda > 0$.*

This assumption implies that the local loss functions $f_i, \forall i \in [n]$ and the global loss function f are also λ-strongly convex.

Convergence Criteria. In the convergence rate analysis of CPFed for convex loss functions, we use the expected optimality gap as the convergence criteria, i.e., after K iterations the algorithm achieves an expected γ-suboptimal solution if

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=0}^{K-1} f(\boldsymbol{\theta}^k) - f^*\right] \leq \gamma, \qquad (21)$$

where γ is arbitrarily small and f^* is the objective value at optimal solution $\boldsymbol{\theta}^*$. Specifically, we have the following main convergence results:

Theorem 2 (Convergence of CPFed for Convex Losses)**.** *For the CPFed algorithm, suppose the total number of iterations $K = T\tau$ where T is the number of communication round and τ is the round length. Under Assumptions 1-4, if the learning rate satisfies $5\eta L + 20\tau^2\eta^2 L^2 \leq 1$, and all clients are initialized at the same point $\boldsymbol{\theta}^0 \in \mathbb{R}^d$. Then after K iterations, the expected optimality gap is bounded as follows*

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=0}^{K-1} f(\boldsymbol{\theta}^k) - f^*\right] \leq \frac{(1-\eta\lambda)}{K\eta\lambda}\big(f(\boldsymbol{\theta}^0) - f^*\big) + H(\tau, \sigma^2), \qquad (22)$$

where $H(\tau, \sigma^2) := \eta L/2\lambda(4\eta L(\tau-1)(2\tau-1)+1)(d\sigma^2 + \beta - \beta/m)$. Here, ξ^2 is the variance bound of mini-batch stochastic gradients, σ^2 is the variance of Gaussian noise, L is the Lipschitz constant of the gradient, λ is the constant of strongly convexity and m is the size of local datasets.

Proof: According to Assumption 1, the global loss function f is L-smooth. Let $\mathcal{G}^k := \frac{1}{r}\sum_{r\in\Omega_k}\big(g(\boldsymbol{\theta}_i^k) + \mathbf{b}_i^k\big)$, we have the expectation of the objective gap between two iterations, i.e.,

$$\begin{aligned}
&\mathbb{E}\left[f(\hat{\boldsymbol{\theta}}^{k+1}) - f(\hat{\boldsymbol{\theta}}^k)\right] \\
&\leq \frac{\eta^2 L}{2}\mathbb{E}\left[\left\|\mathcal{G}^k\right\|^2\right] \\
&\quad - \eta\mathbb{E}\left[\frac{1}{r}\sum_{i\in\Omega_k}\langle\nabla f(\hat{\boldsymbol{\theta}}^k), \mathbb{E}\left[g(\boldsymbol{\theta}_i^k) + \mathbf{b}_i^k\right]\rangle\right] \\
&= \frac{\eta^2 L}{2}\mathbb{E}\left[\left\|\mathcal{G}^k\right\|^2\right] - \eta\mathbb{E}\left[\frac{1}{r}\sum_{i\in\Omega_k}\langle\nabla f(\hat{\boldsymbol{\theta}}^k), \nabla f_i(\boldsymbol{\theta}_i^k)\rangle\right] \\
&\leq \frac{\eta^2 L}{2}\mathbb{E}\left[\left\|\mathcal{G}^k\right\|^2\right] - \frac{\eta}{2}\|\nabla f(\hat{\boldsymbol{\theta}}^k)\|^2 - \frac{\eta}{2n}\sum_{i=1}^{n}\|\nabla f_i(\boldsymbol{\theta}_i^k)\|^2 \\
&\quad + \frac{\eta}{2}\mathbb{E}\left[\frac{1}{r}\sum_{i\in\Omega_k}\left\|\nabla f(\hat{\boldsymbol{\theta}}^k) - \nabla f_i(\boldsymbol{\theta}_i^k)\right\|^2\right]
\end{aligned}$$

where we use the inequality $-2\langle\mathbf{a}, \mathbf{b}\rangle = \|\mathbf{a} - \mathbf{b}\|^2 - \|\mathbf{a}\|^2 - \|\mathbf{b}\|^2$ for any two vectors $\mathbf{a}, \mathbf{b}$. According to Assumption 2 and 4, the expected objective gap between two iterations can be written as

