A Privacy-Preserving and Trustable Multi-agent Learning Framework

Anudit Nagar  
Bennett University  
anudit@bennett.edu.in  

Cuong Tran  
Syracuse University  
cutran@syr.edu  

Ferdinando Fioretto  
Syracuse University  
ffiorett@syr.edu  

Abstract

Distributed multi-agent learning enables agents to cooperatively train a model without requiring to share their datasets. While this setting ensures some level of privacy, it has been shown that, even when data is not directly shared, the training process is vulnerable to privacy attacks including data reconstruction and model inversion attacks. Additionally, malicious agents that train on inverted labels or random data, may arbitrarily weaken the accuracy of the global model. This paper addresses these challenges and presents Privacy-preserving and trustable Distributed Learning (PT-DL), a fully decentralized framework that relies on Differential Privacy to guarantee strong privacy protection of the agents data, and Ethereum smart contracts to ensure trustability. The paper shows that PT-DL is resilient up to a 50% collusion attack, with high probability, in a malicious trust model and the experimental evaluation illustrates the benefits of the proposed model as a privacy-preserving and trustable distributed multi-agent learning system on several classification tasks.

1 Introduction

Agencies are increasingly leveraging distributed data shared across organizations to augment their AI-powered services. Examples include transportation services sharing location-based data to improve their on-demand capabilities and hospitals sharing patient data to prevent epidemic outbreaks. The proliferation of these applications leads to a transition from proprietary data acquisition and processing to distributed data ecosystems, where agents learn and make decisions using data owned by different organizations.

While the cooperative use of rich dataset may enhance the accuracy and robustness of the individual agents’ models, many of these datasets contain sensitive information whose use is strictly regulated by local policies. For example, medical data is regulated by HIPAA regulations [21] in the US, and the E.U. General Data Protection Regulation (GDPR) requires the consumer data privacy to be protected [20].

Distributed multi-agent learning enables agents to cooperatively train a learning model without requiring them to share their dataset. Typical multi-agent learning frameworks, including federated learning [29], allow individual agents to train their local models on their own datasets and share model parameters with a centralized agent that aggregates the received parameters and sends them back to the agents. This simple, yet effective procedure, repeats for several iterations and allows the participating agents to learn a global model without accessing data of other agents.

However, standard multi-agent learning algorithms are susceptible to privacy and trustability issues. Indeed, using solely the model parameters, model inversion attacks can be performed to gain insight about the input data used during the training [19,13], and membership inference attacks have been shown successful to determine if an individual data was used or not during model training [34].

Work done while A. Nagar was interning at Syracuse University.
privacy-invasive attacks can be mitigated with the adoption of Differential Privacy, a formal privacy model that computes and bounds the privacy loss associated to the participation of an individual into a computation. However, differential privacy does not protect against malicious agents that may not comply with the protocol–e.g., by altering their data labels, performing what known as model inversion attacks \[18\], or training over random data. These behaviors can strongly penalize the accuracy of the resulting global model. Defending against such attacks requires some form of trustability and exclude the flagged contributions from the global model updates. The decentralized setting of multi-agent learning renders this task particularly challenging.

To address these issues, this paper proposed the use a decentralized computational environment, such as blockchains, that enables differentially private multi-agent training, guaranteeing both privacy and trustability. The resulting framework, called trustable and Private Distributed Learning (PT-DL) relies on the Ethereum blockchain, that combines an immutable data storage with a Turing-complete computational environment \[36\] and guarantees the correctness of the programs executed over the blockchain.

The privacy requirement is enforced by using a clipping approach on the model parameters and the privacy analysis relies on composition methods \[12\] and the moment accountant for the Sampled Gaussian Mechanism \[1, 31\]. trustability is achieved directly through the use of the computation ran on the immutable blockchain combined with a decentralized verification procedure that validates the genuineness of the agent contributions to the model. The paper shows that the trustable scheme implemented is robust, with high probability, up to the case where half of the agents may collude.

In addition to these properties, PT-DL requires no central trusted server and allows agents to exploit several network communication topologies to reduce the negative impact of frequent model aggregations as well as lowering the communication costs. The empirical analysis illustrates the benefits of the proposed model on classification and medical diagnosis tasks, based on a COVID-19 dataset, in terms of accuracy, communication cost, and resiliency to biased data distributions and malicious updates. Additionally, it reports an excellent trade-off between accuracy and privacy and shows that PT-DL may represent a promising step towards a practical tool for privacy-preserving and trustable multi-agent learning.

This work is an extended version of \[32\].

2 Related Work

Multi-agent learning has been studied extensively, especially in the context of optimization \[2, 14, 33, 17\] and communication efficiency \[33, 41, 37, 40, 26, 16\], e.g., by compressing the information that is passed among the agents during training.

In the context of learning non-convex functions, as in deep learning tasks, from decentralized data, Federated learning \[25, 28\] is considered the de-facto standard framework. A cornerstone in federated learning algorithms is Federated Averaging (FedAvg), a variant of distributed stochastic gradient descent (SGD). At each iteration, a selected subset of FedAvg agents locally and independently take a gradient descent step on the current model parameters using their local data, and the server then computes a weighted average of the resulting parameters. There are several contributions that apply differential privacy in the context of multi-agent learning. Cheng et al. \[6, 7\] focus on a differentially private decentralized learning system, but their analysis is applied to strongly convex problems only. Bellet et al. \[4, 3\] proposes a differentially private asynchronous decentralized learning systems for convex problems and uses the Laplace mechanism over a block coordinate descent algorithm to achieve privacy. Fioretto and Van Hentenryck \[15\] develops a federated data sharing protocol which guarantees differential privacy.

A different line of work applies the Alternating Direction Method of Multipliers to solve a distributed optimization problem under privacy constraints \[5, 38, 39\]. These, however, are not generally adopted for training deep learning models in a distributed system of agents. An important contribution, by Geyer et al. \[20\], develops a differentially private federated learning system which builds on FedAvg. This paper uses it as a baseline.

Although these systems guarantee privacy, the reported experiments show that the induced accuracy is not always satisfactory and, in addition, consider privacy in isolation, ignoring trustability. Finally,
there have also been several proposals of using blockchain to support the training of deep neural networks [35] and federated learning [24] for incentive purpose.

In contrast to the work above, this paper develops a distributed multi-agent learning framework that ensures both privacy, through the use of differentially private model updates, and trustability, coordinating computations on a blockchain. Finally, it proposes a simple, yet effective, communication model that may reduce the number of communication rounds required during model training when compared to standard federated learning approaches.