$$\begin{aligned}
\mathbb{E}\left[f(\hat{\boldsymbol{\theta}}^{k+1})\right] &\leq \eta\lambda f^* + (1-\eta\lambda)f(\hat{\boldsymbol{\theta}}^k) + \frac{\eta^2 L}{2}\mathbb{E}\left[\left\|\mathcal{G}^k\right\|^2\right] \\
&\quad + \frac{\eta L^2}{2}\mathbb{E}\left[\frac{1}{r}\sum_{i\in\Omega_k}\left\|\hat{\boldsymbol{\theta}}^k - \boldsymbol{\theta}_i^k\right\|^2\right] - \frac{\eta}{2n}\sum_{i=1}^{n}\|\nabla f_i(\boldsymbol{\theta}_i^k)\|^2,
\end{aligned} \qquad (23)$$

where $\|\hat{\boldsymbol{\theta}}^k - \boldsymbol{\theta}_i^k\|^2$ represents the network error at iteration k and $\|\mathcal{G}^k\|^2$ represents the square norm of perturbed stochastic gradients at iteration k. By Lemma 6 and Lemma 7, we have that the last three terms of (23) is bounded by C_k, i.e.,

$$\begin{aligned} C_k \ge\ & \frac{4\eta^2 L(n-r)^2 + (\eta^2 L - \eta)n^2}{2n^3}\sum_{i=1}^{n}\left\|\nabla f_i(\boldsymbol{\theta}_i^k)\right\|^2 \\ & + 2s\eta^3 L^2 \frac{3n^2 + 2(n-r)^2}{n^3}\sum_{h=0}^{s-1}\sum_{i=1}^{n}\left\|\nabla f_i(\boldsymbol{\theta}_i^{t\tau+h})\right\|^2 \\ & + \frac{\eta^2 L}{2}(4s^2\eta L + 1)(d\sigma^2 + \beta - \frac{\beta}{m}). \end{aligned} \tag{24}$$

Then, taking the total expectation and averaging over K iterations based on (23), one can obtain

$$\begin{aligned} \mathbb{E}\left[\frac{1}{K}\sum_{k=0}^{K-1} f(\hat{\boldsymbol{\theta}}^{k+1}) - f^*\right] \le\ & \frac{1-\eta\lambda}{K\eta\lambda}\left(f(\boldsymbol{\theta}^0) - f^*\right) \\ & + \frac{1}{K\eta\lambda}\sum_{k=0}^{K-1} C_k. \end{aligned} \tag{25}$$

Next, our goal is to find the upper bound of $\frac{1}{K}\sum_{k=0}^{K-1} C_k$. Based on (24), we have

$$\begin{aligned} \frac{1}{K}\sum_{k=0}^{K-1} C_k \le\ & \frac{4\eta^2 L(n-r)^2 + (\eta^2 L - \eta)n^2}{2Kn^3}\sum_{k=0}^{K-1}\sum_{i=1}^{n}\left\|\nabla f_i(\boldsymbol{\theta}_i^k)\right\|^2 \\ & + 2\tau^2\eta^3 L^2 \frac{3n^2 + 2(n-r)^2}{Kn^3}\sum_{k=0}^{K-1}\sum_{i=1}^{n}\left\|\nabla f_i(\boldsymbol{\theta}_i^k)\right\|^2 \\ & + \frac{\eta^2 L}{2}\left(4\eta L(\tau-1)(2\tau-1)+1\right)(d\sigma^2 + \beta - \frac{\beta}{m}). \end{aligned}$$

based on the fact that $1^2 + \cdots + n^2 = n(n+1)(2n+1)/6$. Since we have

$$\begin{aligned} & 2\tau^2\eta^3 L^2 \frac{3n^2 + 2(n-r)^2}{Kn^3} + \frac{4\eta^2 L(n-r)^2 + (\eta^2 L - \eta)n^2}{2Kn^3} \\ & \le \frac{(20\tau^2\eta^3 L^2 + 5\eta^2 L - \eta)n^2}{2Kn^3}, \end{aligned}$$

then if the learning rate η satisfies that $5\eta L + 20\tau^2\eta^2 L^2 \le 1$, we can finally obtain a constant bound for $\frac{1}{K}\sum_{k=1}^{K}\frac{C_k}{\eta\lambda}$, i.e.,

$$\frac{1}{K}\sum_{k=0}^{K-1} C_k \le \frac{\eta L}{2\lambda}(4\eta L(\tau-1)(2\tau-1)+1)(d\sigma^2 + \beta - \frac{\beta}{m}).$$

Theorem 2 follows by substituting the expression of $\frac{1}{K}\sum_{k=0}^{K-1} C_k$ back to (25). ■

B. Non-convex Setting

This subsection describes the convergence rate of CPFed for smooth and non-convex loss functions.

Convergence Criteria. In the error-convergence analysis, since the objective function is non-convex, we use the expected gradient norm as an indicator of convergence, i.e., after K iterations the algorithm achieves an expected κ-suboptimal solution if:

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=0}^{K-1}\|\nabla f(\hat{\boldsymbol{\theta}}^k)\|^2\right] \le \kappa, \tag{26}$$

where κ is arbitrarily small. This condition guarantees the convergences of the algorithm to a stationary point.

Theorem 3 (Convergence of CPFed for Non-convex Losses)**.** *For the CPFed algorithm, suppose the total number of iterations $K = T\tau$ where T is the number of communication rounds and τ is the round length. Under Assumptions 1-3, if the learning rate satisfies $5\eta L + 20\tau^2\eta^2 L^2 \le 1$, and all clients are initialized at the same point $\boldsymbol{\theta}^0 \in \mathbb{R}^d$, then after K iterations the expected gradient norm is bounded as follows*

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=0}^{K-1}\|\nabla f(\hat{\boldsymbol{\theta}}^k)\|^2\right] \le \frac{2(f(\boldsymbol{\theta}^0) - f^*)}{\eta K} + P(\tau, \sigma^2), \tag{27}$$

where $P(\tau, \sigma^2) := \eta L(4\eta L(\tau-1)(2\tau-1)+1)(d\sigma^2 + \beta - \frac{\beta}{m})$. Here, σ^2 is the variance of Gaussian noise, L is the Lipschitz constant of the gradient, and m is the size of local datasets.

Proof: According to L-smoothness of the objective function f, we have

$$\begin{aligned} & \mathbb{E}\left[f(\hat{\boldsymbol{\theta}}^{k+1}) - f(\hat{\boldsymbol{\theta}}^k)\right] \\ & \le \frac{\eta^2 L}{2}\mathbb{E}\left[\left\|\mathcal{G}^k\right\|^2\right] - \eta\mathbb{E}\left[\frac{1}{r}\sum_{i\in\Omega_k}\langle\nabla f(\hat{\boldsymbol{\theta}}^k), \mathbb{E}\left[g(\boldsymbol{\theta}_i^k) + \mathbf{b}_i^k\right]\rangle\right] \\ & = \frac{\eta^2 L}{2}\mathbb{E}\left[\left\|\mathcal{G}^k\right\|^2\right] - \eta\mathbb{E}\left[\frac{1}{r}\sum_{i\in\Omega_k}\langle\nabla f(\hat{\boldsymbol{\theta}}^k), \nabla f_i(\boldsymbol{\theta}_i^k)\rangle\right] \\ & \le \frac{\eta^2 L}{2}\mathbb{E}\left[\left\|\mathcal{G}^k\right\|^2\right] - \frac{\eta}{2}\|\nabla f(\hat{\boldsymbol{\theta}}^k)\|^2 - \frac{\eta}{2n}\sum_{i=1}^{n}\|\nabla f_i(\boldsymbol{\theta}_i^k)\|^2 \\ & + \frac{\eta}{2r}\mathbb{E}\left[\sum_{i\in\Omega_k}\left\|\nabla f(\hat{\boldsymbol{\theta}}^k) - \nabla f_i(\boldsymbol{\theta}_i^k)\right\|^2\right]. \end{aligned}$$

After minor rearranging, it is easy to show

$$\begin{aligned} \mathbb{E}\left[\|\nabla f(\hat{\boldsymbol{\theta}}^k)\|^2\right] \le\ & \frac{2}{\eta}\mathbb{E}\left[f(\hat{\boldsymbol{\theta}}^k) - f(\hat{\boldsymbol{\theta}}^{k+1})\right] + L\eta\mathbb{E}\left[\left\|\mathcal{G}^k\right\|^2\right] \\ & - \frac{1}{n}\sum_{i=1}^{n}\|\nabla f_i(\boldsymbol{\theta}_i^k)\|^2 + L^2\mathbb{E}\left[\frac{1}{r}\sum_{i\in\Omega_k}\left\|\hat{\boldsymbol{\theta}}^k - \boldsymbol{\theta}_i^k\right\|^2\right]. \end{aligned} \tag{28}$$