3 Preliminaries

3.1 Problem Settings and Goals

The paper considers a collection of $K$ agents, each holding a dataset $D_a$ ($a \in [K]$) consisting of $n_a$ individual data points $(X_i, Y_i)$, with $i \in [n_a]$ drawn from an unknown distribution. Therein, $X_i \in \mathcal{X}$ is a feature vector and $Y_i \in \mathcal{Y}$ is a label. The paper assumes that an individual data is not repeated across datasets, and, thus, the agents’ datasets are disjoint. Finally, it denotes with $D = \cup_{a \in [K]} D_a$ as the union of the agents’ datasets and with $n = \sum_{a \in [K]} n_a$ the size of $D$. For example, consider the case of a classifier that needs to predict the presence of a particular disease. The training example features $X_i$ may describe the individual’s X-Ray image, gender, age, and demographics, and $Y_i$ represents whether the individual has the disease. Each agent may hold a small dataset with samples representative of a specific demographics and, thus, the learning task may benefit from using data samples of other agents. Therefore, the goal is to learn a global classifier $\mathcal{M}_\theta : \mathcal{X} \rightarrow \mathcal{Y}$, where $\theta$ is a real valued vector describing the model parameters. The model quality is measured in terms of a non-negative, and assumed differentiable, loss function $L : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_+$, and the problem is that of minimizing the empirical risk function:

$$\min_\theta J(\mathcal{M}_\theta, D) = \frac{1}{K} \sum_{a \in [K]} J_a(\mathcal{M}_\theta^a, D_a), \quad (1)$$

where $\mathcal{M}_\theta^a$ is a local classifier associated to agent $a \in [K]$ and $J_a$ is its empirical risk function, defined as

$$J_a(\mathcal{M}_\theta^a, D_a) = \frac{1}{n_a} \sum_{(X_i, Y_i) \in D_a} L(\mathcal{M}_{\theta}^a(X_i), Y_i). \quad (2)$$

The paper focuses on instances of the problem (1) where the functions $J_a$ are non-convex, as in deep learning tasks.

The setting does not assume that the data contributed by each of the $K$ agents are i.i.d. or balanced, that is, different agents may have data that is not representative of the whole population.

As discussed in the introduction, there are two practical aspects that make the development of systems that solve problem (1) challenging: privacy and trustability. The following sections review the main concepts adopted to address these challenges.

3.2 Blockchains

Blockchains are decentralized general transaction ledgers that allow participants to create unchangeable records, each time-stamped and linked to the previous one, making each operation verifiable and auditable. A protocol executed on a blockchain is referred to as smart contract and this paper adopts the Ethereum protocol, a Turing-complete computational environment [36]. The Ethereum protocol ensures that smart contracts are executed correctly, thus, agents can trust that any data sent to the blockchain will not be corrupted and that smart contracts logic will be executed as intended. With these guarantees, blockchains are appropriate to run distributed learning trustability procedures.

While this environment guarantees that the data stored on the blockchain is immutable, it does not guarantee data privacy.

3.3 Differential Privacy

Differential privacy (DP) [11] is a strong privacy notion used to quantify and bound the privacy loss of an individual participation to a computation. The privacy goal of this work is to guarantee that the
output of the learning model does not differ much when a single individual is added or removed to the dataset, limiting the amount of information that the model reveals about any individual.

The action of changing a single attribute from a dataset \( D \), resulting in a new dataset \( D' \), defines the notion of dataset adjacency. Two dataset \( D \) and \( D' \in \mathcal{X}^n \) are said adjacent, denoted \( D \sim D' \), if they differ in at most the addition or removal of a single entry (e.g., in one individual’s participation).

**Definition 1** (Differential Privacy). A randomized mechanism \( \mathcal{M} : \mathcal{X}^n \to \mathcal{R} \) with domain \( \mathcal{X}^n \) and range \( \mathcal{R} \) is \((\epsilon, \delta)\)-differentially private if, for any two adjacent datasets \( D, D' \in \mathcal{X}^n \), and any subset of output responses \( R \subseteq \mathcal{R} \):

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

When \( \delta = 0 \) the algorithm is said to satisfy pure differential privacy. Parameter \( \epsilon > 0 \) describes the privacy loss of the algorithm, with values close to 0 denoting strong privacy, while parameter \( \delta \in [0, 1] \) captures the probability of failure of the algorithm to satisfy \( \epsilon \)-differential privacy. The global sensitivity \( \Delta_f \) of a real-valued function \( f : \mathcal{X}^n \to \mathcal{R}^k \) is defined as the maximum amount by which \( f \) changes in two adjacent inputs \( D \) and \( D' \): \( \Delta_f = \max_{D \sim D'} \| f(D) - f(D') \|_2 \). In particular, the Gaussian mechanism, defined by

\[
\mathcal{M}(D) = f(D) + \mathcal{N}(0, \Delta_f^2 \sigma^2),
\]

where \( \mathcal{N}(0, \Delta_f^2 \sigma^2) \) is the Gaussian distribution with 0 mean and standard deviation \( \Delta_f \sigma^2 \), satisfies \((\epsilon, \delta)\)-differential privacy for \( \delta > \frac{\epsilon^2}{8} \exp(-\sigma^2/2) \) and \( \epsilon < 1 \) [12].

Differential privacy also satisfies several important properties. In particular, sequential composition theorems allow to reason about the privacy loss resulting by the repeated application of a function to a dataset [12] [22], parallel composition [12] ensures that the application of an \((\epsilon, \delta)\)-differentially private mechanism to disjoint dataset does not induce additional privacy loss, and post-processing immunity [12] ensures that differential privacy is preserved even if the output of a mechanism is post-processed throughout an arbitrarily data-independent transformation.

Although advanced composition theorems allow a tight privacy loss analysis, which scales sublinearly in the number of applications of the DP mechanisms, in iterative algorithms, like those used in this paper, this process quickly leads to large privacy losses. To address the shortcomings, this paper adopts the Rényi Differential Privacy (RDP) model by Mironov [30]. RDP shares many important properties with the standard definition of differential privacy, including parallel composition and post-processing immunity, while additionally allowing a tighter privacy loss analysis for iterative processes (see Section 6 for details).

### 4 The PT-DL Framework

Private and trustable Distributed Learning (PT-DL) is a fully distributed learning framework that ensures privacy and trustability while keeping the network bandwidth low. The framework is schematically shown in Figure 1 and its main components are:

![Flow diagram of the PT-DL Framework](image-url)
• A Smart Contract Coordinator (SCC): It is a program executed on the blockchain that orchestrates the interaction among PT-DL agents to ensure the correct data exchange aimed at training a global model. The SCC operates in rounds. At each round a set of agents is invoked according to a predefined ordering and their responses are aggregated.

• A provably private PT-DL agent training procedure: At each round, the invoked agents use the parameters obtained by the SCC to train a model over their dataset. Each training step is ensured to guarantee \((\epsilon, \delta)\)-differential privacy.

• trustability: Prior being able to submit a model update, a PT-DL agent is required to invoke a verification step that ensures its trustworthiness.

5 Privacy-Preserving and trustable Distributed Learning

Private and trustable Distributed Learning (PT-DL) is a fully distributed learning framework that ensures privacy and trustability while keeping the network bandwidth low. The framework is schematically illustrated in Figure 1 and its main components are:

• A Smart Contract Coordinator (SCC): It is a program executed on the blockchain that orchestrates the interaction among PT-DL agents to ensure the correct data exchange aimed at training a global model. The SCC operates in rounds. At each round a set of agents is invoked according to a predefined ordering (outlined in Section 5.4) and their responses are aggregated.

• A provably private PT-DL agent training procedure: At each round, the invoked agents use the parameters obtained by the SCC to train a model over their dataset. Each training step is ensured to guarantee \((\epsilon, \delta)\)-differential privacy (see Section 6).

• trustability: Prior being able to submit a model update, a PT-DL agent is required to invoke a verification step that ensures its trustworthiness.