By Lemma 6 and Lemma 7, we have that the last three terms of (28) is bounded by B_k, which is equivalent to $\frac{2}{\eta}C_k$. Then, taking the total expectation and averaging over all iterations, we have

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=0}^{K-1}\|\nabla f(\hat{\boldsymbol{\theta}}^k)\|^2\right] \le \frac{2(f(\boldsymbol{\theta}^0) - f^*)}{\eta K} + \frac{1}{K}\sum_{k=0}^{K-1} B_k, \tag{29}$$

where we use the fact that $f(\boldsymbol{\theta}^K) \ge f^*$. If the learning rate η and the number of selected clients r satisfy that

$$5\eta L + 20\tau^2\eta^2 L^2 \le 1,$$

we have

$$\frac{1}{K}\sum_{k=0}^{K-1} B_k \le \eta L(4\eta L(\tau-1)(2\tau-1)+1)(d\sigma^2 + \beta - \frac{\beta}{m}).$$

Substituting the expression of $\frac{1}{K}\sum_{k=0}^{K-1} B_k$ back to (29), we finally obtain Theorem 3. ■

VII. Experiments

In this section, we evaluate the performance of our proposed scheme CPFed. We first describe our experimental setup and then show the convergence properties of CPFed. Next, we demonstrate the communication efficiency of CPFed by comparing it with a baseline approach. Finally, we show the trade-off between privacy and model accuracy in CPFed and how our secure aggregation protocol improves the accuracy of the learned model.

A. Experimental Setup

Datasets and Learning Tasks. We explore the benchmark dataset *Adult* [16] using both logistic regression and neural network models in our experiments. The Adult dataset contains 48,842 samples with 14 numerical and categorical features, with each sample corresponding to a person. The task is to predict if the person's income exceeds $\$50,000$ based on the 14 attributes, namely, *age, workclass, fnlwgt, education, education-num, marital-status, occupation, relationship, race, sex, capital-gain, capital-loss, hours-per-week, and native-country*. To simulate a distributed setting based on the Adult dataset, we evenly assign the original Adult data to 16 devices such that each device contains 3,052 data samples. We train a logistic regression classifier and a 3-layer neural network classifier (using rectified linear activation function) on the 16 devices with just the categorical features and use the softmax cross-entropy as the loss function.

Baseline. We select the state-of-the-art differentially private learning scheme named DP-SGD [17] as a strong baseline to evaluate the efficiency of our proposed scheme. In DP-SGD, only one step of stochastic gradient descent is performed to update the local model on each device during each aggregation period, and Gaussian noise is added to each model update before sending it out.

Hyperparameters. We take 80% of the data on each device for training, 10% for testing and 10% for validation. We tune the hyperparameters on the validation set and report the average accuracy on the testing sets of all devices. For all experiments, we set the Lipschitz constant of loss function $L = 1$, privacy failure probability $\delta = 10^{-4}$ and the number of selected clients $r = 10$.

B. Convergence Property of CPFed

In this subsection, we show the algorithmic convergence properties of CPFed with respect to the number of communication rounds T under several settings of noise magnitude σ and local iteration number τ. Specifically, for the logistic regression, we show the testing accuracy and the expected training loss with respect to the number of communication rounds T when $\sigma = \{10^{-5}, 10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$ respectively. For each case, we consider 4 different values of the local iteration number, i.e., $\tau = \{1, 5, 10, 40\}$. The results for the logistic regression are finally shown in Fig. 4. For the neural network, we show the testing accuracy and expected gradient norm with respect to the number of communication rounds T, when $\sigma = \{10^{-4}, 10^{-3}, 5 \times 10^{-3}, 10^{-2}\}$ respectively. For each case, we consider 4 different values of the local iteration number, i.e., $\tau = \{1, 5, 10, 20\}$. The results for the neural network are finally shown in Fig. 5.

For the logistic regression classifier, the testing accuracy and expected loss will generally decrease sharply and then slowly afterwards. As the noise magnitude σ increases, the expected training loss of the logistic regression converges to a higher bound and the testing accuracy decreases, which is consistent with the convergence properties of CPFed where larger σ implies larger convergence error. For all settings of noise, with larger local iteration number, the expected loss drops more sharply at the beginning and arrives at a higher stationary point, which is consistent with CPFed's convergence properties where larger τ implies larger convergence error. When $\sigma = 10^{-3}$ and $\tau = 40$, we can see that after the expected loss decreased to 7 using about 40 rounds of communication, it increases as more computations and communications are involved. The reason is that after the loss arrived at a stationary point, keeping training brings additional noise into the well-trained model and hence the model performance drops. Similar trends have been observed for the neural network classifier. When $\sigma = 10^{-2}$, the testing accuracy drops from the initialized value quickly as the noise is added into the system, and then it increases as more computations and communications are involved.

C. Communication Efficiency of CPFed

In this subsection, we show the communication efficiency of CPFed comparing with baseline approach DP-SGD. Specifically, for the logistic regression, we set the number of communication rounds $T = 20$ for both approaches and $\tau = 10$ for CPFed. We have both approaches preserve $(10, 10^{-4})$-DP after 20 rounds of communication. For CPFed, we compute the noise magnitude σ by Theorem 1. For DP-SGD, we first convert the $(10, 10^{-4})$-DP guarantee to a ρ-zCDP guarantee by Lemma 3, and then evenly assign the zCDP budget ρ to 20 communication rounds and compute the noise magnitude σ by Lemma 1. Finally, we have CPFed and DP-SGD preserve the same level of privacy at each communication round. The testing accuracy and expected loss with respect to the number of communicate rounds are shown in Fig. 7. For the neural network, we set the number of communication rounds $T = 50$, the overall privacy budget $\epsilon = 10$ for both approaches, and $\tau = 5$ for CPFed. The testing accuracy and expected gradient norm with respect to the number of communication rounds are shown in Fig. 6. Note that due to the randomized nature of the differentially private mechanism, we repeat all the experiments for 5 times and report the average results.

For the logistic regression, we observe that CPFed exhibits faster convergence than DP-SGD at the beginning, and finally achieves higher accuracy and lower expected loss than DP-SGD within 20 rounds of communication. For the neural network, we observe the similar trend. CPFed converges faster than DP-SGD and achieves higher accuracy and lower expected gradient norm than DP-SGD. Accordingly, CPFed achieves higher communication efficiency than DP-SGD in both convex and non-convex cases.

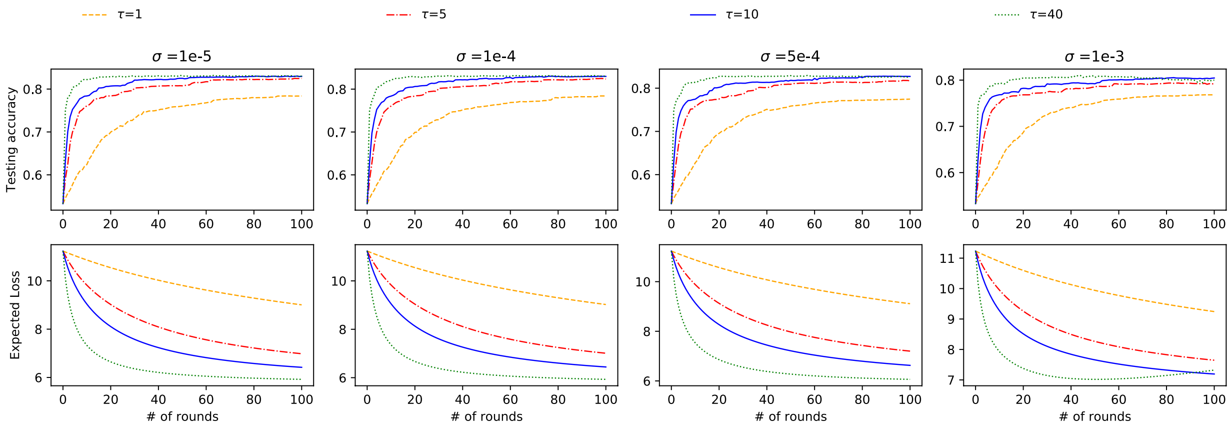

Fig. 4. Convergence of the expected loss (logistic regression). Here, we show the convergence of the first 100 communication rounds.