Prior to discussing these components in detail, this section introduces the following notation. The PT-DL protocol processes agents according to a given ordering \(\prec\), which, in turn, allows the procedure to exploit an underlying topology to perform parallel updates or to reduce the number of aggregation operations, as described in Section 5.4. An agent \(a\) is a successor of agent \(a'\) if \(a' \prec a\) and \(\text{lev} (a) = \text{lev} (a') + 1\), where \(\text{lev} (a)\) denotes the level of \(a\) in the ordering \(\prec\). The set \(\text{succ} (a) = \{a' \prec a | \forall a' \in [K] \land \text{lev} (a) = \text{lev} (a') + 1\}\) describes the set of agents that succeed agents \(a\) in the ordering \(\prec\). Finally, roots denotes the set of all nodes with no predecessors and leaves the set of all nodes with no successors.

In the following, to simplify exposition, the discussion treats interactions between the SCC and PT-DL agents as direct data exchange. However, as sketched in Figure 1 the SCC coordinates a peer-to-peer (P2P) interaction between agents, that directly exchanges data using a secure protocol, as introduced below.

5.1 PT-DL Smart Contract Coordinator

The PT-DL SCC is described in Algorithm 1. It takes as input the list of registered agents, the agent ordering \(\prec\), and threshold values \(\kappa_1, \kappa_2 > 0\) that control the acceptance of the model updates submitted by an agent. The algorithm starts by initializing the model parameters \(\theta\) (line 1) and, for each round, it coordinates the model updates for the distributed training. At each round, the SCC starts by sending the current model parameters \(\theta\) to the collection of agents that are currently active in roots (lines 3, 5). Here, the word active is used to emphasize that the algorithm is resilient to agents that may drop out from the process, and thus become inactive. Each invoked agent \(a\) responds with a triple, containing the trained model parameters \(\theta_{a'}\), an evaluation score \(\phi_{a} \in [0, 1]\), e.g., a scalar describing the agent model \(\mathcal{M}_{\theta_{a'}}\) accuracy, and a Zero Knowledge Proof value \(Z_{a}\), which is used by the SCC to verify that the information sent by \(a\) is indeed computed by \(a\) (see Supplemental material for additional information). Next, for every active agent \(a\) in \(A\), the SCC invokes a distributed evaluation procedure whose goal is to determine whether the model update was genuine or malignant.

The evaluation involves a sub-sample \(A_S\) of agents \(a'\) that is asked to test model \(\mathcal{M}_{\theta_{a'}}\) over their test data (a subset of \(D_{a'}\)) and to report the associated accuracy metrics \(\phi_{a'}\) (line 8). The collection of these metrics is denoted \(\phi_{A_S}\). A model update is retained genuine if the median accuracy score \(\phi_{A_S}\) does not degrade substantially the current (global) model accuracy \(\phi_A\) (first condition of line 9) and if it is not too far from the score \(\phi_{a}\) reported by agent \(a\) (second condition of line 9). SCC instructs
Algorithm 1: PT-DL Smart Contract Coordinator (SCC)

 Input: $\{a_i\}_{i \in [K]}$: The set of participating agents; 
 $\prec$: The agent ordering; 
 $\kappa_1, \kappa_2 \in [0, 1]$: Accuracy thresholds 
 Output: Global trained model $\theta$

begin
  $\theta \leftarrow 0$
  for round $r = 1, 2, \ldots$ do 
    $A \leftarrow \{(a, \theta) \mid a \in \text{roots}\}$
    while $A \neq \emptyset$ do
      send $\theta$ to all active $a \in A$
      receive $(\theta_a, \phi_a, Z_a)$ from all active $a \in A$
      for all $(a, \theta) \in A$ do
        if $\phi_a - \bar{\phi}_a \leq \kappa_1$ and $|\bar{\phi}_a - \phi_a| \leq \kappa_2$ then
          $A \leftarrow A \cup \{(a', \theta_a) \mid a' \in \text{succ}(a)\}$
        else
          $A \leftarrow A \cup \{(a', \theta) \mid a' \in \text{succ}(a)\}$
        $A \leftarrow A \cup \{(a', \theta) \mid a' \in \text{succ}(a)\}$
      end
      $\theta \leftarrow \text{AGGREGATE}(\theta_a \mid a \in \text{leaves} \land \text{accepted}(\theta_a))$
  end
end

the successors of $a$ to continue the model training starting from parameter $\theta_a^*$, if the model update is accepted (line 10), or from $\theta$ if it is rejected (line 12). Finally, agent $a$ is removed from the set $A$ of the agents to invoke in the current round (line 13). When all agents have been invoked, all valid models are aggregated via averaging (line 14), and the procedure repeats.

While not explicitly described, notice that the set of participating agents can be dynamically updated during the procedure.

5.2 PT-DL Training Agent

The PT-DL agents interact with the SCC using an Ethereum client, which is used to send and receive updates from the blockchain. PT-DL agents use the InterPlanetary File System (IPFS) to store and share their models. IPFS is an efficient, secure, and peer-to-peer protocol. Prior to sending their data to the IPFS, the agents encrypt it using an asymmetric encryption scheme with keys derived from the sender and receivers’ Ethereum accounts. Since IPFS is a public system, encryption secures the (differentially private) model updates to external accesses.

Each PT-DL agent operates as described in Algorithm 2. Given its training data $D_a$, the model learning rate $\eta$, and a mini-batch size $b$, the agent starts registering with the SCC (line 1) and hence it listens for updates. The agent responds to any direct request of its model parameters $\theta_a$ by sharing them with the requesting agent using IPFS (lines 1 and 2). Upon receiving the model parameters $\theta_a$ from some agent $a'$, the agent decrypts them and uses them to initialize its model $\mathcal{M}^a$ (line 6). The training processes computes the gradients over the loss function $\mathcal{L}(\mathcal{M}^a_{\theta_a}(X), Y)$ for each data sample $(X_i, Y_i)$ in a minibatch $B$ of $D_a$. The computation of these gradients is made differentially private by the introduction of Gaussian noise calibrated as shown in Definition 3 (lines 7 and 8). The concept relies on performing a differentially private Stochastic Gradient Descent (DP-SDG) step [1]. In a nutshell, DP-SDG computes the gradients for each data sample in a mini-batch $B$, clips their $L2$-norm, computes the average, and add noise to ensure privacy. The process is illustrated in line 8. Therein, $\nabla^c x = \nabla x / \max(1, \frac{\nabla x}{c})$ denotes the gradient of a given scalar loss $x$ clipped in a $c$-ball for $c > 0$, $\sigma > 0$ is the Gaussian standard deviation value, and $I$ is the identity matrix. A discussion on the privacy analysis is provided in Section 6. Finally, the agent evaluates the trained model over its test data (line 9) and sends the evaluation metrics, its encrypted model parameters $\theta_a$, and a zero knowledge proof $Z_a$ of $\theta_a$ to the SCC (lines 10 and 11).