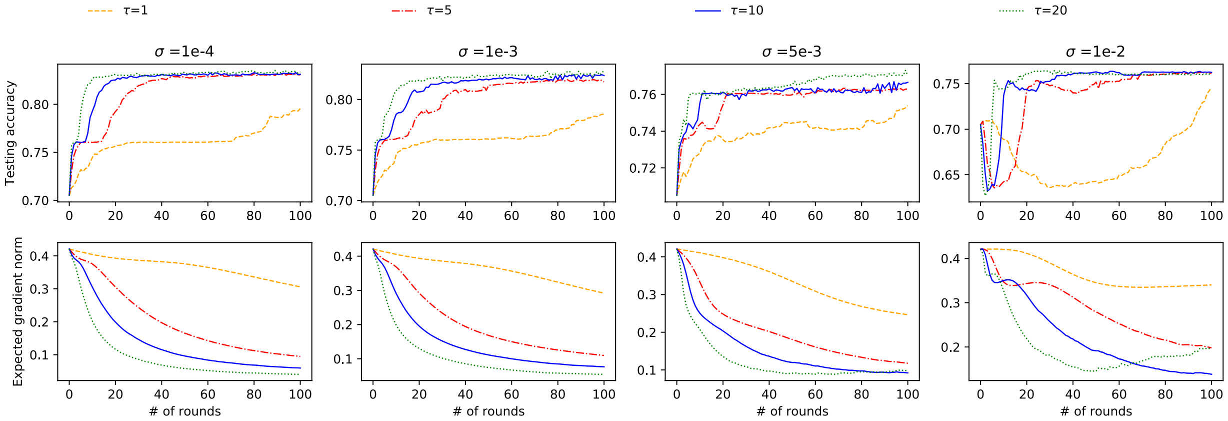

Fig. 5. Convergence of the expected gradient norm (neural network). Here, we show the convergence of the first 100 communication rounds.

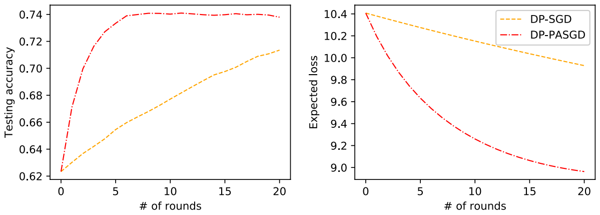

Fig. 6. Communication efficiency of CPFed (logistic regression). Here, we set $T = 20$ and $\epsilon = 10$ for both approaches, and set $\tau = 10$ for CPFed.

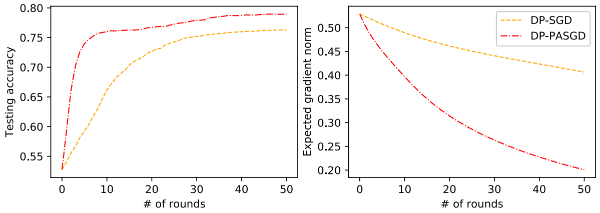

Fig. 7. Communication efficiency of CPFed (neural network). Here, we set $T = 50$ and $\epsilon = 10$ for both approaches, and set $\tau = 5$ for CPFed.

D. *Trade-off between Privacy and Accuracy*

In this subsection, we evaluate the effects of different values of privacy budgets ϵ on the accuracy of trained classifiers. In addition, we compare our approach with the algorithm without secure aggregation (i.e., same as CPFed but without secure aggregation), to show how secure aggregation improves the accuracy. For the logistic regression, we set the local iteration number $\tau = 2$ and the number of communication rounds $T = 20$. For the neural network, we set the local iteration number $\tau = 5$ and the number of communication rounds $T = 50$. We show the testing accuracy with respect to different values of privacy budget ϵ of logistic regression and neural network in Fig. 8 and Fig. 9, respectively. Note that due to the randomized nature of differentially private mechanisms, we repeat all the experiments for 5 times and report the average results. As expected, a larger ϵ value results in higher accuracy while providing lower differential privacy guarantee. However, our approach with secure aggregation always outperforms the approach without secure aggregation because less noise is added at each iteration of our approach.

VIII. Related Work

Distributed machine learning based on (stochastic) gradient descent has been well studied in the literature with both theoretical convergence analysis [18], [19], [20] and real-world experiments [21]. However, traditional distributed learning algorithms do not fit into federated learning wherein the communication cost is usually high. Recent studies have started to reduce the communication cost in distributed learning [8], [9], [10], [11], [22], [23], [24], which could be divided into two categories. The first category is to reduce the size of messages

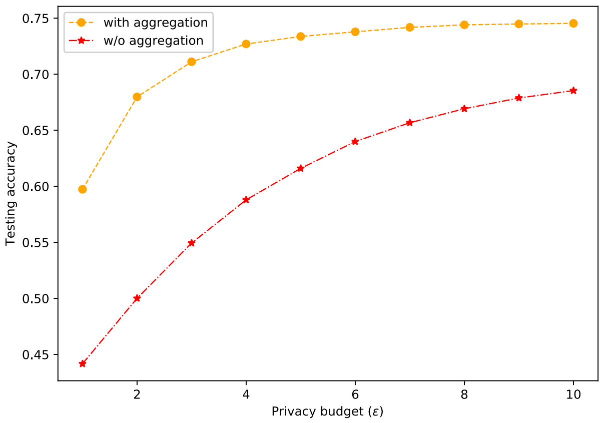

Fig. 8. Trade-off between privacy and accuracy (logistic regression). Here, we set $\tau = 2$ and $T = 20$.

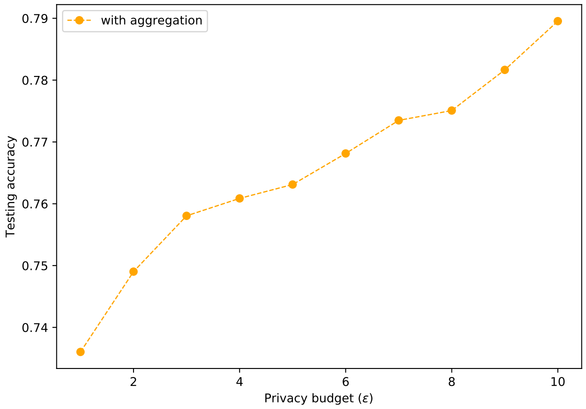

Fig. 9. Trade-off between privacy and accuracy (neural network). Here, we set $T = 50$ and $\tau = 5$.

transmitted between the device and server per communication round by compression, e.g., quantizing and/or sparsifying gradients computed by each device before aggregation [8], [9], [22]. The second category is to reduce the number of communication rounds by techniques such as periodic averaging that pay more local computation for less communication [10], [11], [23], [24] and adaptive aggregation where devices selectively upload their messages [25], [26]. However, most of the above communication-efficient schemes ignore the privacy aspect.

Besides, privacy issue has received significant attention recently in distributed learning scenarios handling user-generated data. Among distributed learning schemes that preserve privacy, many of them rely on secure multi-party computation or homomorphic encryption, which involve both high computation and communication overhead and are only applicable to simple learning tasks such as linear regression [27] and logistic regression [28]. Furthermore, these privacy-preserving solutions could not prevent the information leakage from the final learned model. Differential privacy has become the de-facto standard for privacy notion and is being increasingly adopted in private data analysis [14]. A wide range of differentially private distributed learning algorithms (see [17], [29], [30], [31] and references therein) have been proposed based on different optimization methods (e.g., alternating direction method of multipliers, gradient descent, and distributed consensus) and noise addition mechanisms (e.g., output perturbation, objective perturbation, and gradient perturbation). However, most of them do not consider the communication efficiency aspect and therefore are not suitable for federated learning. Moreover, few of them could provide a rigorous performance guarantee without much accuracy degradation of the final learned model.

A few very recent works [32], [33], [34] have started to consider both the communication and privacy aspects in federated learning. Specifically, Agarwal et al. [32] proposed a modified distributed SGD scheme based on gradient quantization and binomial mechanism to make the scheme both private and communication-efficient. Li et al. [33] developed a method that compresses the transmitted messages via sketches to simultaneously achieve communication efficiency and differential privacy in distributed learning. Our work is orthogonal to theirs by focusing on reducing the number of communication rounds instead of the size of messages transmitted per round. McMahan et al. [34] also designed an approach to reduce the number of communication rounds while preserving differential privacy in federated learning. However, their approach assumes a fully trusted server and does not provide any rigorous performance guarantee. In comparison, our proposed CPFed scheme achieves high communication efficiency and model accuracy while preserving differential privacy without assuming a fully trusted server. Furthermore, the scheme has a rigorous performance guarantee.

IX. Conclusions

This paper focuses on privacy-preserving and communication-efficient federated learning. We have proposed a new distributed learning scheme based on distributed SGD with rigorous privacy guarantee. Our methodology saves the number of communications and preserves the differential privacy of participants, while achieving high accuracy of the resulting model. We provided rigorous convergence analysis of our proposed approach and extensive experiments based on the real-world dataset have verified the effectiveness of the proposed scheme and shown the trade-off between model accuracy and privacy. In future work, we plan to study the performance of CPFed in other learning settings such as multi-task learning and privacy considerations such as personalized differential privacy.

References

[1] A. Pothitos, "IoT and wearables: Fitness tracking," 2017, http://www.mobileindustryreview.com/2017/03/iot-wearables-fitness-tracking.html.