\cite{1}While the coordination process is orchestrated via the SCC the model requests are peer-to-peer, and thus agents use direct data exchange to execute the PT-DL protocol.
Algorithm 2: Agent a Training Routine

\textbf{Input:} \(D_a\): The agent training data, \\
\(\eta\): The learning rate \\
b: The mini-batch size

1. Register \(a\) with the PT-DL SCC

2. \(\text{repeat :} \)

3. \(\text{if agent } a' \text{ requests } \theta_a \text{ then} \)

4. \(\text{encrypt } \theta_a \text{ and share it with } a' \text{ on the IPFS} \)

5. \(\text{upon receiving } \theta (\text{as } \theta_a) \text{ from agent } a' : \)

6. \(\text{decrypt } \theta_a \text{ and initialize model } M_{\theta_a} \)

7. \(\text{for mini-batch } B \text{ of size } b \text{ in } D_a \text{ do} \)

8. \(\theta_a \leftarrow \theta_a - \eta \frac{1}{b} \sum_{(X_i, Y_i) \in B} \nabla_{\theta_a} \left[ \mathcal{L}(M_{\theta_a}^a(X_i), Y_i) \right] + \xi \)

9. \(\phi_a \leftarrow \text{EVAL}(\theta_a) \text{ over test data} \)

10. \(\text{Generate Zero knowledge proof } Z_a \)

11. \(\text{send encrypted } (\theta_a, \phi_a, Z_a) \text{ on the IPFS} \)

5.3 Trustability and Incentives

PT-DL agents are required to assess other agents’ contribution at each round of the distributed training process. The agents, which agree on an evaluation metric to adopt, are encouraged to submit highly accurate models. As reviewed in Section 5.1, a model update is accepted if the agent evaluation of that model does not deviate too much from the median evaluation score of a subsample \(A_S\) of other agents and if the median score is also not much worse than the evaluation of the current global model. Submitting a low quality model would result in a low median score, while submitting an incorrect quality assessment would result in a high deviation from the median quality analysis. Both cases would result in rejecting the model update. Agents associated to valid (malicious) models can be rewarded (made trustable) through the use of the zero knowledge proof. Storing the zero knowledge proof on-chain allows the progress made by an agent to be verified by other agents without revealing any data. The process can be used to disincentive agents publishing poor model updates. A formal analysis on the trustability guarantee is given in Section 6.

5.4 PT-DL Communication Topologies

In addition to privacy and trustability, a further aspect that challenges the development of systems that solve problem [1] are the frequency of communication and model aggregations. The former is a bottleneck in distributed training due to the large size of the parameters being exchanged. The latter, as shown in the experimental analysis, may be deleterious to the model accuracy. This section proposes several agent communication topologies that can ensure a contained communication cost and model aggregations.
The classical topology used in distributed learning is a star topology (see Figure 2 (PT-DL, T)), where, at each round all agents send their updates to the SCC, which, in turn, aggregates them. This topology allows the maximal level of parallelism, as all agents can perform local updates at the same time. However, the resulting aggregations may decrease the quality of the overall model as well as increase the network bandwidth utilization. Two further topologies adopted in this work are PT-DL, T, based on a tree decomposition, and PT-DL, C, that uses a chain, as illustrated conceptually in Figure 2. These topologies are encoded in the agent ordering ≺ given as input to Algorithm 1. The tree decomposition ensures that the aggregation is to be performed only by the leaf agents, while the chain topology requires no aggregations.

While the number of aggregation operations and the network bandwidth reduces as the diameter of the graph induced by the agent communication topology increases, the algorithm runtime increases, since the number of parallel operations that can be performed decreases (logarithmically for the tree topology and linearly for the chain one) when compared to the start topology. This trade-off is the subject of study in Section 7. To quantify this degree of parallelism the paper introduces the following terminology.

**Definition 2 (ACR).** An Asynchronous Communication Round (ACR) is defined as the total number of parallel data exchanges amongst the agents and the SCC to complete an algorithmic round.

Notice that the chain topology requires K exchanges, the tree topology requires log(K) exchanges, while the star topology requires a single parallel data exchange to complete a round.

### 6 Trustability and Privacy Analysis

To detect malicious updates, PT-DL requires each agent a to report its accuracy metrics φ, and compare it against the median metrics φS assessed by other agents in AS ⊆ [K]. The trustability analysis relies on bounding the distance from the (possibly corrupted) median value φS to the real (not corrupted) mean value of φS, denoted here with µ ∈ R. The analysis assumes that the evaluation returned by the subsample AS of the selected agents follows a Gaussian distribution with bounded variance σ. Let γ < 1/2 and AS = AS ∪ AS be the set of m agents reporting an evaluation for the accuracy score of parameters θa, where AS denotes the benign agents and AS the malicious agents. The analysis assumes that the points in AS are i.i.d. from N(µ, σ2) and |AS| < γm.

**Theorem 1.** Let t ≥ Φ−1(1/2 + β), where Φ(t) = Pr[X ≤ t] is the cdf of the standard Normal distribution. Then the probability that the PT-DL median estimator φAS differs from the real mean value µ is bounded by:

\[
\Pr \left[ |\bar{\phi}_{AS} - \mu | > t\sigma \right] \leq 2 \exp \left( -2m \left( \Phi(t) - \beta/2 \right)^2 \right).
\]

**Proof.** By scaling, and w.l.o.g. assume σ = 1 so that scores in AS are i.i.d. from N(µ, 1) and the RHS of (3) becomes 2 exp \left( -2m \left( \Phi(t) - 1/2 - \beta \right)^2 \right). By assumption, |AS| > (1 - β)m and thus the median of AS is at most the \( \frac{1}{2} + \beta \)-quantile of AS, since AS contains only an \( \beta \)-fraction of all evaluations. It thus suffices to show that the \( \frac{1}{2} - \beta \)-quantile of AS is not too large.

For each φi ∈ AS, let Yi be the \{0, 1\}-valued random variable which is 1 if Xi − µ > t and 0 otherwise. Yi follows an i.i.d. Bernoulli random process and

\[
E[Y_i] = \Phi(-t) = 1 - \Phi(t).
\]

Moreover, the \( \frac{1}{2} - \beta \)-quantile of AS exceeds µ + t if \( \frac{1}{2} \sum_{\phi_i \in AS} Y_i > \frac{1}{2} - \beta \). Using a Chernoff bound, it follows that:

\[
\Pr \left[ \frac{1}{m} \sum_{\phi_i \in AS} Y_i > 1 - \Phi(t) + s \right] \leq \exp(-2ms^2).
\]

Since, s > 0, by substituting s with \( \Phi(t) - \beta/2 \) gives the claim.

The privacy analysis of PT-DL relies on the moment accountant for Sampled Gaussian (SG) mechanism [31], whose privacy is analyzed using Rényi Differential Privacy (RDP) [30].

1
Definition 3. [Sampled Gaussian Mechanism] Let \( f : S \subseteq D \to \mathbb{R}^d \) be a function mapping subsets \( S \) of the input data \( D \) to \( \mathbb{R}^d \). The Sampled Gaussian (SG) mechanism with sampling rate \( 0 < q \leq 1 \) and standard deviation \( \sigma > 0 \) is defined as follows:

\[
SG_{q,\sigma}(D) \triangleq f(\{x : x \in D \text{ is sampled with probability } q\}) + \mathcal{N}(0, \sigma^2 I),
\]

where each element of \( D \) is sampled independently at random without replacement with probability \( q \), and \( \mathcal{N}(0, \sigma^2 I) \) is the spherical d-dimensional Gaussian noise with per-coordinate variance \( \sigma^2 \).

Theorem 2. ((\( \epsilon, \delta \))-RDP) A mechanism \( f : D \to \mathcal{R} \) with domain \( D \) and range \( \mathcal{R} \) is said to have \( \epsilon \)-Rényi differential privacy of order \( \alpha \), or \((\epsilon, \delta)\)-RDP for short, if for any adjacent \( D, D' \in D \) it holds that

\[
\mathcal{D}_\alpha(f(D) \parallel f(D')) \leq \epsilon,
\]

where \( \mathcal{D}_\alpha(P \parallel Q) \triangleq \frac{1}{1-\alpha} \log E_{x \sim Q} \left( \frac{P(x)}{Q(x)} \right)^\alpha \) is the Rényi divergence of order \( \alpha \) between two probability distributions \( P \) and \( Q \).

The privacy analysis of the SG mechanism is described by the following Theorem from Mironov et al. \[31\].

Theorem 3. For a function \( f : D \to \mathbb{R}^d \), such that for all \( D \sim D' \), \( \|f(D) - f(D')\|_2 \leq 1 \), the SG mechanism \( SG_{q,\sigma} \) with sampling rate \( q \) and standard deviation \( \sigma \) satisfies \((\epsilon, \delta)\)-RDP with:

\[
\begin{align*}
\epsilon & \leq \mathcal{D}_\alpha \left[ \mathcal{N}(0, \sigma^2) \parallel (1-q)\mathcal{N}(0, \sigma^2) + q\mathcal{N}(1, \sigma^2) \right] \\
\epsilon & \leq \mathcal{D}_\alpha \left[ (1-q)\mathcal{N}(0, \sigma^2) + q\mathcal{N}(1, \sigma^2) \parallel \mathcal{N}(0, \sigma^2) \right].
\end{align*}
\]

The Rényi divergences appearing in Equations (4a) and (4b) can be computed numerically according to the procedure described in \[31\]. The final privacy loss in the \((\epsilon, \delta)\)-differential privacy model is obtained by observing that a mechanism satisfying \((\epsilon, \delta)\)-Rényi differential privacy also satisfies \((\epsilon + \log \frac{1/\delta}{\alpha+1}, \delta)\)-differential privacy, for any \( 0 < \delta < 1 \).

Theorem 4. Each agent update is differentially private. Additionally, denote \( \epsilon_a, a \in [K] \) the per-round privacy loss of a PT-DL agent \( a \). Regardless of the topology adopted, the per-round privacy loss of the PT-DL framework is: \( \epsilon = \max_{a \in [K]} \epsilon_a \).

Proof. For an agent \( a \in [K] \), the differential privacy of the computations of the model parameters \( \theta_a \) follows from the application of the SG mechanism in line (8) of Algorithm \[2\]. Next, since each agent holds a non-overlapping subsets of datasets \( D \), by parallel composition of DP, the total privacy loss is \( \max_{a \in [K]} \epsilon_a \).

Theorem 5. Let PT-DL be executed for \( N \) asynchronous communication rounds (ACRs). Then, the total privacy budget loss for topologies chain, tree, and star is, respectively: \( \frac{K}{N} \), \( \frac{N}{K} \), and \( N\epsilon \).

The result follows directly from the application of the sequential composition and observing that the PT-DL with chain topology requires \( K \) ACRs to complete 1 round, tree requires \( \log(K) \) ACRs, and star requires a single ACR.

Theorem \[5\] shows that under a fixed communication cost constraint, PT-DL on a chain topology incurs less privacy loss than on a tree topology, and in turn less privacy loss than on a star topology.

7 Experimental Analysis

Datasets, Models, and Metrics This section studies the behavior of the proposed PT-DL architectures on three classification tasks: (1) MNIST and (2) Fashion MNIST comprises of 0 to 9 handwritten digits and articles images, respectively. These datasets have a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. The task is to correctly classify the class associated with an image. (3) COVID-19 Chest X-Ray is the first public COVID-19 CXR image data collection \[10\] \[9\], currently totaling 900 frontal chest X-ray images from over 26 countries. We combine it with Healthy Chest X-Ray images \[23\] to create a dataset containing, in total, 1800 data samples. The task is to correctly classify from an XRAY image whether a person is COVID-19 positive.
The experiments consider a centralized neural network with 6 hidden layers as baseline with a total of 1,199,882 trainable parameters, and denoted SGD. Since this is a centralized model training on the whole dataset \( D \), it is a representative of an upper bound on the task accuracy. The experiments further consider three PT-DL topologies, as described in Section 5.4: a chain topology, denoted PT-DL\(_C\), a binary tree, denoted PT-DL\(_T\), and a star topology, denoted PT-DL\(_S\) and compare them against FedAvg. When not otherwise specified, the experiments use 100 agents, each with 600 training and 100 testing examples. For the COVID-19 Chest X-Ray Dataset, each device has 7 samples of each class to train and 2 to test. All models are executed for 30 runs (a.k.a., epochs) and the result report average and standard deviations of a 5-fold cross-validation. A systematic model tuning for the proposed PT-DL was not investigated.

The section first analyze the non-private versions of the models above an report their accuracy and scalability, their robustness to agents with highly biased datasets, to model inversion attacks, and to agent dropouts. Next, the section compares the differentially private version of the proposed models and of FedAvg to analyze the trade-off between privacy, accuracy, and communication costs.

### 7.1 Accuracy and Scalability

Figure 3: Algorithms accuracy per round on MNIST(left) and FMNIST(middle) and COVID-19 X-Ray(right) for \( K = 100 \) agents.

| Agents | 10    | 100   | 1000  |
|--------|-------|-------|-------|
| PT-DL\(_C\) | 0.102 | 1.020 | 10.20 |
| PT-DL\(_T\) | 0.132 | 1.510 | 15.28 |
| PT-DL\(_S\) | 0.204 | 2.040 | 20.40 |
| FedAvg   | 0.600 | 6.000 | 60.00 |

Figure 4: Time (left) in seconds and network bandwidth (right) in GB, required to complete 1 round on MNIST data.

Figure 4 reports a comparison of the accuracy of all models analyzed on the MNIST (right), FMNIST (center), and COVID-19 Chest X-Ray (left) datasets. Observe that the PT-DL models outperform FedAvg in all settings and datasets analyzed. The increase in accuracy is particularly notable for PT-DL\(_C\), which performs consistently better than all other topologies analyzed. PT-DL\(_T\) obtains models that are typically dominated by PT-DL\(_C\), but superior to PT-DL\(_S\). This result is explained by noticing that PT-DL\(_S\) aggregates the parameters of all \( K \) agents at each round. PT-DL\(_T\), instead, aggregates only about \( K/2 \) model parameters (i.e., those corresponding to the tree leaves). Finally, PT-DL\(_C\) does not perform any aggregation operation. By contrast, FedAvg aggregates the parameters of all \( K \) agents multiple times per each round.

Next, Figure 4a compares the time (in seconds) required by the algorithms to complete a single training round (i.e., it processes the updates of all the agents in the distributed network) at the increasing of the number of agents. The results are illustrated for the MNIST dataset, but they are consistent across all benchmarks. The algorithms performance follows the following order, from faster...
to slower: PT-DL$_S$, FedAvg, PT-DL$_T$, SGD, and PT-DL$_C$. The runtime is directly proportional with the level of parallelization brought by the different distributed topologies adopted. In particular, notice that both PT-DL$_T$ and PT-DL$_S$, obtain an exponential speedup when compared with PT-DL$_C$ and the centralized baseline. The result reveal a trade-off between computational runtime and accuracy of the PT-DL models, where PT-DL$_C$, the slowest protocol, is also the most accurate, while PT-DL$_S$, the fastest protocol, is the least accurate. On the other hand, the PT-DL$_T$ topology represents a good trade-off between accuracy and runtime. These observations demonstrate the practical benefits of the proposed framework.