[2] P. Goldstein, "Smart cities gain efficiencies from IoT traffic sensors and data," 2018, https://statetechmagazine.com/article/2018/12/smart-cities-gain-efficiencies-iot-traffic-sensors-and-data-perfcon.

[3] A. Weinreic, "The future of the smart home: Smart homes and IoT: A century in the making," 2018, https://statetechmagazine.com/article/2018/12/smart-cities-gain-efficiencies-iot-traffic-sensors-and-data-perfcon.

[4] E. Folk, "How IoT is transforming the energy industry," 2019, https://www.renewableenergymagazine.com/emily-folk/how-iot-is-transforming-the-energy-industry-20190418.

[5] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[6] M. Al-Rubaie and J. M. Chang, "Reconstruction attacks against mobile-based continuous authentication systems in the cloud," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2648–2663, 2016.

[7] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.

[8] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "Atomo: Communication-efficient learning via atomic sparsification," in *Advances in Neural Information Processing Systems*, 2018, pp. 9850–9861.

[9] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signSGD: Compressed optimisation for non-convex problems," *arXiv preprint arXiv:1802.04434*, 2018.

[10] J. Wang and G. Joshi, "Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms," *arXiv preprint arXiv:1808.07576*, 2018.

[11] F. Haddadpour, M. M. Kamani, M. Mahdavi, and V. Cadambe, "Local SGD with periodic averaging: Tighter analysis and adaptive synchronization," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 080–11 092.

[12] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1310–1321.

[13] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for federated learning on user-held data," in *Proc. NIPS Workshop Private Multi-Party Mach. Learn.*, 2016.

[14] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[15] M. Bun and T. Steinke, "Concentrated differential privacy: Simplifications, extensions, and lower bounds," in *Theory of Cryptography Conference*. Springer, 2016, pp. 635–658.

[16] C. L. Blake, "UCI repository of machine learning databases, Irvine, University of California," 1998, http://www.ics.uci.edu/~mlearn/MLRepository.

[17] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 308–318.

[18] H. Kushner and G. G. Yin, *Stochastic approximation and recursive algorithms and applications*. Springer Science & Business Media, 2003, vol. 35.

[19] O. Shamir and T. Zhang, "Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes," in *International Conference on Machine Learning*, 2013, pp. 71–79.

[20] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.

[21] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *Advances in neural information processing systems*, 2012, pp. 1223–1231.

[22] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Advances in Neural Information Processing Systems*, 2017, pp. 1709–1720.

[23] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 63–71.

[24] N. H. Tran, W. Bao, A. Zomaya, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1387–1395.

[25] T. Chen, G. Giannakis, T. Sun, and W. Yin, "LAG: Lazily aggregated gradient for communication-efficient distributed learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 5050–5060.

[26] J. Sun, T. Chen, G. Giannakis, and Z. Yang, "Communication-efficient distributed learning via lazily aggregated quantized gradients," in *Advances in Neural Information Processing Systems*, 2019, pp. 3365–3375.

[27] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *IEEE Symposium on Security and Privacy*, 2013, pp. 334–348.

[28] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *IEEE Symposium on Security and Privacy*, 2017, pp. 19–38.

[29] Y. Guo and Y. Gong, "Practical collaborative learning for crowdsensing in the internet of things with differential privacy," in *2018 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2018, pp. 1–9.

[30] Z. Huang, R. Hu, Y. Guo, E. Chan-Tin, and Y. Gong, "DP-ADMM: ADMM-based distributed learning with differential privacy," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1002–1012, 2019.

[31] Z. Huang, S. Mitra, and G. Dullerud, "Differentially private iterative synchronous consensus," in *Proceedings of the 2012 ACM workshop on Privacy in the electronic society*. ACM, 2012, pp. 81–90.

[32] N. Agarwal, A. T. Suresh, F. X. X. Yu, S. Kumar, and B. McMahan, "cpSGD: Communication-efficient and differentially-private distributed SGD," in *Advances in Neural Information Processing Systems*, 2018, pp. 7564–7575.

[33] T. Li, Z. Liu, V. Sekar, and V. Smith, "Privacy for free: Communication-efficient learning with differential privacy using sketches," *arXiv preprint arXiv:1911.00972*, 2019.

[34] B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *International Conference on Learning Representations (ICLR)*, 2018. [Online]. Available: https://openreview.net/pdf?id=BJ0hF1Z0b