Finally, Figure 4b compares the network bandwidth consumption among the algorithms analyzed at varying of the number of agents. The network bandwidth is measured in terms of total amount of data the agents send in Gigabytes. Once again, the results are illustrated for the MNIST dataset but are consistent throughout all datasets. The table illustrates a clear trend: The proposed PT-DL framework is able to compute models more efficiently than FedAvg as it requires less network bandwidth. Further, note that the bandwidth used is inversely proportional to the diameter of the communication graph induced by the PT-DL topologies, with PT-DL$_C$ being the most efficient, followed by PT-DL$_T$, and finally PT-DL$_S$.

### 7.2 Robustness to Attacks and Biased Data

Next, the analysis focuses on the resilience of the decentralized algorithms to adversarial attacks, agent dropouts, and to agents training over highly biased data distributions.

In the first experiment, reported in Figure 5 (left), an increasing number of agents perform a model inversion attack [19] prior sending their updates to the SCC or to the centralized server (as in the case of FedAvg). For a fair comparison, the experiments implement the same malicious agent detection scheme in FedAvg as that used by the proposed PT-DL with parameters $\kappa_1 = \kappa_2 = 0.05$, which rejects an update if the accuracy discrepancy between the median evaluation and the reported one differs by more than 5%. The experiments use $A_S = A$. Observe that even with a significant portion (50%) of the model updates are being compromised, the global model trained by the proposed framework sees only a small drop in accuracy.

Next, Figure 5 (right) reports the behavior of the models as an increasing percentage of the agents drops out the training process. The trends are similar to those summarized above, where only a small drop in accuracy is observed, even when half of the agents drop out. Once again, the results illustrate that the proposed PT-DL framework outperforms the distributed baseline, FedAvg. These trends are a clear indication towards the ability of the proposed protocol to mitigate adversarial attacks as well as agents drop out.

Finally, the section analyses the results in a highly biased setting using the MNIST and FMNIST datasets. In the experiments each agent is given a dataset that contains a certain percentage of one particular label, while the rest of the data represents uniformly data with the remaining labels. This is useful to test agents with biased datasets influenced by a majority class. The results are summarized in Figure 6. Firstly, observe that all the PT-DL versions dominate FedAvg in terms of accuracy. Next, observe that the drop in accuracy is almost imperceptible for PT-DL$_C$ and PT-DL$_T$, while PT-DL on the star topology reports the largest drops, at the increasing of the distribution bias. The results show
that the proposed framework performs reliably in real-world conditions where agent data distribution may vary widely.

### 7.3 Privacy/Accuracy trade-off

Finally, the analysis focuses on the accuracy of the distributed models under the differential privacy constraints. It follows similar settings as those described above and, additionally, it focuses on evaluating the models quality on two additional dimensions: varying of the number of agents $K$, and the number of Asynchronous Communication Rounds (ACRs) (See Definition 2). To ensure privacy, this section uses DP-FedAvg [27] (in lieu of FedAvg), under the privacy model adopted in the paper. The privacy settings for all models are: $c = 10$ and $\sigma = 2.0$ (line 8 of Algorithm 2) and the probability of pure DP violation $\delta = 1e-3$. The total privacy loss is computed based on the moment accountant method, which was discussed in Section 6.

The first experiment evaluates all models using (unbiased) MNIST (Figure 7 (top)) using 10 (left), 50 (center), and 100 (right) agents. The results for other datasets follow the same trends and are reported in the supplemental material. The experiment fixes the maximal number of rounds each algorithm makes to 20, and reports, on the x-axis, the number of ACRs performed by the algorithms, which can also be interpreted as the number of parallel communications the agents perform during the evolution of the distributed process. A discussion on the relation of the ACRs and the number of updates illustrated in the figure plots is given in the appendix. The resulting final privacy losses for all algorithms are $\epsilon = 0.5$ (for K=10), 1.1 (for K=50), and 1.6 (for K=100). The privacy loss increases (sub-linearly) with the number of agents as the dataset size $n_a$ each agent is given decreases while the mini-batch size $b$ is fixed.
Firstly, observe that PT-DL produces private models that are significantly more accurate to those produced by DP-FedAvg, under the same privacy constraints. Our analysis indicates that this is due to the different number of aggregation operations performed by the various algorithms. This crucial behavior was also observed in McMahan et al. [27] that noted that for general non-convex objectives, averaging model parameters could produce an arbitrarily bad model. Under a very tight privacy constraint $\epsilon \approx 1$, PT-DL$_C$ and PT-DL$_T$ consistently achieve models with more than 80% and 76% accuracy, respectively, while models produced by PT-DL$_S$ degrade their performance as the number of agents increase. This is due because experiments with small (large) $K$ have agents holding larger (smaller) datasets: $n_a = n/K$ for each agent $a \in [K]$. Since the privacy loss is affected by the data set size, for fixed mini-batch size $b$ and sampling rate $q$ (See Theorem[2]), models with larger $K$ will result in more noisy updates. This is however, greatly mitigated by the chain and tree topologies of PT-DL that perform far less aggregation operations, when compared to PT-DL$_S$ or DP-FedAvg.

The second experiment, illustrated in Figure[7] (bottom), focuses on biased datasets (bias rate of 30%). The figure reports results for MNIST, but the conclusions extend to the other datasets adopted (see appendix). The results follow the same trends as those outlined above. Additionally, notice that the proposed models experience only a negligible drop in accuracy.

These experiments demonstrate the robustness of the proposed models under a non-IID setting, a varying number of agents, and strict privacy constraints.

8 Conclusions

This paper presented a privacy-preserving and trustable distributed learning (PT-DL) framework for multi-agent learning. The proposed framework is fully decentralized and relies on the notion of Differential Privacy to protect the privacy of the agents data and Ethereum smart contracts to ensure trustability. The paper showed the trustable scheme implemented in PT-DL is robust, with high probability, up to the case where half of the agents may collude. The experimental evaluation illustrates the benefits of the proposed model in terms of accuracy and scalability, robustness to agents with highly biased datasets, to model inversion attacks, and to agent dropouts. Additionally, the model is shown to outperform standard differentially private federated learning algorithms in terms of accuracy and communication costs, for fixed privacy constraints. The results show that PT-DL may be a step toward a practical tool for privacy-preserving and trustable multi-agent learning.

References

[1] Abadi and et al. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016.
[2] M. F. Balcan, A. Blum, S. Fine, and Y. Mansour. Distributed learning, communication complexity and privacy. In Conference on Learning Theory, pages 26–1, 2012.
[3] A. Bellet, R. Guerraoui, M. Taziki, and M. Tommasi. Fast and differentially private algorithms for decentralized collaborative machine learning. ArXiv, abs/1705.08435, 2017.
[4] A. Bellet, R. Guerraoui, M. Taziki, and M. Tommasi. Personalized and private peer-to-peer machine learning, 2018.
[5] R. Carli and M. Dotoli. Distributed alternating direction method of multipliers for linearly constrained optimization over a network. IEEE Control Systems Letters, 2020.
[6] H.-P. Cheng, P. Yu, H. Hu, F. Yan, S. Li, H. Li, and Y. Chen. Leasgd: an efficient and privacy-preserving decentralized algorithm for distributed learning, 2018.
[7] H.-P. Cheng, P. Yu, H. Hu, S. Zawad, F. Yan, S. Li, and Y. Chen. Towards decentralized deep learning with differential privacy, 06 2019.
[8] M. E. H. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M. A. Kadir, Z. B. Mahbub, K. R. Islam, M. S. Khan, A. Iqbal, N. A. Emadi, and et al. Can ai help in screening viral and covid-19 pneumonia? IEEE Access, 2020.
[9] J. P. Cohen and et al. Covid-19 image data collection: Prospective predictions are the future. arXiv 2006.11988, 2020. URL https://github.com/ieee8023/covid-chestxray-dataset.
[10] J. P. Cohen, P. Morrison, and L. Dao. Covid-19 image data collection. arXiv 2003.11597, 2020. URL https://github.com/ieee8023/covid-chestxray-dataset.
[11] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

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

[13] M. Fang, X. Cao, J. Jia, and N. Z. Gong. Local model poisoning attacks to byzantine-robust federated learning, 2019.

[14] O. Fercoq, Z. Qu, P. Richtárik, and M. Takáč. Fast distributed coordinate descent for non-strongly convex losses. In *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2014.

[15] F. Fioretto and P. Van Hentenryck. Privacy-preserving federated data sharing. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 638–646, 2019.

[16] F. Fioretto, W. Yeoh, and E. Pontelli. Multi-variable agents decomposition for DCOPs. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 2480–2486, 2016. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12093.

[17] F. Fioretto, E. Pontelli, and W. Yeoh. Distributed constraint optimization problems and applications: A survey. *Journal of Artificial Intelligence Research*, 61:623–698, 2018. doi: http://dx.doi.org/10.1613/jair.5565. URL http://www.jair.org/papers/paper5565.html.

[18] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM Conference on Computer and Communications Security (CCS)*, 2015.

[19] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller. Inverting gradients – how easy is it to break privacy in federated learning?, 2020.

[20] R. C. Geyer, T. Klein, and M. Nabi. Differentially private federated learning: A client level perspective, 2017.

[21] HIPAA. S. a. health insurance portability and accountability act. Public law, 104:191, 1996.

[22] P. Kairouz, S. Oh, and P. Viswanath. The composition theorem for differential privacy. In *International conference on machine learning*, pages 1376–1385. PMLR, 2015.

[23] D. S. Kermany and et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122 – 1131.e9, 2018. ISSN 0092-8674. doi: https://doi.org/10.1016/j.cell.2018.02.010. URL http://www.sciencedirect.com/science/article/pii/S0092867418301545.

[24] H. Kim, J. Park, M. Bennis, and S.-L. Kim. On-device federated learning via blockchain and its latency analysis. *arXiv preprint arXiv:1808.03949*, 2018.

[25] J. Konečný, B. McMahan, and D. Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

[26] C. Ma, V. Smith, M. Jaggi, M. I. Jordan, P. Richtárik, and M. Takáč. Adding vs. averaging in distributed primal-dual optimization. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pages 1973–1982, 2015.

[27] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[28] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[29] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data, 2017.

[30] I. Mironov. Rényi differential privacy. *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, Aug 2017. doi: 10.1109/csfn.2017.11. URL http://dx.doi.org/10.1109/CSF.2017.11.

[31] I. Mironov, K. Talwar, and L. Zhang. Rényi differential privacy of the sampled gaussian mechanism, 2019.
[32] A. Nagar, C. Tran, and F. Fioretto. A privacy-preserving and accountable multi-agent learning framework. In Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pages 1605–1606, 2021.

[33] O. Shamir and N. Srebro. Distributed stochastic optimization and learning. In 2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton), pages 850–857. IEEE, 2014.

[34] 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), pages 3–18. IEEE, 2017.

[35] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo. Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive. IEEE Transactions on Dependable and Secure Computing, 2019.

[36] G. Wood et al. Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper, 151(2014):1–32, 2014.

[37] T. Yang. Trading computation for communication: Distributed stochastic dual coordinate ascent. In Advances in Neural Information Processing Systems, pages 629–637, 2013.

[38] T. Zhang and Q. Zhu. Dynamic differential privacy for admm-based distributed classification learning. IEEE Transactions on Information Forensics and Security, 12(1):172–187, 2017.

[39] X. Zhang, M. M. Khalili, and M. Liu. Improving the privacy and accuracy of admm-based distributed algorithms, 2018.

[40] Y. Zhang and L. Xiao. Communication-efficient distributed optimization of self-concordant empirical loss. In Large-Scale and Distributed Optimization, pages 289–341. Springer, 2018.

[41] Y. Zhang, J. Duchi, M. I. Jordan, and M. J. Wainwright. Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In Advances in Neural Information Processing Systems, pages 2328–2336, 2013.
A Additional Background on Blockchain

EVM

The Ethereum Virtual Machine (EVM) is a distributed Turing complete system that allows us to build Smart Contracts written in Solidity or Vyper, and execute them using the Ethereum Blockchain.

A temporary Ethereum address generated to sign transactions and to communicate with the AP-DL Smart Contract. This ensures that the Smart Contract does not use the same Ethereum address to refer to the device only once.

Zero-Knowledge Proof

A Zero-Knowledge Proof is a method by which one participant in the network (prover) can prove to another participant (verifier) that they know a value x, without revealing the value of x. There are two types of ZKPs — interactive and non-interactive. An interactive proof involves a series of questions from the verifier to the prover to prove their knowledge whereas a non-interactive proof relies on the verifier picking a random challenge for the prover to solve. Multiple interactions between the prover and verifier become unnecessary as the proof exists in a single message sent from the prover to the verifier. We utilize a non-interactive ZK proof also known as ZK-STARK. The Zero Knowledge proof allows the progress made by the agent to be verified without revealing any data. By storing the proof on-chain agents can verify the validity of the progress of any other agent without requiring any interaction. Secondly, as anyone who has the private key to the signing account of the ZKP can prove that this progress was indeed done by them, they can be rewarded or penalized depending on the update they sent. So in an incentivized system, agents would be motivated to publish good updates.

Blockchain Operations

Connecting to the blockchain and sending transactions to it requires access to an Ethereum Node. There are two ways of accessing one. First, running a node locally on a system that directly connects to the network, downloading it’s entire state. Secondly, by using a Remote RPC Endpoint which is essentially a node running on a remote machine that handles running the node and relays our transaction data. The transaction data cannot be manipulated as it is signed by a private key. This allows IoT Devices to data centers to interact with blockchains. Permissionless Blockchains like Ethereum allow anyone to create decentralized applications using Smart Contracts. These smart contracts are written in programming languages like Solidity and Vyper. All this combined creates a decentralized and turing complete atomic computation environment.

B Incentivization

One of abilities of blockchain is to support decentralized finance. This allows us to incentivize the training process rewarding and penalizing agents relative to their contribution towards improving the accuracy of the global model. After an agent completes its training process it creates a ZKP of its progress. It internally uses a Commit Reveal Scheme. The agent will locally generate a random number(the reveal) with a salt and then hash it and send it on-chain (the commit). Finally, we’ll hash their random number (that the miner shouldn’t know about) with the block-hash on the commit block (that the player couldn’t know about). This final hash is a pretty good source of randomness on-chain because it gives the player an assurance that the miner didn’t manipulate it. This process is used to hide the identity of the agent if they choose to redeem their rewards using the reveal to a new address with no transaction history. If the model update from this agent is accepted, then the ZKP is stored on the chain with the users address and can be redeemed by any address as long as the have the reveal to value of the commit. The ZKP also allows another agent to validate the progress of an other agent using the ZKP without requiring to interact with the agent. These measuring help in incentivizing the users to push good updates and make agents accountable to the other agents in a privacy-preserving manner.
C Software and Models Settings

Computing Infrastructure All experiments were performed on a cluster equipped with Intel(R) Xeon(R) Platinum 8260 CPU @ 2.40GHz and 8GB of RAM.

Software and Libraries All models and experiments were written in Python 3.7. All models in the paper were implemented in Pytorch 1.5.0. The Tensorflow Privacy library is utilized for privacy computation.

Architectures The network architecture for MNIST/FMNIST dataset is reported in Table 1. For COVID dataset, the RESNET18 model was utilized due to its superior performance in classifying normal and COVID-19 pneumonia classes. To speed up the training progress, all layers except the last fully connected one were frozen. Hence, the parameter learning takes place only at this last layer.

| Layer (type) | Output Shape | Param # | Tr. Param # |
|--------------|--------------|---------|-------------|
| Conv2d-1     | [1, 32, 26, 26] | 320     | 320         |
| Conv2d-2     | [1, 64, 24, 24] | 18,496  | 18,496      |
| Dropout2d-3  | [1, 64, 12, 12] | 0       | 0           |
| Linear-4     | [1, 128]      | 1,179,776 | 1,179,776 |
| Dropout2d-5  | [1, 128]      | 0       | 0           |
| Linear-6     | [1, 10]       | 1,290   | 1,290       |

Table 1: Model Structure

Code The implementation of all models and the relevant experiments will be released upon publication.

Algorithms’ Setting The parameters settings for all models across datasets are given in Table 2. The settings for private extension of these models are summarized in Table 3.

- Inner epochs: the number of local epochs each agent performs on its own data in a single round.
- ACRs: the total number of asynchronous communication rounds (ACRs, see Definition 2) that each agent perform during the course of model’s training. This parameter depends strongly on model, the number of agents \(K\) and the total number of training sample \(n\) in case of Fed-Avg. Given a fixed number of rounds \(N\), with \(K\) agents, the number of ACRs for chain, tree, star topology respectively is: \(NK\), \(N \log_2 K\) and \(N\).

| Model | Data      | Batch size | # Rounds | # Inner epochs | # ACRs |
|-------|-----------|------------|----------|---------------|--------|
| SGD   | MNIST/FMNIST | 64        | NA       | 30            | NA     |
| PA-DL_C | MNIST/FMNIST | 64        | 30       | 1             | 30 \(K\) |
| PA-DL_T | MNIST/FMNIST | 64        | 30       | 1             | 30 \(\log_2 K\) |
| PA-DL_S | MNIST/FMNIST | 64        | 30       | 1             | 30     |
| FedAvg | MNIST/FMNIST | 64        | 30       | NA            | 30 \(\frac{n}{K\times 16}\) |
| SGD   | COVID     | 8          | NA       | 30            | NA     |
| PA-DL_C | COVID     | 8          | 30       | 1             | 30 \(K\) |
| PA-DL_T | COVID     | 8          | 30       | 1             | 30 \(\log_2 K\) |
| PA-DL_S | COVID     | 8          | 30       | 1             | 30     |
| FedAvg | COVID     | 8          | 30       | NA            | 30 \(\frac{n}{K\times 16}\) |

Table 2: Parameters settings for all non-private models presented in the paper, where \(K\) is the number of agents, \(n\) is the total number of training samples.
Table 3: Parameters settings for private extensions of all models presented in the paper, where $K$ is the number of agents, $n$ is the total number of training samples.

| Model    | Data          | Batch size | # Rounds | # Inner epochs | # ACRs          |
|----------|---------------|------------|----------|----------------|-----------------|
| PA-DL$_C$| MNIST/FMNIST  | 64         | 20       | 1              | $20 \times K$  |
| PA-DL$_T$| MNIST/FMNIST  | 64         | 20       | 1              | $20 \times \log_2 K$ |
| PA-DL$_S$| MNIST/FMNIST  | 64         | 20       | 1              | $20$           |
| DP-FedAvg| MNIST/FMNIST  | 64         | 20       | NA             | $\frac{20n}{K^{16}}$ |
| PA-DL$_C$| COVID        | 16         | 20       | 1              | $20 \times K$  |
| PA-DL$_T$| COVID        | 16         | 20       | 1              | $20 \times \log_2 K$ |
| PA-DL$_S$| COVID        | 16         | 20       | 1              | $20$           |
| DP-FedAvg| COVID        | 16         | 20       | NA             | $\frac{20n}{K^{16}}$ |

D Extended Results

The experiments reported below extend the analysis reported in Section 7.3 of the main paper, that focuses on the impact of the privacy constraints onto the accuracy of the different models. Figure 8 provides a comparison across all models on the unbiased versions of MNIST (left) and FMNIST (right) datasets, while Figure 9 illustrates the performance of all models on the biased counterparts of these datasets. The parameter settings for all models are summarized in Table 3. The resulting final privacy losses for all algorithms are $\epsilon = 0.5$ (for $K=10$), 1.1 (for $K=50$), and 1.6 (for $K=100$). The privacy loss increases (sub-linearly) with the number of agents as the dataset size $n$, each agent is given decreases while the mini-batch size $b$ is fixed.

Notice that the reason why the algorithms stop to different ACR is because they are all ran for 20 rounds, and their total number of ACRs is reported in Table 3.

The results follow trends as those reported in the main paper. In particular, PA-DL produces private models that are significantly more accurate to those produced by DP-FedAvg, under the same privacy constraints.

Similar observation are also made for the COVID dataset, illustrated in Figure 10. Since this is a very small dataset, the experiments use $K = 5$ and $K = 10$ agents, and the data samples are distributed uniformly across the $K$ agents.
Figure 8: Accuracy vs ACRs on unbiased MNIST (left) and unbiased FMNIST (right) datasets for $K = 10$ (top), $K = 50$ (middle), and $K = 100$ (bottom) agents. The final privacy loss of each model for $K = 10$, $K = 50$, and $K = 50$ respectively are 0.5, 1.1 and 1.6.
Figure 9: Accuracy vs ACRs on biased MNIST (left) and biased FMNIST (right) datasets for $K = 10$ (top), $K = 50$ (middle), and $K = 100$ (bottom) agents. The final privacy loss of each model for $K = 10$, $K = 50$, and $K = 50$ respectively are 0.5, 1.1 and 1.6.

Figure 10: Accuracy vs ACRs on COVID-19 dataset for $K = 5$ (left) and $K = 10$ (right). The final privacy loss of each model for $K = 5$ and $K = 10$ respectively are 2.4 and 3.5.